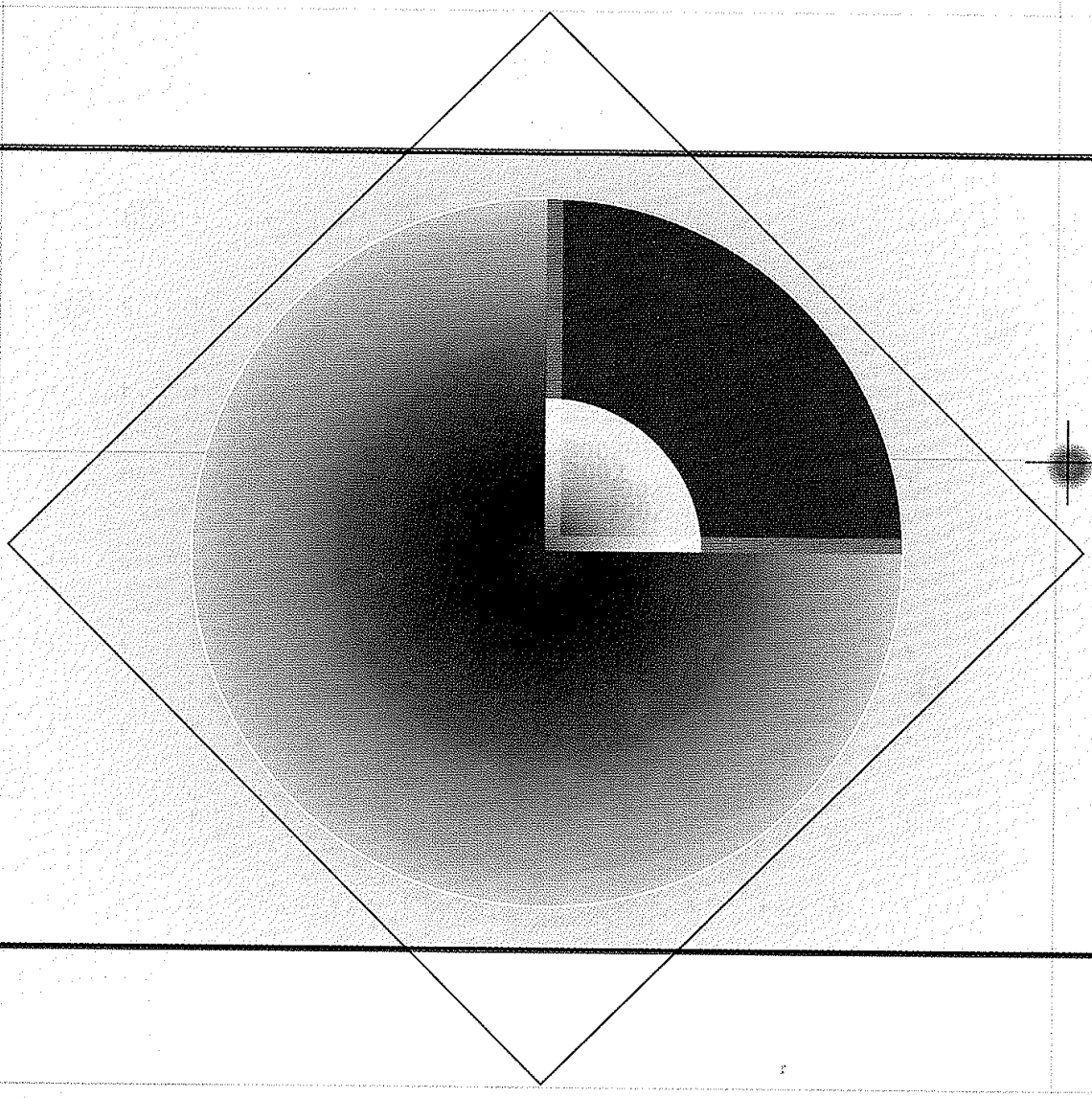
Jiawei Han
Micheline Kamber

ADATBÁNYÁSZAT

Koncepciók és technikák



PANEM

JIAWEI HAN–MICHELINE KAMBER

Adatbányászat

Koncepciók és technikák

ELSEVIER

PANEM

Ez a könyv az Oktatási Minisztérium támogatásával a Felsőoktatási Pályázatok Irodája által lebonyolított Felsőoktatási Tankönyv- és Szakkönyv-támogatási Program keretében jelent meg.

OM
Oktatási
Minisztérium

ISBN 963 545 394 9

A kiadásért felel a Panem Kft. ügyvezetője, Budapest, 2004

Szerkesztette és a magyar előszót írta: Benczúr András

Fordította: Buza Antal (6. fejezet), Fogarassyné Vathy Ágnes (4. fejezet),
Hernáth Szabolcs (3. fejezet), Hernáth Zsolt (1. fejezet), Kósa Balázs (7. fejezet),
Limbek Réka (2. fejezet), Malicskó László Gábor (10. fejezet, Függelék),
Nikovits Tibor (8. fejezet), Szekér István (9. fejezet), Vincellér Zoltán (5. fejezet)

Lektorálta: Kiss Attila
Szerkesztette: Sághi Márta
Borítóterv: Édelkraut Róbert
Tipográfia és tördelés: Székffy Tamás

panem@panem.hu
www.panem.hu

Tartalomjegyzék

Előszó a magyar kiadáshoz

Az adatok világa egyre gazdagabb, változatosabb lesz, és másfél-két évente megkétszereződik a világhálón gyűlő adattömeg. Másként érzékeltetve, a következő két év során annyi adat kerül majd be a számítógépek adatállományába, mint amennyi jelenleg összesen van. Az adatok sokáig hagyományos adatbázisokban vagy szöveges, rendszerezett könyvtári állományokban gyűltek. Többségük jól megtervezett működési célt szolgált, s a feldolgozó erőforrások szűkössége csak korlátozott típusú tömeges feldolgozást tett lehetővé. A működtetési, operatív adatbázisokra épülve ilyenek voltak a vezetői információs rendszerek, míg a szöveges adatbázisokban az információ-visszakereső rendszerek. Azután másfél évtizede megjelentek az adattárházak, az on-line analitikus feldolgozás (OLAP), majd megindult a dokumentumok – HTML, XML – áradata, jöttek a digitális multimédia állományai, érzékelők, megfigyelő kamerák, űrfelvételek, mobil kommunikáció adatai, és legújabban a mozgó alkatrészekkel is rendelkező, kommunikálni képes mikroprocesszorok potenciális jelözöne.

Ez az egyre szabályozatlanabb adatözön a feldolgozás, felhasználás technológiája számára állandóan új kihívásokat jelent. Az egyik nagy kérdéscsoport az adattömegekben rejtőző fontos, hasznos, érdekes ismeretek kinyerése. Találunk-e valamit, ami többet mond az adatokban tükröződő múltról, mint amit eddig tudtunk, tudunk-e tenni valamit azért, hogy a jövőt a hasznos jelenségek bekövetkezésének irányába befolyásoljuk, s elkerüljük a káros jelenségeket, mérsékeljük esélyüket, felkészüljünk bekövetkezésükre. Az erre irányuló tevékenységek, új technológiák gyűjtőneve az *adatbányászat*. Amikor az 1980-as évek végén megjelent, még nem lehetett tudni, hogy ez egy új reklámfogás vagy új, tartósan felfutó technológia lesz-e. Az 1990-es évek folyamatos fejlődése azt mutatta, hogy az utóbbi nyert igazolást, megkezdődött az adatbányászat technológiai érettségének szakasza, komplex, nagy teljesítményű piaci termékek jelentek meg. Ami karakterisztikusan megkülönbözeti az előző adatelemző technológiáktól, az a nagy fokú automatizálás, a sokféle technika integrálása, ugyanakkor a felhasználói interakció hatékony támogatása.

A könyv szerzői arra vállalkoztak, hogy adatbázis-nézőpontból rendszerezett keretbe foglalják az adatbányászat kiterjedten multidiszciplináris és gyorsan fejlődő területét. Erre a rendszerezésre már igen nagy szükség volt, s ez az igény a hazai szakmai és felsőoktatási területen is megjelent. A könyv érdemeit igen jól foglalja össze a könyvsorozat szerkesztőjének, az adatbázis-technológia fejlesztésében elért eredményeiért Turing-díjban részesült Jim Gray előszava.

A könyv tanulmányozása előtt mindenképpen érdemes a szerzők előszavát is elolvasni, ami hasznos eligazítást ad arra nézve, ki mit várhat a könyvtől, hogyan, milyen célokra használható a könyv. Mindezek azt is alátámasztják, miért ennek a könyvnek fordítására vállalkoztunk.

A fordítócsapat az ELTE Informatikai Kar Információs Rendszerek Tanszék oktatóiból és doktoranduszaiból szerveződött.

Bízunk benne, hogy olyan könyvet adunk a magyar olvasók kezébe, ami a szerzők szándékának megfelelően hasznos eligazítást ad az adatbányászat kibontakozó területén, s inspiráló lehet mind a felhasználásban, mind a technika fejlesztésében.

Budapest, 2003. november

Benczúr András
Eötvös Loránd Tudományegyetem
Információs Rendszerek Tanszék

Előszó az angol kiadáshoz

Elárasztanak bennünket az adatok – tudományos adatok, orvosi adatok, demográfiai adatok, pénzügyi adatok és üzleti adatok. Az embereknek nincs ideje megvizsgálni ezeket az adatokat. Az emberi figyelem rendkívül értékes és nem elpazarolható erőforrássá vált. Megoldást kell tehát találnunk arra, hogyan tudjuk automatikusan elemezni, automatikusan osztályozni, automatikusan összesíteni az adatokat, automatikusan megtalálni a bennük lévő tendenciákat, illetve automatikusan kimutatni az anomáliákat. Ez egyike az adatbázisok kutatóközössége által legaktívabban kutatott és legizgalmasabb területeknek. A statisztika, a vizuális megjelenítés, a mesterséges intelligencia és számítógépes tanulás kutatói működnek közre ebben a kutatásban. A terület széles spektruma igen nehézzé teszi, hogy áttekintsük azt a rendkívüli előrehaladást, ami ezen a területen az elmúlt néhány évben történt.

Jaiwei Han és Micheline Kamber igen szép munkát végeztek, amikor rendszerbe szervezték és bemutatták az adatbányászatot ebben a jól olvasható tankönyvben. Kezdik az adatbázisok és adatbányászat fogalmainak gyors bevezetésével, melynek során kiemelt hangsúlyt kap az adatelemzés. Áttekintik a jelenlegi termékkínálatot egy általános keret bemutatásával, amely mindegyiket lefedi. Majd fejezetről fejezetre sorra veszik azokat a fogalmakat és technikákat, amelyek alátámasztják az osztályozást, előrejelzést, társítást és klaszterezést. Ezeket példákkal illusztrálva mutatják be, körbejárva minden egyes feladatosztály legjobb algoritmusait, valamint szabályokat mutatva arra, hogy mikor érdemes az egyes technikákat alkalmazni. A tárgyalásmód stílusát igen jól olvashatónak találtam és jómagam határozottan sokat tanultam a könyv olvasása során. Jiawei Han és Micheline Kamber az adatbányászat kutatásának vezéralakjai. Ez a könyv gyors előmenetelt biztosít a hallgatóknak ebben a tudományban. A terület rendkívül gyorsan fejlődik, ez a könyv mégis lehetőséget ad az alapelvek megtanulásához, illetve annak megértéséhez, hol is tart ma ez a szakterület. Igen informatívnak és motiválónak találtam, és bizonyos vagyok abban, hogy az olvasó is ilyennek találja majd.

Jim Gray
Microsoft Research

A szerzők előszava

Az utóbbi néhány évtizedben mind az adatok előállításának, mind gyűjtésének lehetőségei igen gyorsan növekedtek. Az ebben szerepet játszó tényezők között említendő a vonalkódok elterjedése szinte valamennyi kereskedelmi termékre, az üzleti, kutatási, közigazgatási tranzakciók nagy részének számítógépesítése, az adatgyűjtő eszközök terén bekövetkezett fejlődés, ami a szövegek és képek számítógépes beolvasó berendezéseitől a műholdas távérzékelőkig terjed. Ezenfelül a World Wide Web globális információs rendszerként való használatának népszerűsége további rendkívüli adat- és információáradatot zúdított ránk. Az adatok robbanásszerű növekedése sürgős igényt váltott ki olyan technológiák és automatizált eszközök iránt, amelyek intelligens segítséget nyújtanak számunkra abban, hogy ezt a rendkívüli adattömeget hasznos információvá és tudássá alakítsuk.

Ez a könyv az adatbázisrendszerek és új adatbázis-alkalmazások ígéretes és virágzó frontvonalának, az *adatbányászatnak* alapelveit és technológiáit tárja fel. Az adatbányászat, vagy ahogy népszerűen *adatbázisokban végzett tudásfeltárásnak* (KDD, knowledge discovery in databases) nevezik, automatizált vagy igen kényelmes kinyerése olyan tudást hordozó alakzatoknak, amelyeket a nagy adatbázisok, adattárházak vagy egyéb tömeges információ-gyűjtemények implicit módon tartalmaznak.

Az adatbányászat multidiszciplináris tudomány, olyan területekre támaszkodik, mint az adatbázis-technológia, a mesterséges intelligencia, a számítógépes tanulás, a neurális hálók, a statisztika, az alakfelismerés, a tudás alapú rendszerek, a tudásmegerősítés, az információ-visszakeresés, a nagy teljesítményű számítások és a vizuális adatmegjelenítés. A könyv anyagát az *adatbázis nézőpontjából* mutatjuk be, vagyis olyan kérdésekre koncentrálunk, amelyek a *nagy adatbázisokban* rejtőző alakzatok feltárására szolgáló technológiák megvalósíthatóságával, hatékonyságával, skálázhatóságával vannak összefüggésben. Következésképpen ebben a könyvben nem szándékoztunk bevezetni az olvasót az adatbázisrendszerek, a számítógépes tanulás, a statisztika világába vagy más határterületre, azonban megadjuk azt a szükséges hátteret, amivel lehetővé tesszük, hogy megértse szerepüket az adatbányászatban. A könyv elsősorban teljes körű bevezető az adatbányászatba, amelynek bemutatása az adatbázisok kérdéseit helyezi középpontba. Egyaránt hasznos lehet informatikus hallgatók, alkalmazásfejlesztők, gazdasági szakemberek, valamint a kutatók számára, akik a fentebb említett diszciplínák bármelyikén tevékenykednek.

Az adatbányászat az 1980-as évek végén jelent meg, jelentős felfutást ért el az 1990-es évek során, és várhatóan tovább virágzik az új évezredben is. Ez a könyv ennek a terület-

nek átfogó képét mutatja be az adatbázis-kutató szemszögéből, amelynek során bevezet érdekes adatbányászati technikákba és rendszerekbe, tárgyalja az alkalmazásokat és kutatási irányokat. A könyv megírásához fontos motivációt jelentett az az igény, hogy építsünk fel egy rendszerezett keretet az adatbányászat tanulmányozásához. Ez ugyanakkor nagy kihívást jelentő feladat ennek az igen gyorsan fejlődő területnek erősen multidiszciplináris jellege miatt. Azt reméljük, hogy ez a könyv bátorítást ad a különböző háttérrel és tapasztalatokkal rendelkezők számára az adatbányászatra vonatkozó nézeteik kicserélésére úgy, hogy ezzel hozzájáruljanak ennek az izgalmas és dinamikus területnek a további fejlődéséhez és formálódásához.

A tanárokhoz

A könyvet úgy terveztük, hogy átfogó, ugyanakkor mégis mély ismereteket adjon az adatbányászat területéről. A tanárok számára hasznos az adatbányászati tantárgy oktatásában haladó BSc (az amerikai rendszerben undergraduate) vagy MSc (graduate) szinten. Ezen túlmenően egyes fejezetek beépíthetők válogatott témaként az adatbázisrendszerek vagy a mesterséges intelligencia tárgyakba. Arra törekedtünk, hogy mindegyik fejezet – amennyire csak lehetséges – egységes egész legyen, s így nem szükséges a fejezeteket egymás után olvasni. Az alapszintű tantárgy számára törzsanyagként használhatók fel az 1–8. fejezetek, illetve haladóbb témakörök választhatók még a 9. és 10. fejezetből. Haladó (MSc, graduate) kurzus számára akár a teljes könyv is választható egyetlen szemeszterben.

Minden egyes fejezetet feladatok zárnak le, amelyek alkalmasak házi feladat kijelölésére. A feladatok vagy rövid kérdések a lefedett anyagban való jártasságot tesztelik, a hosszabb kérdések pedig elemző gondolkodást igényelnek.

A diákokhoz

Azt reméljük, hogy ez a tankönyv felkelti a tanulók érdeklődését az adatbányászat fiatal, de fejlődő területe iránt. Arra törekedtünk, hogy az anyagot világos módon mutassuk be, gondos magyarázattal ellátva az érintett témákat. Minden fejezetet összefoglaló zár, amely kiemeli a kulcspontokat. A könyvet élvezhetővé teszi a sok ábra és illusztráció.

Jóllehet a könyvet tankönyvnek terveztük, olyan felépítésre törekedtünk, hogy az olvasó számára hasznos legyen referencia- vagy kézikönyvként is, amennyiben a későbbiekben úgy döntene, hogy az adatbányászat terén folytatja pályafutását.

Milyen előismertek szükségesek a könyv olvasásához?

- Szükséges némi ismeret az adatbázisrendszerekhez kapcsolódó fogalmakról és terminológiákról. Mindenesetre arra törekedtünk, hogy az alapvető adatbázis-technológiához elegendő hátteret szolgáltassunk, s ezzel segítsük, hogy ha az olvasó emlékezete kissé megkopott is, problémamentesen követni tudja a könyv tárgyalásmenetét. Szükség van az adatbázis-lekérdezések valamilyen szintű ismeretére, de nem követelmény egyetlen konkrét lekérdező nyelv ismerete sem.

- Szükség van programozási tapasztalatra. Nevezetesen, olvasni kell tudni a pszeudokódokat és érteni kell az egyszerű adatszerkezeteket, például a többdimenziós tömböket.
- Segítséget jelent a statisztikában, számítógépes tanulásban vagy az alakfelismerésben való valamilyen bevezető szintű ismeret. Hozzászoktatjuk azonban az olvasót ezeknek a területeknek minden olyan fogalmához, amelyek az adatbázis nézőpontjából tekintve fontosak az adatbányászathoz.

A szakemberekhez

A könyv írásakor az volt a célunk, hogy az adatbányászat területén felbukkanó témák széles tartományát fedje le. Ennek eredményeképpen kiváló kézikönyve ennek a tárgykörnek. Minthogy a fejezeteket úgy terveztük, hogy önmagukban is megálljanak, amennyire ez lehetséges, az olvasó koncentrálhat a számára legérdekesebb témákra. A könyv nagy része az Önhöz hasonló alkalmazásprogramozó vagy információszolgáltatási menedzser számára alkalmas arra, hogy önállóan tanulhassa az adatbányászat kulcsfogalmait.

A bemutatott technikák és algoritmusok mind gyakorlati hasznossággal bírnak. A kis „játék" adatbázisokon jól működő algoritmusok helyett a nagy, valós adatbázisokban rejtőző minták feltárását végző algoritmusok leírása mellett döntöttünk. A 10. fejezetben röviden bemutatunk kereskedelmi forgalomban levő adatbányászati rendszereket, valamint ígéretes kutatási prototípusokat is. Minden algoritmust pszeudokódban illusztrálunk. A pszeudokód hasonlít a C programozási nyelvhez, azonban úgy tervezték, hogy olyan programozók is könnyen megértsék, akik nem jártasak a C vagy C++ nyelvekben. Aki bármelyik algoritmust implementálni szeretné, azt fogja tapasztalni, hogy igen egyszerű módon átírható a választott programozási nyelvre.

A könyv felépítése

A könyv a következőképpen épül fel:
Az 1. fejezet bevezet az adatbányászat multidiszciplináris területébe. Tárgyalja az adatbázis-technológia fejlődési útját, amely az adatbányászat szükségessé válásához vezetett, és kiemeli az adatbányászat alkalmazási lehetőségeinek fontosságát. Bemutatásra kerül az adatbányászati rendszer alapvető felépítése, valamint röviden tárgyaljuk az adatbázis-rendszerek és adattárházak fogalomkörét. Az adatbányászati feladatok osztályozását a kinyerésre szánt tudás különböző típusaira építve mutatjuk be. Áttekintjük az adatbányászati rendszerek osztályozását és a terület előtt álló fő kihívásokat.

A 2. fejezet az adattárházak és az on-line analitikus feldolgozás (OLAP, On-Line Analytical Processing) világába kalauzol. A tárgyalt témák között szerepelnek az adattárházak, multidimenziós adatbázisok, adatkockák létrehozása, az on-line analitikus feldolgozás megvalósítása, valamint az adattárház-kezelés és az adatbányászat viszonya.

A 3. fejezet írja le azokat a technikákat, amelyekkel az adatelőkészítés végezhető az adatbányászat megkezdése előtt. Tárgyaljuk az adattisztítás, az adatintegrálás és -transzformálás, valamint az adatredukálás eljárásait, beleértve a fogalmi hálók használatát és

a statikus és dinamikus diszkretizálást. Szó van a fogalmi hálók automatikus generálásáról is.

A 4. fejezet vezeti be az adatbányászat építőelemeit, formálisabb szóhasználattal primitívjeit, amelyek az adatbányászati feladat specifikálását definiálják. Leírásra kerül egy adatbányászati lekérdező nyelv (DMQL, DataMining Query Language) és példák szerepelnek adatbányászati lekérdezésekre. Egyéb nyelveket is bemutatunk, és leírjuk a grafikus felhasználói felületek és az adatbányászati architektúrák konstruálását is.

Az 5. fejezet ismerteti a fogalomleírás technikáit, ezen belül a karakterizációt és diszkriminációt. Egy attribútumorientált általánosítási technikát vezet be, egyúttal megmutatja annak különböző megvalósításait, beleértve az általánosított relációs technikát és a sokdimenziós adatkocka technikáját. A tudás megmutatásának és vizuális megjelenítésének különféle módjait adja meg. Tárgyalja a fontossági elemzést. Bemutatja az osztályoknak több absztrakciós szinten való összehasonlítási módszerét, valamint a karakterizáló szabályok érdekességi mértékére alapuló kinyerésének módszerét. Ezenfelül tárgyalja még a leíró bányászat statisztikai mértékét is.

A 6. fejezet mutatja be a tranzakciós adatbázisokból, relációs adatbázisokból és adattárházakból történő társítási szabályok bányászatának módszerét. Tartalmazza a társítási szabályok osztályozását, ismerteti az Apriori algoritmus alapmegoldását és variációit, a többszintű társítási szabályok, a többdimenziós társítási szabályok, a mennyiségi társítási szabályok és a korrelációs társítási szabályok bányászatának technikáit. Bevezet egy új technikát, amelyet a gyakori minták növelésének neveznek, és a jelöltek halmazának generálása nélkül bányássza a gyakori mintákat. Leír még stratégiákat az érdekes szabályok megszorítás alapú bányászatához és az érdekességi mértéknek a szabálykeresés fokuszálására való felhasználásához.

A 7. fejezet írja le az osztályozás és előrejelzés módszereit. Magába foglalva a döntési fa indukciót, a Bayes-osztályozást, a hatás-visszaterjedés neurális hálós technikáját, a *k*-legközelebbi szomszéd alapján történő osztályozást, az eset alapú következtetést, a genetikus algoritmusokat, a nyers halmazok elméletét és a fuzzy halmazos megközelítést. Bemutatja a társítási szabályok bányászatából nyert fogalmakra épülő osztályozást. Bevezeti a regressziós módszereket és tárgyalja az osztályozás pontosságára vonatkozó kérdéseket.

A 8. fejezet írja le a klaszterezés módszereit. Az adatok klaszterezésével kapcsolatos fogalmak tisztázása után több alapvető klaszterezési megközelítést mutat be, közöttük a partíció alapú, a hierarchikus és a modell alapú klaszterezést. Ismertet folytonos adatok, diszkrét adatok, többdimenziós adatkockák adatainak klaszterezésére szolgáló módszereket. A klaszterező algoritmusok skálázhatóságát részletesen tárgyalja.

A 9. fejezet témája a korszerű adatbázisokban történő adatbányászat módszerei. Ez tartalmazza az objektumorientált adatbázisokban, a téradatbázisokban, a multimédia-adatbázisokban, az idősor-adatbázisokban, a szöveges adatbázisokban és a World Wide Weben történő adatbányászást.

Végül a 10. fejezetben összefoglaljuk a könyvben bemutatott fogalmakat és elemezzük az adatbányászat alkalmazásait és néhány kihívást jelentő kutatási kérdést.

A szövegben dőlt betűt használunk a definiált fogalom hangsúlyozására, míg a vastag betűket a fő fogalmak kiemelésére használjuk.

Hibák

Előfordulhat, hogy a könyv elírásokat, hibákat, kihagyásokat tartalmaz. Aki bármilyen hibát észlel, további feladatokra tenne javaslatot vagy építő kritikája lenne, számíthat rá, hogy örömmel fogadjuk jelentkezését. Köszönettel vesszük és értékeljük javaslataikat. Megjegyzéseiket a következő címre küldhetik:

Data Mining: Concepts and Techniques
Intelligent Database Systems Research Laboratory
School of Computing Science
Simon Fraser University
Burnbay, British Columbia
Canada V5A 1S6
Fax: (604) 291-3045

Alternatívaként használhatják az elektronikus levelezést is az esetleges újabb hibák jelzésére, a már feltárt hibák lekérdezésére vagy konstruktív javaslat tételére. Útmutatás kéréséhez küldjenek e-mailt a *dmbook@cs.sfu.ca* címre „Subject: help" megjelöléssel az üzenet fejlécében. Sajnáljuk, de nem tudunk minden levélre személyesen válaszolni. A könyv javításai és egyéb, a könyvvel kapcsolatos naprakész információk megtalálhatók a *www.cs.sfu.cs/~han/DM_Book* webcímen.

Köszönetnyilvánítás

Legmélyebb köszönetünket szeretnénk kifejezni mindazoknak, akik jelenleg velünk dolgoznak az adatbányászathoz kapcsolódó kutatásban és/vagy a DBMiner projektben, vagy különféle támogatásban részesítettek bennünket az adatbányászatban. Ezek sorába tartoznak: Rakesh Agrawal, Stella Atkins, Yvan Bedard, Binary Bhattacharya, Dora (Yandong) Cai, Nick Cercone, Surajit Chaudhuri, Sonny H. S. Chee, Jianping Chen, Ming-Syan Chen, Qiming Chen, Qing Chen, Shan Cheng, David Cheung, Shi Cong, Son Dao, Umeshwar Dayal, James Delgrande, Guozhu Dong, Carole Edwards, Max Egenhofer, Martin Ester, Usama Fayyad, Ling Feng, Ada Fu, Yongjian Fu, Daphne Gelbart, Randy Goebel, Jim Gray, Robert Grossman, Wan Gong, Yike Guo, Eli Hagen, Howard Hamilton, Jing He, Larry Henschen, Jean Hou, Mei-Chun Hsu, Kan Hu, Haiming Huang, Yue Huang, Julia Itskevitch, Wen Jin, Tiko Kameda, Hiroyouki Kawano, Rizwan Kheraj, Eddie Kim, Won Kim, Krzisztof Koperski, Hans-Peter Kriegel, Vipin Kumar, Laks V. S. Lakshmanan, Joyce Man Lam, James Lau, Deyi Li, George (Wenmin) Li, Jin Li, Ze-Nian Li, Nancy Liao, Gang Liu, Junqiang Liu, Ling Liu, Alan (Yijun) Lu, Hongjun Lu, Tong Lu, Wei Lu, Xuebin Lu, Wo-Shun Luk, Heikki Mannila, Runying Mao, Abhay Mehta, Gabor Melli, Alberto Mendelzon, Tim Merrett, Harvey Miller, Drew Miners, Behzad Mortazavi-Asl, Richard Muntz, Raymond T. Ng, Vicent Ng, Shojiro Nishio, Beng-Chin Ooi, Tamer Ozsu, Jian Pei, Gregory Piatetsky-Shapiro, Helen Pinto, Fred Popowich, Amynmohamed Rajan, Peter Scheuermann, Shashi Shekar, Wei-Min Shen,

Avi Silberschatz, Evangelos Simoudis, Nebojsa Stefanovic, Yin Jenny Tam, Simon Tang, Zhaohui Tang, Dick Tsur, Anthony K. H. Tung, Ke Wang, Wei Wang, Zhaoxia Wang, Tony Wind, Lara Winstone, Ju Wu, Betty (Bin) Xia, Cindy M. Xin, Xiaowei Xu, Quiang Yang, Yiwen Yin, Clement Yu, Jeffrey Yu, Philip S. Yu, Osmar R. Zaine, Carlo Zaniolo, Shuhua Zhang, Zhong Zhang, Yvonne Zheng, Xiaofang Zhou és Hua Zhu. Hálával tartozunk továbbá Jean Hounak, Helen Pintonak, Lara Winstone-nak, és Hua Zounak a könyv néhány eredeti ábrájának elkészítésében nyújtott segítségéért, valamint Eugene Belchevnek valamennyi fejezet igen alapos korrektúrájának elvégzéséért.

Köszönetet szeretnénk mondani még a Morgan Kaufmann felelős szerkesztőjének, Diana Cerrának lelkesedéséért, türelméért és támogatásáért, amit a könyv megírása során tanúsított, szintúgy technikai szerkesztőnknek, Howard Seversonnak és munkatársainak a nyomdai előállítás gondossága terén nyújtott erőfeszítésükért. Lekötelezettek vagyunk lektoraink felé értékes visszajelzéseikért. Végül, köszönjük családjainknak a projekt folyamán kapott szíves támogatásukat.

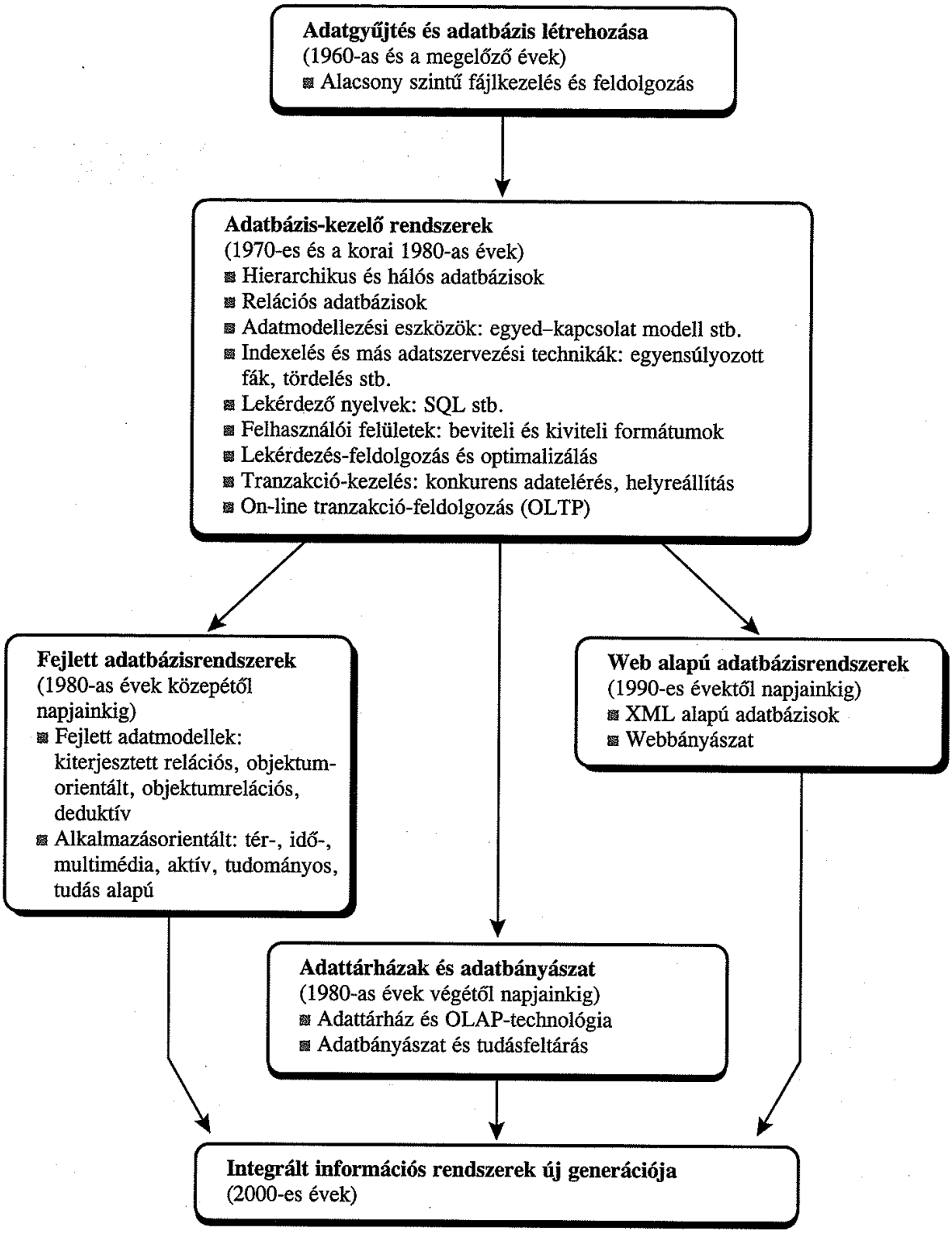
1. FEJEZET
Bevezetés

Ez a könyv bevezet az *adatbányászatként* és *tudásfeltárásként* ismert és emlegetett témakörbe. A szerzők az adatbázisok jövőképe szempontjából a hangsúlyt azokra az adatbányászati fogalmakra és technikákra helyezik, amelyekkel a *terjedelmes adathalmazokban* elrejtett, számunkra érdekes adatmintákra bukkanhatunk. A tárgyalt eljárások kifejezetten a *skálázható* és *hatékony* adatbányászati eszközök kifejlesztésére irányulnak. Ez a fejezet rávilágít arra, hogy az adatbányászat hogyan vált az adatbázis-technológiák természetes fejlődésének szerves részévé, miért fontos, és valójában mi is az. Megismerteti az olvasót az adatbányászati rendszerek általános szerkezeti felépítésével, áttekintést ad a bányászható adatokról, az adatmintákról és arról, hogyan választható ki a hasznos ismeretet rejtő adatminta. Az adatbányászati rendszerek osztályozásának tanulmányozásán túl betekintést enged a jövő adatbányászati eszközeinek kifejlesztésére irányuló kutatásokba.

1.1. | Az adatbányászat kialakulása és fontossága

A szükség a fejlődés szülőanyja

A fő ok, amelynek köszönhetően az adatbányászat az utóbbi időben az információipar területén a figyelem középpontjába került, a széles körben megjelenő hatalmas méretű adatmennyiség, és a növekvő igény arra, hogy abból mihamarabb használható információ és tudás váljék. Az így megszerzett információ és tudás a gazdaság legkülönbözőbb területein alkalmazva, az üzleti irányítástól kezdve a termékszabályozáson keresztül a piackutatásig arra használható, hogy segítse a tervezést és a tudományos kutatást. Az adatbányászatot az információs technológia természetes fejlődése hívta életre. Az adatbázisok világában az *adatgyűjtést* és az *adatbázisok létrehozását*, az *adatfeldolgozást* (ide értve az adatok tárolását, elérését és a tranzakció-kezelést), valamint az *adatok elemzését* és *megértését* (ezek vezettek az adattárházak és adatbányászat megjelenéséhez) támogató, illetve megvalósító fejlődési folyamatot követhetjük nyomon az 1.1. ábrán. A korai adatgyűjtési és adatbázis-létrehozási eljárások kifejlesztése például előfeltétele volt a hatékony adattárolást, adatelérést, lekérdezés-feldolgozást és tranzakció-kezelést megvalósító mechanizmusok megjelenésének. A lekérdezés-feldolgozás és a tranzakció-kezelés általános gyakorlattá válása után a következő magától értetődő cél az adatok elemzése és megértése volt.



1.1. ábra | Az adatbázis-technológia fejlődése

Az 1960-as évektől kezdve az adatbázis- és információs technológia lépésről lépésre jutott el az alacsony szintű egyszerű műveletekre épülő fájlkezeléstől a kifinomult és hatékony adatbázisrendszerekhez. Az 1970-es évektől kezdve az adatbázisrendszerek területén folyó kutatás-fejlesztés eredményeképpen a korai hierarchikus és hálós adatbázisrendszereket relációs adatbázisrendszerek (az adatok relációs tábla struktúrákban táro-

lódnak, lásd 1.3.1. alfejezet), különböző adatmodellezési eszközök, adatszervezési technikák és indexelés váltották föl. A felhasználók a lekérdező nyelvek, a felhasználói felületek, a lekérdezések optimalizált feldolgozása és a tranzakció-kezelés segítségével kényelmesen és rugalmasan férhettek hozzá az adatokhoz. Az **on-line tranzakció-feldolgozás (On-line Transaction Processing, OLTP)** hatékony eljárásai, amelyek a lekérdezéseket csak olvasó (read-only) tranzakcióknak tekintik, alapvetően hozzájárultak a relációs technológia fejlődéséhez, és a nagy mennyiségű adat hatékony tárolása, elérése és feldolgozása fő eszközeként való széles elterjedéséhez.

Az 1980-as évek közepétől kezdve az adatbázis-technológiát a relációs technológia széles körű adaptálása, valamint új, hatékony adatbázisrendszerek létrehozására irányuló kutatás-fejlesztési tevékenységek jellemezték. Ezekben olyan fejlett adatmodelleket használtak, mint a kiterjesztett relációs, objektumorientált, objektumrelációs és deduktív modellek. Megjelentek és virágzásukat élték az alkalmazásorientált adatbázisrendszerek, például a tér- (térbeli), idő- (időbeli), multimédia, aktív és tudományos adatbázisok, tudásbázisok és hivatali információs bázisok. Előtérbe került az elosztott rendszerek és az adatokon történő osztozás kérdéseinek tanulmányozása. Alapvető szerephez jutottak az információiparban a heterogén adatbázisrendszerek és az internet alapú, világot átfogó információs rendszerek, például a World Wide Web (www). Az elmúlt három évtizedben a hardvertechnológia fantasztikus méretű és szakadatlan fejlődése a hatékony számítógépek, az adatgyűjtést segítő eszközök és a tároló médiumok elérhető árú kínálatához vezetett. Az említett technológia maga is nagy mértékben hozzájárult az adatbázis- és információipar fejlődéséhez, biztosítva a nagy számban jelen levő adatbázisok és információtárak számára a tranzakció-kezelést, az információk elérését és az elérhető adatok elemzését.

Az adatok ettől kezdve több különböző típusú adatbázisban tárolhatók. Ilyen adatbázis-architektúra az utóbbi időben megjelenő **adattárház** (1.3.2. alfejezet), amely több heterogén adatforrást fog össze egy egységes sémában, egyetlen egységes felületen a vezetői döntések egyszerűsítése érdekében. Az adattárház-technológia magában foglalja az adattisztítást, az adatintegrációt és az **on-line elemző feldolgozást (On-line Analytical Processing, OLAP)**, azaz elemzési technikákat összesítés, csoportműveleti, konszolidációs műveleti támogatással és az információk különböző szemszögből való értékelhetőségének biztosításával. Jóllehet az OLAP-eszközök támogatják a többdimenziós (több szempont szerinti) elemzést és döntéshozatalt, a mélységi elemzésekhez további adatelemzési eszközök is szükségesek, például az adatok osztályozása, klaszterekbe foglalása és az adatváltozások idő függvényében történő jellemzése.

Az adatok bősége és a hatékony adatelemzési eszközök hiánya *adatokban gazdag, de információban szegény* helyzetet eredményezett. A gyorsan növekvő és ijesztően nagy méreteket öltő, számtalan hatalmas adatbázisban tárolt adatmennyiség hatékony segédeszközök használata nélkül messze meghaladja az emberi elme felfogó- és befogadóképességét (1.2. ábra). Ennek eredményeképpen a nagy méretű adatbázisokban összegyűjtött adatok ritkán látogatott „adatsíremlékekké", „adatarchívumokká" válhatnak. Ebben a helyzetben a fontos döntések sokkal inkább a döntéshozó intuícióin nyugszanak, mintsem az információgazdag adatokon. Ennek oka az az egyszerű tény, hogy a döntéshozók nem rendelkeznek olyan eszközökkel, amelyek a hatalmas adatmennyiségbe beágyazott értékelhető tudás kiválasztását elősegítenék. A jelenleg is használt szakértői rendszer tipikusan a felhasználóra, a terület szakértőjére bízza a tudás tudásbázisba történő *„kézi*

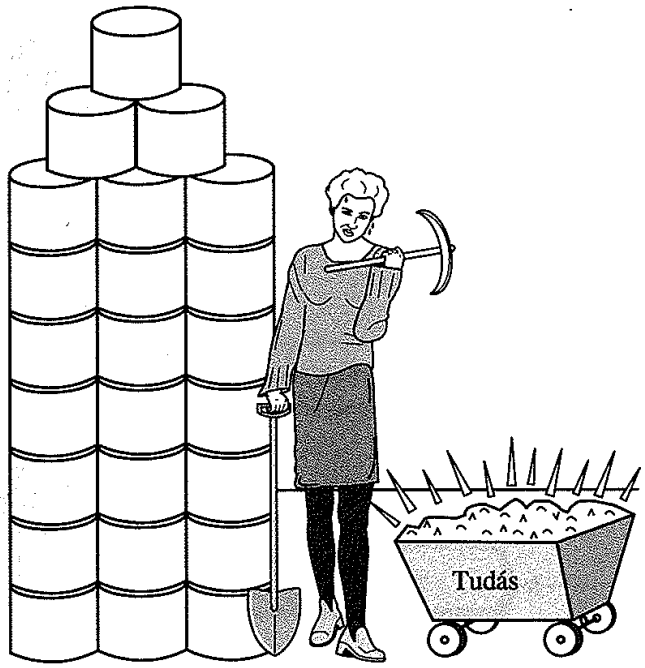
1.2. ábra | Adatokban gazdagok, de információban szegények vagyunk

bevitelét". Sajnos ez az eljárás hibás és előítéletekre épített, félrevezető tudáshoz vezet, és emellett rendkívül időigényes és költséges. Az *adatbányászati eszközök* adatelemzések alkalmazásával képesek fontos adatmintákat feltárni, nagymértékben támogatva ezzel az üzleti stratégiát, a tudásbázisokat, a tudományos és egészségügyi kutatásokat. Az adatok és információ közötti egyre szélesedő űr megköveteli az *adatbányászati eszközök* szisztematikus fejlesztését, hogy az „adatsíremlékeket" a tudás „aranyrögeivé" változtassa.

1.2. | Mi az adatbányászat?

Az adatbányászat *a tudás nagy mennyiségű adatból történő kiválasztása, kibányászása.* Maga az adatbányászat szó félrevezető, hiszen amikor aranyat sziklákból vagy homokból bányászunk, *aranybányászatról,* és nem kő- vagy homokbányászatról beszélünk. A fentiek alapján a helyes kifejezés az „adatokban való tudásbányászat" lenne, azonban ez nehézkes és hosszú. A rövidebb „tudásbányászat" már jobban cseng, és bár a *bányászat* szó elevenen érzékelteti azt a folyamatot, amelynek során kevés apró értékes anyag kerül a felszínre nagy mennyiségű nyersanyag mozgatásával és átvizsgálásával, nem utal arra és nem hangsúlyozza azt, hogy a tudást nagy mennyiségű adatból kell feltárni (1.3. ábra). Valószínűleg ez az oka a „félrevezető" adatbányászat terminológia elterjedésének. A témakörhöz kapcsolódóan meg kell emlékeznünk még a **tudásbányászat adatbázisokból**, **tudáskinyerés**, **adatelemzés/mintaelemzés**, **adatrégészet** és **adatkotrás** kifejezésekről is, amelyek hasonló jelentést hordoznak.

Nagyon sokan az adatbányászatot az **adatbázisokban végzett tudásfeltárás (Knowledge Discovery in Databases, KDD)** kifejezés szinonimájaként használják, míg mások a KDD



1.3. ábra | Adatbányászat – tudás (érdekes minták) kutatása az adatokban

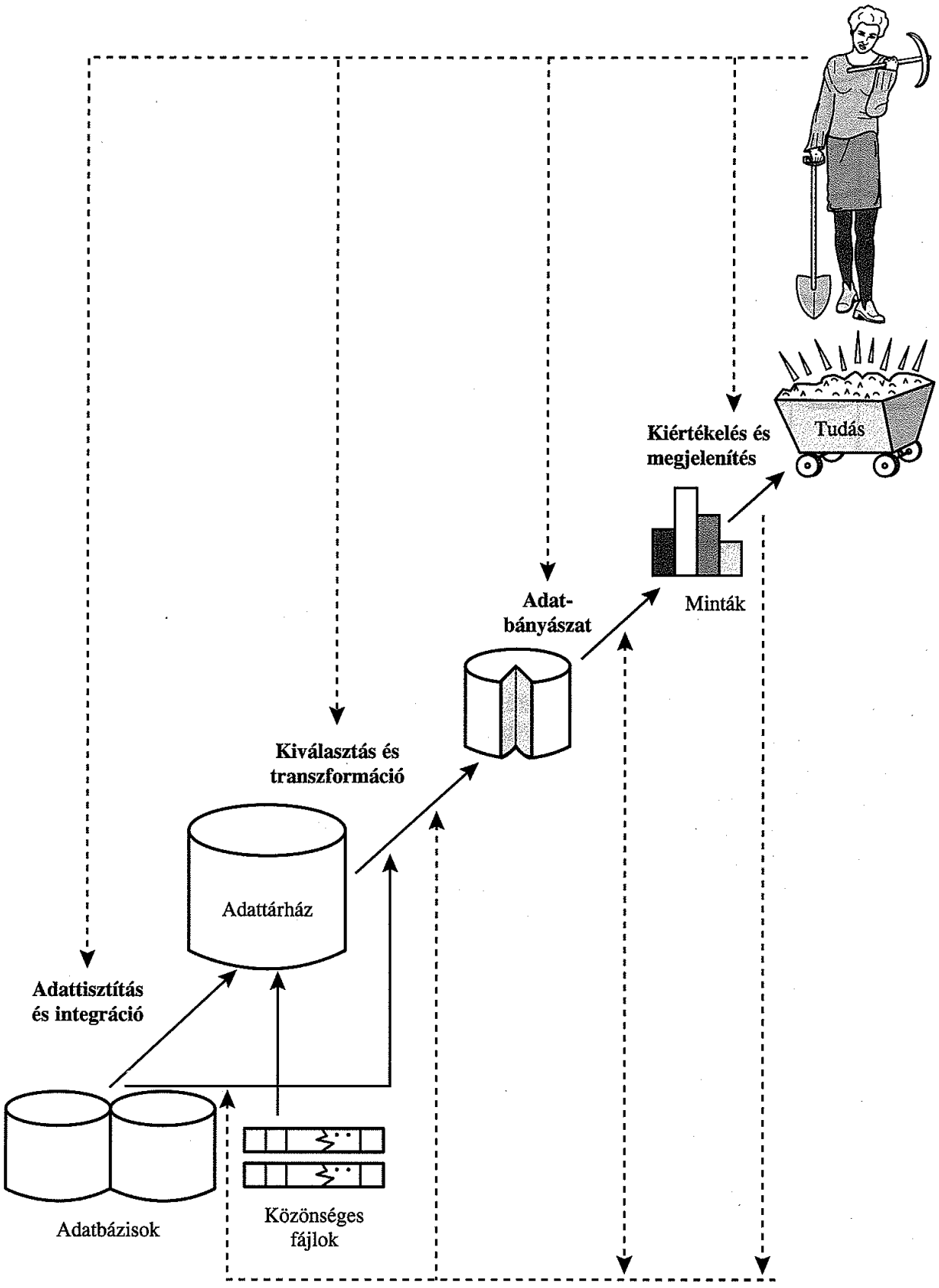
egyik meghatározó részének tekintik. A KDD az 1.4. ábrán vázolt iteratív folyamat, amely a következő lépésekből áll:

(1) **Adattisztítás** – Zajok és inkonzisztens adatok eltávolítása.
(2) **Adatintegrálás** – Több adatforrás összekapcsolása, egyesítése.[1]
(3) **Adatkiválasztás** – Az elemző feladathoz tartozó adatok kiolvasása az adatbázisból.
(4) **Adat-transzformáció** – Ennek során az adatokat olyan formára hozzuk, hogy azok bányászhatók legyenek, például összefoglaló vagy csoportműveleteket végrehajtva.[2]
(5) **Adatbányászat** – Kiemelt fontosságú eljárás, amelynek során intelligens műveleteket, műveletsort hajtunk végre az adatminták kiemelése érdekében.
(6) **Minta kiértékelése** – A tudást reprezentáló, valóban érdekes minták meghatározása valamilyen érdekességi mérték alapján.
(7) **Tudásmegjelenítés** – Ennek során a „kitermelt" tudást tudáskifejező technikákkal és megjelenítéssel tárjuk a felhasználó elé.

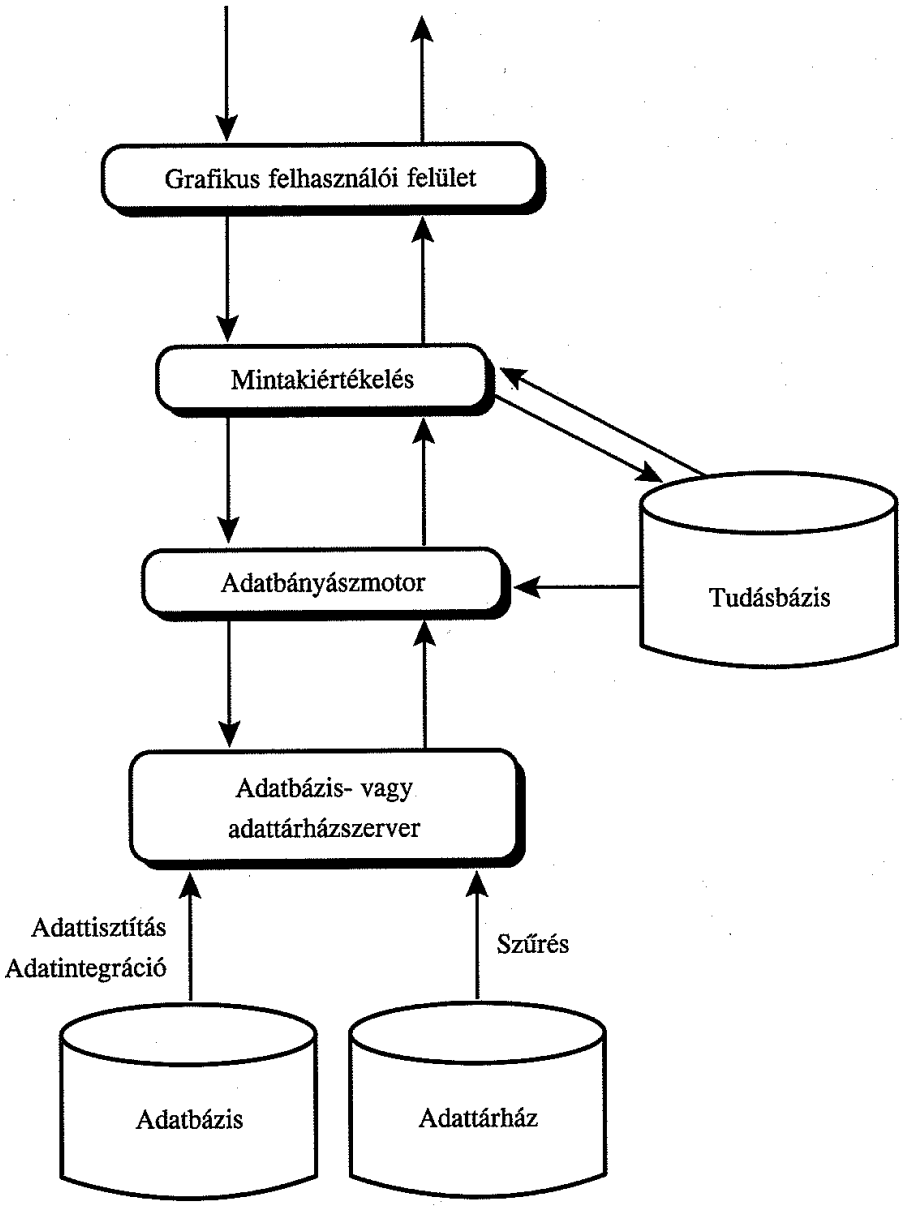
Az adatbányászati fázis során interaktív kapcsolatba kerülhetünk a felhasználóval és a tudásbázissal. Az érdekes mintát átadjuk a felhasználónak, és esetleg új tudásként bekerül a tudásbázisba. Vegyük észre, hogy az adatbányászat ebben a nézetben tényleg csak egy, de központi fontosságú lépése a fent bemutatott folyamatnak, hiszen rejtett, elfedett mintákat tár föl a mintakiértékelési fázis számára.

Egyetértünk abban, hogy az adatbányászat csupán egy lépés a tudásfeltárás folyamatá-

1 | Az információiparban elterjedt terminológia szerint az adattisztítás és adatintegráció az adatok előfeldolgozását jelenti, az ennek során keletkezett eredményadatokat egy adattárházban tárolják.
2 | Néha, különösen az adattárházak esetében, az adat-transzformáció és adatkonszolidáció az adatkiválasztás előtt történik.



1.4. ábra | Az adatbányászat mint a tudásfeltárási folyamat egy lépése



1.5. ábra | Tipikus adatbányászati rendszer architektúrája

ban. Ugyanakkor az iparban, a médiában és adatbázisok környezetében az „adatbányászat" terminológia jóval népszerűbb, mint a hosszabb „tudásfeltárás adatbázisokban" kifejezés. Ezért ebben a könyvben az adatbányászat szó használatát részesítjük előnyben. Átvesszük azt a széles körben elfogadott tartalmi jelentést, hogy az adatbányászat olyan folyamat, amely érdeklődésre számot tartó tudás nagy mennyiségű adatból történő feltárása, függetlenül attól, hogy az adatok adatbázisokban, adattárházakban vagy egyéb információtárakban tárolódnak.

Ilyen megközelítésben a tipikus adatbányászati rendszer szerkezetileg az alábbi fő alkotórészekből épülhet föl (1.5. ábra):

- **Adatbázis, adattárház, egyéb információtárak** – Egy vagy több adatbázis, táblázatok, egyéb információkészletek. Adattisztítás és integrálás esetleges végrehajthatása.

- **Adatbázis- vagy adattárházszerver** – Felelős a felhasználó adatbányászati igényeinek megfelelő, a tárgykörhöz tartozó adatok eléréséért, szolgáltatásáért.
- **Tudásbázis** – Tárgyköri tudás, amely a keresés vezérfonalát adja, vagy meghatározza a megtalált minták érdekességét. Ilyen tudás magába foglalhat **fogalmi hierarchiákat**, amelyek használatával attribútumok és attribútumértékek különböző absztrakciós szintekbe szervezhetők. Az olyan tudás, mint a felhasználói meggyőződés, amely a minta váratlanságából következtet a minta érdekességére, szintén része lehet a tudásbázisnak. Más példák a tárgyköri tudásra egyéb érdekesség-megszorítások, küszöbértékek és metaadatok (például adatleírások több heterogén forrásból).
- **Adatbányászmotor** – Központi fontosságú az adatbányászati rendszerben, és ideális esetben funkcionális modulok halmaza karakterizációs, társítási, osztályozási, klaszter-, fejlődés- és devianciaelemzési feladatok támogatására.
- **Mintakiértékelő modul** – Ez a komponens tipikusan érdekességi mértékeket (1.5. alfejezet) használ, és kölcsönhatásban áll az adatbányászati modullal, vagyis a keresést az érdekes minták irányába tereli. Esetleg felhasznál érdekességi küszöbértékeket, hogy azok segítségével kiszűrjön megtalált érdektelen mintákat. A mintakiértékelő modul más esetben integrált része lehet az adatbányászati modulnak, a használt adatbányászati modul megvalósításának függvényében. A hatékony adatbányászat érdekében kiemelten ajánlott a minta kiértékelését végző modult az adatbányászati folyamat minél mélyebb szintjére helyezni, és ezzel a keresést kizárólag az érdekes mintákra korlátozni.
- **Grafikus felhasználói felület** – A felhasználó és az adatbányászati rendszer közötti kommunikációt biztosítja, megengedve, hogy a felhasználó adatbányászati lekérdezéseket és feladatokat specifikálva, a minta keresési irányát befolyásoló információkat átadva és köztes adatbányászati eredménymintákra épülő tájékozódó bányászatot folytatva interaktív kapcsolatba lépjen a rendszerrel. Ez a komponens az előzőkön túlmenően lehetővé teszi a felhasználó számára, hogy tallózzon adatbázisokban és adattárházsémákban vagy adatstruktúrákban, a kitermelt mintákat kiértékelje, illetve azokat különböző formában megtekintse.

Az adattárházak megvilágításában az adatbányászat az on-line elemző feldolgozás (OLAP) fejlett platformjának tekinthető, de valójában messze túllépi az adattárházrendszerek összesítő elemzéseinek meglehetősen szűk mezsgyéjét, miután tartalmi összefüggések feltárását támogató, még fejlettebb technikákat is magába foglal.

Bár számos adatbányászati rendszer érhető el a piacon, közülük nem mindegyik tekinthető igazi adatbányászatnak. Egy adatelemző rendszer, amely alkalmatlan nagy mennyiségű adatok kezelésére, sokkal inkább a tanuló gépek, statisztikai elemző rendszerek vagy a tapasztalati rendszerek közé sorolható. Azok a rendszerek pedig, amelyek nagy méretű adatbázisokban csoportértékek feltárását vagy deduktív lekérdezések megválaszolását biztosítják, sokkal inkább tartoznak az adatbázisrendszerek, az információ-visszakereső vagy a deduktív adatbázisrendszerek közé.

Az adatbányászat egyesíti magában a különböző kutatási és fejlesztési területeken – az adatbázis-technológiában, a statisztikában, a gépi tanulásban, a nagy hatékonysággal végzett számítások területén, a mintafelismerésben, a neurális háló kutatásokban, az adat-megjelenítésben, az információ-visszakeresésben, a kép- és jelfeldolgozásban és a téradatelem-

zésben – kidolgozott, illetve alkalmazott technikákat. Ez a könyv az adatbázisok szempontjából tárgyalja az adatbányászatot, vagyis a hangsúlyt a *nagy méretű* adatbázisoknál alkalmazható *hatékony* és *skálázható* technikák bemutatására helyezi. Ahhoz, hogy egy algoritmus **skálázható** legyen, végrehajtási idejének lineárisan kell növekednie az adatbázis méretének növekedésével, az adott rendszererőforrások, például a lemezterület és az operatív tár figyelembevételével. Adatbányászatot folytatva érdekes tudás, szabályszerűség vagy magas szintű információk emelhetők ki az adatbázisokból, és válhatnak különböző szempontok szerint láthatóvá és olvashatóvá. A feltárt tudás döntéshozatalokban, folyamatirányításban, információkezelésben és lekérdezések feldolgozásában egyaránt felhasználható. Ez az oka annak, hogy az adatbányászatot az információipar egyik legígéretesebb, több kutatás-fejlesztési területet átfogó és az adatbázisok tudáshatárát átlépő fejlesztési területnek tekintik.

1.3. | Adatbányászat – az adatok típusa

Ebben az alfejezetben megvizsgálunk számos különböző típusú adattárat, amelyeken adatbányászat folytatható. Az adatbányászat általánosságban bármilyen jellegű információtár esetében (beleértve a relációs adatbázisokat, az adattárházakat, tranzakciós adatbázisokat, fejlett adatbázisrendszereket, közönséges fájlokat és a World Wide Webet) alkalmazható. A fejlett adatbázisrendszerek felölelik az objektumorientált és objektumrelációs adatbázisok, a specifikus alkalmazásorientált adatbázisok (téradatbázisok, idősor-adatbázisok, szöveges adatbázisok és multimédia-adatbázisok) rendszerét. A bányászati feladatok és technikák a különböző tárolórendszerek mindegyikében különbözők lehetnek.

Jóllehet feltételezzük, hogy az olvasó ismeri az információs rendszerekkel kapcsolatos alapfogalmakat, a következőkben röviden bevezetjük az olvasót a felsorolt főbb adattároló rendszerek világába. A szövegben idézett fogalmak szemléltetésére az *ElektroMind* képzeletbeli áruházlánc példáját használjuk.

1.3.1. | Relációs adatbázisok

Egy adatbázisrendszer, vagy másként nevezve **adatbáziskezelő-rendszer (Database Managament System, DBMS)** egymással kapcsolatban álló adatok gyűjteménye – ez **adatbázisként** ismert – és számítógépes programok egy halmaza, amely kezeli és eléri az adatokat. A programok olyan mechanizmusokat valósítanak meg, amelyek támogatják, illetve biztosítják az adatbázis szerkezeti meghatározását, az adatok tárolását és konkurens, osztott és számítógépes hálózaton elosztott elérését, garantálva a tárolt adatok konzisztenciáját és hozzáférési biztonságát rendszerösszeomlás vagy illetéktelen hozzáférési kísérlet esetén is.

A **relációs adatbázis** egyedi nevekkel ellátott **táblák** halmaza. Mindegyik tábla **attribútumok** (*oszlop* vagy *mező*) egy halmaza, és szokásosan **többesek (tuples)** (*sor* vagy *rekord*) nagy elemszámú halmazát tárolja. Egy relációs tábla minden ilyen többese egy objektumot reprezentál, amelyet egy egyedi kulcs, attribútumértékek egy halmaza azono-

sít. A relációs adatbázishoz gyakran készítenek szemantikus adatmodellt, például az **egyed–kapcsolat (EK) adatmodellt (Entity-Relationship, ER)**, amely az adatbázist egyedek egy halmazával és a közöttük fennálló kapcsolatokkal modellezi.

Tekintsük a következő példát.

1.1. példa | Az *ElektroMind* áruházlánc adatait a következő relációs táblákkal modellezzük: *vásárló, árucikk, alkalmazott* és *áruház*. Az itt bevezetett táblák által tárolt adatrészletek az 1.6. ábrán láthatók.

- A *vásárló* reláció az egyes vásárlók adatait – egy egyedi azonosítót *(v_ID)*, a vásárló nevét, címét, életkorát, foglalkozását, évi keresetét, hitelképességét stb. – tartalmazza.
- Hasonlóan, az *árucikk, alkalmazott* és *áruház* táblák is jellemző tulajdonságokat megadó attribútumok összességeként jelennek meg.
- A táblák reprezentálhatnak két vagy több relációs tábla közötti kapcsolatot is. Esetünkben ilyen tábla a *vásárlás* (a vásárló árucikkeket vásárol, ezzel létrehozva egy eladási tranzakciót, amit egy alkalmazott bonyolít le), az *eladások* (azoknak az árucikkeknek a listája, amelyeket a vevő a vásárlás keretében megvásárolt) és a *munkahely* (az *ElektroMind* áruházláncnak az az áruháza, ahol a tranzakciót bonyolító eladó dolgozik). ■

A relációs adatokhoz **adatbázis-lekérdezésekkel** férhetünk hozzá. A lekérdezések valamilyen lekérdező nyelven (ilyen az SQL) vagy a grafikus felhasználói felület kínálta eszközök segítségével fogalmazhatók meg. Az utóbbi esetben a felhasználó például egy menü segítségével adhatja meg azokat az attribútumokat, amelyek értékeire kíváncsi, illetve az attribútumok értékeire kirótt megszorításokat. A megadott lekérdezésből relációs operációkat, összekapcsolást, kiválasztást és vetítést tartalmazó kifejezés keletkezik, amely a lehetséges optimalizálás után végrehajtódik. Egy lekérdezés lehetővé teszi a tárolt adatok egy részhalmazának elérését. Tételezzük föl, hogy a feladat az *ElektroMind* áruházlánc adatainak elemzése. Relációs lekérdezéssel például kérhetjük az utolsó negyedévben eladott árucikkek listáját. A relációs lekérdező nyelvek tartalmaznak csoportműveleteket is, összegzést (`sum`), átlagszámítást (`avg`), számlálást (`count`), értékhalmazok legnagyobb (`max`) és legkisebb (`min`) értékének meghatározását. Ezeknek a műveleteknek a segítségével különféle lekérdezések fogalmazhatók meg, például a legutóbbi hónapban lebonyolított vásárlások száma áruházankénti csoportosításban, a decemberben lebonyolított forgalomkimutatás, illetve melyik eladó bonyolította le a legtöbb áruforgalmi tranzakciót.

Amikor adatbányászatot folytatunk relációs adatbázisokban, a fentieken túl kereshetünk *trendeket* vagy *adatmintákat*. Példának okáért az adatbányászati rendszerek segítségével végezhetünk olyan elemzéseket, amelyekkel a vásárlók adatai alapján felmérhető egy új vásárlónak nyújtandó hitel kockázata az új vásárló jövedelmének, életkorának és korábbi hitelakcióinak alapján. Adatbányászati rendszerek képesek olyan eltérések feltárására is, hogy például bizonyos árucikkek kereslete messze eltér az előző évi eladás alapján becsült elvárt mennyiségtől. Az ilyen elhajlások azután tovább elemezhetők (például történt-e lényeges változás az árucikk csomagolásában, lényegesen drágult-e a szóban forgó árucikk). A relációs adatbázisok a legelterjedtebbek és a legjobban hozzáférhető gazdag információtárak, ezért adatbányászati vizsgálatunk során központi szerepet játszanak.

vásárló

v_ID	név	cím	életkor	jövedelem	hitel_info	...
V1	Kovács Rozi	1033 Bp. Szentendrei út 133.	32	100 000	i	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...

árucikk

árucikk_ID	megnev	márka	kategória	típus	ár	gyártási_hely	szállító	költség
C3	nagy felbontású tv	Thoshiba	nagy felbontású	tv	59 998	Japán	NikoX	30 000
C8	többlemezes	Sanyo	többlemezes	CD-lejátszó	9 899	Japán	Music Front	3 000
...	CD-lejátszó	...	...	...	...	...	...	...

alkalmazott

alkalmazott_ID	név	osztály	besorolás	fizetés	jutalék
A55	Jánosi Janka	Szórakoztató eletronika	oszt.vez.	300 000	2%
...	...	...	...	...	...

áruház

áruház_ID	név	cím
ARH1	Városter	1130 Bp., Angyalföldi út 44.
...	...	...

vásárlás

tranzakció_ID	v_ID	alkalmazott_ID	dátum	idő	fizetési_mód	összeg
T100	V1	A55	1998. jan. 1.	17:41	OTP-kártya	97 387
...	...	...	...	...	...	...

eladások

tranzakció_ID	árucikk_ID	mennyiség
T100	C3	1
T100	C5	2
...	...	...

munkahely

alkalmazott_ID	áruház_ID
A55	ARH1
...	...

1.6. ábra | Az *ElektroMind* relációs adatbázisának részlete

1.3.2. | Adattárházak

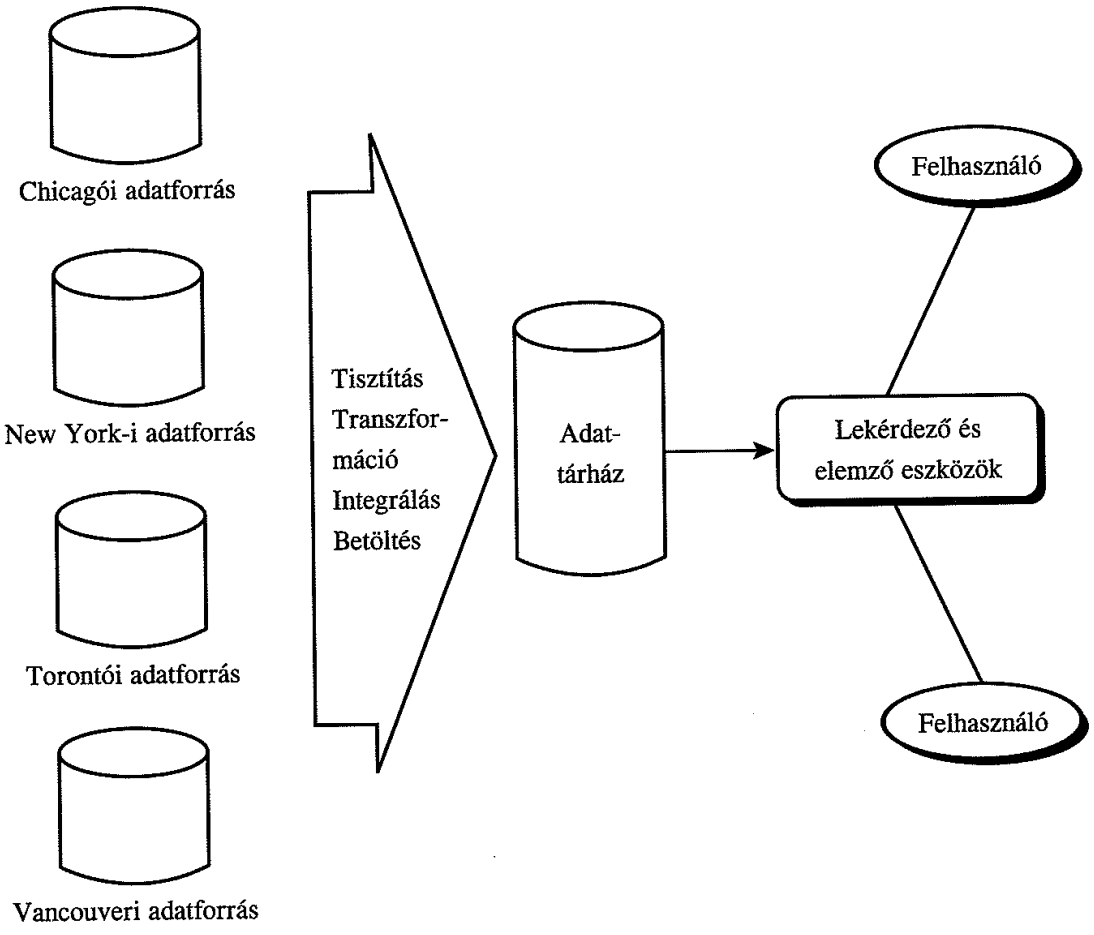
Tételezzük föl, hogy az *ElektroMind* áruházlánc sikeres nemzetközi vállalat, amelynek világszerte vannak áruházai. Minden áruház saját adatbázist tart fenn. A vállalat elnökének az a kívánsága, hogy készüljön elemzés a vállalat harmadik negyedévi forgalmáról árucikkek és áruházak szerinti bontásban. Ez nem könnyű feladat, ha tekintetbe vesszük, hogy az elemzéshez szükséges adatok különböző adatbázisokban, különböző földrajzi helyeken találhatók.

Ha az *ElektroMind* áruházlánc egy áruházzal rendelkezne, a feladat egyszerű lenne.

Az **adattárház** olyan információtár, amely különböző adatforrásokat egységes séma szerint integrál, és rendszerint fizikailag nem osztott környezetben valósul meg. Az adattárházak megvalósítása az adatokon végzett olyan feldolgozási folyamat eredménye, amely magába foglalja az adatok tisztítását, egyesítését, meghatározott transzformációkat, az adatok betöltését és folyamatos frissítését. Ezzel a folyamattal részletesen a 2. fejezet foglalkozik. Az 1.7. ábra az *ElektroMind* áruházlánc adattárházának felépítését illusztrálja.

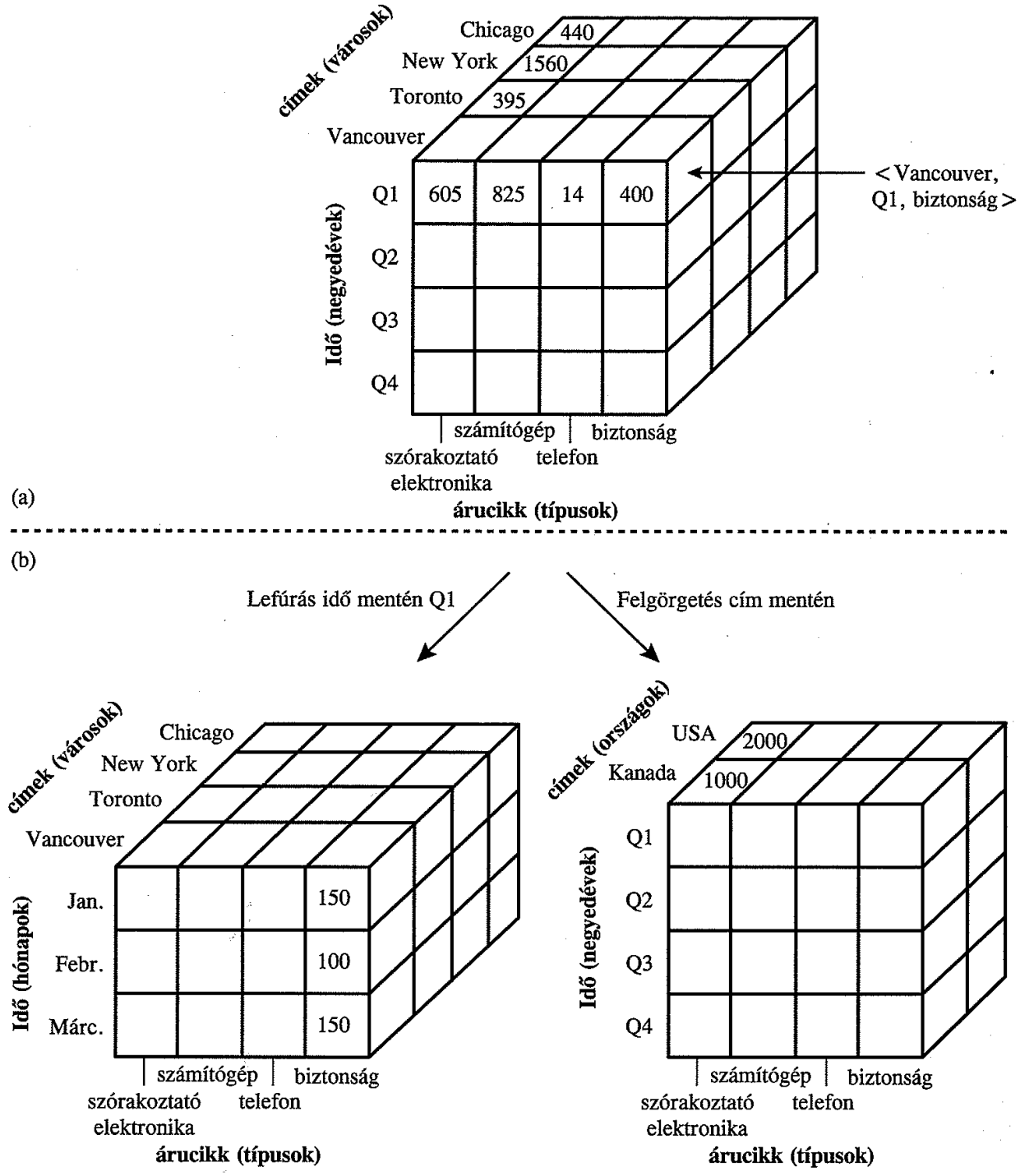
Annak érdekében, hogy könnyítsük a döntéshozatalt, az adatok egy adattárházban *fő témakörök mentén szerveződnek*, például az *ElektroMind* áruházlánc esetében vásárlók, árucikkek, szállítók, tevékenységek stb. A tárolt adatok *történeti információkat* is mutatnak (például az elmúlt 5–10 év távlatából) és tipikusan *aggregátumok*. Az egyes eladási tranzakciók részleteit leíró adatok tárolása helyett például az adattárház árucikk-típusonkénti összesített értékeket tárolhat minden egyes áruházra vagy magasabb szinten régióra.

Az adattárház rendszerint többdimenziós adatbázis-szerkezettel modellezhető, ahol az egyes **dimenziók** az adattárházséma egy-egy attribútumának, attribútumcsoportjának felelnek meg, és ahol az egyes **cellák** (tárolóelemek) aggregált értékeket tárolnak, például *darabszám*, *forgalom*. Az adattárházak tényleges fizikai felépítése lehet relációs adattár vagy **többdimenziós adatkocka**. Az adattárház adatoknak egy többdimenziós nézetét kínálja, és lehetővé teszi az összesített adatokon végezhető előszámításokat, illetve a tárolt értékekhez történő gyors hozzáférést.



1.7. ábra | Az *ElektroMind* áruházlánc adattárházának tipikus felépítése

1.2. példa | Az *ElektroMind* vállalat összesített forgalmi értékeinek adatkockáját láthatjuk az 1.8.(a) ábrán. A kocka három *dimenziója* a cím (*Chicago, New York, Toronto* és *Vancouver* városnevekkel), az idő (*Q1, Q2, Q3* és *Q4* negyedévekkel) és az árucikk (*szórakoztató elektronika, számítógép, telefon, biztonság árucikktípusokkal* megadva). Az adatkocka által



1.8. ábra | A többdimenziós adatkocka általában adattárházaknál használatos. (a) Az *ElektroMind* áruházlánc fő témakörök szerinti összesített (aggregált) értékei. (b) Az (a) adatkocka értékeiből lefúrással, illetve összesítéssel származtatott részösszegek (lefúrás), illetve göngyölt összegfokozatok (felgörgetés). Az olvashatóság érdekében a kockának csak néhány tárcellájában tüntettük föl az értéket

tárolt összesített érték minden esetben az ezrekben mért *forgalom*. Például a biztonsági berendezések (*biztonság*) első negyedévben mért összforgalma Vancouverben 400 000 dollárt tesz ki (*Vancouver, Q1, biztonság*). További adatkockák használhatók abból a célból, hogy az egyes dimenziók mentén különböző SQL-csoportosításoknak (group by záradék) megfelelő aggregátumokat tároljunk (például az eladások száma városok és negyedévek vagy városok és árucikkek, vagy akár minden egyes dimenzió szerint). ■

Azok az olvasók, akik már hallottak *adatpiacról,* fölvethetik azt a kérdést, hogy *mi a különbség az adatpiac és az adattárház között*. Az adattárház egy *teljes szervezethez* tartozó tárgyakat, tárgyköröket átfogó adatok összessége és *vállalatméretű*, míg az **adatpiac** egy szervezet egy szervezeti egységére vonatkozó adatok gyűjteménye és így *részleg méretű* nyilvántartás.

Az adattárházak az általuk nyújtott többdimenziós adatnézeteknek és az aggregált értékek előfeldolgozásának köszönhetően különösen alkalmasak az **on-line elemző feldolgozásra**, vagyis **OLAP**-ra. Az OLAP-műveletek a vizsgált adatok tárgykörére vonatkozó háttérismereteket használnak abból a célból, hogy az adatok különböző absztrakciós szinteken történő megjelenítését biztosítsák. Ezek a műveletek különböző felhasználói nézőpontokhoz illeszkednek. Az OLAP-műveletek tartalmaznak például részaggregátum- (**lefúrás**) és összegfokozat-képző (**felgörgetés**) műveleteket, amelyek lehetővé teszik a felhasználó számára, hogy az adatokat különböző összegzési szinteken láthassa, ahogy ezt az 1.8.(b) ábra szemlélteti: valaki a *negyedéves* forgalmat *havi bontásban*, más a *városokra* kimutatott forgalomból az illető *ország* forgalmát szeretné látni.

Jóllehet az adattárházeszközök lehetővé teszik az adatok elemzését, az adatbányászathoz további eszközökre van szükség, amelyek mélyebb és automatizált adatelemzést tesznek lehetővé. Az adattárház-technológiát részletesen a 2. fejezetben tárgyaljuk.

1.3.3. | Tranzakciós adatbázisok

Általánosságban, egy **tranzakciós adatbázis** egy állomány, amelynek minden rekordja egy tranzakciót reprezentál. Egy tipikus tranzakció tartalmaz egy egyedi azonosítót *(tranzakció_ID)* és a tranzakciót alkotó **tételek** listáját (például egy vásárlás alkalmával megvásárolt árucikkek listája). A tranzakciós adatbázis további táblákkal is rendelkezhet, amelyek a vásárlással kapcsolatos további információkat tartalmaznak, olyanokat, mint a tranzakció dátuma, a tranzakcióban részt vevő vásárló azonosítókódja, a tranzakciót bonyolító eladó azonosítókódja, az áruház kódja az áruházláncon belül stb.

1.3. példa. | A tranzakciók tárolhatók egy táblában – egy rekord egy tranzakció. Az *ElektroMind* áruházlánc tranzakciós adatbázisának egy részlete látható az 1.9. ábrán. A relációs adatbázisok szemszögéből nézve az ábrán látható *értékesítés* tábla egy beágyazott reláció, mert az *árucikkcikk_ID_lista* attribútum *árucikkek* egy halmazát tartalmazza. Miután a legtöbb relációs adatbázisrendszer nem engedi meg a beágyazott relációs szerkezeteket, a tranzakciós adatbázist rendszerint az 1.9. ábrán látható táblaszerkezethez hasonló szerkezetű, közönséges fájlban tároljuk. Az 1.9. ábra *értékesítés* táblájának minden rekordját több

értékesítés

tranzakció_ID	árucikk_ID_lista
T100	C1, C3, C8, C16
...	...

1.9. ábra | Az *ElektroMind* vállalat tranzakciós adatbázisának részlete

rekordra bontjuk, amelyek szabványos relációként tárolt adatokként, az 1.6. ábrán látható *eladások* táblában tárolt adatokhoz hasonlóan jelennek meg. ■

Az adatelemző nagy valószínűséggel kíváncsi lehet például a Kovács Rozi által vásárolt árucikkek listájára, vagy hogy hány tranzakció hivatkozik a C3-mal azonosított árucikkre. Az ilyen kérdések megválaszolása a teljes tranzakciós adatbázis átvizsgálását igényelheti.

Tegyük föl, hogy mélyebbre szeretnénk hatolni az adatok vizsgálatában a „Melyek az együtt jól eladható termékek?" kérdéssel. Az ilyen *„bevásárlókosár"* típusú adatelemzés lehetővé tenné számunkra egyes különböző árucikkek készletbe foglalását, amely stratégiailag fontos lépés lehet a vállalat forgalmának jelentős növelésében. Például ha kiderül, hogy a nyomtatókat általában számítógéppel együtt vásárolják, akkor egy magas áron kínált nyomtatómodellt a vevő által kiválasztott számítógéppel együtt kínálunk engedményes áron, abban a reményben, hogy a drága nyomtatómodellből több fog elfogyni. A szabványos adatlekérdező rendszerek nem képesek a fenti kérdésekre választ adni, az adatbányászati rendszerek tranzakciós adatokból képesek olyan árucikkekből álló készletek feltárására, amelyekhez tartozó cikkek gyakran együtt kelnek el.

1.3.4. | Fejlett adatbázisrendszerek és adatbázis-alkalmazások

A relációs adatbázisok hosszú ideje széles körben használtak üzleti alkalmazásokban. Az adatbázisrendszerek fejlődésével különböző fejlett adatbázisrendszerek jöttek létre, illetve vannak fejlesztés alatt, amelyek új adatbázis-alkalmazások igényeinek kielégítését célozzák.

Az új adatbázis-alkalmazások téradatok (térképek), mérnöki tervezési adatok (épületek, rendszerkomponensek, integrált áramkörök) hipertext és multimédia-adatok (szövegek, képek, hangfelvételek), időkapcsolatú adatok (történelmi, történeti adatok, tőzsdei adatok) és a World Wide Web (tetemes, világméretekben megosztott információtár, amely az interneten keresztül vált elérhetővé) kezelésére irányulnak. Ezek az alkalmazások hatékony adatszerkezeteket és skálázható eljárásokat igényelnek, hogy képesek legyenek összetett objektumszerkezetek, változó hosszúságú rekordok, félig strukturált vagy strukturálatlan adatok, szöveges és multimédia-adatok és adatbázissémák összetett szerkezeteinek és dinamikus változásainak kezelésére.

Válaszként az igényekre, fejlett adatbázisrendszerek és specifikus alkalmazásorientált adatbázis-kezelő rendszerek jöttek létre. Ezek felölelik az objektumorientált, objektumrelációs, térbeli, időbeli, idősor-, szöveges és multimédia-adatbázisokon át a legkülönbözőbb heterogén és örökölt adatbázisokat és web alapú általános információs rendszereket.

Míg az ilyen adatbázisok vagy információtárak kifinomult képességeket követelnek meg annak érdekében, hogy hatékonyan oldják meg nagy mennyiségű összetett adat tárolását, elérését és módosítását, termékeny táptalajként szolgálnak az adatbányászat számára, és sok kihívó, implementációs eredménnyel is járó kutatást inspiráltak ezen a területen.

Objektumorientált adatbázisok

Az **objektumorientált adatbázisok** alapja az objektumorientált programozási paradigma, ahol általában véve minden egyed egy objektum. Az *ElektroMind* példájánál maradva, objektumok lehetnek az egyes alkalmazottak, vásárlók vagy árucikkek. Egy objektumhoz kapcsolódó adatok és programkódok *egyetlen egységként* jelennek meg, és egy ilyen egységhez minden objektum az alábbiak szerint kapcsolódik:

- **Változók** halmaza, amelyek leírják az objektumot. A változók megfelelnek az egyed–kapcsolat adatmodell vagy relációs modell attribútumainak.
- **Üzenetek** halmaza, amelyet az objektum más objektumokkal vagy az adatbázis fennmaradó részével történő kommunikációra használ.
- **Metódusok** halmaza, amelyben minden metódusba be van égetve egy üzenet megvalósítása. Egy üzenet fogadásakor a metódus válaszként visszaad egy értéket. Például, a *get_photo(alkalmazott)* üzenetre a metódus kiolvassa és visszaadja a megadott alkalmazott fényképét.

Az azonos tulajdonságokkal rendelkező objektumok **objektumosztályokba** sorolhatók. Minden objektum az őt befogadó osztály egyik **előfordulása**. Az objektumosztályok osztály–alosztály hierarchiába szervezhetők úgy, hogy mindegyik osztály tulajdonságokat határoz meg, amelyek az osztályhoz tartozó valamennyi objektumra jellemzők. Példának okáért, az *alkalmazott* osztály változói lehetnek a *név*, *cím* és *születési_dátum*. Tegyük föl, hogy az *eladó* osztály alosztálya az *alkalmazott* osztálynak. Egy *eladó* objektum örökli szülőosztályának, az *alkalmazott* osztálynak minden változóját. Ezeken túl pedig rendelkezik olyan változókkal, amelyek az eladókat jellemző tulajdonságokat írnak le (például *jutalék*). Az osztályöröklődés előnye az információk megoszthatóvá/közössé tétele.

Objektumrelációs adatbázisok

Az **objektumrelációs adatbázisok** alapja az objektumrelációs adatmodell. Az objektumrelációs adatmodell a relációs modell olyan kiterjesztése, amely adattípusok gazdag kínálatával támogatja összetett objektumok kezelését és az objektumorientációt, továbbá a relációs lekérdező nyelveket speciális konstrukcióival kiegészítve támogatja az új adattípusok kezelését. Az objektumrelációs modell tehát kiterjeszti az alap relációs modellt a már fent említett összetett adattípusok, osztályhierarchiák és objektumöröklődés kezeléséhez szükséges többlet apparátus kínálatával. Az objektumrelációs adatbázisok egyre növekvő népszerűségnek örvendenek az iparban, az alkalmazásokban.

Az objektumorientált és objektumrelációs rendszerekben alkalmazott adatbányászat számos hasonlóságot mutat, és a relációs adatbányászattal szemben olyan technikák kifejlesztését igényli, amelyek összetett objektumstruktúrák, összetett adattípusok, osztály–alosztály hierarchiák, tulajdonságöröklődés, metódusok és eljárások kezelését célozzák.

Téradatbázisok

A **téradatbázisok** térbeli kapcsolatokat leíró információkat tárolnak. Ilyenek a földrajzi (térkép), VLSI-chip tervezési, orvosi vagy műholdfelvételeket tároló képi adatbázisok. A téradatok raszteres formában (ezek *n* dimenziós bit- vagy pixeltérképek) ábrázolhatók. Például egy kétdimenziós műholdfelvétel raszteradatként ábrázolható, ahol minden pixel egy adott terület csapadékmennyiségét regisztrálja. A térképek vektorformában reprezentálhatók: az utak, hidak, épületek, tavak geometriai alapelemek (pontok, vonalak, sokszögek) egyesítéseként azok részhalmazaként, illetve ilyen alakzatok hálójaként jelennek meg. A földrajzi adatbázisoknak számos alkalmazásuk van. Ilyenek az erdőgazdálkodás, az ökológia, a telefon- és elektromos kábelek vagy a víz- és szennyvízhálózatok karbantartása és telepítése. Ugyancsak földrajzi adatbázisokat használnak a személy- és áruszállítást segítő rendszerek. Az utóbbi rendszerek a taxisok számára például olyan térképet tárolhatnak, amely egy város úthálózatán kívül megjelöli az egyirányú utcákat, a fontosabb középületeket, kórházakat, éttermeket és a különböző városrészek közötti csúcsforgalom idején javasolt útvonalakat, továbbá valamennyi taxis kolléga pillanatnyi tartózkodási helyét.

Felmerülhet a kérdés, hogy milyen jellegű adatbányászat folytatható téradatbázisokban. Az itt végzett adatbányászat feltárhat olyan adatmintákat, amelyek leírják egy meghatározott hely (például park) közelében található épületek jellegét. Más adatminták leírhatják a hegyes vidékek éghajlati viszonyait különböző magasságokban, vagy a nagyvárosok szegénységi rátájában mutatkozó eltéréseket a városoknak a főbb autópályáktól mért távolságaik függvényében. Sőt, „térbeli adatkockák" hozhatók létre, amelyekkel az adatokat többdimenziós struktúrákba és hierarchiákba szervezhetjük, amelyeken az OLAP-műveletek (lefúrás, felgörgetés) végrehajthatók.

Időbeli és idősor-adatbázisok

Az időbeli és idősor-adatbázisok egyaránt időkapcsolatú adatokat tartalmaznak. Az **időbeli adatbázisok** rendszerint olyan relációs adatokat tárolnak, amelyeknek vannak időkapcsolatú attribútumai. Ezek az attribútumok különböző jelentéssel bíró eltérő időbélyegeket hordoznak. Egy **idősor-adatbázis** időben változó értékek sorozatát tárolja, például értéktőzsdei adatokat.

Adatbányászati technikák alkalmazhatók az objektumevolúciók jellegének vagy az adatbázisban tárolt objektumokon végbement változások tendenciájának feltárására. Az ilyen jellegű információk nagyon hasznosak lehetnek a különböző döntéshozataloknál és stratégiai tervezésekben. Példának okáért, banki adatok bányászása segítségünkre lehet a banktisztviselők egyes területekre történő irányításában az ügyfélforgalom függvényében. Értéktőzsdei adatok bányászhatók olyan adatminták feltárása érdekében, amelyek

befektetési stratégiák tervezését segítik (például melyik a legalkalmasabb idő *ElektroMind*-részvények vásárlására). Az ilyen elemzések elvégzéséhez tipikusan több különböző időgranulátum meghatározása szükséges: például az idő mérhető költségvetési, naptári vagy tanítási években. Az évek tovább finomíthatók negyedévekre vagy hónapokra.

Szöveges és multimédia-adatbázisok

A **szöveges adatbázisokban** az objektumokat szavak írják le. A szavakon nem egyszerűen kulcsszavakat kell értenünk, hanem sokkal inkább hosszú mondatokat vagy bekezdéseket, például egy termék leírását, hiba- vagy üzemzavar-jelentést, figyelmeztető üzeneteket, összefoglaló jelentéseket, megjegyzéseket vagy másfajta dokumentumokat. A szöveges adatbázisok nagymértékben strukturálatlanok lehetnek (például néhány weboldal a World Wide Weben). Más szöveges adatbázisok lehetnek valamennyire strukturáltak, vagyis *félig strukturáltak* (ilyenek az e-mail üzenetek és sok HTML/XML weboldal), míg megint mások relatíve jól strukturáltak (például könyvtári adatbázisok). A megszokott struktúrákkal rendelkező jól strukturált szöveges adatbázisok könnyen implementálhatók relációs adatbázis-kezelő rendszerekkel.

Mit hozhatunk felszínre adatbányászattal a szöveges adatbázisokból? Végső soron objektumosztályok általános leírását, kulcsszavakat, tartalmi kapcsolódásokat és nem utolsó sorban a szövegobjektumok klaszterviselkedését. Ahhoz, hogy ezt elérhessük, a szabványos adatbányászati funkciókat integrálni kell az információelérési technikákkal és a hierarchiák létrehozásával vagy használatával, különös tekintettel a szöveges adatokra (szószedet, szótár) vagy tudományág-orientált fogalmi osztályozási rendszerekre (vegyészeti, orvosi, jogi vagy közgazdasági).

A **multimédia-adatbázisok** képi, hang- és videoadatokat tárolnak. Ezeknek az adatbázisoknak az alkalmazási területei magukba foglalják a képi tartalmon alapuló visszakeresést, hangpostarendszereket, igény szerinti videorendszereket, a World Wide Webet és beszéd alapú felhasználói felületeket, amelyek közül az utóbbiak beszélt nyelven kiadott parancsokat ismernek föl. A multimédia-adatbázisoknak számolniuk kell nagy méretű objektumokkal, hiszen az olyan objektumoknak, mint a videofelvételek akár gigabájtokban mérhető tárolóterület is szükséges lehet. Különleges tárolási és keresési technikák szintén kívánatosak. Miután a video- és hangfelvételek valós idejű visszakeresést igényelnek annak érdekében, hogy a képi és hangi szünetek, illetve a rendszerpuffer túlcsordulásai elkerülhetők legyenek, ezekre az adatokra mint folytonos médiaadatokra szokás hivatkozni.

A multimédia-adatbázisokban folyó adatbányászathoz tárolási és keresési technikákat kell a szabványos adatbányászati funkciókkal integrálni. Ígéretes megközelítések közé sorolhatók a multimédia-adatkocka létrehozása, a különböző jellemző tulajdonságok leválasztása a multimédia-adatokról és a hasonlóságon alapuló mintaillesztés.

Heterogén és örökölt adatbázisok

Egy **heterogén adatbázis** kölcsönösen összekapcsolt autonóm adatbázis-komponensekből áll. A komponensek a lekérdezések megválaszolása érdekében történő információcsere céljából kommunikálnak egymással. Az egyik komponens adatbázis által tárolt objek-

tum jelentős mértékben különbözhet egy másik komponens adatbázis által tárolt objektumoktól, ami megnehezíti az objektumoknak a heterogén adatbázison vett átfogó értelmezését.

Az információtechnológiai fejlődés hosszú történetének szakaszait csak részben követő fejlesztések (magukba foglalva különböző hardver- és szoftverkörnyezetben elkészült alkalmazásokat) eredményeképpen számos vállalkozás rendelkezik ún. örökölt adatbázissal. Egy **örökölt adatbázis** *heterogén adatbázisok* egy csoportja, amely különböző jellegű adatrendszereket kapcsol össze, például relációs vagy objektumorientált adatbázisokat, hierarchikus és hálós adatbázisokat, táblázatokat, multimédia-adatbázisokat és fájlrendszereket. Egy örökölt adatbázisban a heterogén adatbázisok intranet vagy internet alapú hálózaton kapcsolódhatnak egymáshoz.

Az ilyen adatbázisok közötti információcserét nehezíti az, hogy adatreprezentációk közötti precíz transzformációs szabályok kidolgozása szükséges, figyelemmel az eltérő szemantikára. Vegyük például a különböző egyetemeken folyó képzést. Mindegyik egyetem rendelkezhet saját számítógépes rendszerrel, továbbá képesítési és tantervi követelményrendszerrel. Az egyik egyetemen negyedéves rendszerű képzés folyik, három adatbáziskurzust kínál és A+-tól F-ig terjedő osztályzatokkal értékel. Egy másik egyetemen féléves rendszerű képzésben két adatbáziskurzus végezhető 1–10 osztályzatú minősítéssel. Rendkívül nehéz a precíz kurzus–osztályzat megfeleltetés kidolgozása a két egyetemen honos rendszer között, ezért maga az információcsere is nehézségekbe ütközik. Az adatbányászati technikák esetleg érdekes megoldást kínálnak az információcsere problémájára azáltal, hogy az adatokat egy magasabb, jobban általánosított fogalmi szintre képezik le (például a *megfelel*, *jó* és *kiváló* minősítő kategóriák bevezetésével, illetve a diákok osztályzatának ezekre a kategóriákra való leképezésével), amellyel jelentősen megkönnyítik az információcserét.

A World Wide Web

A World Wide Web és a hozzá kapcsolódó osztott információs szolgáltatások, például az America OnLine, Yahoo!, AltaVista és Prodigy, gazdag világméretű on-line információs szolgáltatásokat kínálnak, amelyek az adatobjektumok egymáshoz láncolásával azok interaktív elérését biztosítják. A felhasználók az őket érdeklő információkat keresve az egyik objektumból egy linken keresztül a másikba léphetnek át. Az ilyen rendszerek széles körű lehetőségeket és feladatokat jelentenek az adatbányászat számára. Például a felhasználó hozzáférési sémáinak megértése nem csak a rendszertervezés tökéletesítéséhez (erősen korreláló objektumokhoz való hatékony hozzáférés), de jobb piaci döntésekhez is vezet (reklámok elhelyezése gyakran látogatott dokumentumokban vagy jobb vásárló–felhasználó osztályozás és magatartáselemzés). A felhasználók hozzáférési mintáinak ilyen osztott információs környezetben történő kigyűjtését *hozzáférési útvonalminta* bányászatának nevezik.

Jóllehet a weboldalak emberi szemmel nézve tetszetősek és informatívak, ugyanakkor erősen strukturálatlanok és nem rendelkeznek előre definiált sémával, típussal és mintával. Ily módon számítógéppel nehezen értelmezhető az eltérő weboldalak tartalmi jelentésköre, és az oldalak nehezen strukturálhatók módszeresen szervezett információelérés és adatbányászat céljából. Azok a webszolgáltatások, amelyek kulcsszavakra épülő kere-

sést biztosítanak anélkül, hogy az egyes weblapok mögötti tartalmi környezetről ismeretekkel rendelkeznének, csak korlátozott mértékben segítik a felhasználókat. Az olyan webkeresés pédául, amely egyetlen kulcsszó keresésén alapszik, a keresett szót tartalmazó weboldalakra mutató linkek százaival tér vissza, jóllehet a megtalált weboldalak többsége nincs kapcsolatban azzal, amit a felhasználó valójában keres. Vajon az adatbányászat nagyobb segítséget nyújt, mint a webes keresőszolgálat? Segítségünkre lehet abban, hogy általánosságban megismerjük az információ eloszlását a weben, a weboldalak jellegét és a különböző weboldalak közötti kapcsolatokat? Segítségünkre lehet-e egy meghatározott témát feldolgozó weboldal megtalálásában? Jó osztályozást tud-e adni az internetes oldalakról? Ezek a kérdések további feladatokat vetnek fel az adatbányászat számára.

1.4. | Milyen minták bányászhatók?

Az eddigiekben számos különböző típusú adattároló- és adatbázisrendszert tekintettünk át, amelyeken adatbányászat folytatható. Vizsgáljuk most azt, hogy milyen típusú adatminták bányászhatók.

Adatbányászatban lehetőség van arra, hogy megadjuk azokat a mintákat, amelyeket az adatbányászati feladatokban meg akarunk találni. Az adatbányászati feladatok általában két osztályba sorolhatók: leíró és előrejelző. A leíró adatbányászati feladatok az adatbázisban tárolt adatok általános jellemzőit tárják fel. Az előrejelző adatbányászati feladatok a meglevő adatokból következtetnek, prognosztizálnak.

Néhány esetben a felhasználónak még elképzelése sincs arról, hogy milyen adatminták lehetnek érdekesek számára, ennél fogva több különböző minta keresésével próbálkozik párhuzamosan. Szükség lehet tehát olyan adatbányászati rendszerre, amely lehetővé teszi több különböző minta bányászatát, hogy ezzel alkalmazkodjék a különböző felhasználói elvárásokhoz vagy alkalmazásokhoz. További elvárás, hogy az adatbányászati rendszerek képesek legyenek különböző finomságú minták (minták különböző absztrakciós szinteken történő) feltárására. Az adatbányászati rendszereknek azt is lehetővé kell tenniük, hogy a felhasználók ajánlásokat fogalmazhassanak meg, ezzel irányítva azoknak a mintáknak a keresését, amelyek érdeklik őket. Miután néhány minta nem alkalmazható az adatbázisban tárolt valamennyi adatra, minden feltárt mintát rendszerint egyfajta bizonyossági vagy „hitelességi" mérték is kísér.

A következőkben bemutatjuk az adatbányászati funkciókat és az általuk feltárható adatmintákat.

1.4.1. | Fogalom-/osztályleírások: karakterizáció és diszkrimináció

Az adatok osztályokhoz vagy fogalmakhoz kapcsolhatók. Példának okáért az *ElektroMind* áruházlánc esetében az eladásra szánt árucikkek egy-egy osztálya a *számítógép* és a *nyomtató*, a vásárlók pedig fogalmi besorolás szerint a kiadásokat tekintve lehetnek *takarékos* és *nem takarékos* vásárlók. Hasznos lehet az egyes osztályok és fogalmak tömör leírása rövid, velős és mégis precíz kifejezések használatával. Az ilyen leírásokat **fogalom-/osztályleírásoknak** nevezzük. Ezek a leírások származtathatók (1) *adatkarakterizációval*

– általában véve összesítjük a vizsgált osztályhoz (gyakran **célosztálynak** nevezik) tartozó adatokat; (2) *adatdiszkriminációval* – a célosztályt összehasonlítjuk egy összehasonlító osztállyal (gyakran **kontrasztosztálynak** hívják) vagy ezek egy halmazával; vagy (3) adatkarakterizációval és adatdiszkriminációval.

Az **adatkarakterizáció** az adatok célosztálya általános jellemzőinek és jellegzetességeinek összesítése. A felhasználó által definiált osztályhoz tartozó adatok jellemzően adatbázis-lekérdezéssel gyűjthetők össze. Például ahhoz, hogy tanulmányozhassuk azoknak a szoftvertermékeknek a jellemzőit, amelyek forgalma az elmúlt évben 10%-kal nőtt, az azokat leíró adatokat megkaphatjuk egy SQL-lekérdezés végrehajtásával.

Számos funkció létezik hatékony adatösszesítésre és -karakterizációra. Például, az adatkocka alapú OLAP-felgörgetés (1.3.2. alfejezet) használható a felhasználói irányítású, megadott dimenzió mentén történő adatösszevonásra. Ezt a folyamatot majd a 2. fejezetben, az adattárházak vizsgálata során fejtjük ki részletesen. Az *attribútumorientált adatelemzési* technika használható az adatok általánosítására a felhasználói interaktivitás mellőzésével. Ezt a technikát az 5. fejezetben mutatjuk be.

Az adatkarakterizáció kimenete változatos formákban prezentálható. Példaként említhetők a különböző **kör- és oszlopdiagramok**, **görbék**, **többdimenziós adatkockák**, **táblázatok** és kereszttáblázatok. Az eredményként előálló leírások is megjeleníthetők általánosított reláció vagy szabály formában (**karakterisztikus szabályok**). Ezeket a kimeneti formátumokat, illetve azok transzformációit az 5. fejezetben tárgyaljuk.

1.4. példa | Egy adatbányászati rendszertől elvárható, hogy összegyűjtse azoknak a vásárlóknak a jellemző adatait, akik évente 1000 dollár fölötti értékben vásárolnak az *ElektroMind* áruházlánc üzleteiben. Az eredmény egy általános profillap lehet olyan információkkal, mint kitűnő hitelképességű, 40–50 év közötti alkalmazottak. A rendszernek biztosítania kell, hogy bármelyik dimenzió mentén *lefúrást* lehessen végrehajtani, például foglakozás szerinti bontásban. ■

Az **adatdiszkrimináció** a célosztályhoz tartozó adatobjektumok és egy kontrasztosztály vagy kontrasztosztályok egy halmazához tartozó adatobjektumok általános jellemzőinek összehasonlítása. A cél- és kontrasztosztályokat a felhasználó adhatja meg, a megfelelő adatobjektumok pedig lekérdezésekkel gyűjthetők össze. Például a felhasználó szeretné összehasonlítani azoknak a szoftvertermékeknek az általános jellemzőit, amelyek előző évi eladási forgalma 10%-kal nőtt, azokéval, amelyek ugyanabban a periódusban legalább 30%-kal csökkenő eladási forgalmat mutattak. Az adatdiszkriminációnál használt eljárások hasonlóak az adatkarakterizációnál használtakhoz.

Hogy jelennek meg a diszkriminatív leírások a kimeneten? A kimeneti prezentációk formája hasonló a karakterisztikus leírásokéhoz, azzal az eltéréssel, hogy a diszkriminatív leírásokban hasonlósági mértékek szükségesek, hogy megkönnyítsék a cél- és kontrasztosztályok közötti különbségtételt. A szabály formában kifejezett diszkriminatív leírásokat **diszkriminatív szabályoknak** nevezzük. A felhasználónak lehetőséget kell adni, hogy a karakterisztikus és diszkriminatív leírások kimenetén manipulálhasson.

1.5. példa | Egy adatbányászati rendszernek össze kell tudni hasonlítani az *ElektroMind* vásárlóinak két olyan csoportját, amelyek számítógéppel kapcsolatos termékek rendszeres

(havonta kettőnél több alkalommal), illetve esetenkénti (kevesebb mint évi három alkalommal) vásárlói. Az eredményül kapott leírás egy általános komparatív profillap lehet, amiből például az derül ki, hogy a számítógéppel kapcsolatos termékek gyakori vásárlóinak 80%-a 20 és 40 év közötti életkorú és egyetemi képesítéssel rendelkezik, míg az esetenkénti vásárlók 60%-a egyetemi képesítéssel nem rendelkezik, és 40 évesnél idősebb vagy 20 évesnél fiatalabb. Egy dimenzió, például a *foglalkozás* mentén *lefúrva*, vagy új dimenziót, például a *jövedelmi_szintet* bevezetve, a két osztály között még mélyebb diszkriminatív tulajdonságok feltárása lehetséges. ■

A karakterizációt és diszkriminációt tartalmazó fogalomleírásokat az 5. fejezet tárgyalja.

1.4.2. | Társításelemzés

Mi a társításelemzés? A **társításelemzés** *társítási szabályok* föltárása, amelyek egy adott adathalmazban bizonyos attribútumértékek együttes előfordulásaiból következtethetők ki. A társításelemzés széles körben használt eljárás a „bevásárlókosár" típusú elemzésben és a tranzakcióadatok elemzésében. Formálisabb megközelítésben a **társítási szabályok** $X \Rightarrow Y$ alakúak, azaz $A_1 \wedge \ldots \wedge A_m \rightarrow B_1 \wedge \ldots \wedge B_n$, ahol A_i $(i \in \{1, \ldots, m\})$ és B_j $(j \in \{1, \ldots, n\})$ attribútum–érték párok. Az $X \Rightarrow Y$ társítási szabály úgy értendő, hogy azok az adatbázis-sorok, amelyek teljesítik az X-beli feltételeket, feltehetően teljesítik az Y-belieket is.

1.6. példa | Az *ElektroMind* relációs adatbázisban az adatbányászat feltárhatja például a következő társítási szabályt:

életkor(X, "20 ... 29") ∧ *jövedelem*(X, "20 000 ... 29 000") ⇒ *vesz*(X, "*CD-lejátszót*")
[*megalapozottság* = 2%, *megbízhatóság* = 60%],

ahol X a vásárlót azonosító változó. A szabály azt fejezi ki, hogy a vizsgált vásárlók 2%-ánál teljesült az életkorra és a keresetre megfogalmazott feltétel, valamint az, hogy vásároltak CD-lejátszót (**megalapozottság**), és 60% a valószínűsége annak, hogy az életkorra és keresetre megfogalmazott feltételeknek eleget tevő vásárlók (**megbízhatóság**) fognak CD-lejátszót vásárolni.

Vegyük észre, hogy ez egynél több attribútum vagy predikátum (azaz *életkor, jövedelem* és *vesz*) közötti társítás. A többdimenziós adatbázisok terminológiáját használva, ahol az attribútumokat dimenzióknak nevezzük, a fenti szabályt **többdimenziós társítási szabálynak** is szokás nevezni.

Tegyük föl, hogy az *ElektroMind* vállalat értékesítési igazgatója szeretné meghatározni, hogy mely árucikkek kelnek el együttesen egy tranzakció keretében. Példa ilyen szabályra a

tartalmaz(T, "*számítógép*") ⇒ *tartalmaz*(T, "*szoftver*")
[*megalapozottság* = 1%, *megbízhatóság* = 50%],

vagyis ha egy tranzakció tartalmaz "*számítógép*"-et, akkor 50% az esélye annak, hogy "*szoftver*"-t is tartalmaz, és a tranzakciók 1%-a tartalmazza mind a kettőt. Ebben a társítási szabályban egyetlen ismétlődő (azaz *tartalmaz*) attribútum vagy predikátum van. Azokat a tár-

sítási szabályokat, amelyek egyetlen predikátumra hivatkoznak, **egydimenziós társítási szabályoknak** nevezzük. Megszabadulván a predikátum jelöléstől, a fenti szabály átírható az egyszerű *számítógép* ⇒ *szoftver* [1%, 50%] alakba. ■

Az utóbbi években számos algoritmust terveztek társítási szabályok hatékony bányászására. Társítási szabályok bányászatával a 6. fejezetben foglalkozunk.

1.4.3. | Osztályozás és előrejelzés

Az **osztályozás** az a folyamat, amelynek során **modellek** (vagy funkciók) olyan halmazát keressük, amelyek leírják és megkülönböztetik az adatosztályokat és fogalmakat abból a célból, hogy a modellek használatával képesek legyünk prognosztizálni, hogy egy adott objektum melyik osztályhoz tartozik. A származtatott modell **tanuló adatok** elemzésén alapszik (ezek olyan adatobjektumok, amelyek osztályának címkéje ismert).

Hogyan jelenik meg a származtatott modell? A származtatott modell számos különböző formában jeleníthető meg, például *osztályozási (IF-THEN) szabályokként*, *döntési fákként*, *matematikai formulákkal* vagy *neurális hálókkal*. A **döntési fa** egy folyamatábraszerű fastruktúra, amelyben minden csomópont egy attribútumértéken végrehajtott tesztet, minden elágazás a teszt egy kimenetét jelöli, a fa levelei pedig osztályokat, illetve osztályeloszlásokat reprezentálnak. A döntési fák könnyen alakíthatók át osztályozási szabályokká. A **neurális háló**, amikor osztályozásra használjuk, jellemzően neuronszerű feldolgozási egységek gyűjteménye, az egységek közötti súlyozott kapcsolatokkal.

Az osztályozás az adatobjektumok osztálycímkéinek előrejelzésére használható, habár számos alkalmazásban a felhasználók az osztályok címkéi helyett inkább hiányzó, nem elérhető adatértékekre szeretnének előrejelzést kapni. Rendszerint ez a helyzet akkor, amikor a megjelenő érték numerikus adat, és gyakran specifikusan **becslésként** hivatkoznak rá. Bár az előrejelzés vonatkozhat mind értékekre, mind osztálycímkékre, rendszerint adatértékek becslésére korlátozódik, és így elkülönül az osztályozástól. Az előrejelzés felöleli az elérhető adatokon alapuló eloszlási *tendenciák* azonosítását is.

Az osztályozást és az előrejelzést esetenként meg kell előznie **fontossági elemzésnek**, amely meghatározza azokat az attribútumokat, amelyek nem lényegi összetevői az osztályozási, előrejelzési folyamatnak. Az így meghatározott attribútumok azután kizárhatók.

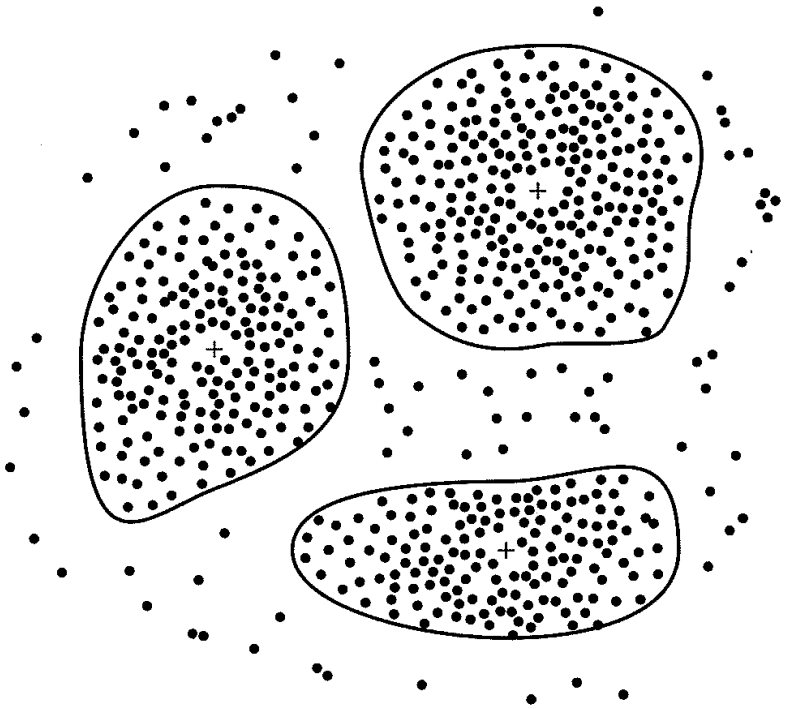
1.7. példa | Tegyük föl, hogy az *ElektroMind* kereskedelmi vezetője szeretné osztályozni az áruházlánc árucikkeiből álló terjedelmes halmazt egy reklámkampány három *(sikeres, közepes, sikertelen)* lehetséges visszajelzésére építve. A kereskedelmi igazgató mind a három osztályhoz szeretne létrehozni egy modellt az árucikkek leírható jellemzői alapján, például *ár*, *márka*, *gyártási_hely*, *típus* és *kategória*. Az eredményül kapott osztályozás maximálisan meg kell különböztesse mindegyik osztályt a többitől az adathalmaz szervezett ábrázolásával. Tegyük föl, hogy az eredményül kapott osztályozás döntési fa formájában jelenik meg. A döntési fa például az *árat* az egyetlen olyan faktorként azonosíthatja, amely legjobban tesz különbséget a három osztály között. A fa az *ár* után mutathat más jellemzőket, *márka*, *gyártási_hely*, amelyek további segítséget nyújtanak az egy osztályhoz tartozó

objektumoknak a többi osztályokhoz tartozó objektumoktól való megkülönböztetéséhez. Egy ilyen döntési fa segíthet a szóban forgó reklámkampány megértésében, illetve egy a jövőben folytatandó hatékonyabb reklámkampány tervezésében. ■

Az osztályozás és előrejelzés további részleteivel a 7. fejezet foglalkozik.

1.4.4. | Klaszterelemzés

Mi a klaszterelemzés? Ellentétben az osztályozással és előrejelzéssel, amelyek osztálycímkékkel ellátott adatobjektumokat elemeznek, a **klaszterezés** az adatobjektumokat már ismert osztálycímkék figyelembevétele nélkül elemzi. Általánosságban az osztálycímkék nincsenek jelen a kísérleti adatokban, egész egyszerűen azért, mert még nem ismertek. A klaszterezés az ilyen címkék előállítására használható. Az objektumokat azon elv alapján soroljuk klaszterekbe vagy csoportokba, hogy *maximalizáljuk az osztályon belüli objektumok hasonlóságát, és minimalizáljuk a különböző osztályokhoz tartozó objektumok közötti hasonlóságot*. Más szavakkal, objektumklasztereket formálunk úgy, hogy az egy klaszterhez tartozó objektumokat összehasonlítva, azok nagy fokú hasonlóságot, de más klaszterekhez tartozó objektumokkal összehasonlítva nagy fokú eltérést mutassanak. Minden kialakított klaszter felfogható egy-egy objektumosztálynak, amelyekből szabályok származtathatók. A klaszterezés rendszertan kialakítását, azaz, hasonló eseményeket összefogó megfigyelések hierarchikus osztályokba szervezését is megkönnyíti.



1.10. ábra | Egy kétdimenziós diagram, amely a vásárlók egy városon belüli lakóhely szerinti három klaszterét mutatja (a klaszterek centrumát + jelöli)

1.8. példa | Klaszterelemzést folytathatunk az *ElektroMind* vevőket leíró adatain olyan céllal, hogy a vevők homogén alcsoportjait határozzuk meg. Ezek a klaszterek kereskedelmi célosztályokat reprezentálhatnak. Az 1.10. ábra a vásárlók kétdimenziós klaszterdiagramját mutatja a vásárlók egy városon belüli lakóhelyeik figyelembevételével. Az ábra az adatokat reprezentáló pontok három klaszterét szemlélteti. ■

A klaszterelemzés a 8. fejezet témája.

1.4.5. | Szélsőséges értékek elemzése

Egy adatbázisban lehetnek olyan adatok, amelyek az adatmodellt vagy az adatok általános viselkedését alapul véve sehova sem sorolhatók be. Ezek az adatok (mindenen) „kívül esnek". A legtöbb adatbányászati eljárás ezeket az adatokat zajnak vagy **szélsőséges értéknek** tekinti, és eldobja azokat. Ugyanakkor néhány alkalmazásban, mint például a csalásérzékelés, a ritka események sokkal érdekesebbek, mint a szabályszerűen előfordulók. A szélsőséges értékadatok elemzését gyakran **szélsőséges értékek bányászataként** említik.

A szélsőséges értékadatok az adatok egy eloszlását vagy valószínűségi modelljét feltételezve, statisztikai tesztek végrehajtásával detektálhatók. Másik lehetőség valamilyen távolsági mérték használata, amelyben bármely klasztertől mért szignifikáns távolságra levő objektumok tekintendők szélsőséges értékeknek. A statisztikai és távolsági mértékek használata helyett alkalmazhatunk deviancia alapú eljárásokat, amelyek az egy csoporton belüli objektumok fő jellemzőiben való eltéréseket vizsgálják, és így ismerik föl a szélsőséges értékeket.

1.9. példa | A szélsőséges értékek elemzése föltárhatja hitelkártyák illetéktelen használatát azáltal, hogy egy bankszámlán szokásosan jelentkező kiadási pénzforgalomhoz képest különösen nagy volumenű kifizetést észlel ugyanazon a számlán. Szélsőséges értékek fedezhetők föl a vásárlás helyében, a vásárolt árucikkek körében és a vásárlási gyakoriságban is. ■

A szélsőséges értékek elemzésével a 8. fejezetben is találkozunk.

1.4.6. | Fejlődésanalízis

Az adatok **fejlődésanalízise** az időben változó objektumok időben változó viselkedési szabályosságait, viselkedési trendjét modellezi, írja le. Bár ezek az elemzések magukba foglalják *időkapcsolatú* adatok karakterizációját, diszkriminációját, kapcsolódását, osztályozását vagy klaszterezését, az ilyen elemzések jellegzetességeihez hozzá tartoznak még idősoradatok elemzései, sorozati, illetve periodikus mintaillesztések és hasonlóság alapú adatelemzések.

1.10. példa | Tegyük föl, hogy évekre visszamenőleg rendelkezésünkre állnak a New York-i Értéktőzsde fontosabb adatai, és szeretnénk high-tech ipari társaságok részvényeibe

befektetni. Egy adatbányászati vizsgálat feltárhat átfogó és bizonyos társaságok értékpapírjaira vonatkozó evolúciós szabályszerűségeket. Az ilyen szabályszerűségek segítségünkre lehetnek az értékpapírárak jövőbeli alakulásának előrejelzésében, hozzájárulva ezzel értékpapír-befektetési döntésnk meghozatalához. ■

A fejlődésanalízissel a 9. fejezetben foglalkozunk.

1.5. | Minden minta érdekes?

Egy adatbányászati rendszer elvben minták vagy szabályok ezreit, sőt millióit állíthatja elő. Kézenfekvő kérdés, hogy *vajon az összes előállított minta érdekes-e*. Természetesen nem. Az előállított mintáknak csak töredéke lesz bármilyen felhasználó esetében érdekes. Az adatbányászat esetében ez fölvet néhány komoly kérdést. Az olvasó elgondolkodhat azon, hogy *mitől érdekes egy minta*, *az adatbányászati rendszer elő tudja-e állítani az összes érdekes mintát*, vagy hogy *az adatbányászati rendszer képes-e arra, hogy kizárólag az érdekes mintákat állítsa elő*.

Az első kérdés megválaszolásához szükséges tisztáznunk, hogy egy minta akkor **érdekes**, ha (1) *egyszerűen érthető*, (2) *bizonyos megbízhatósággal érvényes új vagy kísérleti adatokon*, (3) *potenciálisan hasznos*, (4) *újszerű*. Egy minta akkor is érdekes, ha olyan hipotézist igazol, amelyet a felhasználó bizonyítani szeretne. Az érdekes minta **tudást** reprezentál.

Számos **objektív mérték** létezik a **minta érdekességének** megítélésére. Valamennyi a feltárt minta szerkezetén és a minta statisztikáin alapszik. Az $X \Rightarrow Y$ alakú társítási szabályok egyik objektív mértéke a szabály **megalapozottsága**, egy tranzakciós adatbázisban a szabálynak eleget tevő tranzakciók százalékos aránya. Ez valójában a $P(X \cup Y)$ valószínűség, ahol $X \cup Y$ mutatja, hogy egy tranzakció tartalmazza mind X-et, mind Y-t, azaz az X és Y részhalmazok egyesítését. A társítási szabályok másik objektív mértéke a **megbízhatóság**, amely a felfedezett társítási szabály megbízhatósági fokát jelenti. Ez pedig a $P(Y \mid X)$ valószínűség, vagyis annak a valószínűsége, hogy egy X-et tartalmazó tranzakció tartalmazza Y-t is. Formálisan a megalapozottság és a megbízhatóság definíciói az alábbiak:

$$\textit{megalapozottság}(X \Rightarrow Y) = P(X \cup Y),$$
$$\textit{megbízhatóság}(X \Rightarrow Y) = P(Y|X).$$

Általánosságban minden érdekességi mérték a felhasználó ellenőrzése alatt álló küszöbértékkel van kapcsolatban. Például azok a szabályok, amelyek mondjuk egy 50%-os megbízhatósági küszöbérték alatt vannak, érdektelennek tekinthetők. A küszöbérték alatti szabályok feltehetően zajok, szélsőséges értékek vagy jelentéktelen esetek, amelyek valószínűleg elhanyagolhatók.

Jóllehet az objektív mértékek segítségünkre vannak az érdekes minták felismerésében, ez mégsem elegendő, hacsak nem kombináljuk azokat szubjektív mértékekkel, melyek egyes felhasználói igényeket és érdekeket tükröznek. Például azok a minták, amelyek az *ElektroMind* áruházak gyakori vásárlóinak jellemzőit írják le, érdeklik az értékesítési igazgatót, de kevésbé lehet érdekes az olyan elemző számára, aki ugyanazt az adatbázist

tanulmányozza az alkalmazottak teljesítményét vizsgálva. Továbbmenve, számos minta, amely objektív szabványok által érdekes, általános tudást reprezentálhat, így valójában érdektelen. A **szubjektív érdekességi mértékek** többnyire felhasználói meggyőződéseken alapulnak. Ezek a mértékek érdekesnek találnak mintákat, ha azok **nem az elvárásnak megfelelők** (ellentmondanak a felhasználói meggyőződésnek), vagy ha azok olyan stratégiai információkat nyújtanak, amelyek ismeretében a felhasználók helyzetbe kerülnek. Ez utóbbi mintákat **helyzetbe hozó** mintáknak is nevezik. Az **elvárásnak megfelelő minták** akkor érdekesek, amikor megerősítik a felhasználót a bizonyítani kívánt hipotézisében, vagy erősítik a felhasználói gyanút.

A második kérdés az adatbányászati algoritmus **teljességére** kérdez. Az, hogy egy adatbányászati rendszer minden lehetséges érdekes mintát előállítson, sok esetben irreális és nem is hatékony. Ehelyett a felhasználók által előírt megszorítások és a megadott érdekességi mértékek használatával kell a keresést irányítani. Néhány bányászati feladat esetében, ilyenek a társítás feltárások, elegendő az algoritmus teljességének biztosítása. A társítási szabályok bányászata példa arra, hogy az előírt megszorítások és a megadott érdekességi mértékek használata biztosítja a bányászat teljességét. Az ide tartozó eljárásokat részletesen a 6. fejezetben vizsgáljuk.

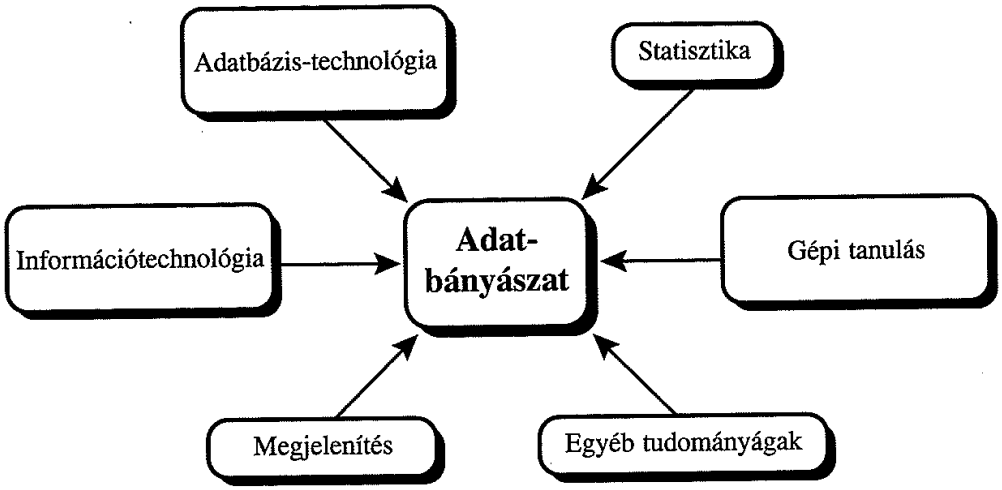
Végül, a harmadik kérdés adatbányászatbeli optimalizálási problémát feszeget. Az adatbányászati rendszerek esetében igencsak kívánatos, hogy kizárólag érdekes minták keletkezzenek. Ez annál is inkább kívánatos mind a felhasználóknak, mind magának az adatbányászati rendszernek a számára, mert akkor utólag senkinek sem kell átkutatni a megtalált mintákat, hogy megtalálja a valóban érdekeseket. Történtek már lépések ebben az irányban, az ilyen optimalizálás az adatbányászatban egyelőre mégis kihívás marad.

A mintákra vonatkozó érdekességi mértékek központi fontosságúak az adott felhasználó számára értékes minták hatékony feltárásában. Ezeknek a mértékeknek a figyelembe vétele történhet az adatbányászati lépés után, amikor is a feltárt mintákat érdekességük szerint rangsorolva, kiszűrjük a számunkra érdekteleneket. Sokkal fontosabb azonban az ilyen mértékeknek feltárási folyamatot irányító és korlátozó célú használata, amelynek során a keresés hatékonyságát a mintatér előre definiált érdekességi megszorításoknak eleget nem tevő részhalmazainak kizárásával biztosítjuk.

A minta érdekességének előírására való eljárások és az adatbányászati hatékonyságot biztosító használatuk tárgyalására a könyvben valamennyi bányászható minta esetében sort kerítünk.

1.6. | Az adatbányászati rendszerek osztályozása

Az adatbányászat interdiszciplináris terület, amely több tudományágat, például adatbázisrendszereket, statisztikát, gépi tanulást, megjelenítést és információtechnológiát fog egybe (ahogy ez az 1.11. ábráról is leolvasható). A fentieken túlmenően a használt adatbányászati szemléletmód függvényében más tudományágakban alkalmazott technikák is bevetésre kerülhetnek, például neurális hálók, fuzzy és/vagy durva halmazelmélet, tudásreprezentáció, induktív logikai programozás vagy nagy hatékonysággal végzett számítások. A bányászandó adatfajták, valamint az adatbányászati alkalmazások függvényé-



1.11. ábra | Az adatbányászat mint a különböző tudományágak összeolvadása

ben az adatbányászati rendszerek integrálhatnak a téradatbázis-elemzés, az információ ki-/visszakeresés, a mintafelismerés, a képelemzés, a jelfeldolgozás, a számítógépes grafika, a webtechnológia, a közgazdaságtan, az üzleti tevékenység, a bioinformatika vagy a pszichológia területén alkalmazott technikákat is.

Az adatbányászatban érdekelt tudományterületek szerteágazása miatt az adatbányászati kutatásoktól a legkülönbözőbb adatbányászati rendszerek nagy számban történő előállítását várják. Ezért szükséges az adatbányászati rendszerek osztályzása. Egy ilyen osztályozás segítheti a potenciális felhasználókat az adatbányászati rendszerek közötti válogatásban és az igényeikhez legjobban illeszkedő kiválasztásában. Az adatbányászati rendszerek az alábbiakban felsorolt különböző követelmények szerint osztályozhatók.

Osztályozás a bányászott *adatbázisok fajtái* szerint – Az adatbányászati rendszerek osztályozhatók a bányászott adatbázisok fajtái szerint. Maguk az adatbázisrendszerek különböző szempontok szerint (használt adatmodell, adattípusok, befoglalt alkalmazások) osztályozhatók, amelyek mindegyike saját adatbányászati technikát igényel. Az adatbányászati rendszerek ezért ennek megfelelően osztályozhatók.

Például, adatmodell szerinti osztályozást tekintve használhatunk relációs, tranzakciós, objektumorientált, objektumrelációs vagy adattárház-bányászati rendszert. A használt speciális adattípusok szerinti osztályozásban beszélhetünk térbeli, szöveges, idősor-, multimédia- vagy World Wide Web bányászati rendszerekről.

Osztályozás a bányászott *tudás fajtája* szerint – Az adatbányászati rendszerek osztályozhatók az általuk bányászott tudásfajtának megfelelően is, azaz az adatbányászati funkciók alapján, vagyis annak megfelelően, hogy a funkció karakterizáció, diszkrimináció, társítás, osztályozás, klaszterezés, szélsőséges értékek elemzése vagy fejlődésanalízis. Egy átfogó adatbányászati rendszer általában több különböző és/vagy integrált funkciót kínál.

Továbbá, az adatbányászati rendszerek megkülönböztethetők aszerint, hogy a bányászott tudás milyen finomságú és milyen szinten absztrahált. Eszerint beszélhetünk, egyebek között, általánosított (magas absztrakciós szint) tudásról, alapszintű (nyers adat szint) tudásról, és többszintű (különböző absztrakciós szintek) tudásról. Egy fejlett adatbányászati rendszer ez utóbbi feltárását kell hogy megkönnyítse.

Az adatbányászati rendszerek aszerint is osztályozhatók, hogy adatokban rejlő szabályosságokat (általánosan előforduló minták) vagy szabálytalanságokat (szélsőséges értékek) tárnak föl. Általánosságban, a fogalomleírások, a társítási elemzés, az osztályozás, az előrejelzés és a klaszterezés az adatokban rejlő szabályosságokat bányászó eljárások, amelyek a szélsőséges értékeket zajnak tekintik. Ezek az eljárások segíthetnek a szélsőséges értékek észlelésében.

Osztályozás a felhasznált *technikák fajtái* szerint – Az adatbányászati rendszerek osztályozhatók az alkalmazott adatbányászati technikák szerint. Ezek a technikák a felhasználói interakció fokával (például autonóm rendszerek, interaktív felderítő rendszerek, lekérdező rendszerek), vagy az alkalmazott adatelemzési eljárásokkal (például adatbázis vagy adattárház-orientált technikák, gépi tanulás, statisztika, megjelenítés, mintafelismerés, neurális hálók stb.) jellemezhetők. Egy kifinomult adatbányászati rendszer gyakran több adatbányászati technikát alkalmaz, vagy olyan hatékony integrált technikát valósít meg, amely jó néhány individuális megközelítés előnyeit egyesíti magában.

Osztályozás az *adaptált alkalmazások* szerint – Az adatbányászati rendszerek csoportosíthatók az általuk adaptált alkalmazások szerint is. Például lehetnek speciálisan testre szabott pénzügyi, telekommunikációs, DNS, értékpiaci, elektronikus levelezési stb. adatbányászati rendszerek. A különböző alkalmazások gyakran alkalmazásspecifikus eljárások integrációját igénylik. Ennek következtében egy általános célú adatbányászati rendszer nem feltétlenül alkalmas területspecifikus adatbányászati feladatokra.

Az 5–8. fejezetek a különböző bányászott tudásfajták szerint szerveződtek. A 9. fejezetben a különböző fejlett adatbázisrendszerek összetett adattípusainak bányászatát tárgyaljuk, a 10. fejezetben pedig néhány adatbányászati alkalmazást vizsgálunk.

1.7. | Az adatbányászat fő kérdései

A könyv, témáját tekintve, az adatbányászati módszertan, a felhasználói interakció, a hatékonyság és az adattípusok sokféleségének kérdésköreit célozza meg. Ezeket a kérdésköröket és az azon belüli kérdéseket vezetjük be a következőkben.

A bányászati módszertan és a felhasználói interakció kérdései – Ezek a bányászott tudásfajtákat, a különböző finomságú tudás bányászatának a lehetőségét, a szakterületi tudás használatát, az ad hoc bányászatot, és a tudásmegjelenítést tükrözik.

- *Különböző fajtájú tudás bányászása adatbázisokban.* Miután az egyes felhasználók különböző fajtájú tudásokban lehetnek érdekeltek, az adatbányászatnak adatelemzések és tudásfeltárási feladatok széles spektrumát kell lefednie, amely magába foglal adatkarakterizációt, diszkriminációt, társítást, osztályozást, klaszterezést, fejlődésanalízist és devianciaelemzést, illetve hasonlósági elemzéseket. Ezek a feladatok ugyanazt az adatbázist különbözőféleképpen használhatják, és számos adatbányászati technika kifejlesztését igénylik.
- *Interaktív tudásbányászat különböző absztrakciós szinteken.* Miután nem tudjuk pontosan, hogy egy adatbázisban mit tárhatunk föl, az adatbányászati folyamatnak interaktívnak kell lennie. A hatalmas adatmennyiséget tároló adatbázisokra előzetes megfelelő mintavétel-technikákat alkalmazva segíthetjük elő az adatok interaktív

feltárását. Az interaktív bányászat irányított mintakeresést, illetve kapott eredményektől függő bányászati igények megadását és finomítását biztosítják a felhasználók számára. Specifikusan a tudás bányászható kell legyen lefúrással, felgörgetéssel és pivotálással interaktívan bejárva az adat- és tudásteret hasonlóan ahhoz, ahogy ezt az OLAP az adatkockákon teszi. Ezen a módon a felhasználó az adatbányászati rendszerrel interakcióban szemlélheti az adatokat és a feltárt mintákat különböző nézetekből és különböző finomságban.

- *A háttértudás beépítése.* A tanulmányozott területre jellemző háttértudás vagy információ részben a feltárási folyamat irányításában, részben a feltárt minták tömör formában és különböző absztrakciós szinteken történő megjelenítésében használható. Az adatbázisvilággal kapcsolatos tudás, például az integritási megszorítások vagy a dedukciós szabályok segítségünkre lehetnek az adatbányászati folyamat sebességének növelésében, a feltárandó információkör kijelölésében vagy a feltárt minták érdekességének eldöntésében.
- *Adatbányászati lekérdező nyelvek és ad hoc adatbányászat.* Relációs lekérdező nyelvek (például SQL) biztosítják a felhasználóknak, hogy *ad hoc* kérdéseket fogalmazzanak meg adatkikeresési/visszakeresési céllal. Ennek mintájára magas szintű **adatbányászati lekérdező nyelvek kifejlesztése** szükséges, hogy *ad hoc* adatbányászati feladatok megfogalmazása is lehetséges legyen. Ennek célja, hogy megkönnyítsék az elemzendő adatok halmazának, a szakterületi tudásnak és a feltárt minták által kielégítendő feltételeknek és megszorításoknak specifikálását. Az ilyen nyelveket az adatbázis- vagy adattárház-lekérdező nyelvekbe kell integrálni, hogy optimalizált rugalmas adatbányászatot folytathassunk.
- *Adatbányászati eredmények láttatása és prezentációja.* A feltárt tudást magas szintű nyelveken, képi reprezentációkban vagy más formákban kell kifejezni, hogy a tudás egyszerűen érthetővé és közvetlenül használhatóvá váljék az emberek számára. Ez különösen akkor kritikus, ha az adatbányászati rendszernek interaktívnak kell lennie. A rendszernek ebben az esetben olyan szemléletes tudásreprezentációs technikákat kell alkalmaznia, mint fák, táblázatok, szabályok, gráfok, diagramok, kereszttáblázatok, mátrixok vagy grafikonok megjelenítése.
- *Zajos és nem teljes adatok kezelése.* Az adatbázisokban tárolt némely adatok zajokra, szélsőséges értékekre vagy nem teljes adatobjektumokra utalhatnak. Amikor adatszabályosságokat bányászunk, ezek az objektumok zavarólag hathatnak a folyamatra, a létrehozott tudásmodellnek az adatokon való egyfajta túlilleszkedését okozva. Ennek eredményeképpen a feltárt minták hiteltelenek vagy csak kevéssé hitelesek lehetnek. Zajokat kezelni képes adattisztítási és elemzési eljárások és olyan bányászati módszerek szükségeltetnek, melyek feltárják és elemzik a szélsőséges értékeket.
- *Mintaevolúció, az érdekesség problémája.* Egy adatbányászati rendszer minták ezreit képes feltárni. A feltárt minták közül igen sok érdektelen lehet az adott felhasználó számára vagy az újdonság hiánya miatt, vagy mert általános tudást hordoz. Rengeteg feladat maradt a feltárt minták érdekességét megállapító technikák kifejlesztését illetően, különös tekintettel az olyan szubjektív mértékekre, amelyek a minták egy adott felhasználói csoport meggyőződésére és elvárására épülő értékének megbecslését célozzák. A feltárási folyamatot irányító és a keresési tartományt leszűkítő érdekességi mértékek használata egy másik aktív kutatási terület.

A hatékonyság kérdései – Ezek az adatbányászati eljárások hatékonysági, skálázhatósági és párhuzamosítási kérdéseit foglalják magukban.

- *Adatbányászati rendszerek hatékonysága és skálázhatósága.* Az adatbányászati algoritmusoknak hatékonyaknak és skálázhatóknak kell lenniük annak érdekében, hogy eredményesen emelhessünk ki információt az adatbázisokban tárolt hatalmas mennyiségű adatból. Más szavakkal, az adatbányászati algoritmus futási ideje nagy adatbázisok esetében megjósolható és elfogadható kell legyen. Az adatbázis nézőpontjából tekintve a tudásfeltárásra, a hatékonyság és skálázhatóság kulcsfontosságú kérdés az adatbányászati rendszerek megvalósításában. A föntebb tárgyalt bányászati módszertan és felhasználói interakció fő kérdéseinek számos esetében szintén figyelembe kell venni a hatékonyságot és a skálázhatóságot.
- *Párhuzamos, osztott és inkrementális bányászati algoritmusok.* Számos adatbázis nagy mérete, az adatok osztott tárolása és egyes adatbányászati eljárások kiszámítási bonyolultsága olyan tényezők, amelyek indokolttá tették **párhuzamos és osztott adatbányászati algoritmusok** kifejlesztését. Ezek az algoritmusok az adatokat partíciókra osztják, a partíciók feldolgozása párhuzamosan történik, majd az eredmény a partíciókon képződött részeredmények összerendezésével adódik. Emellett néhány adatbányászati folyamat magas költsége felvetette az inkrementális adatbányászati algoritmusok szükségességét, amelyek a folyamat közben végrehajtott adatbázis-módosításokat beépítik folyamatba, szükségtelenné téve ezzel a teljes adatállomány újrabányászását. Az ilyen algoritmusok inkrementális tudásmódosításokat hajtanak végre, hogy kiegészítsék és megerősítsék a korábbi feltárásokat.

Az adatbázistípusok sokféleségének kérdései

- *Relációs és összetett adattípusok.* Miután a relációs adatbázisok és adattárházak széles körben használt adattárak, az ilyen adatokon használható adatbányászati rendszerek kifejlesztése fontos. Mindamellett, más adatbázisok összetett adattípusokhoz tartozó objektumokat, hipertextet és multimédia-adatokat, tér-, idő- és tranzakcióadatokat tárolhatnak Figyelembe véve az adattípusok és adatbányászati célok különbözőségét, nem várható el, hogy valamennyi lehetséges adat bányászata egyetlen rendszerrel történjék. Specifikus adatok bányászásához specifikus adatbányászati rendszereket kell létrehozni. Ennek megfelelően az várható el, hogy a különböző adattípusokhoz különböző adatbányászati rendszerek léteznek.
- *Információ bányászása heterogén adatbázisokból és globális információs rendszerekből.* A lokális és nagy távolságú számítógépes hálózatok (például internet) sok adatforrást kapcsolnak össze, megvalósítván egy hatalmas, osztottan tárolt és heterogén adatbázist. A különböző forrásokból elérhető, strukturált, félig strukturált vagy nem strukturált, eltérő szemantikájú adatokból történő tudásfeltárás nagy kihívás az adatbányászat számára. Az adatbányászat segítségünkre lehet olyan magas szintű adatszabályosságoknak különböző heterogén adatbázisokból történő feltárásában, amelyek felfedése egyszerű lekérdező rendszerekkel valószínűtlen, továbbá javítja az adatcserét és együttműködési képességet heterogén adatbázisok között. A webes bányászat, amely érdekes tudást tár föl a webtartalmakról, a webhasználatról és a webdinamikáról, az adatbányászatnak igen kihívó és hatalmas mértékben fejlődő területe.

A fentiekben bemutatott kérdéseket tekintik az adatbányászati technológia további fejlődésében alapvető követelményeknek és kihívásoknak. A kihívások közül néhánnyal az újabb keletű adatbányászati kutatásokban és fejlesztésekben bizonyos mértékig már foglalkoztak, és most mint követelmények szerepelnek, míg mások még mindig a kutatási szakaszban vannak. A kérdések mindemellett folyamatosan további vizsgálatokra és tökéletesítésre sarkallnak. További, alkalmazásokkal és titkosítással kapcsolatos, illetve az adatbányászat társadalomra gyakorolt hatását vizsgáló kérdések tárgyalására a könyv befejező, 10. fejezetében kerül sor.

1.8. | Összefoglalás

- Az **adatbázis-technológia** az egyszerű fájlfeldolgozástól a lekérdezéseket és tranzakció-kezelést kínáló adatbázisrendszerek kifejlesztéséhez jutott el. A technológia szakadatlan fejlődése a legkülönbözőbb alkalmazási ágak, üzleti élet és vezetés, államigazgatás, tudomány és technika, környezetvédelem tárgykörébe tartozó adatok mennyiségének robbanásszerű növekedését eredményezte, amely hatékony és eredményes adatelemző és az adatokat értelmező eszközök kifejlesztésének egyre növekvő igényéhez vezetett.
- Az **adatbányászat** az adatbázisokban, adattárházakban vagy egyéb információtározókban tárolt nagy mennyiségű adatból tár fel érdekes mintákat. Egy fiatal interdiszciplináris terület, amely az adatbázisrendszerek, adattárház-kezelés, statisztika, gépi tanulás, adatmegjelenítés, információ-visszakeresés és nagy hatékonyságú számítások területeit öleli föl. További társterületek még a neurális hálók, a mintafelismerés, a téradatelemzés, a képi adatbázisok, a jelfeldolgozás és további alkalmazások, például az üzleti élet, a gazdaság és a bioinformatika.
- A **tudásfeltárás folyamata** magába foglalja az adatok tisztítását, integrációját, kiválasztását és transzformációját, adatbányászatot, mintakiértékelést és a tudás reprezentálását.
- **Adatminták** bányászhatók különféle **adatbázisokból**, ami lehet például relációs adatbázis, adattárház, tranzakciós adatbázis, objektumrelációs és objektumorientált adatbázis. Érdekes adatminták más típusú információtárakból, például térbeli, időbeli, szöveges, multimédia- és örökölt adatbázisok, valamint a World Wide Web, is kiemelhetők.
- Egy **adattárház** a sok különböző forrásból származó, vezetői döntéshozatalt elősegítő szervezettségű adatok hosszú távú tárolására szolgáló tározó. A tipikusan összesített adatok egy egységes séma szerint tárolódnak. Az adattárházrendszerek bizonyos adatelemzési lehetőségeket kínálnak, amelyekre együttesen az **OLAP** névvel hivatkoznak.
- Az **adatbányászati funkciók** magukba foglalják a fogalom-/osztályleírások meghatározását, társítást, osztályozást, előrejelzést, klaszterezést, trendelemzést, deviancia- és hasonlóságelemzést. A karakterizáció és diszkrimináció adatösszesítési formák.
- Egy minta **tudást** reprezentál, ha az emberi elme számára világosan érthető, a tesztadatokon valamilyen fokú bizonyossággal érvényes, hasznosnak ítélhető, újszerű, vagy alátámaszt egy a felhasználó által megfogalmazott gyanút. A mintaérdekességi mértékek, akár *objektívak*, akár *szubjektívak*, a feltárási folyamat irányítására használhatók.

- Az **adatbányászati rendszerek osztályozhatók** a bányászott adatbázisok típusai szerint, a feltárt tudás fajtája szerint, valamint a használt technikák és adaptált alkalmazások szerint.
- A nagy adatbázisokban folytatott hatékony és eredményes adatbányászat számos követelményt és nagy kihívást fogalmaz meg a kutatók és fejlesztők számára. Az érintett kérdéskörök magukba foglalják a bányászati módszertant, a felhasználói interakciót, a hatékonyságot és skálázhatóságot, az adattípusok nagy választékát, valamint ide sorolhatók még az adatbányászati alkalmazások és az adatbányászat társadalomra gyakorolt hatását vizsgáló kérdések.

1.9. | Feladatok

1.1. Mi az *adatbányászat*? Válaszolja meg, illetve fejtse ki a következőket:
(a) Egy új léggömb?
(b) Az adatbázisokból, a statisztikából és a gépi tanulásból egyszerű változtatásokkal kialakított technológia?
(c) Fejtse ki, hogy az adatbázis-technológia fejlődése hogyan vezetett el az adatbányászathoz!
(d) Sorolja fel az adatbányászati lépéseket, ha az adatbányászatot tudásfeltárási folyamatnak tekinti!

1.2. Mutassa be egy példán, hogy az adatbányászat döntő szerepet játszik egy üzleti vállalkozás sikerében! Milyen *adatbányászati funkciók* szükségesek a példabeli esetben? Végrehajthatók az említett funkciók adatlekérdezésekkel vagy egyszerű statisztikai elemzésekkel?

1.3. Tételezzük föl, hogy Ön a *Csúcs-Egyetem* szoftvermérnökeként azt a feladatot kapja, hogy tervezzen egy adatbányászati rendszert, amely az egyetemi kurzusokat tároló adatbázist vizsgálja. Az adatbázis a következő információkat tárolja. A hallgató neve, címe és státusa (például egyetemi hallgató, diplomás), a választott kurzus, összesített osztályzati átlag. Mutassa be az ön által javasolt *architektúrát*! Mi a szerepük a javasolt architektúra egyes komponenseinek?

1.4. Miben különbözik, és miben egyezik egy *adattárház* és egy adatbázis?

1.5. Adjon rövid jellemzést a következő *fejlett adatbázisrendszerekről* és alkalmazásokról: objektumorientált adatbázisok, téradatbázisok, szöveges adatbázisok, multimédia-adatbázisok, a World Wide Web!

1.6. Határozza meg egyenként a következő *adatbányászati funkciókat*: karakterizáció, diszkrimináció, társítás, osztályozás, előrejelzés, klaszterezés és fejlődésanalízis! Adjon példát mindegyik adatbányászati funkcionalitásra egy ön által ismert létező adatbázist használva!

1.7. Mi a különbség a karakterizáció és a diszkrimináció, a karakterizáció és a klaszterezés, az osztályozás és az előrejelzés között? A felsorolt feladatpárok miben hasonlítanak?

1.8. Személyes megfigyelése alapján adjon meg egy ebben a fejezetben nem említett lehetséges tudásfajtát, amelynek feltárásához adatbányászati eljárások szükségesek! A megadott tudásfajta igényel-e olyan bányászati metodikát, amely különbözik az ebben a fejezetben vázoltaktól?

1.9. Fogalmazzon meg három adatbányászati feladatot, az *adatbányászati módszertant* és a *felhasználói interakciót* illetően!

1.10. Fogalmazzon meg két adatbányászati feladatot, a *hatékonyságot* illetően.

1.10. | Irodalom

A *Knowledge Discovery in Databases* c. könyv Piatetsky-Shapiro és Frawly szerkesztésében [PSF91] az adatbázisokban folytatott tudásfeltárás témakörben megjelent tudományos cikkek korai gyűjteménye. A Fayyad, Piatetsky-Shapiro, Smyth és Uthurusamy [FPS+96] szerkesztette *Advances in Knowledge Discovery and Data Mining* c. könyv a tudásfeltárás és adatbányászat témakörben folytatott újabb keletű kutatási eredmények figyelemre méltó gyűjteménye. További adatbányászatot tárgyaló könyvek Weiss, Indurkhya: *Predictive Data Mining* [WI98], Michalski, Brakto, Kubat: *Machine Learning and Data Mining: Methods and Application* [MBK98], Westphal, Blaxton: *Tools for Solving Real-World Problems* [WB98], Berry, Linoff: *Mastering Data Mining: The Art and Science of Customer Relationship Management* [BL99], Berson, Smith, Thearling: *Building Data Mining Applications for CRM* [BST99] és Groth: *Data Mining: Building Competitive Advantage* [Gro99]. További, cikkek gyűjteményét tartalmazó kiadványok még a Ziarko szerkesztette *Rough Sets, Fuzy Sets and Knowledge Discovery* [Zia94], és számos az adatbányászatba bevezető oktató segédanyag, mint az ACM Press gondozásában megjelent *Tutorial Notes of the 1999 International Conference on Knowledge Discovery and Data Mining (KDD'99)*.

A *KDD Nuggets* egy rendszeres szabadon elérhető elektronikus hírlevél a tudásfeltárásról és adatbányászatról szóló információval. A témakörhöz tartozó információk elektronikus levélben téma és URL megjelöléssel elküldhetők az *editor@kdnuggets.com* címre. Az előfizetőkről szóló információk olvashatók a *http://www.kdnuggets.com./news/subscribe.html* linken. A *KDD Nuggets* felügyeletét 1991 óta Piatetsky-Shapiro látja el. A *http://www.kdnuggets.com/* linken elérhető *Knowledge Discovery Mine* internetoldal KDD-vel kapcsolatos információk jó gyűjteménye. Az adatbányászok első nemzetközi tudásfeltárási és adatbányászati konferenciájukat 1995-ben tartották [FU96]. A konferencia az 1989–1994 között a tudásfeltárás adatbázisokban téma köré szerveződő négy nemzetközi workshop [PS89, PS91a, FUP93, FU94] nyomán került megrendezésre. Az adatbányász kutatók 1998-ban megalakították az ACM-SIGKDD nevű (Special Interested Group on Knowledge Discovery in Databases under ACM) tudományos szervezetüket. Az ACM-SIGKDD 1999-ben rendezte meg a tudásfeltárás és adatbányászat ötödik nemzetközi konferenciát (KDD'99). A témával foglalkozó *Data Mining and Knowledge Discovery* c. folyóirat 1997 óta van jelen a Kluwers Kiadó gondozásában. Az ACM-SIGKDD negyedévenként megjelenő *SIGKDD Explorations* hírlevele csak a SIGKDD-tagok számára elérhető kiadvány. Meg kell emlékeznünk még néhány (regionálisan) nemzetközi adatbányászati konferenciáról, mint például a *Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD)*, a *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, és az *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*.

A téma iránt érdeklődő olvasó kedvére tallózhat az alább felsorolt adatbányászati ku-

tatásokról szóló könyvek, konferenciakiadványok, adatbázisokkal, statisztikával, gépi tanulással és adatmegjelenítéssel foglalkozó tudományos folyóiratok között.

Népszerű, adatbázisokkal foglalkozó tankönyvek között említhetjük Ullman: *Principles of Database and Knowledge-Base Systems, Vol. 1* [Ull88]; Elmasri, Navathe: *Fundamentals of Database Systems, 2nd ed.* [EN94]; Silberschatz, Korth, Sudarshan: *Database System Concepts, 3rd ed.* [SKS97]; Ullman, Widom: *A First Course in Database Systems* [UW97]; illetve Ramakrishnan, Gehrke: *Database Management System, 2nd ed.* [RG00] c. munkákat. Említésre méltó a *Readings in Database Systems,* egy adatbázisokról szóló cikkek szerkesztett gyűjteménye [SH98] Stonebraker és Hellerstein szerkesztésében. Adatbázis-kutatási feladatokról és kutatási eredményekről szóló áttekintő tanulmányok találhatók [SAD+93]-ban Stonebraker, Agrawal, Dayal és más szerzők tollából, illetve [SSU96]-ban Silberschatztól, Stonebrakertől és Ullmantól.

Az elmúlt néhány évben publikált könyvek közül meg kell említenünk Kimball: *The Data Warehouse Toolkit* [Kim96], Inmon: *Building the Data Warehouse* [Inm96] és Thomsen: *OLAP solutions: Building Multidimensional Information Systems* [Tho97]. Chaudhuri és Dayal [CD97]-ben az adattárház technológiáról nyújt részletes áttekintést.

Adatbányászati és adattárházzal kapcsolatos kutatási eredményekről számol be számos nemzetközi adatbázis-konferencia kiadványa. Ilyenek egyebek között az *ACM-SIGMOD International Conference on Management of Data (SIGMOD),* az *International Conference on Very Large Databases (VLDB),* az *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (POD),* az *International Conference on Data Engineering (ICDE),* az *International Conference on Extending Database Technology (EDBT),* az *International Conference on Database Therory (ICDT),* az *International Conference on Information and Knowledge Management (CIKM),* az *International Conference on Database and Expert Systems Application (DEXA)* és az *International Symposium on DataBase Systems for Advanced Applications (DASFAA).* Jelentősebb adatbázis folyóiratok adatbányászati kutatásokról is beszámolnak. Ilyenek például, az *IEEE Transactions on Knowledge and Data Engineering (TDKE), ACM Transactions on Database Systems (TODS), Journal of ACM (JACM), Information Systems, The VLDB Journal, Data and Knowledge, Engineering* és *International Journal of Intelligent Information Systems (JIIS).*

Szót kell ejtenünk néhány, a statisztikai analízis különböző témaköreivel foglalkozó tankönyvről, például Devore: *Probability and Statistics for Engineering and the Sciences 4th ed.* [Dev95]; Neter, Kutner, Nachtsheim, Wasserman: *Applied Linear Statistical Models 4th ed.* [NKNW96], Dobson: *An Introduction to Generalized Linear Models* [Dob90]; Shumway: *Applied Statistical Time Series Analysis* [Shu88] és Johnson, Wichern: *Applied Multivariate Statistical Analysis 3rd ed.* [JW92].

Statisztikai kutatásokról fontosabb statisztikai konferenciakiadványok is beszámolnak. Így megemlíthetjük a *Joint Statistical Meetings, International Conference of the Royal Statistical Society* és *Symposium on the Interface: Computing Science and Statistics.* Más publikációs források a *Journal of the Royal Statistical Society,* a *The Annals of Statistics,* a *Journal of American Statistical Association,* a *Technometrics* és a *Biometrika.*

A gépi tanulásról szóló ismeretterjesztő kiadványok és tankönyvek között figyelemre méltók a *Machine Learning, An Artificial Intelligence Approach Vols. 1–4.* Michalski és mások szerkesztésében [MICM83, MCM86, KM90, MT94], Quinlan: *C4.5: Programs for Machine Learning* [Qui93], Langley: *Elements of Machine Learning* [Lan96] és Mitchel:

Machine Learning [Mit97] c. könyvek. A Weiss és Kulikowski szerzőpáros *Computer System That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems* c. könyve a különböző területek osztályozási és előrejelzési eljárásait hasonlítja össze. A gépi tanulásról szóló érdekes cikkek szerkesztett gyűjteménye Shalvik, Dietterich: *Readings in Machine Learning* c. kiadványa [SD90].

A gépi tanulás témakörben folyó kutatásokról a gépi tanulással foglakozó és a mesterséges intelligencia témakörben tartott konferenciák kiadványai is tájékoztatnak. Példaképpen megemlítjük az *International Conference on Machine Learning (ML)*, az *ACM Conference on Computational Learning Theory (COLT)*, az *International Joint Conference on Artificial Intelligence (IJCAI)* és az *American Association of Artificial Intelligence Conference (AAAI)*. A témához kapcsolódó publikációk olvashatók még néhány már fentebb idézett folyóiratban, továbbá a *Machine Learning (ML)*, az *Artificial Intelligence Journal (AI)* és a *Cognitive Science* közleményekben. Egy a statisztikus mintafelismerés perspektívájával foglalkozó áttekintő osztályozás olvasható Duda és Hart munkájában [DH73].

Az adatmegjelenítési technikákat bemutató úttörő munka Tufte: *The Visual Display of Quantitative Information* [Tuf83], *Envisioning Information* [Tuf90] és Bertin: *Graphics and Graphic Information Processing* [Ber81] c. munkái. Keim: *Visual Techniques for Exploring Databases* [Kei97] c. könyve az adatbányászati megjelenítésben átfogó oktatási anyag. A megjelenítés témakörében rendezett jelentősebb szimpóziumok és konferenciák az *ACM Human Factors in Computing Systems (CHI)*, a *Visualization* és az *International Symposium on Information Visualization*. A megjelenítéssel kapcsolatos kutatásokról információk olvashatók még a *Transaction on Visualization and Computer Graphics*, a *Journal of Computational and Graphical Statistics* és az *IEEE Computer Graphics and Applications* kiadványokban.

2. FEJEZET

Adattárház és OLAP-technológia

Az adattárházak létrehozását, amely magába foglalja az adattisztítás és az adategyesítés folyamatát is, az adatbányászat fontos előfeldolgozási lépésének is tekinthetjük. Emellett az adattárházak **on-line analitikus feldolgozó (On-Line Analytical Processing, OLAP)** eszközként is szolgálnak a többdimenziós, változó finomságú adatok interaktív elemzéséhez, amelyek elősegítik a hatékony adatbányászatot. Ráadásul sok más adatbányászati függvény, például osztályozás, előrejelzés, társítás és klaszterezés is beépíthető az OLAP-műveletekbe, amely több absztrakciós szint jelenléte esetén javítja a tudásbázis bányászatát. Ezért az adattárház az adatelemzés és az on-line analitikus feldolgozás terén egyre fontosabbá vált, és az adatbányászatnak is hatékony felületet biztosít. Éppen ezért, mielőtt a könyv további fejezeteiben részletesen megismerkednénk az adatbányászat technológiáival, ezt a fejezetet az adattárházzal kapcsolatos technológiák áttekintésének szenteljük. Erre az adatbányászati technológiák megértéséhez mindenképpen szükség van.

Ebben a fejezetben megismerkedünk az alapfogalmakkal, az általános architektúrákkal, az adattárház-kezelésben és az OLAP-technológiákban alkalmazott fő implementációs technikákkal, és mindezeknek az adatbányászathoz fűződő kapcsolatával.

2.1. | Mi az adattárház?

Az adattárház olyan architektúrákat és eszközöket biztosít az üzletemberek számára, amelyek segítségével megfelelően rendszerezhetik, értelmezhetik és használhatják az adataikat a stratégiai döntések meghozatalához. A különféle szervezetek nagy része gondolja úgy, hogy az adattárházrendszer a mai gyorsan változó, versenyorientált világban értékes eszköz. Az utóbbi néhány évben sok vállalat költött dollármilliókat az egész vállalkozást átfogó adattárház építésére. Sokan érzik úgy, hogy az adattárház kezelése a legújabb „kötelező" marketingfegyver a minden területen érzékelhető versenynövekedés pillanataiban – fontos módszer a vásárlók megtartására azáltal, hogy többet tudunk az igényeikről.

Nos, jön az intrikus kérdés, *pontosan mi is az az adattárház?* Az adattárház fogalmát többféleképpen is meghatározták már, ezért nehéz most pontos definícióval előállni. Az adattárház nagy vonalakban egy adatbázis, amelyet az adott szervezet operatív adatbázisától elkülönülten tartanak fenn. Az adattárházrendszer lehetővé teszi különböző felhasználói rendszerek integrációját. Az információfeldolgozást is támogatja az elemzés céljára felhasználható, egyesített, történeti adatok biztosításával.

Az adattárházrendszerek építésében vezető szakember, W. H. Inmon szerint „Az adattárház a menedzsment szintű döntéshozás folyamatát támogató témaorientált, integrált, időben változó, nem felejtő adatgyűjtemény." [Inm96] Ez a rövid, de velős meghatározás felsorakoztatja az adattárház fő jellemzőit. A négy kulcsszó: *témaorientált, integrált, időben változó, nem felejtő,* különbözteti meg az adattárházat más adattároló rendszerektől, mint például a relációs adatbázis-, tranzakció-kezelő vagy a fájlkezelő rendszerektől. Vizsgáljuk meg közelebbről ezeket a kulcsfontosságú jellemzőket!

- **Témaorientált** – Az adattárház főbb témák köré van szervezve, például vevő, szállító, áru vagy eladás. A szervezet napi műveleteinek és tranzakcióinak feldolgozása helyett az adattárház az adatok modellezésére és elemzésére koncentrál és így segíti a döntéshozást. Ezért az adattárház a döntési folyamat során nem használható adatok kizárásával jellemzően egyszerű és tömör képet ad az adott témáról és a hozzá kapcsolódó kérdésekről.
- **Integrált** – Az adattárház általában több heterogén forrás, például relációs adatbázisok, közönséges fájlok, on-line tranzakciós feljegyzések egyesítésével jön létre. Adattisztító és adategyesítő technikákat alkalmaznak annak érdekében, hogy konzisztens legyen a mezők elnevezése, a struktúrák kódolása, a használt mértékek és így tovább.
- **Időben változó** – Az adatokat azért tároljuk, hogy történeti távlatból (például az elmúlt 5–10 év alapján) nyújtsanak információt. Az adattárházban minden kulcsfontosságú struktúra közvetve vagy közvetlenül tartalmaz valamilyen időelemet.
- **Nem felejtő** – Az adattárház mindig fizikailag elkülönített tár, amelyben az adatok az operatív környezetben található alkalmazási adatok transzformáltjai. Az elkülönítés miatt az adattárház nem igényel tranzakció-kezelő, helyreállító és konkurenciakezelő mechanizmusokat. Általában csak két hozzáférési műveletre van szükség: *kezdeti adatfeltöltésre* és *adatelérésre*.

Összegezve, az adattárház szemantikusan konzisztens adattár, amely egy döntéstámogató adatmodell fizikai megvalósításaként szolgál, és azokat az információkat tartalmazza, amelyekkel kapcsolatban a vállalatnak stratégiai döntéseket kell hoznia. Az adattárházat gyakran olyan architektúrának is szokták tekinteni, amely több heterogén forrásból vett adat egyesítéseként jön létre, hogy támogassa a strukturált és/vagy *ad hoc* lekérdezéseket, elemző beszámolókat és a döntéshozatalt.

Jó, jön a következő kérdés, *és akkor mi az adattárház-kezelés?* A fentiek alapján az *adattárház felépítésének és használatának a folyamatát* tekintjük adattárház-kezelésnek. Az adattárház felépítése adategyesítést, adattisztítást és adatvéglegesítést igényel. Az adattárház hatékony felhasználásához gyakran van szükség *döntéstámogató* technológiákra is. Ez teszi lehetővé a „tudásmunkásoknak" (igazgatóknak, elemzőknek, vezetőknek), hogy az adattárházat felhasználva gyorsan és kényelmesen szerezzenek áttekintést az adatokról, és hogy alaposan megfontolt döntéseket hozzanak az adattárházban tárolt információk alapján. Néhány szerző adattárház-kezelés alatt csak az adattárház *megépítésének* a folyamatát érti, és az „adattárház ABKR"[1] kifejezést használja az adattárház *kezelésére és felhasználására*. Mi ebben a könyvben nem tesszük meg ezt a megkülönböztetést.

1 | ABKR = adatbázis-kezelő rendszer (angolul: database managing System, DBMS).

Hogyan használják fel a szervezetek az adattárházakból nyert információt? Sokan közülük ezt az információt az üzleti döntéshozó tevékenységek támogatására használják. A döntéshozó tevékenységek közé tartoznak a következők: (1) a vásárlói szokások figyelése, amely magába foglalja a fogyasztási minták elemzését (például vásárlási preferencia, vásárlással töltött idő, a családi kassza változásai, költekezési hullámok); (2) árucikkek áthelyezése és a termékportfólió kezelése az értékesítési teljesítmények negyedéves, éves és földrajzi elhelyezkedés szerinti összehasonlítása alapján, hogy a termelési stratégiákat pontosan be lehessen állítani; (3) műveletek elemzése és profitlehetőségek keresése; (4) az ügyfélkapcsolatok kezelése a környezeti tényezők figyelembevételével, és a vállalati vagyon költségeinek a kezelése.

Az adattárház építése a *heterogén adatbázisok egyesítése* szempontjából is nagyon hasznos. Sok szervezet jellemzően sokféle adatot gyűjt, és nagy adatbázisokat tart fenn többszörös, heterogén, önálló és megosztott információforrásokból. Ilyen adatokat egyesíteni és azok könnyű, hatékony elérését biztosítani igen kívánatos, de kihívó feladat. Az adatbázisiparban és -kutatásban komoly erőfeszítéseket tettek e cél elérése érdekében.

A heterogén adatbázisok egyesítésének hagyományos megoldása a **becsomagolók** és **egyesítők** (vagy **közvetítők**) (wrappers, integrators, mediators) építése a többszörös, heterogén adatbázisok fölé (példaként említhető az IBM Data Joiner és az Informix DataBlade). Amikor a kliens oldalnak végre kell hajtania egy lekérdezést, akkor ezt egy metaadatszótár felhasználásával olyan lekérdezésekre fordítja, bontja le, amelyek már megfelelőek lesznek az érintett, heterogén munkaállomások számára. Ezeket a lekérdezéseket aztán leképezik és elküldik a helyi lekérdezés-feldolgozóknak. A munkaállomásokról visszaküldött eredmények egyesítésével áll elő a lekérdezés eredménye. Ez a **lekérdezésvezérelt megközelítés** bonyolult információszűrést és egyesítő eljárásokat igényel, továbbá a helyi adatok feldolgozásának jelent konkurenciát az erőforrások megosztásánál. Gyakori lekérdezések esetén nem hatékony és várhatóan költséges, különösen ha a lekérdezésekben összesítőfüggvények is előfordulnak.

Az adattárházak érdekes alternatívát kínálnak a heterogén adatbázisok egyesítésének fent leírt hagyományos megoldásával szemben. A lekérdezésvezérelt megközelítés helyett az adattárházak inkább **frissítésvezérelt** megközelítést alkalmaznak, amelyben a többszörös, heterogén forrásokból nyert információt már előre egyesítik és egy tárházban tárolják a közvetlen lekérdezések és elemzések végrehajtásához. Az adattárházak, szemben az on-line tranzakció-feldolgozó adatbázisokkal, nem a legfrissebb információkat tartalmazzák. Mégis magas teljesítményt nyújtanak egy integrált heterogén adatbázisrendszerben, mivel az adatok másolás, előfeldolgozás, egyesítés, magyarázatokkal való ellátás, összegzés és átstrukturálás után egyetlen szemantikus adattárban találhatók meg. Ezenkívül az adattárházas lekérdezések feldolgozása sem zavarja a helyi forrásokban zajló feldolgozást. Ráadásul az adattárház történeti információkat is tárolhat és egyesíthet, és összetett, többdimenziós lekérdezéseket is képes támogatni. Ezek eredményeképpen az adattárház-kezelés nagyon népszerűvé vált az adatbázisiparban.

2.1.1. | Különbségek az operatív adatbázisrendszerek és az adattárházak között

Mivel a legtöbb ember ismeri a forgalomban lévő relációs adatbázisrendszereket, a legkönnyebb úgy megérteni az adattárház lényegét, ha összehasonlítjuk ezt a két rendszert.

Az on-line operatív adatbázisrendszerek fő feladata, hogy végrehajtsák az on-line tranzakciókat és feldolgozzák a lekérdezéseket. Az ilyen rendszereket **on-line tranzakció-feldolgozó (On-Line Transaction Processing, OLTP)** rendszereknek hívjuk. Ezek lefedik egy szervezet legtöbb naponta használt műveletét, például a beszerzést, a raktározást, a gyártást, a bankügyleteket, a bérszámfejtést, a nyilvántartást és a könyvelést. Az adattárházrendszerek ezzel szemben a felhasználókat, illetve a tudásmunkásokat az adatelemzés és a döntéshozás feladatában segítik. Ezek a rendszerek az eltérő felhasználói igényeknek megfelelően sokféle formában képesek rendszerezni és bemutatni az adatokat. Ezeket **on-line analitikus adatfeldolgozó (OLAP)** rendszereknek nevezzük.

Az OLTP- és az OLAP-rendszerek fő megkülönböztető jellemzői a következőkben foglalhatók össze:

- **Felhasználó- és rendszerorientált** – Az OLTP-rendszer *ügyfélorientált,* és az ügyintézők, kliensek, illetve információtechnológiai szakemberek használják a tranzakciók és a lekérdezések feldolgozása céljára. Az OLAP-rendszer *piacorientált,* és a tudásmunkások, beleértve az igazgatókat, vezetőket és elemzőket, használják adatelemzés céljából.
- **Adattartalom** – Az OLTP-rendszer naprakész adatokat kezel, amelyek jellemzően túl részletesek ahhoz, hogy a döntéshozásban könnyen használhatók legyenek. Az OLAP-rendszer nagy mennyiségű történeti adatot kezel, lehetőséget nyújt az összefoglalásra és az összesítésre, és az információkat különböző finomsági szinteken tárolja és kezeli. Ezek a jellemzők megkönnyítik, hogy az adatokat körültekintő döntéshozatal során használják.
- **Adatbázis-szerkezet** – Az OLTP-rendszer általában az egyed–kapcsolat (EK) adatmodellt és alkalmazásorientált adatbázis-szerkezetet használ. Egy OLAP-rendszer jellemzően a *csillag* vagy a *hópehely* adatmodellt (bővebben a 2.2.2. alfejezetben) és témaorientált adatbázis-szerkezetet alkalmaz.
- **Nézőpont** – Az OLTP-rendszer főleg a vállalaton vagy részlegen belüli aktuális adatokra koncentrál, történeti adatokra vagy más szervezetek adataira való utalás nélkül. Ezzel szemben az OLAP-rendszer gyakran átível az adatbázisséma több verzióján is, a szervezet fejlődési folyamatát követve. Az OLAP-rendszerek olyan információkkal is dolgoznak, amelyek más szervezetektől származnak, egyesítve az információkat, amelyek sok különböző adatforrásból származnak. Az OLAP-adatokat hatalmas mennyiségük miatt több adathordozón tárolják.
- **Hozzáférési minták** – Az OLTP-rendszerhez való hozzáféréseket főleg rövid, atomi tranzakciók jelentik. Egy ilyen rendszer konkurenciavezérlő és helyreállító mechanizmusokat igényel. Az OLAP-rendszerekhez való hozzáféréseket azonban zömében csak olvasó (read-only) műveletek jelentik (mivel a legtöbb adattárház inkább történeti, mint naprakész információkat tárol), bár ezek között sok összetett lekérdezés is előfordulhat.

További jellemzők, amelyek az OLTP- és OLAP-rendszereket egymástól megkülönböztetik, az adatbázis mérete, a műveletek gyakorisága, illetve hogy mi alapján mérjük a teljesítményt. A 2.1. táblázat ad erről összefoglalást.

2.1. táblázat | Az OLTP- és OLAP-rendszerek összehasonlítása

Tulajdonság	OLTP	OLAP
Jellegzetesség	Operatív feldolgozás	Információfeldolgozás
Orientáció	Tranzakció	Elemzés
Felhasználó	Ügyintéző, adatbázis-rendszergazda, adatbázis-szakember	Tudásmunkás (például menedzser, igazgató, elemző)
Funkció	Napi műveletek	Hosszú távú információs igények, döntéstámogatás
Adatbázisséma	EK alapú, alkalmazásorientált	Csillag/hópehely, témaorientált
Adatok	Aktuális; garantáltan a legfrissebb	Történeti; folyamatos pontosítás
Összegzés szintje	Primitív, nagyon részletes	Összegzett, egyesített
Adatok felbontása	Részletes, közönséges relációs	Összegzett, többdimenziós
Munkaegység	Rövid, egyszerű tranzakció	Komplex lekérdezés
Elérés	Olvasó/író	Főleg olvasó
Fókusz	Adatok begyűjtése	Információ kiadása
Műveletek	Index/tördelés az elsődleges kulcs szerint	Sok vizsgálat
Az elért rekordok száma	Tízes nagyságrend	Milliós nagyságrend
A felhasználók száma	Ezres nagyságrend	Százas nagyságrend
Aatbázis mérete	100 MB-tól GB-ig	100 GB-tól TB-ig
Prioritás	Magas teljesítmény és elérhetőség	Nagy rugalmasság, végfelhasználói önállóság
A teljesítmény mértéke	Tranzakciós teljesítmény	Lekérdezés végrehajtásának hatékonysága, válaszidő

Megjegyzés: A táblázatot részben [CD97] alapján állítottuk össze.

2.1.2. | Miért legyen mégis külön adattárház?

Az operatív adatbázisok óriási mennyiségű adatot tárolnak – jegyzi meg az olvasó. *Az online analitikus adatfeldolgozást miért nem közvetlenül ezeken hajtjuk végre ahelyett, hogy további időt és erőforrásokat felhasználva külön adattárházat építünk?* A fő indok erre az elválasztásra az, hogy *mindkét rendszer magas hatékonyságát* elősegítsük. Egy operatív adatbázist ismert feladatok és munkaterhelés alapján tervezték és állították be. Ilyen feladat például az elsődleges kulcs szerinti indexelés és tördelés, adott rekordok keresése vagy az előre elkészített lekérdezések optimalizálása. Ezzel szemben az adattárház-lekérdezések gyakran összetettek. Ide tartozik az összesített szintű nagy adatcsoportok számítása, amely speciális adatszervezést, adatelérést és többdimenziós nézettáblákon alapuló implementációs módszerek használatát igényelheti. Ha az OLAP-lekérdezéseket operatív adatbázison hajtanánk végre, az lényegesen csökkentené az operatív feladatok hatékonyságát.

Ezenkívül az operatív adatbázis sok tranzakció egyidejű feldolgozását is támogatja. Ahhoz, hogy a tranzakciók konzisztenciáját és hibatűrő képességét biztosítsuk, konku-

renciavezérlő és helyreállító mechanizmusokra (például zárolásra és naplózásra) van szükség. Az OLAP-lekérdezésnek gyakran csak olvasó adatelérésre van szüksége az összefoglalások és az összesítések elkészítéséhez. Ha ilyen OLAP-műveletek esetén alkalmaznánk a konkurenciavezérlő és helyreállító mechanizmusokat, az veszélyeztetné a konkurens tranzakciók végrehajtását, és így lényegesen csökkentené az OLTP-rendszer teljesítményét.

Végül az operatív adatbázis elválasztása az adattárháztól a két rendszeren használt eltérő adatszerkezetek, a tartalom és az adatok eltérő felhasználása miatt is hasznos. A döntéshozás támogatásához történeti adatokra van szükség, egy operatív adatbázis viszont jellemzően nem ilyen adatokat tárol. Bár az itt található adatok mennyisége bőséges, mégis általában távol áll a tökéletestől ahhoz, hogy a döntéshozatal során használható legyen. A döntéstámogatás heterogén forrásokból származó adatok egyesítését igényli (például összesítést és összefoglalást). Eredményül jó minőségű, tiszta, integrált adatokat kapunk. Ezzel szemben egy operatív adatbázis csak részletes, nyers adatot tartalmaz (például tranzakciókat), amelyeket elemzés előtt egyesíteni kell. Mivel a két rendszer működése egymástól eléggé eltérő és másféle adatokat igényelnek, ezért van szükség külön adatbázisok fenntartására. Azonban sok operatív relációs adatbázis-kezelő rendszert árusító cég elkezdte úgy optimalizálni a rendszereit, hogy OLAP-lekérdezéseket is támogassanak. Ennek a tendenciának a folytatásával az OLTP- és OLAP-rendszerek elválasztása várhatóan csökkenni fog.

2.2. | A többdimenziós adatmodell

Az adattárházak és az OLAP-eszközök egy **többdimenziós adatmodellt** vesznek alapul, amely az adatot *adatkockának* tekinti. Ebben a részben megismerjük, hogy az adatkocka hogyan modellezi az *n*-dimenziós adatokat. Szó lesz még a fogalmi hierarchiákról, és arról, hogyan lehet ezeket OLAP-alapműveletekben használni, hogy lehetőség nyíljon különböző absztrakciós szinteken történő interaktív adatbányászatra.

2.2.1. | A táblázattól és számolótáblától az adatkockáig

Mi az adatkocka? Az **adatkocka** lehetővé teszi, hogy az adatot több dimenzióban modellezzük és több dimenzióban tekintsünk rá. Az adatkockát dimenziók és tények definiálják.

Általában véve egy **dimenzió** egy olyan nézőpont vagy elem, amellyel kapcsolatban a szervezet feljegyzéseket kíván készíteni. Például az *ElektroMind* nevű cég létrehozhat egy *értékesítés* adattárházat, hogy áruházainak eladásait nyilvántartsa az *idő, árucikk, üzlet, cím* dimenziók szerint. Ezekkel a dimenziókkal lehetővé válik olyan dolgok nyomon követése, mint például az árucikkek iránti havi kereslet alakulása, az üzlet és annak címe, ahol az értékesítés történt. Minden dimenzióhoz hozzárendelhető egy táblázat, a **dimenziótáblázat**, amely további leírást ad a dimenzióról. Például az *árucikk* dimenziótáblázata tartalmazhatja az *árucikk_név, márka, típus* attribútumokat. A dimenziótáblázatokat specifikálhatják a felhasználók vagy szakértők, de az adateloszlások alapján is lehet automatikusan generálni, majd finomítani ilyen táblázatokat.

A többdimenziós adatmodell jellemzően egy központi téma köré, például az *értékesítés* köré szerveződik. Ezt egy ténytáblázat fejezi ki. A **tények** számértékek, valamilyen mértékegységben megadva. Úgy kell rájuk gondolni, mint azokra a mennyiségekre vagy mértékekre, amelyek szerint elemezni akarjuk a dimenziók közötti összefüggéseket. Az eladásokat leíró adattárházhoz tartozó tények lehetnek például *dollár_eladott* (az értékesítések összege dollárban), *darab_eladott* (az értékesített árucikkek darabszáma) és *összeg_könyvelt*. A **ténytáblázat** tartalmazza a *tények* vagy mértékek nevét és egy-egy kulcsot minden kapcsolódó dimenziótáblázathoz. Nemsokára tisztább képet kapunk erről, amikor a többdimenziós sémákkal kezdünk el foglalkozni.

Habár a kockára általában mint háromdimenziós geometriai struktúrára szokás tekinteni, az adattárház-kezelésben az adatkocka *n*-dimenziós. Hogy jobban megértsük az adatkockát és a többdimenziós adatmodellt, kezdetben vegyünk egy egyszerű kétdimenziós adatkockát, amely tulajdonképpen az *ElektroMind* értékesítési adataira vonatkozó táblázat vagy számolótábla. Nézzük azokat az adatokat, amelyek a Vancouverben eladott árucikkekre vonatkoznak negyedéves bontásban. Ezek a 2.2. táblázatban találhatók. Ebben a kétdimenziós ábrázolásban a Vancouverben történt eladások láthatók az *idő* (negyedéves léptékben) és az *árucikk* (az eladott árucikk típusa szerinti bontásban) dimenziókra vonatkozóan. A feltüntetett tény vagy mérték a *dollár_eladott* (ezrekben kifejezve).

2.2. táblázat | Az *ElektroMind* értékesítési adatainak kétdimenziós nézettáblázata az *idő* és az *árucikk* dimenziók szerint. A vásárlások a vancouveri üzletekben történtek. A feltüntetett mérték a *dollár_eladott* (ezrekben megadva)

idő (negyedév)	**cím** = "Vancouver"			
	árucikk (típus)			
	Szórakoztató elektronika	számítógép	telefon	biztonság
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

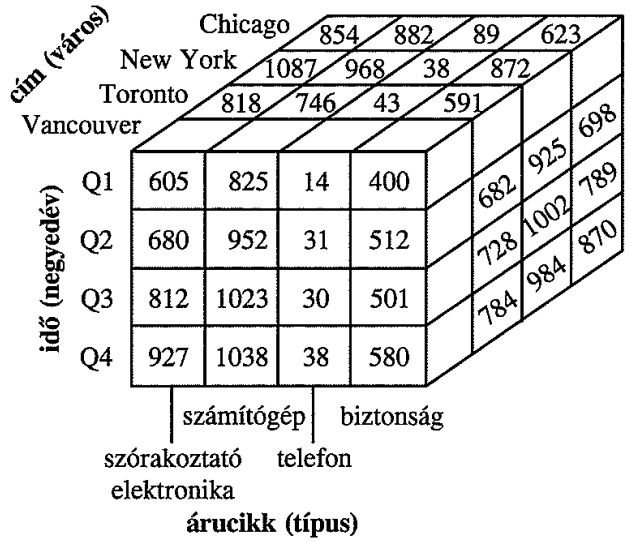
Most tegyük fel, hogy egy harmadik dimenzió mentén is szeretnénk látni az eladási adatokat. Az *idő* és *árucikk* dimenziókhoz például hozzá akarjuk venni a *cím* dimenziót is Chicago, New York, Toronto és Vancouver esetén. Ezeket a háromdimenziós adatokat a 2.3. táblázatban találjuk meg. Itt az adatok kétdimenziós táblázatok sorozataként vannak feltüntetve, de ugyanezeket az adatokat háromdimenziós adatkocka formájában is ábrázolhatjuk, ahogy ez a 2.1. ábrán látható.

Tegyük fel, hogy egy negyedik dimenziót is meg szeretnénk jeleníteni, például a *szállítót*. Négy dimenzióban látni már trükkös dolog, de egy négydimenziós kockát felfoghatunk háromdimenziós kockák sorozataként is, ahogy ezt a 2.2. ábra jól szemlélteti. Ha ezt így folytatjuk, bármely *n*-dimenziós adatot ábrázolhatunk $n - 1$-dimenziós „kockák" sorozataként. Az adatkocka tulajdonképpen egy metafora a többdimenziós adattárolásra. Az ilyen adatok valódi fizikai tárolása eltérhet a tárolás logikai reprezentációjától. Azt

2.3. táblázat | Az *ElektroMind* értékesítési adatainak háromdimenziós nézettáblázata az *idő*, *árucikk* és a *cím* dimenziók szerint. A feltüntetett mérték a *dollár_eladott* (ezrekben megadva)

	cím = "Chicago"				**cím** = "New York"				**cím** = "Toronto"				**cím** = "Vancouver"			
	árucikk				**árucikk**				**árucikk**				**árucikk**			
idő	SZ	SZG	T	B	SZ	SZG	T	B	SZ	SZG	T	B	SZ	SZG	T	B
Q1	854	882	89	623	1087	968	38	872	818	746	43	605	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	680	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	812	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	927	927	1038	38	580

Jelölések: SZ: szórakoztató elektronika, SZG: számítógép, T: telefon, B: biztonság



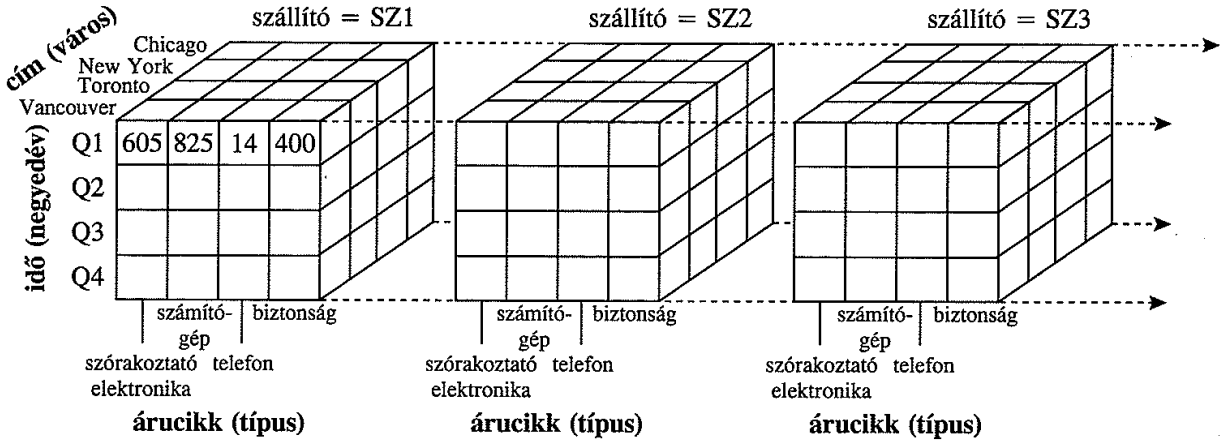
2.1. ábra | A 2.3. táblázat adatainak háromdimenziós adatkockában történő ábrázolása az *idő*, az *árucikk* és a *cím* dimenziók szerint. A feltüntetett mérték a *dollár_eladott* (ezrekben megadva)

fontos mindig szem előtt tartani, hogy az adatkockák *n*-dimenziósak, és nem szorítják az adatokat három dimenzióba.

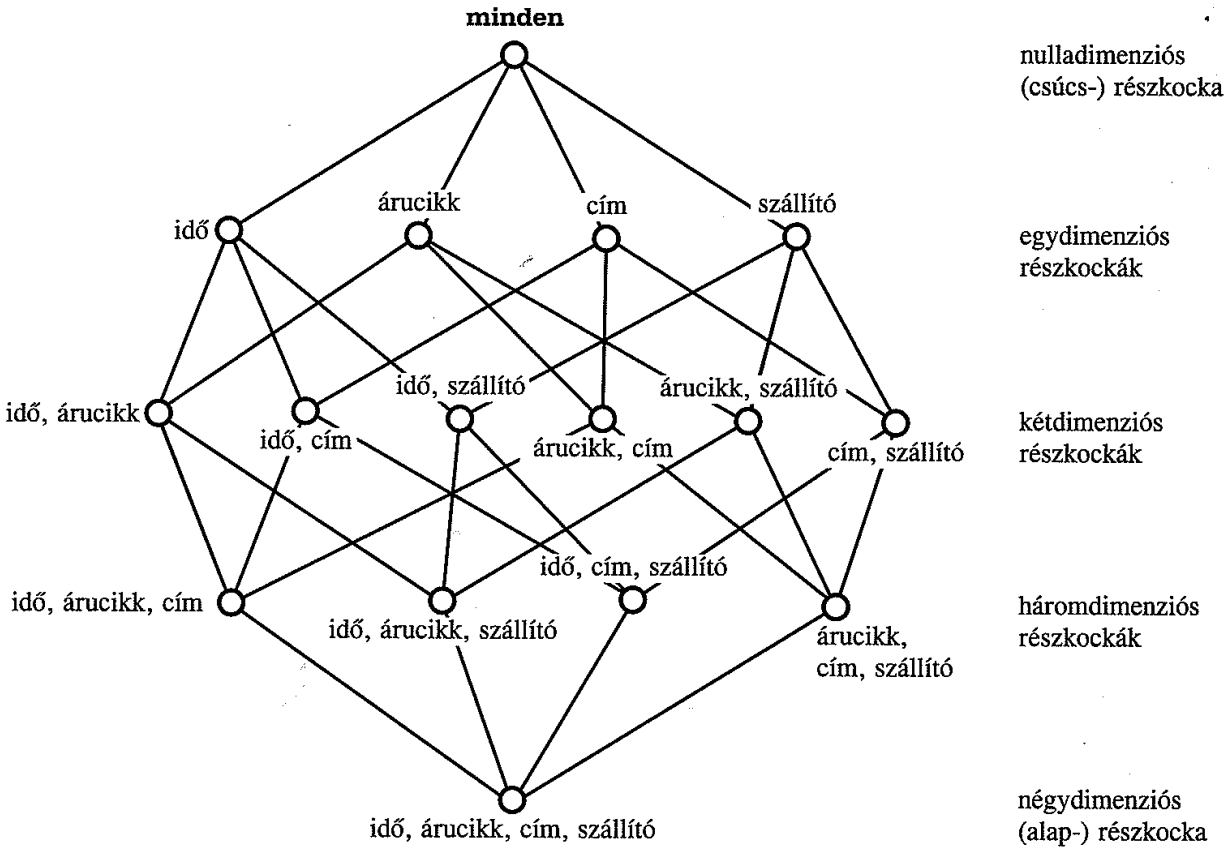
A fenti táblázatok az adatokat az összefoglalás különböző fokán mutatják. Az adattárház-kezelés irodalmában az ilyen adatkockákat **részkockának** hívják. Ha adott a dimenziók halmaza, létre tudjuk hozni a részkockák *hálóját*, amelyben minden részkocka az adatok egy más összefoglalási vagy **group by**[2] (azaz a dimenziók egy más részhalmaza szerinti összefoglalás) szintjét mutatja. Erre a hálóra hivatkozunk aztán úgy, hogy adatkocka. A 2.3. ábrán láthatunk egy részkockahálót, amely az *idő, árucikk, cím* és *szállító* dimenziókkal alkot egy adatkockát.

Azt a részkockát, amely az összefoglalás legalacsonyabb szintjét képviseli, **alapkockának** nevezzük. Például a 2.2. ábrán látható négydimenziós részkocka az alapkocka az *idő, árucikk, cím* és *szállító* dimenziókkal. A 2.1. ábrán egy háromdimenziós (nem alap) részkocka szerepel az *idő, árucikk,* és *cím* dimenziókkal, az összes szállítóra

2 | A lekérdező nyelvek parancsszavai mindig félkövér betűvel és Serifa betűtípussal jelennek meg.



2.2. ábra | Értékesítési adatok egy négydimenziós adatkocka ábrázolása az *idő*, *árucikk*, *cím* és a *szállító* dimenziók szerint. A feltüntetett mérték a *dollár_eladott* (ezrekben megadva). A jobb olvashatóság kedvéért csak néhány kockaérték látható



2.3. ábra | Egy négydimenziós adatkockát alkotó részkockaháló. A kocka dimenziói *idő*, *árucikk*, *cím* és *szállító*. Minden részkocka az összegzés különböző fokát jelöli

összegezve. A nulladimenziós részkockát, amely az összefoglalás legmagasabb szintjét képviseli, **csúcskockának** nevezzük. Példánkban ez az összes eladás vagy a *dollár_eladott* összegezve mind a négy dimenzió mentén. A csúcskockát tipikusan a **minden** kifejezés jelöli.

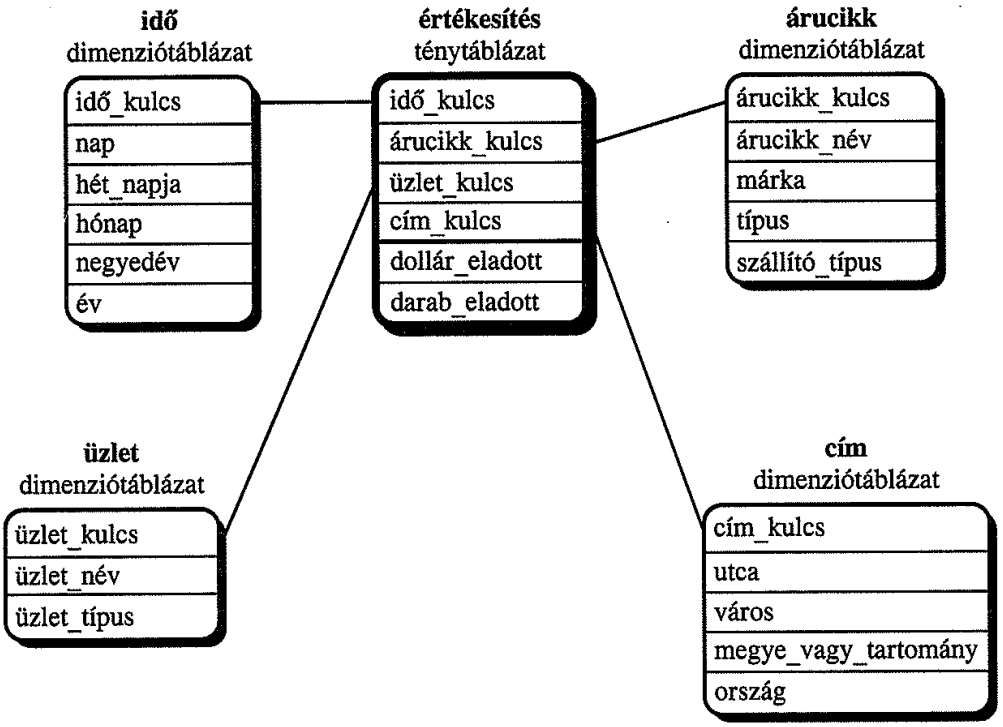
2.2.2. | Csillagok, hópelyhek és csillagképek: sémák többdimenziós adatbázisokhoz

Az egyed–kapcsolat adatmodellt rendszerint relációs adatbázisok tervezésénél használják, ahol az adatbázisséma egyedek halmazából és a köztük lévő kapcsolatokból áll. Az ilyen adatmodell alkalmas az on-line tranzakció-feldolgozásra. Egy adattárházban azonban tömör, témaorientált sémára van szükség, amely lehetővé teszi az on-line adatelemzést.

Adattárházakhoz a legnépszerűbb adatmodell a **többdimenziós modell**. Ilyen séma létezhet **csillagséma**, **hópehelyséma** vagy **csillagképséma** formájában. Most nézzük meg mindegyik sématípust.

Csillagséma – A leggyakoribb modellező minta a csillagséma, amelyben az adattárház tartalmaz (1) egy nagy központi táblázatot (**ténytáblázat**), amely nagymennyiségű adatot tárol redundancia nélkül, és (2) kisebb kísérő táblázatokat (**dimenziótáblázatok**), egyet minden dimenzióhoz. A sémagráf egy csillaghoz hasonlít, ahol középen van a ténytáblázat és ezt a dimenziótáblázatok veszik körül.

2.1. példa | A 2.4. ábrán láthatunk egy példát a csillagsémára az *ElektroMind* értékesítési adataihoz. Az eladásokat négy dimenzió mentén tekintjük: *idő, árucikk, üzlet* és *cím*. A sémában megtalálható egy központi ténytáblázat az *értékesítés* adataihoz, amely mind a négy dimenzióhoz tartalmaz kulcsokat, és még két mértéket is: *dollár_eladott* és *darab_eladott*. Hogy a ténytáblázat méretét minimalizáljuk, a dimenzióazonosítókat (például *idő_kulcs* vagy *árucikk_kulcs*) a rendszer generálja. ■



2.4. ábra | Csillagséma az értékesítési adatok tárházához

Vegyük észre, hogy a csillagsémában minden dimenziót csak egyetlen táblázat képvisel és minden ilyen táblázat tartalmaz attribútumokat. Például a *cím* dimenziótáblázat a következő attribútumhalmazzal rendelkezik: {*cím_kulcs, utca, város, megye_vagy_tartomány, ország*}. Ez a megszorítás valamekkora redundanciával járhat. Például *Vancouver* és *Victoria* is a kanadai British Kolumbia megye városa. A *cím* dimenziótáblázat ezekhez a városokhoz tartozó bejegyzései redundanciát eredményezhetnek a *megye_vagy_tartomány* és *ország* attribútumokat illetően, azaz a következő sorok redundánsak: (..., Vancouver, Brit Kolumbia, Kanada), (..., Victoria, Brit Kolumbia, Kanada). Ehhez még hozzájön az is, hogy az attribútumok egy dimenziótáblázaton belül vagy hierarchiát (teljes rendezés), vagy hálót (részben rendezés) alkotnak.

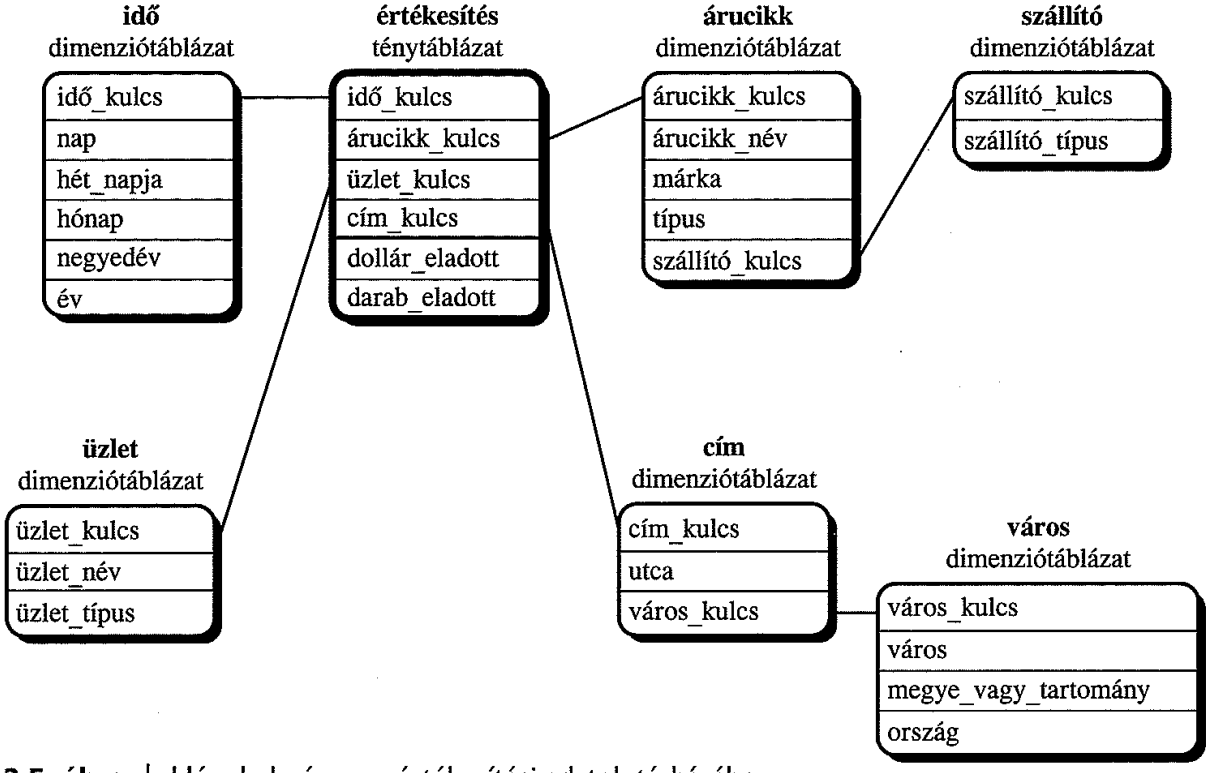
Hópehelyséma – A hópehelyséma a csillagséma egy változata, ahol néhány dimenziótáblázat *normalizálva* van, és így az adatok további táblázatokba vannak felosztva. Az eredményül kapott sémagráf alakja hópehelyre hasonlít.

A fő különbség a két modell között az, hogy a hópehelymodell dimenziótáblázatait normalizált alakban lehet tartani, hogy a redundancia mértéke csökkenjen. Egy ilyen táblázatot könnyű karbantartani, és kevesebb helyet is foglal, hiszen egy nagy dimenziótáblázat hatalmassá nőhet, ha a dimenzió egész szerkezete megjelenik az oszlopokban. Ez a tárnyereség azonban elhanyagolható a ténytáblázat tipikus nagyságrendjéhez képest. Ezenkívül a hópehely szerkezet csökkentheti a keresés hatékonyságát, mivel egy lekérdezés végrehajtásához több összekapcsolásra lesz szükség. Ebből következik, hogy a séma használata a rendszer teljesítményére hátrányos kihatással járhat. Ennélfogva a hópehelyséma nem olyan népszerű az adattárház-tervezésben, mint a csillagséma.

2.2. példa | A 2.5. ábrán láthatunk egy példát a hópehelysémára az *ElektroMind* értékesítési adataihoz. Az *értékesítés* ténytáblázat megegyezik a 2.4. ábrán látható csillagséma ténytáblázatával. A fő különbség a két séma között a dimenziótáblázatok definíciójában van. A csillagséma *árucikk* dimenziótáblázata a hópehelysémában normalizálva van, és ez azt eredményezi, hogy az eredeti egy táblázatból két új táblázat lesz: *árucikk* és *szállító*. Az *árucikk* táblázat most például tartalmazhatja a *árucikk_kulcs, árucikk_név, márka, típus* és *szállító_kulcs* attribútumokat, ahol *szállító_kulcs* a *szállító* dimenziótáblázathoz kapcsolódik, amely a *szállító_kulcs* és *szállító_típus* információkat tartalmazza. Hasonlóan, a csillagsémában eddig egy dimenziótáblázatként megjelenő *cím* normalizálás után két új táblázatban ölt testet: *cím* és *város*. A *város_kulcs* az új *cím* táblázatban a *város* dimenziótáblázathoz kapcsolódik. Vegyük észre, hogy szükség esetén további normalizálások is elvégezhetők a 2.5. ábra hópehelysémájában a *megye_vagy_tartomány* és *ország* attribútumokat illetően. ■

Csillagképséma – kifinomult alkalmazásokban szükség lehet arra, hogy több ténytáblázat *megosszon* egymás között egy-egy dimenziótáblázatot. Az efféle séma csillagok gyűjteményének is tekinthető, ezért **galaxissémának** vagy **csillagképnek** nevezik.

2.3. példa | A 2.6. ábrán látható példa a csillagképsémára. Ebben a sémában két ténytáblázat szerepel: *értékesítés* és *fuvarozás*. Az *értékesítés* táblázat definíciója megegyezik a csillagsémabeli ténytáblázat definíciójával (2.4. ábra). A *fuvarozás* táblázatban öt dimen-



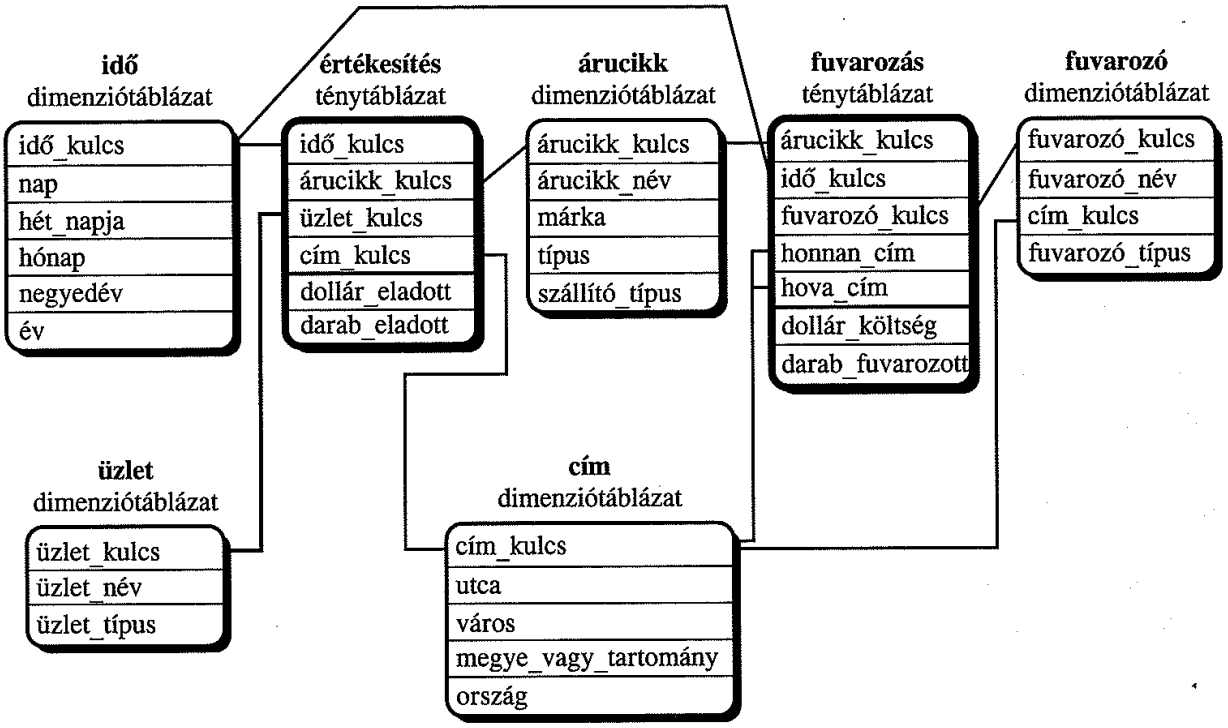
2.5. ábra | Hópehelyséma az értékesítési adatok tárházához

ziót, illetve kulcsot találunk: *árucikk_kulcs, idő_kulcs, fuvarozó_kulcs, honnan_cím, hova_cím,* és még két mértéket: *dollár_költség, darab_fuvarozott*. A csillagképsémában lehetőség van arra, hogy a dimenziótáblázatokat a ténytáblázatok megosszák egymás között. Például az *idő, árucikk* és *cím* dimenziótáblázatokat megosztja a két ténytáblázat: *értékesítés* és *fuvarozás*. ■

Az adattárház-kezelésben különbséget teszünk adattárház és adatpiac között. Az **adattárház** olyan témákról gyűjt információkat, amelyek az *egész szervezeten* átívelnek, például *vevő*, *árucikk*, *értékesítés*, *nyereség* vagy *személyzet*, és így az alkalmazási területe az egész vállalatra kiterjed, *vállalatméretű*. Adattárházakban rendszerint a csillagképsémát használják, mivel ezzel több, egymással kölcsönösen kapcsolatban álló témát is lehet modellezni. Egy **adatpiac** azonban az adattárház egy részhalmaza, amely kiválasztott témákkal foglalkozik csak, így az alkalmazási területe részleges, *osztályméretű*. Adatpiacokban általában a *csillag-* vagy a *hópehelysémát* alkalmazzák, mivel mindkettő magában álló témák modellezésére használatos, a csillagséma azonban népszerűbb és hatékonyabb is.

2.2.3. | Példák csillag-, hópehely- és csillagképsémák definiálására

Hogyan határozzak meg egy többdimenziós sémát az adataimhoz? Ahogy a relációs lekérdezések meghatározásához relációs lekérdező nyelveket, például az SQL-t lehet használni, ugyanúgy egy **adatbányász lekérdező nyelv (DMQL, Data Mining Query Language)** használható az adatbányászati feladatok megadásához. Mi egy SQL alapú adatbányász



2.6. ábra | Csillagképséma az értékesítési és fuvarozási adatok tárházához

lekérdező nyelvet, a továbbiakban **DMQL**-t fogjuk megvizsgálni, amely adattárházak és adatpiacok definiálásához tartalmaz nyelvprimitíveket. Más adatbányász feladatok meghatározásához, például fogalom-/osztályleírások, társítások, osztályozások stb. bányászatához használható nyelvprimitívek a 4. fejezetben jönnek majd elő.

Az adattárházak és az adatpiacok két primitív használatával definiálhatók. Az egyik a kocka definiálására, a másik a dimenziók definiálására való. A *kockadefiniáló* utasítás szintaxisa a következő:

define cube ⟨kockanév⟩ **[**⟨dimenzó_lista⟩**]** : ⟨mérték_lista⟩

A *dimenziódefiniáló* utasítás szintaxisa:

define dimension ⟨dimenzionév⟩ **as** (⟨attribútum_vagy_aldimenzió_lista⟩)

Most nézzük, hogyan definiáljuk DMQL-ben a 2.1., 2.2., 2.3. példa csillag-, hópehely- és csillagképsémáit. A DMQL-kulcsszavak Serifa betűvel vannak szedve.

2.4. példa | A 2.1. példa és 2.4. ábra csillagsémája DMQL-ben a következőképpen definiálható:

define cube értékesítés_csillag [idő, árucikk, üzlet, cím]: dollár_eladott =
= `sum`(eladások_dollárban), darab_eladott = `count(*)`
define dimension idő **as** (idő_kulcs, nap, hét_napja, hónap, negyedév, év)
define dimension árucikk **as** (árucikk_kulcs, árucikk_név, márka, típus, szállító_típus)

define dimension üzlet **as** (üzlet_kulcs, üzlet_név, üzlet_típus)
define dimension cím **as** (cím_kulcs, utca, város, megye_vagy_tartomány, ország)

A **define cube** utasítás egy *értékesítés_csillag* nevű adatkockát definiál, amely a 2.1. példa központi *értékesítés* ténytáblázatának felel meg. Az utasítás megadja a dimenziókat és a két mértéket: *dollár_eladott* és *darab_eladott*. Az adatkockának négy dimenziója van: *idő, árucikk, üzlet* és *cím*. Az egyes dimenziók meghatározásához a **define dimension** utasítást használtuk. ■

2.5. példa | A 2.2. példa és 2.5. ábra hópehelysémája a következőképpen definiálható DMQL-ben:

define cube értékesítés_hópehely [idő, árucikk, üzlet, cím]: dollár_eladott =
= sum(eladások_dollárban), darab_eladott = count(*)
define dimension idő **as** (idő_kulcs, nap, hét_napja, hónap, negyedév, év)
define dimension árucikk **as** (árucikk_kulcs, árucikk_név, márka, típus,
szállító(szállító_kulcs, szállító_típus))
define dimension üzlet **as** (üzlet_kulcs, üzlet_név, üzlet_típus)
define dimension cím **as** (cím_kulcs, utca, város(város_kulcs, város,
megye_vagy_tartomány, ország))

A definíció hasonló az *értékesítés_csillag* meghatározásához (2.4. példa), attól eltekintve, hogy itt az *árucikk* és *cím* dimenziótáblázatok normalizálva vannak. Az *értékesítés_csillag* adatkocka *árucikk* dimenzióját például két dimenziótáblázatba normalizáltuk az *értékesítés_hópehely* kockában: *árucikk* és *szállító*. Vegyük észre, hogy a *szállító*-hoz tartozó dimenziódefiníció az *árucikk* meghatározásán belül van megadva. A *szállító* definíciója értelemszerűen létrehoz egy *szállító_kulcs*-ot az *árucikk* dimenziótáblázat definíciójában. Hasonlóan, az *értékesítés_csillag* adatkocka *cím* dimenziója az *értékesítés_hópehely* kockában *cím* és *város* dimenziótáblázatokra normalizált. A *város*-hoz tartozó dimenziódefiníció a *cím* meghatározásán belül van megadva. Ily módon értelemszerűen létrejött egy *város_kulcs* a *cím* dimenziótáblázat definíciójában. ■

Végül nézzünk egy példát a csillagképsémára. Ez a séma kölcsönösen összekapcsolt kockák halmazaként definiálható.

2.6. példa | A 2.3. példa és 2.5. ábra csillagképsémája DMQL-ben a következőképpen definiálható:

define cube értékesítés [idő, árucikk, üzlet, cím]: dollár_eladott =
= sum(eladások_dollárban), darab_eladott = count(*)
define dimension idő **as** (idő_kulcs, nap, hét_napja, hónap, negyedév, év)
define dimension árucikk **as** (árucikk_kulcs, árucikk_név, márka, típus, szállító_típus)
define dimension üzlet **as** (üzlet_kulcs, üzlet_név, üzlet_típus)
define dimension cím **as** (cím_kulcs, utca, város, megye_vagy_tartomány, ország)

define cube fuvarozás [idő, árucikk, fuvarozó, honnan_cím, hova_cím]:
dollár_költség = sum(költség_dollárban), darab_fuvarozott = count(*)
define dimension idő **as** idő **in cube** értékesítés
define dimension árucikk **as** árucikk **in cube** értékesítés
define dimension fuvarozó **as** (fuvarozó_kulcs, fuvarozó_név,
cím **as** cím **in cube** értékesítés, fuvarozó_típus)
define dimension honnan_cím **as** cím **in cube** értékesítés
define dimension hova_cím **as** cím **in cube** értékesítés

A **define cube** utasítást használtuk a 2.3. példa két ténytáblázatának megfelelő *értékesítés* és *fuvarozás* adatkockák meghatározására. Vegyük észre, hogy az *értékesítés* kocka az *idő*, *árucikk* és *cím* dimenzióit megosztja a *fuvarozás* kockával. Ezt például az *idő* dimenzió esetén a **define cube** *fuvarozás* utasítás alatti **define dimension** *idő* **as** *idő* **in cube** *értékesítés* utasítással jelezzük. ■

2.2.4. | A mértékek osztályozása és számítása

Hogyan számítjuk a mértékeket? Hogy erre a kérdésre válaszoljunk, először azt nézzük meg, hogyan lehet a mértékeket osztályozni. Vegyük észre, hogy az adatkockatérben egy többdimenziós pont meghatározható a dimenziók segítségével, például ⟨*idő* = "*Q*1", *cím* = "*Vancouver*", *árucikk* = "*számítógép*"⟩. Az **adatkockamérték** egy numerikus függvény, amely az adatkockatér minden pontján kiértékelhető. Egy adott pontban a mérték értékét a pontot leíró dimenzióértékeknek megfelelő adatok összesítésével adhatjuk meg. Erre rövidesen konkrét példát is látunk majd.

A mértékek három osztályba sorolhatók, a felhasznált *összesítőfüggvény* típusa alapján.

- **Disztributív** – Az összesítőfüggvény *disztributív*, ha az értéke a következő módon számolható ki. Tegyük fel, hogy az adatok *n* részre vannak osztva. Minden részhez számítsuk ki az összesítőfüggvény értékét. Az összesítőfüggvény disztributív, ha ugyanazt az értéket kapjuk akkor, ha erre az *n* összesített értékre alkalmazzuk az összesítést, és akkor is, ha az összesítést az eredeti, osztatlan adatokra alkalmazzuk. Például egy adatkockán kiszámíthatjuk a `count()` függvényt úgy, hogy először a kockát felosztjuk alkockákra, azokon kiszámítjuk a `count()` függvény értékét, majd ezeket az értékeket összeadjuk. Tehát a `count()` függvény disztributív összesítőfüggvény. Hasonlóan a `sum()`, `min()` és `max()` függvények is disztributív összesítőfüggvények. Egy mérték *disztributív*, ha disztributív összesítőfüggvény alkalmazásával számítottuk ki.
- **Algebrai** – Az összesítőfüggvény *algebrai*, ha kiszámítható egy *M* argumentumú algebrai függvény segítségével, ahol *M* korlátos pozitív egész, és az argumentumok mindegyike disztributív összesítőfüggvény eredménye. Például az `avg()` (átlag) kiszámítható `sum()/count()` alakban, ahol `sum()` és `count()` is disztributív összesítőfüggvények. Hasonlóan megmutatható, hogy `min_N()`, `max_N()` és `standard_deviation()` függvények is algebrai összesítőfüggvények. Egy mérték *algebrai*, ha algebrai összesítőfüggvény alkalmazásával számítottuk ki.

- **Teljes körű** – Az összesítőfüggvény *teljes körű*, ha egy alösszesítés leírásához szükséges tár méretére nincs konstanssal megadható korlátozás. Ez azt jelenti, hogy nincs olyan algebrai függvény M argumentummal (ahol M konstans), amely leírná a számítás menetét. A gyakori példák között szerepelnek a `median()`, `mode()` [a leggyakrabban előforduló elem(ek)] és `rank()` függvények. Egy mérték *teljes körű*, ha teljes körű összesítőfüggvény alkalmazásával számítottuk ki.

A legtöbb adatkocka-alkalmazás a disztributív és algebrai mértékek hatékony számítását igényli. Erre sok eredményes technika létezik. Ezzel ellentétben a teljes körű mértékek hatékony számítása nehézségekbe ütközhet. A teljes körű mértékek *közelítő* számítására azonban vannak hatékony technikák. Például pontos `median()` számítás helyett egy nagy adathalmazhoz kielégítő eredménnyel becsülhető egy közelítő érték a megfelelő technikák alkalmazásával. Az ilyen technikák sok esetben megfelelnek arra, hogy a teljes körű mértékek hatékony számításából adódó nehézségeken túljussunk.

2.7. példa | Egy adatkockában sok mérték kiszámítható relációs összesítőműveletekkel. A 2.4. ábrán az *ElektroMind* értékesítési adatainak csillagsémáját láthattuk, amely két mértéket tartalmaz: *dollár_eladott* és *darab_eladott*. A 2.4. példában a sémának megfelelő *értékesítés_csillag* adatkocka DMQL-utasításokkal volt megadva. *De hogyan értelmezzük ezeket a parancsokat, hogy a megadott adatkockát kapjuk?*

Tegyük fel, hogy az *ElektroMind* relációs adatbázissémája a következő:

idő(idő_kulcs, nap, hét_napja, hónap, negyedév, év)
árucikk(árucikk_kulcs, árucikk_név, márka, típus, szállító_típus)
üzlet(üzlet_kulcs, üzlet_név, üzlet_típus)
cím(cím_kulcs, utca, város, megye_vagy_tartomány, ország)
értékesítés(idő_kulcs, árucikk_kulcs, üzlet_kulcs, cím_kulcs, darabszám_eladott, ár)

A 2.4. példa DMQL-specifikációja az alábbi SQL-lekérdezésre fordítható le, amely a kívánt *értékesítés_csillag* kockát fogja létrehozni. Itt a `sum()` összesítőfüggvényt használjuk a *dollár_eladott* és *darab_eladott* mértékek kiszámítására.

```
select é.idő_kulcs, é.árucikk_kulcs, é.üzlet_kulcs, é.cím_kulcs,
       sum(é.darabszám_eladott * é.ár),
       sum(é.darabszám_eladott)
from idő i, árucikk á, üzlet ü, cím c, értékesítés é
where é.idő_kulcs = i.idő_kulcs and é.árucikk_kulcs = c.árucikk_kulcs and
      é.üzlet_kulcs = ü.üzlet_kulcs and é.cím_kulcs = c.cím_kulcs
group by é.idő_kulcs, é.árucikk_kulcs, é.üzlet_kulcs, é.cím_kulcs
```

A fenti lekérdezés által létrehozott kocka az *értékesítés_csillag* adatkocka alapkockája. Megtalálható benne az összes dimenzió, amely az adatkocka definíciójában le volt írva, és mindegyik finomsága az **összekapcsoló kulcs** szintjén van. Összekapcsoló kulcsnak nevezzük a ténytáblázatot a dimenziótáblázattal összekapcsoló kulcsot. Az alapkockának megfelelő ténytáblázatot néha **alaptény táblázatnak** is nevezik.

A **group by** záradék megváltoztatásával az *értékesítés_csillag* adatkockához más részkockákat is létrehozhatunk. Például *é.idő_kulcs* helyett csoportosíthatunk *i.hónap* szerint, és így az egyes csoportok mértékeit hónaponként összegezzük. Ha elhagyjuk a **group by** *é.üzlet_kulcs* kifejezést, akkor egy magasabb szintű részkockát kapunk (amelyben az eladások nincsenek üzletenként lebontva, hanem minden üzletre összegezve állnak rendelkezésre). Tegyük fel, hogy az egész **group by** záradékot töröljük. Ekkor megkapjuk a *dollár_eladott* teljes végösszegét és a *darab_eladott* összes darabszámát. Ez a nulladimenziós részkocka az *értékesítés_csillag* adatkocka csúcskockája. Kiválasztás és/vagy vetítés alkalmazásával további részkockák nyerhetők az alapkockából, és így felépíthető a 2.2.1. alfejezetben leírt részkockaháló. Minden részkocka az adatok egy más szintű összegzésének felel meg. ■

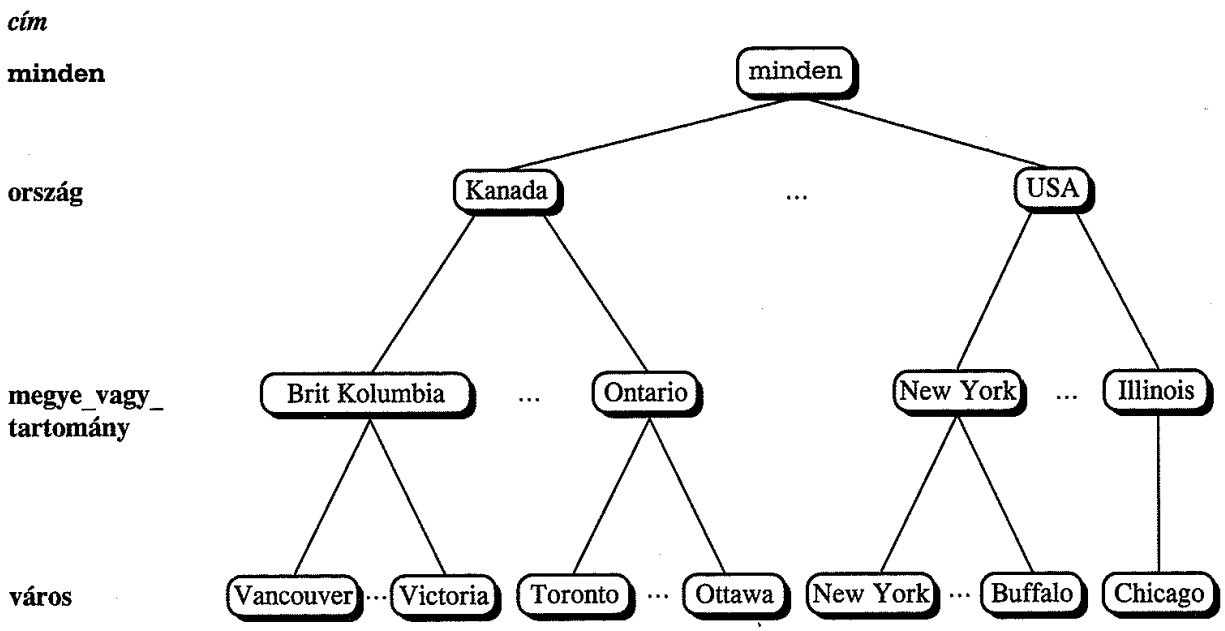
A legtöbb mai adatkocka-technológia csak *numerikus adatokkal* kapcsolatban vesz fel mértékeket a többdimenziós adatbázisba. Mértékek azonban másféle adatokra is alkalmazhatók, például térbeli, multimédiás vagy szöveges adatokra. Az ehhez szükséges technológiákat a 9. fejezetben tárgyaljuk.

2.2.5. | Fogalmi hierarchiák bevezetése

Mi a fogalmi hierarchia? A **fogalmi hierarchia** az alacsony szintű fogalmakról a magasabb szintű, általánosabb fogalmakra való leképezések egy sorozatát adja meg. Vegyünk a *cím* dimenzióhoz egy fogalmi hierarchiát. A *cím* dimenzióhoz tartozó városok Vancouver, Toronto, New York és Chicago. Minden város leképezhető abba a megyébe vagy tartományba, ahol megtalálható. Vancouver például leképezhető Brit Kolumbiába, Chicago pedig Illinoisba. A megyék és a tartományok viszont leképezhetők abba az országba, ahova tartoznak, esetünkben Kanadába és az USA-ba. Ezek a leképezések egy fogalmi hierarchiát alkotnak a *cím* dimenzióhoz, az alacsony szintű fogalmaktól (városok) a magasabb szintű, általánosabb fogalmak felé haladva (országok). A fent leírt fogalmi hierarchiát a 2.7. ábrán ábrázoltuk.

Sok fogalmi hierarchia magától értetődő az adatbázissémán belül. Tegyük fel például, hogy a *cím* dimenzió a következő attribútumokkal van leírva: *házszám, utca, város, megye_vagy_tartomány, irszám, ország*. Ezeken az attribútumokon létezik egy teljes rendezés, amely az *utca* < *város* < *megye_vagy_tartomány* < *ország* fogalmi hierarchiát adja. Ez a 2.8.(a) ábrán látható. A dimenzió attribútumain megadhatunk részleges rendezést is, amely egy **hálót** eredményez. Példa lehet erre az *idő* dimenzió *nap, hét, hónap, negyedév, év* attribútumain a *nap* < {*hónap* < *negyedév*; *hét*} < *év*[3] részleges rendezés. Az ehhez tartozó hálót a 2.8.(b) ábrán láthatjuk. Egy fogalmi hierarchiát, azaz egy adatbázissémában az attribútumok közötti teljes vagy részleges rendezést **sémahierarchi-**

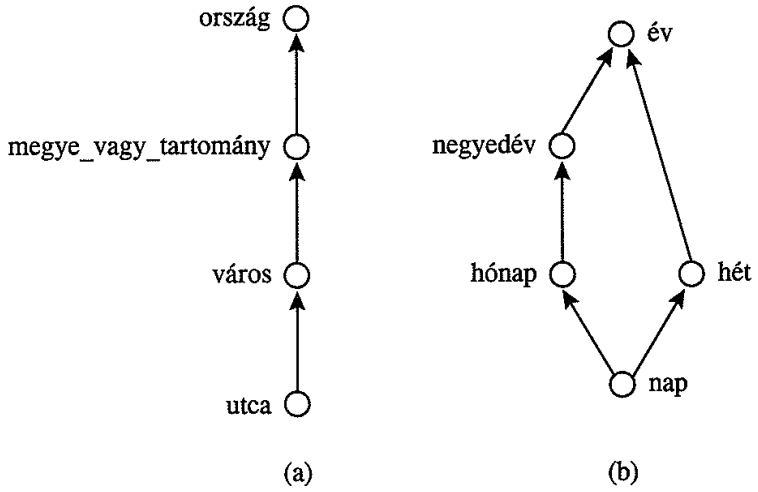
3 | Mivel egy *hét* gyakran két egymást követő hónaphoz is tartozhat, általában nem kezelik a *hónap*nál alacsonyabb szintű absztrakcióként. Ehelyett sokszor inkább az fordul elő, hogy az *év* alá helyezik alacsonyabb szintként, mivel egy évben megközelítőleg 52 hét van.



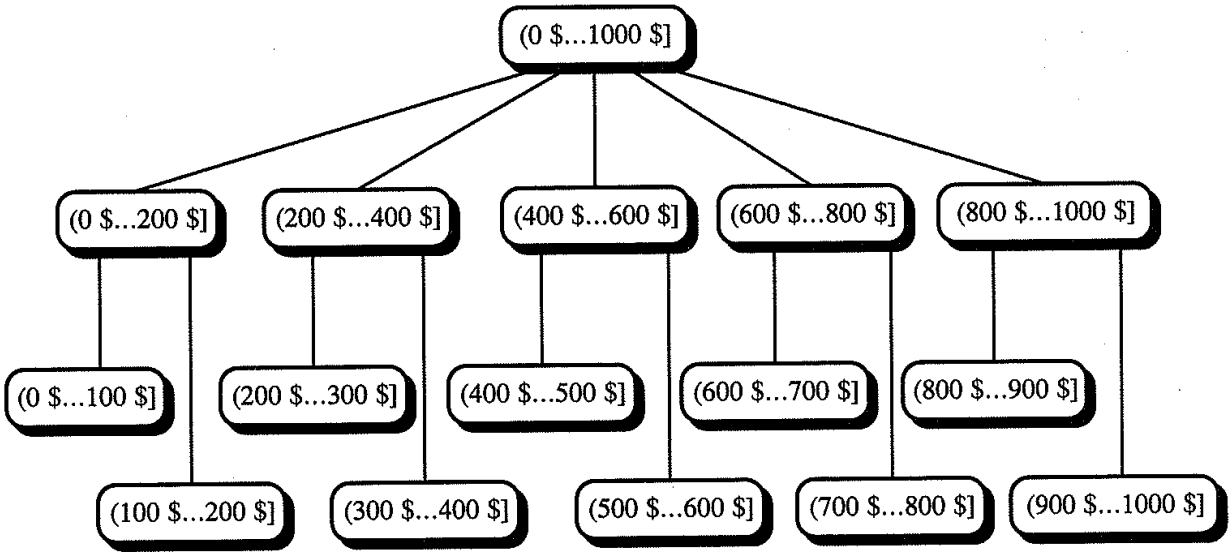
2.7. ábra | Fogalmi hierarchia a *cím* dimenzióhoz. Helyszűke miatt nem látható a hierarchia összes csúcspontja [erre a három pont (...) használata is utal]

ának nevezünk. Az adatbányászrendszerekben az alkalmazásokban gyakori fogalmi hierarchiák, például az *idő* fogalmi hierarchiája, előre definiálható. A rendszernek azonban olyan rugalmasnak kell lennie, hogy a felhasználó az elődefiniált hierarchiákat a saját igényei szerint megváltoztathassa. A felhasználó például definiálhatja a költségvetési évet április 1-jétől vagy a tanévet szeptember 1-jétől.

Fogalmi hierarchiákat egy dimenzió vagy attribútum értékeinek a szétválasztásával, illetve csoportosításával is létrehozhatunk. Ilyenkor **halmazcsoportosító hierarchiáról** beszélünk. Az értékcsoportokon megadhatunk teljes vagy részleges rendezést. A 2.9.



2.8. ábra | Hierarchikus és hálóstruktúrák a tárházdimenziók attribútumaihoz: (a) hierarchia a *cím*-hez; (b) háló az *idő*-höz



2.9. ábra | Fogalmi hierarchia az *ár* attribútumhoz

ábrán láthatunk példát az *ár* dimenzió halmazcsoportosító hierarchiájára, ahol a balról nyílt, jobbról zárt intervallumot a szokásos módon jelöltük.

Egy adott attribútumhoz vagy dimenzióhoz egynél több fogalmi hierarchia is tartozhat, a különböző felhasználói nézőpontok alapján. Egy másik felhasználó például jobbnak láthatja az *ár* dimenziót az *olcsó, elfogadható, drága* intervallumok megadásával szervezni.

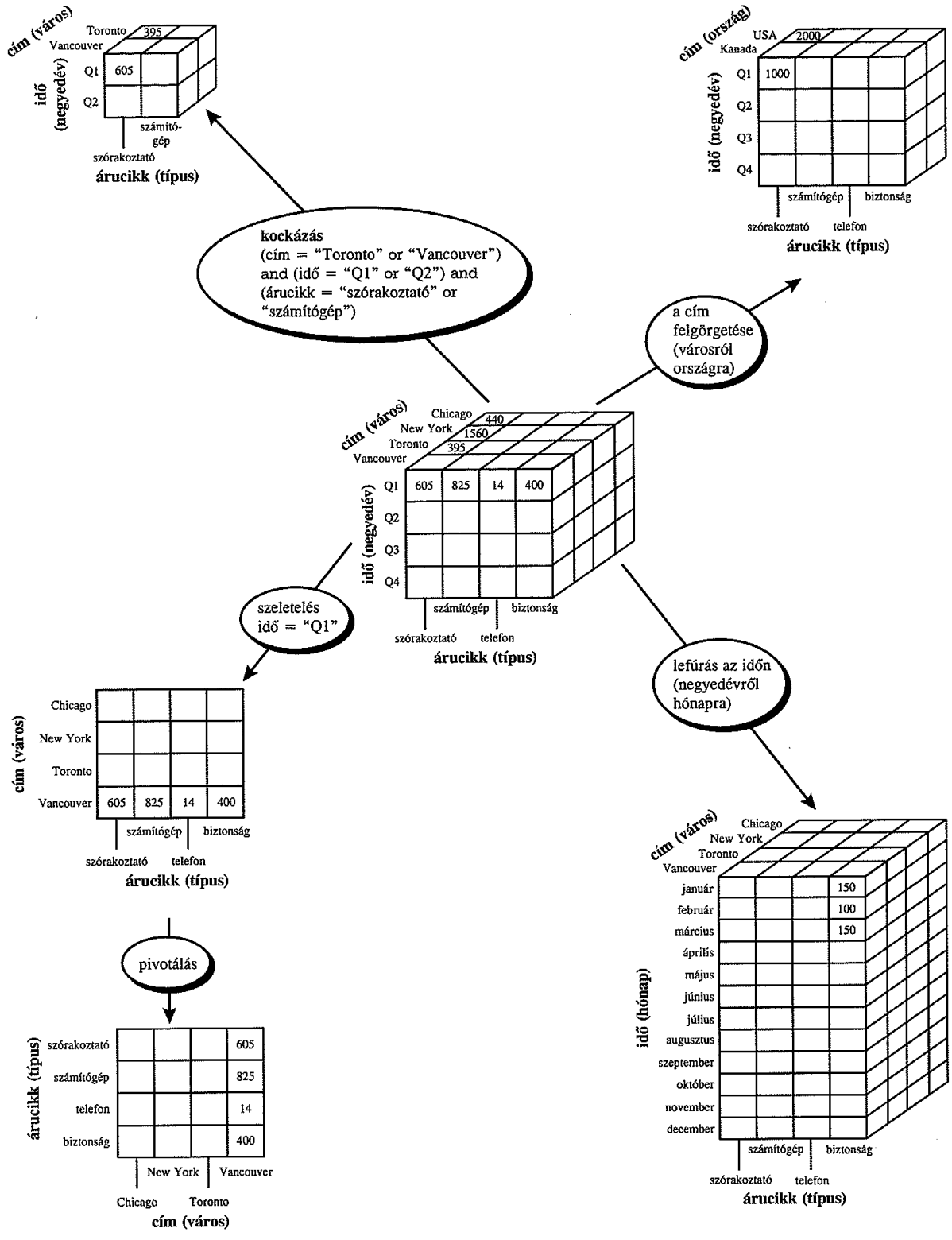
A fogalmi hierarchiákat létrehozhatják manuálisan a felhasználók, szakértők, tudásmérnökök, de az adateloszlás statisztikai elemzése alapján automatikusan is generálhatók. Ezzel bővebben a 3. fejezet foglalkozik. A fogalmi hierarchiák további tárgyalására a 4. fejezetben kerül sor.

A fogalmi hierarchiák lehetővé teszik az adatok különböző absztrakciós szinteken való kezelését, ahogy ezt a következő alfejezetben látni fogjuk.

2.2.6. | OLAP-műveletek a többdimenziós adatmodellben

Hogyan hasznosíthatók a fogalmi hierarchiák az OLAP-ban? A többdimenziós modellben az adatok értelemszerűen több dimenzió mentén szerveződnek, és minden dimenzió több absztrakciós szintet foglal magába, a fogalmi hierarchiának megfelelően. Ez a szervezés biztosítja a felhasználónak, hogy különböző távlatokból tekintsen az adatokra. Számos OLAP adatkocka-művelet létezik a különböző nézetek megvalósítására, amely a bemutatott adatok interaktív lekérdezésére és elemzésére is lehetőséget nyújt. Az OLAP tehát felhasználóbarát környezetet biztosít az interaktív adatelemzéshez.

2.8. példa | Nézzünk néhány tipikus, többdimenziós adatokon alkalmazott OLAP-műveletet. Mindegyiküket a 2.10. ábrán ábrázoltuk. Az ábra közepén látható az *ElektroMind* eladásaihoz tartozó adatkocka. A kocka a *cím*, *idő* és *árucikk* dimenziókat tartalmazza. A *cím* dimenzió a városokra, az *idő* negyedévekre, az *árucikk* pedig típus szerint van összesítve.



2.10. ábra | Többdimenziós adatok tipikus OLAP-műveletei

Erre az adatkockára a továbbiakban mint középső kockára fogunk hivatkozni. A feltüntetett mérték (ezrekben kifejezve): *dollár_eladott*. A jobb olvashatóság kedvéért csak néhány cellaértéket írtunk be a kockába. Az adatokat Chicago, New York, Toronto és Vancouver esetére vizsgáljuk.

- **Felgörgetés** – A felgörgetés művelet (amelyet *felfúrásnak* is neveznek) az összesítést egy adott dimenzió *fogalmi hierarchiáján való felfelé haladással* vagy *dimenziócsökkentéssel* hajtja végre. A 2.10. ábrán látható a középső kocka felgörgetése, amely a 2.7. ábrán megadott, *cím* dimenzióhoz tartozó fogalmi hierarchián való haladással készült. Ez a hierarchia az *utca < város < megye_vagy_tartomány < ország* teljes rendezésnek felel meg. Ez a felgörgetés a *cím* hierarchiáján a *város* szintjéről az *ország* szintjére való emelkedéssel összesíti az adatokat. Tehát az új kockában városok helyett országok szerint vannak csoportosítva az adatok.

  Amikor a felgörgetést dimenziócsökkentéssel hajtjuk végre, akkor egy vagy több dimenziót eltávolítunk az adott kockából. Vegyünk például egy értékesítési adatokat tartalmazó adatkockát *cím* és *idő* dimenziókkal. A felgörgetés végrehajtható mondjuk az *idő* dimenzió eltüntetésével, és így az összes eladás *cím* és *idő* helyett csak *cím* szerint lesz összesítve.
- **Lefúrás** – A lefúrás a felgörgetés ellenkezője: a kevésbé részletes adatoktól a még részletesebb adatokig halad. A lefúrást vagy egy adott dimenzió *fogalmi hierarchiáján való lefelé haladással*, vagy *új dimenziók bevezetésével* hajthatjuk végre. A 2.10. ábrán látható egy lefúrás a középső kockán, amely az *idő* dimenzióhoz megadott *nap < hónap < negyedév < év* fogalmi hierarchián lefelé haladással készült. A *negyedév* szintjéről a részletesebb *hónap* szintjére léptünk. Az eredményül kapott adatkocka az összes eladást negyedévek helyett hónapokra bontja le.

  Mivel a lefúrás több részletet ad hozzá az adatokhoz, ez úgy is elérhető, ha új dimenziókat veszünk fel az adatkockába. Például a 2.10. ábra középső kockáján úgy is végrehajthatunk lefúrást, ha bevezetjük a *vásárló_típus* dimenziót.
- **Szeletelés és kockázás** – A *szeletelés* művelet egy kiválasztást hajt végre az adott kocka egyik dimenzióján, és ennek az eredménye egy alkocka lesz. A 2.10. ábrán látható szeletelés a középső kocka *idő* dimenzióján hajtja végre a kiválasztást az *idő* = „Q1" feltétel szerint. A *kockázás* művelet kettő vagy több dimenzión hajt végre kiválasztást, amelynek az eredménye szintén egy alkocka. A 2.10. ábrán látható kockázás a következő, három dimenziót magába foglaló feltétel szerint történt: (*cím* = "*Toronto*" **or** "*Vancouver*") **and** (*idő* = "*Q1*" **or** "*Q2*") **and** (*árucikk* = "*szórakoztató*" **or** "*számítógép*").
- **Pivotálás (forgatás)** – A *pivotálás* (vagy *forgatás*) képi művelet, amely elforgatja az adattengelyeket, hogy az adatokat másmilyen ábrázolásban lássuk. A 2.10. ábrán látható pivotálás egy kétdimenziós szelet *árucikk* és *cím* tengelyét cserélte fel. Példaként említhető még egy háromdimenziós kocka tengelyeinek az elforgatása vagy kétdimenziós síkokba való transzformálása.
- **Más OLAP-műveletek** – Néhány OLAP-rendszer további fúróműveleteket is lehetővé tesz. Például az **átfúrás** olyan lekérdezéseket hajt végre, amelyek egynél több ténytáblázatot foglalnak magukba. A **keresztülfúrás** művelete pedig a relációs SQL-lehetőségeket hasznosítva keresztülfúrja az adatkocka alapját egészen a háttérben lévő relációs táblázatokig.

OLAP-műveletek közé tartozhat még listák első vagy utolsó *N* elemének osztályozása, mozgó átlag, gazdaságnövekedési mutatók, kamatok, belső haszonkulcs, értékcsökkenés, valutakonverzió és statisztikai függvények számítása. ■

Az OLAP-rendszerek analitikus modellező képességekkel is rendelkezhetnek, például olyan számolómotorral, amely arányok, szórások és így tovább származtatására, valamint több dimenziót átfogó mértékek számítására is alkalmas. Ez összegzések, összesítések és hierarchiák generálására is használható, mindenféle finomsági szinten és bármely dimenziókombinációban. Az OLAP előrejelzések készítésére, trendek elemzésére és statisztikai elemzésekre alkalmas funkcionális modelleket is támogat. Ebben a tekintetben egy OLAP-motor hatékony adatelemző eszköz is egyben.

OLAP-rendszerek kontra statisztikai adatbázisok

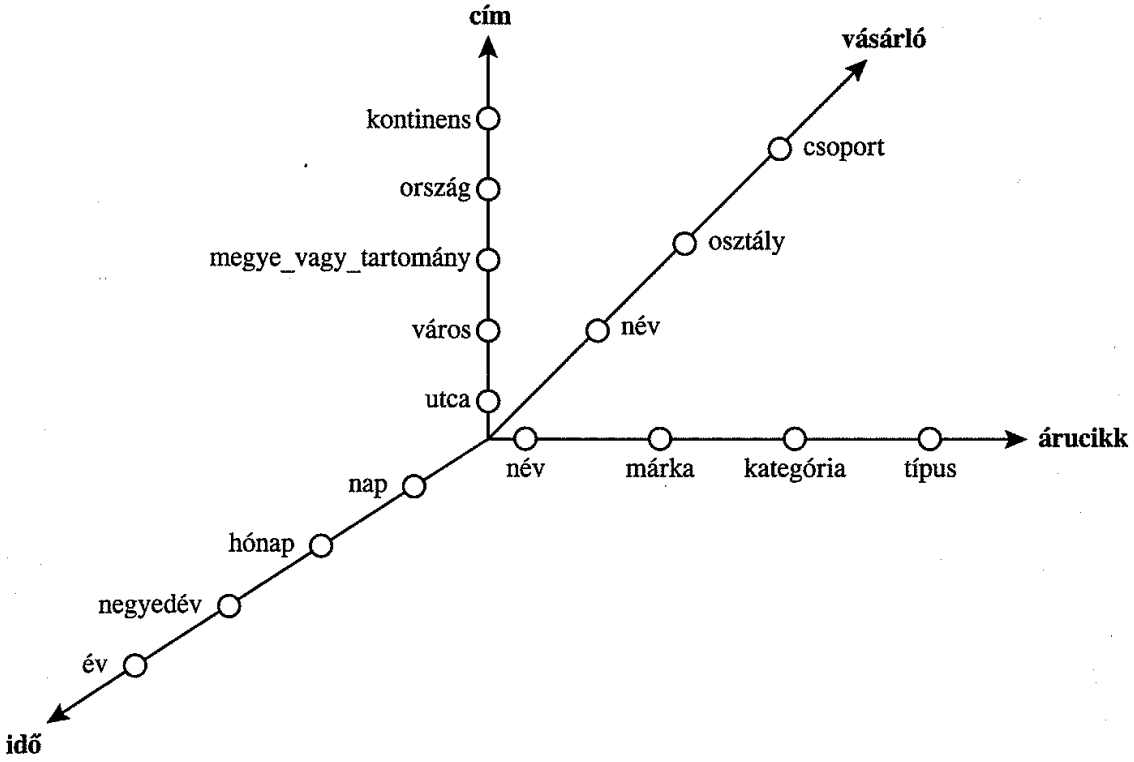
Az OLAP-rendszerek sok jellemzője, például a többdimenziós adatmodell és fogalmi hierarchiák használata, mértékek társítása dimenziókhoz, felgörgetés és lefúrás, szintén megtalálható a statisztikai adatbázisokkal (SAB) kapcsolatban végzett tevékenységek között. **Statisztikai adatbázisnak** nevezzük az olyan adatbázisrendszert, amelyet statisztikai alkalmazások támogatására terveztek. A két rendszer közötti hasonlóságok tárgyalására ritkán kerül sor, főleg a terminológia és az alkalmazási területek eltérése miatt.

Az OLAP- és a SAB-rendszerek között azonban különbségek is fellelhetők. A SAB-ok inkább szociogazdasági alkalmazásokra összpontosítanak, míg az OLAP-rendszerek üzleti alkalmazásokban használatosak. A SAB-rendszerekben a fogalmi hierarchiákkal kapcsolatos adatvédelmi kérdések jelentik a fő problémát. Összegzett szociogazdasági adatok esetén például vitatott kérdés, hogy megengedjük-e a felhasználóknak a hozzáférést az alacsonyabb szintű adatokhoz is. Végül azt is megjegyezzük, hogy az OLAP-rendszereket, ellentétben a SAB-rendszerekkel, nagy mennyiségű adatok hatékony kezelésére tervezték.

2.2.7. | Csillaghálós lekérdező modell többdimenziós adatbázisok lekérdezéséhez

A többdimenziós adatbázisok lekérdezése **csillaghálós modellen** is alapulhat. A csillaghálós modell a középpontból kiinduló, sugárirányú vonalakból áll, amelyek mindegyike egy-egy dimenzióhoz tartozó fogalmi hierarchiát ábrázol. A hierarchiában minden absztrakciós szintet **lábnyomnak** nevezünk. Ezek az OLAP-műveletek, például lefúrás vagy felgörgetés által használható finomsági szinteket jelölik.

2.9. példa | Az *ElektroMind* adattárházához tartozó csillaghálós lekérdező modell látható a 2.11. ábrán. Ez a csillagháló négy sugárvonalból áll, amelyek a *cím*, *vásárló*, *árucikk* és *idő* dimenziókhoz tartozó fogalmi hierarchiákat jelölik. Minden vonal az absztrakciós szinteket ábrázoló lábnyomokból áll. Az *idő* vonalon például négy lábnyom található: *nap*, *hónap*, *negyedév* és *év*. Egy fogalmi hierarchia egyetlen attribútumot (például *dátum* az *idő* hierarchiá-



2.11. ábra | Üzleti lekérdezések modellezése: csillaghálós modell

hoz) vagy több attribútumot (például a *cím* hierarchiához az *utca, város, megye_vagy_tartomány* és *ország* attribútumok) foglalhat magába. Az *ElektroMind* eladásainak a vizsgálatához a felhasználók például *hónap*ról *negyedév*re görgethetik az adatokat az *idő* dimenzió mentén, vagy mondjuk az *ország*tól kiindulva lefúrást végezhetnek a *város*okig a *cím* dimenzió mentén. A fogalmi hierarchiák használhatók az adatok **általánosítására**, ha az alacsony szintű értékeket (például *nap* az *idő* dimenzióban) magasabb szintű absztrakciókkal (például év) váltjuk fel, vagy az adatok **részletezésére**, ha a magasabb szintű absztrakciókat alacsonyabb szintű értékekkel cseréljük fel. ■

2.3. | Adattárház-architektúra

Ebben az alfejezetben az adattárház felépítését érintő kérdéseket tárgyaljuk meg. A 2.3.1. alfejezet egy általános leírást ad arról, hogy hogyan tervezzünk és építsünk adattárházakat. A 2.3.2. alfejezet háromszintű adattárház-architektúrát mutat be. A 2.3.3. alfejezetben különböző típusú, OLAP-feldolgozáshoz használható adattárház szerverekkel ismerkedhetünk meg.

2.3.1. | Adattárház tervezésének és építésének lépései

Ebben az alfejezetben az adattárház tervezéséhez használható üzleti elemző keretrendszert mutatjuk be. A tervezési folyamat alaplépéseit is leírjuk.

Adattárház tervezése: egy üzleti elemző keretrendszer

Mit nyújt az adattárház az üzleti elemzők számára? Egy adattárház először is *versenyelőnyt* jelenthet, mivel olyan információkat képes nyújtani, amelyek alapján mérhető a teljesítmény, és fontos kiigazítások tehetők, amelyek segítségével a versenytársakat le lehet győzni. Másodsorban fokozhatja a *termelékenységet*, hiszen az adattárház képes gyorsan és hatékonyan összegyűjteni azokat az információkat, amelyek pontosan leírják a vállalatot. Harmadsorban lehetővé teszi a *vásárlói kapcsolattartás menedzselését*, mivel következetes képet ad a vásárlókról és az árucikkekről minden üzleti vonalon, minden osztályon és minden piacon. Végül az adattárház használata *megtakarítást* is eredményezhet a vállalaton belül, hiszen hosszú időn keresztül, következetesen és hatékonyan követi nyomon az előforduló tendenciákat, mintákat és kivételeket.

Hatékony adattárház megtervezéséhez meg kell értenünk és elemeznünk kell az üzleti elvárásokat, és létre kell hoznunk egy üzleti elemző keretrendszert. Nagy és összetett információs rendszer megépítése egy nagy és összetett épület felépítéséhez hasonló, ahol a tulajdonos, az építész és a kivitelező mind másképp látják a dolgokat. A különböző nézeteket egy komplex keretrendszerré dolgozzuk össze, amely képviseli a felülről lefelé irányuló, üzletmotivált vagy tulajdonosi szempontot csakúgy, mint az alulról felfelé irányuló, építésmotivált vagy kivitelezői szempontot.

Az adattárház tervezését illetően négy különböző nézőpontot kell figyelembe venni: *felülről lefelé, adatforrás-, adattárház-* és *üzleti lekérdezés nézet*.

- A **felülről lefelé nézet** lehetővé teszi az adattárházhoz szükséges, lényeges információk kiválasztását. Ez az információ megfelel az aktuális és a későbbi üzleti igényeknek.
- Az **adatforrásnézet** megvilágítja azokat az információkat, amelyeket az operatív adatbázisrendszerek megragadnak, tárolnak és kezelnek. Ezt az információt különböző részletességgel és pontossággal dokumentálhatják, az egyedi adatforrás-táblázatoktól az egyesített adatforrás-táblázatokig. Az adatforrásokat gyakran hagyományos adatmodellező technikákkal modellezik, például egyed–kapcsolat modellel vagy CASE- (Computer-Aided Software Engineering, számítógéppel támogatott szoftvertervezés) eszközökkel.
- Az **adattárháznézet** ténytáblázatokat és dimenziótáblázatokat tartalmaz. Ez a nézet fejezi ki az adattárházon belül tárolt információkat, amelybe az előre számított végösszegek és megszámlálások éppúgy benne foglaltatnak, mint a történeti áttekintést biztosító információk az adatok eredetét illetően (adatforrás, dátum, idő).
- Az **üzleti lekérdezés nézet** a végfelhasználó szempontjából tükrözi az adattárházban található adatokat.

Egy adattárház felépítése és használata összetett feladat, mivel *üzleti, technológiai* és *programkezelői jártasságot* is igényel. Az *üzleti jártasság* szempontjából az adattárház építése azt jelenti, hogy meg kell érteni, hogy az ilyen rendszerek hogyan tárolják és kezelik az adatokat, hogyan kell **adatkinyerőket** létrehozni, amelyek az operatív adatbázisrendszerből az adattárházba helyezik át az adatokat, és hogyan kell **tárházfrissítő szoftvereket** készíteni, amelyek ésszerű határokon belül, az operatív adatbázisrendszer adatai alapján frissítik az adattárház adatait. Az adattárház használatához hozzátartozik, hogy tisztában legyünk a benne megtalálható adatok jelentőségével, továbbá hogy értel-

mezzük és le is tudjuk fordítani az üzleti igényeket olyan lekérdezések formájába, amelyeket az adattárház képes kielégíteni. A *technológiai jártasságot* illetően az adatelemzőknek érteniük kell a tárházban megtalálható mennyiségi adatok megfelelő kiértékeléséhez, illetve a történeti adatokból levont következtetések alapján megfelelő tények, állítások levezetéséhez. Ezek a készségek magukba foglalják különböző minták és tendenciák felismerését, történeti adatok alapján tendenciák következtetését, anomáliák és paradigmaváltások keresését, és az efféle elemzéseken alapuló következetes vezetői javaslatok kidolgozását. A *programkezelői készségek* közé tartozik, hogy képesek legyünk a sokféle technológia, forgalmazó és felhasználó között közvetíteni, közéjük interfészt alkotni úgy, hogy az eredmények felmutatása idő és költség szempontjából is hatékonyan történjen.

Az adattárház tervezésének folyamata

Hogyan tervezzük meg az adattárházat? Egy adattárház megépítéséhez használható a *felülről lefelé*, az *alulról felfelé megközelítés* vagy ezek *kombinációja*. A **felülről lefelé módszer** átfogó elgondolással és tervezéssel indul. Ez olyan esetekben hasznos, ahol kiforrott és jól ismert technológiák állnak rendelkezésre, és ahol a megoldásra váró üzleti problémák világosak és jól értelmezhetők. Az **alulról felfelé módszer** alkalmazása kísérletekkel és prototípusokkal kezdődik. Ez az üzletmodellezés és technológiafejlesztés korai szakaszában hasznosítható. A vállalkozás előrejutását jóval olcsóbbá teszi, és a technológiákból származó előnyök kiértékelését is biztosítja még a jelentősebb elkötelezettségek vállalása előtt. A **kombinált módszerrel** kihasználhatjuk a felülről lefelé módszer tervszerű és stratégiai természetét, megőrizve eközben az alulról felfelé módszer opportunista megközelítését és gyors megvalósíthatóságát.

A szoftverfejlesztő szempontjából az adattárház tervezése és felépítése a következő lépésekből állhat: *tervezés, követelménytanulmány, problémaelemzés, tárháztervezés, adategyesítés és tesztelés*, és végül *az adattárház telepítése*. Nagy programrendszereket kétféle módszer felhasználásával lehet fejleszteni: a *vízesés módszerrel* vagy a *spirál módszerrel*. A **vízesés módszer** minden továbblépés előtt végrehajt egy strukturált, szisztematikus analízist, és csakúgy, mint a vízesés, lépésről lépésre halad előre. A **spirál módszer** egyre funkcionálisabb rendszerek gyors generálásából áll, az egymást követő verziók közötti rövid szünetekkel. Adattárházak, de különösen adatpiacok fejlesztéséhez ezt jó választásnak szokták tekinteni, mert a verzióváltási idő rövid, a módosításokat gyorsan el lehet végezni, és az új elgondolásokat illetve technológiákat időben lehet alkalmazni.

A tárháztervezés folyamata általában a következő lépésekből áll:

(1) A modellezni kívánt *üzletfolyamat* kiválasztása, például megrendelések, számlák, szállítások, raktározás, számlaügyintézés, értékesítés, általános főkönyv. Ha ez egy szervezeti szintű üzletfolyamat, és több, összetett objektumgyűjteményt is magába foglal, akkor az adattárházmodellt kell követni. Ha azonban ez osztály, illetve részleg szintű folyamat, akkor az adatpiacmodellt kell választani.
(2) Az üzletfolyamat *magjának* a megválasztása. A mag a folyamathoz tartozó ténytáblázatban ábrázolni kívánt adat alapvető, atomi szintje, például egyes tranzakciók, egyes napi kivonatok és így tovább.
(3) A *dimenziók* megválasztása, amelyek az egyes ténytáblázatokra vonatkoznak. Jellegzetes dimenziók az idő, árucikk, vásárló, szállító, raktár, tranzakciótípus, illetve státus.

(4) A *mértékek* kiválasztása a ténytáblázatokhoz. Tipikus mértékek a numerikus additív mennyiségek, mint *dollár_eladott* és *darab_eladott*.

Mivel egy adattárház megépítése nehéz és hosszú távú munka, ezért az implementáció határvonalait tisztán és pontosan kell látni. A kezdő adattárház megvalósításának a céljait *jól körül kell írni*, továbbá ezeknek *elérhetőknek* és *mérhetőknek* kell lenniük. Ez magába foglalja a ráfordítandó idő és a költségvetési lehetőségek meghatározását, a vállalkozás modellezni kívánt részlegének a megadását, a kiválasztott adatforrások számának a felállítását, és a kiszolgált részlegek számának és típusának a meghatározását.

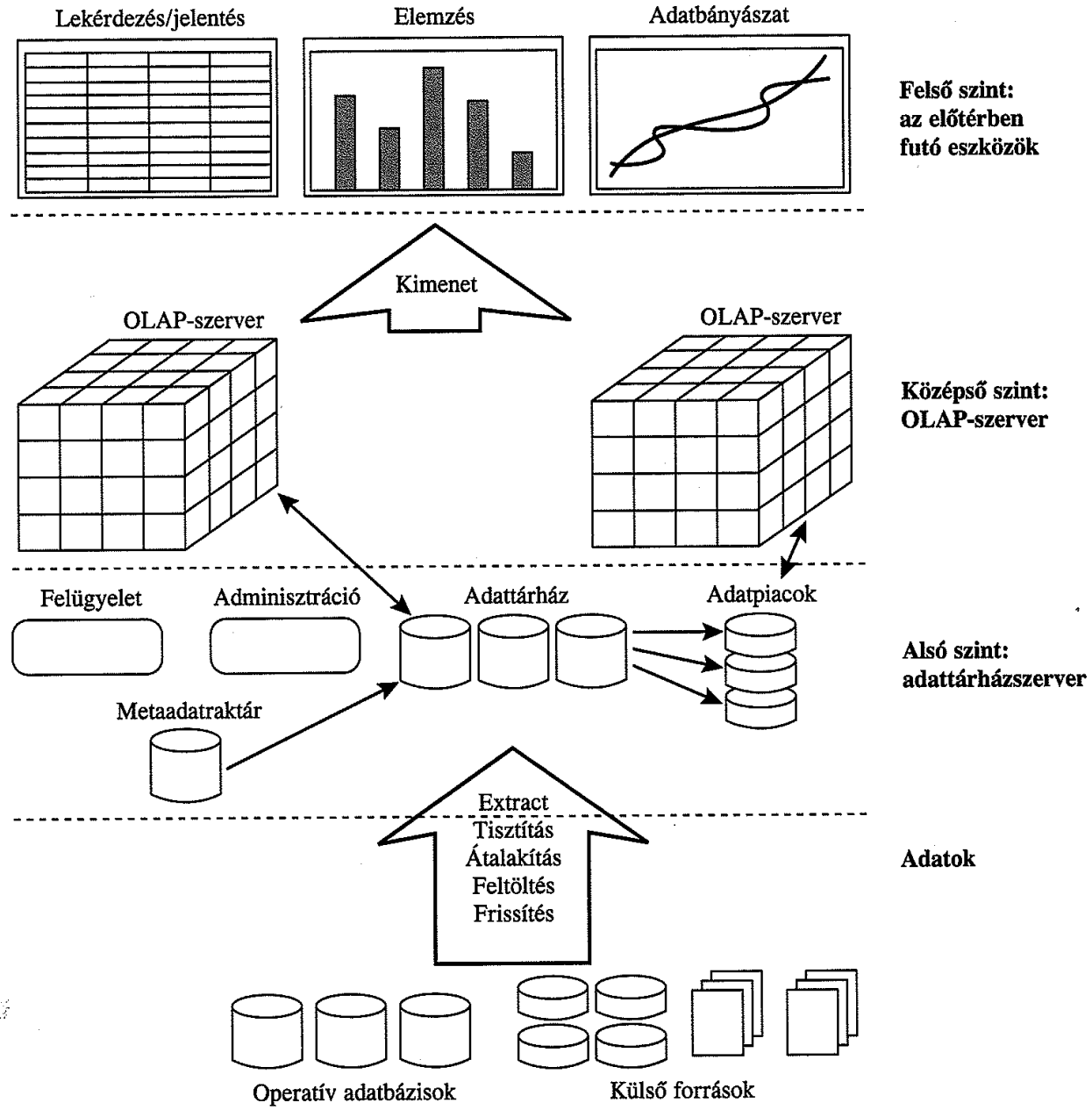
Ha az adattárházat megtervezték és létrehozták, az első telepítéskor sor kerül a kezdő installálásra, az üzembe helyezési terv kidolgozására és a tréningekre, illetve tájékoztatásra. A platform bővítésére és karbantartására is gondolni kell. Az adattárház felügyeletéhez hozzátartozik az adatok frissítése, az adatforrások szinkronba hozása, katasztrófaterv kidolgozása, hozzáférési jogok és biztonsági kérdések kezelése, az adatnövekedés kezelése, az adatbázis teljesítményének menedzselése, valamint az adattárház fejlesztése és kiterjesztése. Az alkalmazási terület kezelése magába foglalja a lekérdezések, dimenziók és jelentések kiterjedésének és számának a szabályozását, az adattárház méretének, illetve az ütemezés, a költségvetés és az erőforrások korlátozását.

Különböző típusú adattárház-tervező eszközök léteznek már. Az **adattárház-fejlesztő eszközök** olyan funkciókat biztosítnak, amelyek segítségével a metaadattárak tartalma (például sémák, parancssorok, szabályok) határozható, illetve változtatható meg, lekérdezésekre lehet választ adni, jelentéseket lehet készíteni, és metaadatokat lehet relációs adatbázisrendszer katalógusokból, illetve azokba áthelyezni. A **tervező és elemző eszközök** a séma változtatások hatását és a frissítés teljesítményének a megváltozását vizsgálják a frissítési arányok és időközök változtatásának a függvényében.

2.3.2. | A háromszintű adattárház felépítése

Milyen az adattárház felépítése? Az adattárházak gyakran a 2.12. ábrán látható háromszintű felépítést alkalmazzák.

(1) Az alsó szint a **tárházadatbázis-szerver**, amely majdnem mindig egy relációs adatbázisrendszer. *Hogyan nyerjük ki erről a szintről az adatokat, hogy az adattárház létrejöhessen?* Az operatív adatbázisokból és a külső forrásokból az adatokat **átjárókként** ismert alkalmazói programinterfészek felhasználásával nyerjük ki. Az átjárót a mögöttes ABKR támogatja. A kliensprogramok számára lehetővé teszi, hogy az általuk generált SQL-kódok egy szerveren lefussanak. Példaként említhető az ODBC (Open Database Connection, nyílt adatbázis-összekapcsolhatóság) és az OLE-DB (Open Linking and Embedding for Databases, objektumcsatolás és -beillesztés adatbázisokhoz) a Microsofttól, és a JDBC (Java Database Connection, Java adatbázis-összekapcsolhatóság).
(2) A középső szint egy **OLAP-szerver**, amely jellemzően vagy egy **relációs OLAP (Relational OLAP, ROLAP)** -modell használatával van implementálva, vagy egy **többdimenziós OLAP (Multidimensional OLAP, MOLAP)** -modell segítségével.



2.12. ábra | A háromszintű adattárház architektúrája

Az első esetben egy kiterjesztett relációs ABKR-t kapunk, amely a többdimenziós adatok műveleteit szabványos relációs műveletekre képezi le. A második esetben egy sajátos célú szerverhez jutunk, amely közvetlenül implementálja a többdimenziós adatokat és a hozzájuk tartozó műveleteket. Az OLAP-szerverekről a 2.3.3. alfejezetben esik bővebben szó.

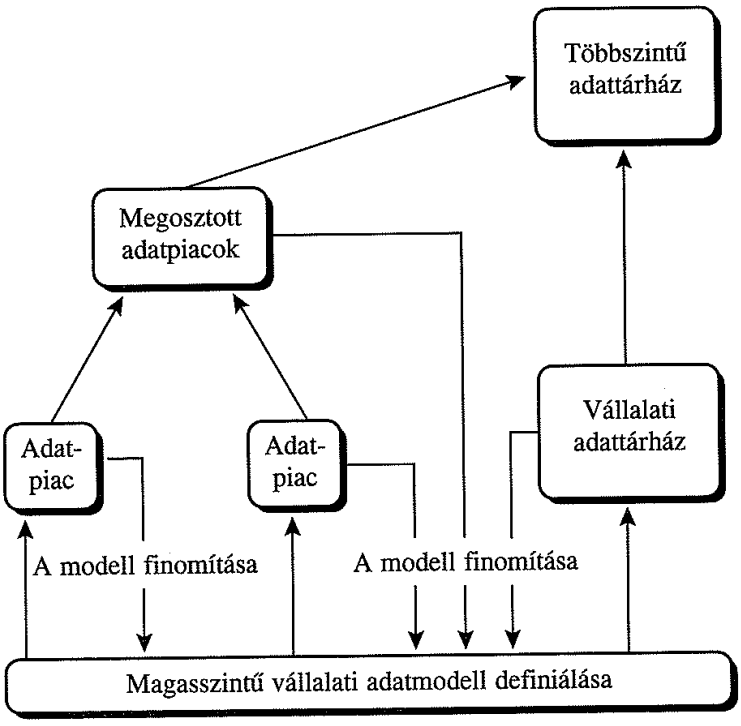
(3) A felső szint egy **kliens**, amely lekérdező és jelentéskészítő eszközöket, elemző és/vagy adatbányász eszközöket (például tendenciaelemzés, előrejelzés és így tovább) tartalmaz.

Az architektúra szempontjából három adattárházmodell létezik: a *vállalati tárház*, az *adatpiac* és a *virtuális tárház*.

- **Vállalati tárház** – A vállalati tárház az egész szervezeten átívelő témákról gyűjti össze az információkat. Vállalati szintű adatintegrációt biztosít, általában egy vagy több operatív adatbázisrendszerről vagy külső információforrásból, működési terében összetett. Jellemzően részletes és összegzett adatokat is tartalmaz, és méretét tekintve néhány gigabájttól néhány száz gigabájtig, terabájtig vagy még tovább terjedhet. Egy vállalati tárházat hagyományos nagyszámítógépeken, Unix-szuperszervereken vagy párhuzamos felépítésű felületeken is lehet implementálni. A tárház tervezése és felépítése kiterjedt üzletmodellezést igényel és akár évekig is eltarthat.
- **Adatpiac** – Az adatpiac a vállalati méretű adatok olyan részhalmazát tartalmazza, amely egy speciális felhasználócsoport számára értékes. Az adatpiac csak speciális, kiválasztott témákra szorítkozik. Például egy marketing-adatpiac témái között csak a vásárló, árucikk és az értékesítés fog szerepelni. Az adatpiacban tárolt adatok általában összegzett formában jelennek meg.

  Az adatpiacokat a legtöbb esetben alacsony költségű, Unix vagy Windows/NT alapú szervereken implementálják. A telepítési időszak inkább hetekben mérhető, mint hónapokban vagy években. Ez azonban hosszú távon bonyolult egyesítési folyamatokat is magába foglalhat, ha az elgondolás és a tervezés nem ölelte fel az egész vállalatot.

  Az adatforrástól függően az adatpiacok két típusát különböztetjük meg: függő és független. *Független* adatpiac esetén az adatokat egy vagy több operatív adatbázisrendszer vagy külső információforrás által megragadott adatokból veszik át, vagy helyileg, egy bizonyos részlegen vagy földrajzi területen belül generálják. *Függő* adatpiac esetén az adatok közvetlenül a vállalati adattárházból jönnek.
- **Virtuális tárház** – A virtuális tárház operatív adatbázisok nézettábláinak egy halmaza. A lekérdezések hatékony kezelése miatt a lehetséges összegző nézettáblákból csak néhányat valósítanak meg. Virtuális tárházat könnyű építeni, de az operatív adatbázisszerverektől túl sok kapacitást követel meg.



2.13. ábra | Az adattárház fejlesztéséhez javasolt megközelítés

A vállalati tárház felülről lefelé fejlesztése módszeres megoldást kínál és minimalizálja az integrációs problémákat. Ezzel szemben drága, hosszú ideig tart, és az egész szervezetre kiterjedő közös, konzisztens adatmodell felállításának a nehézségei miatt egyáltalán nem rugalmas. Az adatpiacok tervezéséhez, fejlesztéséhez és telepítéséhez alkalmazott alulról felfelé megközelítés rugalmasságot, alacsony költségeket és a befektetés gyors megtérülését biztosítja. Problémákhoz vezethet azonban, ha több különálló, eltérő adatpiacot kell egyesíteni egy konzisztens vállalati adattárházzá.

Az adattárházrendszerek fejlesztéséhez javasolható a megvalósítás egy növekményes, evolúciós jellegű módja, amely a 2.13. ábrán látható. Először egy magasszintű vállalati adatmodellt definiálunk viszonylag rövid időn belül (egy-két hónap), amely szervezeti szintű, konzisztens, egyesített rálátást biztosít az adatokra a különböző témák és lehetséges felhasználási lehetőségek között. Ez a magasszintű modell nagymértékben csökkenti a későbbi integrációs problémákat, bár a vállalati adattárház és az osztály szintű adatpiacok további fejlesztése során még finomításra szorul. A második lépésben ezt az adatmodellt felhasználva a független adatpiacok és a vállalati adattárház megvalósítására egymással párhuzamosan kerül sor. A harmadik lépésben jönnek létre a megosztott adatpiacok, amelyek az eltérő adatpiacokat fejszervereken (hub) keresztül egyesítik. Végül létrejön a **többszintű adattárház**, ahol a vállalati tárház az egyetlen őre minden tárházadatnak, amelyek később a függő adatpiacok alapját képezik.

2.3.3. | Az OLAP-szerverek típusai: ROLAP, MOLAP, HOLAP

Milyen OLAP-szerverek léteznek? Logikusan, az OLAP-szerverek az üzleti felhasználóknak adattárházak vagy adatpiacok többdimenziós adatait biztosítják, anélkül, hogy a felhasználónak törődnie kéne azzal, hogy ezeket az adatokat hogyan vagy hol tárolják. Az OLAP-szerverek fizikai felépítése és megvalósítása azonban nem háríthatja el magától ezeket a kérdéseket. Az OLAP-feldolgozáshoz használható tárházszerverek fajtái a következő lehetnek:

- **Relációs OLAP (ROLAP) -szerverek** – Ezek azok a köztes szerverek, amelyek a háttérben működő relációs szerver és a kliens előtérben futó eszközei között állnak. Relációs vagy kiterjesztett relációs ABKR-t használnak a tárházadatok tárolására és kezelésére, és OLAP köztes szoftvereket a hiányzó részek támogatására. A ROLAP-szerverekhez hozzátartozik egy optimalizációs egység a háttérben lévő ABKR-hez, az összesítésnavigáló logika megvalósítása, és további eszközök és szolgáltatások. A ROLAP-technológia általában jobban skálázható, mint a MOLAP-technológia. A Microstrategy DSS-szervere és az Informix Metacube például a ROLAP-megközelítést alkalmazza.[4]
- **Többdimenziós OLAP (MOLAP) -szerverek** – Ezek a szerverek vektor alapú, többdimenziós tárolási algoritmusokon keresztül az adatok többdimenziós nézeteit támogatják. A többdimenziós nézeteket közvetlenül adatkocka vektorszerkezetbe képezik le. Az Arbor Essbase szervere például MOLAP-szerver. Az adatkocka használatának az az előnye, hogy lehetővé teszi az előszámított összegzett adatok gyors indexelését.

4 | A termékekről további információ a *www.microstrategy.com,* illetve a *www.informix.com* címeken található.

Megjegyezzük, hogy a többdimenziós adattárolás használatával a tár kihasználtsága ritka adatok esetén alacsony lehet. Ilyen esetekben érdemes a ritka mátrixok tömörítő technikáit (lásd 2.4. alfejezet) megvizsgálni.

Sok MOLAP-szerver használ kétszintű tárábrázolást a ritka és a sűrű adatok kezelésére. A sűrű alkockákat vektorszerkezetként kezelik és tárolják, a ritka alkockákra pedig tömörítési technológiákat alkalmaznak a hatékony tárkihasználás érdekében.

- **Hibrid OLAP (Hybrid OLAP, HOLAP) -szerverek** – A hibrid OLAP-megközelítés kombinálja a ROLAP- és a MOLAP-technológiákat, kihasználva ezzel a ROLAP skálázhatóságát és a MOLAP gyorsabb számítási képességét. Egy HOLAP-szerver például lehetővé teheti nagymennyiségű részletes adat relációs adatbázisban való tárolását, miközben az összesítéseket külön MOLAP-tárban tartja. A Microsoft Server 7.0 OLAP Services hibrid OLAP-szervert támogat.
- **Szakosított SQL-szerverek** – A relációs adatbázisok OLAP-feldolgozása iránti igények kielégítése érdekében néhány relációs és adattárházas cég (például Redbrick of Informix) olyan szakosított SQL-szervereket alkalmaz, amelyek fejlett lekérdező nyelvet biztosítanak, és csillag- és hópehelysémák feletti SQL-lekérdezések feldolgozását is támogatják csak olvasó környezetben.

Nos, valójában hogyan is vannak tárolva az adatok ROLAP-, illetve MOLAP-felépítésben? Ahogy ez a nevéből is következik, a ROLAP-megközelítés relációs táblázatokat használ az adatok tárolására az on-line analitikus feldolgozáshoz. Emlékezzünk vissza, hogy az alapkockának megfelelő ténytáblázatot *alap ténytáblázatnak* neveztük. Az alap ténytáblázat az adott adatkockához tartozó sémában az összekapcsoló kulcs által jelölt absztrakciós szinten tárol adatokat. Összesített adatok is tárolhatók a ténytáblázatokban, amelyeket ilyenkor **összefoglalt ténytáblázatoknak** nevezünk. Néhány összefoglalt ténytáblázat tárolhat alap tényadatokat és összesített adatokat is, ahogy ezt a 2.10. példában látjuk. Egy másik megoldás, ha külön összefoglalt ténytáblázatokat használunk minden absztrakciós szinthez, és csak összesített adatokat tárolunk bennük.

2.10. példa | A 2.4. táblázat egy összefoglalt ténytáblázatot mutat, amely alap tényadatokat és összesített adatokat is tartalmaz. A táblázat sémája: ⟨ *sor_ID (SID), árucikk, ..., nap, hónap, negyedév, év, dollár_eladott* (azaz az eladások összege) ⟩, ahol *nap, hónap, negyedév* és *év* az értékesítés időpontját adja meg. Tekintsük az 1001 és 1002 azonosítóval rendelkező sorokat. Az itt található adatok az alaptény szintjén vannak; az értékesítések időpontja pedig 2000. október 15., illetve 2000. október 23. Most nézzük az 5001-es sort. Ez a sor egy általánosabb absztrakciós szinten van, mint az 1001-es és 1002-es sorok. A *nap* érté-

2.4. táblázat | Egyetlen táblázat alap tényadatok és összesített adatok tárolására

SID	árucikk	...	nap	hónap	negyedév	év	dollár_eladott
1001	tv	...	15	10	Q4	2000	250,60
1002	tv	...	23	10	Q4	2000	175,00
...							
5001	tv		**minden**	10	Q4	2000	45 786,08
...		...					

két **minden**-re általánosították, és így a megfelelő *idő* érték 2000 októbere. Ez azt jelenti, hogy az itt látható *dollár_eladott* érték egy összesített érték, amely az egész hónapra (2000 októbere) vonatkozik, nem pedig csak október 15-re vagy 23-ra. A különleges **minden** értéket használjuk az összegzett adatokban a részösszegek megjelenítésére. ■

A MOLAP-megközelítés többdimenziós vektorszerkezetet használ az adatok tárolására az on-line analitikus feldolgozáshoz. Az adatkocka-szerkezet például, amit ebben a fejezetben írtunk le, és a fejezet során sokszor hivatkoztunk rá, ilyen vektorszerkezet.

A legtöbb adattárházrendszer kliens–szerver felépítést használ. A relációs adattár mindig az adattárház-/adatpiacszerver munkaállomásán található meg. A többdimenziós adattár vagy az adatbázisszerveren vagy a kliens munkaállomásán lelhető fel.

2.4. | Adattárházak megvalósítása

Az adattárházak óriási mennyiségű adatot tartalmaznak. Az OLAP-szerverek azonban megkövetelik, hogy a döntéstámogató lekérdezések másodpercek alatt eredményt hozzanak. Ezért az adattárházak részéről elengedhetetlen, hogy kiemelkedően hatékony kockaszámító technológiákat, hozzáférési módszereket és feldolgozó technikákat támogassanak. *Hogyan lehet ezt elérni?* Ebben az alfejezetben az adattárházrendszerek hatékony megvalósítási módszereit vizsgáljuk.

2.4.1. | Adatkockák hatékony számítása

A többdimenziós adatok elemzésének mélyén a sokféle dimenzió mentén vett összesítések hatékony számítása nyugszik. SQL-es szóhasználattal élve ezeket az összesítéseket **csoportosításoknak** nevezzük.

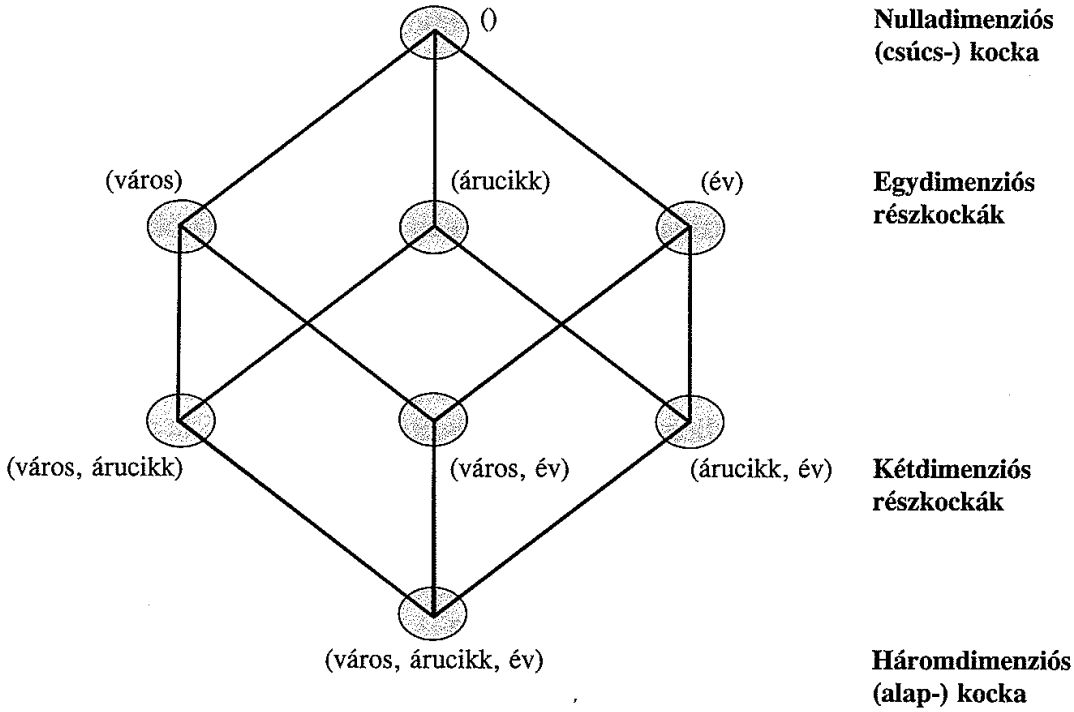
A compute cube művelet és megvalósítása

A kockaszámítások egy megközelítése kiterjeszti az SQL-t, hogy az a **compute cube** műveletet is tartalmazza. A **compute cube** művelet a megadott dimenziók minden részhalmazán kiszámítja az összesítéseket.

2.11. példa | Tegyük fel, hogy az *ElektroMind* eladásaihoz létre akarunk hozni egy adatkockát, amely a következőket tartalmazza: *árucikk, város, év* és *eladások_dollárban*. Az adatokat elemezni is szeretnénk, a következőkhöz hasonló lekérdezésekkel:

- *Számítsuk ki az eladások összegét árucikk és város szerint csoportosítva.*
- *Számítsuk ki az eladások összegét az árucikkek szerint csoportosítva.*
- *Számítsuk ki az eladások összegét városok szerint csoportosítva.*

Összesen hány részkockát vagy csoportosítást lehet kiszámítani ehhez az adatkockához? Ha a három attribútumot (*város, árucikk* és *év*) dimenzióként vesszük, és hozzátesszük még



2.14. ábra | Egy háromdimenziós adatkockát alkotó részkockaháló. Minden részkocka egy másféle csoportosítást jelöl. Az alapkocka három dimenziót tartalmaz: *város*, *árucikk* és *év*

az *eladások_dollárban*t mint mértéket, akkor a kiszámítható részkockák vagy csoportosítások száma $2^3 = 8$. A lehetséges csoportosítások a következők: {(*város, árucikk, év*), (*város, árucikk*), (*város, év*), (*árucikk, év*), (*város*), (*árucikk*), (*év*), ( )}, ahol ( ) azt jelenti, hogy a csoportosítás üres (azaz a dimenziók nincsenek csoportosítva). Ezek a csoportosítások egy részkockahálót alkotnak, ahogy ez a 2.14. ábrán látható. Az alapkocka mind a három dimenziót (*város, árucikk, év*) tartalmazza. Vissza tudja adni az eladások összértékét a három dimenzió bármely kombinációjában. A csúcskocka vagy nulladimenziós részkocka arra az esetre vonatkozik, amikor a csoportosítás üres. Ez az összes eladás összértékét adja meg. ■

Az olyan SQL-lekérdezés, amely nem tartalmaz csoportosítást (például „számítsuk ki az összes eladás összértékét") *nulladimenziós művelet*. Az az SQL-lekérdezés, amely egyetlen csoportosítást tartalmaz (például „számítsuk ki az eladások összegét városokra lebontva") *egydimenziós művelet*. Egy *n* dimenzión vett kockaművelet ekvivalens a **group by** parancsok egy készletével – az *n* dimenzió minden részhalmazához egy **group by** tartozik. Vagyis a kockaművelet a **group by** művelet *n*-dimenziós általánosítása. A 2.2.3. alfejezetben bevezetett DMQL-szintaxisa alapján a 2.11. példa adatkockája így definiálható:

define cube értékesítés [árucikk, város, év]: **sum**(eladások_dollárban)

Egy *n*-dimenziós kockához 2^n részkocka tartozik, az alapkockát is beszámítva. A következő parancs

compute cube értékesítés

határozottan arra utasítja a rendszert, hogy az {*árucikk, város, év*} halmaz mind a nyolc részhalmazához, az üres halmazt is beleértve, számítsa ki az értékesítési összesítő részkockákat. A kockaszámító műveletet először [GCB+97] javasolta és tanulmányozta.

Az on-line analitikus feldolgozás során előfordulhat, hogy különböző lekérdezéseknél különböző részkockákhoz kell hozzáférni. Ezért jó ötletnek tűnik az összes részkockát, vagy legalább egy részüket előre kiszámítani. Az előszámítás gyors válaszidőt eredményez, és segítségével felesleges számítások is elkerülhetők. Tulajdonképpen ha nem is mindegyik, de a legtöbb OLAP-termék valamilyen fokon igénybe veszi a többdimenziós összesítések előszámítását.

Az előszámítással kapcsolatos fő probléma azonban az, hogy ha az adatkockában minden részkockát előre kiszámítunk, akkor a tárigény robbanásszerűen megnő, különösen ha sok dimenzióhoz tartozik többszintű hierarchia.

Hány részkocka van egy n-dimenziós adatkockában? Ha a dimenziókhoz nem tartoznának hierarchiák, akkor ez a szám 2^n lenne, ahogy ezt az előbb láttuk. A gyakorlatban azonban bizony sok dimenzióhoz tartozik hierarchia. Például az *idő* dimenzió általában nem csak egyszintű, mint az *év*, hanem hierarchiát vagy hálót alkot, például *nap* < *hét* < *hónap* < *negyedév* < *év*. Az *n*-dimenziós adatkockához generálható részkockák száma (azokat a részkockákat is beleértve, amelyeket a hierarchiákon való felfelé haladással kapunk):

$$T = \prod_{i=1}^{n} (L_i + 1),$$

ahol L_i az *i*-edik dimenzióhoz tartozó szintek száma (a *virtuális* **minden** szinttől eltekintve, mivel a legfelső szintre való általánosítás a dimenzió eltávolításával egyenértékű). Ez a képlet azon alapul, hogy minden egyes dimenziónál legalább egy absztrakciós szint megjelenik egy részkockában. Ha például a kocka 10 dimenziós és minden dimenzióhoz négy szint tartozik, akkor az előállítható részkockák száma $5^{10} \approx 9{,}8 \times 10^6$.

Mostanra talán világossá vált, hogy egy adatkockából (vagy az alapkockából) előállítható összes lehetséges részkocka előszámítása és megvalósítása nem reális elképzelés. Ha sok, nagyméretű részkockánk van, akkor ésszerűbb megoldás a *részleges megvalósítás*, ami azt jelenti, hogy az összes lehetséges részkocka közül csak *néhányat* valósítunk meg.

Részleges megvalósítás: a kiválasztott részkockák számítása

Adott alapkocka esetén az adatkocka megvalósítását illetően három lehetőségünk van: (1) nem számítjuk ki előre semelyik „nem alap" részkockát (**nincs megvalósítás**); (2) az összes részkocka előszámítása (**teljes megvalósítás**); és (3) a lehetséges részkockák csak egy részhalmazát számítjuk ki előre (**részleges megvalósítás**). Az első választás a drága, többdimenziós összesítések menet közbeni számításához vezet, és ez lassú lehet. A második választás óriási méretű memóriát igényel az előszámított részkockák tárolásához. A harmadik megoldás érdekes kompromisszum a tárolási hely és a válaszidő között.

A részkockák részleges megvalósításánál három tényezőt kell figyelembe venni: (1) a megvalósítani kívánt részkockák meghatározása; (2) a megvalósított részkockák hasznosítása a lekérdezések feldolgozása alatt; és (3) a megvalósított részkockák hatékony frissítése a feltöltések és frissítések során.

A megvalósítandó részkockák kiválasztásánál figyelembe kell venni az előforduló lekérdezéseket, azok gyakoriságát és elérési költségét. Ezenkívül a munkaterhelés jellegét, a növekményes frissítés költségeit és az összes tárigényt is tekintetbe kell venni. Szintén fontolóra kell venni a fizikai adatbázisterv összefüggéseit, például az indexek generálását és kiválasztását. Több OLAP-termék heurisztikus megoldást alkalmaz a részkockák kiválasztására. Elterjedt módszer azoknak a részkockáknak a megvalósítása, amelyek gyakran hivatkozott részkockák alapjául szolgálnak.

Ha a kiválasztott részkockákat megvalósítottuk, akkor fontos, hogy a lekérdezések feldolgozása alatt ki is használjuk őket. Ehhez hozzátartozik a releváns részkocka (illetve részkockák) kiválasztása a többi megvalósított részkocka közül, annak meghatározása, hogy hogyan használjuk a megvalósított részkockákon az elérhető indexstruktúrákat, és hogy hogyan vigyük át az OLAP-műveleteket a kiválasztott részkocká(k)ra. Ezekkel a kérdésekkel a 2.4.3. alfejezet foglalkozik bővebben.

A megvalósított részkockák hatékony frissítésére is figyelni kell a feltöltések és frissítések alkalmával. Ehhez meg kell vizsgálni a párhuzamossági és a növekményes frissítési technikákat.

Többutas tömbösszesítés az adatkockák számításánál

Előfordulhat, hogy a gyors on-line analitikus feldolgozás érdekében szükség van az összes részkocka kiszámítására az adott adatkockához. A részkockákat tárolhatjuk másodlagos tárolón, és amikor szükséges, elérhetjük őket. Tehát fontos hatékony módszereket keresni a teljes megvalósítás esetére is. Ezeknek a módszereknek figyelembe kell venniük, hogy a részkockák számításához rendelkezésre álló központi memória mérete és az idő is korlátozott. Hogy egyszerűsítsük a dolgunkat, kihagyhatjuk azokat a részkockákat, amelyek a létező hierarchiákon való felfelé haladással jönnek létre.

Mivel a relációs OLAP (ROLAP) sorokat és relációs táblázatokat használ alap adatszerkezetként, míg ugyanez a többdimenziós OLAP (MOLAP) esetében a többdimenziós vektor, ezért azt várhatnánk, hogy a kettő egymástól lényegesen eltérő kockaszámító technikák után kutat.

A ROLAP-kockaszámítás a következő fő optimalizációs technikákat használja:

- A dimenzióattribútumok rendezése, tördelése és csoportosítása az egymással kapcsolatban lévő sorok újrarendezése és klaszterezése céljából.
- Mint egy „részleges csoportosító lépést", a csoportosítást valamilyen alösszesítéseken hajtjuk végre. Ezek a „részleges csoportosítások" más alösszesítések számítását gyorsítják fel.
- Az összesítések az alap ténytáblázatok helyett korábban kiszámított összesítések alapján is kiszámíthatók.

Hogyan alkalmazhatók ezek az optimalizációs technikák MOLAP esetében? A ROLAP érték alapú címzést használ, ahol a dimenzióértékeket kulcs alapú címkeresési stratégiák segítségével érik el. Ezzel szemben a MOLAP közvetlen tömbcímzést használ, ahol a dimenzióértékeket a nekik megfelelő tömbblokk pozícióján vagy indexén keresztül érik el. Ezért a MOLAP nem tudja végrehajtani az első ROLAP optimalizációs technika érték

alapú újrarendezését. Vagyis más módszert kell alkalmazni a MOLAP tömb alapú kockaszerkezetéhez, például a következőt:

(1) A tömböt nagyobb darabokra vágjuk szét. **Darabnak** nevezünk egy olyan alkockát, amely elég kicsi ahhoz, hogy a kockaszámításokhoz elérhető memóriába beférjen. **Darabolásnak** nevezzük az *n*-dimenziós tömb felosztását kis *n*-dimenziós darabokra, ahol minden darabot objektumként tárolunk a lemezen. A darabokat tömörítjük, hogy az üres tömbcellákból (érvényes adatot nem tartalmazó cella) adódó tárpazarlást elkerüljük. Például *„darabID + offset"* használható cellacímző mechanizmusként a **ritkatömb-szerkezet tömörítéséhez**, és egy darabon belüli cella kereséséhez. Egy ilyen tömörítő technika elég hatékony a ritka kockák kezeléséhez, lemezen és memóriában egyaránt.
(2) Összesítések számítása kockacellák látogatásával (a cellaértékek elérésével). A cellalátogatások sorrendje úgy optimalizálható, hogy a *szükséges újralátogatások száma* minimális maradjon, és így a memória-hozzáférések száma és a tárolási költségek is csökkenjenek. Ezt a rendezést trükkösen ki lehet használni úgy, hogy a részleges összesítéseket párhuzamosan lehessen számítani, és a cellák szükségtelen újralátogatását is el lehessen kerülni.

Mivel ez a daraboló technika azzal jár, hogy néhány összesítés számítása átfedi egymást, ezért ezt **többutas tömbösszesítésnek** nevezzük az adatkocka-számítás területén.

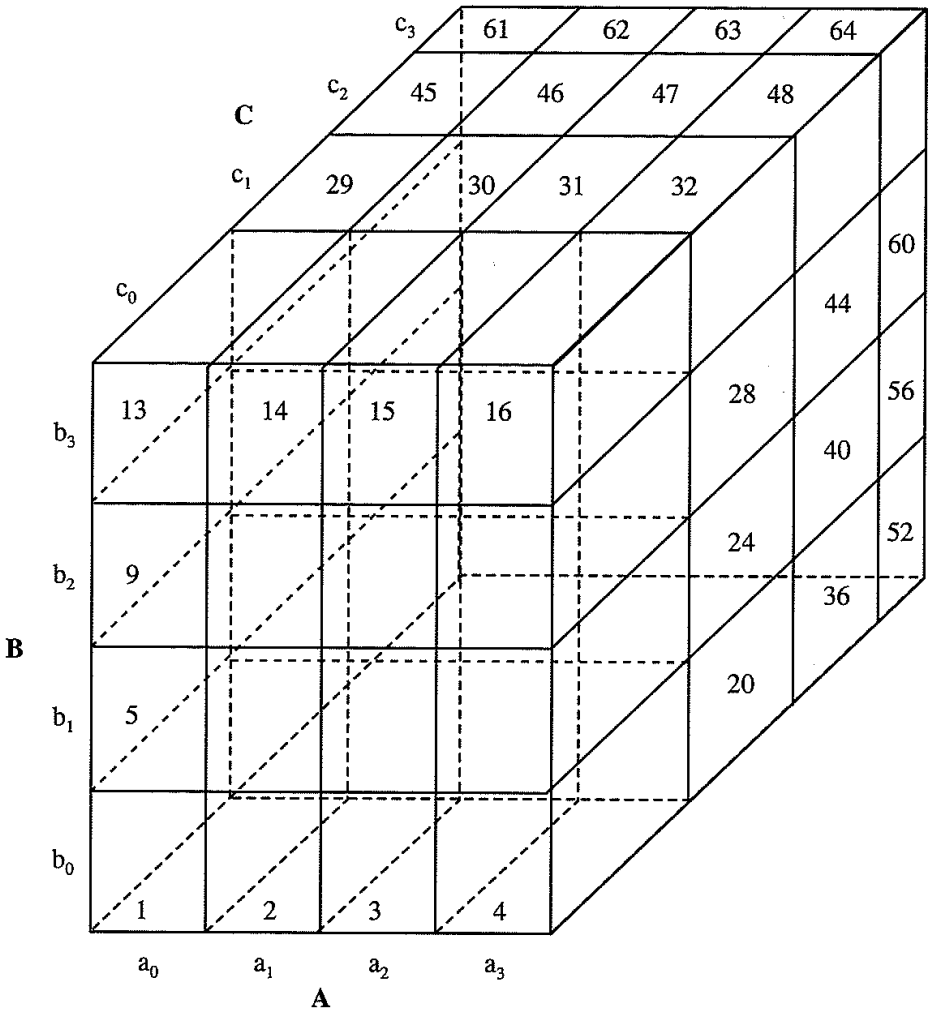
A MOLAP-kockaszerkesztés e megközelítését egy példán keresztül magyarázzuk el.

2.12. példa | Tekintsünk egy háromdimenziós adattömböt az *A*, *B*, *C* dimenziókkal. A háromdimenziós tömböt kis, memória alapú darabokra bontjuk fel. Ebben a példában a tömböt 64 darabra osztottuk fel, ahogy ez a 2.15. ábrán látható. Az *A* dimenziót négy egyenlő méretű részre vágtuk: a_0, a_1, a_2 és a_3. A *B* és *C* dimenziók is hasonlóan négyfelé vannak osztva. Az 1-es, 2-es, ..., 64-es darabok rendre az $a_0b_0c_0$, $a_1b_0c_0$, ..., $a_3b_3c_3$ alkockáknak felelnek meg. Tegyük fel, hogy az *A*, *B* és *C* dimenziókhoz a tömbméret rendre 40, 400 és 4000. Az *A*, *B*, és *C* partícióinak mérete tehát rendre 10, 100 és 1000.

A megfelelő adatkocka teljes megvalósítása a kockát definiáló összes részkocka kiszámítását foglalja magába. Ezek a részkockák a következőkből állnak:

- Az alapkocka, amelyet *ABC*-vel jelölünk (amelyből az összes többi részkockát számítjuk közvetve vagy közvetlenül). Ez a kocka már ki van számolva és megfelel az adott háromdimenziós tömbnek.
- A kétdimenziós részkockák: *AB*, *AC* és *BC*, amelyek rendre megfelelnek az *AB*, *AC* és *BC* csoportosításoknak. Ezeket a részkockákat ki kell számítani.
- Az egydimenziós részkockák: *A*, *B* és *C*, amelyek rendre megfelelnek az *A*, *B* és *C* csoportosításoknak. Ezeket a részkockákat is ki kell számítani.
- A nulladimenziós részkocka, **minden**-nel jelölve, amely a () csoportosításnak felel meg; azaz itt nincs csoportosítás. Ezt a részkockát is ki kell számítani.

Nézzük, hogyan lehet a többutas tömbösszesítő technikát alkalmazni ebben a számításban. A kockaszámításhoz használatos darabok sokféle sorrendben olvashatók be a memóriába.



2.15. ábra | Egy háromdimenziós tömb *A*, *B* és *C* dimenziókkal, 64 *darabbal*

Most vegyük a 2.15. ábrán látható sorrendet 1-től 64-ig. Tegyük fel, hogy a *BC* részkocka b_0c_0 darabját akarjuk kiszámítani. Helyet foglalunk ennek a darabnak a *darabmemóriában*. Az *ABC* 1-es, 2-es, 3-as és 4-es darabjainak a vizsgálatával kiszámítjuk a b_0c_0 darabot, azaz b_0c_0 celláit összesítjük $a_0, \ldots, a_3$ szerint. A darabmemóriában ezután a következő darabnak, b_1c_0-nak foglalunk helyet. Az összesítésekhez az *ABC* következő négy darabját, az 5-östől a 8-asig kell megvizsgálni. Így folytatva az egész *BC* részkocka kiszámítható. Az egész *BC* darab kiszámításához egyszerre mindig csak a *BC* egy *darabjának* kell a memóriában lennie.

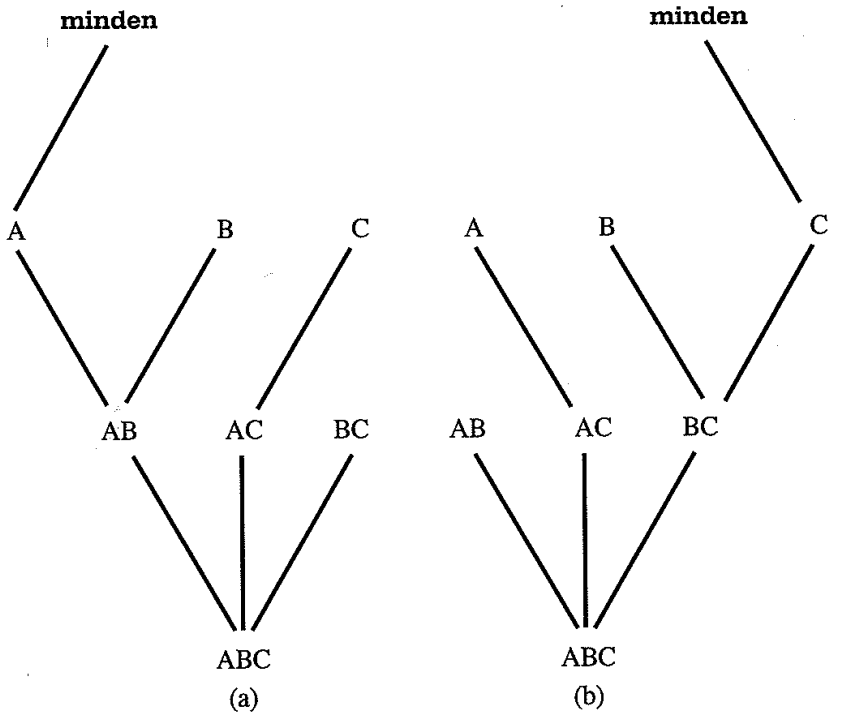
A *BC* részkocka számítása során mind a 64 darabot megvizsgáljuk. *Van arra lehetőség, hogy más részkockák számításához, például AC-hez vagy AB-hez ne kelljen újra megvizsgálnunk ezeket a darabokat?* A válasz határozott *igen*. Ez az a pont, ahol a többutas számítási elképzelés a segítségünkre siet. Amikor például az 1-es darabot ($a_0b_0c_0$) vizsgáljuk (mondjuk a *BC* kétdimenziós b_0c_0 darabjának a fenti számításához), akkor ezzel együtt az $a_0b_0c_0$-hoz tartozó összes többi kétdimenziós darab is kiszámítható. Tehát amikor $a_0b_0c_0$-t vizsgáljuk, akkor mind a három darabot, b_0c_0-t, a_0c_0-t és a_0b_0-t is ki kell számítani a három kétdimenziós összesítő síkon: *BC*-n, *AC*-n és *AB*-n. Más szavakkal a többutas számítás minden egyes kétdimenziós síkra is összesít, amíg egy háromdimenziós darab van a memóriában.

Most nézzük meg, hogy ha a darabokat vagy a részkockákat másmilyen sorrendben vizs-

gáljuk, illetve számítjuk, az milyen hatással van az általános adatkocka-számítás hatékonyságára. Emlékeztetőül az *A*, *B* és *C* dimenziók mérete rendre 40, 400 és 4000. Így a legnagyobb kétdimenziós sík *BC* (mérete 400 × 4000 = 1 600 000). A második legnagyobb kétdimenziós sík *AC* (mérete 40 × 4000 = 160 000). A legkisebb kétdimenziós sík *AB* (mérete 40 × 400 = 16 000).

Tegyük fel, hogy a darabokat 1-től 64-ig a megadott sorrendben vizsgáljuk. Így a legnagyobb kétdimenziós sík, *BC* egy darabját minden vizsgált sornál *teljes egészében* kiszámítjuk. Vagyis az 1–4-es darabokat tartalmazó sor vizsgálata után b_0c_0-t teljes egészében összesítjük; az 5–8-as darabokat tartalmazó sor vizsgálata után b_1c_0-t teljes egészében összesítjük, és így tovább. Összehasonlításképpen, a második legnagyobb kétdimenziós sík, *AC* egy darabjának a teljes számításához 13 darab vizsgálatára van szükség (az adott sorrenddel 1-től 64-ig). Például az 1-es, 5-ös, 9-es és 13-as darabok vizsgálata után a_0c_0 összesítése teljes. Végül a legkisebb kétdimenziós sík, *AB* egy darabjának a teljes számításához 49 darab vizsgálatára van szükség. Például az 1-es, 17-es, 33-as és 49-es darabok vizsgálata után a_0b_0 összesítése teljes mértékben megtörtént. Vagyis *AB* számításához van szükség a darabok leghosszabb vizsgálatára. Ahhoz, hogy elkerülhető legyen egy háromdimenziós darab többszöri memóriába mozgatása, a minimális memóriaigény, hogy az összes releváns kétdimenziós síkot a darabmemóriában tartsuk (az 1-től 64-ig sorrend szerint) a következő: 40 × 400 (az egész *AB* síkhoz) + 40 × 1000 (az *AC* sík egy sorához) + 100 × 1000 (a *BC* sík egy darabjához) = 16 000 + 40 000 + 100 000 = 156 000.

Most tegyük fel, hogy a darabokat 1, 17, 33, 49, 5, 21, 37, 53, ... sorrendben vizsgáljuk



2.16. ábra | A 2.12. példa háromdimenziós kockájának a kiszámításához használható többutas tömbösszesítés két sorrendje: (a) a tömbösszesítés leghatékonyabb sorrendje (minimális memóriaigény 156 000 memóriaegység); (b) a tömbösszesítés legkevésbé hatékony sorrendje (minimális memóriaigény 1 641 000 memóriaegység)

meg. Azaz először az *AB* sík felé, majd az *AC* felé, végül a *BC* sík felé összesítünk. Hogy a kétdimenziós síkokat a darabmemóriában tartsuk, a minimális memóriaigény a következő lenne: 400 × 4000 (az egész *BC* síkhoz) + 40 × 1000 (az *AC* sík egy sorához) + 10 × 100 (az *AB* sík egy darabjához) = 1 600 000 + 40 000 + 1000 = 1 641 000. Vegyük észre, hogy ez *több mint tízszerese* az 1, ..., 64 sorrend igényének.

Hasonlóan megadható a minimális memóriaigénye az egy- és nulladimenziós részkockák többbutas kiszámítási módszerének is. A 2.16. ábrán látható a leghatékonyabb és a legkevésbé hatékony sorrend, az adatkocka számításához szükséges minimális memóriaigény alapján. A leghatékonyabb a darabok 1, ..., 64 sorrendje. ■

A 2.12. példában feltettük, hogy van elég memória az *egymenetes* adatkocka-számításhoz (amikor az összes részkocka a darabok egyszeri vizsgálata alapján kiszámítható). Ha nincs elég hely a memóriában, akkor a háromdimenziós tömbön többször végig kell menni. Ebben az esetben azonban a sorszámozott darabszámítás alapelve ugyanaz marad.

Melyik gyorsabb, a ROLAP- vagy a MOLAP-kockaszámítás? Kísérletekkel igazolták, hogy megfelelő ritkatömb-tömörítő technikák használatával és a részkockaszámítások sorrendjének körültekintő megválasztásával a MOLAP-kockaszámítás lényegesen gyorsabb, mint a ROLAP- (relációs rekord alapú) számítás. A MOLAP tömbszerkezete, a ROLAP-tól eltérően, nem igényel tármegtakarítást a keresőkulcsok tárolásához. A MOLAP továbbá közvetlen tömbcímzést használ, ami gyorsabb a ROLAP kulcs alapú címkeresési stratégiájánál. A ROLAP-kockaszámításnál a táblázat közvetlen kockázása helyett gyorsabb lehet a táblázatot tömbbé alakítani, a tömböt kockázni, majd az eredményt visszaalakítani táblázattá. Ez a megfigyelés azonban csak a viszonylag kevés dimenzióval rendelkező kockák esetén működik, mivel a kiszámítandó részkockák száma a dimenziók számának exponenciális függvénye. A dimenziók átkának elkerülésére a legfrissebb kutatások azt javasolják, hogy csak a **jéghegykockákat** számítsuk ki. Ezek olyan kockák, amelyek csak olyan kockarészeket tárolnak, ahol a partíció minden cellájához az összesített érték (például darabszám) valamilyen minimális szint vagy előfordulási küszöbérték felett van.

Mi történik akkor, ha új adatot akarunk felvenni az előszámított kockába, vagy néhány régebbi adattól meg akarunk szabadulni? Hatékony **növekményes frissítő** módszereket dolgoztak ki erre a célra, amelyekkel lehetőség van adatok felvételére, illetve törlésére az előszámított kockából anélkül, hogy a kockát elölről újra kéne számítani. A konkrét módszereket az ilyen frissítésre gyakorlat gyanánt az olvasóra hagyjuk.

2.4.2. | OLAP-adatok indexelése

A hatékony adatelérés lehetővé tétele érdekében a legtöbb adattárházrendszer támogatja az indexszerkezeteket és a megvalósított nézettáblákat (részkockák használatával). A megvalósítandó részkockák kiválasztásának módszereivel az előző részben foglalkoztunk. Ebben a részben azt vizsgáljuk meg, hogyan lehet az OLAP-adatokat a *bittérképindex* és az *összekapcsoló index* segítségével indexelni.

A **bittérképindexelés** módszere azért népszerű az OLAP-termékeknél, mert gyors keresést biztosít az adatkockában. A bittérképindex a *rekord_ID (RID)* lista alternatív ábrázolása. A bittérképindexben egy adott attribútum minden lehetséges *v* értékéhez van egy külön bitvektor: *Bv*. Ha az attribútum értékkészlete *n* értékből áll, akkor minden

bejegyzéshez a bittérképindexben n bitre van szükség (azaz n bitvektorunk lesz). Ha az adattáblázat adott sorában az attribútum értéke v, akkor a bittérképindex megfelelő sorában a v értékét jelölő bit értéke 1, az összes többié pedig 0.

2.13. példa | Az *ElektroMind* adattárházban tegyük fel, hogy az *árucikk* dimenzió a legfelső szinten négy értéket vehet fel (árucikktípusokat jelölve): *szórakoztató, számítógép, telefon* és *biztonság*. Minden értéket (például *számítógép*) egy bitvektorral ábrázolunk az *árucikk*-hez tartozó bittérképindexben. Tegyük fel, hogy a kockát egy 100 000 soros relációs táblázatban tároljuk. Mivel az *árucikk* értékkészlete négy elemű, a bittérképindex-táblázatban négy bitvektorra (vagy listára) van szükség, mindegyik 100 000 bittel. A 2.17. ábrán látható az alap (adat)táblázat az *árucikk* és *város* dimenziókkal, és az egyes dimenziókhoz tartozó bittérképindex-táblázatokkal. ■

Alaptáblázat

RID	árucikk	város
R1	SZ	V
R2	SZG	V
R3	T	V
R4	B	V
R5	SZ	NY
R6	SZG	NY
R7	T	NY
R8	B	NY

Árucikk bittérképindex-táblázat

RID	SZ	SZG	T	B
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

Város bittérképindex-táblázat

RID	V	NY
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Jelölések: Sz: szórakoztató, SzG: számítógép, T: telefon, B: biztonság, V: Vancouver, NY: New York

2.17. ábra | OLAP-adatok bittérképes indexelése

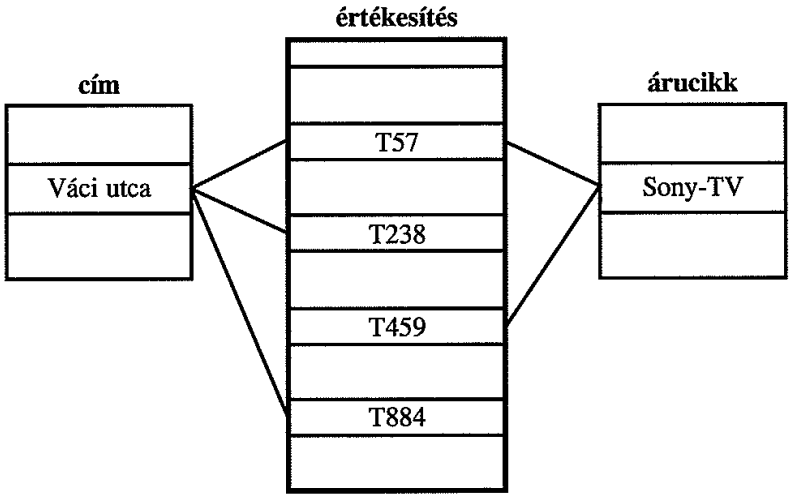
A bittérképindexelés előnyösebb a tördelésnél és a faindexeknél. Különösen hasznos alacsony számosságú értékkészlet esetén, mert az összehasonlítás, összekapcsolás és összesítés műveletek bitaritmetikára vezethetők vissza, amely alapvetően csökkenti a feldolgozási időt. A bittérképindexelés lényegesen csökkenti a tár- és I/O-költségeket is, hiszen egy karakterlánc most egyetlen bittel van ábrázolva. Nagyobb számosságú értékkészletekhez is alkalmazható a módszer, tömörítőtechnikák felhasználásával. Az **összekapcsoló index** onnan nyerte a népszerűségét, hogy a relációs adatbázis lekérdezésében használták. A hagyományos indexelés az egy adott oszlopban található értéket azon sorok listájára képezi le, amelyekben ugyanaz az érték fordul elő. Ezzel szemben az összekapcsoló index a relációs adatbázis két relációjának az összekapcsolható sorait jegyzi fel. Például ha két reláció, *R(RID, A)* és *S(B, SID)* az *A* és *B* attribútumok mentén kapcsolódnak, akkor az összekapcsoló index rekordja az *(RID, SID)* párt tartalmazza, ahol *RID*, illetve *SID* az *R* és *S* relációk sorazonosítói. Így az összekapcsoló index rekordja azonosítani tudja az összekapcsolható sorokat anélkül, hogy az összekapcsolás drága műveletét elvégezné. Az összekapcsoló indexelés különösen hasznos az idegen kulcs[5] és az ezzel azonos értékű elsődleges kulcs közötti kapcsolat fenntartásához, összekapcsolható relációk esetén.

5 | Egy attribútumhalmazt egy relációsémában, amely elsődleges kulcsként szolgál egy másik relációsémában, **idegen kulcsnak** nevezünk.

Az adattárházak csillagséma modellje vonzóvá teszi az összekapcsoló indexet a táblaközi kereséshez, mert a ténytáblázat és a dimenziótáblázatok közti kapcsolatot a ténytáblázat idegen kulcsa és a dimenziótáblázatok elsődleges kulcsa jelenti. Az összekapcsoló indexelés fenntartja a kapcsolatot a dimenzió attribútumértékei és a ténytáblázat megfelelő sorai között. Az összekapcsoló indexek több dimenziót is felölelhetnek. Ilyenkor **összetett összekapcsoló indexről** beszélünk. Az összekapcsoló indexek érdekes alkockák azonosítására is használhatók.

2.14. példa | A 2.4. példában definiáltunk egy csillagsémát az *ElektroMind* áruházhoz a következő formában: *értékesítés_csillag*[*idő, árucikk, üzlet, cím*]:*dollár_eladott* = **sum**(*eladások_dollárban*). A 2.18. ábrán láthatunk példát egy összekapcsoló index típusú kapcsolatra az *értékesítés* ténytáblázat és a *cím* és *árucikk* dimenziótáblázatok között. A *cím* dimenziótáblázat *Váci utca* értéke például az *értékesítés* ténytáblázat T57, T238 és T884-es soraihoz kapcsolódik. Hasonlóan, az *árucikk* dimenziótáblázat *Sony-TV* értéke például az *értékesítés* ténytáblázat T57 és T459-es soraihoz kapcsolódik. A megfelelő összekapcsoló-index-táblázatok a 2.19. ábrán láthatók.

Tegyük fel, hogy 360 féle időérték, 100 árucikk, 50 üzlet, 30 cím és 10 millió értékesítési sor található az *értékesítés_csillag* adatkockában. Ha az *értékesítés* ténytáblázat csak 30 féle árucikk eladásáról tárol feljegyzést, akkor a maradék 70 árucikk nyilvánvalóan nem vesz részt az összekapcsolásban. Ha nem használunk összekapcsoló indexeket, akkor további I/O-műveleteket kell végrehajtani ahhoz, hogy a ténytáblázat és a dimenziótáblázatok összekapcsolódó részeit összehozzuk. ■



2.18. ábra | Kapcsolat az *értékesítés* ténytáblázat és a *cím* és *árucikk* dimenziótáblázatok között

A lekérdezések feldolgozásának további gyorsításához az összekapcsoló és a bittérképindexelési módszereket egyesítve **bittérképezett összekapcsoló indexeket** hozhatunk létre. A Microsoft SQL Server és a Sybase IQ a bittérképindexeket támogatja. Az Oracle 8 bittérkép- és összekapcsoló indexeket is használ.

Összekapcsolóindex-táblázat
cím/értékesítés

cím	értékesítés_kulcs
...	...
Váci utca	T57
Váci utca	T238
Váci utca	T884
...	...

Összekapcsolóindex-táblázat
árucikk/értékesítés

árucikk	értékesítés_kulcs
...	...
Sony-TV	T57
Sony-TV	T459
...	...

Összekapcsolóindex-táblázat két dimenzió kapcsolásához
cím/árucikk/értékesítés

cím	árucikk	értékesítés_kulcs
...	...	...
Váci utca	Sony-TV	T57
...	...	...

2.19. ábra | Összekapcsoló indextáblázatok a 2.18. ábrán látható kapcsolatokhoz

2.4.3. | OLAP-lekérdezések hatékony feldolgozása

A részkockák megvalósításával és OLAP-indexstruktúrák létesítésével az a célunk, hogy felgyorsítsuk a lekérdezések feldolgozását az adatkockákban. Adott megvalósított nézettáblákkal a lekérdezésnek a következőképpen kell folynia:

(1) **Határozzuk meg, hogy milyen műveleteket kell végrehajtani a rendelkezésre álló részkockákon:** ebbe beleértendő a lekérdezésben megadott bármilyen kiválasztás, vetítés, felgörgetés (csoportosítás) és lefúrás művelet megfelelő átültetése SQL-be és/vagy OLAP-műveletekbe. Az adatkocka szeletelése és kockázása például egy megvalósított részkockán való kiválasztásnak és/vagy vetítésnek felelhet meg.

(2) **Határozzuk meg, hogy a releváns műveleteket melyik megvalósított részkocká(ko)n kell végrehajtani:** ide tartozik az összes olyan megvalósított részkocka azonosítása, amelyről feltehető, hogy használható a lekérdezés megválaszolásához, ennek a halmaznak a további szűkítése a részkockák közti „dominancia"-kapcsolatok ismeretének a segítségével, az így megmaradt megvalósított részkockák használatával járó költségek becslése, végül a legolcsóbb részkocka kiválasztása.

2.15. példa | Tegyük fel, hogy az *ElektroMind* adataihoz a következő formában adunk meg egy adatkockát: *értékesítés*[*idő, árucikk, cím*]:**sum**(*eladások_dollárban*). Az *idő*höz használt dimenzióhierarchia: *nap* < *hónap* < *negyedév* < *év*, az *árucikk*hez: *árucikk_név* < *márka* < *típus*, és a *cím*hez: *utca* < *város* < *megye_vagy_tartomány* < *ország*.

Tegyük fel, hogy a feldolgozandó lekérdezés a {*márka, megye_vagy_tartomány*} felbontást kérdezi le az *év* = *2000* kiválasztási feltétellel. Tegyük fel továbbá, hogy a következő négy megvalósított részkocka áll rendelkezésre:

- 1. részkocka: {*árucikknév, város, év*}
- 2. részkocka: {*márka, ország, év*}
- 3. részkocka: {*márka, megye_vagy_tartomány, év*}
- 4. részkocka: {*árucikknév, megye_vagy_tartomány*}, ahol *év = 2000*

A fenti négy részkocka közül melyiket kellene választani a lekérdezéshez? Durvább felosztású adatokból nem kaphatunk finomabb felosztású adatokat. Így a 2. részkockát nem választhatjuk, mivel az *ország* általánosabb fogalom a *megye_vagy_tartomány*nál. A többi részkocka használható a lekérdezéshez, mert (1) a dimenziók halmaza ugyanaz vagy bővebb, mint a lekérdezésben; (2) a lekérdezésben szereplő kiválasztás értelemszerűen tartalmazhatja a részkockabeli kiválasztást; és (3) az *árucikk* és *cím* dimenziókhoz tartozó absztrakciós szintek ezekben a részkockákban rendre finomabb szintűek, mint a *márka* és *megye_vagy_tartomány*.

Mekkora költséggel járna az egyes részkockák használata? Valószínű, hogy az 1. részkocka lenne a legdrágább, mivel a *árucikknév* és a *város* is alacsonyabb szinten van, mint a lekérdezésben megadott *márka* és *megye_vagy_tartomány* fogalmak. Ha a kockában az *árucikk*hez nem tartozik sok *év* érték, de minden *márká*hoz több *árucikknév* is tartozik, akkor a 3. részkocka kisebb a 4-nél, tehát a lekérdezéshez a 3. részkockát választanánk. Ha azonban a 4. részkockához hatékony indexek is vannak, akkor ez lenne a jobb választás. Tehát szükségünk van valamilyen költség alapú becslésre, hogy eldöntsük, a részkockák melyik halmazát válasszuk a lekérdezéshez. ■

Mivel egy MOLAP-szerver tárolási modellje az *n*-dimenziós tömb, ezért a rendszer felé érkező többdimenziós lekérdezéseket közvetlenül a szerver tárolási struktúrájára képezik le, amely közvetlen címzési lehetőséget biztosít. Az adatkocka kézenfekvő tömbábrázolása jó indexelési tulajdonságokkal rendelkezik, viszont ritka adat esetén rossz a tárfelhasználása. A hatékony tárkihasználás és feldolgozás érdekében tehát ritkamátrix- és adattömörítési technikákat kell alkalmazni.

A sűrű és ritka tömbök által használt tárolási szerkezetek eltérőek lehetnek, emiatt előnyös lehet kétszintű megközelítést alkalmazni a MOLAP-lekérdezések feldolgozásánál: sűrű tömbök esetén használjunk tömbszerkezetet, ritka tömbök esetén pedig ritkamátrix-szerkezetet. A kétdimenziós sűrű tömböket B-fákkal lehet indexelni.

A MOLAP-lekérdezés feldolgozásához először is az egy- és kétdimenziós sűrű tömböket kell azonosítani. Ezekhez a tömbökhöz aztán a hagyományos indexstruktúrákkal lehet indexeket létrehozni. A kétszintű megközelítés anélkül növeli a tár kihasználtságát, hogy a közvetlen címzés lehetőségét feláldozná.

2.4.4. | Metaadatraktár

Mik a metaadatok? A **metaadat** az adatokról szóló adat. Amikor adattárházban használjuk, akkor a metaadat az az adat, amely a tárházobjektumokat definiálja. Metaadatot hozunk létre az adott tárház adatneveire és adatdefinícióira. További metaadatok jönnek létre és kerülnek felhasználásra az adatforrások és a belőlük kinyert adatok időpecsételésénél, és a hiányzó mezők időpecsételésénél, amelyek adattisztítás vagy adategyesítés során kerültek az adatbázisba.

Egy metaadatraktárnak a következőket kell tartalmaznia:

- Az *adattárház szerkezetének* a leírása, amely tartalmazza a tárházsémát, a nézetet, a dimenziókat, a hierarchiákat, a származtatott adatdefiníciókat és az adatpiacok helyét és tartalmát.
- *Operatív metaadatok,* amelyek tartalmazzák az adatok származását (a valahonnan átvett adatok történetét és a rajtuk végrehajtott transzformációk sorozatát), az adatok időszerűségét (aktív, archivált, törölt) és a felügyelettel kapcsolatos információkat (tárházhasználati statisztikák, hibaüzenetek, eseménynapló).
- *Az összegzéshez használt algoritmusok,* amelyek tartalmazzák a mérték- és dimenzió-meghatározó algoritmusokat, a felbontás finomságával, a partíciókkal, a különböző témákkal, az összesítésekkel és összegzésekkel kapcsolatos adatokat, az előre definiált lekérdezéseket és jelentéseket.
- *A leképezés az operatív környezetből az adattárházba,* amely magába foglalja a forrás-adatbázisokat és tartalmukat, az átjárók leírását, az adatpartíciókat, adatkinyerő, -tisztító és -transzformáló szabályokat és alapértelmezéseket, adatfrissítő és törlő szabályokat és a biztonsági kérdéseket (felhasználóazonosítás és hozzáférés-vezérlés).
- *A rendszer teljesítményéhez kapcsolódó adatok,* amelyek tartalmazzák azon indexeket és profilokat, amelyek az adatelérési és adatvisszanyerési teljesítményt növelik, ezenfelül a frissítési, módosítási és többszörözési ciklus időzítésére és ütemezésére vonatkozó szabályokat.
- *Üzleti metaadat,* amely magába foglalja az üzleti szóhasználatot és definíciókat, az adatok tulajdonjogára vonatkozó információkat és a díjazási irányelveket.

Egy adattárház különböző szintű összegzéseket tartalmaz, és ezeknek csak az egyik típusa a metaadat. Ilyenek lehetnek még az aktuális részletes adatok (amelyek majdnem mindig lemezen vannak), régebbi részletes adatok (amelyek általában harmadlagos tárolón vannak), alacsony szinten, illetve magas szinten összesített adatok (amelyek fizikailag vagy tárolva vannak valahol, vagy nem).

A metaadatok teljesen más szerepet töltenek be, mint az adattárház más adatai, és sok szempontból fontosak. Használhatók például címtárként, hogy a döntéstámogató rendszer elemzője megállapítsa az adattárház tartalmának a helyét, útmutatóként az adatok transzformálásánál, amikor az operatív környezetből átkerülnek az adattárház környezetbe, ismertetőként az aktuális részletes adatok alacsony, majd magasabb szintű összegzésénél használt algoritmusokhoz. A metaadatokat maradandó formában (azaz lemezen) kell tárolni és kezelni.

2.4.5. | Az adattárházak háttéreszközei és segédprogramjai

Az adattárházrendszerek háttéreszközöket és segédprogramokat használnak az adataik feltöltésére és frissítésére. Ezek az eszközök és lehetőségek a következő funkciókat foglalják magukba:

- **Adatkinyerés**, amely jellemzően többszörös, heterogén és külső forrásokból gyűjti össze az adatokat.
- **Adattisztítás**, amely a hibákat észleli az adatokban és ki is javítja azokat, ha lehetséges.
- **Adattranszformáció**, amely a tárháznak megfelelő formátumúra alakítja az adatokat.
- **Feltöltés**, amely rendez, összegez, egyesít, nézettáblákat számít, integritást ellenőriz, indexeket és partíciókat hoz létre.
- **Frissítés**, amely az adatforrások módosításait viszi át a tárházba.

A tisztítási, feltöltési, frissítési és metaadat-definíciós eszközök mellett az adattárház-rendszerek általában szép számmal biztosítanak tárházkezelő eszközöket is.

Az adattisztítás és az adattranszformáció fontos lépések az adat minőségének a fejlesztésében, ennélfogva az adatbányászati eredmények javításában is szerepet játszanak. Ezekkel a 3. fejezetben foglalkozunk. Mivel most az adattárház-technológiák főleg az adatbányászattal kapcsolatban érdekesek számunkra, ezért a további eszközök részleteiben nem mélyedünk el. Az érdeklődő olvasó számára az adattárház-technológiáknak szentelt könyveket javasoljuk.

2.5. | Az adatkocka-technológia továbbfejlesztése

Ebben a részben az adatkocka-technológia további fejlődését tanulmányozzuk. A 2.5.1. alfejezet az *adatkocka felfedezésvezérelt feltárásával* történő adatbányászatot írja le, ahol is az adatokban található szabálytalanságok automatikusan kimutathatók és a felhasználó számára láthatóan megjelölhetők. A 2.5.2. alfejezet a *többtulajdonságú kockákkal* foglalkozik. Ezek olyan összetett adatbányász-lekérdezések esetén is használhatók, amelyek több finomsági szinten több függő összesítést is tartalmaznak. További fejlesztésekről van szó a 2.5.3. alfejezetben.

2.5.1. | Az adatkockák felfedezésvezérelt feltárása

Ahogy ezt ebben a fejezetben láttuk, az adatokat egy OLAP-rendszer többdimenziós adatkockájában összegezhetjük és tárolhatjuk. A felhasználó vagy elemző számos OLAP-művelet, például lefúrás, felgörgetés, szeletelés és kockázás segítségével érdekes mintákat kereshet a kockában. Ezek az eszközök segítik a felhasználót az adatok vizsgálatában, de maga a feltáró folyamat nem automatizált. Itt a felhasználó az, aki a megérzéseire és feltevéseire hagyatkozva megpróbál kivételes eseteket vagy szabálytalanságokat felismerni az adatokban. A **feltevésvezérelt feltárás** számos hátránnyal jár. A keresési tér nagyon nagy lehet, és ezzel az adatok kézi vizsgálata nyomasztó és kilátástalan feladattá válhat. A magas szintű összesítések nem mindig jelzik az alacsonyabb szinten előforduló szabálytalanságokat, így érdekes minták könnyen észrevétlenek maradhatnak. Még ha csak a kockának egy részhalmazát (például egy szeletet) vizsgálja a felhasználó, akkor is nagy mennyiségű adattal találja magát szemben. Pusztán az adatértékek hatalmas mennyisége miatt nagyon könnyű a kivételes esetek felett elsiklani, ha feltevésvezérelt feltárást választunk.

A **felfedezésvezérelt feltárás** egy másik megközelítési mód, amelyben a kivételes eseteket jelző, előre kiszámított mértékek segítségével vezetik a felhasználót az adatelemző folyamatban, az összesítések minden szintjén. Ezekre a mértékekre ezentúl mint *kivételmutatókra* fogunk hivatkozni. Értelemszerűen **kivételnek** számít az adatkocka olyan cellaértéke, amely lényegesen eltér a valamilyen statisztikai modell alapján várt értéktől. A modell a mérték értékváltozásait és a benne megfigyelhető mintákat a cellához tartozó *dimenziók mindegyike* mentén figyelembe veszi. Például, ha az árucikk-értékesítés adatainak az elemzéséből az derül ki, hogy a többi hónappal összehasonlítva decemberben nagyobb az eladások száma, akkor ez az idődimenzió mentén kivételes esetnek tűnhet. Az árucikk dimenzió felől nézve azonban ez nem számít kivételnek, hiszen az értékesítések hasonló növekedése figyelhető meg decemberben az összes árucikk tekintetében. A modell olyan kivételeket vesz figyelembe, amelyek az adatkocka minden összesített csoportosításában megfigyelhetők. Vizuális jelek, például háttérszínek tükrözik minden cella kivételességének a fokát az előre kiszámított kivételmutatók alapján. A kockaépítéshez használható hatékony algoritmusokkal foglalkoztunk a 2.4.1. alfejezetben. A kivételmutatók számítása átfedésbe kerülhet a kockaépítéssel, és így az adatkockáknak a felfedezésvezérelt feltáráshoz való általános felépítése hatékonnyá válik.

Kivételmutatóként három mértéket használnak az adatokban előforduló szabálytalanságok azonosítására. Ezek a mértékek a cellaérték által okozott meglepetés fokát mutatják a várható értékhez képest. Értéküket minden cellára kiszámítják, az összesítés minden fokán. A három mérték a következő:

- **SelfExp** – A cellaérték által okozott meglepetés foka a többi azonos összesítési szinten lévő cellához képest.
- **InExp** – A meglepetés foka valahol a cella alatt, ha lefúrást végeznénk belőle.
- **PathExp** – A meglepetés foka a cellából indított összes lefúrási útvonalhoz.

A következő példán keresztül bemutatjuk, hogy hogyan használjuk ezeket a mértékeket az adatkockák felfedezésvezérelt feltárásához.

2.16. példa | Tegyük fel, hogy az *ElektroMind* havi eladásait akarjuk elemezni az előző hónaptól való százalékos eltérés formájában. Az érintett dimenziók: *árucikk, idő* és *terület*. Az összes árucikkre és értékesítési területre összesített havi eladások vizsgálatával kezdjük, ahogy ezt a 2.20. ábra mutatja.

A kivételmutatók megjelenítéséhez a **highlight exceptions** gombra kellene kattintanunk a képernyőn. Ennek eredményeképpen a SelfExp és InExp értéke minden cellában vizuális jelekké alakul. A cellák háttérszíne a SelfExp értékétől függ. Emellett az InExp érték

Eladások összértéke	Hónap											
	jan.	febr.	márc.	ápr.	máj.	jún.	júl.	aug.	szept.	okt.	nov.	dec.
Összesen		1%	-1%	0%	1%	3%	-1%	-9%	-1%	2%	-4%	3%

2.20. ábra | Az értékesítésben megfigyelhető változások az idő függvényében

függvényében a cellák különböző vastagságú és színű vonalakkal is be vannak keretezve. Minél vastagabb a keretvonal, annál magasabb az InExp érték, és mindkét esetben minél sötétebb a szín, annál kivételesebb celláról van szó. A sötét és vastag keretek például júliusban, augusztusban és szeptemberben arra hívják fel a felhasználó figyelmét, hogy ezeknek a celláknak az alacsonyabb szintű összesítését is vizsgálja meg lefúrással.

A lefúrást az összesített *árucikk* vagy *terület* dimenzió mentén is el lehet végezni. *Melyik útvonalon található több kivétel?* Hogy ezt megtudjuk, kiválasztunk egy érdekesnek tűnő cellát, és elindítunk egy **path exception** modult, amely a cella PathExp értéke alapján minden dimenziót kiszínez valahogy. Ez az érték az adott útvonal meglepetésszintjét tükrözi. Tegyük fel, hogy az *árucikk* mentén van több kivétel.

Az árucikk dimenzió mentén történő lefúrás eredménye a 2.21. ábrán látható kockaszelet, ahol az árucikkek szerinti eladásokat havonkénti bontásban kapjuk meg. Ezen a ponton az elemzéshez sok különböző érték áll a rendelkezésünkre. A **highlight exceptions** gombra kattintva megjelenő vizuális jelzések a figyelmünket a kivételek felé terelik. Vegyük a *Sony ff nyomtató* szeptemberi 41%-os értékesítési különbségét. A cella háttérszíne sötét, amely magas SelfExp értéket jelöl, azaz ez egy kivételes cella. Nézzük most a novemberi –15%-os és a decemberi –11%-os különbséget. A decemberi érték kivételként van megjelölve, a novemberi viszont nincs megjelölve, pedig a –15% nagyobb eltérést jelent, mint a –11%. Ez azért van, mert a kivételmutatók az összes dimenziót számításba veszik, amely a cellához tartozik. Figyeljük meg, hogy a legtöbb árucikk decemberi eladása nagy pozitív értékkel rendelkezik, a novemberi értékesítések pedig nem. Tehát a cella helyzetét a kockában figyelembe véve a *Sony ff nyomtató* eladására vonatkozó decemberi különbség kivételes, a novemberi különbség viszont nem az.

Az InExp értékek az alacsonyabb szinten előforduló, az aktuális szinten nem látható kivételek jelzésére használhatók. Nézzük az *IBM asztali számítógép* júliusi és szeptemberi eladásait. Mindkét cella körül egy sötét, vastag keret látható, amely magas InExp értéket jelent. Úgy is határozhatunk, hogy tovább vizsgáljuk az *IBM asztali számítógép* értékesítési adatait, és lejjebb fúrunk a *terület* mentén. Az eredményül kapott adatok a 2.22. ábrán láthatók, ahol már a kivételes cellák is meg vannak jelölve. Ezek alapján azonnal szembetűnő a déli területen tapasztalható –39%-os, illetve –34%-os esés július és szeptember hónapokban.

Eladások átlagos értéke	Hónap											
Árucikk	jan.	febr.	márc.	ápr.	máj.	jún.	júl.	aug.	szept.	okt.	nov.	dec.
Sony ff nyomtató		9%	-8%	2%	-5%	14%	-4%	0%	41%	-13%	-15%	-11%
Sony színes nyomtató		0%	0%	3%	2%	4%	-10%	-13%	0%	4%	-6%	4%
HP ff nyomtató		-2%	1%	2%	3%	8%	0%	-12%	-9%	3%	-3%	6%
HP színes nyomtató		0%	0%	-2%	1%	0%	-1%	-7%	-2%	1%	-4%	1%
IBM asztali számítógép		1%	-2%	-1%	-1%	3%	3%	-10%	4%	1%	-4%	-1%
IBM laptop		0%	0%	-1%	3%	4%	2%	-10%	-2%	0%	-9%	3%
Toshiba asztali számítógép		-2%	-5%	1%	1%	-1%	1%	-5%	-3%	-5%	-1%	-1%
Toshiba laptop		1%	0%	3%	0%	-2%	-2%	-5%	3%	2%	-1%	0%
Logitech egér		2%	-2%	-1%	0%	4%	6%	-11%	2%	1%	-4%	0%
Ergo-way egér		0%	0%	2%	3%	1%	-2%	-2%	-5%	0%	-5%	8%

2.21. ábra | Az értékesítésben megfigyelhető változás minden árucikk–idő párosításhoz

Ezek a részletes kivételek egyáltalán nem voltak nyilvánvalók, amikor az adatokat a 2.21. ábrán látható árucikk–idő bontásban láttuk, a területekre nézve összesítve. Tehát az InExp érték hasznosnak bizonyul a kocka alacsonyabb szintű, kivételes celláinak a keresése során. Mivel a 2.22. ábrán nincs több cella, amely magas InExp értékkel rendelkezne, az adatokat visszagörgethetjük a 2.21. ábra állapotába, és ott egy másik cellából újra lefúrhatunk. Így használhatók a kivételmutatók az adatok érdekes eltéréseinek a felfedezésében. ■

Eladások átlagos értéke	Hónap											
Terület	jan.	febr.	márc.	ápr.	máj.	jún.	júl.	aug.	szept.	okt.	nov.	dec.
Észak		-1%	-3%	-1%	0%	3%	4%	-7%	1%	0%	-3%	-3%
Dél		-1%	1%	-9%	6%	-1%	-39%	9%	-5%	4%	1%	7%
Kelet		-1%	-2%	2%	-3%	1%	18%	-2%	11%	-3%	-2%	-1%
Nyugat		4%	0%	-1%	-3%	5%	1%	-18%	8%	5%	-8%	1%

2.22. ábra | Az *IBM asztali számítógép* értékesítésében megfigyelhető változás területenként

Hogyan számítjuk ki a kivételmutatókat? A SelfExp, InExp és PathExp mértékek a táblázatelemzéshez használt statisztikai módszereken alapulnak. Figyelembe veszik az összes csoportosítást (összesítést), amelyben az adott cella részt vehet. A cella értéke attól függően tekinthető kivételesnek, hogy mekkora mértékben tér el az alábbi statisztikai modell segítségével meghatározott várható értéktől. Az adott cella aktuális értéke és várható értéke közti különbséget **különbözetnek** nevezzük. Értelemszerűen, minél nagyobb a különbözet, annál kivételesebb a cellaérték. A különbözeti értékek összehasonlításához azok normalizálására van szükség, a különbözetekhez tartozó várható standard eltérés alapján. A cellaérték tehát kivételnek tekintendő, ha a normalizált különbözeti értéke meghalad egy előre rögzített küszöbértéket. A SelfExp, InExp és PathExp mértékek ezen a normalizált különbözeten alapulnak.

Egy adott cella várható értéke a cella magasabb szintű csoportosításainak a függvénye. Ha például adott egy kocka három dimenzióval: *A*, *B* és *C*, akkor a dimenziók mentén rendre az *i*-edik, *j*-edik és *k*-adik pozíciót elfoglaló cella várható értéke a használt statisztikai modell együtthatóinak: γ, γ_i^A, γ_j^B, γ_k^C, γ_{ij}^{AB}, γ_{ik}^{AC}, γ_{jk}^{BC} a függvénye. Az együtthatók azt tükrözik, hogy a részletesebb szinteken mennyire különböznek az értékek, a magasabb szintű összesítések alapján alkotott benyomások általánosítása alapján. Így a cellaérték kivételes minősége a cella alatti értékek kivételességétől függ. Ezért ha kivételt találunk, természetes, hogy ezt lefúrás segítségével tovább vizsgáljuk.

Hogyan lehet a felfedezésvezérelt feltáráshoz hatékonyan felépíteni az adatkockát? Ez a számítás három fázisból áll. Az első lépésben a kockát definiáló összesítő értékek, például `sum` vagy `count` kiszámítására kerül sor, amelyek között majd kivételeket keresünk. Több hatékony kockaszámító eljárás is létezik, mint például a 2.4.1. alfejezetben tárgyalt többutas tömbösszesítési technika. A második fázisban a modell illesztése történik meg, amely során a fent említett együtthatók meghatározására, majd ezek segítségével a standardizált különbözetek kiszámítására kerül sor. Ez átfedésbe kerülhet az első fázissal, mivel a számítások hasonlók. A harmadik lépésben a standardizált különbözetek

alapján a SelfExp, InExp és PathExp értékek kiszámítására történik meg. Ez a fázis a számítások tekintetében hasonló az első lépéshez, tehát a felfedezésvezérelt feltáráshoz használatos adatkockák építését hatékonyan el lehet végezni.

2.5.2. | Bonyolult összesítések több finomsági szinten: többtulajdonságú kockák

Az adatkockák elősegítik az adatbányász lekérdezések megválaszolását, mivel lehetővé teszik az összesített adatok számítását több finomsági szinten is. Ebben a részben a *többtulajdonságú kockákról* lesz szó, amelyek olyan összetett lekérdezéseket is képesek kiszámítani, amelyek több finomsági szinten több függő összesítést is tartalmaznak. Ezek a kockák a gyakorlatban nagyon hasznosak. Sok összetett adatbányász-lekérdezés megválaszolható többtulajdonságú kockák segítségével, lényegében a szokványos adatkockák egyszerű lekérdezésekkel járó kockaszámítási költsége mellett.

Ebben a részben valamennyi példa az *ElektroMind Vásárlás* adataiból való, ahol egy *árucikk*et egy értékesítési *terület*en egy üzleti napon (*év*, *hónap*, *nap*) megvesznek. Egy adott árucikk polcon töltött idejét hónapokban megadva a *polc*ban tároljuk. Egy adott területen az árucikk árát és az eladásokat (dollárban kifejezve) az *ár*ban, illetve az *értékesítés*ben tároljuk. A többtulajdonságú kockák tanulmányozásához segítségül nézzünk először példát egyszerű adatkockára.

2.17. példa | 1. lekérdezés: egyszerű adatkocka-lekérdezés. Adjuk meg az eladás összértékét 2000-ben, árucikkekre, területekre és hónapokra lebontva, a megfelelő részösszegekkel együtt.
Az 1. lekérdezés megválaszolásához olyan adatkockát hozunk létre, amely az eladás összértékét a következő nyolc finomsági fokon összesíti: {(*árucikk*, *terület*, *hónap*), (*árucikk*, *terület*), (*árucikk*, *hónap*), (*terület*, *hónap*), (*árucikk*), (*terület*), (*hónap*), ()}, ahol () jelöli a **minden**-t. Különböző technikák léteznek az efféle adatkockák hatékony számításához (2.4.1. alfejezet). ■

Az 1. lekérdezés olyan adatkockát használt, amit eddig ebben a fejezetben tanulmányoztunk. Az ilyen adatkockát egyszerű adatkockának nevezzük, mert nem tartalmaz függő összesítéseket.

Mit értünk függő összesítés alatt? Erre a következő lekérdezés vizsgálatával válaszolunk.

2.18. példa | 2. lekérdezés: bonyolult lekérdezés. Csoportosítsuk az adatokat az {*árucikk*, *terület*, *hónap*} összes részhalmaza szerint, és mindegyikhez adjuk meg a legmagasabb árat 2000-ben. A legdrágább eladott árucikkek összértékét is adjuk meg.

Egy ilyen lekérdezés megadása standard SQL-ben hosszú lenne, sok ismétlődést tartalmazna, és nehéz lenne optimalizálni és támogatni. A kiterjesztett SQL-szintaxis azonban a következő tömör formát eredményezi:

```
select    árucikk, terület, hónap, MAX(ár), SUM(R.értékesítés)
from      Vásárlás
```

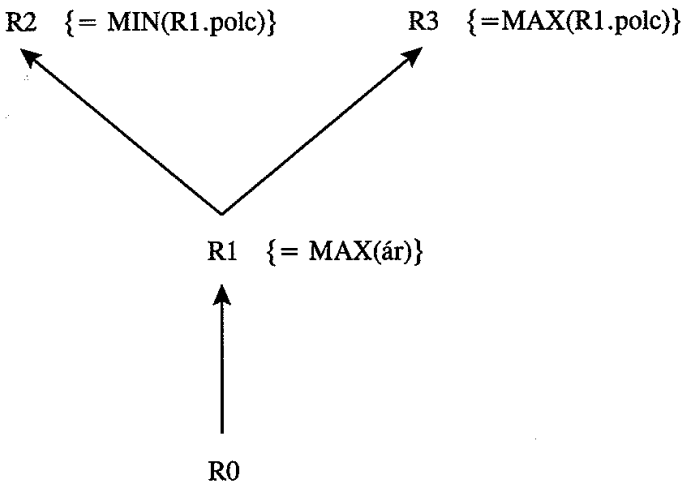
where év = 2000
cube by árucikk, terület, hónap: R
such that R.ár = MAX(ár)

A lekérdezés először azokat a sorokat választja ki, amelyek 2000-beli vásárlásokat jelölnek. A **cube by** záradék az *árucikk, terület* és *hónap* attribútumok minden lehetséges kombinációjára kiszámítja az összesítéseket (vagy csoportosításokat). Ez a **group by** záradék egy *n*-dimenziós általánosítása. A **cube by** záradékban megadott attribútumok a **csoportosító attribútumok**. Azok a sorok alkotnak egy csoportot, amelyeknek az összes csoportosító attribútum mentén azonos az értéke. Legyenek a csoportok $g_1, \ldots, g_r$. A lekérdezés minden g_i csoporthoz kiválasztja a csoportot alkotó sorok alapján a maximális árat: max_{g_i}-t. Az *R* változó egy csoportosító változó, amely azon sorok értékét veszi fel a g_i csoportban, amelyek az ár mentén azonosak max_{g_i}-vel (ahogy ez a **such that** záradékban meg van adva). A lekérdezés minden g_i csoporthoz kiszámítja az *R* által megadott sorok eladásainak összértékét és ezt az értéket adja vissza a g_i csoportosító attribútum értékeivel és a maximális árral együtt. Az eredményül kapott kocka egy **többtulajdonságú kocka**, amely támogatja az olyan összetett adatbányász-lekérdezéseket, amelyekhez különböző finomsági szinteken több függő összesítést kell kiszámítani. A 2. lekérdezésben például az eladások összértéke minden csoportnál függ a maximális árat megadó soroktól. ■

Most nézzünk egy másik példát.

2.19. példa | 3. lekérdezés: még bonyolultabb lekérdezés. Csoportosítsunk az {*árucikk, terület, hónap*} összes részhalmaza szerint, és minden csoporthoz adjuk meg a maximális árat 2000-ben. A legdrágább árucikkek polcon töltött idejéből adjuk meg a legrövidebbet, illetve a leghosszabbat. Adjuk meg még a legdrágább árucikkekből származó összeladás, és ezen belül a legrövidebb, illetve leghosszabb időt a polcon töltő árucikkekből származó összeladás értékét.

A 2.23. ábrán látható **többtulajdonságú kocka gráfja** segít átlátni az összesítések függőségi viszonyait a lekérdezésben. Minden csoportosító változóhoz tartozik egy csúcs, ezen-



2.23. ábra | Többtulajdonságú kocka gráfja a 3. lekérdezéshez

kívül van egy kezdő csúcs: R0. R0-ból indulva először a 2000-ben maximális árú sorok halmazát számítjuk ki (R1). A gráf alapján látszik, hogy az R2 és R3 csoportosító változók R1-től függnek, mivel R1-ből irányított él fut R2-be és R3-ba is. Ha egy többtulajdonságú kocka gráfjában egy irányított él fut R_i csoportváltozóból R_j-be, akkor ez azt jelenti, hogy R_j sorai mindig R_i sorainak egy részhalmazát képezik. Kiterjesztett SQL-ben ezt az $\mathbf{R_j}$ **in** $\mathbf{R_i}$ kifejezéssel jelöljük. Az R2-beli minimális polcéletű sorok például az R1-beli maximális árú sorok közül kerülnek ki, azaz **R2 in R1**. Hasonlóan, az R3-beli maximális polcéletű sorok az R1-beli maximális árú sorok közül kerülnek ki, azaz **R3 in R1**.

A gráf alapján a 3. lekérdezés kiterjesztett SQL-ben a következő:

```
select      árucikk, terület, hónap, MAX(ár), MIN(R1.polc), MAX(R1.polc),
            SUM(R1.értékesítés), SUM(R2.értékesítés), SUM(R3.értékesítés)
from        Vásárlás
where       év = 2000
cube by     árucikk, terület, hónap: R1, R2, R3
such that   R1.ár = MAX(ár) and
            R2 in R1 and R2.polc = MIN(R1.polc) and
            R3 in R1 and R3.polc = MAX(R1.polc)                    ■
```

Hogyan lehet a többtulajdonságú kockákat hatékonyan számítani? Ez a kockában felhasznált összesítőfüggvények típusától függ. A 2.2.4. alfejezetben volt arról szó, hogy egy összesítőfüggvény vagy *disztributív* (például `count()`, `sum()`, `min()` és `max()`), vagy *algebrai* (például `avg()`, `min_N()`, `max_N()`), vagy *teljes körű* [például `median()`, `mode()` és `rank()`]. A többtulajdonságú kockák ugyanezekbe az osztályokba sorolhatók.

A többtulajdonságú kocka típusa meghatározza a számításánál használt megközelítési módot. Az adatkockák hatékony számítására számos módszer létezik. Ezen algoritmusok alapvető stratégiája, hogy kihasználják a kockát definiáló többszörös felbontási szintek hálóstruktúráját, ahol a magasabb finomsági szinteket az alacsonyabb finomsági szintek alapján számítják. Ez a megoldás megfelelő a disztributív többtulajdonságú kockák esetén. A 2. lekérdezés egy disztributív többtulajdonságú kocka, mivel a számítások részenként, növekményesen is elvégezhetők, a magasabb finomsági szintű számításokhoz mindig csak az alacsonyabb szintű eredményeket használva. A `MAX(ár)` egy magasabb felbontású csoportban kiszámítható úgy, hogy vesszük az összes alacsonyabb felbontású csoport `MAX(ár)` értékei közül a maximálist. Hasonlóan, a `SUM(értékesítés)` egy magasabb felbontású csoportban kiszámítható az összes alacsonyabb felbontású csoport `SUM(értékesítés)` összegeként. Néhány hatékony kockaépítő algoritmus optimalizációs technikákat alkalmaz, amelyek az adatkockán belüli csoportok által megadott eredmények becsült méretén alapulnak. Mivel ez a méret egy többtulajdonságú kockában minden csoportra konstans, ugyanaz a becslő algoritmus használható a köztes eredmények becslésére is. Vagyis az egyszerű adatkockák hatékony számításához használt alapalgoritmusok ugyanúgy használhatók disztributív többtulajdonságú kockák esetén is az I/O-bonyolultság bármilyen növekedése nélkül. A CPU költségében felmerülhet elhanyagolható növekedés, ha a többtulajdonságú kocka összesítőfüggvénye mondjuk bonyolultabb egy egyszerű `SUM()`-nál. Az algebrai többtulajdonságú kockákat először disztributív több-

tulajdonságú kockákká kell alakítani, hogy ezeket az algoritmusokat alkalmazni tudjuk. A teljes körű többtulajdonságú kockák számítása néhány esetben lényegesen drágább a disztributív kockák számításánál, bár a CPU-költség általában elfogadható. Vagyis a többtulajdonságú kockák összehasonlítva az egyszerű adatkocka lekérdezésekkel nagyon kis többletköltség mellett használhatók bonyolult lekérdezések megválaszolásához.

2.5.3. | További fejlesztések

Létezik-e valamilyen stratégia a lekérdezések gyors megválaszolásához? Az efféle stratégiák arra koncentrálnak, hogy *közbülső visszajelzéseket* biztosítsanak a felhasználó felé. Például az **on-line összesítés** közben az adatbányászrendszer kiírhatja, hogy mi az, amit eddig megtudott, ahelyett, hogy a lekérdezés teljes lefutását megvárná és csak akkor jelezne vissza. Ezek a közelítő válaszok az adott adatbányász-lekérdezés esetén periodikusan frissülnek és finomodnak a számítási folyamat előrehaladtával. Minden becsléshez tartozik egy bizonyossági intervallum, amely további információval látja el a felhasználót arról, hogy az adott ponton mennyire tekinthető megbízhatónak a válasz. Ez elősegíti a rendszerrel való párbeszédet – a felhasználónak kialakul egy képe arról, hogy „jó" irányban próbálkozik-e anélkül, hogy meg kellene várnia a lekérdezés végét. Az on-line összesítés ugyan nem rövidíti meg a lekérdezéshez szükséges időt, az egész adatbányász folyamat azonban felgyorsulhat a rendszer interaktív volta miatt.

Egy másik megközelítés az **első *N* lekérdezés** alkalmazása. Tegyük fel, hogy az *ElektroMindnél* árult árucikkek milliói közül minket csak a legjobban menők érdekelnek. Ahelyett, hogy arra várnánk, amíg elkészül az összes árucikk listája eladás szerint csökkenő sorrendbe rendezve, mi csak az első *N* árucikket szeretnénk látni. Statisztikák használatával a lekérdezés feldolgozását úgy optimalizálhatjuk, hogy az egész rendezett lista helyett csak az első *N* árucikket kapjuk vissza. Ez gyorsabb válaszidőt eredményez, másrészt segíti a felhasználó (inter)aktivitásának a növelését és az elpazarolt erőforrások csökkentését.

2.6. | Az adattárház-kezeléstől az adatbányászatig

Hogyan kapcsolódik az adatbányászathoz az adattárház-kezelés és az OLAP? Ebben a részben az adattárházak használatát tanulmányozzuk az információfeldolgozásban, az analitikus feldolgozásban és az adatbányászatban. Bevezetjük az on-line analitikus bányászat (OLAM, On-Line Analytical Mining) fogalmát is, egy hathatós paradigmát, amely egyesíti az OLAP-ot az adatbányászati technológiákkal.

2.6.1. | Adattárházak használata

Az adattárházak és adatpiacok az alkalmazások széles körében használatosak. Az üzleti vezetők az adatelemzéshez és stratégiai döntések meghozatalához majdnem minden iparágban olyan adatokat használnak, amelyek gyűjtésére, egyesítésére, előfeldolgozására és

tárolására adattárházakban és adatpiacokon kerül sor. Sok cégnél nélkülözhetetlen szerepet kapnak az adattárházak a vállalati vezetés *tervezés-végrehajtás-értékelés* „önmagába visszatérő" visszajelző rendszerében. Az adattárházak kiterjedt használata figyelhető meg a banki és pénzügyi szektorban, a fogyasztási cikkek és a kiskereskedelmi forgalmazás területén, és az irányított termelésnél, például az igény alapú gyártásnál is.

Jellemzően, minél hosszabb ideje használnak egy adattárházat, az annál fejlettebb. Ez a fejlődés számos fázison keresztül történik. Kezdetben az adattárházat főleg jelentések készítésére és előre meghatározott lekérdezések megválaszolására használják; azután fokozatosan összegzett és részletes adatok elemzésére, ahol az eredmények jelentések és diagramok formájában jelennek meg. Később az adattárházat már stratégiai célokból használják, többdimenziós elemzések és kifinomult szeletelés és kockázás műveletek elvégzésére. Végül adatbányász eszközöket felhasználva az adattárházat tudásfeltárás és stratégiai döntéshozatal céljából is alkalmazzák. Ebben az összefüggésben az adattárházkezeléshez használt eszközök a következőképpen osztályozhatók: *hozzáférési és visszakeresési eszközök, jelentéskészítő eszközök, adatelemző eszközök* és *adatbányász eszközök.*

Az üzleti felhasználóknak valamilyen módon tudniuk kell, hogy mi van az adattárházban (a metaadatokon keresztül), hogyan lehet elérni az adattárház tartalmát, hogyan lehet megvizsgálni ezt az elemző eszközök segítségével, és miképpen lehet bemutatni az efféle vizsgálatok eredményét.

Háromféle adattárház-alkalmazás létezik: *információfeldolgozás, analitikus feldolgozás* és *adatbányászat.*

- Az **információfeldolgozás** támogatja a lekérdezéseket, az alapvető statisztikai elemzést és a jelentéskészítést kereszttáblázatok, táblázatok, diagramok és grafikonok segítségével. Az adattárházas információfeldolgozásban most aktuális irányvonal az alacsony költségű web alapú hozzáférési eszközök megteremtése, amelyek aztán webböngészőkbe építhetők be.
- Az **analitikus feldolgozás** támogatja az alapvető OLAP-műveleteket, beleértve a szeletelés és kockázást, a lefúrást, a felgörgetést és a pivotálást. Az analitikus feldolgozás általában történeti adatokon folyik, de mind összegzett, mind részletes formában. Az analitikus feldolgozás fő erőssége az információfeldolgozással szemben az adattárház adatainak többdimenziós adatelemzése.
- Az **adatbányászat** a tudásfeltárást támogatja rejtett minták és társítások kutatásával, analitikus modellek építésével, osztályozással és előrejelzéssel, és az eredmények vizuális eszközökkel való bemutatásával.

Hogyan viszonyul az adatbányászat az információfeldolgozáshoz és az analitikus feldolgozáshoz? Az információfeldolgozás a lekérdezések alapján hasznos információkat találhat. Az ilyen lekérdezések eredménye azonban csupán azokat az információkat tükrözi, amelyek egyébként is közvetlenül megtalálhatók az adatbázisban, vagy az összesítőfüggvények segítségével kiszámíthatók; nem fejezik ki az adatbázis mélyén nyugvó finom kis mintákat vagy szabályszerűségeket. Azaz az információfeldolgozás nem adatbányászat.

Az on-line analitikus feldolgozás már egy lépéssel közelebb áll az adatbányászathoz, amennyiben az adattárház a felhasználó által definiált részhalmazból több felbontási szin-

ten összegzett információt tud kinyerni. Az ilyen leírások ekvivalensek az 1. fejezetben tárgyalt osztály-/fogalomleírásokkal. Mivel az adatbányászrendszerek szintén képesek általánosított osztály-/fogalombányászatra, ez néhány érdekes kérdést vet fel: *Az OLAP-rendszerek foglalkoznak adatbányászattal? Az OLAP-rendszerek tulajdonképpen adatbányász rendszerek?*

Az OLAP és az adatbányászat működési területét különállónak lehet tekinteni: az OLAP egy adatösszegző/-összesítő *eszköz*, amely segít az adatelemzés egyszerűsítésében, míg az adatbányászat lehetővé teszi a nagy mennyiségű adatban rejtőző minták és érdekes tudás *automatizált kutatását*. Az OLAP-eszközök az interaktív adatelemzés egyszerűsítését és támogatását célozzák meg, az adatbányász eszközök célja pedig a folyamat lehető legnagyobb fokú automatizálása, a felhasználói irányítás megőrzésével. Ilyen értelemben az adatbányászat egy lépéssel a hagyományos on-line analitikus feldolgozás előtt van.

Az adatbányászat felfogható azonban szélesebb értelemben is, amely szerint mind az adatleírást, mind az adatmodellezést lefedi. Mivel az OLAP-rendszerek be tudják mutatni az adattárház adatainak az általános leírását, ezért az OLAP-függvények alapvetően a felhasználó által irányított adatösszegzésre és összehasonlításra valók (fúrással, pivotálással, szeleteléssel, kockázással és más műveletek segítségével). Ezek, bár korlátozott függvények, már adatbányászati funkciókat látnak el. Mégis e felfogás szerint az adatbányászat sokkal szélesebb kört fed le, mint az egyszerű OLAP-műveletek, hiszen nem csak adatösszegzést és összehasonlítást hajt végre, hanem ezenkívül még az adatok társítását, osztályozását, előrejelzését, klaszterezését, idősorok elemzését és más adatelemző feladatokat is.

Az adatbányászat nincs megszorítva csupán az adattárházban tárolt adatok elemzésére. Az adattárházban elérhető összegzett adatoknál sokkal részletesebb felbontású adatok elemzésére is lehetőség van. Sőt, elemezhetünk tranzakciós, tér-, szöveges és multimédiás adatokat is, amelyeket az aktuális többdimenziós adatbázis-technológiával nehéz modellezni. Ebben az értelemben az adatbányászat az adatbányászati funkciók és a kezelt adatok bonyolultságát illetően szélesebb kört fed le, mint az OLAP.

Mivel az adatbányászat az OLAP-nál jobban automatizált és mélyebb elemzéseket foglal magába, ezért azt várhatjuk, hogy az alkalmazási területe is kiterjedtebb. Az adatbányászat segíti az üzleti vezetőket abban, hogy megfelelőbb ügyfeleket kutassanak fel és érjenek el, továbbá hogy megszerezzék azokat a kulcsfontosságú üzleti információkat, amelyek birtokában előmozdítható a piaci részesedés és a nyereség növelése. Ezenkívül az adatbányászat segítségével a vezetők megismerhetik a különböző ügyfélcsoportok jellegzetességeit, és ennek megfelelően optimális árazási stratégiákat dolgozhatnak ki, az árucikkek intuitív alapú csoportosításán is javíthatnak a vásárlási minták figyelembevételével, csökkenthetik a reklámköltségeket, ugyanakkor növelhetik a reklámozás általános hatékonyságát.

2.6.2. | Az on-line analitikus feldolgozástól az on-line analitikus bányászatig

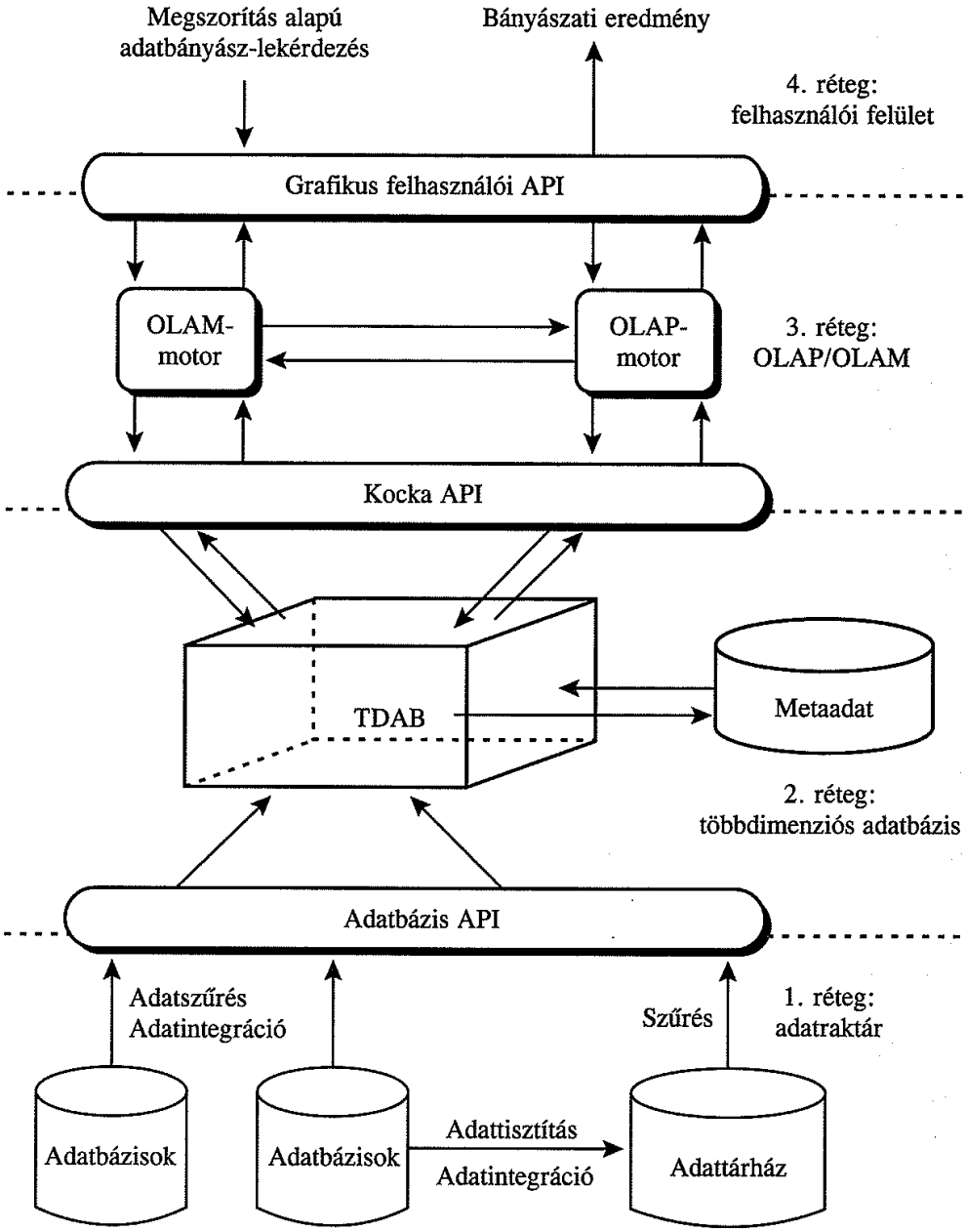
Az adatbányászat területén alapos kutatásokat végeztek a különböző felületeken való bányászatot illetően. A vizsgált felületek közé tartozik a tranzakciós adatbázis, a relációs adatbázis, a téradatbázis, a szöveges adatbázis, az idősor-adatbázis, a közönséges fájlok, az adattárházak és így tovább.

Az adatbányászrendszerek sok különböző modellje és felépítése között az **on-line analitikus bányászat (OLAM) (OLAP-bányászatként** is ismert), amely egyesíti az on-line analitikus feldolgozást (OLAP) a többdimenziós adatbázisok adat- és tudásbányászatával, a következők miatt különösen fontos szerepet kapott:

- **Jó minőségű adatok az adattárházakban** – A legtöbb adatbányász eszköznek egyesített, konzisztens és tisztított adatokon kell dolgoznia, amely költséges adattisztító, adatátalakító és adategyesítő előfeldolgozási lépéseket igényel. Az ilyen lépések során létrejött adattárház a jó minőségű adatok értékes forrását jelenti mind az OLAP, mind az adatbányászat számára. Vegyük észre, hogy az adatbányászat is értékes eszköze lehet az adattisztításnak és az adategyesítésnek is.
- **Elérhető egy, az adattárházat körülvevő információfeldolgozó infrastruktúra** – Átfogó információfeldolgozó infrastruktúrákat módszeresen építettek vagy fognak építeni adattárházak köré, amihez hozzátartozik az adatok elérése, egyesítése, véglegesítése, több heterogén adatbázis átalakítása, ODBC/OLE DB kapcsolatok, webelérési és szolgáltatási lehetőségek, jelentéskészítés és OLAP-elemzőeszközök. Bölcsebb a rendelkezésre álló infrastruktúrát a lehető legjobban kihasználni, mint mindent újra felépíteni.
- **OLAP alapú kutató adatelemzés** – A hatékony adatbányászathoz kutató adatelemzésre van szükség. A felhasználó gyakran áttekinti az egész adatbázist, kiválasztja a releváns részt, ezt több finomsági szinten elemzi, majd az így szerzett tudást/eredményeket különböző formákban mutatja be. Az on-line analitikus bányászat lehetővé teszi az adatok különböző részhalmazainak különböző absztrakciós szinteken történő adatbányászatát egy adatkocka és a köztes adatbányászati eredmények fúrásával, pivotálásával, szűrésével, kockázásával és szeletelésével. Ez az adat-/tudásmegjelenítő eszközökkel együtt nagyban fokozza a kutató adatbányászat teljesítményét és rugalmasságát.
- **Az adatbányász függvények on-line kiválasztása** – A felhasználó gyakran nem tudja, hogy milyen fajta tudást szeretne bányászni. Az OLAP és az adatbányász függvények egyesítésével az on-line analitikus bányászat biztosítja a felhasználónak azt a rugalmasságot, hogy kiválaszthassa a kívánt adatbányász függvényeket és dinamikusan cserélgesse az adatbányászati feladatokat.

Architektúra az on-line analitikus bányászathoz

Egy OLAM-szerver az adatkockák analitikus bányászatát hasonló módon végzi, mint ahogy egy OLAP-szerver az on-line analitikus feldolgozást. Integrált OLAM- és OLAP-felépítést láthatunk a 2.24. ábrán, ahol mind az OLAM-, mind az OLAP-szerver egy grafikus felhasználói API-felületen (Application Program Interface, alkalmazásprogramozói felület) keresztül fogadja a felhasználó on-line lekérdezéseit (vagy utasításait), és az adatkocka elemzését egy kocka API-n keresztül végzi. Az adatkocka elérését egy metaadat címtár irányítja. Az adatkockát több adatbázisnak a TDAB API-n keresztüli elérésével és/vagy egyesítésével hozhatjuk létre és/vagy az adattárház szűrésével az adatbázis API-n keresztül, amely OLE DB vagy ODBC-kapcsolatokat támogathat. Mivel egy OLAM-szerver több adatbányászati feladatot is elláthat, mint például fogalomleírás, társítás, osztályozás, előrejelzés, klaszterezés, idősorelemzés és így tovább, ezért ez általában több integrált adatbányászmodult tartalmaz, és kifinomultabb az OLAP-szervernél.



2.24. ábra | Integrált OLAM- és OLAP-architektúra

A könyv következő fejezeteit az adatbányászati technikák tanulmányozásának szenteljük. Láthattuk, hogy az ebben a fejezetben bemutatott adattárház-kezelés és OLAP-technológiák nélkülözhetetlenek az adatbányászat tanulmányozásához. Ez azért van így, mert az adattárház-kezelés nagy mennyiségű tiszta, szervezett és összegzett adatot biztosít a felhasználó számára, amely nagyban elősegíti az adatbányászatot. Ahelyett például, hogy minden eladási tranzakció részleteit tárolná, az adattárház tárolhatja a tranzakciók összegzését, mondjuk az árucikkek típusa szerint minden üzlethez megadva, vagy egy magasabb szinten: országonként megadva. Az OLAP azon képessége, hogy az adattárházban tárolt összegzett adatok többféle és dinamikus képét nyújtja, biztos alapot jelent a sikeres adatbányászathoz.

Emellett úgy gondoljuk, hogy az adatbányászatnak emberközpontú folyamatnak kell lennie. Ahelyett, hogy az adatbányászrendszert különféle minták és tudás automatikus előállítására kérné, a felhasználónak gyakran kell párbeszédet folytatnia a rendszerrel a

kutató adatelemzés végrehajtása érdekében. Az OLAP jó példával jár elöl az interaktív adatelemzés tekintetében és biztosítja a szükséges előkészületeket a kutató adatelemzéshez. Vegyük például a társítási minták felderítését. A tranzakciók közti társítások primitív (azaz alacsony) adat szintű bányászata helyett a felhasználóknak lehetővé kell tenni, hogy bármely dimenzió mentén felgörgető műveleteket adhassanak meg. A felhasználó például az *árucikk* dimenzió mentén akar felgörgetni úgy, hogy az adott tv-készülékekre vonatkozó vásárlások helyett a márkákra (például SONY vagy Panasonic) vonatkozó vásárlásokat lássa. A felhasználók az érdekes társítások keresése közben eljuthatnak a tranzakciós szintről a vásárló vagy egy vásárló típusú szintre. Az adatbányászat egy efféle OLAP-stílusa jellemző az OLAP-bányászatra. Az adatbányászat alapelveinek a tanulmányozása során a következő fejezetekben különös hangsúlyt fektetünk az OLAP-bányászatra, azaz *az adatbányászat és az OLAP-technológia egyesítésére.*

2.7. | Összefoglalás

- Az **adattárház** *témaorientált, integrált, időben változó* és *nem felejtő* adatgyűjtemény, amely az üzleti döntéshozatal támogatására szolgál. Az adattárházat az operatív adatbázistól számos tényező különbözteti meg. Mivel a két rendszer elég eltérő működést biztosít és más típusú adatokat igényel, az adattárházat szükségszerűen az operatív adatbázistól elválasztva kell működtetni.
- A **többdimenziós adatmodellt** jellemzően vállalati *adattárházak* és *osztály szintű adatpiacok* tervezéséhez használják. Egy ilyen modell alkalmazhatja a *csillagsémát*, a *hópehelysémát* vagy a *csillagképsémát*. A *többdimenziós modell* magja az **adatkocka**, amely a *tények* (vagy *mértékek*) nagy halmazából és számos *dimenzióból* áll. A dimenziók olyan egyedek vagy nézőpontok, amelyekkel kapcsolatban a vállalat feljegyzéseket kíván tárolni, és természetüket tekintve hierarchikus jellegűek.
- A **fogalmi hierarchiák** az attribútumok vagy dimenziók értékeit fokozatos absztrakciós szintekbe szervezik. Többféle absztrakciós szinten való bányászathoz hasznosak.
- Az **on-line analitikus feldolgozás (OLAP)** adattárházakban vagy adatpiacokon hajtható végre a többdimenziós adatmodell felhasználásával. A jellemző OLAP-műveletek közé tartozik a *felgörgetés*, *fúrás (le-, át-, keresztül-)*, *szeletelés és kockázás*, *pivotálás (forgatás)*, továbbá statisztikai műveletek, például osztályozás, mozgó átlag és növekedési mutatók számítása és így tovább. Az OLAP-műveletek az adatkocka-szerkezet használatával hatékonyan implementálhatók.
- Az adattárházak gyakran **háromszintű felépítést** alkalmaznak. Az alsó szint egy *tárházadatbázis-szerver*, amely jellemzően relációs adatbázisrendszer. A középső szint egy *OLAP-szerver*, a felső szint pedig egy *kliens*, amely lekérdező és jelentéskészítő eszközöket tartalmaz.
- Az OLAP-szerverek használhatnak **relációs OLAP-ot (ROLAP)**, **többdimenziós OLAP-ot (MOLAP)** vagy **hibrid OLAP-ot (HOLAP)**. A ROLAP-szerver egy kiterjesztett relációs ABKR-t használ, amely a többdimenziós adatok OLAP-műveleteit standard relációs műveletekre képezi le. A MOLAP-szerver a többdimenziós adatnézeteket közvetlenül tömbszerkezetbe képezi le. A HOLAP-szerver kombinálja a kettőt. Például ROLAP-ot használ a történeti adatokra, miközben a gyakori hozzáférést igénylő adatokat egy külön MOLAP-tárban tartja fenn.

- Egy adatkocka a **részkockák hálójából** áll, ahol egy-egy részkocka az adott többdimenziós adat mindig más szintű összegzésének felel meg. A **részleges megvalósítás** a részkockák egy részhalmazának a szelektív kiszámítására utal. A **teljes megvalósítás** a háló összes részkockájának a kiszámítását jelenti. Ha a kockákat MOLAP segítségével implementáljuk, akkor használható a **többutas tömbösszesítés**. Ezzel a technikával néhány összesítés számítása „átfedi" egymást, így a teljes megvalósítás hatékonyan számítható.
- Az indextechnikák bevezetésével az OLAP-lekérdezések feldolgozása hatékonyabbá tehető. A **bittérképindexelésben** minden attribútumnak megvan a saját bittérképindex-táblázata. A bittérképindexelés az összekapcsolás, összesítés és összehasonlítás műveleteket bitaritmetikára vezeti vissza. Az **összekapcsoló index** a relációs adatbázis kettő vagy több relációjának az összekapcsolható sorait jegyzi fel, így csökkenti az OLAP-összekapcsolás műveleteinek általános költségét. A **bittérképezett összekapcsoló indexelést**, amely a bittérkép- és az összekapcsoló módszert egyesíti, az OLAP-lekérdezések feldolgozásának további gyorsításához lehet használni.
- Az adattárház-**metaadatok** definiálják a tárházobjektumokat. A metaadatraktár részletekkel szolgál a következőkkel kapcsolatban: tárházszerkezet, adattörténet, az összegzéshez használt algoritmusok, az adatforrások leképezése tárházformátumba, rendszerteljesítmény, üzleti szóhasználat és hasonló témák.
- Egy adattárházban **háttéreszközök** és **segédprogramok** működnek a tárházadatok betöltése és frissítése érdekében. Ezek a következő feladatokat látják el: adatkinyerés, adattisztítás, adatátalakítás, feltöltés, frissítés és tárházkezelés.
- Az adatkockák **felfedezésvezérelt feltárása** előszámított mértékeket és vizuális jeleket használ a kivételes adatok megjelölésére minden összesítési szinten, így vezeti a felhasználót az adatelemző folyamatban. A **többtulajdonságú kockák** olyan bonyolult lekérdezéseket számítanak ki, amelyek több finomsági szinten többféle függő összesítést is tartalmaznak. A többtulajdonságú kockák és a felfedezésvezérelt feltárásra használt kockák számítása hatékonnyá tehető a hatékony, standard adatkocka-számító algoritmusok kihasználásával.
- Az adattárházakat használják *információfeldolgozáshoz* (lekérdezés és jelentéskészítés), *analitikus feldolgozáshoz* (amely lehetővé teszi, hogy a felhasználó OLAP-műveletek segítségével az összegzett adatoktól a részletes adatokig eljusson), és *adatbányászathoz* (amely a tudásfeltárást támogatja). Az OLAP alapú adatbányászatot **OLAP-bányászatnak** vagy on-line analitikus bányászatnak (**OLAM**) nevezzük, amely az OLAP-bányászat interaktív és kutató természetét hangsúlyozza.

2.8. | Feladatok

2.1. A többféle heterogén információforrás egyesítéséhez miért részesíti sok vállalat előnyben a *frissítésvezérelt megközelítést* (amely adattárházat hoz létre és használ) a *lekérdezésvezérelt megközelítéssel* szemben (amely becsomagolókat és egyesítőket alkalmaz)? Adjunk meg olyan helyzeteket, ahol a lekérdezésvezérelt megközelítés a kívánatosabb!

2.2. Röviden hasonlítsuk össze a következő fogalmakat! Használhatunk példát a fogalmak magyarázatához.
(a) Hópehelyséma, csillagkép, csillaghálós lekérdező modell
(b) Adattisztítás, adatátalakítás, frissítés
(c) Felfedezésvezérelt kocka, többtulajdonságú kocka, virtuális tárház

2.3. Tegyük fel, hogy egy adattárház az *idő*, *orvos* és *páciens* dimenziókból, továbbá a *darab* és *díj* mértékekből áll, ahol a *díj* az orvosi vizsgálatért fizetendő összeg.
(a) Soroljunk fel három olyan sémaosztályt, amelynek elterjedt a használata az adattárház-modellezés területén!
(b) Az (a) feladatban megadott sémák egyikét használva készítsünk sémagráfot a fenti adattárházhoz!
(c) A [*nap, orvos, páciens*] alapkockából kiindulva milyen konkrét *OLAP-műveleteket* kell végrehajtani ahhoz, hogy az egyes orvosoknak 2000-ben fizetett díjak összegét kilistázzuk?
(d) Ugyanezt a lekérdezést írjuk meg SQL-ben, a következő relációs adatbázissémát felhasználva: *honorárium(nap, hónap, év, orvos, kórház, páciens, darab, díj)*.

2.4. Az *Csúcs-Egyetem* adattárháza a *diák*, *szak*, *félév* és *szakfelelős* dimenziókból továbbá a *darab* és *átl_jegy* mértékekből áll. A legalacsonyabb fogalmi szinten (például adott diák, szak, félév és szakfelelős kombinációban) az *átl_jegy* a diák jegyét jelenti, amit az adott szakon szerzett. Magasabb fogalmi szinteken az *átl_jegy* az adott kombinációhoz tartozó átlagosan elért jegyet jelenti.
(a) Rajzoljunk az adattárházhoz *hópehelysémagráfot*!
(b) A [*diák, szak, félév, szakfelelős*] alapkockából kiindulva milyen konkrét *OLAP-műveleteket* (például a *félév*ek felgörgetése *év*ekké) kell végrehajtani ahhoz, hogy kilistázzuk az egyes diákok által elért átlagjegyet az *informatika* szakon?
(c) Ha minden dimenziónak a **minden**nel együtt öt szintje van (például *diák* < *szak* < *évfolyam* < *egyetem* < **minden**), akkor hány részkocka van ebben a kockában (az alap- és a csúcskockával együtt)?

2.5. Egy adattárház a *dátum*, *néző*, *hol* és *meccs* dimenziókból továbbá a *darab* és *díj* mértékekből áll. A *díj* az adott helyen az adott meccsre való belépés díját adja meg. A nézők három kategóriába esnek: diák, felnőtt és nyugdíjas, és minden kategóriában más a belépési díj mértéke.
(a) Rajzoljunk az adattárházhoz *csillagsémagráfot*!
(b) A [*dátum, néző, hol, meccs*] alapkockából kiindulva milyen konkrét *OLAP-műveleteket* kell végrehajtani ahhoz, hogy kilistázzuk a diákok által 2000-ben a Népstadionban fizetett belépti díjak összegét?
(c) Az adattárház-kezelésben hasznos a *bittérképindexelés*. A fenti kockát használva példaként, röviden fejtsük ki a bittérképindex-szerkezet használatából fakadó előnyöket és problémákat!

2.6. Tervezzünk adattárházat egy helyi meteorológiai intézet számára! Az intézetnek körülbelül 1000 szondája van elhelyezve különböző vízi és szárazföldi pozíciókon, amelyek óránként alapvető meteorológiai adatokat gyűjtenek (légköri nyomás, hőmérséklet, csapadék). Minden adatot a központi állomásnak küldenek át, amely már 10 éve gyűjti az efféle mérések eredményeit. Az általunk készített tervnek elő

kell segítenie a hatékony lekérdezést és az on-line analitikus feldolgozást, továbbá az általános meteorológiai minták levezetését a többdimenziós térben.

2.7. Az adatkockában a *mértékek számítását* illetően:

(a) Soroljuk fel a mértékek három osztályát, az adatkocka számításánál alkalmazott összesítőfüggvények fajtájától függően!

(b) Az *idő*, *cím* és *termék* dimenziókkal rendelkező adatkocka esetén melyik kategóriába tartozik a *szórásnégyzet* függvény? Írjuk le, hogyan számítjuk ki ennek az értékét, ha a kocka sok darabba van vágva!

Segítség: a *szórásnégyzet* kiszámításához használt képlet: $\frac{1}{n}\sum_{i=1}^{n}(x_i)^2 - x_i^2$, ahol $\bar{x}_i$ az x_i-k átlaga.

(c) A függvény legyen most *„a 10 legnagyobb értékű vásárlás"*. Vitassuk meg, hogyan lehet ezt a mértéket az adatkockában hatékonyan kiszámítani!

2.8. Tegyük fel, hogy egy adatkockában három mértéket kell feljegyeznünk: min, average és median. Tervezzünk hatékony számítási és tárolási módszereket az egyes mértékekhez, feltéve hogy lehetőség van a kockából az *adatok növekményes* (azaz kis részletenkénti) *törlésére*.

2.9. Elterjedt adattárház-megvalósítási módszer a többdimenziós adatbázis, azaz adatkocka létrehozása. Sajnos ennek során gyakran hatalmas, de nagyon ritka többdimenziós mátrix jön létre.

(a) Mutassunk be egy példát egy ilyen hatalmas és ritka adatkockára!

(b) Tervezzünk olyan megvalósítási módszert, amely elegánsan túlteszi magát ezen a ritkamátrix-problémán! Az adatszerkezetet és a tárigényt részletesen mutassuk be, és azt is, hogy hogyan lehet ezzel a szerkezettel az adatokat visszakeresni!

(c) Módosítsuk a (b) feladatban megadott módszert úgy, hogy tudjuk kezelni a *növekményes adatfrissítést*! Magyarázzuk el, hogy mi áll az új elgondolás mögött!

2.10. Az adattárház-technológiában többdimenziós nézeteket implementálhatunk a relációs adatbázis-technika (ROLAP) vagy a többdimenziós adatbázis-technika (MOLAP) vagy a hibrid adatbázis-technika (HOLAP) segítségével.

(a) Röviden írjuk le mindegyik megvalósítási technikát!

(b) Magyarázzuk el, hogy a következő függvényeket hogyan lehetne implementálni az egyes technikák esetén:

(i) adattárház generálása (összesítéssel együtt),

(ii) felgörgetés,

(iii) lefúrás,

(iv) növekményes frissítés.

Melyik implementációs technikát részesítsük előnyben és miért?

2.11. Tegyük fel, hogy egy adattárházban 20 dimenzió van, és mindegyiknek körülbelül öt finomsági szintje létezik.

(a) A felhasználókat főleg ugyanaz a négy dimenzió érdekli, ezeknek három szintjét használják gyakran a felgörgetéshez és a lefúráshoz. Hogyan tervezzük meg az adatkocka szerkezetét, hogy ezeket a preferenciákat hatékonyan érvényesítse?

(b) Időnként a felhasználó *keresztül* akar *fúrni* a kockán egészen a nyers adatokig egy vagy két dimenzió mentén. Hogyan lehetne ezt elérni?

2.12. Tegyük fel, hogy az alapkockának három dimenziója van: A, B és C, a következő mennyiségű cellákkal: $|A| = 1\ 000\ 000$, $|B| = 100$, $|C| = 1000$. Tegyük fel még, hogy mindegyik dimenzió 10 egyenlő részre van osztva a daraboláshoz.
(a) Feltéve, hogy mindegyik dimenziónak csak egyetlen szintje van, rajzoljuk le a kocka teljes hálóját!
(b) Mekkora a kiszámított kocka *sűrű* kocka esetén, ha minden cellában egy 4 bájtos mértéket tárolunk?
(c) Adjuk meg a darabok kiszámításának azt a sorrendjét, amelyik a legkevesebb tárat igényli, és számítsuk ki a kétdimenziós síkok számításához szükséges központi memória nagyságát!

2.13. Vegyük a következő *többtulajdonságú kocka* lekérdezést: az {*árucikk, terület, hónap*} összes részhalmaza szerint csoportosítva adjuk meg minden csoporthoz a 2000-ben a polcon eltöltött legrövidebb idő hosszát, és az olyan árucikkek eladásából származó összbevétel értékét, amelyek olcsóbbak 100 Ft-nál és a polcon eltöltött idejük a minimális 1,25-szöröse és 1,5-szöröse közé esik.
(a) Rajzoljuk fel a lekérdezéshez tartozó többtulajdonságú kocka gráfot!
(b) Adjuk meg a lekérdezést kiterjesztett SQL-ben!
(c) Vajon ez *disztributív* többtulajdonságú kocka? Miért? Miért nem?

2.14. Mi a különbség az adattárház felhasználásának a három fő típusa között: *információfeldolgozás*, *analitikus feldolgozás* és *adatbányászat*? Részletezzük, hogy mi motiválja az *OLAP-bányászatot (OLAM)*!

2.9. | Irodalom

Szép számú bevezető szintű tankönyv foglalkozik az adattárház-kezeléssel és az OLAP-technológiával. Ezek között kell megemlíteni Inmon [Inm96], Kimball [Kim96], Berson és Smith [BS97], és Thomsen [Tho97] könyveit. Chaudhuri és Dayal [CD97] általános áttekintést nyújt a témában.

A döntéstámogató rendszerek története az 1960-as évekig nyúlik vissza. A többdimenziós adatok elemzésére létrehozott nagy adattárházak ötlete azonban Coddtól [CCS93] származik, aki megalkotta az *OLAP* kifejezést az *on-line analitikus feldolgozáshoz*. Az OLAP-tanácsot 1995-ben alapították. Widom [Wid95] számos kutatási problémát azonosított az adattárház-kezelés területén. Kimball [Kim96] áttekinti az SQL hiányosságait az üzleti életben gyakori összehasonlítások támogatását illetően. Az OLAP-rendszerek és a statisztikai adatbázisok áttekintő összehasonlítását adja Shoshani [Sho97].

A DMQL adatbányászati lekérdező nyelvet Han és mások [HFW+96a] ajánlották. Az adatbányászati lekérdező nyelvekről bővebben a 4. fejezetben lesz szó. Adatbányászathoz használható más SQL alapú nyelveket Imielinski, Virmani és Abdulghani [IVA96], Meo, Psaila és Ceri [MPC96], és Baralis és Psaila [BP97] indítványoztak.

Gray és mások [GCB+97] hozakodtak elő az adatkocka ötletével, mint olyan relációs összesítő művelettel, amely a csoportosítást, kereszttáblázatokat és a részösszeget általánosítja. Harinarayan, Rajaraman és Ullman [HRU96] vetettek fel egy mohó algoritmust az adatkocka számításában a részkockák részleges megvalósításához. Agarwal és mások [AAD+96] különböző módszereket ajánlottak a többdimenziós összesítések hatékony szá-

mításához ROLAP-szerverek esetén. A 2.4.1. alfejezetben tárgyalt darab alapú többutas tömbösszesítő módszer a MOLAP adatkocka-számításban Zhao, Deshpande és Naughton [ZDN97] ötlete. Az adatkockák gyors számításához további módszerek találhatók Beyer és Ramakrishnan [BR99], és Ross és Srivastava [RS97] publikációiban. Sarawagi és Stonebraker [SS94] fejlesztettek ki egy darab alapú számítási technikát a nagy, többdimenziós tömbök hatékony szervezéséhez. A jéghegy típusú lekérdezéseket Fang, Shivakumar, Garcia-Molina és mások [FSGM+98] írja le.

A relációs lekérdezések feldolgozásának a felgyorsítását célzó összekapcsoló indexek használatát Valduriez [Val87] javasolta. O'Neil és Graefe [OG95] ajánlotta a bittérképezett összekapcsoló indexelést az OLAP alapú lekérdezések felgyorsításához. A bittérképezés és más, nem hagyományos indexelési technikák teljesítményét tárgyalja O'Neil és Quass [OQ97].

Az OLAP-lekérdezések hatékony feldolgozását célzó megvalósított részkockák kiválasztására vonatkozó kutatások Chaudhuri és Dayal [CD97], Harinarayan, Rajaraman és Ullman [HRU96], és Sristava és mások [SDJL96] művei. A kockaméret becsléséhez használható módszereket ír le Deshpande és mások [DNR+97], Ross és Srivastava [RS97], és Beyer és Ramakrishnan [BR99]. A többdimenziós adatbázisok modellezésénél alkalmazható műveleteket javasol Agrawal, Gupta és Sarawagi [AGS97].

Megjelent néhány frissebb tanulmány a felfedezésorientált adatkockák megvalósítását illetően. Ide tartozik Sarawagi, Agrawal és Megiddo [SAM98] az OLAP-adatkockák felfedezésvezérelt feltárásával, és Ross, Srivastava és Chatziantoniou [RSC98] a többtulajdonságú adatkockák létrehozásával. Hellerstein, Haas és Wang [HHW97], illetve Hellerstein és mások [HAC+99] írnak le módszereket arra vonatkozóan, hogy hogyan lehet a lekérdezéseket gyorsan megválaszolni az on-line összesítések alkalmazásával. Az első N lekérdezések becsléséhez használható technikákat ajánl Carey és Kossman [CK98], és Donjerkovic és Ranakrishnan [DR99]. Az OLAM-módszertant tárgyalja Han [Han98].

3. FEJEZET

Az adatok előfeldolgozása

Napjaink valós adatbázisai – rendszerint hatalmas, sokszor jó néhány gigabájtos méretük révén – igen gyakran tartalmaznak zajos, hiányos és inkonzisztens adatokat. Felmerül a kérdés: *Hogyan lehet előkészíteni, előfeldolgozni az adatokat, azok minőségének, és a bányászat eredményességének javítása érdekében? Milyen előzetes feldolgozás könnyítheti meg a bányászat folyamatát, illetve növelheti annak hatékonyságát?*

Az adatok előfeldolgozására számos módszer létezik. Az *adattisztítás* segítségével eltávolítható az adatokból a zaj, az inkonzisztens adatok pedig kijavíthatók. Az *adatok integrálása* során a különböző forrásból származó adatokat egyetlen koherens struktúrában (például adattárházban, adatkockában) egyesítjük. Az adatokat különféle *transzformációknak* vethetjük alá – ilyen a normalizálás is. Ez utóbbi például javíthatja a távolságmérést is használó adatbányászati algoritmusok pontosságát és hatékonyságát. Az *adatredukció* összevonással, az ismétlődő tulajdonságok figyelmen kívül hagyásával vagy klaszterezéssel csökkenti az adatok méretét. Ezek a módszerek a bányászat előtt alkalmazva jelentősen növelhetik a feltárt minták minőségét, illetve csökkenthetik az adatbányászatra fordított időt.

Ebben a fejezetben az adatok előfeldolgozásának módszereit fogjuk megismerni. Ezeket a következő csoportokra osztjuk: adattisztítás, adatok integrálása és transzformálása, valamint adatok redukálása. Az adatok redukálásának egy alternatív formájával, az ún. fogalmi hierarchiáknak adatdiszkretizációra való felhasználásával is foglalkozunk. A fogalmi hierarchiák ezenfelül használhatók az absztrakció több szintjén végzett bányászatban is. Látni fogjuk továbbá, hogy miként generálhatók automatikusan a fogalmi hierarchiák a rendelkezésre álló adatokból.

3.1. | Az adatok előfeldolgozásának szükségessége

Képzeljük el, hogy az *ElektroMind* cég menedzsereként azt a feladatot kapjuk, hogy analizáljuk a vállalat adatait az áruházunk eladási kvótái szempontjából. Rögtön hozzá is látunk a feladathoz. Alaposan megvizsgáljuk a cég adatbázisát és adattárházát, azonosítjuk és kiválasztjuk azokat az attribútumokat vagy dimenziókat, amelyeket az elemzésünk során figyelembe akarunk venni, például *árucikk, ár* és *eladott_darabszám*. Hoppá! Szembeötlő, hogy számos sorban bizonyos attribútumok nem rendelkeznek értékkel. Analízisünkbe azt az információt is bele szeretnénk venni, hogy vajon minden eladott árucikket

meg is hirdettünk-e előzőleg, de észrevesszük, hogy az erre vonatkozó adatokat nem rögzítették. Adatbázisrendszerünk felhasználói továbbá hibákról, szokatlan értékekről, bizonyos tranzakciók esetében pedig inkonzisztens adatokról tesznek jelentést. Más szóval, az adatbányászati módszerekkel elemezni kívánt adatok **hiányosak** (bizonyos attribútumok értékei részben vagy teljes egészében hiányoznak, illetve csak átlagértékek állnak rendelkezésre), **zajosak** (hibákat vagy szélsőséges értékeket tartalmaznak, melyek eltérnek a várttól) és **inkonzisztensek** (például az áruházaknak az árucikkek osztályozásában használt kódjai között diszkrepanciák mutatkoznak). Isten hozott a valóságos világban!

A gyakorlatban előforduló nagyméretű adatbázisokban és adattárházakban gyakran hemzsegnek a hiányos, zajos és inkonzisztens adatok. Hiányos adatok számos okból előfordulhatnak. A számunkra érdekes jellemzők, például az eladási tranzakciók esetében a vevőre vonatkozó információk, nem mindig hozzáférhetők. Más adatok egyszerűen azért nem szerepelnek, mert a bevitel időpontjában ezt nem tartották fontosnak. Lehet, hogy releváns adatokat félreértések vagy a berendezések hibája miatt nem rögzítettek. Olyan adatok, amelyek más rögzített adatokkal inkonzisztensek voltak, törlésre kerülhettek. Mindezeken túl az is lehetséges, hogy nem végeztek naplózást, vagy bizonyos módosítások elkerülték a figyelmüket. Egyes hiányzó adatokat, különösen olyan sorok esetében, ahol csak néhány attribútum hiányzik, nekünk kell kikövetkeztetnünk.

A **zajos adatok** (bizonyos attribútumok hibás értékkel rendelkeznek) előfordulásának is sok oka lehet. Előfordulhat, hogy az adatgyűjtéshez használt eszközök hibásak voltak. Az adatrögzítés során emberi vagy gépi hibák léphettek fel. Az adatok továbbítása során is történhetnek hibák. Az alkalmazott technológiá(k)nak is lehetnek korlátai, mint például a szinkronizált adatátvitelhez és kiolvasáshoz használt puffer véges mérete. Az adatkódok, illetve elnevezési konvenciók használatában fellépő inkonzisztenciák is eredményezhetnek hibás adatokat. Az ismétlődő, identikus sorok szintén adattisztítást igényelnek.

Az **adattisztító rutinok** működésük során kitöltik a hiányzó értékeket, a zajos adatokon simítást végeznek, azonosítják és/vagy eltávolítják a szélsőséges értékeket, valamint megszüntetik az inkonzisztenciákat. A „piszkos" adatok zavart okozhatnak az adatbányászat során, ami megbízhatatlan eredményre vezet. Bár a legtöbb bányászati rutin tartalmaz olyan eljárásokat, amelyek a hiányos vagy zajos adatok feldolgozására szolgálnak, ezek általában nem túl robusztus algoritmusok. Működésükben általában fontosabb az a szempont, hogy elkerüljék az adatok „túlillesztését" (overfitting) a modellezett függvényre. Emiatt az előfeldolgozás hasznos eleme az adattisztító rutinok alkalmazása. Az adatok tisztításának módszereit a 3.2. alfejezetben tárgyaljuk majd.

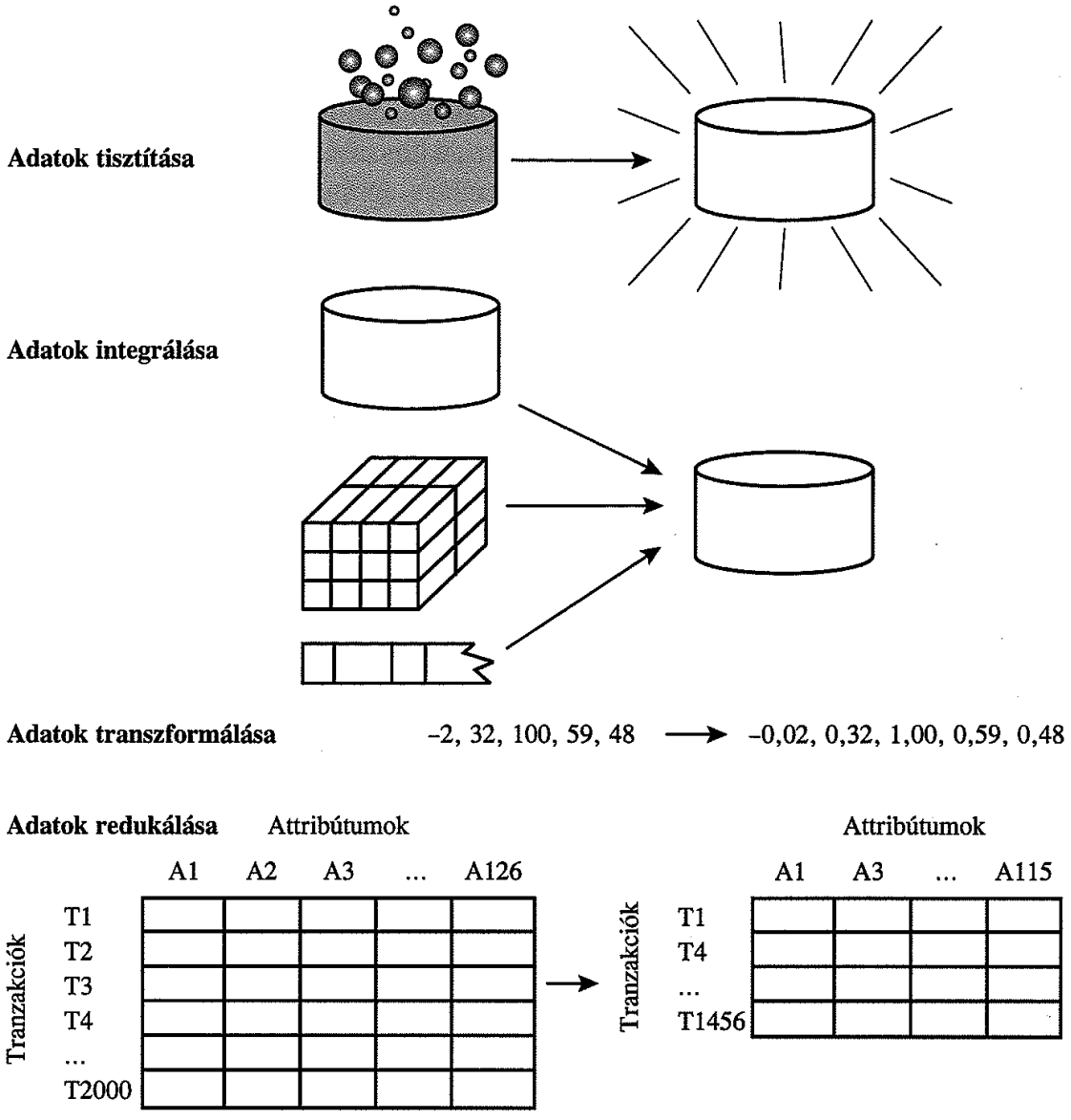
Visszatérve az *ElektroMindnél* végzett munkánkhoz, tegyük fel, hogy elemzésünkben különböző forrásokból származó adatokat szeretnénk felhasználni. Ennek során több adatbázist, adatkockát és fájlt kell egyesítenünk, vagyis **adatok integrálását** végezzük. Óvatosan kell azonban eljárnunk, mert előfordulhat, hogy ugyanazt az adott fogalmat reprezentáló attribútumok különféle adatbázisokban más-más nevet viselnek, ami inkonzisztenciát, illetve redundanciát okoz. Például lehet, hogy a vevő azonosítására szolgáló attribútum az egyik adattárban a *vevő_azonosító*, a másikban a *vásárló_id* nevet viseli. Elnevezésbeli inkonzisztenciák az attribútumok értékeinél is előfordulhatnak. Megeshet például, hogy ugyanazt a keresztnevet egy adatbázisban „Bill"-ként, egy másikban

„William"-ként, egy harmadikban pedig rövidítve csak „B."-ként rögzítették. Gyanítjuk továbbá azt is, hogy egyes attribútumok értékeire más attribútumokból következtethetünk (például éves jövedelem). A nagy mennyiségű redundáns adat lelassíthatja vagy megzavarhatja az tudásfeltárás folyamatát. Világos, hogy az adattisztításon túl további lépéseket kell tenni annak érdekében, hogy az adatok integrálása során fellépő redundanciákat lehetőség szerint elkerüljük. Az adattisztítás és az adatok integrálása az előfeldolgozás tipikus lépései az adattárházak összeállítása során. Az adatok integrálása közben fellépő redundanciák kiszűrése további adattisztítást tehet szükségessé.

Tételezzük fel, hogy a továbbiakban valamilyen távolságmérésen alapuló bányászati algoritmust kívánunk alkalmazni, mint amilyenek a neurális hálók, legközelebbi szomszéd szerinti osztályozás vagy a klaszterezés.[1] Ezek a módszerek jobb eredményt szolgáltatnak, ha az adatokat elemzés előtt *normalizáljuk*, vagyis egy jól meghatározott tartományra (például [0, 1] intervallum) skálázzuk át. A vásárlókra vonatkozó adataink tartalmazzák például az *életkor* és *éves_kereset* attribútumokat. Az *éves_kereset* természetesen sokkal többféle értéket felvehet, mint az *életkor*. Ezért ha az adatokat nem normalizáljuk, akkor az *éves_kereset* alapján végzett távolságmérések általában jóval nagyobb eltéréseket mutatnak majd, mint amiket az *életkor* alapján kapunk. Elemzésünk szempontjából hasznos lenne továbbá az egyes vásárlói rétegekre vonatkozó összesített adatok figyelembevétele is – ilyen információ azonban adattárházunk egyetlen előre kiszámított adatkockájában sem szerepel. Jól látható, hogy az adatokon végzett különféle transzformációk, például a normalizálás és az összesítés, szintén az előfeldolgozás hasznos lépései lehetnek, és hozzájárulhatnak az adatbányászat sikerességéhez. Az adatok integrálását és transzformálását a 3.3. alfejezetben tárgyaljuk.

Az adatok további vizsgálata ismét gondolkodóba ejt: *Az elemzésre kiválasztott adathalmaz hatalmas méretű – ez nyilvánvalóan lelassítja az adatbányászati munkát. Hogyan csökkenthetnénk az adathalmaz méretét anélkül, hogy az adatbányászat eredményességét kockáztatnánk?* Az adatok redukálása során a felhasznált adathalmaz olyan csökkentett reprezentációját kapjuk, amely méretében sokkal kisebb, ennek ellenére az elemzés során ugyanazokat (vagy majdnem ugyanazokat) az eredményeket szolgáltatja, mint az eredeti halmaz. Az adatok redukálására számos stratégiát ismerünk. Ezek közé tartozik az *adatok összesítése* (például egy adatkocka létrehozása), a *dimenziócsökkentés* (például az érdektelen attribútumok eltávolítása korrelációs analízissel), az *adattömörítés* (például minimumhossz vagy wavelet alapú kódolással) és a *számosságcsökkentés* (például az adatok „kicserélése" egy kisebb, alternatív reprezentációval, például a klaszterekkel vagy a parametrizált modellekkel). Az adatok mennyisége *általánosítás* révén is csökkenthető, amikor is az alacsony szintű fogalmakat, például a vásárlók lakhelyét megadó *város*t, magasabb szintű fogalmakkal helyettesítjük, ami ez esetben mondjuk *régió*, illetve *megye_vagy_tartomány* lehet. A fogalmak különféle absztrakciós szintekbe való szervezésére a *fogalmi hierarchiák* szolgálnak. Az adatok redukálásával a 3.4. alfejezetben foglalkozunk. Mivel a fogalmi hierarchiák igen hasznos eszközei az absztrakció több szintjén végzett adatbányászatnak, ezért külön alfejezetet szentelünk e fontos adatstruktúrák automatikus generálásával kapcsolatos ismereteknek. A 3.5. alfejezetben tárgyaljuk a

1 | A neurális hálókat és a legközelebbi szomszéd szerinti osztályozást a 7., a klaszterezést pedig a 8. fejezetben tárgyaljuk.



3.1. ábra | Az adatok előfeldolgozásának formái

fogalmi hierarchiák generálását, ami az adatok redukálásának egy diszkretizációra épülő típusa.

Az adatok előfeldolgozásának imént felsorolt lépéseit a 3.1. ábra foglalja össze. Fontos megemlítenünk, hogy a fenti osztályozás nem egymást kizáró kategóriákra épül. A redundáns adatok eltávolítása például egyrészt adattisztításnak, másrészt adatok redukálásának is tekinthető.

Összegzésként elmondhatjuk, hogy a valós feladatokban előforduló adatok rendszerint zajosak, hiányosak és inkonzisztensek. Az adatok előfeldolgozására használt módszerek segítségével az adatok minősége javítható, ezáltal fokozva a későbbi adatbányászat pontosságát és hatékonyságát. Az adatok előfeldolgozása a tudásfeltárási folyamat fontos lépése, hiszen megalapozott döntések csak megbízható adatok birtokában születhetnek. Az adatokban fellépő anomáliák felkutatására, korai kijavítására és az elemzendő adatmennyiség csökkentésére fordított idő busásan megtérülhet a döntéshozatal során.

3.2. | Adattisztítás

A való világban használt adatok gyakran hiányosak, zajosak és inkonzisztensek. Az *adattisztító rutinok* megkísérlik a hiányzó értékek kitöltését, a szélsőséges adatok eltávolításával együtt simítást alkalmaznak a zaj csökkentésére, és kiküszöbölik az adatokban fellépő inkonzisztenciákat. Ebben az alfejezetben az adattisztítás alapvető módszereivel ismerkedünk meg.

3.2.1. | Hiányzó értékek

Tegyük fel, hogy az *ElektroMind* eladási és vásárlói adatait kell elemeznünk. Észrevesszük, hogy számos sor jó néhány attribútuma, például a vásárlóra vonatkozó *kereset*, nem rendelkezik semmilyen értékkel. Hogyan adhatunk értéket egy ilyen attribútumnak? Vizsgáljuk meg a következő lehetőségeket:

(1) **Figyelmen kívül hagyjuk a sort** – Általában akkor járunk el így, amikor az osztálycímke hiányzik (feltéve, hogy a bányászat során osztályozást vagy leírásokat használunk). Ez a módszer nem túl hatékony, hacsak nincsen sok érték nélküli attribútum az adott sorban. Különösen gyengén teljesít ez a módszer akkor, ha a hiányzó értékekkel rendelkező attribútumok aránya nagy ingadozásokat mutat.

(2) **A hiányzó értékeket manuálisan töltjük ki** – Ez a megközelítés általában időigényes, és a sok hiányzó értékkel rendelkező, nagy adathalmazok esetében nem használható.

(3) **A hiányzó értéket egy globális konstanssal pótoljuk** – Minden hiányzó értéket ugyanazzal a konstanssal, például $-\infty$-nel vagy *ismeretlen* címkével helyettesítünk. Viszont ha a hiányzó értékek mindegyikét, mondjuk, *ismeretlen*re cseréljük, a bányászati program hibásan azt gondolhatja, hogy ezek érdekes fogalmat alkotnak, hiszen egy bizonyos érték mindegyikben közös – tudniillik maga az *ismeretlen* címke. Ezért, bár ez a módszer egyszerű, használata nem javasolt.

(4) **A hiányzó értéket az attribútum átlagértékével pótoljuk** – Tegyük fel például, hogy az *ElektroMind* vevőinek átlagos keresete 28 000 $. Ezt az értéket használjuk a *kereset* hiányzó értékeinek kitöltésére.

(5) **Az attribútumnak az adott sorral azonos osztályba tartozó minták alapján számított átlagértékét használjuk** – Ha például a vásárlókat a *hitel_kockázat* alapján soroljuk osztályokba, akkor a hiányzó értéket az adott sorral egyazon hitelkockázati kategóriába tartozó adatok átlagos *kereset* értékével helyettesítjük.

(6) **A legvalószínűbb értéket használjuk a hiányzó érték pótlására** – Ennek meghatározása történhet regresszióval, Bayes-formalizmusra alapuló függőségvizsgálati eszközökkel vagy döntési fa indukcióval. Például az adathalmazunk többi vásárlóra vonatkozó attribútumainak felhasználásával létrehozhatunk egy döntési fát a *kereset* hiányzó értékeinek becslésére. A döntési fákat részletesen a 7. fejezetben tárgyaljuk.

A 3–6. pontokban felsorolt módszerek torzítják az adatokat. Így lehet, hogy nem megfelelő értéket kapunk. A legutóbbi módszert mindazonáltal előszeretettel alkalmazzák, mert a többi módszerhez képest a rendelkezésre álló adatokból a legtöbb információt használja fel a hiányzó értékek megbecslésére. Mivel a *kereset* hiányzó értékének becslésekor a többi attribútum értékét is figyelembe veszi, nagyobb az esély arra, hogy a *kereset* és más attribútumok közti összefüggések megmaradnak.

3.2.2. | Zajos adatok

Mi a zaj? A zaj a mért értékek véletlenszerű hibája vagy ingadozása. Ha numerikus attribútumokkal foglalkozunk, mint például az *ár*, vajon hogyan „simíthatjuk" az adatokat a zaj kiszűrése érdekében? Az alábbiakban áttekintünk néhány adatsimítási technikát:

(1) **Kosarazás** – A kosarazási módszerek a rendezett adatok értékeit a szomszédos (a rendezett mintában körülöttük elhelyezkedő) értékekkel összevetve simítják. A rendezett értékeket valahány csoportba, ún. *kosarakba* osztják szét. Mivel a kosarazási módszerek az értékek közeli szomszédait vizsgálják, így *lokális* simítást végeznek. A 3.2. ábra néhány kosarazási módszert mutat be. Ebben a példában az *ár*ra vonatkozó adatokat először *nagyság szerint* rendezik, majd azonos, 3 mélységű kosarakba csoportosítják (vagyis minden kosár 3 értéket tartalmaz). A **kosárátlag alapján végzett simításkor** egy kosár minden értékét az adott kosár átlagértékével helyettesítenek. Például az 1. kosárba tartozó 4, 8 és 15 átlaga 9, így az eredeti értékeket ebben a kosárban 9-cel helyettesítik. Hasonlóan végezhető a **simítás a kosarak mediánértéke alapján**. A **kosarak határai szerint végzett simítás** esetén egy adott kosár minimális és maximális értékét a kosár *határainak* nevezzük. A kosár értékeit ekkor a legközelebbi határ értékével helyettesítjük. Általában véve, minél szélesebb a kosár (azaz minél nagyobb a határok közti különbség), annál nagyobb a simítás hatása. Egy má-

Az *árra* vonatkozó rendezett adatok (dollárban): 4, 8, 15, 21, 21, 24, 25, 28, 34

Particionálás azonos mélységű kosarakba:

1. kosár: 4, 8, 15
2. kosár: 21, 21, 24
3. kosár: 25, 28, 34

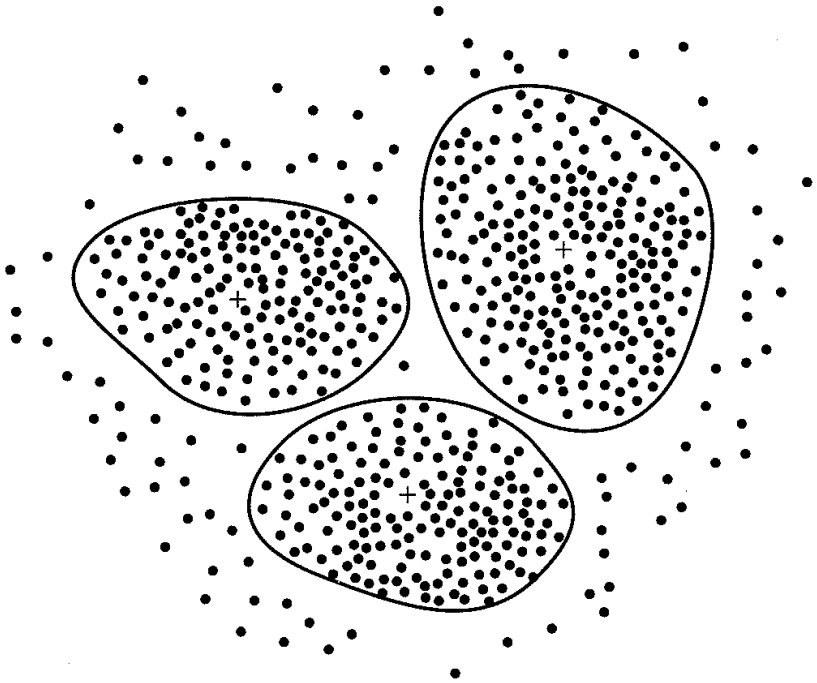
Simítás a kosárátlagok szerint:

1. kosár: 9, 9, 9
2. kosár: 22, 22, 22
3. kosár: 29, 29, 29

Simítás a kosarak határai szerint:

1. kosár: 4, 4, 15
2. kosár: 21, 21, 24
3. kosár: 25, 25, 34

3.2. ábra | Az adatok simítására szolgáló kosarazási módszerek



3.3. ábra | A szélsőséges értékek klaszterelemzés segítségével ismerhetők fel

sik lehetőség, hogy a kosarakat azonos szélességűnek választjuk, amikor is a kosarak határai egyforma nagyságú intervallumokat fognak át. A kosarazást mindezeken túl diszkretizációs módszerként is alkalmazzák. A kosarazási módszerek témájára még visszatérünk a 3.5. alfejezetben és a 6. fejezetben.

(2) **Klaszterezés** – A szélsőséges értékek kiszűrésére használható; a hasonló értékeket ilyenkor *klasztereknek* nevezett csoportokra osztjuk fel. Azokat az értékeket tekintjük szélsőségesnek, amelyek egyetlen klaszterhez sem tartoznak („kilógnak" a többi adatcsoport közül, lásd a 3.3. ábrát). A klaszterezés témájával a 8. fejezet foglalkozik.

(3) **Kombinált számítógépes/emberi vizsgálat** – Szintén használható a szélsőséges értékek felismerésére. Egy alkalmazás esetében például egy információelméleti mérték segítségével azonosították egy kézírásos karakter-adatbázis szélsőséges mintáit. A mérték nagysága arra utalt, mennyire tért el az adatok alapján „jósolt" karakter a konkrét, ismert karaktertől. A szélsőséges minták hordozhatnak értékes információt (azonosíthatnak például fontos kivételeket, mint mondjuk a „0" vagy „7" karakterek szokásostól eltérő verzióit), de lehetnek semmitmondóak is (például hibásan azonosított karakterek). Azokat a mintákat, amelyeknél az eltérés meghaladott egy adott küszöbértéket, egy listán rögzítették. Ezután egy ember átvizsgálta a listát, és kiszűrte a ténylegesen irreleváns mintákat. Ez a módszer sokkal gyorsabb, mintha a teljes adatbázist manuálisan kellene átvizsgálni. Az ily módon azonosított irreleváns minták a további adatbányászatból kizárhatók.

(4) **Regresszió** – Az adatok simítása úgy is elvégezhető, hogy azokra egy függvényt illesztünk, például regresszió segítségével. A *lineáris regresszió* esetében két változó értékeire „legjobban" illeszkedő egyenest keresünk, amivel az egyik változó alapján a másik értéke jósolható. A *többdimenziós lineáris regresszió* az előbbinek olyan kiterjesztése, amelyben kettőnél több változót vizsgálunk, és az adatokra egy többdi-

menziós felületet (síkot) illesztünk. A regressziós eljárással az adatokra illesztett matematikai egyenlet segít a zaj eliminálásában. A regresszióval foglalkozunk még a 3.4.4. alfejezetben, valamint a 7. fejezetben.

Számos, az adatok simítására szolgáló módszer egyúttal diszkretizációs adatredukálási módszernek is tekinthető. A fent említett kosarazási módszerek például csökkentik az attribútumok által felvett értékek számát. Ez egyfajta adatredukálásként működik a logikai alapú adatbányászati módszerek esetében, mint amilyenek a döntési fákat használó algoritmusok is, ahol újra és újra összehasonlítják a rendezett adatok értékeit. A fogalmi hierarchiák szintén olyan diszkretizációs módszerek, amelyek adatsimításra is használhatók. Egy, az *ár* attribútumhoz tartozó fogalmi hierarchia például a valós *ár*-értékeket *olcsó*, *elfogadható* és *drága* értékekre képezheti le, csökkentve ezzel a bányászati eljárás során kezelendő értékek számát. Az adatok diszkretizációjával a 3.5. alfejezetben foglalkozunk majd. Néhány osztályozási eljárás, például neurális hálók, beépített adatsimítási mechanizmusokkal rendelkeznek. Az osztályozás témakörét a 7. fejezetben tárgyaljuk.

3.2.3. | Inkonzisztens adatok

A tranzakciók során rögzített adatok között inkonzisztenciák is lehetnek. Bizonyos inkonzisztenciák külső források alapján kézzel kijavíthatók. Példának okáért az adatbevitel során elkövetett hibák papíron rögzített adatok alapján korrigálhatók. Ezt ötvözhetjük olyan rutinok használatával, amelyek a különféle kódok inkonzisztens használatát gátolják. Az ismeretanyagot rendszerező, szervező eszközök szintén segíthetnek az érvénytelen adatok kiszűrésében. Az attribútumok közt fennálló ismert függvénykapcsolatok például jelezhetik a hibás értékeket.

Inkonzisztenciák az adatok integrálása során is felléphetnek, amikor egy adott attribútum különböző adatbázisokban különböző néven szerepel. Az adatokban találhatunk redundanciákat is. Az adatok integrálását és a redundáns adatok eltávolítását a következő alfejezetben tárgyaljuk.

3.3. | Adatok integrálása és transzformálása

Az adatbányászat gyakran igényli az adatok integrálását, vagyis az adatok különböző adattárakból való egyesítését. Szükség lehet arra is, hogy az adatokat a bányászat céljainak megfelelően transzformáljuk. Ebben az alfejezetben adatok integrálásával és transzformálásával foglalkozunk.

3.3.1. | Adatok integrálása

Adataink elemzése során valószínűleg *adatok integrálására* is sor kerül, amikor is a különböző forrásból származó adatokat egyetlen koherens adattárban egyesítjük, mint az adattárházak esetében. Ezen források között lehetnek különféle adatbázisok, adatkockák vagy egyszerű fájlok.

Az adatok integrálása közben számos problémával kell megküzdenünk. Már a *sémák integrálása* is igen nehéznek bizonyulhat. *Hogyan feleltethetők meg egymásnak az ekvivalens valódi egyedek a különböző forrásokban?* Ezt a kérdést **egyedazonosítási problémának** nevezzük. Például hogyan lehet biztos egy adatelemző vagy számítógép abban, hogy egy adatbázisban a *vevő_szám* ugyanarra a dologra utal, mint egy másikban a *vásárló_ID*? Az adatbázisok és adattárházak tipikusan metaadatokat, vagyis az adatokra vonatkozó adatokat is tartalmaznak, amely metaadatok segíthetnek a hibák elkerülésében a sémák integrálása során.

A *redundancia* szintén fontos kérdés. Egy attribútum redundáns akkor, ha egy másik táblából kiszámítható, ilyen például az *éves_bevétel*. Az attribútumok vagy dimenziók elnevezésében fellépő inkonzisztenciák szintén redundáns adathalmazt eredményezhetnek.

Bizonyos típusú redundanciák **korrelációanalízis** segítségével felderíthetők. Ha például adott két attribútum, korrelációanalízissel mérhető, hogy valamelyik attribútumból az ismert adatok alapján milyen valószínűséggel következtethetünk a másik értékére. Az A és B attribútumok (mennyiségek) korrelációját a következő képlettel számíthatjuk ki:

$$r_{A,B} = \frac{\Sigma(A - \bar{A})(B - \bar{B})}{(n - 1)\sigma_A \sigma_B}, \tag{3.1}$$

ahol n a sorok száma, továbbá $\bar{A}$ és $\bar{B}$, illetve σ_A és σ_B az A és B mennyiségek átlagát, illetve szórását jelöi.[2] Ha a fenti egyenletből kapott mennyiség pozitív, akkor A és B pozitívan korrelálnak, ami azt jelenti, hogy A növekedésével B is növekszik. Minél nagyobb ez az érték, annál inkább következik egyik mennyiség a másikból. Ily módon a nagy korreláció azt jelenti, hogy A (vagy B) redundáns mennyiségként eltávolítható. Amennyiben az eredményül kapott érték 0, akkor A és B függetlenek,[3] és nincs köztük korreláció. Ha az eredményül kapott szám negatív, akkor A és B negatívan korrelálnak, vagyis az egyik mennyiség növekedésével a másik csökken. A (3.1) egyenlet lehetőséget nyújt a fenti *vevő_szám* és *vásárló_ID* attribútumok közti korreláció kimutatására. A korrelációs analízisre visszatérünk még a 6.5.2. alfejezetben.

Az attribútumok közti redundanciák felderítésén túl az ismétlődéseket a sorok szintjén is ki kell szűrni (például amikor egy bizonyos bevitt adathoz két vagy több azonos sor tartozik).

Az adatok integrálásának harmadik fontos problémája az *ellentmondó adatértékek felderítése és kijavítása*. Például az ugyanazon valós egyedhez tartozó attribútumok értékei a különböző forrásokban eltérhetnek egymástól. Ennek oka lehet a különbözőképpen választott reprezentáció, skálázás vagy kódolás. Lehetséges például, hogy a *súly* attribú-

2 | Az A mennyiség átlagát, illetve szórását a következőképpen definiáljuk:

$$\bar{A} = \frac{\Sigma A}{n},$$

$$\sigma_A = \sqrt{\frac{\Sigma(A - \bar{A})^2}{n - 1}},$$

3 | Ez így nem precíz; a függetlenségből következik a korrelálatlanság, de a fordított implikáció nem igaz. (*A fordító megjegyzése*)

tum értékeit az egyik rendszerben metrikus, egy másikban pedig angolszász mértékegységekben tárolják. Az egyes szállodák *árai* nemcsak különböző valutanemben értendők, de különböző szolgáltatásokat (például reggeli) és adótartalmat tükrözhetnek. Az ilyesfajta szemantikai sokrétűség nagy kihívást jelent az adatok integrációja során.

A több forrásból származó adatok figyelmes integrálásával mérsékelhető az előállított adathalmazban a redundancia, illetve az inkonzisztencia, ily módon javítva az adatbányászat pontosságát és sebességét.

3.3.2. | Adatok transzformálása

Az *adatok transzformálása* során a kiindulási adatokat a bányászat céljainak megfelelő alakra hozzuk. Ennek leggyakoribb eszközei a következők:

- **Simítás**, amelynek során az adatokból eltávolítjuk a zajt. Ennek módszerei közé tartozik a kosarazás, a klaszterezés és a regresszió.
- **Összevonás**, amikor az adatokon összegzési vagy összevonási műveleteket hajtunk végre. Így például a napi eladási adatok összevonhatók havi és éves adatokká. Ezt a lépést tipikusan akkor használják, amikor a részletesség több szintjén végzett elemzéshez készítenek adatkockákat.
- Az **adatok általánosítása**, amikor az alacsony szintű vagy „primitív" (nyers) adatokat fogalmi hierarchiák segítségével magas szintű fogalmakra cserélik. Például az olyan, kategória jellegű attribútumokat, mint az *utca*, olyan magasabb szintű, általánosabb attribútumokkal helyettesítik, mint a *város* vagy *ország*. Hasonlóképp, a numerikus attribútumok értékei, mint az *életkor*, magasabb szintű attribútumokra képezhetők le, úgymint *fiatal*, *középkorú* vagy *idős*.
- **Normalizálás**, amikor az attribútumok értékeit úgy skálázzák át, hogy azok egy kis, előre megadott tartományba essenek, mint például a [–1, 1] vagy [0, 1] intervallum.
- **Attribútumok konstrukciója**, amikor a bányászat megkönnyítése érdekében a meglévő attribútumokból újakat hoznak létre.

A simítás az adattisztítás egy formája, ezzel már a 3.2.2. alfejezetben megismerkedtünk. Az összevonás és általánosítás adatok redukálására is szolgálhat, ezeket a 3.4. és 3.5. alfejezetben tárgyaljuk. Ebben az alfejezetben a normalizálással és attribútumok konstrukciójával foglalkozunk.

Egy attribútum normalizálása során annak értékeit úgy skálázzuk át, hogy azok egy előre megadott kis tartományba, például a [0, 1] intervallumba essenek. A normalizálás különösen hasznos olyan osztályozási algoritmusok esetében, amelyek neurális hálókat vagy távolságmérést használnak, például a legközelebbi szomszéd szerinti osztályozás vagy a klaszterezés. Ha a neurális hálók visszaterjesztő algoritmusát használjuk klasszifikációs bányászat céljára (lásd 7. fejezet), akkor érdemes a tanulásra használt minták minden attribútumának inputértékeit normalizálni, mert ez jelentősen rövidítheti a tanulási fázist. A távolság alapú módszerek esetében a normalizálás segítségével elkerülhető, hogy az eredetileg nagy értékkészlettel rendelkező attribútumok „elnyomják" az eredetileg kis értékkészletű (például bináris) attribútumokat. Az adatok normalizálására számos

módszer létezik. Mi hármat vizsgálunk meg ezek közül: a *min-max normalizálást*, a *standardizálást* és a *decimális skálázású normalizálást*.

A min-max normalizálás lineárisan transzformálja az adatokat. Jelölje egy A attribútum minimális és maximális értékét $\min_A$ és $\max_A$. A min-max normalizálás az A mennyiség egy v értékéhez az új [$new\_\min_A$, $new\_\max_A$] értékkészletből a következőképp rendeli hozzá az új v' értéket:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A. \tag{3.2}$$

A min-max normalizálás megőrzi az eredeti adatértékek közti összefüggéseket. Az eljárás „határon kívüli érték" hibát ad, amennyiben egy későbbi inputérték az A mennyiség eredeti értékkészletén kívül esik.

3.1. példa | Tegyük fel, hogy a *kereset* attribútum minimális, illetve maximális értéke 12 000, illetve 98 000 $. Szeretnénk a *kereset* értékét a [0, 1] intervallumra képezni. A min-max normalizálással egy 73 600 $ érték transzformálódik $\frac{73\,600 - 12\,000}{98\,000 - 12\,000}(1-0)+0 =$ $= 0{,}716$-dé. ■

A **standardizálás** (vagy **zéró-átlag normalizálás**) során egy A attribútum értékeit annak átlaga és szórása alapján normalizáljuk. Az A mennyiség egy v értékéhez a normalizált v' értéket a következőképpen kapjuk:

$$v' = \frac{v - \bar{A}}{\sigma_A}, \tag{3.3}$$

ahol $\bar{A}$ és σ_A az A attribútum átlagát, illetve szórását jelöli. Ez a normalizálási módszer akkor hasznos, amikor az A attribútum minimális, illetve maximális értéke nem ismert, vagy ha szélsőséges (a tulajdonképpeni minimumnál kisebb, illetve maximumnál nagyobb) értékek eltorzítják a min-max normalizálást.

3.2. példa | Tegyük fel, hogy a *kereset* attribútum átlagértéke és szórása 54 000, illetve 16 000 $. Ekkor a standardizálás a *kereset* 73 600 $ értékét $\frac{73\,600 - 54\,000}{16\,000} = 1{,}225$-dé transzformálja. ■

A **decimális skálázású normalizálás** úgy működik, hogy az A attribútum értékeiben áthelyezi a tizedesvesszőt. Az, hogy hány helyi értékkel tolódik el a tizedesvessző, az A attribútum legnagyobb abszolút értékétől függ. Az A mennyiség egy v értékének normalizált v' megfelelője:

$$v' = \frac{v}{10^j}, \tag{3.4}$$

ahol j a legkisebb olyan egész szám, amelyre $Max(|v'|) < 1$.

3.3. példa | Tegyük fel, hogy az *A* mennyiség rögzített értékei –986-tól 917-ig terjednek. Ekkor *A* maximális abszolút értéke 986. A decimális skálázású normalizálás során ezért minden értéket 1000-rel osztunk (vagyis $j = 3$), így a –986 normalizált értéke –0,986 lesz. ■

Jegyezzük meg, hogy a normalizálás jócskán megváltoztathatja az adatokat, főként a két utóbbi módszer esetében. Fontos továbbá, hogy a normalizációs paramétereket (például a standardizálás esetében az átlagot és a szórást) is minden esetben rögzítsük, hogy az esetleges későbbi adatokat az eddigiekkel összhangban normalizálhassuk.

Attribútumok konstrukciója során a rendelkezésre álló attribútumokból újabbakat hozunk létre, s ezekkel kibővítjük az eddigieket a pontosság javítása és a sokdimenziós adatstruktúrák jobb feltérképezhetősége érdekében. Például a *szélesség* és *magasság* attribútumai alapján kiegészíthetjük adatainkat a *terület* attribútumával. Az új attribútumok konstrukciója segíthet a *töredezettség problémájának* megoldásában, ami olyankor léphet fel, amikor döntésifa-algoritmusokat használnak osztályozásra – ebben az esetben egy attribútumot újra és újra megvizsgálunk a megkonstruált döntési fa egy ága mentén.[4] Az attribútumok konstrukciójára használt operátorok között találjuk a bináris attribútumokra vonatkozó logikai **ÉS** műveletet, valamint a nominális attribútumokra vonatkozó **szorzás** műveletet. Az attribútumok kombinációjából létrehozott új attribútumok olyan hiányzó információkat tárhatnak fel az adatok közötti összefüggésekről, amelyek hasznosak lehetnek a tudásfeltárási folyamat során.

3.4. | Adatok redukálása

Képzeljük el, hogy az *ElektroMind* adattárházából adatokat választottunk ki elemzés céljából. Az adathalmaz minden valószínűség szerint hatalmas méretű lesz! Az összetett adatelemzés és a hatalmas mennyiségű adaton végzett bányászat igen hosszú időt vesz igénybe, praktikus okokból lehetetlenné téve az ilyen analízist.

Az **adatok redukálásának** módszereivel az adathalmaz olyan csökkentett reprezentációját kapjuk, amely méretében sokkal kisebb, mégis igen jól megőrzi az eredeti adatok integritását. Más szóval, a redukált adathalmazon végzett bányászat várhatóan sokkal hatékonyabb (kevesebb ideig tart), mégis az eredetivel azonos, vagy majdnem azonos eredményt szolgáltat.

Az adatok redukálásának főbb stratégiái többek között:

(1) **Összevonás adatkockába**, amikor az adat-összevonási műveleteket adatkocka létrehozásakor alkalmazzuk.
(2) **Dimenziócsökkentés**, amikor az irreleváns vagy csak kevéssé releváns, valamint redundáns adatokat kiszűrjük.
(3) **Adattömörítés**, amikor kódolási mechanizmusokat használunk az adathalmaz méretének csökkentésére.

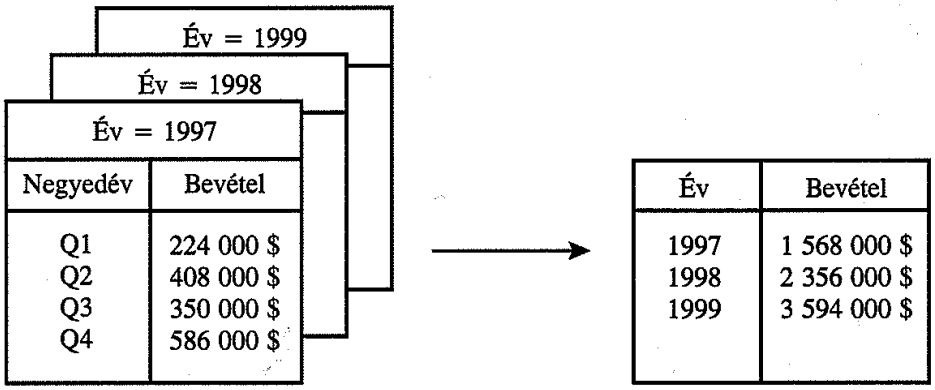
4 | A döntési fákkal részletesen a 7. fejezetben foglalkozunk.

(4) **Számosságcsökkentés**, amikor az adatokat kisebb méretű reprezentációkkal helyettesítjük vagy közelítjük. Ilyenek például a parametrizált modellek (amelyeknél csak a paraméterek értékét kell tárolni a tényleges adatok helyett) vagy egyes nem paraméteres módszerek, mint a klaszterezés, mintavételezés és a hisztogramok használata.
(5) **Diszkretizáció és fogalmi hierarchiák generálása**, amikor az attribútumok konkrét értékeit értékkészletükkel (intervallumokkal) vagy magasabb szintű fogalmakkal helyettesítjük. A fogalmi hierarchiák lehetővé teszik az adatoknak egyszerre több absztrakciós szinten végzett bányászatát, és általában véve is igen hatékony adatbányászati eszközt képeznek. A fogalmi hierarchiák témája tehát kitüntetett figyelmet érdemel, ezért később, a 3.5. alfejezetben kizárólag ezzel foglalkozunk majd.

A vázolt 1–4. módszerek képezik jelen alfejezet tárgyát. Fontos szempont, hogy az adatok redukálására fordított idő ne haladja meg a csökkentett méretű adathalmazon folytatott bányászat által elért időbeli nyereséget.

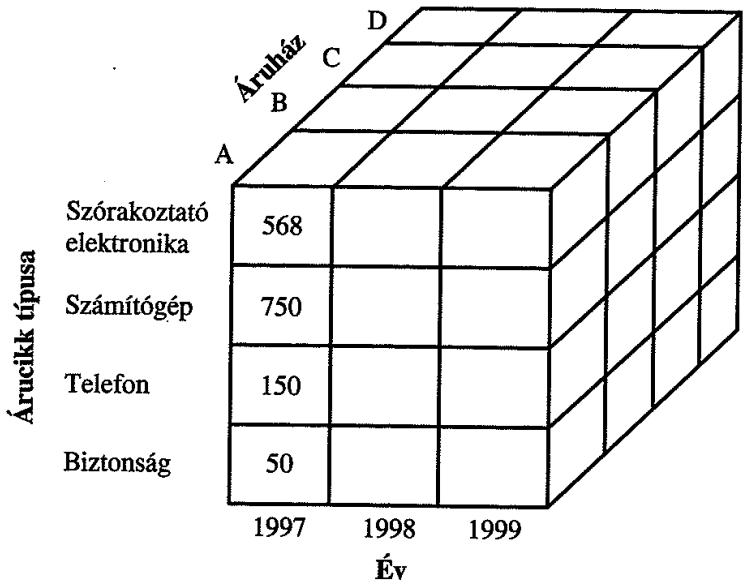
3.4.1. | Összevonás adatkockába

Tegyük fel, hogy összegyűjtöttük az elemezni kívánt adatokat. Ezek az adatok az *ElektroMind* negyedéves bevételi mutatóit tartalmazzák az 1997–1999. évekre. Minket azonban most inkább az éves szinten összesített bevételek érdekelnek. Ezért az adatokat oly módon összesítjük, hogy az egyes évekhez tartozó negyedévek bevétel értékeit összeadjuk. Ezt az összevonási műveletet mutatja be a 3.4. ábra. Az előálló adathalmaz kisebb méretű, de az elemzés szempontjából fontos információ nem veszett el.



3.4. ábra | Az *ElektroMind* egy adott áruházának eladásból származó bevételei az 1997–1999. évekre. A bal oldalon a negyedéves bevételek láthatók. A jobb oldalon az éves szinten összesített adatok szerepelnek

Az adatkockákkal a 2. fejezetben foglalkoztunk. A teljesség kedvéért az ott tárgyaltak egy részét röviden újra áttekintjük. Az adatkockák sokdimenziós összevont információt tartalmaznak. Például a 3.5. ábrán látható adatkocka az *ElektroMind* áruházakra bontott, valamennyi árucikkre vonatkozó éves eladási adatainak sokdimenziós elemzésére készült. Minden adatcella egy összesített értéket tartalmaz, ami a sokdimenziós adattér egy pontjának felel meg. Minden attribútum számára készíthetünk fogalmi hierarchiát, ami



3.5. ábra | Adatkocka az *ElektroMind* eladási adatairól

lehetővé teszi az adatok egyszerre több absztrakciós szinten folytatott elemzését. Például az *áruház*hoz tartozó hierarchia segítségével az áruházakat címük alapján régiók szerint csoportosíthatjuk. Az adatkockák az előre kiszámított, összegzett adatokhoz való gyors hozzáférést teszik lehetővé, elősegítve ezzel az on-line elemzéseket és az adatbányászatot.

Az absztrakció legalacsonyabb szintjén létrehozott kockát *alapkockának* nevezzük. A legmagasabb absztrakciós szintnek megfelelő kocka a *csúcskocka*. A 3.5. ábrán látható adatkockához tartozó csúcskocka egyetlen összesített adatot tartalmaz – a mindhárom évre, az összes árucikkre és valamennyi áruházra vonatkozó együttes *eladások* értékét. A különféle absztrakciós szintekhez tartozó adatkockákat egyszerűen csak részkockáknak nevezik, úgyhogy az adatkocka kifejezés gyakran *részkockák hálózatát* vagy *részkockahálóját* jelöli. Az egyre magasabb absztrakciós szinteken az adathalmazok mérete egyre kisebb.

Az alapkocka rendesen egy bizonyos, számunkra érdekes egyedre vonatkozó információt tartalmaz, mit például az *eladott_darabszám* vagy *vevő*. Más szóval, a legalacsonyabb szintnek az elemzés szempontjából használhatónak, illetőleg hasznosnak kell lennie. Mivel az adatkockák az előre kiszámított, összegzett adatokhoz nyújtanak gyors hozzáférést, az összesített információkra vonatkozó lekérdezések esetén lehetőség szerint ezeket kell használnunk. Amikor ilyen OLAP-lekérdezésekre vagy adatbányászati kérésre válaszolunk, mindig az adott feladatnak megfelelő *legkisebb* rendelkezésre álló részkockát kell használni. Ezzel a kérdéssel a 2. fejezet is foglalkozik.

3.4.2. | Dimenzócsökkentés

Az elemzésre használt adathalmazok attribútumok százait tartalmazhatják, amelyek között számos olyan is lehet, amely az adatbányászati feladat szempontjából érdektelen vagy redundáns. Például ha az a feladat, hogy osztályozzuk vevőinket abból a szempontból, hogy feltehetőleg vásárolnak-e az *ElektroMindnél* egy népszerű új CD-ből, amennyiben azt meghirdetjük, a vevő telefonszáma és a hasonló attribútumok valószínűleg irrele-

vánsak, míg ellenben az *életkor* vagy a *zenei_ízlés* fontos lehet. Bár lehet, hogy egy szakértő ki tudja választani a fontos attribútumok némelyikét, ez nehéz és időigényes feladat, főleg akkor, ha az adatok viselkedése nem jól ismert (ezért van szükség elemzésre!). Fontos attribútumok kihagyása vagy érdektelenek megtartása káros lehet, és megzavarhatja az alkalmazott bányászati algoritmust. Mindez pedig rossz minőségű mintákat eredményezhet. Az irreleváns vagy redundáns attribútumokkal járó adattömeg továbbá le is lassíthatja a bányászatot.

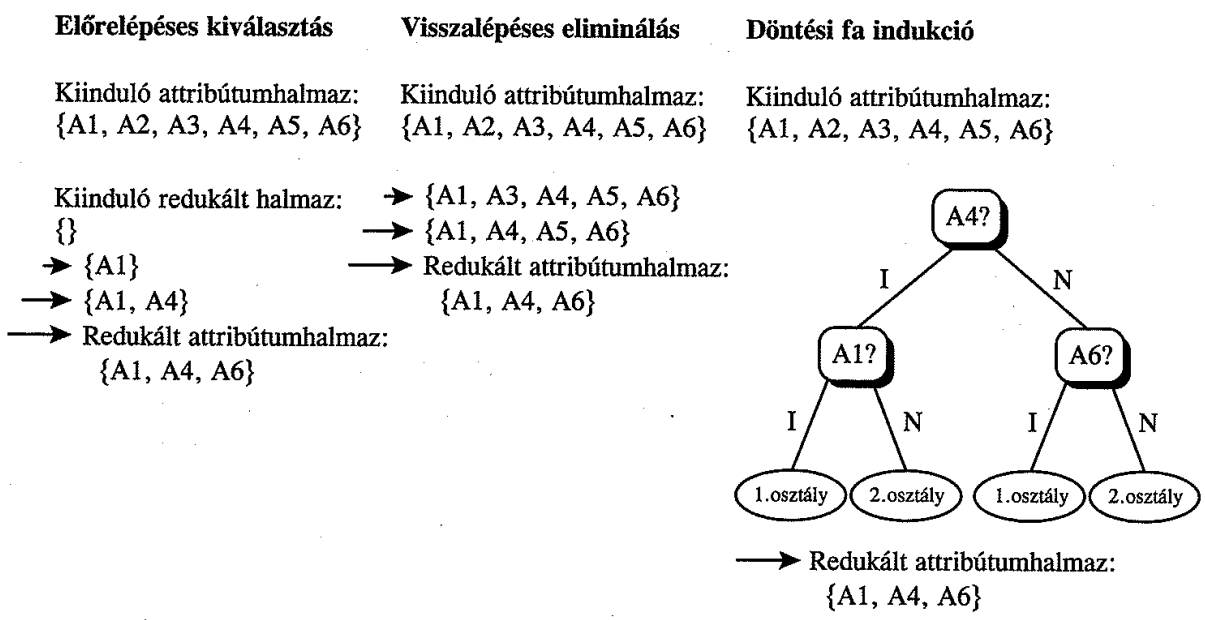
A *dimenziócsökkentés* úgy redukálja az adathalmaz méretét, hogy az ilyen attribútumokat (vagy dimenziókat) eltávolítja. Erre tipikusan az attribútumok egy részhalmazának kiválasztására szolgáló módszereket használnak. Az **attribútumok részhalmazának kiválasztása** azzal a céllal történik, hogy találjunk egy olyan minimális attribútumhalmazt, amelyben az adatosztályok valószínűségi eloszlása a lehető legjobban megközelíti az összes attribútum alapján kapható eredeti eloszlást. A redukált attribútumhalmazon történő adatbányászatnak van egy további előnye is: csökkenti a megtalált mintákban előforduló attribútumok számát, ami megkönnyíti a minták értelmezését.

Hogyan találhatunk az attribútumok között egy „jó" részhalmazt? Ha d számú attribútumunk van, akkor a lehetséges részhalmazok száma 2^d. Az attribútumok optimális részhalmazának megtalálását célzó kimerítő keresés megengedhetetlenül idő-, illetve erőforrás-igényes lehet, különösen, ha d és az adatosztályok száma nagy. Ezért az attribútumok részhalmazának kiválasztására rendszerint olyan heurisztikus módszereket alkalmaznak, amelyek egy csökkentett keresési téren dolgoznak. Ezek a módszerek tipikusan *mohók*, amennyiben az attribútumok terének átvizsgálása során mindig azt a lépést választják, amely *az adott pillanatban a legjobbnak tűnik.* Azzal a stratégiával dolgoznak, amely mindig a lokálisan optimális választást részesíti előnyben annak reményében, hogy ez globálisan optimális megoldásra vezet. Ezek a módszerek a gyakorlatban hatékonynak bizonyulnak, és gyakran igen jól megközelítik az optimális megoldást.

A „legjobb" (illetve „legrosszabb") attribútumokat jellemzően a statisztikus szignifikanciát mérő tesztek alapján határozzák meg, amelyek feltételezik, hogy az attribútumok függetlenek egymástól. Sok más attribútumkiértékelő eszköz is használatos, mint például az *információnyereség* mérése, amit az osztályozásra szolgáló döntési fák megalkotásakor alkalmaznak.[5] Az attribútumok részhalmazának kiválasztására szolgáló alapvető heurisztikus módszerek többek között a következők (némelyikük a 3.6. ábrán látható):

(1) **Előrelépéses kiválasztás** – Az eljárás üres attribútumhalmazzal indul. Meghatározza az eredeti attribútumok közül a legjobbat, és hozzáveszi a kiválasztandó részhalmazhoz. Minden rákövetkező lépés (iteráció) során a megmaradt attribútumok közül a legjobbal bővíti a részhalmazt.
(2) **Visszalépéses eliminálás** – Az eljárás a teljes attribútumhalmazból indul ki. Minden lépés során a megmaradt halmaz legrosszabb elemét dobja ki.
(3) **Az előzőek kombinációja** – Az előrelépéses kiválasztás és a visszalépéses elimináció oly módon kombinálható, hogy a meglévő legrosszabb attribútumot eldobjuk, a maradék attribútumok közül a legjobbat pedig hozzávesszük a részhalmazhoz.

5 | Az információnyereség mérésével részletesen az 5.3.2. és a 7.3.1. alfejezetben foglalkozunk. A fogalom rövid bemutatására 3.5.1. alfejezetben az attribútumok diszkretizációjával kapcsolatban kerítünk sort.



3.6. ábra | Az attribútumok részhalmazának kiválasztására szolgáló mohó (heurisztikus) módszerek

A fenti három módszer megállási kritériumai különfélék lehetnek. Az eljárás használhatja valamilyen mérték küszöbértékét a megállás időpontjának meghatározásához.

- **Döntési fa indukció** – A döntésifa-algoritmusokat, például az ID3-at és a C4.5-öt, eredetileg osztályozási célokra dolgozták ki. A döntési fát használó módszer egy folyamatábrára emlékeztető diagramot (fagráfot) készít, amelynek minden belső (azaz nem vég-) pontja egy, valamely attribútumon végrehajtott tesztet jelent; minden kiinduló ág a teszt egy kimenetelének felel meg, és végül minden külső (vég)pont egy osztály predikcióját jelenti. Minden csúcspontban az algoritmus a „legjobb" attribútumot használja az adatok különböző osztályokba történő besorolásához.

  Amikor a döntési fákat attribútumok részhalmazának kiválasztására használják, a rendelkezésre álló adatokból először egy fát hoznak létre. Irrelevánsnak tekintjük azokat az attribútumokat, amelyek a fában nem jelennek meg. A fában megjelenő attribútumok alkotják a csökkentett attribútumhalmazt. Az attribútumok kiválasztásának ezt a módszerét részletesebben szemügyre vesszük az 5. fejezetben.

Azt a módszert, amikor a bányászati feladat osztályozás jellegű, és magát a bányászati algoritmust használjuk az attribútumok részhalmazának meghatározására, ún. **becsomagoló (wrapper) stratégiának** nevezzük; egyéb esetekben ún. **szűrő (filter) stratégiáról** beszélünk. A becsomagoló stratégiák általában pontosabbak, mivel az attribútumok eltávolítása közben folyamatosan optimalizálják az algoritmus által a kiértékeléshez használt mértéket. Másrészt viszont több számítás elvégzésére van szükségük, mint a szűrő stratégiáknak.

3.4.3. | Adatok tömörítése

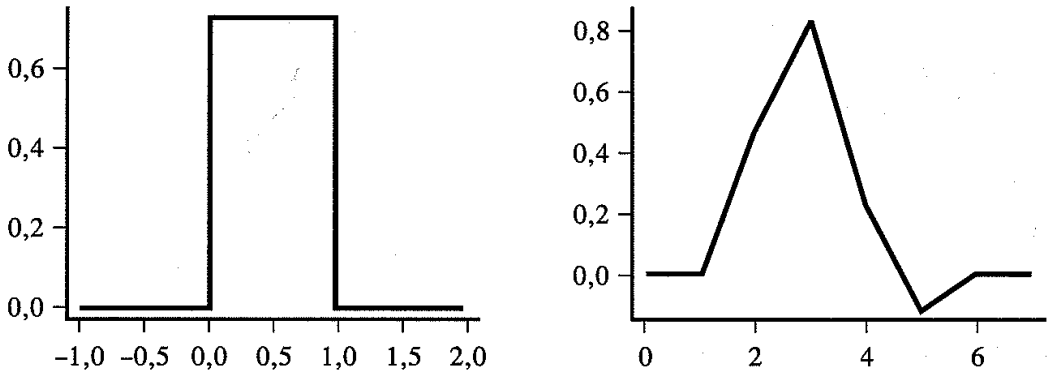
Az *adatok tömörítése* során azokat kódolási vagy transzformációs műveleteknek vetik alá annak érdekében, hogy az eredeti adatoknak egy csökkentett vagy „tömörített" reprezentációját állítsák elő. Amennyiben az eredeti adatok a tömörített adatokból bármiféle információveszteség nélkül *visszaállíthatók*, ún. **veszteségmentes** tömörítési eljárásról beszélünk. Ha azonban az eredeti adatoknak csak valamilyen közelítését kapjuk vissza, akkor a tömörítést **veszteségesnek** nevezzük. Karakterláncok tömörítésére számos kifinomult algoritmus létezik. Bár ezek jobbára veszteségmentesek, csak az adatok korlátozott manipulációját teszik lehetővé. Mi most inkább két népszerű és hatékony veszteséges tömörítési eljárásra koncentrálunk: a *wavelet-transzformációkra* és a *főkomponens-analízisre*.

Wavelet-transzformációk

A **diszkrét wavelet-transzformáció (DWT)** egy lineáris jelfeldolgozási módszer, amely egy D adatvektort az ún. **wavelet-együtthatókat** tartalmazó D' vektorba visz át. A két vektor hossza megegyezik.

Miként lehet alkalmas ez a technika az adatok redukálására, ha a wavelet-transzformáció után kapott adatvektor az eredetivel egyező hosszúságú? A módszer hasznossága abban rejlik, hogy a wavelet-transzformált adatvektorok vége „levágható". Az adatok tömörített approximációjához elegendő néhány domináns wavelet-koefficiens tárolása. Dönthetünk például úgy, hogy csak egy bizonyos küszöbérték felett tartjuk meg a koefficienseket, s a többi együtthatót nullával tesszük egyenlővé. Ily módon az adatoknak igen takarékos reprezentációját kapjuk, amit az alkalmazott műveletek során is kihasználhatunk, hiszen a wavelet-térben a számításokat igen gyorsan tudjuk elvégezni. A módszer a zajt is csökkenti anélkül, hogy az adatok jellegzetes vonásait „túlsimítaná", ami adattisztítás céljára is alkalmassá teszi. Adott koefficiensekből az eredeti adatok egy közelítését az *inverz* DWT alkalmazásával kaphatjuk meg.

A DWT-módszer közeli rokonságban áll az ún. *diszkrét Fourier-transzformációval* (DFT), ami egy trigonometrikus függvényekre (sin, cos) épülő jelfeldolgozási technika.



(a) Haar 2 (b) Daubechies 4

3.7. ábra | Példák wavelet-családokra. A waveletek neve mellett álló szám az adott wavelet *eltűnő momentumainak* számát adja meg. Ezek bizonyos matematikai összefüggéseket határoznak meg, amelyek az együtthatók számával függenek össze

Általában a DWT használatával jobb veszteséges tömörítés érhető el. Más szóval, ha egy rögzített adatvektor esetében ugyanannyi koefficienst tartunk meg a DWT, illetve DFT alkalmazásakor, a DWT által kapott transzformált az eredeti adatoknak pontosabb approximációját adja. Ezért adott pontosságú közelítés eléréséhez a DWT kevesebb helyet igényel, mint a DFT. A DFT alapfüggvényeitől eltérően a waveletek térben eléggé lokalizáltak, ami kedvez a lokális részletek megőrzésének.

Míg DFT-ből csak egy van, a DWT-knek számos családja létezik. A 3.7. ábra néhány wavelet-családot mutat be. A népszerű típusok között találjuk a Haar_2, a Daubechies_4 és Daubechies_6 transzformációkat. A diszkrét wavelet-transzformációk alkalmazásának általános sémája egy hierarchikus *piramis algoritmust* használ, ami minden iteráció során megfelezi az adatokat, s ezzel nagy számítási sebességet tesz lehetővé. A módszer az alábbi lépésekből áll:

(1) Az adatvektor L hosszának 2 valamely hatványával kell megegyeznie. Ez a feltétel szükség esetén az adatvektor nullákkal való kiegészítésével biztosítható.
(2) Minden elemi transzformáció két függvény alkalmazásából áll. Az első egyfajta simítást végez, például részösszegzés vagy súlyozott átlagolás révén. A második súlyozott különbséget képez, ami az adatok részleteit emeli ki.
(3) A két függvényt az input adatokból vett párokra alkalmazzuk, eredményül így két $L/2$ hosszúságú adatvektort kapunk. Általánosságban ez a két vektor az adatok simított vagy alacsony frekvenciás, illetve nagyfrekvenciás részét reprezentálja.
(4) A két függvényt rekurzív módon alkalmazzuk ismét az előző körben kapott adatvektorokra mindaddig, amíg a kapott vektorok hossza 2 nem lesz.
(5) A fenti eljárás során konstruált adatvektorok értékeiből meghatározzuk a transzformált adatok wavelet-koefficienseit.

Ugyanezeket az együtthatókat úgy is megkaphatjuk, hogy a bemeneti adatokat egy mátrixszal szorozzuk meg, amelynek értékei az alkalmazott DWT típusától függenek. A mátrixnak ortonormáltnak kell lennie, ami azt jelenti, hogy oszlopai egymásra páronként merőleges egységvektorok, vagyis a mátrix inverze megegyezik a transzponáltjával. Bár a hely rövidsége miatt ezt most nem tudjuk bemutatni, a mátrix ezen tulajdonsága lehetővé teszi az adatok visszaállítását a simított összeg- és különbségvektorokból. Az alkalmazott mátrixnak néhány ritka mátrix szorzatára történő faktorizálásával adódó ún. *gyors DWT* algoritmus n hosszúságú inputvektor esetén $O(n)$ bonyolultságú lesz.

A wavelet-transzformációkat többdimenziós adatokra, például adatkockákra is alkalmazhatjuk. Ennek során a transzformációt először az első, majd a második, harmadik stb. dimenzióra alkalmazzuk. Az elvégzett számítások bonyolultsága lineárisan nő az adatkockában lévő cellák számával. A wavelet-transzformációk jó eredményeket adnak a ritka vagy egyenetlen, illetve a rendezett attribútumokkal rendelkező adatok esetében. A wavelet alapú veszteséges tömörítés jobbnak tűnik a JPEG algoritmusnál, ami jelenleg ipari szabvány ezen a téren. A wavelet-transzformációkat az élet számos területén alkalmazzák, jelesül az ujjlenyomatok tömörítésében, a számítógépes látáskutatásában, idősorok elemzésénél, valamint adattisztítási céllal.

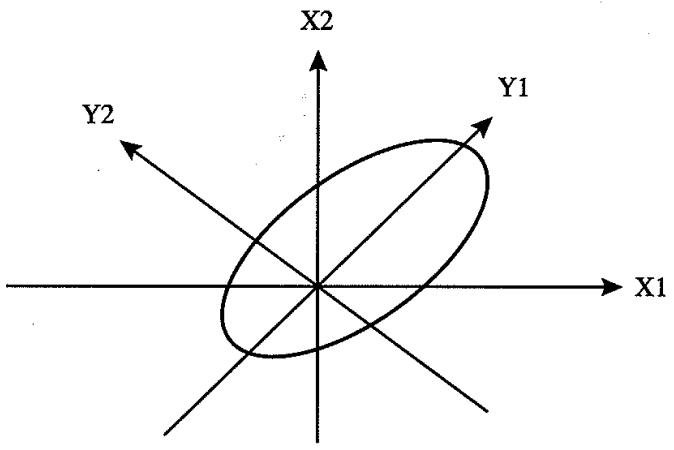
Főkomponens-analízis

Ebben az alfejezetben rövid bevezetést nyújtunk az adatok redukálásának módszereként alkalmazott főkomponens-analízisbe. A téma részletes elméleti tárgyalása meghaladja e könyv kereteit.

Tegyük fel, hogy a tömörítendő adatok N darab k-komponensű sorból vagy adatvektorból állnak. A **főkomponens-analízis** vagy **PCA (Principal Component Analysis)** (hívják még Karhunen–Loeve- vagy K–L-módszernek is) célja c darab olyan k-dimenziós vektor meghatározása, amelyekkel az adatok a lehető legjobban reprezentálhatók, ahol $c \leq k$. Az eredeti adatokat ily módon egy jóval kisebb altérre vetítjük, ami tömörítést eredményez. A PCA a dimenziócsökkentésre is alkalmazható módszer. Azonban amíg az attribútumok részhalmazának kiválasztásakor az eredeti attribútumok közül válogatunk, a PCA „kombinálja" az attribútumok tartalmát a kisebb számú új változó létrehozásával. Az eredeti adatok ezután erre a szűkebb halmazra vetíthetők.

Az alapvető eljárás a következő:

(1) Az input adatokat normalizálják, hogy minden attribútum értéke ugyanabba a tartományba essen. Ez a lépés biztosítja, hogy a nagy értékkészletű attribútumok ne nyomják el a kisebb értékkészletűeket.
(2) A PCA c darab ortonormált vektort határoz meg, amelyek a normalizált adatok egy bázisát adják. Ezek tehát egymásra páronként merőleges egységvektorok. Ezeket a vektorokat *főkomponenseknek* nevezzük. Az inputadatok a főkomponensek lineáris kombinációiként állnak elő.
(3) A főkomponenseket „jelentőség" vagy erősség szerint csökkenő sorba rendezzük. A főkomponensek gyakorlatilag új koordinátatengelyeket jelölnek ki az adatok számára, és fontos információt adnak azok szórásáról. A tengelyek rendezése ugyanis olyan, hogy az első tengelyhez tartozik a legnagyobb szórás, majd nagyságban ezt a második követi stb. A 3.8. ábrán például az eredetileg $X1$ és $X2$ tengelyek mentén ábrázolt adatok $Y1$ és $Y2$ főkomponenseit láthatjuk. Ez az információ segít az adatokban fellelhető csoportok, illetve minták felismerésében.



3.8. ábra | Főkomponens-analízis. Y1 és Y2 a két első főkomponens

(4) Mivel a komponenseket „jelentőség" szerint csökkenő sorba rendeztük, az adatok mérete csökkenthető a gyengébb, vagyis a kisebb szórással rendelkező komponensek eliminálásával. A domináns főkomponensek használatával ily módon az eredeti adatok jó közelítése adható meg.

A PCA nem igényel sok számítást; alkalmazható rendezett és rendezetlen attribútumok, illetve ritka és egyenetlen adatok kezelésére. A kettőnél több dimenziós adatok kezelésekor a problémát először kétdimenziósra kell redukálni. Például az eladási adatokra vonatkozó, *árucikk_típus*, *áruház* és *év* attribútumokat tartalmazó háromdimenziós adatkockát először kétdimenzióssá kell redukálni, mondjuk az *árucikk_típus* és az *áruház* × *év* dimenziókat használva. Az adattömörítés szempontjából a wavelet-transzformációkkal összehasonlítva elmondható, hogy a PCA rendszerint jobban kezeli a ritka adatokat, míg a wavelet-transzformációk jobban megfelelnek a sokdimenziós adatok kezelésére.

3.4.4. | Számosságcsökkentés

Csökkenthetjük-e az adatok méretét alternatív, „kisebb" reprezentációk használatával? A *számosságcsökkentés* módszerei valóban alkalmasak erre a feladatra. Ezek a technikák lehetnek paraméteresek vagy nem paraméteresek. A *paraméteres módszerek* esetében egy modellt használunk az adatok becslésére, így az eredeti adatok helyett csak a modell paramétereit kell tárolni (a szélsőséges értékeket szintén tárolhatjuk). Példaként említhetjük a loglineáris modelleket, amelyekkel sokdimenziós diszkrét valószínűség-eloszlások becsülhetők. A csökkentett reprezentációt adó *nem paraméteres* modellek közé tartoznak a hisztogramok, a klaszterezés és a mintavételezés.

A következőkben ezeket a számosságcsökkentési módszereket tekintjük át.

Regressziós és loglineáris modellek

A regressziós és loglineáris modellek az adatok közelítésére szolgálnak. A **lineáris regresszió** esetében az adatokra egyenest illesztünk. Az Y valószínűségi változót (ezt *válasz változónak* nevezzük) ekkor az X valószínűségi változó *(prediktor változó)* lineáris függvényeként modellezzük az alábbi egyenlet szerint:

$$Y = \alpha + \beta X, \tag{3.5.}$$

ahol az Y szórását konstansnak tekintjük. Az α és β együtthatók (nevük *regressziós koefficiens*) adják meg az illesztett egyenes meredekségét, illetve Y tengellyel vett metszetét. Ezeket az együtthatókat a *legkisebb négyzetek módszerével* határozhatjuk meg, ami minimalizálja az adatok számított egyenestől való eltéréseinek négyzetösszegét (vagyis a becslés hibáját). A **többdimenziós regresszió** a lineáris regresszió kiterjesztése, amelynél az Y válasz-változót egy többdimenziós vektorváltozó lineáris függvényeként modellezzük.

A **loglineáris modellek** többdimenziós diszkrét valószínűség-eloszlások approximációjára szolgálnak. A módszer segítségével egy diszkretizált attribútumhalmazhoz tartozó alapkocka minden cellájának valószínűsége megbecsülhető a kockahálót alkotó kisebb

részkockák alapján. Ennek segítségével az alacsonyabb rendű részkockákból magasabb rendűek képezhetők. A loglineáris modellek így egyrészt adattömörítésre is alkalmasak (mivel az alacsonyabb rendű részkockák együttesen rendszerint az alapkockánál kevesebb helyet igényelnek), másrészt adatsimításra is használhatók (mivel az alacsonyabb rendű részkockák esetében a cellák becslése kevésbé érzékeny a mintavételezés ingadozásaira, mint az alapkocka esetében).

Mind a regressziós, mind pedig a loglineáris modellek használhatók ritka adatokon, bár ilyen alkalmazhatóságuk korlátozott. Bár mindkét módszer tudja kezelni az egyenetlen adatokat, a regresszió különösen hatékony ezen a téren. A regresszió sokdimenziós adatok esetében gyakran sok számítást igényel, a loglineáris modellek azonban kb. 10 dimenzióig jól skálázhatók. A regressziós és loglineáris modelleket részletesebben tárgyaljuk a 7.8. alfejezetben.

Hisztogramok

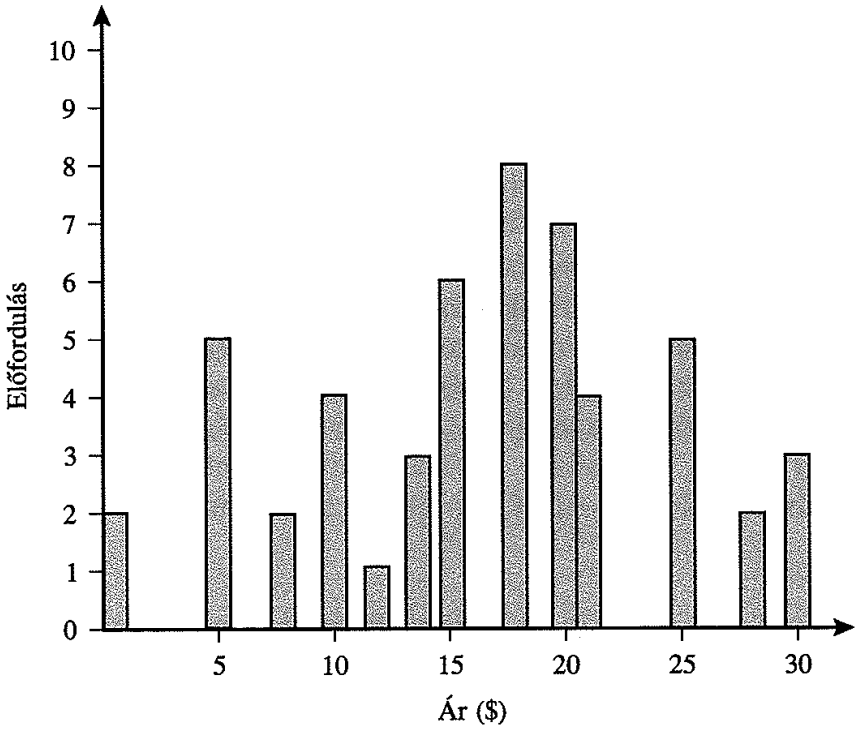
A **hisztogramok** kosarazási technikát alkalmaznak az adatok eloszlásának közelítésére, és az adatok redukálásának közkedvelt eszközei. Egy A attribútumhoz tatozó hisztogram az A attribútum adatait osztja fel *kosaraknak* nevezett diszjunkt részhalmazokra. A kosarakat a vízszintes tengelyen ábrázoljuk, a hozzájuk tartozó oszlopok magassága (és területe) pedig rendszerint a kosár által reprezentált értékek átlagos gyakoriságát tükrözi. Ha minden kosár csak egyetlen attribútumérték–gyakoriság párt reprezentál, akkor *egypontos kosárról* beszélünk. A kosarak azonban általában inkább az adott attribútum értékkészletének egy-egy folytonos tartományát reprezentálják.

3.4. példa | A következő adatsor az *ElektroMind* által forgalmazott leggyakoribb termékek árát tartalmazza (dollárra kerekítve). A sorba rendezett értékek: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

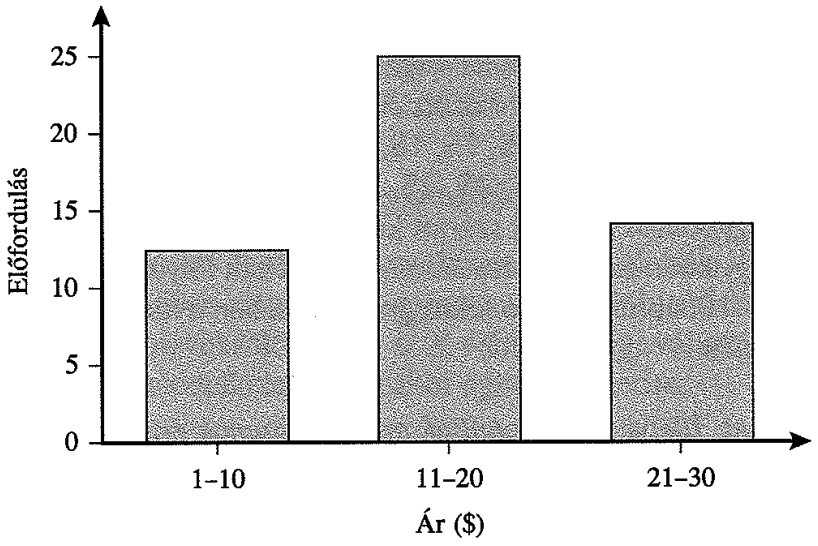
A 3.9. ábrán látható az adatok egypontos kosarak szerint képzett hisztogramja. Az adatok további redukálása érdekében gyakran az adott attribútum értékének folytonos tartományait rendelik a kosarakhoz. A 3.10. ábrán látható hisztogram esetében a kosarak az *ár* 10 dolláros tartományait képviselik. ■

Hogyan határozzuk meg a kosarakat, illetve particionáljuk az attribútumok értékeit? Számos partíciós stratégiát ismerünk, nézzünk ezek közül néhányat.

- **Azonos (oszlop)szélességű** – Ennél a hisztogramnál a kosarak (oszlopok) szélessége (az intervallumok nagysága) egyforma. Ilyenek a 3.10. ábra 10 dolláros kosár intervallumai.
- **Azonos (oszlop)magasságú** (vagy elemszámú) – Ebben az esetben a kosarakat úgy alakítjuk ki, hogy a hozzájuk tartozó gyakoriság nagyjából egyforma legyen (vagyis minden kosárhoz körülbelül ugyanannyi adatérték tartozik).
- **V-optimális** – A rögzített számú kosárral rendelkező összes lehetséges hisztogram közül ez rendelkezik a legkisebb szórással. A hisztogramok szórása az eredeti értékek-



3.9. ábra | Az *ár* adatainak hisztogramja egypontos kosarakkal – minden oszlop egy árérték–gyakoriság párt reprezentál.



3.10. ábra | Az *ár* azonos (oszlop)szélességű hisztogramja. Az adatok összevonásánál egyforma, 10 $ szélességű kosarakat használtunk

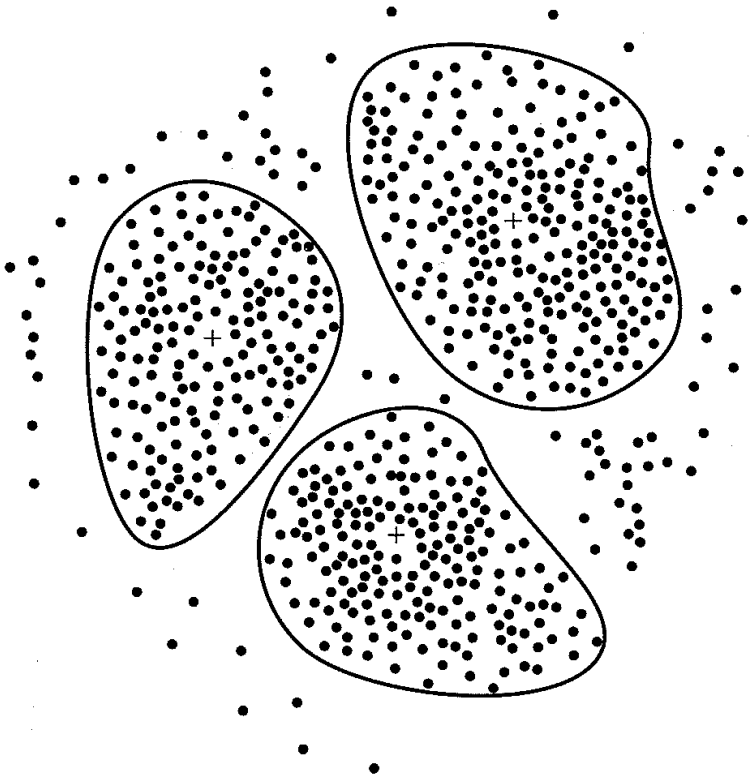
ből képzett súlyozott átlag, ahol az egyes kosarakhoz rendelt súlyok a kosár által reprezentált értékek számával egyeznek meg.

- **MaxDiff** – Ekkor a szomszédos értékek közötti különbségeket vesszük alapul a kosarak kialakításában. A kosarakat elválasztó határok a $\beta - 1$ darab, legnagyobb különbséggel rendelkező pár között húzódnak, ahol β értékét a felhasználó adja meg.

Általában a V-optimális és MaxDiff hisztogramok a leginkább pontos és praktikus változatok. A hisztogramok rendkívül hatékonyak mind a ritka, mind a sűrű, illetve akár a nagyon egyenetlen, akár az igen egyenletes adatok approximációjában. Az imént bemutatott hisztogramtípusok több attribútum együttes kezelésére is használhatók. A *többdimenziós hisztogramok* kimutathatják az attribútumok közötti függőségeket. Az ilyen hisztogramok még öt attribútum kezelésével is képesek az adatok jó közelítésére. A hisztogramok hatékonyságának vizsgálata nagyon sok dimenzió esetén még további vizsgálatokat igényel. Az egypontos kosarak alkalmazása a nagy gyakoriságú szélsőséges értékek reprezentációjában hasznos. A hisztogramokkal még foglalkozunk az 5.5. alfejezetben.

Klaszterezés

A klaszterezési technikák az adatok sorait tekintik alapvető objektumnak. Ezeket az objektumokat csoportokba, ún. *klaszterekbe* sorolják oly módon, hogy a „hasonló" objektumok azonos, míg az „eltérőek" különböző klaszterbe kerüljenek. A hasonlóság fogalmát rendszerint annak alapján definiálják, hogy az objektumok milyen „közel" vannak egymáshoz valamilyen távolságmérő függvény szerint. A klaszter „minősége" annak *átmérőjével* jellemezhető, ami nem más, mint a klaszterbeli objektumok maximális távolsága. Az ún. *centrumtávolság* a klaszter minőségének alternatív jellemzésére szolgálhat – ezt a klaszter objektumainak a klaszter centrumától (ami az „átlagos objektumot", vagy másképp a klaszter pontjainak átlagát jelenti) vett átlagos távolságaként definiálják. A 3.11. ábra a vásárlók

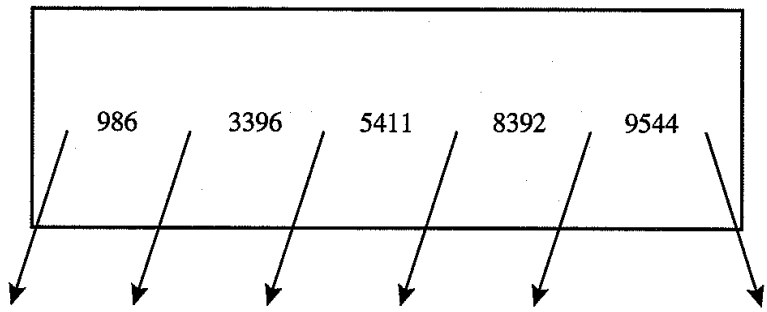


3.11. ábra | A vevők lakhelyének 2-dimenziós ábrázolása egy városban, három klaszterrel. A klaszterek centrumát + jelzi

lakhelyének elhelyezkedését mutatja 2 dimenzióban, a klaszterek centrumát pedig + jelöli. Az ábrán három klaszter látható.

Adatok redukálása során a konkrét adatértékeket a klaszterekkel történő reprezentáció váltja fel. E módszer hatékonysága az adatok természetétől függ. Jól elkülönülő adatcsoportok esetén természetesen hatékonyabb, mint az „elkent", diffúz struktúrájú adatoknál.

Az adatbázisrendszerek a **többdimenziós indexfákat** elsősorban a gyors adatelérés biztosításához használják. Emellett használhatók hierarchikus adatredukálás céljából is, amennyiben az adatok többféle részletességű klaszterezését adják. Ennek segítségével a lekérdezésekre közelítő válaszok adhatók. Az indexfák adatobjektumok adott halmazának többdimenziós terét rekurzív módon pairticionálják, ahol a fa gyökerét jelző pont (a fa kiindulópontja) maga a teljes tér. Az ilyen fák tipikusan kiegyenlítettek, belső és végpontokkal rendelkeznek. Minden szülő (azaz nem végpont) a hozzá kapcsolódó utódpontokra (vagyis az eggyel alacsonyabb szintű pontokra) utaló kulcsokat és mutatókat tartalmaz, amelyek együttesen leírják a szülő által reprezentált teret. Minden végpont az általa reprezentált adatsorokra utaló mutató(ka)t tartalmaz.



3.12. ábra | Egy konkrét adatokhoz készített B+ fa gyökere

Egy indexfa ily módon összevont és részletes adatokat egyszerre tartalmaz a részletesség vagy absztrakció több szintjén. Ezáltal az adathalmaz klaszterezési hierarchiáját is nyújtja, amelyben minden klaszter tartalmaz egy címkét, ami a hozzá tartozó adatokra utal. Ha szülők utódait kosaraknak tekintjük, akkor az indexfákat *hierarchikus hisztogramnak* is tekinthetjük. Példaként tekintsük a 3.12. ábrán látható B+ fa gyökerét, ami a 986, 3396, 5411, 8392 és 9544 adatkulcsokhoz tartalmaz mutatókat. Tegyük fel, hogy a fa 10 000 sort tartalmaz, melyeknek adatkulcsai 1-től 10 000-ig terjednek. A fa által reprezentált adatok egy azonos magasságú hisztogrammal közelíthetők, amelynek hat kosara az [1, 985], [986, 3395], [3396, 5410], [5411, 8391], [8392, 9543] és [9544, 10 000] intervallumokat fedi le. Minden kosár durván 10 000/6 értéket tartalmaz. Hasonlóképp, minden kosár kisebb kosarakra osztható fel, ami a részletesség finomabb szintjén összesített adatok tárolását teszi lehetővé. A többdimenziós indexfák adatok redukálására való felhasználása az attribútumok értékeinek rendezésére alapul (minden dimenzióban el kell végezni a rendezést). A többdimenziós indexfák között találjuk az R-fákat, négyesfákat és variációikat. Ezek jól használhatók mind ritka, mind egyenetlen adatok kezelésére.

A klaszterek és a klaszterminőség definiálására számos mértéket használnak. A klaszterezési módszerek bemutatását a 8. fejezetben folytatjuk.

Mintavételezés

A mintavételezés szintén használható adatok redukálására, mivel lehetővé teszi, hogy egy nagyméretű adathalmazt egy sokkal kisebb véletlen mintával (vagy részhalmazzal) reprezentáljunk. Tegyük fel, hogy egy nagy D adathalmaz N darab sort tartalmaz. Tekintsünk át néhány mintavételezési módszert:

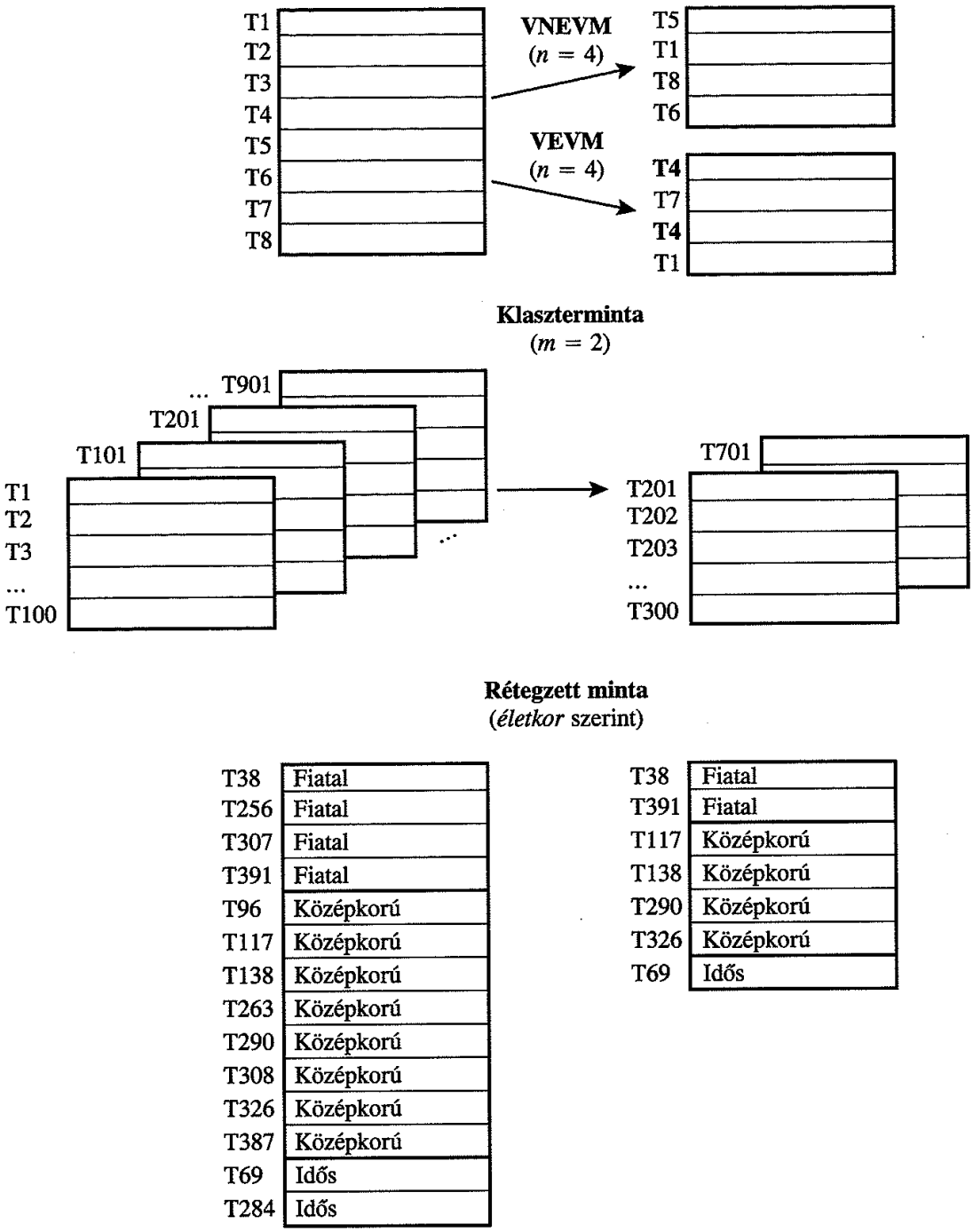
- **Visszatevés nélküli, n elemű egyszerű véletlen minta (VNEVM)**[6] – Ezt úgy kapjuk, hogy az N darab sorból n-et választunk ki ($n < N$); a D halmaz minden sorának kiválasztása egyforma, $1/N$ valószínűséggel történhet meg.
- **Visszatevéses, n elemű egyszerű véletlen minta (VEVM)**[7] – Hasonló az előbbihez, de a kiválasztott sort a húzás eredményének rögzítését követően *visszatesszük* a D halmazba. Más szóval, ugyanazt az elemet többször is kiválaszthatjuk.
- **Klaszterminta** – Ha a D halmaz sorait M darab páronként diszjunkt „klaszterbe" soroljuk, ezek közül kiválaszthatunk egyszerű véletlen mintavételezéssel m darabot ($m < M$). Például az adatbázisokból általában laponként olvassuk ki a sorokat, így ezeket a lapokat tekinthetjük klasztereknek. Az adatok redukált reprezentációját adja például a lapokból történő VNEVM-kiválasztás, ezt hívjuk klaszтermintának.
- **Rétegzett minta** – Ha a D halmazt páronként diszjunkt részekre, ún. *rétegekre* osztjuk fel, és mindegyikre elvégzünk egy-egy egyszerű véletlen mintavételezést, rétegzett mintát kapunk. Ez növeli annak az esélyét, hogy reprezentatív mintát kapjunk, különösen egyenetlen adatok esetén. Például a vásárlók adataiból rétegzett mintát vehetünk úgy, hogy az életkor szerinti csoportokat választjuk rétegeknek. Ily módon a legkevesebb vásárlót tömörítő korcsoport is biztosan reprezentálva lesz.

A fenti mintavételezési módszereket mutatja be sematikusan a 3.13. ábra. Az adatok redukálására ezeket használják a leggyakrabban.

A mintavételezési módszerek egyik előnye az adatok redukálásában az, hogy a minta előállításának költsége a minta elemszámával, n-nel, nem pedig az adathalmaz méretével, N-nel arányos. Ezért a mintavételezés bonyolultsága potenciálisan *szublineárisan* viszonyul az adathalmaz méretéhez. Más adatredukálási módszerek esetében legalább egyszer végig kell vizsgálni a teljes D halmazt. Rögzített mintaméret esetén a mintavételezés bonyolultsága az adatok dimenziójának d számával csak lineárisan nő, míg hisztogramok használatával például ez a növekedés exponenciális.

Az összesített lekérdezésekre adott közelítő válaszok esetében a mintavételezés a leggyakrabban alkalmazott adatredukálási eszköz. A centrális határeloszlási tétel segítségével meg lehet határozni a minta szükséges méretét egy függvény adott hibával történő közelítéséhez. Ez az n mintaméret rendkívül kicsi lehet az adathalmaz N méretéhez képest. A mintavételezés a legkézenfekvőbb módszer egy redukált adathalmaz progresszív finomítására. Az ilyen halmazok egyszerűen tovább finomíthatók a minták méretének növelésével.

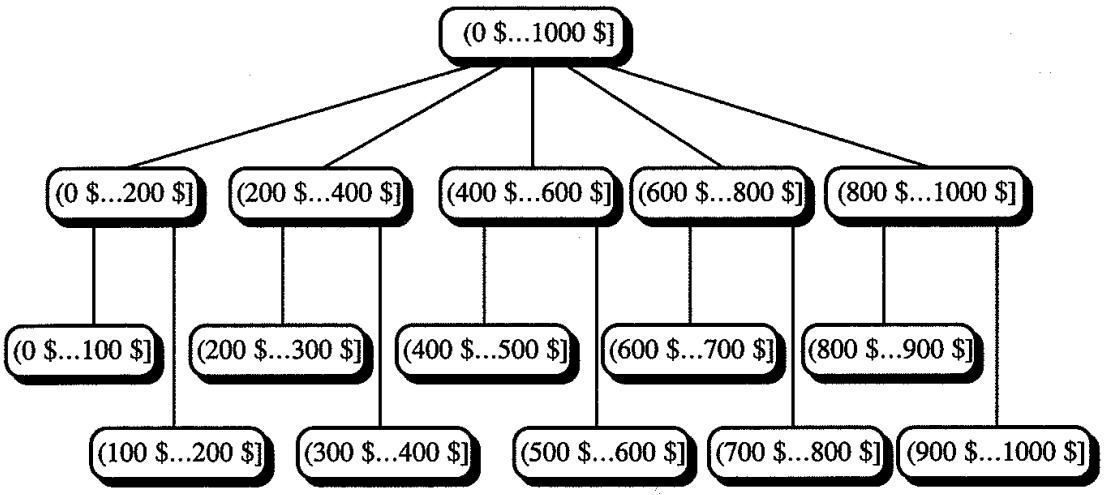
6 | Angolul: Simple Random Sample Without Replacement (SRSWOR).
7 | Angolul: Simple Random Sample With Replacement (SRSWR).



3.13. ábra | A mintavételezés használata adatcsökkentésre

3.5. | Diszkretizáció és fogalmi hierarchiák generálása

A diszkretizációs módszerek arra szolgálnak, hogy a folytonos attribútumok által felvett értékek számát csökkentsék az attribútum értékkészletének intervallumokra osztásával. Ezután az intervallumcímkék helyettesíthetik a tulajdonképpeni értékeket. Az attribútumok értékeinek csökkentése különösen hasznos abban az esetben, ha az előfeldolgozott adatokon döntési fára épülő módszerekkel végzünk klasszifikációs bányászatot. Ezek a módszerek jellemzően rekurzívak, és minden lépésben jelentős időt fordítanak az adatok



3.14. ábra | Az *ár* attribútumhoz tartozó fogalmi hierarchia

rendezésére. Ezért minél kevesebb különböző értéket kell rendezni, annál gyorsabban működnek. Számos diszkretizációs technika alkalmazható rekurzívan azzal a céllal, hogy attribútumok értékeinek hierarchikus vagy többszintű részletességgel rendelkező felosztását adja, amit fogalmi hierarchiának nevezünk. A fogalmi hierarchiák, amiket a 2. fejezetben vezettünk be, az absztrakció több szintjén végzett bányászatban alkalmazhatók.

Egy adott numerikus attribútumhoz tartozó fogalmi hierarchia az attribútum egy diszkretizációját határozza meg. A fogalmi hierarchiák használhatók az adatok redukálására az alacsony szintű fogalmak (például az *életkor* attribútumhoz tartozó numerikus értékek) összevonása, és magasabb szintű fogalmakkal (például *fiatal*, *középkorú* és *idős*) való helyettesítése révén. Bár az adatok ilyesfajta általánosítása során a részletek elvesznek, az általánosított adatok informatívabbak és könnyebben interpretálhatók lehetnek, valamint az eredeti adatoknál kevesebb helyet foglalnak. A redukált adathalmazon folytatott bányászat kevesebb ki- és beviteli műveletet igényel, és hatékonyabb a nagyobb, nem általánosított adatokon végzett bányászatnál. Az *ár* attribútumhoz tartozó fogalmi hierarchiára mutat egy példát a 3.14. ábra. Ugyanahhoz az attribútumhoz a különböző felhasználók egyéni igényeinek megfelelően többféle fogalmi hierarchia is készíthető.

A fogalmi hierarchiák manuális felépítése roppant körülményes és időigényes feladat a felhasználók vagy a szakterület szakértője számára. Szerencsére számos hierarchia implicit módon jelen van az adatbázis sémájában, és így a sémák szintjén definiálható. A fogalmi hierarchiák gyakran automatikusan generálhatók, illetve dinamikus módon finomíthatók az adatok eloszlásának statisztikai elemzése révén.

Az alábbiakban a fogalmi hierarchiák generálását tekintjük át a numerikus és kategória típusú adatok esetében.

3.5.1. | Diszkretizáció és fogalmi hierarchiák generálása numerikus adatokból

Az adatok lehetséges értékkészleteinek sokfélesége és az adatértékek gyakori frissítése miatt a numerikus attribútumokhoz tartozó fogalmi hierarchiák explicit megadása rendkívül időigényes és fáradságos munka. Emellett a manuális módszer meglehetősen önkényes is.

A numerikus attribútumokhoz tartozó fogalmi hierarchiák automatikusan is létrehoz-

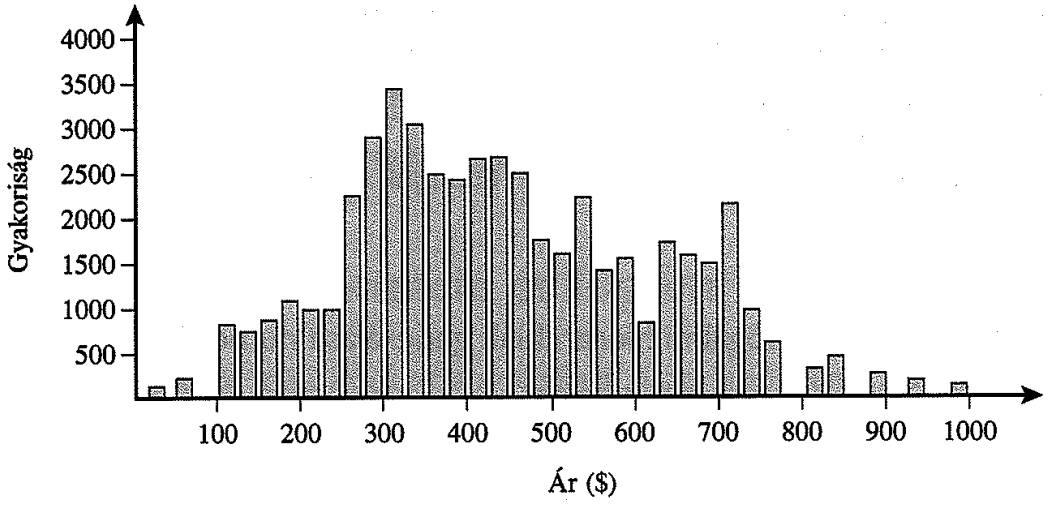
hatók az adatok eloszlásának elemzésével. A numerikus fogalmi hierarchiák generálására szolgáló módszerek közül ötöt ismertetünk: a *kosarazás*, a *hisztogram-analízis*, a *klaszterelemzés*, az *entrópia alapú diszkretizáció* és a *„természetes pairticionálás" szerinti adatszegmentálás* technikáját.

Kosarazás

A 3.2.2. alfejezetben tárgyaltuk az adatsimításra használt kosarazási módszereket. Ezek egyúttal diszkretizációra is alkalmasak. Az adatok diszkretizálhatók például úgy, hogy az értékeket először kosarakba csoportosítjuk, majd az egyes kosarakba tartozó értékeket a kosár átlagával vagy mediánjával helyettesítjük, éppen úgy, mint a *kosárátlagok* vagy *kosármediánok szerinti adatsimítás* esetében. Ezek a módszerek a kapott partíciókon rekurzívan alkalmazva fogalmi hierarchiák generálására is használhatók.

Hisztogramanalízis

A hisztogramok, amikkel a 3.4.4. alfejezetben foglalkoztunk, szintén használhatók adatok diszkretizációjára. A 3.15. ábrán egy hisztogram látható, amely az *ár* attribútum adatainak egy konkrét eloszlását mutatja be. A leggyakoribb árak durván 300–325 $ között vannak. Az értékek tartományainak meghatározásához különféle partíciós szabályokat használhatunk. Például egy *azonos oszlopszélességű* (vagy *ekvidisztans*) hisztogram esetében az értékeket egyforma méretű intervallumokba particionáljuk (például (0 $...100 $], (100 $...200 $], ..., (900 $...1000 $]). *Azonos oszlopmagasságú* (vagy *azonos elemszámú*) hisztogram esetén az értékeket úgy osztjuk fel, hogy lehetőség szerint minden partíció egyforma számú mintát (adatértéket) tartalmazzon. A hisztogram analízis algoritmusa rekurzív módon alkalmazható minden partícióra; ezzel a módszerrel egy többszintű fogalmi hierarchia hozható létre, az eljárás pedig addig folytatható, amíg a fogalmi szintek előre megadott számát el nem érjük. A rekurzív algoritmus szabályozására a *minimális intervallumméret* szintenként adható meg. Ez adja meg szintről szintre a partíciók minimális szélességét, illetve a partíciókban szereplő értékek minimális számát.



3.15. ábra | Az *ár* attribútum értékeinek eloszlását mutató hisztogram

Klaszterelemzés

Klaszterezési algoritmus használatával az adatok csoportokba vagy klaszterekbe pairticionálhatók. Minden klaszter a fogalmi hierarchia (mint gráf) egy-egy csomópontja lesz, és minden csomópont ugyanahhoz a fogalmi szinthez tartozik. Minden klaszter tovább bontható szubklaszterekre, a hierarchia alacsonyabb szintjét adva. A klaszterek össze is vonhatók a hierarchia magasabb fogalmi szintjét képezve. Az adatbányászatban használt klasztermódszereket a 8. fejezetben tárgyaljuk.

Entrópia alapú diszkretizáció

Egy A numerikus attribútum értékeinek rekurzív pairticionálására egy *entrópiának* nevezett információ alapú mérték is használható, ami az attribútum hierarchikus diszkretizációját adja. Az ilyen diszkretizáció az attribútum numerikus fogalmi hierarchiáját szolgáltatja. Ha adott az adatsorok egy S halmaza, az A attribútum entrópiára épülő diszkretizációjának alapvető módszere a következő:

(1) Az A minden értékét potenciális intervallumhatárnak vagy T küszöbértéknek tekintjük. Például az A egy v értéke az S halmaz elemeit két részhalmazra osztja az $A < v$, illetve $A \geq v$ feltételeknek megfelelően, vagyis bináris diszkretizációt ad.

(2) Adott S halmazhoz a küszöbértéket úgy választjuk meg, hogy a létrejövő pairticionálásból adódó információnyereség maximális legyen. Az információnyereség mértéke:

$$I(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2), \tag{3.6}$$

ahol S_1 és S_2 az $A < T$, illetve $A \geq T$ feltételeket kielégítő mintákat jelenti. Egy adott halmaz *Ent* entrópia függvénye a halmazban található minták osztályeloszlása alapján számítható ki. Például m osztály esetén az S_1 entrópiája a következő:

$$Ent(S_1) = -\sum_{i=1}^{m} p_i \log_2(p_i), \tag{3.7}$$

ahol p_i jelöli az i-edik osztály valószínűségét (relatív gyakoriságát) az S_1 részhalmazban. Az S_2 halmaz entrópiája ugyanígy kapható meg.

(3) A küszöbérték meghatározására szolgáló eljárást rekurzívan alkalmazzuk a kapott partíciókra mindaddig, amíg valamilyen megállási feltétel nem teljesül, mint például

$$Ent(S) - I(S,T) > \delta. \tag{3.8}$$

Az entrópia alapú diszkretizáció csökkentheti az adatok méretét. Az eddig felsorolt módszerekkel ellentétben az entrópia alapú diszkretizáció felhasználja az osztályokra vonatkozó információt. Emiatt valószínűbb, hogy az intervallumok határai megfelelő helyre kerülnek, ami növeli az osztályozás pontosságát. Az információnyereség és entrópia itt bemutatott mértékeit a döntésifa-algoritmusok is használják. Ezeket a mértékeket az 5.3.2. és a 7.3.1. alfejezetben részletesebben is megvizsgáljuk.

Természetes partíció szerinti szegmentálás

Bár a kosarazás, a hisztogram analízis, a klaszterezés és az entrópia alapú diszkretizáció mind hasznos a numerikus hierarchiák generálásában, számos felhasználó előnyben részesíti a numerikus tartományoknak viszonylag egyforma méretű, könnyen azonosítható, „természetes" intervallumokra történő pairticionálását. Például az éves fizetések felosztása (50 000 $, 60 000 $] típusú „szép" intervallumokra gyakran kívánatosabb, mint egy kifinomult klaszterelemzés által kapott partíció, mondjuk (51 263,98 $, 60 872,34 $] és hasonló intervallumokkal.

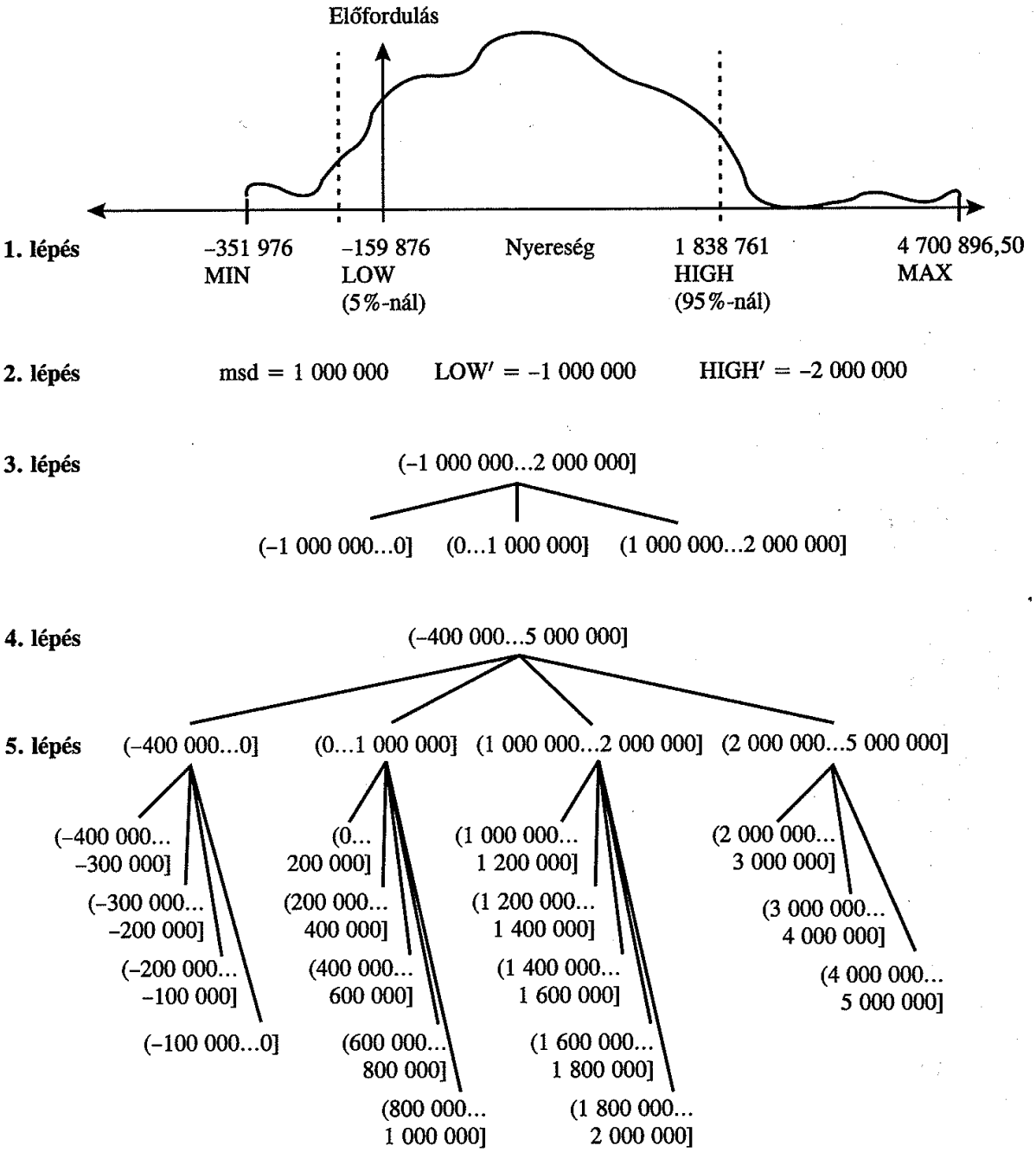
A numerikus adatoknak egy viszonylag egyenletes, „természetes" intervallumokra való felosztása a 3-4-5 szabállyal kapható meg. Általában véve a módszer adatok rögzített tartományát rekurzív módon lépésről lépésre 3, 4 vagy 5, többé-kevésbe egyenlő méretű intervallumra osztja fel a nagyságrendet meghatározó helyi érték alapján. Az alkalmazott szabály a következő:

- Ha egy intervallum 3, 6, 7 vagy 9 különböző értéket fed le a legnagyobb (nagyságrendet meghatározó) helyi értéken, akkor a tartományt 3 intervallumra osztjuk fel (3 azonos szélességű intervallumra a 3, 6 vagy 9 esetén, és 2 : 3 : 2 arányú intervallumokra 7 esetén).
- Ha az intervallum 2, 4 vagy 8 különböző értéket fed le a legnagyobb helyi értéken, akkor 4 azonos méretű intervallumra osztjuk.
- Ha 1, 5 vagy 10 különböző érték jelenik meg a legnagyobb helyi értéken, akkor a tartományt 5 azonos méretű intervallumra osztjuk.

A szabály minden így kapott intervallumra rekurzívan alkalmazható, ily módon az adott numerikus attribútum egy fogalmi hierarchiáját kapjuk. Mivel az adathalmazban előfordulhatnak kivételesen nagy pozitív vagy negatív értékek, a legfelső szinten történő szegmentáció, amely kizárólag a minimális és maximális értékeket veszi figyelembe, helytelen eredményt adhat. Például valamely mintában néhány személy bizonyos adatai kiugróan eltérhetnek a többiekétől. A maximális adatértékekre épülő szegmentáció torz hierarchiát eredményezhet. Ezért a legfelső szinten a szegmentálást az adatok többségét (például nagyság szerinti eloszlásban az 5%-tól a 95%-ig) reprezentáló értékek alapján érdemes végezni. A kiugróan nagy vagy kis értékek e szegmentációtól elkülönítve, önálló intervallumként kezelhetők ugyanezzel a módszerrel.

A következő példa a 3-4-5 szabály alkalmazását mutatja be egy numerikus hierarchia automatikus létrehozása során.

3.5. példa | Tegyük fel, hogy az *ElektroMind* különböző áruházainak 1999-ben elért nyeresége széles tartományt fog át, –351 976 dollártól (veszteség) 4 700 896,5 dollárig *(a következőkben a numerikus értékek dollárban értendők)*. Egy felhasználónak szüksége van a *nyereség* értékeihez tartozó fogalmi hierarchia automatikus generálására. A könnyebb olvashatóság érdekében az (*l*, *r*] alakú intervallumokat a továbbiakban (*l*...*r*] módon jelöljük, így például (–1 000 000...0] a –1 000 000 $-tól 0 $-ig terjedő, balról nyílt, jobbról zárt intervallumot jelenti.



3.16. ábra | A *nyereség* automatikusan generált fogalmi hierarchiája a 3-4-5 szabály alapján (az értékek dollárban értendők)

Tegyük fel, hogy a nagyság szerinti eloszlásban az 5%-tól 95%-ig terjedő adatok –159 876 és 1 838 761 között vannak. A 3-4-5 szabály alkalmazásának eredményét a 3.16. ábrán láthatjuk.

(1) A fentiek szerint a minimális és maximális értékek: *MIN* = –351 976 és *MAX* = 4 700 896,5. Az alsó (vagyis 5%-os) és felső (azaz 95%-os) értékek a legfelső szintű felosztáskor: *LOW* = –159 876 és *HIGH* = 1 838 761.

(2) A *LOW* és *HIGH* értékek alapján a nagyságrendet megadó (első releváns) helyi érték a hetedik jegyben van (milliós érték, $msd^8 = 1\ 000\ 000$). A *LOW* és *HIGH* értékek milliósra való le-, illetve felkerekítésével kapjuk: $LOW' = -1\ 000\ 000$, illetve $HIGH' = 2\ 000\ 000$.

(3) Mivel az így kapott tartomány a legmagasabb helyi értéken három jegyet fog át (hiszen a fenti kerekített értékek különbsége 3 000 000), ezért a 3-4-5 szabály alapján három, azonos méretű intervallumra osztjuk fel: (–1 000 000...0], (0...1 000 000] és (1 000 000...2 000 000]. Ezek alkotják a hierarchia legfelső szintjét.

(4) Most megvizsgáljuk a *MIN* és *MAX* értékeknek az imént előállított első szintű intervallumokhoz való viszonyát. Mivel az első intervallum lefedi a *MIN* értéket, vagyis $LOW' < MIN$, így ennek bal oldali határát feljebb vihetjük a méret csökkentése érdekében. A *MIN* érték nagyságrendjét megadó számjegy a százezres helyi értéken van, így eszerint lefelé kerekítve a korrigált $MIN' = -400\ 000$ értéket kapjuk. Ezért az újradefiniált első intervallum (–400 000...0] lesz.

Másrészt, mivel az utolsó intervallum nem foglalja magába a *MAX* értéket, azaz $MAX > HIGH'$, ezért új intervallumot kell létrehoznunk ennek lefedésére. A *MAX* értéket nagyságrendjének megfelelően kerekítve ez az új intervallum a (2 000 000...5 000 000] lesz. Mindezek alapján a hierarchia legfelső szintje végül négy intervallumból áll: (–400 000...0], (0...1 000 000], (1 000 000...2 000 000] és (2 000 000...5 000 000].

(5) A kapott intervallumok mindegyike rekurzív módon tovább pairticionálható a 3-4-5 szabálynak megfelelően. Ezzel a hierarchia eggyel alacsonyabb szintjét kapjuk meg:

- Az első intervallumot négy részre osztjuk: (–400 000...–300 000], (–300 000...–200 000], (–200 000...–100 000] és (–100 000...0].
- A második intervallumot 5 részre osztjuk: (0...200 000], (200 000...400 000], (400 000...600 000], (600 000...800 000] és (800 000...1 000 000].
- A harmadik intervallumot szintén öt részre osztjuk: (1 000 000...1 200 000], (1 200 000...1 400 000], (1 400 000...1 600 000], (1 600 000...1 800 000] és (1 800 000...2 000 000].
- Az utolsó intervallumot három részre osztjuk: (2 000 000...3 000 000], (3 000 000...4 000 000] és (4 000 000...5 000 000].

A 3-4-5 szabályt szükség szerint ugyanígy alkalmazhatjuk a mélyebb szinteken is. ■

3.5.2. | Fogalmi hierarchiák generálása kategória típusú adatokból

A kategória típusú adatok diszkrét adatok. A kategória-attribútumok véges (de esetenként igen nagy) számú különböző értéket vehetnek fel, amelyek körében nincs rendezési reláció. Példaként említhető a *földrajzi elhelyezkedés*, a *munkakör* vagy az *árucikk típusa*. A fogalmi hierarchiák kategória típusú adatokból történő automatikus generálására számos módszer létezik.

8 | Az angol *Most Significant Digit*, azaz legelső értékes jegy vagy helyi érték mozaikszava. *(A fordító megjegyzése)*

Az attribútumok egy részleges rendezésének megadása a séma szintjén – A kategória-attribútumok vagy dimenziók fogalmi hierarchiája rendszerint attribútumok egy csoportját kezeli. Egy felhasználó vagy szakértő könnyen definiálhat fogalmi hierarchiát az attribútumok sémaszintű részleges vagy teljes rendezésének megadásával. Egy relációs adatbázis vagy adattárház *hely* dimenziója például a következő attribútumokat foglalhatja össze: *utca*, *város*, *megye_vagy_tartomány* és *ország*. Egy hierarchiát definiálhatunk az attribútumoknak a séma szintjén történő teljes rendezésével, ami ez esetben mondjuk *utca* < *város* < *megye_vagy_tartomány* < *ország*.

A hierarchia egy részének megadása explicit adatcsoportosítással – Ez gyakorlatilag a fogalmi hierarchia egy darabjának manuális megadását jelenti. Nagyméretű adatbázisokban lehetetlen teljes fogalmi hierarchiát az értékek explicit felsorolásával megadni. A köztes adatszintek kis darabkáin azonban lehetséges explicit módon csoportosítani az adatokat. Például ha már a séma szintjén definiáltuk, hogy a *tartomány* és az *ország* hierarchiát alkot, egyes köztes szintek megadásához az adatok manuálisan is kiegészíthetők, például [*Alberta, Saskatchewan, Manitoba*] ⊂ *Kanadai_préri*, illetve {*Brit_Kolumbia, Kanadai_préri*} ⊂ *Nyugat_Kanada*.

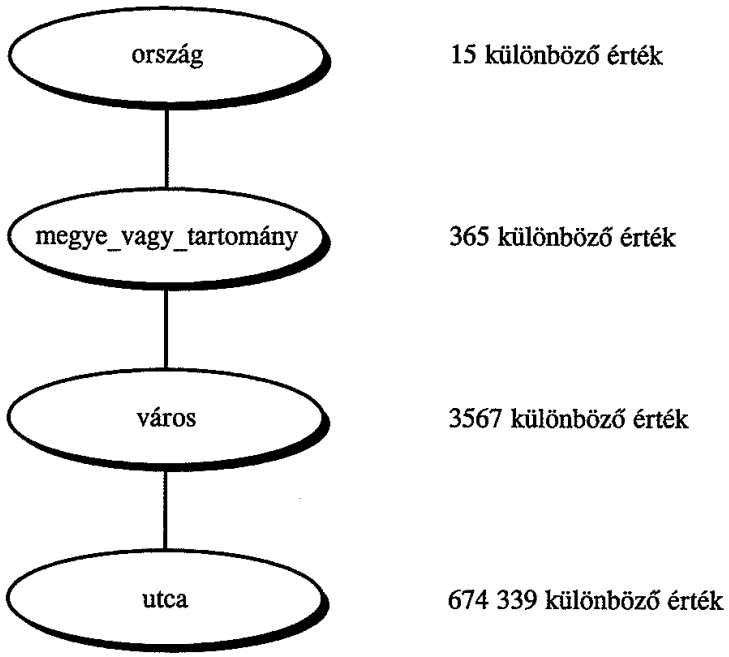
Egy attribútumhalmaz megadása részleges rendezés nélkül – A felhasználó a fogalmi hierarchiát alkotó attribútumok halmazát megadhatja anélkül is, hogy a rájuk vonatkozó rendezést definiálná. A rendszer ekkor megkísérelheti automatikusan generálni az attribútumoknak egy értelmes fogalmi hierarchiára vezető rendezését.

Miként lehet az adatszemantika bármiféle ismeretének hiányában kategóriaattribútumok egy tetszőleges halmazán hierarchikus rendezést értelmezni? Ennek megválaszolásában segítségünkre lehet az az észrevétel, hogy a magasabb szintű fogalmak – mivel ezek általában számos alárendelt, alacsonyabb szintű fogalmat fednek le – leírására szolgáló attribútumok kevesebb különböző értékkel rendelkeznek, mint az alacsonyabb szintű fogalmakra vonatkozók. Ennek ismeretében a fogalmi hierarchia automatikusan generálható az attribútumok által felvett különböző értékek száma alapján. A legtöbb különböző értékkel rendelkező attribútum a hierarchia legalacsonyabb szintjére kerül. Általában minél kevesebb különböző értéke van egy attribútumnak, annál magasabb szintre kerül a generált hierarchiában. Ez a heurisztikus szabály számos esetben beválik. A generált hierarchiát megvizsgálva némi lokális csereberét és finomhangolást a felhasználó vagy szakértő már manuálisan is elvégezhet.

Vizsgáljuk meg a módszer működését egy példán keresztül!

3.6. példa | Tegyük fel, hogy egy felhasználó az *ElektroMind* vásárlói adatbázisának *cím* dimenziójából az *utca*, *ország*, *megye_vagy_tartomány* és *város* attribútumokat választja ki, de nem adja meg ezek hierarchikus rendezését.

A *cím* fogalmi hierarchiájának automatikus generálása a következőképpen történik. Elsőként az attribútumokat azok különböző értékeinek száma alapján csökkenő sorba rendezik. Ennek eredménye a következő (zárójelben az előforduló értékek számával): *ország* (15), *megye_vagy_tartomány* (365), *város* (3567) és *utca* (674 339). Másodszor, a hierarchiát felülről lefelé haladva a kapott sorrendnek megfelelően készítjük el, vagyis a sorrendben első attribútum a legfelső, az utolsó a legalsó szintre kerül. Az ily módon előálló hierarchiát



3.17. ábra | Egy séma fogalmi hierarchiájának automatikus generálása az attribútumok különböző értékeinek száma alapján

a 3.17. ábrán láthatjuk. Végezetül, a felhasználó megvizsgálhatja, és szükség szerint módosíthatja a generált hierarchiát, hogy az a kívánt szemantikai összefüggéseket (is) tükrözze. Nyilvánvaló, hogy ebben a példában nincs szükség a generált hierarchia módosítására. ■

Jegyezzük meg, hogy ez a heurisztikus szabály nem alkalmazható megfontolás nélkül minden alkalommal, hiszen vannak triviális esetek is, amelyek eltérő viselkedést mutatnak. Tegyük fel például, hogy egy adatbázis *idő* dimenziója 20 különböző évet, 12 különböző hónapot és a hét 7 különböző napját tartalmazza. Hiba lenne mindazonáltal azt gondolnunk, hogy az *év* < *hónap* < *hét_napja* lenne a helyes idő szerinti hierarchia (a *hét_napja* attribútummal a hierarchia csúcsán).

Az attribútumok egy részleges halmazának megadása – Néha a felhasználók hanyagul definiálják a hierarchiát, vagy csak távoli elképzelésük van arról, hogy mely attribútumokat kellene annak tartalmaznia. Ezért lehetséges, hogy a felhasználó a hierarchia megadásakor a fontos attribútumoknak csak egy szűk részhalmazát jelölte ki. Például előfordulhat, hogy a *cím* hierarchiájának szempontjából releváns attribútumok közül csak az *utcát* és a *várost* adták meg. Hogy az ilyen, részben megadott hierarchiák esetét is kezelni tudjuk, fontos, hogy az adatbázis sémájába az adatok szemantikáját is beágyazzuk azzal a céllal, hogy a szoros szemantikus kapcsolatban lévő attribútumok összeköthetők legyenek. Ily módon az egyik attribútum megadása egy egész sor, szemantikusan szorosan kapcsolódó attribútumnak a feladatba való bevonását eredményezheti, teljessé téve ezzel a hierarchiát. A felhasználóknak mindenesetre meg kell hagyni a lehetőséget, hogy szükség esetén ezt az opciót kiiktathassák.

3.7. példa | Tegyük fel, hogy egy adatbázisrendszer összeköti a *házszám*, *utca*, *város*, *megye_vagy_tartomány* és *ország* attribútumokat, mert a *cím* szempontjából szemantikailag szorosan kapcsolódnak egymáshoz. Amennyiben a *cím* leírására szolgáló hierarchia részeként a felhasználó csak a *város* attribútumot adná meg, a rendszer automatikusan bevonná a többi négy, szemantikailag kapcsolt attribútumot is a hierarchia létrehozásába. A felhasználó ezután dönthet úgy, hogy a *házszám* és *utca* attribútumokat elhagyja, és a *várost* tartja meg a fogalmak legalacsonyabb szintjén. ■

3.6. | Összefoglalás

- Az **adatok előfeldolgozása** mind az adattárházak, mind pedig az adatbányászat szempontjából fontos kérdés, mivel a valós adatok gyakran hiányosak, zajosak és inkonzisztensek. Az adatok előfeldolgozása magában foglalja az adattisztítást, az adatok integrálását, transzformálását és redukálását.
- Az **adattisztítási** eljárások a hiányzó értékek megadására, a zajos adatok simítására, a szélsőséges értékek felderítésére és az inkonzisztenciák kiküszöbölésére szolgálnak.
- Az **adatok integrálása** során a különböző forrásból származó adatokat egyetlen koherens struktúrában egyesítik. A metaadatok használata, a korrelációs analízis, az ellentmondó adatok felderítése és a szemantikai heterogenitás feloldása elősegíti az adatok zökkenőmentes integrálását.
- Az **adatok transzformálására** szolgáló eljárások az adatokat a bányászat céljainak megfelelő formába öntik, például összevonják vagy normalizálják.
- Az **adatok redukálásának** eszközei, úgymint az adatkockába való összevonás, a dimenziócsökkentés, az adattömörítés, a számosságcsökkentés és a diszkretizáció az adatoknak csökkentett reprezentációját adja a lehető legkisebb információveszteség mellett.
- **A fogalmi hierarchiák automatikus generálása** numerikus adatok esetén a kosarazás, a hisztogramanalízis, a klaszterelemzés, az entrópia alapú diszkretizáció és a természetes partíció szerinti szegmentálás technikáját használja fel. Kategória típusú adatok esetében a hierarchiát alkotó attribútumok különböző értékeinek száma alapján lehetséges a fogalmi hierarchiák generálása.
- Bár az adatok előfeldolgozásának már számos módja ismert, az adatok előkészítése még mindig aktívan kutatott terület.

3.7. | Feladatok

3.1. Az adatok minőségét értékelhetjük pontosság, teljesség és konzisztencia alapján. Adjunk meg két további jellemzőt az adatok minőségének mérésére!

3.2. Valós adatok esetében gyakran előfordulnak olyan sorok, amelyekben egyes attribútumok értékei hiányoznak. Ismertessük e probléma kezelésének különféle módszereit!

3.3. Tegyük fel, hogy az elemzésre kiválasztott adatok tartalmazzák az *életkor* attribútumot. Ennek értékei a rendelkezésre álló sorokban (növekvő sorrendben): 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.
   (a) Simítsuk a fenti adatokat kosárátlagok szerint, 3 mélységű kosarakat használva! Illusztráljuk a lépéseket! Elemezzük a módszer eredményét!
   (b) Hogyan határoznánk meg a szélsőséges értékeket?
   (c) Milyen más adatsimítási módszereket ismerünk?

3.4. Ismertessük az adatok integrálásával kapcsolatos problémákat!

3.5. A 3.3. feladat *életkor* adatait használva válaszoljunk az alábbi kérdésekre:
   (a) Használjunk min-max normalizációt a „35" [0,1] intervallumba való leképezésére!
   (b) Standardizáljuk a „35" értéket, az *életkor* szórásának 12,94 évet választva!
   (c) Decimális skálázással normalizáljuk a „35" értéket!
   (d) Melyik módszert használnánk az adatok normalizálására? Miért?

3.6. Folyamatábrával szemléltessük az attribútumok részhalmazának kiválasztására szolgáló alábbi módszereket:
   (a) előrelépéses kiválasztás,
   (b) visszalépéses eliminációs,
   (c) az előző kettő kombinációja.

3.7. A 3.3. feladat *életkor* adatainak felhasználásával
   (a) Készítsünk azonos, „10-es" oszlopszélességű hisztogramot!
   (b) Szemléltessük példákon a következő mintavételezési technikákat: VNEVM, VEVM, klaszter-mintavételezés, rétegzett mintavételezés. Használjunk 5 elemű mintákat, és *fiatal*, *középkorú*, illetve *idős* rétegeket!

3.8. Vázoljunk fel algoritmusokat (pszeudokódban vagy kedvenc programozási nyelvünkön) a következő feladatokra:
   (a) fogalmi hierarchia automatikus generálása kategória típusú adatokból az *attribútumok különböző értékeinek száma* alapján;
   (b) fogalmi hierarchia automatikus generálása numerikus adatokból az *azonos szélességű* particionálás módszerével.
   (c) fogalmi hierarchia automatikus generálása numerikus adatokból az *azonos elemszámú* particionálás módszerével.

3.8. | Irodalom

Az adatok előfeldolgozását számos átfogó munka tárgyalja, többek között Kennedy, Lee, Van Roy és mások [KLV+98], Weiss és Indurkhya [WI98], valamint Pyle [Pyl99]. Az egyes előfeldolgozási módszerekre vonatkozó specifikus referenciákat az alábbiakban vesszük sorra.

Az adatok minőségével kapcsolatban lásd Redman [Red92], Wang, Storey és Firth [WSF95], Wand és Wang [WW96], illetve Ballou és Tayi [BT99] munkáit. A hiányzó attribútumértékek kezelését tárgyalja Friedman [Fri77], Breiman, Friedman, Olshen és Stone [BFOS84], továbbá Quinlan [Qui89]. Egy kézírásos karakter adatbázis szélsőséges

mintáinak megtalálására mutat be egy módszert Guyon, Matic és Vapnik [GMV96]. A kosarazás és az adatok normalizálásának témáját számos mű tárgyalja, többek között [KLV+98, WI98, Pyl99]. Attribútumok konstrukcióját is használó rendszer többek között Langley, Simon, Bradshaw és Zytkow BACON-je [LSBZ87], Schlimmer Stagger-je [Schl87], Pagallo FRINGE programja [Pag89], valamint a Bloedorn és Michalski által írt AQ17-DCI [BM98]. Az attribútumok konstrukcióját szintén tárgyalja Liu és Motoda [LM98a, LM98b].

Az adatredukálási módszerek jó áttekintését adják Barbará és mások [BDF+97]. Az adatkockák kiszámításának és kezelésének algoritmusaival kapcsolatban lásd az [SS94, AAD+96, HRU96, RS97, ZDN97] munkákat. Az attribútumok részhalmazának kiválasztását számos szerző tárgyalja, többek között Neter, Kutner, Nachtsheim és Wasserman [NKNW96], Dash és Liu [DL97], továbbá Liu és Motoda [LM98, LM98b]. Az előreléptéses kiválasztás és visszalépéses eliminálás egy kombinált modelljét javasolta Siedlecki és Sklansky [SS88]. Egy attribútumok kiválasztására használt becsomagoló stratégiát tárgyal Kohavi és John [KJ97]. Felügyelet nélküli attribútumkiválasztásról ír Dash, Liu és Yao [DLY97]. A waveletek adattömörítésre való használatáról lásd Press, Teukolosky, Vetterling és Flannery munkáját [PTVF96]. A waveletek általános tárgyalását adja Hubbard [Hub96]. A wavelet-programcsomagokról közöl egy listát Bruce, Donoho és Gao [BDG96]. A Daubechies-transzformációt tárgyalja Daubechies [Dau92]. Press és mások könyve [PTVF96] tartalmaz egy bevezetést a főkomponens-analízisben használt szinguláris érték felbontásról. PCA rutinok a legtöbb statisztikai programcsomagban szerepelnek, így például az SAS-ben is *(http://www.sas.com/SASHome.html)*.

A regressziós és log-lineáris modellek témkörébe nyújt bevezetést számos átfogó mű, köztük [Jam85, Dob90, JW92, Dev95, NKNW96]. A log-lineáris modellekről (a számítástudományban *multiplikatív modellekként* említik) lásd Pearl munkáját [Pea88]. A hisztogramok elméletébe való általános bevezetést adnak Barbará és mások [BDF+97], illetve Devore és Peck [DP97]. Az egy attribútumot reprezentáló hisztogramok többre való kiterjesztéséről lásd Muralikrishna és DeWitt [MD88], valamint Poosala és Ioannidis [PI97]. A klaszteralgoritmusokra vonatkozólag számos referenciát találunk e könyv 8. fejezetében, ami teljes egészében ezzel a témával foglalkozik. A többdimenziós indexstruktúrák áttekintését adja Gaede és Günther [GG98]. Az adatok összevonására használt többdimenziós index algoritmusokat tárgyalja Aoki [Aok98]. Az indexfák között találjuk az R-fákat (Guttman [Gut84]), a négyesfákat (Finkel és Bentley [FB74]) és variációikat. A mintavételezés és adatbányászat témájában lásd Kivinen és Mannila [KM94], illetve John és Langley [JL96] munkáit.

A Kerber által készített ChiMerge [Ker92], valamint a Liu és Setiono által írt Chi2 [LS95] algoritmusok a χ^2 eloszlást használják a numerikus attribútumok automatikus diszkretizációjára. Fayyad és Irani [FI93] a minimális leírási hossz elvét használja a numerikus diszkretizációban használt intervallumok számának meghatározására. Catlett munkája [Cat91] a D-2 rendszert alkalmazza egy numerikus attribútum rekurzív binarizálására. A C4.5 algoritmust használó entrópia alapú diszkretizációt tárgyalja Quinlan [Qui93]. A fogalmi hierarchiákkal és azok kategória típusú adatokból való generálásával foglalkozik Han és Fu [HF94].

4. FEJEZET

Adatbányászó primitívek, nyelvek és rendszerarchitektúrák

Az adatbányászatot illetően gyakori az a téves elvárás, hogy az adatbányászati rendszerek *önállóan*, emberi beavatkozás és irányítás nélkül kiásnak *minden* értékelhető tudást, amely egy adott nagy adatbázisba van beágyazva. Jóllehet első hallásra vonzónak tűnhet egy ilyen önálló rendszer megléte, a gyakorlatban azonban az ilyen rendszerek a mintázatok túl nagy halmazát fednék fel, és a kialakított mintázatok teljes halmaza könnyedén meghaladhatná az adott adatbázis teljes méretét is! Ha engednénk az adatbányászati rendszereket a mintázatok felfedezése során „lazán futni" anélkül, hogy kijelölnénk az adatbázisnak azt a részét, amelyet a felhasználó vizsgálni kíván, vagy megadnánk a keresett mintázat típusát, az ahhoz vezetne, hogy kiszabadítanánk az adatbányászat „szörnyetegét". Az így felfedezett mintázatok nagy része a vizsgálat szempontjából lényegtelen lenne a felhasználó számára. Továbbá számos megtalált mintázat, jóllehet kapcsolódik az elemző folyamathoz, mégis nehézkesen értelmezhető, vagy az érvényesség, az újszerűség és a használhatóság hiányában érdektelenné válik. Így az adatbázisból felfedezhető összes mintázat generálása, tárolása és megjelenítése nem is valósítható meg, és nem is kívánatos tevékenység.

Egy megvalósíthatóbb forgatókönyv szerint elvárható, hogy a felhasználó a hatékony és gyümölcsöző tudásfeltárás elősegítése céljából kifejlesztett, úgynevezett *adatbányászó primitívek* halmazát használva kommunikálhasson az adatbányászati rendszerrel. Az ilyen primitívek tartalmazzák az adatbázis egy részének, illetve a felhasználót érdeklő adathalmazának meghatározását (tartalmazva az adatbázis érdekes attribútumait vagy az érdekes adattárház-dimenziókat), a bányászandó tudás típusát, a felfedezés folyamatának irányítása során hasznos háttértudást, a mintázat kiértékelésére vonatkozó érdekességi mértékeket és azt, hogy a felfedezett tudást hogyan kell ábrázolni. Ezek a primitívek a felfedezés során *interaktív* kommunikációt biztosítanak az adatbányászati rendszerrel abból a célból, hogy a felhasználó különféle szempontokból, vagy mélységekben vizsgálhassa meg a találatokat, és irányíthassa a bányászat folyamatát.

Egy adatbányászati lekérdező nyelv megtervezhető oly módon, hogy magában foglalja ezeket a primitíveket, és ezáltal rugalmas interakciót biztosít az adatbányászati rendszerekkel. Ha pedig már van egy adatbányászati lekérdező nyelvünk, akkor ez megfelelő

alapot szolgáltathat a felhasználóbarát grafikus felület kialakítására. Ezenkívül, az ilyen adatbányászati rendszerek fejlesztéséhez fontos egy jól tervezett rendszerarchitektúra megléte is. Mindezek elősegítenék az adatbányászati rendszerek más információs rendszerekkel való kommunikációját, és beillesztésüket az általános információfeldolgozó környezetbe.

Ebben a fejezetben az adatbányászó primitívekről, az adatbányászati lekérdező nyelvek ilyen elven történő fejlesztéséről, és az adatbányászati rendszerek rendszerarchitektúrájáról olvashat a kedves olvasó.

4.1. | Adatbányászó primitívek: mit jelent az adatbányászati feladat?

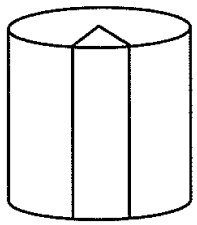
Minden felhasználó rendelkezik egyfajta elgondolással az **adatbányászati folyamatról**, ami az általa végrehajtani kívánt valamilyen adatanalízis. Az adatbányászati feladat **adatbányászati lekérdezés** formájában határozható meg, ami az adatbányászati rendszer bemenete. Egy adatbányászati lekérdezés a következő primitívek termjeiként definiálható:

- **A feladat szempontjából fontos adatok** – Az adatbázisnak az a része, amelyet megvizsgálunk. Tegyük fel például, hogy az *ElektroMind* vállalat Egyesült Államokba és Kanadába irányuló termékeladásainak felelős vezetőjeként az ügyfelek kanadai vásárlásainak tendenciáit szeretné tanulmányozni. A teljes adatbázis bányászata helyett meghatározhatjuk, hogy csupán az ügyfelek kanadai vásárlásaival és az ezekhez kapcsolódó ügyfelekre jellemző információkkal kapcsolatos adatok kinyerése szükséges. Azt is meghatározhatjuk, hogy a bányászat folyamán mely érdeklődésre számot tartó attribútumokat vegyünk figyelembe. Ezeket hívjuk **fontos attribútumoknak**.[1] Például ha csak az eladott termékek, valamint a vásárlók jövedelme és életkora közti lehetséges kapcsolat érdekel, akkor az *árucikk_név*, valamint a *vásárló_jövedelme* és *életkora* attribútumok határozhatók meg fontos attribútumokként a bányászat számára.
- **A bányászandó tudás típusa** – Meghatározza a végrehajtandó *adatbányászó függvényeket*, például karakterizáció, diszkrimináció, társítás, osztályozás, klaszterezés vagy fejlődésanalízis. Ha a kanadai vásárlók vásárlási szokásait tanulmányozzuk, akkor választhatjuk a vásárlók jellemzői és az általuk szívesen vásárolt termékek közti társítások bányászatát.
- **Háttértudás** – A felhasználó meghatározhat háttértudást vagy ismeretet a bányászandó területre vonatkozóan. Ez az ismeret hasznos a tudásfeltárás folyamatának irányítása és a megtalált mintázatok kiértékelése során. A háttértudásnak számos fajtája létezik. Ebben a fejezetben a hangsúlyt a háttértudás népszerű formáira, a *fogalmi hierarchiákra* helyezzük. A fogalmi hierarchiák hasznosak, mivel lehetővé teszik az adatok több absztrakciós szinten történő bányászatát. Más példák felhasználói előítéletekkel rendelkeznek az adatok közötti kapcsolatra vonatkozóan. Ezeket az előítéleteket a felfedezett mintázatok váratlansági fokának (ahol a váratlan mintázatok tűnnek érde-

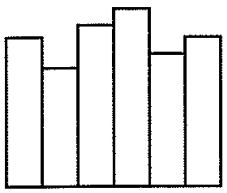
1 | Ha a bányászat többdimenziós adatkockákon történik, akkor a felhasználó fontos dimenziókat határozhat meg.

kesnek), vagy várhatósági fokának (ahol azokat a mintázatokat tekintik érdekesnek, amelyek megerősítik a felhasználói feltevéseket) kiértékelése során lehet használni.

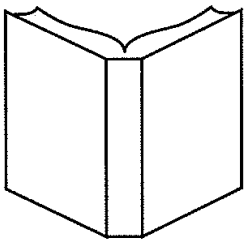
- **Érdekességi mértékek** – Ezek a függvények használatosak az érdektelen mintázatoknak a tudástól való elkülönítésére. Továbbá használhatók még a bányászat folyamatának irányítására, illetve a feltárás után a felfedezett mintázatok kiértékelésére is. A különféle tudásfajták különféle érdekességi mértékekkel rendelkezhetnek. Például a társító szabályok érdekességi mértéke tartalmazza a **megalapozottságot** (azok a feladat szempontjából fontos adatsorok, amelyeknél a szabály mintázata megjelenik, százalékban kifejez-



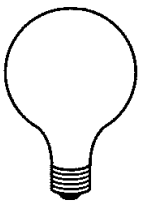
A feladat szempontjából fontos adatok
Az adatbázis vagy az adattárház neve
Az adatbázis táblái vagy az adattárház kockái
Az adatok kiválogatásának feltételei
Fontos attribútumok vagy dimenziók
Adatcsoportosító kritériumok



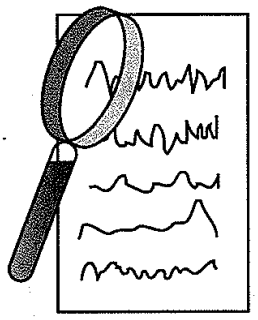
A bányászandó tudás típusa
Karakterizáció
Diszkrimináció
Társítás
Osztályozás/előrejelzés
Klaszterezés



Háttértudás
Fogalmi hierarchiák
Az adatok közötti kapcsolatokra vonatkozó felhasználói előítéletek



A mintázat érdekességi mértékei
Egyszerűség
Bizonyosság (például megbízhatóság)
Hasznosság (például megalapozottság)
Újszerűség



A felfedezett mintázat megjelenítése
Szabályok, táblák, jelentések, diagramok, grafikonok, döntési fák és kockák
Lefúrás, felgörgetés

4.1. ábra | Primitívek az adatbányászati feladat meghatározására

ve), és a **megbízhatóságot** vagy **konfidenciát** (a szabály alkalmazásának erősségére vonatkozó becslés). Azok a szabályok, amelyeknek megalapozottsága és megbízhatósága a felhasználó által megadott küszöb alatt található, érdektelennek minősülnek.

- **A felfedezett mintázatok bemutatása és ábrázolása** – A felfedezett mintázatok megjelenítési formájára vonatkoznak. A felhasználó a tudásreprezentáció különféle formáiból választhat, például szabályok, táblázatok, diagramok, grafikonok, döntési fák és kockák.

A következőkben ezeket a primitíveket vizsgáljuk meg részletesebben. A primitívek leírását a 4.1. ábra foglalja össze.

4.1.1. | A feladat szempontjából fontos adatok

Az első primitív azoknak az adatoknak a meghatározásával kapcsolatos, amelyeken a bányászatot végrehajtjuk. A felhasználót általában csak az adatbázis egy részhalmaza érdekli. Az egész adatbázis válogatás nélkül történő bányászata nem praktikus, különösen azért, mert a létrejött mintázatok száma az adatbázis méretével exponenciálisan növekedhet. Továbbá számos megtalált mintázat lényegtelen lehet a felhasználó szempontjából.

A relációs adatbázisban a feladat szempontjából fontos adatokat a kiválsztást, a vetítést, az összekapcsolást, az aggregációt és az ezekhez hasonló operátorokat tartalmazó relációs lekérdezések segítségével lehet összegyűjteni. Ezt az adatgyűjtést tekinthetjük az adatbányászati feladat „alfeladatának". Az adatgyűjtés folyamatának eredménye egy új adatrelációban jelenik meg, amelyet **kiindulási adatrelációnak** hívunk. A kiindulási adatrelációban az adatokat a lekérdezésben meghatározott feltételek szerint rendezhetjük vagy csoportosíthatjuk. Az adatokat az adatbányászati elemzés előtt megtisztíthatjuk, transzformálhatjuk (például bizonyos attribútumok aggregációja). A kiindulási reláció nem minden esetben felel meg az adatbázis fizikai relációinak. Mivel a virtuális relációkat az adatbázis mezőin alkotott **nézeteknek** hívjuk, ezért az adatbányászat szempontjából fontos adatokat **bányászható nézeteknek** nevezzük.

4.1. példa | Ha az adatbányászati feladat azt vizsgálja, hogy a Kanadában gyakran vásárolt *ElektroMind* termékek között milyen kapcsolat áll fenn, akkor a feladat szempontjából fontos adatok a következő információkkal határozhatók meg:

- A felhasznált *adatbázis* vagy *adattárház* neve (például *ElektroMind_DB*).
- A fontos adatokat tartalmazó *adattáblák* vagy *adatkockák* neve (például *árucikk*, *vásárló*, *vásárlás* és *eladások*).
- A fontos adatok kiválogatásának *feltételei* (például a folyó évi kanadai vásárlások adatainak visszakeresése).
- A *fontos attribútumok* vagy *dimenziók* (például *név* és *ár* az *árucikk* táblából, *jövedelem* és *életkor* a *vásárló* táblából).

Továbbá a felhasználó meghatározhatja, hogy a visszakapott adatok bizonyos attribútumok alapján, így például **group by** *dátum*, legyenek csoportosítva. Megadva ezeket az információkat, a feladat szempontjából fontos adatokat egy SQL-lekérdezéssel kaphatjuk vissza. ■

Az adattárházban az adatok jellemzően az **adatkocka** néven ismert többdimenziós adatbázisban vannak tárolva. Az adatkocka többdimenziós tömbstruktúraként, relációs szerkezetként, vagy a kettő kombinációjaként szerkeszthető meg, mint ahogy azt a 2. fejezetben már ismertettük. A feladat szempontjából fontos adatok halmaza feltétel alapú adatszűréssel, az adatkocka *szeletelésével* (az adatok kinyerése megadott attribútumérték vagy „szelet" alapján), vagy az adatkocka *kockázásával* (szeletek metszetének meghatározása) határozható meg.

Megjegyzendő, hogy az adatbányászati lekérdezésben az adatleválogatásról gondoskodó feltételek az adatok adatbázisbeli vagy adattárházbeli szintjénél magasabb koncepcionális szinten is megadhatók. Például a felhasználó az *ElektroMind* árucikkein a *típus* = "*szórakoztató elektronika*" fogalommal is meghatározhatja a szűrést, még akkor is, ha az adatbázisban az egyes árucikkek nem tárolhatók típus alapján, hanem inkább egy alacsonyabb rendű fogalom, például "*TV*", "*CD-lejátszó*" vagy "*videó*" szerint. Az *árucikkek* fogalmi hierarchiája, ami meghatározza, hogy a "*szórakoztató elektronika*" magasabb fogalmi szint, amely az alacsonyabb szintű {"*TV*", "*CD-lejátszó*", "*videó*"} fogalmakból áll, felhasználható a feladat szempontjából fontos adatok összegyűjtéséhez.

A fontos attribútumok vagy dimenziók meghatározása nehéz feladat lehet a felhasználó számára. Előfordulhat, hogy a felhasználónak csak egy durva elképzelése van arra vonatkozóan, hogy mely attribútumok lehetnek érdekesek a felfedezés szempontjából. Ezenkívül a bányászandó adatok meghatározásakor a felhasználó elfeledkezhet további, az előzőekkel szoros szemantikai kapcsolatban lévő fontos adatokról. Például bizonyos termékek eladásai közelebbi kapcsolatban állhatnak különleges eseményekkel (például karácsony), vagy a vásárlók bizonyos csoportjaival, mégis előfordulhat, hogy ezek a tényezők nem részei az általános adatelemző kérésnek. Az ilyen esetekben olyan mechanizmusok használhatók, amelyek a feladat szempontjából fontos adatokról segítenek sokkal precízebb meghatározást adni. Ezek az eljárások tartalmaznak beépített függvényeket, amelyek az attribútumok fontosságát értékelik ki, és besorolják őket a meghatározott műveletnek megfelelően. Ezenkívül, a szoros szemantikai kötésben lévő attribútumokat kereső technikák felhasználhatók a felhasználó által meghatározott kiindulási adathalmaz növelésére is.

4.1.2. | A bányászandó tudás típusa

Fontos meghatározni a bányászandó tudás típusát, mivel ez határozza meg a megvalósítandó adatbányászó függvényt. A tudás típusai a következők lehetnek: fogalomleírás (karakterizáció és diszkrimináció), társítás, osztályozás, előrejelzés, klaszterezés és fejlődésanalízis.

Továbbá, meghatározva az adott adatbányászati feladat során bányászandó tudás típusát, a felhasználó sokkal inkább domináns szerepet tölthet be, és ezáltal gondoskodhat a mintázat sablonjáról, amire minden megtalált tudásnak illeszkednie kell. Ezek a sablonok vagy **metamintázatok** (hívják még **metaszabályoknak** vagy **metalekérdezéseknek**) használhatók a felfedezés folyamatának irányítására. A metamintázatok használatát a következő példa illusztrálja.

4.2. példa | A felhasználó az *ElektroMind* ügyfeleinek vásárlási szokásait tanulmányozva a következő formájú társítási szabályok bányászatát választja:

$$P(X : \textit{vásárló}, W) \wedge Q(X, Y) \Rightarrow \textit{vesz}(X, Z),$$

ahol X a *vásárló* reláció kulcsa; P és Q **predikátumváltozók**, amelyek a feladat szempontjából fontos adatok részeként meghatározott fontos attribútumokkal vagy dimenziókkal helyettesíthetők, W, Y és Z pedig **objektumváltozók**, amelyek az X vásárló megfelelő predikátumainak értékeit vehetik fel.

Az erre a metamintázatra illeszkedő társító szabályok keresésének eredményei lehetnek például a következő szabályok:

$$\textit{életkor}(X, \text{"30...39"}) \wedge \textit{jövedelem}(X, \text{"40 000...49 000"}) \Rightarrow \textit{vesz}(X, \text{"videó"}) \quad [2{,}2\%, 60\%] \qquad (4.1)$$

és

$$\textit{foglalkozás}(X, \text{"hallgató"}) \wedge \textit{életkor}(X, \text{"20...29"}) \Rightarrow \textit{vesz}(X, \text{"számítógép"}) \quad [1{,}4\%, 70\%] \qquad (4.2)$$

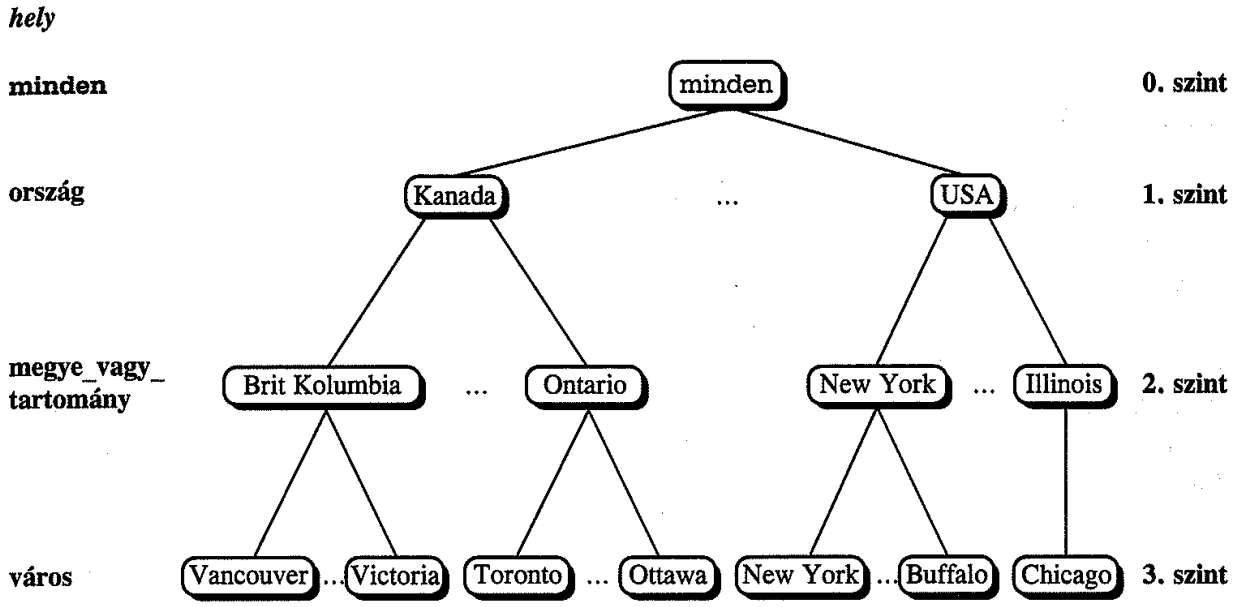
Az előbbi szabály azt állítja, hogy a harmincas éveiben járó vásárlók, akiknek az éves jövedelmük 40 000 és 49 000 közé esik, valószínűleg (60%-os megbízhatósággal) videót fognak vásárolni, és ezek a vásárlások az összes tranzakciók számának körülbelül 2,2%-át teszik ki. Az utóbbi szabály azt állítja, hogy húszas éveikben járó diákok valószínűleg (70%-os megbízhatósággal) számítógépet fognak vásárolni, és ezek a vásárlások az összes tranzakciók számának mintegy 1,4%-át képviselik. ■

4.1.3. | Háttértudás: fogalmi hierarchiák

A háttértudás olyan információ a bányászandó területre vonatkozóan, amely hasznos lehet a feltárás folyamata során. A fejezet során a hangsúlyt csupán a háttértudás egyik egyszerű, de mégis nagy erejű formájára, a *fogalmi hierarchiákra* helyezzük. A fogalmi hierarchiák az ismeret felfedezését az absztrakció több szintjén is lehetővé teszik.

Mint ahogy a 2. fejezetben már ismertettük, a **fogalmi hierarchia** alacsony szintű fogalmak halmazának magasabb szintre, általánosabb fogalmakra történő leképezéseinek sorozatát határozza meg. A 4.2. ábra a *hely* dimenzió fogalmi hierarchiáját ábrázolja, az alacsonyabb szintű fogalmakat (például városok) a magasabb szintű fogalmakra (például országok) képezve le.

Megjegyezzük, hogy ez a fogalmi hierarchia fába rendezett **csomópontok** halmazaként került ábrázolásra, ahol minden csomópont önmagában egy fogalmat képvisel. A **minden** speciális csomópont, amely a fa gyökerének van lefoglalva. Ez jelöli az adott dimenzió legáltalánosabb értékét. Ha ez nincs egyértelműen jelölve, akkor is odaértendő. A fenti fogalmi hierarchia **4 szintből** áll. A fogalmi hierarchia szintjeit szokás szerint fentről lefelé, a **minden** csomóponttól, 0-val kezdve számozzuk. A mi példánkban az 1. szint az *ország* fogalmat ábrázolja, míg a 2. és a 3. szint a *megye_vagy_tartomány*, illetve



4.2. ábra | A *hely* dimenzió fogalmi hierarchiája

a *város* fogalmakat képviselik. A hierarchia levelei a dimenzió adatsor értékeinek (**primitív szintű adatok**) felelnek meg. Ezek az adott attribútum vagy dimenzió legjobban meghatározott értékei vagy fogalmai. Jóllehet a fogalmi hierarchia gyakran fa formában határoz meg taxonómiát, de előfordulhat általános háló vagy parciális rendezés alakjában is.

A fogalmi hierarchiák azáltal, hogy a nyers adatok magasabb, általánosabb fogalmi szinten történő kezelését biztosítják, a háttértudás hasznos formái. Az adatok általánosítása vagy **felgörgetése** a primitív szintű adatok (például a városnevek a *hely*nél vagy a számszerű adatok az *életkor*nál) magasabb szintű fogalmakra (például a kontinensek a *hely*nél vagy a 20...39, 40...59, 60+-hoz hasonló tartományok az *életkor*nál) történő cseréjével valósul meg. Ez lehetővé teszi a felhasználók számára, hogy az adatokat *lényegretörőbb* és kifejezőbb elvont fogalmakban szemléljék, és jobban érthetővé teszi a felfedezett mintázatokat. Az általánosítás járulékos előnye az adatok tömörítése. A tömörített adatokon történő bányászat kevesebb input/output műveletet igényel, és hatékonyabb, mint a nagyobb, tömörítetlen adathalmazon történő adatbányászat.

Ha az eredményadatok túlságosan általánosítottnak tűnnek, a fogalmi hierarchiák lehetővé teszik a specializációt vagy **lefúrást** is, melyek során a fogalmi értékek alacsonyabb fogalmakra cserélődnek ki. A felgörgetés és lefúrás által a felhasználó az adatokat különféle perspektívákból szemlélheti, s ezáltal további bepillantást nyer a rejtett adatkapcsolatokba.

A fogalmi hierarchiákat a rendszer felhasználói, a terület szakértői és a tudásmérnökök szolgáltathatják. A leképzések jellemzően adat- vagy alkalmazásspecifikusak. A fogalmi hierarchiák az adatok eloszlásának statisztikai elemzése alapján gyakran automatikusan felfedezhetők, vagy akár dinamikusan frissíthetők. A fogalmi hierarchiák automatikus generálását részletesebben a 3. fejezet tárgyalja.

Lehetséges, hogy egy adott attribútumhoz vagy dimenzióhoz a különböző felhasználói nézőpontok alapján több fogalmi hierarchia is létezik. Például az *ElektroMind* regionális üzletvezetője, aki a vásárlók különböző helyeken történő vásárlási szokásainak tanulmá-

nyozásában érdekelt, a 4.2. ábra fogalmi hierarchiáját részesítheti előnyben. Ugyanakkor a kereskedelmi igazgató a reklámok terjesztésének elősegítése érdekében a *hely* nyelvi szempontból történő rendezését részesítheti előnyben.

A fogalmi hierarchiáknak négy fő típusa létezik. A 2. fejezet már bevezette a leggyakoribb típusokat, vagyis a *sémahierarchiákat* és *halmazcsoportosító hierarchiákat*, amiket itt is áttekintünk. Ezenkívül tanulmányozzuk még a *műveletvezérelt hierarchiákat* és *szabály alapú hierarchiákat* is.

Sémahierarchiák – A sémahierarchia (vagy szigorúbban véve a sémadefiníciós hierarchia) az adatbázisséma attribútumainak teljes vagy részleges rendezése. A sémahierarchiák attribútumok közötti szemantikai kapcsolatokat fejezhetnek ki formálisan. Az adattárház dimenziót jellegzetesen a sémahierarchia határozza meg.

4.3. példa | Adott egy reláció sémája, amely a *cím* szempontjából a következő attribútumokat tartalmazza: *utca*, *város*, *megye_vagy_tartomány* és *ország*. A *hely* sémahierarchiát a következő teljes rendezéssel határozhatjuk meg:

utca < *város* < *megye_vagy_tartomány* < *ország*

Ez azt jelenti, hogy az *utca* koncepcionálisan alacsonyabb szinten helyezkedik el, mint a *város*, amely alacsonyabb szinten van, mint a *megye_vagy_tartomány*, ami pedig az *országnál* alacsonyabb szinten van. A sémahierarchia metaadat-információt, vagyis az adatokról szóló adatokat szolgáltat. Ez a meghatározás, amely az attribútumok teljes vagy részleges rendezésének formájában van megadva, sokkal tömörebb, mint egy vele egyenértékű definíció, ami az utcák, megyék vagy tartományok és országok összes előfordulását sorolja fel. ■

Halmazcsoportosító hierarchiák – A halmazcsoportosító hierarchia egy adott attribútum vagy dimenzió értékeit konstansok vagy értéktartományok csoportjaiba rendezi. A csoportok közt teljes vagy részleges rendezés határozható meg. A halmazcsoportosító hierarchiák a sémadefiníciós hierarchiák finomítására, gazdagítására használhatók úgy, hogy a hierarchia két típusát kombináljuk. Ezeket jellemzően az objektumkapcsolatok kis halmazainak meghatározására használjuk.

4.4. példa | Az *életkor* attribútum halmazcsoportosító hierarchiája tartományok által kialakított termekben a következőképpen határozható meg:

{*fiatal*, *középkorú*, *idős*} ⊂ **minden**(*életkor*)
{20...39} ⊂ *fiatal*
{40...59} ⊂ *középkorú*
{60...89} ⊂ *idős*

Megjegyezzük, hogy hasonló tartománymeghatározások a 3. fejezetben részletezett módon automatikusan is generálhatók. ■

Műveletvezérelt hierarchiák – A *műveletvezérelt hierarchia* a felhasználók, a szakértők vagy az adatbányászati rendszer által meghatározott műveleteken alapul. A műveletek tartalmazhatják a kódolt információsorozatok dekódolását, az összetett adatobjektumokból való információkinyerést és az adatok klaszterezését.

4.5. példa | Egy e-mail- vagy egy URL-cím az országokkal, az egyetemekkel (vagy cégekkel) és tanszékekkel kapcsolatos hierarchikus információkat tartalmazhat. A fogalmi hierarchiák kialakítása céljából az ilyen jellegű információk kinyerésére dekódoló műveletek határozhatók meg.

Például a *dmbook@cs.sfu.ca* e-mail-cím az e-mail-címek fogalmi hierarchiáját kialakítva a *login név* < *tanszék* < *egyetem* < *ország* parciális rendezést adja. Hasonlóan a *http://www.cs.sfu.ca/research/DB/DBMiner* URL-cím dekódolható úgy, hogy parciális rendezést adjon, ami az URL-címek fogalmi hierarchiájának alapját jelenti. ■

Ezenkívül matematikai és statisztikai műveletek, például adatklaszterezés és adateloszlás-elemző algoritmusok is használhatók fogalmi hierarchiák megalkotására, mint ahogy azt a 3.5. alfejezetben már bemutattuk.

Szabály alapú hierarchiák – Szabály alapú hierarchia akkor fordul elő, ha a teljes fogalmi hierarchia vagy annak egy része szabályok halmazában van meghatározva, és dinamikusan értékelődik ki az aktuális adatbázis adat- és szabálydefiníciója alapján.

4.6. példa | A következő szabályok az *ElektroMind* termékeinek osztályozására használatosak: *alacsony_profittartalmú*, *közepes_profittartalmú* és *magas_profittartalmú* termékekre, ahol az *X* termék profittartalmát a termék kiskereskedelmi árának és a tényleges költségének különbsége határozza meg. Azok a termékek, amelyek profittatalma kevesebb, mint 50 $, *alacsony_profittartalmú* termékként határozhatók meg, az 50 $ és 250 $ közötti profitot hozók a *közepes_profittartalmú* termékek, és azok, amelyek több mint 250 $ profitot termelnek, *magas_profittartalmú* termékként definiálhatók.

$$alacsony\_profittartalmú(X) \Leftarrow ár(X, P1) \wedge költség(X, P2) \wedge ((P1 - P2) < 50\ \$)$$
$$közepes\_profittartalmú(X) \Leftarrow ár(X, P1) \wedge költség(X, P2) \wedge ((P1 - P2) \geq 50\ \$) \wedge$$
$$\wedge ((P1 - P2) \leq 250\ \$)$$
$$magas\_profittartalmú(X) \Leftarrow ár(X, P1) \wedge költség(X, P2) \wedge ((P1 - P2) > 250\ \$)$$ ■

A fogalmi hierarchiák adatbányászatban történő felhasználásait a könyv későbbi fejezetei ismertetik.

4.1.4. | Érdekességi mértékek

Jóllehet a feladat szempontjából fontos adatok és a bányászandó tudás típusának meghatározása (például karakterizáció, társítás stb.) lényegileg csökkentheti a kialakított mintázatok számát, az adatbányászati folyamat azonban még mindig a mintázatok nagy számát állíthatja elő. Az adott felhasználó számára jellemzően a mintázatoknak csak egy kis

része lesz valójában érdekes. Így a felhasználók szempontjából szükséges, hogy a folyamat által visszaadott érdektelen mintázatok számát tovább korlátozzák. Ez megvalósítható a mintázatok egyszerűségét, bizonyosságát, hasznosságát és újszerűségét megbecsülő érdekességi mértékek meghatározásával.

Ebben a fejezetben a mintázatok érdekességének néhány objektív mértékét tekintjük át. Az objektív mértékek a mintázatok struktúráján alapulnak és statisztikailag is alátámasztottak. Általában minden mértékhez társul egy, a felhasználó által ellenőrzött *küszöb*. Azok a szabályok, amelyek nem érik el a küszöbértéket, érdektelenek, és így nem kerülnek tudásként bemutatásra a felhasználó számára.

- **Egyszerűség** – A mintázat szubjektív szempontból tekintett általános egyszerűsége a mintázat érdekességének csak egy összetevő tényezője. A mintázat egyszerűségének objektív mértéke tekinthető a mintázat szerkezeti függvényének. Ez a függvény meghatározható a mintázat bitekben mért méretének vagy a mintázatban megjelenő attribútumok, operátorok számának kifejezéseként. Például, minél összetettebb a szabály szerkezete, annál nehezebb értelmezni, és így valószínűleg kevésbé érdekes.

  A **szabály hossza** például egy egyszerűségi mérték. A konjunktív normálformában (azaz konjunktív predikátumok halmazaként) kifejezett szabályok hossza a szabályban előforduló konjunkciók számaként határozható meg. Azok a társítások, diszkriminációk vagy osztályozások, amelyek hossza meghalad egy, a felhasználó által meghatározott küszöböt, érdektelennek tekinthetők. A döntési fákban kifejezett mintázatok egyszerűsége a fa levelei vagy csomópontjai számának függvénye lehet.
- **Bizonyosság** – Minden felfedezett mintázatnak rendelkeznie kell bizonyossági mértékkel, amely kiértékeli a mintázat érvényességét vagy „megbízhatóságát". Az „$A \Rightarrow B$" alakú szabályok – ahol A és B elemek halmazai – bizonyossági mértéke a **megbízhatóság (konfidencia)**. Ha adott a feladat szempontjából fontos adatsorok halmaza (vagy tranzakciók halmaza egy tranzakciós adatbázisban) az „$A \Rightarrow B$" alakú szabály megbízhatósága a következőképpen határozható meg:

$$megbízhatóság(A \Rightarrow B) = \frac{\#\_A\text{-}t\_és\_B\text{-}t\_tartalmazó\_sorok}{\#\_A\text{-}t\_tartalmazó\_sorok} \tag{4.3}$$

4.7. példa | Tegyük fel, hogy a feladat szempontjából fontos adatok az *ElektroMind* számítástechnikai részlegén történt termékek eladásainak tranzakcióit tartalmazó adatokból állnak. A

$$vesz(X, \text{"számítógép"}) \Rightarrow vesz(X, \text{"szoftver"}) \tag{4.4}$$

szabály 85%-os megbízhatósága azt jelenti, hogy a számítógépet vásárolt vevők 85%-a egyúttal szoftvert is vásárolt. ■

A 100%-os vagy 1-es megbízhatósági érték azt jelenti, hogy a szabály mindig igaz a vizsgált adatokon. Ezeket a szabályokat **precíz (exact)** szabályoknak hívjuk.

A (4.3) egyenlet könnyen alkalmazható az osztályozási szabályok bizonyossági mértékének maghatározására is, amelyet **megbízhatóságnak** vagy **pontosságnak** neve-

zünk. Az osztályozási szabályok egy modellt javasolnak a célosztálybeli objektumok vagy sorok (mondjuk *költekezőVásárlók*) más, tőle különböző osztályokhoz tartozó objektumoktól (mondjuk *takarékosVásárlók*) való megkülönböztetésére. A kis megbízhatósági érték azt jelenti, hogy a kérdéses szabály a célosztálytól eltérő osztályhoz tartozó objektumok nagy számát helytelenül osztályozza célosztálybeli objektumként. A szabály megbízhatósága a **szabály erősségeként**, a **szabály minőségeként**, **bizonyossági faktorként** és **megkülönböztető súlyként** is ismert.

- **Hasznosság** – A mintázat felhasználhatósága az érdekességet meghatározó faktor. Értéke megbecsülhető hasznossági függvénnyel, például a megalapozottsággal. Egy társítási mintázat **megalapozottsága** a feladat szempontjából fontos adatsorok (vagy tranzakciók) azon százalékát adja meg, amelyekre igaz a mintázat. Az $A \Rightarrow B$" alakú társítási szabályokra, ahol A és B elemek halmazai, a megalapozottság a következőképpen határozható meg:

$$megalapozottság(A \Rightarrow B) = \frac{\#\_A\text{-}t\_és\_B\text{-}t\_tartalmazó\_sorok}{\#\_összes\_sorok} \tag{4.5}$$

4.8. példa | Tegyük fel, hogy a feladat szempontjából fontos adatok az *ElektroMind* számítástechnikai részlegének tranzakcióit tartalmazzák. A (4.4) szabály 30%-os megalapozottsága azt jelenti, hogy azoknak a vásárlóknak a 30%-a, akik a számítástechnikai részlegen vásároltak, számítógépet és szoftvert is vásároltak. ■

Azokat a társítási szabályokat, amelyek kielégítik a felhasználó által meghatározott *minimális megbízhatósági küszöböt* és *minimális megalapozottsági küszöböt*, **erős társítási szabályoknak** nevezzük, és ezeket a szabályokat tekintjük érdekes szabályoknak. Az alacsony megalapozottságú szabályok valószínűleg zajokat vagy kivételes eseteket ábrázolnak.

A (4.5) megalapozottsági egyenlőség számlálója a szabály **darabszámaként** is ismert. Meglehetősen gyakran ezt adják meg a megalapozottság helyett. A megalapozottság értéke ebből könnyen származtatható.

Az általánosított sorok lényegében karakterisztikus és diszkriminációs leírások. Bármely általánosított sor, amely a feladat szempontjából fontos soroknak kevesebb mint Y%-át képviseli, zajnak tekinthető. Az Y értékét **zajküszöbnek** nevezzük.

- **Újszerűség** – Azok a mintázatok tekinthetők újszerű mintázatoknak, amelyek a már adott mintázathalmazhoz új információt adnak vagy megnövelik a teljesítményt. Például egy kivételes adat abban tekinthető újszerűnek, hogy különbözik attól, ami statisztikai alapon, vagy a felhasználó hite szerint elvárható. Az újszerűség felfedezésének másik módszere, hogy a redundáns mintázatokat töröljük. Ha a tudásbázisban vagy a levezetett szabályhalmazban már meglévő másik szabály magában foglalhat egy felfedezett szabályt, akkor a lehetséges redundancia megszüntetésének céljából mind a két szabályt újra meg kell vizsgálni.

A fogalmi hierarchiák használatával történő bányászat nagy számú redundáns szabályt eredményezhet. Például tegyük fel, hogy az *ElektroMind* adatbázis bányászata, melynek során felhasználtuk a 4.2. ábra *hely*re vonatkozó hierarchiáját is, a következő szabályokat eredményezte:

$$hely(X, \text{"}Kanada\text{"}) \Rightarrow vesz(X, \text{"}SONY\_TV\text{"}) \quad [8\%, 70\%] \quad (4.6)$$

$$hely(X, \text{"}Montreal\text{"}) \Rightarrow vesz\ (X, \text{"}SONY\_TV\text{"}) \quad [2\%, 71\%] \quad (4.7)$$

Tegyük fel, hogy a (4.6) szabály 8%-os megalapozottságú és 70%-os megbízhatóságú. A (4.7) szabálytól elvárhatjuk, hogy a megbízhatósága szintén 70% körül legyen, mivel minden egyes sor, ami Montreal adatobjektumait ábrázolja, egyben Kanada adatobjektumait is tartalmazza. A (4.6) szabály általánosabb, mint a (4.7), és ezáltal elvárható, hogy az előbbi szabály gyakrabban forduljon elő, mint az utóbbi. Következésképpen a két szabály megalapozottságának nem kell megegyeznie. Tegyük fel, hogy a kanadai összes eladásoknak körülbelül egynegyede származik Montrealból. Így elvárhatjuk, hogy a Montrealt tartalmazó szabály megalapozottsága egynegyede legyen a Kanadát tartalmazó szabály megalapozottságának. Másként kifejezve azt várjuk, hogy a (4.7) szabály megalapozottsága 8% × ¼ = 2% legyen. Ha a (4.7) szabály aktuális megbízhatósága és megalapozottsága olyan, mint ahogy vártuk, akkor a szabály redundánsnak tekinthető, mivel nem nyújt semmilyen kiegészítő információt, és kevésbé általános, mint a (4.6) szabály. A 6. fejezetben ezeket az ötleteket tovább fogjuk tárgyalni.

Az adatbányászati rendszereknek biztosítaniuk kell a felhasználók számára, hogy rugalmasan és interaktív módon határozhassák meg, tesztelhessék és módosíthassák az érdekességi mértékeket és a hozzájuk tartozó küszöbértékeket. Az előbbiekben tanulmányozott egyes alapmértékeken kívül még számos más objektív mérték is létezik. Vannak szubjektív mértékek is, amelyek az objektív statisztikai mértékek mellett figyelembe veszik a felhasználók elképzeléseit is az adatok közti kapcsolatokra vonatkozóan. Az érdekességi mértékek a könyv többi fejezetében részletesebben kerülnek megvitatásra.

4.1.5. | A felfedezett mintázat bemutatása és ábrázolása

Hogyan „láthatjuk" a felfedezett mintázatot? Ahhoz, hogy hatékony legyen az adatbányászat, az adatbányászati rendszereknek képeseknek kell lenniük a felfedezett mintázat többfajta formában történő megjelenítésére is. Ezek a formák lehetnek például szabályok, táblázatok, kereszttáblák, torta- vagy oszlopdiagramok, döntési fák, kockák vagy más vizuális megjelenítési formák (4.3. ábra). A felfedezett minták többféle formában történő megjelenítésének lehetővé tétele különféle hátterekkel segítheti a felhasználókat abban, hogy azonosítsák az érdekes mintázatokat és beavatkozzanak vagy irányítsák a rendszert a további felfedezés során. A felhasználónak képesnek kell lennie a felfedezett mintázat megjelenítésére használt bemutatási forma kiválasztására.

A fogalmi hierarchiák használata jelentős szerepet játszik abban, hogy segítse a felhasználót a felfedezett mintázat megjelenítésében. A fogalmi hierarchiákkal történő bányászat lehetővé teszi a felfedezett tudás magas szintű fogalmakban történő bemutatását, amely sokkal érthetőbb lehet a felhasználók számára, mint a primitív (azaz nyers) adatok szintjén kifejezett szabályok, például funkcionális vagy többértékű függőségi szabályok,

Szabályok

életkor(X, "fiatal") és jövedelem(X, "magas") ⇒ osztály(X, "A")
életkor(X, "fiatal") és jövedelem(X, "alacsony") ⇒ osztály(X, "B")
életkor(X, "idős") ⇒ osztály(X, "C")

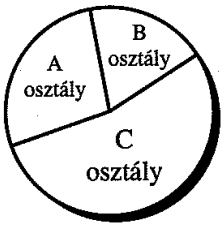
Tábla

életkor	jövedelem	osztály	darab
fiatal	magas	A	1402
fiatal	alacsony	B	1038
idős	magas	C	786
idős	alacsony	C	1374

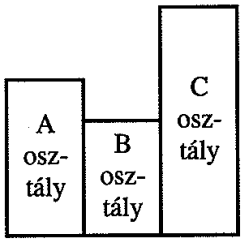
Kereszttábla

életkor	jövedelem		osztály		
	magas	alacsony	A	B	C
fiatal	1402	1038	1402	1038	0
idős	786	1374	0	0	2160
darab	2188	2412	1402	1038	2160

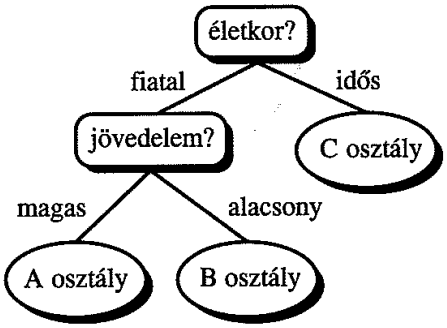
Tortadiagram



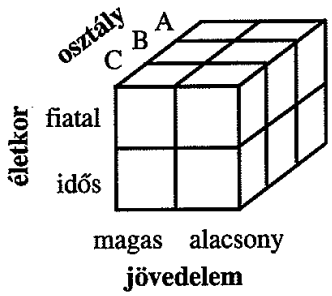
Oszlopdiagram



Döntési fa



Adatkocka



4.3. ábra | A felfedezett mintázat bemutatásának és ábrázolásának különféle formái

vagy integritási megszorítások. Továbbá az adatbányászati rendszereknek fogalmi hierarchiákat kell alkalmazniuk a lefúrás és a felgörgetés operátorok használatakor is, hogy ezáltal a felhasználók a felfedezett mintázatokat az absztrakció több szintjén is megvizsgálhassák. Ezenkívül, a pivotolás (vagy forgatás), a szeletelés és kockázás műveletei segítik a felhasználót abban, hogy az általánosított adatokat és tudást különböző nézőpontból vegye szemügyre. Ezeket az operátorokat részletesebben a 2. fejezet tárgyalta. Az adatbányászati rendszereknek ilyen interaktív műveleteket éppúgy biztosítaniuk kell minden dimenzión, mint ahogy minden dimenzió egyedi értékeire is.

A tudás egyes típusaira bizonyos megjelenítési formák jobban alkalmazhatók, mint mások. Például általánosított táblázatok és a nekik megfelelő kereszttáblák vagy torta-/oszlopdiagramok megfelelők a karakterisztikus leírások ábrázolására, míg az osztályozás megjelenítésére gyakran választanak döntési fákat. Olyan érdekességi mértékek, amelyeket például a 4.1.4. alfejezet ismertet, minden felfedezett mintázathoz megjeleníthetők azzal a céllal, hogy segítsék a felhasználókat a hasznos tudást ábrázoló mintázatok azonosításában.

4.2. | Az adatbányászati lekérdező nyelv

Miért fontos, hogy létezzen adatbányászati lekérdező nyelv? Jusson eszünkbe, hogy a rugalmas és hatékony tudásfeltárás megkönnyítésének érdekében az adatbányászati rendszerek elvárt tulajdonsága az *ad hoc és interaktív adatbányászat támogatása*. Az ilyen tulajdonság támogatására adatbányászati lekérdező nyelv fejleszthető ki.

A jó adatbányászati lekérdező nyelv fejlesztésének fontosságát a relációs adatbázisok történetének megfigyelésével is megérthetjük. A relációs adatbázisrendszerek évtizedekig uralták az adatbázispiacot. A relációs lekérdező nyelv szabványosítása, amely a relációs adatbázisok fejlesztésének korai szakaszában bekövetkezett, széles körben hozzájárult a relációs adatbázisok sikeréhez. Jóllehet mindegyik kereskedelmi relációs adatbázisrendszer saját grafikai felhasználói felülettel rendelkezik, mégis mindegyik interfész magját a szabványosított relációs lekérdező nyelv adja. A relációs lekérdező nyelvek szabványosítása alapot szolgáltatott a relációs rendszerek fejlesztéséhez és kialakításához. Ez könnyebbé tette az információk cseréjét, a technológia átvitelét, és elősegítette a relációs adatbázis-technológia üzleti megjelenését és széles körű elfogadását. Az adatbázisrendszerek területén folyó legújabb szabványosítási tevékenységek, például az SQL-3-mal kapcsolatos munkák és mások, tovább illusztrálják a szabványosított adatbázisnyelv meglétének fontosságát az adatbázisrendszerek fejlesztésének és üzleti sikereinek érdekében. Ezért egy jó adatbányászati lekérdező nyelv segítheti az adatbányászrendszer platformok fejlesztésének szabványosítását.

Átfogó adatbányászati nyelv fejlesztése kihívást jelent, mivel az adatbányászat a feladatok széles spektrumát fedi le, beleértve a mintázatkeresést a karakterizációtól a társítási szabályok bányászatáig, az adatosztályozásig és a fejlődésanalízisig. Mindegyik feladatnak különböző követelményei vannak. A hatékony adatbányászati nyelv megszerkesztése megköveteli a változatos fajtájú adatbányászati rendszerek erejének, korlátainak és a mechanizmus alapjainak megértését.

Hogyan szerkesszük meg az adatbányászati lekérdező nyelvet? A fejezet korábbi ré-

```
⟨DMQL⟩                  ::= ⟨DMQL_Kifejezés⟩; {⟨DMQL_Kifejezés⟩}
⟨DMQL_Kifejezés⟩        ::= ⟨Adatbányászó_Kifejezés⟩
                          | ⟨Fogalmi_Hierarchiát_Meghatározó_Kifejezés⟩
                          | ⟨Ábrázolás_és_Bemutatás⟩
⟨Adatbányászó_Kifejezés⟩        ::=
                          use database ⟨adatbázis_név⟩ | use data warehouse ⟨adattárház_név⟩
                          {use hierarchy ⟨hierarchia_név⟩ for ⟨attribútum_vagy_dimenzió⟩}
                          ⟨Bányászandó_Tudás_Meghatározása⟩
                          in relevance to ⟨attribútum_vagy_dimenzió_lista⟩
                          from ⟨tábla(ák)/kocka(ák) ⟩
                          [where ⟨feltétel⟩]
                          [order by ⟨rendezési_lista⟩]
                          [group by ⟨csoportosítási_lista⟩]
                          [having ⟨feltétel⟩]
                          {with [⟨érdekességi_mérték_név⟩] threshold = ⟨küszöb_név⟩
                          [for ⟨attribútum(ok) ⟩]}
⟨Bányászandó_Tudás_Meghatározása⟩ ::= ⟨Bányász_Karak⟩ | ⟨Bányász_Diszkr⟩ |
        ⟨Bányász_Társ⟩ | ⟨Bányász_Osztály⟩
⟨Bányász_Karak⟩ ::=     mine characteristics [ as ⟨mintázat_név⟩]
                        analyze ⟨mérték(ek) ⟩
⟨Bányász_Diszkr⟩ ::=    mine comparison [ as ⟨mintázat_név⟩]
                        for ⟨cél_osztály⟩ where ⟨cél_feltétel⟩
                        {versus ⟨különböző_osztály_i⟩ where ⟨különböző_feltétel_i⟩
                        analyze ⟨mérték(ek) ⟩
⟨Bányász_Társ⟩ ::=      mine associations [ as ⟨mintázat_név⟩]
                        [matching ⟨metamintázat⟩]
⟨Bányász_Osztály⟩       ::= mine classification [ as ⟨mintázat_név⟩]
                        analyze ⟨osztályozó_attribútum_vagy_dimenzió⟩
⟨Fogalmi_Hierarchiát_Meghatározó_Kifejezés⟩ ::=
                        define hierarchy ⟨hierarchia_név⟩
                        [for ⟨attribútum_vagy_dimenzió⟩]
                        on ⟨reláció_vagy_kocka_vagy_hierarchia⟩
                        as ⟨hierarchia_leírás⟩
                        [where ⟨feltétel⟩]
⟨Ábrázolás_és_Bemutatás⟩ ::=
                        display as ⟨eredmény_forma⟩ | {⟨Többszintű_Művelet⟩}
⟨Többszintű_Művelet⟩ ::= roll up on ⟨attribútum_vagy_dimenzió⟩
                        | drill down on ⟨attribútum_vagy_dimenzió⟩
                        | add ⟨attribútum_vagy_dimenzió⟩
                        | drop ⟨attribútum_vagy_dimenzió⟩
```

4.4. ábra | A DMQL adatbányászati lekérdező nyelv szintaxisának felső szintje

szeiben az adatbányászati feladat lekérdező nyelv formájában történő meghatározása céljából már szemügyre vettük az adatbányászó primitíveket. A primitívek a következőket határozzák meg:

- a feladat szempontjából fontos bányászandó adatok halmaza;
- a bányászandó tudás típusa;

- a felfedezés folyamata során használt háttértudás;
- érdekességi mértékek és küszöbök a mintázat kiértékelésére;
- a felfedezett mintázat megjelenítése.

Ezekre a primitívekre alapozva megszerkesztjük a **DMQL**-nek (Data Mining Query Language) nevezett adatbányászati lekérdező nyelvet. A DMQL lehetővé teszi számos típusú tudás több absztrakciós szinten történő *ad hoc* bányászatát a relációs adatbázisokból és az adattárházakból.[2]

A nyelv egy SQL-szerű szintaktikát alkalmaz, így könnyen integrálható az SQL relációs lekérdező nyelvvel. A DMQL szintaktikáját a kiterjesztett BNF nyelvtanban határozták meg, ahol a [] nulla vagy egy előfordulást, a {} pedig nulla vagy több előfordulást reprezentál, és a **Serifa** betűtípusú szavak pedig kulcsszavakat jelölnek.

A 4.2.1–4.2.5. alfejezetekben kiépítjük a DMQL szintaktikáját minden adatbányászó primitívre. A 4.2.6. alfejezetben pedig egy példát mutatunk be az adatbányászati lekérdezésre az előbbiekben meghatározott szintaktika alapján. A nyelv szintaxisának felső szintje a 4.4. ábrán látható.

4.2.1. | A feladat szempontjából fontos adatok meghatározásának szintaktikája

Az adatbányászati folyamat első lépése a feladat szempontjából fontos adatok meghatározása, vagyis azon adatoké, amelyeken az adatbányászatot végrehajtjuk. Ez magában foglalja a fontos adatokat tartalmazó adatbázis és táblák vagy adattárház meghatározását, a fontos adatokat leválogató feltételek megadását, a feltárás szempontjából jelentős attribútumok vagy dimenziók meghatározását és a visszakapott adatok rendezésére vagy csoportosításra vonatkozó instrukciók megadását. Az ilyen jellegű információk meghatározására a DMQL a következő záradékokat kínálja:

- **use database** ⟨adatbázis_név⟩ vagy **use data warehouse** ⟨adattárház_név⟩ – A **use** záradék az adatbányászati folyamatot a meghatározott adatbázisra vagy adattárházra irányítja.
- **from** ⟨tábla(ák)/kocka(ák)⟩ [**where** ⟨feltétel⟩] – A **from** és a **where** záradékok kijelölik az adatbázistáblákat vagy adatkockákat, illetve a visszakeresendő adatokat meghatározó feltételeket.
- **in relevance to** ⟨attribútum_vagy_dimenzió_lista⟩ – Ez a záradék a feltárás számára listázza az attribútumokat vagy dimenziókat.
- **order by** ⟨rendezési_lista⟩ – Az **order by** záradék a feladat szempontjából fontos adatok rendezési sorrendjét adja meg.
- **group by** ⟨csoportosítási_lista⟩ – A **group by** záradék az adatok csoportosításának kritériumait határozza meg.
- **having** ⟨feltétel⟩ – A **having** záradék azt a feltételt adja meg, amely meghatározza, hogy mely adatcsoportok tekinthetők fontosnak.

2 | A DMQL adattárházak és adatpiacok meghatározására vonatkozó szintaktikáját lásd a 2. fejezetben.

Ezek a záradékok egy a feladat szempontjából fontos adatokat összegyűjtő SQL-lekérdezést alakítanak ki.

4.9. példa | A példa bemutatja, hogy hogyan használjuk a DMQL nyelvet a 4.1. példában leírt feladat fontos adatainak meghatározására. A feladat a kanadai vevők által vásárolt *ElektroMind* termékek közti kapcsolatot vizsgálja a vevők *jövedelmére* és *életkorára* vonatkozóan. Ezenkívül a felhasználó azt is meghatározza, hogy az adatok dátum szerint legyenek csoportosítva. Az adatokat relációs adatbázisból nyerjük.

use database ElektroMind_db
in relevance to T.név, T.ár, U.jövedelem, U.életkor
from ügyfél U, termék T, vásárlás V, eladás E
    where T.termék_ID = E.termék_ID **and** E.tranzakció_ID = V.tranzakció_ID
    and V.vevő_ID = U. vevő_ID **and** U.ország = "Kanada"
group by V.dátum ■

4.2.2. | Szintaktika a bányászandó tudás típusának meghatározására

A ⟨Bányászandó_Tudás_Meghatározása⟩ kifejezés a bányászandó tudás típusának meghatározására használatos. Másként kifejezve, ez mutatja meg a végrehajtandó adatbányászat működési elvét. A kifejezés szintaxisát az alábbiakban a karakterizáció, a diszkrimináció, a társítás és az osztályozás esetére definiáljuk.

Karakterizáció

⟨Bányászandó_Tudás_Meghatározása⟩ ::=
mine characteristics [**as** ⟨mintázat_név⟩]
analyze ⟨mérték(ek)⟩

Meghatározza, hogy karakterisztikus leírást kell bányászni. Ha az **analyze** záradékot karakterizációra használjuk, akkor olyan aggregáló mértéket határoz meg, mint például a `count`, `sum` vagy `count%` (százalékos count, vagyis a feladat szempontjából fontos adatok azon százaléka, amelyekre teljesül a karakterisztika). Ezek a mértékek kiszámítódnak minden megtalált adatkarakterisztikára.

4.10. példa | A következő példa meghatározza, hogy a bányászandó tudás típusa olyan karakterisztikus leírás, ami a vevők vásárlási szokásait jellemzi. Minden karakterisztikára kijelezzük a feladat szempontjából fontos adatok azon százalékát is, amelyek kielégítik a karakterisztikát.

mine characteristics as ügyfélVásárlása
analyze count% ■

Diszkrimináció

⟨Bányászandó_Tudás_Meghatározása⟩ ::=
mine comparison [**as** ⟨mintázat_név⟩]
for ⟨cél_osztály⟩ **where** ⟨cél_feltétel⟩
{**versus** ⟨különböző_osztály_*i*⟩ **where** ⟨különböző_feltétel_*i*⟩
analyze ⟨mérték(ek)⟩

Meghatározza, hogy megkülönböztető leírást kell bányászni. Ezek a leírások az objektumok egy adott osztályát hasonlítják össze más, különböző osztályokkal. Így a tudásnak ezt a fajtáját **összehasonlításnak** (**comparison**) is nevezik. Mint ahogy a karakretizációnál, az **analyze** záradék itt is minden leírásra meghatározza a kiszámítandó és megjelenítendő aggregáló mértéket, például `count`, `sum` vagy `count%`.

4.11. példa | A felhasználó meghatározhatja a vásárlók kategóriáit, majd leírásokat bányászhat minden egyes kategóriára vonatkozóan. Például a felhasználó definiálhatja a *költekezőVásárlók* kategóriát, amelybe azok a vevők tartoznak, akik átlagosan több mint 100 $-ba kerülő termékeket vásárolnak, és a *takarékosVásárlók* kategóriát, ahol az ügyfelek átlagosan 100 $-nál kevesebbe kerülő termékeket vásárolnak. E két kategóriát megkülönböztető leírások bányászatát meghatározó DMQL-lekérdezés az alábbiakban látható, ahol *T* a *termék* relációt jelenti. Szintén megjelenítésre kerül a leírásokat kielégítő, a feladat szempontjából fontos adatsorok száma is.

mine comparison as vásárlásCsoportok
for költekezőVásárlók **where** avg(T.ár) $\geq$ 100 $
versus takarékosVásárlók **where** avg(T.ár) < 100 $
analyze count ■

Társítás

⟨Bányászandó_Tudás_Meghatározása⟩ ::=
mine associations [**as** ⟨mintázat_név⟩]
[**matching** ⟨metamintázat⟩]

A bányászandó mintázatok típusát társításként határozza meg. Ha társítások bányászatát választjuk, akkor a felhasználó a **matching** záradék használatával opcionálisan meghatározhatja a mintázat sablonját is (metamintázatként vagy metaszabályként is ismert). A metamintázatokat arra használhatjuk, hogy a felfedezést egy adott metamintázatra illeszkedő mintázatok bányászata felé irányítsuk, és ezáltal járulékos szintaktikai korlátozásoknak szerezhessünk érvényt a bányászati folyamat során. Emellett, szintaktikai korlátozások megadása esetén a metamintázatok a felhasználó által a vizsgálat szempontjából fontosnak tartott adathalmazokat, vagy hipotéziseket képviselik. A metamintázatokkal történő bányászat, vagy más néven a **metaszabály-vezérelt bányászat**, kiegészítő rugalmasságot biztosít az ad hoc szabálybányászathoz. Bár a metamintázatok más tudásforma bányászatához is használatosak, mégis a társítások bányászata során a leghasznosabbak, mivel a lehetségesen generált társítások száma hatalmas.

4.12. példa | A 4.2. példa metamintázata a következőképpen határozható meg a vevők vásárlási szokásait leíró társító szabályok bányászatának vezérléséhez:

mine associations as vásárlásiSzokások
matching $P(X : vásárló, W) \wedge Q(X, Y) \Rightarrow vásárlás(X, Z)$

ahol X az ügyfél reláció kulcsa, P és Q predikátum változók, melyek helyettesíthetők a feladat szempontjából fontos adatrész fontos attribútumaival, illetve dimenziókkal, W, Y és Z pedig objektumváltozók, amelyek az X vásárló megfelelő predikátumainak értékeit vehetik fel. ■

Osztályozás

⟨Bányászandó_Tudás_Meghatározása⟩ ::=
mine classification [**as** ⟨mintázat_név⟩]
analyze ⟨osztályozó_attribútum_vagy_dimenzió⟩

Meghatározza, hogy adatokat osztályozó mintázatokat kell bányászni. Az **analyze** záradék előírja, hogy az osztályozás az ⟨osztályozó_attribútum_vagy_dimenzió⟩ kifejezés értékeinek megfelelően kell hogy végrehajtódjon. A kategorikus attribútumok vagy dimenziók minden egyes értéke jellemzően egy-egy osztályt képvisel (például *alacsony kockázat*, *közepes kockázat* és *magas kockázat* a *hitel_képesség* attribútum esetében). A numerikus attribútumok, vagy dimenziók esetében minden osztály egy-egy értéktartományként határozható meg (mint például 20–39, 40–59, 60–89 az *életkor* esetében). Az osztályozás tömör keretet szolgáltat, amely a legjobban írja le minden egyes osztály objektumait, megkülönböztetve más osztályoktól.

4.13. példa | A vásárlók hitelképességét osztályozó mintázatok bányászatára, ahol az osztályokat a *hitel_képesség* attribútum határozza meg, a következő DMQL-meghatározás használatos:

mine classification as VásárlókHitelképességénekOsztályozása
analyze hitel_képesség ■

Az adatbányászati rendszereknek a fentiekben említett tudástípusokon kívül más tudástípusok meghatározását is biztosítaniuk kell. Ezek a típusok magukban foglalják az adatklaszterek bányászatát, a fejlődési szabályokat vagy a szekvenciális mintázatokat és az eltéréseket.

4.2.3. | A fogalmi hierarchia meghatározásának szintaktikája

A fogalmi hierarchiák lehetővé teszik a tudás több absztrakciós szinten történő bányászatát. A felhasználók adatokra vonatkozó különböző nézőpontjai alapján lehetséges, hogy attribútumonként vagy dimenziónként egynél több hierarchia is létezik. Például néhány felhasználó az üzletek helyeinek csoportosításánál a megyék vagy tartományok szerinti

csoportosítást helyezheti előtérbe, míg más felhasználók a beszélt nyelvek alapján történő csoportosítást támogathatják. Ilyen esetekben a felhasználó jelölheti meg, hogy a kifejezés melyik fogalmi hierarchiáját használja:

use hierarchy ⟨hierarchia_név⟩ **for** ⟨attribútum_vagy_dimenzió⟩

Egyéb esetekben attribútumonként, vagy dimenziónként az alapértelmezett hierarchia használatos.

Hogyan definiálhatjuk a fogalmi hierarchiát a DMQL használatával? A 4.1.3. alfejezetben a fogalmi hierarchiáknak 4 típusát tekintettük át. Nézzünk néhány példát a fogalmi hierarchiák DMQL-lel történő definiálására.

4.14. példa | A sémahierarchia definíciója. Korábban, a *cím* reláció sémahierarchiáját az *utca* < *város* < *megye_vagy_tartomány* < *ország* teljes rendezésként definiáltuk. Ez a hierarchia az adatbányászati lekérdező nyelvvel a következőképpen adható meg:

define hierarchy hely_hierarchia **on** cím **as**
[utca, város, megye_vagy_tartomány, ország]

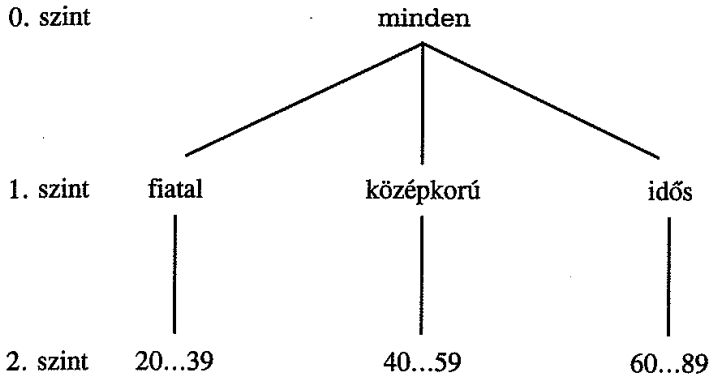
A felsorolt attribútumok sorrendje fontos. Valójában egy teljes rendezés van megadva, amely meghatározza, hogy az *utca* eggyel alacsonyabb szinten van, mint a *város*, amely egy szinttel alacsonyabban helyezkedik el, mint a *megye_vagy_tartomány* és így tovább. ■

4.15. példa | A halmazcsoportosító hierarchia definíciója. A 4.4. példa életkorra vonatkozó halmazcsoportosító hierarchiája a tartományok szintjén a következőképpen határozható meg. A szokásoknak megfelelően a **minden** a legáltalánosabb fogalom, amely a hierarchia gyökerénél helyezkedik el (vagyis a 0. szinten).

define hierarchy életkor_hierarchia **for** életkor **on** vásárló **as**
1. szint: {*fiatal, középkorú, idős*} < 0. szint: **minden**
2. szint: {20...39} < 1. szint: *fiatal*
2. szint: {40...59} < 1. szint: *középkorú*
2. szint: {60...89} < 1. szint: *idős*

A „..." jelölés az adott tartományon belüli értékeket határozza meg hallgatólagosan. Például a {20...39} magában foglalja a 20-as és 39-es végpontú tartomány között található egész számokat. A tartományok valós számokat jelölő végpontokkal is meghatározhatók. A példának megfelelő fogalmi hierarchiát a 4.5. ábra mutatja be. ■

A további példák a feladatok között találhatók.



4.5. ábra | Az *életkor* attribútum fogalmi hierarchiája

4.2.4. | Az érdekességi mérték meghatározásának szintaktikája

A felhasználó az adatbányászati rendszer által visszaadott érdektelen mintázatok számát a mintázatra vonatkozó érdekességi mértékek és a hozzájuk tartozó küszöbértékek meghatározásával korlátozhatja. Az érdekességi mértékek magukban foglalják a 4.1.4. fejezetben ismertetett megalapozottságot, megbízhatóságot (konfidenciát), zajt és az újszerűség mértékét. A felhasználó az érdekességi mértékeket és a küszöbértékeket a következő kifejezéssel határozhatja meg:

with [⟨érdekességi_mérték_név⟩] **threshold** = ⟨küszöb_érték⟩

4.16. példa | A társító szabályok bányászata során a felhasználó korlátozhatja a megtalált szabályokat azáltal, hogy a kifejezésben 5%-os minimális megalapozottságot (support) és 70%-os minimális megbízhatósági (confidence) küszöbértéket (threshold) állít be:

with support **threshold** = 5%
with confidence **threshold** = 70% ■

Az érdekességi mértékek és a küszöbértékek interaktív módon állíthatók be és módosíthatók.

4.2.5. | Szintaxis a mintázat bemutatásának és ábrázolásának meghatározására

Hogyan határozhatja meg a felhasználó a felfedezett mintázat megjelenítésekor használt bemutatási és ábrázolási formákat? Az adatbányászati lekérdező nyelvünknek szüksége van olyan szintaktikára, amely lehetővé teszi a felhasználó számára, hogy a felfedezett mintázat megjelenítésének egy vagy több formáját is megadja. Ezek a formák lehetnek szabályok, táblázatok, kereszttáblák, torta- vagy oszlopdiagramok, döntési fák, kockák, görbék és felületek. A DMQL **display** kifejezését ebből a célból adjuk meg:

display as ⟨eredmény_forma⟩

ahol az ⟨eredmény_forma⟩ bármely fentebb felsorolt tudásmegjelenítési, vagy ábrázolási forma lehet.

Az interaktív bányászatnak biztosítania kell, hogy a felfedezett mintázatok különböző fogalmi szinteken, illetve különböző szempontból is áttekinthetők legyenek. Ez megvalósítható a 2. fejezetben ismertetett lefúrás és felgörgetés operátorokkal. A mintázatok összevonhatók, illetve megtekinthetők egy magasabb, általánosabb szinten, egy attribútum, vagy egy dimenzió fogalmi hierarchiájának *megmászásával* (az alacsonyabb szintű fogalmakat magasabb szintű értékekre cserélve). Attribútumok vagy dimenziók törlésével általánosítás is végrehajtható. Például tegyük fel, hogy a mintázat tartalmazza a *város* attribútumot. Adott a *város* < *megye_vagy_tartomány* < *ország* < *kontinens* hierarchia a *hely*-re vonatkozóan. Ebben az esetben, ha eldobjuk a *város* attribútumot a mintázatból, akkor az adatok a következő magasabb, vagyis a *megye_vagy_tartomány* attribútum szintjére fognak általánosodni. Egy attribútum vagy dimenzió fogalmi hierarchiáján történő lefelé lépkedéssel a mintázatok finomíthatók, illetve egy kevésbé általános szinten tekinthetők meg. A mintázatok úgy is kevésbé általánossá tehetők, hogy a leíráshoz attribútumokat vagy dimenziókat rendelünk. A hozzáadott attribútumnak a feladat szempontjából fontos adatok meghatározásában szerepet játszó **in relevance to** záradékban is szerepelniük kell. A felhasználó a következő DMQL-szintaxis használatával váltakozva nézheti különböző absztrakciós szinteken a mintázatokat:

⟨Többszintű_Művelet⟩ ::= **roll up on** ⟨attribútum_vagy_dimenzió⟩
| **drill down on** ⟨attribútum_vagy_dimenzió⟩
| **add** ⟨attribútum_vagy_dimenzió⟩
| **drop** ⟨attribútum_vagy_dimenzió⟩

4.17. példa | Tegyük fel, hogy a leírás bányászata a *hely*, *életkor* és *jövedelem* dimenziók alapján történik. A felhasználó a felfedezett mintázat általánosítása céljából felgörgetést hajthat végre a *hely* attribútumon, vagy törölheti az *életkor* attribútumot. ■

4.2.6. | Mindent egybevéve – példa a DMQL-lekérdezésre

A fentiekben bemutattuk a DMQL szintaxisát az öt adatbányászó primitív által meghatározott adatbányászati lekérdezések megadására vonatkozóan. Egy adott lekérdezés esetében ezek a primitívek meghatározzák a feladat szempontjából fontos adatokat, a bányászandó tudás típusát, a fogalmi hierarchiákat, a felhasznált érdekességi mértékeket és a mintázat ábrázolásának megjelenési formáit. Nézzünk egy példát a DMQL-lekérdezés teljes leírására!

4.18. példa | **Karakterisztikus leírások bányászata.** Tegyük fel, hogy az *ElektroMind* kereskedelmi igazgatójaként azoknak az ügyfeleknek a vásárlási szokásait szeretnénk jellemezni, akik legalább 100 $-ba kerülő termékeket vásárolnak. Mindehhez az ügyfelek életkorát, a vásárolt termék típusát és gyártásának helyét szeretnénk figyelembe venni. Minden egyes felfedezett karakterisztikára szeretnénk tudni, hogy a vásárlók hány százalékára jellemző ez a karakterisztika. Továbbá, főként az olyan vásárlások érdekelnek bennünket,

amelyeket Kanadában hajtottak végre és American Express ("AmEx") kártyával fizettek ki. Az eredmény leírását pedig táblázat formájában szeretnénk áttekinteni. Ez az adatbányászó lekérdezés DMQL-ben a következőképpen fejezhető ki:

```
use database ElektroMind_db
use hierarchy hely_hierarchia for A.cím
mine characteristics as vásárlóiSzokások
analyze count%
in relevance to U.életkor, T.típus, T.gyártási_hely
from ügyfél U, termék T, vásárlás V, termék_eladás E, működik_hol M, üzlet C
where T.termék_ID = E.termék_ID and E.tranzakció_ID = V.tranzakció_ID
      and V.vásárló_ID = U.vásárló_ID and V.fizetési_mód = "AmEx"
      and V.alkalmazott_ID = M.alkalmazott_ID and M.uzlet_ID = C.uzlet_ID
      and C.cím = "Kanada" and T.ár ? 100
with noise threshold = 5%
display as table
```

Az adatbányászati lekérdezés elemzésével egy olyan SQL-lekérdezés készül, ami a feladat szempontjából fontos adathalmazt adja vissza az *ElektroMind* adatbázisból. A *hely_hierarchia* – amely megfelel a 4.3. ábra fogalmi hierarchiájának – használható az üzletek helyének olyan magas szintű általánosítására, mint például „Kanada". Ezek után futtatható le egy olyan karakterisztikus szabályokat bányászó algoritmus, amely általánosított adatokat használ. A karakterisztikus szabályokat bányászó algoritmusokat az 5. fejezet mutatja be. Az *életkor*ból, *típus*ból és a *gyártási_hely*ből kibányászott karakterisztikus leírások táblában vagy általánosított relációban kerülnek bemutatásra (4.1. táblázat). A feladat szempontjából fontos adatsorok azon százaléka, amelyek kielégítik az általánosított sorokat, `count%`-ként kerülnek kijelzésre. Ha nincs megadva ábrázolási forma, akkor az alapértelmezett forma kerül felhasználásra. Az 5%-os zajküszöb azt jelenti, hogy a sorok kevesebb, mint 5%-át reprezentáló általánosított sorok nem kerülnek bemutatásra. ■

Hasonlóan adhatók meg teljes DMQL-utasítások a diszkrimináció, a társítás, az osztályozás és az előrejelzés esetében is. Ehhez kapcsolódó példalekérdezésekről a következő fejezetekben lesz szó, ahol ezeket a tudástípusokat fogjuk tanulmányozni.

4.1. táblázat | Karakterisztikus leírások táblázat, vagy általánosított reláció formájában

életkor	típus	gyártási_hely	count%
30...39	otthoni biztonsági rendszer	USA	19
40...49	otthoni biztonsági rendszer	USA	15
20...29	CD-lejátszó	Japán	26
30...39	CD-lejátszó	USA	13
40...49	nagyképernyős TV	Japán	8
...	...	...	...
			100%

4.2.7. | További adatbányászati nyelvek és az adatbányászó primitívek szabványosítása

A könyv a DMQL-t tárgyalja. Létezik más adatbányászati nyelv is? A könyvben az adatbányászó primitívek és fogalmak bemutatására a DMQL-t használtuk. Más adatbányászati nyelvekre irányuló kutatások is folynak, és az ipar is erőfeszítéseket tesz az adatbányászó primitívek és nyelvek szabványosítására. Most mi is bemutatunk néhány példát. Ezekről a nyelvekről és szabványokról további információk találhatók a fejezet végén felsorolt irodalomban.

Az **MSQL** Imielinski és Virmani [IV99] által javasolt adatbányászati nyelv. A nyelv SQL-szerű szintaktikát használ és az SQL-primitívek tartalmazzák a rendezést és a csoportosítást. Mivel az adatbányászat során roppant nagy mennyiségű szabály hozható létre, az MSQL **GetRules** és **SelectRules** jellegű primitíveket biztosít a szabályok létrehozására és kiválogatására. Egységesen kezeli az adatokat és a szabályokat, s ezáltal lehetőséget ad az optimalizálásra vagy az adatokból történő szelektív lekérdezésen alapuló szabályok generálásával, vagy a generált szabályhalmaz manipulálásával és lekérdezésével.

Más adatbányászati nyelvek terén végzett kutatási eredmények a Meo, Psaila és Ceri [MPC96] által javasolt **MINE RULE** operátort tartalmazzák. Az operátor SQL-jellegű szintaktikát követ, és a társító szabályok bányászatára szabálygeneráló lekérdezésként szolgál. Egy másik, Tsur, Ullman, Abitboul, Clifton és mások [TUA+98] által adott javaslat Datalog szintaktikát használ a kérdések csoportjainak kifejezésére, amely elősegíti a bányászatot és a szabályok tesztelését is.

Létezik az adatbányászatban valamilyen szabványosítási eredmény is? Az utóbbi időben a Microsoft indítványozott egy adatbányászati nyelvet, amelynek elnevezése **OLE DB for Data Mining (DM)**. Ez figyelemre méltó erőfeszítés az adatbányászati nyelvek primitívjeinek szabványosítása terén. Egy szabványos adatbányászati nyelv megléte segíteni fog az adatbányászati ipar megerősítésében, az adatbányászati platformok és rendszerek fejlesztésével, valamint az adatbányászati eredmények megosztásával.

Az OLE DB for DM együtt az OLE DB-vel és az OLE DB for OLAP-pal a Microsoft három fontos lépést tett az adatbázisok, az adattárházak és az adatbányászati nyelvek szabványosításának irányába. Az OLE DB for DM leírása lefedi a létrehozáshoz szükséges primitíveket, és számos fontos adatbányászó modult használ, beleértve a társítást, az előrejelző modellezést (osztályozás és előrejelzés) és a klaszterezést. A könyvben azért választottuk a DMQL-t adatbányászati nyelvnek, mert a könyv megjelenésekor az OLE DB for DM részletei még nincsenek végleges formába öntve. Az OLE DB for DM-be történő rövid bevezetés az A) függelékben olvasható.

A CRIPS-DM (CRoss-Industry Standard Process for Data Mining) *(http://www.crisp-dm.org/)* a másik, az adatbányászathoz kapcsolódó szabványosítási eredmény. Ez eltávolodik attól az elgondolástól, hogy a hangsúly az adatbázis-technológiában felvonultatott felhasználói igények minden szinten történő kielégítésének technikai megoldásán legyen, és inkább az üzleti problémák megoldása felé mozdul el. A CRISP-DM nemzetközi projekt, amely néhány adattárházzal foglalkozó, adatbányászati és felhasználói céget hozott össze abból a célból, hogy együtt dolgozzanak a platform és a folyamatstruktúra kialakításán. A projekt olyan adatbányászati folyamatot határoz meg, amely általánosan alkalmazható a különféle ipari szektorokban. A következő területeket célozza meg: (1) az

üzleti ügyek leképzése adatbányászati problémákká; (2) az adatok összegyűjtése és megértése; (3) az adatok közt felmerülő problémák azonosítása és megoldása; (4) az adatbányászati technikák alkalmazása; (5) az adatbányászati elemek értelmezése az üzleti környezeten belül; (6) az adatbányászati eredmények telepítése és karbantartása az üzleti környezetben; (7) a szakvélemény megszerzése és átültetése azért, hogy a tapasztalatokból profitálva további projektek végrehajtását biztosítsák. A projekt folyamatstruktúrát szolgáltat az adatbányászat véghezviteléhez, és útmutatást ad az adatbányászati projektek során felmerülő lehetséges problémákra.

Az adatbányászati rendszerek további fejlesztésével és az adatbányászati nyelvek szabványosításával arra lehet számítani, hogy ezen könyv későbbi kiadásaiban használatos adatbányászati nyelveket az OLE DB for DM-hez hasonló primitívek irányába fejlesztik majd tovább, vagy végső fokon lecserélik őket egy sokkal összetettebb, szabványos lekérdező nyelvre, bárhogy is hívják majd azt a nyelvet a későbbiek folyamán.

4.3. | Az adatbányászó lekérdező nyelven alapuló grafikus felhasználói felület fejlesztése

Az adatbányászati lekérdező nyelv szolgáltatja azokat a szükséges primitíveket, amelyek lehetővé teszik a felhasználó kommunikációját az adatbányászati rendszerekkel. Azonban a tapasztalatlan felhasználó az adatbányászati lekérdező nyelvek használatát kényelmetlennek találhatja, és a szintaxisukat is nehéz megjegyezni. A felhasználó ehelyett inkább az olyan adatbányászati rendszereket kedveli, ahol a kommunikáció grafikus felhasználói felületen (GUI) keresztül megy végbe. A relációs adatbázis-technológiában az SQL szabványos „nyelvi magként" szolgál a relációs rendszerek számára, amely fölé a grafikus felhasználói felületek könnyedén kifejleszthetők. Ehhez hasonlóan az adatbányászati lekérdező nyelvek is „nyelvi magként" szolgálhatnak az adatbányászati rendszerek alkalmazásai számára, ezáltal alapot adhatnak a hatékony adatbányászati rendszerek grafikus felhasználói felületeinek fejlesztéséhez.

Az adatbányászati rendszerek grafikus felületei a következő funkcionális komponenseket tartalmazhatják:

- **Adatösszegyűjtő és adatbányászati lekérdezésszerkesztő** – Ez az összetevő lehetővé teszi a felhasználó számára, hogy meghatározza a feladat szempontjából fontos adatok halmazát, és megszerkessze az adatbányászati lekérdezéseket. Ez hasonló a relációs lekérdezések meghatározásakor használt grafikus felhasználói felületekhez.
- **A felfedezett mintázat bemutatása** – Ez a komponens teszi lehetővé a felfedezett mintázatok különböző formákban történő megjelenítését, beleértve a táblázatokat, a grafikonokat, a diagramokat, a görbéket és más megjelenítési technikákat.
- **A hierarchia meghatározása és manipulálása** – Ez a komponens lehetővé teszi a fogalmi hierarchia automatikus (a „kéznél lévő" adatok elemzése alapján) vagy felhasználó által történő meghatározását. Ezenkívül, ennek az összetevőnek lehetővé kell tennie a fogalmi hierarchiák felhasználó által történő, vagy az adott adathalmaz eloszlása alapján automatikusan végbemenő módosítását is.

- **Adatbányászó primitívek manipulációja** – A komponensnek éppúgy lehetővé kell tennie az adatbányászati küszöbök dinamikus igazítását, mint a fogalmi hierarchiák kiválasztását, megjelenítését és módosítását. Továbbá lehetővé kell tennie a korábbi adatbányászati lekérdezések vagy feltételek módosítását is.
- **Interaktív többszintű bányászat** – Ennek a komponensnek kell lehetővé tennie a felfedezett mintázatokon a felgörgetés és a lefúrás végrehajtását.
- **Egyéb különféle információk** – Ez a komponens tartalmazhatja az on-line kézikönyveket, indexelt keresést, hibakeresést és más interaktív grafikai lehetőségeket.

A grafikus felhasználói rendszerek fejlesztésekor azonban figyelembe kell venni az adatbányászati rendszereket használó felhasználók különféle csoportjait is. Ezek a felhasználók általában két csoportra oszthatók: *üzleti elemzők* és *üzleti vezetők*. Az üzleti elemzők rugalmasságot és kényelmet szeretnének az adatok különböző részeinek kiválogatásakor, a dimenziók és szintek beállításakor, a bányászandó mintázatok megadásakor és az adatbányászó folyamat hangolásakor. Másrészről az üzleti vezetőknek az adatbányászati eredmények érthető bemutatására és értelmezésére, a különböző adatbányászati eredmények áttekintésében és összehasonlításában rugalmasságra van szükségük, és arra, hogy az eredmények riportkészítő és bemutató módszerekkel könnyen feldolgozhatók legyenek. A jól tervezett adatbányászati rendszernek mind a két típusú felhasználó számára biztosítania kell a felhasználóbarát felületet.

Várható-e az adatbányászati grafikai felületek tervezéséhez szabványos adatbányászati lekérdező nyelv kifejlesztése? Ha ez lehetséges, akkor a szabvány elő fogja segíteni az adatbányászó szoftverek fejlesztését és a rendszerkommunikációt. Azonban néhány grafikus felhasználói primitív kifejezése – például rámutatni egy görbe vagy egy grafikon bizonyos pontjára – nehézkes a DMQL-hez hasonló szöveg alapú adatbányászati lekérdező nyelvvel. Alternatívaként kifejleszthető egy GUI alapú szabványosított nyelv és ez felválthatja az SQL-szerű adatbányászati nyelveket. Azonban erre a kérdésre majd csak az idő ad választ.

4.4. | Az adatbányászati rendszerek architektúrája

Mivel az adatbányászatnak számos sokrétű és népszerű felhasználása létezik, ezért várható, hogy az elkövetkező években az adatbányászati rendszerek sokfajta típusát fogják kifejleszteni. Jóllehet a szoftverrendszerekhez hasonlóan az adatbányászati rendszer magját is nagy számú és nagy teljesítményű adatbányászati függvény teszi ki, az adatbányászati rendszerek architektúrája, megtervezése mégis nagy fontossággal bíró feladat. A jó rendszerarchitektúra elősegíti, hogy a rendszerek működésük folyamán a szoftverkörnyezetet a lehető leghatékonyabban használják ki, támogatja az adatbányászati feladatok hatékony és gyors végrehajtását, a rendszerek más információs rendszerekkel való együttműködését és információcseréjét, valamint megkönnyíti a rendszerek fejlődését és alkalmazkodását a sokrétű felhasználói igényekhez.

Milyen rendszerarchitektúrákat igényelnek az adatbányászati rendszerek? Az adatbázis- és információs szektor kutatásának és fejlesztésének évtizedei során az adatbázisrendszerek és az adattárházak az információs rendszerek vezető ágazatává váltak. Ezek-

ben a rendszerekben óriási mennyiségű adatot és információt tároltak és/vagy integráltak. Ezenkívül az adatbázisrendszereket és az adattárházakat körbeölelő adatelemző infrastruktúrákat és a széles körű információfeldolgozást folyamatosan és szisztematikusan fejlesztik. Ez magában foglalja a többrétű és heterogén adatbázisok elérését, integrációját, konszolidációját és átalakítását, az ODBC/OLE DB kapcsolatokat, a webes elérést és a webes szolgáltatási lehetőségeket, valamint a jelentéskészítést és az OLAP-elemző eszközöket.

Ebben a helyzetben az adatbányászati rendszerek tervezésének kritikus kérdése, hogy összekapcsolni, vagy integrálni kell-e az adatbányászati rendszereket (DM-) az adatbázis- (DB-) és/vagy adattárház- (DW-) rendszerekkel, és amennyiben igen, hogyan is tegyük meg mindezt? A kérdés megválaszolásához meg kell vizsgálnunk a DM-rendszerek DB/DW-rendszerekkel történő összekapcsolásának, illetve integrálásának lehetséges módozatait. A DM-rendszerek a DB/DW-rendszerekkel a különböző architektúrák tervezése alapján a következő összekapcsolási sémákat használva integrálhatók: *nincs összekapcsolás*, *laza összekapcsolás*, *közepesen szoros összekapcsolás* és *szoros összekapcsolás*. Vizsgáljuk meg ezeket a lehetőségeket egyenként!

Nincs összekapcsolás – A *nincs összekapcsolás* azt jelenti, hogy a DM-rendszer nem használja fel a DB- vagy DW-rendszer egyik funkcióját sem. Az adatokat saját adatforrásból veszi (mint például egy fájlrendszer), az adatbányászó algoritmusok használatával feldolgozza azokat, és a bányászat eredményeit más fájlokban tárolja.

Az ilyen rendszerek, habár egyszerűek, mégis számos hátránnyal járnak. Először is a DB-rendszer rugalmas és hatékony kezelést biztosít az adatok tárolására, rendszerezésére, elérésére és feldolgozására. A DB/DW-rendszer használata nélkül a DM-rendszer tekintélyes időt tölthet az adatok megkeresésével, összegyűjtésével, tisztításával és átalakításával. A DB- és/vagy DW-rendszerekben az adatok jól szervezettek, megfelelően indexeltek, tiszták, jól integráltak vagy konszolidáltak, így a jó minőségű és a feladat szempontjából fontos adatok megtalálása könnyű. Másrészt a DB- és DW-rendszerek számos tesztelt és skálázható algoritmust, valamint adatstruktúrát tartalmaznak. Ezeknek a rendszereknek a használatával hatékony és skálázható alkalmazások hozhatók létre. Továbbá az adatok nagy részét DB/DW-rendszerekben tárolják. Ezekkel a rendszerekkel való bármilyen jellegű összekapcsolás nélkül a DM-rendszernek más adatkinyerő eszközt kell használnia, ami megnehezíti a rendszer beillesztését az információfeldolgozó környezetbe. Az összekapcsolás hiánya tehát szegényes tervezést mutat.

Laza összekapcsolás – A *laza összekapcsolás* azt jelenti, hogy a DB/DW-rendszerek által kezelt adattárházakból való adatelérés során a DM-rendszer használja a DB/DW-rendszer bizonyos lehetőségeit, ezt követi az adatbányászat végrehajtása, majd az adatbányászat eredményeinek fájlban, vagy az adatbázis, illetve az adattárház kijelölt helyein történő tárolása.

A laza összekapcsolás jobb, mintha nem lenne összekapcsolás, mivel a lekérdezések végrehajtásával, az indexeléssel és más DB/DW-rendszer által biztosított lehetőségek használata által a DM-rendszer elérheti és beolvashatja az adatbázisban, illetve az adattárházban tárolt adatok tetszőleges részét. Ezáltal a DM-rendszer magába olvasztja a DB/DW-rendszerek által kínált rugalmasságot, hatékonyságot és egyéb tulajdonságokat. Azonban számos laza kapcsolású rendszer a központi memórián alapszik. Mivel azonban maga

a bányászat nem fedezi fel a DB- és DW-rendszerek által kínált adatstruktúrákat és lekérdezésoptimalizáló eljárásokat, ezért a nagy adathalmazok esetében a laza összekapcsolással nehéz elérni magas szintű skálázhatóságot és jó teljesítményt.

Közepesen szoros összekapcsolás – A közepesen szoros összekapcsolás azt jelenti, hogy a DM-rendszer DB/DW-rendszerhez való kapcsolása mellett, a DB/DW-rendszerben hatékonyan megvalósítható néhány alapvető adatbányászó primitív (amelyeket a gyakran előforduló adatbányászati függvények elemzése azonosít) is. Ezek a primitívek magukba foglalhatják a rendezést, az indexelést, az aggregációt, a grafikon elemzését, az összekapcsolás módozatait és néhány alapvető statisztikai mérték, mint például a sum, count, max, min, standard szórás stb. előzetes kiszámítását. Továbbá számos gyakran használt közbülső bányászati eredmény előre kiszámítható és tárolható a DB/DW-rendszerben. Mivel ezek a közbülső bányászati eredmények vagy előre kiszámítottak, vagy hatékonyan számíthatók, ezért ez a tervezési mód növeli a DM-rendszerek hatékonyságát.

Szoros összekapcsolás – A szoros összekapcsolás azt jelenti, hogy a DM-rendszer be van illesztve a DB/DW-rendszerbe. A DB/DW-rendszer az adatbányász-alrendszert úgy kezeli, mint egy információs rendszer funkcionális komponensét. Az adatbányászati lekérdezések és függvények a DB- vagy DW-rendszer adatbányász-lekérdezés elemzői, adatstruktúrái, indexelési sémái és lekérdezésvégrehajtó eljárásai alapján optimalizáltak. A DB-, DW- és DM-rendszerek többfunkciós információs rendszerré fejlesztése és integrálása további technológiai előnyöket hordoz. Mindez pedig egy általános információfeldolgozó környezetet szolgáltat.

Ez a megközelítés ajánlatos leginkább, mivel elősegíti az adatbányászati függvények hatékony megvalósítását, a rendszerek magas szintű teljesítményét és az integrált információfeldolgozó környezet kialakítását.

Ezzel az elemzéssel mindenki láthatja, hogy az adatbányászati rendszert össze kell kapcsolni a DB/DW-rendszerrel. A laza kapcsolás, mivel használja a DB/DW-rendszer adatait és rendszerének lehetőségeit, bár nem eléggé hatékony, mégis jobb, mintha nem lenne összekapcsolás. Leginkább a szoros összekapcsolás ajánlott, bár ennek megvalósítása nem triviális, és további kutatásokat igényel ezen a területen. A közepesen szoros összekapcsolás kompromisszum a laza és a szoros összekapcsolás között. Fontos a gyakran használt adatbányászó primitívek azonosítása, és ezen primitívek hatékony alkalmazásainak DB-, illetve DW-rendszerekben történő kialakítása.

4.5. | Összefoglalás

- Az adatbányászati feladatokat meghatározó **öt primitívet adatbányászati lekérdezések** formájában tekintettük át. Ezek a primitívek a következők: a feladat szempontjából fontos adatok meghatározása (például a bányászandó adathalmaz), a bányászandó tudás típusa (például karakterizáció, diszkrimináció, társítás, osztályozás, előrejelzés), háttértudás (jellemzően fogalmi hierarchiák formájában), érdekességi mértékek és a felfedezett mintázatok megjelenítésére használatos tudásbemutatási és ábrázolási technikák.

- A **feladat szempontjából fontos adatok** meghatározása során a felhasználó meghatározza a bányászandó adatokat tartalmazó adatbázist és táblákat (vagy adattárházat és adatkockákat), az adatok kiválogatásának és csoportosításának feltételeit és a bányászat során figyelembe veendő attribútumokat (vagy dimenziókat).
- A **fogalmi hierarchiák** hasznos háttértudást nyújtanak a felfedezett mintázatok tömör, magas szintű fogalmakban történő kifejezéséhez, és megkönnyítik a tudás többszörös absztrakciós szinten történő bányászatát.
- A **mintázat érdekességi mértékei** megbecsülik a felfedezett mintázat egyszerűségét, bizonyosságát, hasznosságát és újszerűségét. Az ilyen mértékek használhatók a felhasználó számára visszaadott érdektelen mintázatok számának csökkentésére.
- A felhasználónak képesnek kell lennie arra, hogy meghatározza a felfedezett mintázatok kívánt **ábrázolási formáját**. Ezek a formák lehetnek szabályok, táblázatok, diagramok, döntési fák, kockák, grafikonok, jelentések. A mintázatok többszörös absztrakciós szinten történő megtekintése céljából a felgörgetés és a lefúrás operátoroknak is elérhetőknek kell lenniük.
- Az *ad hoc* és interaktív adatbányászat segítségével tervezhetők **adatbányászati lekérdező nyelvek**. Egy adatbányászati lekérdező nyelvnek, például a DMQL-nek, éppúgy rendelkeznie kell az adatbányászó primitíveket meghatározó parancsokkal, mint a fogalmi hierarchiákat létrehozó és módosító utasításokkal. Az ilyen lekérdező nyelvek SQL alapúak, és olyan szabványosított formát is ölthetnek, amely alapja lehet egy grafikus felhasználói felületet kialakításának is.
- Az **adatbányászati rendszer architektúrája** magában foglalja az adatbányászó rendszer és az adatbázis-/adattárházrendszer összekapcsolásának gondolatát. A következő néhány tervezési mód létezik: *nincs összekapcsolás*, *laza összekapcsolás*, *közepesen szoros összekapcsolás*, *szoros összekapcsolás*. A jól tervezett adatbányászati rendszer az adatbázis- és/vagy adattárházrendszerrel szorosan vagy közepesen szorosan kell összekapcsolódnia.

4.6. | Feladatok

4.1. Soroljuk fel és mutassuk be az adatbányászati feladatokat meghatározó öt primitívet!

4.2. Mutassunk rá a fogalmi hierarchiák hasznosságára az adatbányászatban!

4.3. A fogalmi hierarchiáknak a következő négy fő típusa létezik: *sémahierarchiák*, *halmazcsoportosító hierarchiák*, *műveletvezérelt hierarchiák*, *szabály alapú hierarchiák*.

(a) Röviden definiáljuk az egyes hierarchiatípusokat!

(b) Mindegyik hierarchiatípusra mutassunk olyan példát, amely még nem fordult elő ebben a fejezetben!

4.4. Tegyük fel, hogy a *Csúcs_Egyetem* adatbázis, amely az egyetemi kurzusok adatbázisa, a hallgatók jellemzésére vonatkozóan a következő attribútumokat tartalmazza: *név*, *cím*, *státus* (például alapképzéses hallgató vagy másoddiplomás hallgató), *szak*, *átlag* (tanulmányi átlag).

(a) Javasoljunk fogalmi hierarchiát a *cím*, *státus*, *szak* és *átlag* attribútumokra!

(b) Határozzuk meg a megadott hierarchiák típusait!

(c) Adjuk meg mindegyik hierarchiát a DMQL nyelv szintaxisának segítségével!
(d) Írjunk egy DMQL-lekérdezést a kiváló átlaggal rendelkező hallgatók karakterisztikájának megtalálására!
(e) Írjunk egy DMQL-lekérdezést a természettudományos hallgatók és a bölcsész hallgatók összehasonlítására!
(f) Írjunk egy olyan DMQL-lekérdezést, amely társításokat keres a kurzus oktatói, a hallgatók eredményei és egy választott attribútum között! A keresendő társítások formájának meghatározására használjunk *metaszabályt*! Az eredményül kapott társító szabályokra határozzunk meg a minimális megbízhatósági és megalapozottsági küszöbértékeket!
(g) Írjunk egy olyan DMQL-lekérdezést, amely a hallgatói átlag és a kurzus oktatójának figyelembevételével megjósolja a hallgatók érdemjegyeit a „Számítástudomány" kurzusra vonatkozóan!

4.5. Vegyük szemügyre a (4.8) társítási szabályt, amelyet a *Csúcs_Egyetem* adatbázisból bányásztunk ki!

$$szak(X, \text{“}természettudomány\text{”}) \Rightarrow státus(X, \text{“}alapképzés\text{”}) \quad (4.8)$$

Tegyük fel, hogy az egyetem hallgatóinak a száma (ami a feladat szempontjából fontos adatsoroknak a száma) 5000, valamint hogy az alapképzésben résztvevő hallgatóknak 56%-a természettudományi szakra jár, továbbá, hogy a hallgatók 64%-a van regisztrálva az alapképzést adó kurzusokra, és hogy a hallgatók 70%-a természettudomány szakos.

(a) Számítsuk ki a (4.8) szabály megbízhatóságát és megalapozottságát!
(b) Vegyük az alább látható (4.9) szabályt:

$$szak(X, \text{“}biológia\text{”}) \Rightarrow státus(X, \text{“}alapképzés\text{”}) \quad [17\%, 80\%] \quad (4.9)$$

Tegyük fel, hogy a természettudomány szakos hallgatóknak 30%-a biológia szakos. Tekinthető-e a (4.9) szabály a (4.8) szabályhoz viszonyítva újszerűnek? Állításunkat indokoljuk!

4.6. A ⟨Bányászandó_Tudás_Meghatározása⟩ kifejezés használható annak meghatározására, hogy karakterisztikát, diszkriminációt, társító vagy osztályozó szabályt bányászunk-e. Javasoljunk szintaktikát klaszterek bányászatára!

4.7. A következő példák fogalmi hierarchiák DMQL-szintaktikával történő meghatározására vonatkoznak:

(a) Az adatbányászati rendszer általában a *dátum (nap, hónap, negyedév, év)* sémára vonatkozóan előredefiniált hierarchiát tartalmaz. Adjuk meg egy ilyen hierarchia definícióját DMQL segítségével!
(b) A fogalmi hierarchia meghatározása számos kapcsolatot tartalmazhat. Például egy *termék_hierarchia* tartalmazhatja a *termék* és *szállító* relációkat, amelyeket a következő sémák definiálnak:

termék(termék_ID, márka, típus, gyártási_hely, szállító)
szállító(név, típus, központ_helye, tulajdonos, nagyság, tőke, jövedelem)

Javasoljunk egy DMQL-meghatározást a *termék_hierarchiára*! (Tipp: A **where**-t és a pont (.) jelölést ugyanúgy használhatjuk, mint az SQL-ben.)

(c) Használjunk DMQL-szintaktikát a halmazcsoportosító hierarchiákra azzal a céllal, hogy újradefiniálva finomítsuk a 4.14. példában szereplő *hely* hierarchiát a *kontinens* fogalmi szint hozzáadásával!

(d) A 4.15. példában előforduló *életkor* halmazcsoportosító hierarchiának alternatívájaként, a felhasználó meghatározhat egy, az adatklaszterezési eljárásokon alapuló, műveletvezérelt hierarchiát is. Javasoljunk DMQL-szintaktikát az *életkor*-nak egy ilyen jellegű hierarchiájára vonatkozóan, mondjuk öt klaszter képzésére alapozva!

(e) Fogalmi hierarchia definiálható szabályok halmaza alapján is. Javasoljunk DMQL-szintaktikát a 4.6. példában szereplő *ElektroMind termékek* szabály alapú hierarchiájára a profittartalom alapján!

4.8. Indokoljuk a *szabványosított adatbányászati lekérdező nyelv* kialakításának fontosságát! Milyen lehetséges előnyöket és kihívásokat rejthet magában egy ilyen feladat? Soroljunk fel néhány, erre a területre vonatkozó újabb keletű javaslatot!

4.9. Fogalmazzuk meg az adatbányászati rendszer adatbázis, vagy adattárházrendszerekkel történő integrációi közti különbséget a következő architektúrák esetében: *nincs összekapcsolás*, *laza összekapcsolás*, *közepesen szoros összekapcsolás* és *szoros összekapcsolás*. Melyik architektúrát véljük a leggyakoribbnak, és miért?

4.7. | Irodalom

A szakirodalom számos objektív érdekességi mértéket javasol. Michalski művében [Mic83] egyszerűségi mértékeket találhatunk. A fejezetben ismertetett társítási szabályokra használatos megbízhatóság és megalapozottság mértéket Agrawal, Imielinski és Swami [AIS93a] javasolta. Az általunk ismertetett stratégiát, melynek segítségével a redundáns többszintű társító szabályokat azonosíthatjuk, Srikant és Agrawal [SA95, SA96] javasolta. A [HM91, PS91ab, SG92, AIS93a, MM95, CHY96, KA96] művekben egyéb objektív érdekességi mértékek találhatók. Szubjektív érdekességi mértékekről szóló leírásokat, amelyek figyelembe veszik a felhasználóknak az adatok közti kapcsolatokra vonatkozó elképzeléseit is, a [PSM94, MPSM96, ST96, LHC97] művekben olvashatunk.

A DMQL adatbányászati lekérdező nyelvet Han, Fu, Wang és társai [HFW+96a] javasolták a *DBMiner* adatbányászrendszer számára.

Imielinski, Virmani és Abdulghani [IVA96] egy alkalmazásfejlesztő interfész prototípusaként javasolta a *Discovery Boardot* (korábban *Data Mine*), amely az adatbányászati lekérdezések meghatározására és a szabálykinyerésre egy SQL alapú operátort tartalmaz. Ezt összekapcsolva adatbányászati lekérdező nyelvvel, az MSQL-t Imielinski és Virmani [IV99] indítványozta. A MINE RULE-t, amely egy SQL-szerű egydimenziós társító szabályok bányászatára alkalmas operátor, Meo, Psaila és Ceri [MPC96] javasolta, majd Baralis és Psaila [BP97] egészített ki. Tsur, Ullman, Abitboul, Clifon és társai [TUA+98] egy Datalog szintaktikát [RG00] használó társító szabályokat generáló nyelvet javasoltak, melynek alapját a *lekérdezéshalmok* fogalom adja. Metaszabályokkal történő adatbányászatról Klemettinen, Mannila, Ronkainen és társai [KMR+94], Fu és Han [FH95], Shen,

Ong, Mitbander és Zaniolo [SOMZ96] és Kamber, Han és Chiang [KHC97] műveiben olvashatunk. Egyéb ötletekről, amelyek tartalmazzák a sablonok és a predikátumkonstansok adatbányászatban történő felhasználását is, az [AK93, DT93, LHC97, ST96, SVA97, NLHP98] művek szólnak.

Az SQL relációs adatbázisnyelv szabványosítása fontos faktor, amely hozzájárul a relációs adatbázisrendszerek [SH98] sikeréhez. Az adatbázisrendszerek területén végzett legújabb szabványosítások, például az SQL-3-hoz kapcsolódó munkák és egyéb tevékenységek tovább illusztrálják a szabványos adatbázisnyelv meglétének fontosságát, amellyel hozzájárulnak az adatbázisrendszerek fejlődésének és üzleti alapokra történő helyezésének sikeréhez. A legjelentősebb adatbányászatbeli szabványosítási eredményt a Microsoft Corporation érte el az *OLE DB for Data Mining (DM)* [Mic00] javaslatával. A könyv A) függelékében az *OLE DB for DM* adatbányászati nyelv primitívjeibe történő bevezetés olvasható. A CRISP-DM (Cross-Industry Standard Process for Data Mining) *(http://www.crisp-dm.org/)* egy másik adatbányászathoz kapcsolódó szabványosítási eredmény; jóllehet ez eltávolodik az adatbányászatban megjelenő sokszintű felhasználói igényeket megcímző technológiáktól, az üzleti problémák megoldása felé.

Számos grafikus felhasználói felületet és ábrázolási eszközt fejlesztettek ki, amelyek megtalálhatók a különféle adatbányászati termékekben. Sok adatbányászatról szóló könyv, például a *Data Mining Solutions* Westphaltól és Blaxtontól [WB98] mutat be sok jó példát és vizuális pillanatfelvételt. A megjelenítési technikák áttekintésére Keimtől a *Visual techniques for exploring databases* című művet ajánljuk.

Az adatbányászati rendszerek architektúráját számos kutató vitatta meg konferenciaposzterek és megbeszélések alkalmával. Az adatbányászati nyelvek legújabb tervezési módszerei, mint például a [Mic00, IV99], az on-line elemző bányászat javaslata, például a [Han98], az adatbányászati lekérdezések optimalizálásának tanulmányozása, például [NLHP98, STA98, LNHP99] tekinthetők az adatbányászati rendszerek adatbázisrendszerekkel és adattárházrendszerekkel történő szoros integrációja felé tett lépésként. Számos Sarawagi, Thomas és Agrawal [STA98] által javasolt adatbányászó primitív, például a KWayJoin, a GatherPrune és így tovább, szintén használható az adatbányászat relációs vagy objektumrelációs rendszerekben történő hatékony alkalmazásának építőköveiként.

5. FEJEZET

Fogalomleírás: jellemzés és összehasonlítás

Adatelemzési szempontból az adatbányászatot két kategóriába lehet sorolni: leíró adatbányászat és előrejelző adatbányászat. A *leíró adatbányászat* tömör és összegző módon írja le az adathalmazokat és az adatok érdekes, általános tulajdonságait. Az *előrejelző adatbányászat* elemzi az adatokat azért, hogy egy vagy több modellt hozzon létre, és megkísérli megjósolni az új adathalmazok tulajdonságait.

Az adatbázisok általában nagy mennyiségű, igen részletes adatot tárolnak. A felhasználók azonban tömör, kifejező formában *összegzett* adatokat szeretnek látni. Az ilyen adatleírások átfogó képet adnak egy adatosztályról, vagy elkülönítik azokat hasonló adatosztályok halmazától. Továbbá, a felhasználók szeretik, ha könnyű és rugalmas lehetőségeik vannak, amellyel az adatokat különböző részletezettségi szinten és különböző nézőpontokból lehet látni. Az ilyen leíró adatbányászatot *fogalomleírásnak* hívjuk, amely az adatbányászat fontos részét alkotja.

Ebben a fejezetben megismerjük, hogyan lehet a fogalomleírás eredményesen és hatékonyan végrehajtani.

5.1. | A fogalomleírás

A leíró adatbányászat legegyszerűbb fajtája a *fogalomleírás*. Egy fogalom általában egy adatgyűjteményre utal, például a *gyakori_vásárlók* vagy a *másod-diplomások* és így tovább. A fogelomleírás mint adatbányászati feladat, nem egyszerűen az adatok felsorolása. Ehelyett a **fogalomleírás** az adatok *jellemzésére* és *összehasonlítására* generál leírásokat. Ezt néha **osztályleírásnak** nevezik, amikor a leírandó fogalom objektumosztályokra utal. A **jellemzés** egy adatgyűjtemény rövid, tömör összegzését adja, míg a **fogalom-** vagy **osztályösszehasonlítás** (ami **megkülönböztetésként** is ismert), két vagy több adathalmaz összehasonlításának leírását adja. Mivel a fogalomleírás egyaránt tartalmazza a jellemzést és az összehasonlítást, mindkét feladat végrehajtásának technikáit tanulmányozni fogjuk.

A fogalomleírás közeli kapcsolatban áll az *adatáltalánosítással*. Ha nagy mennyiségű,

adatbázisokban tárolt adatunk van, hasznos lehet, ha a fogalmakat rövid, tömör kifejezésekkel általánosított (inkább mintsem alacsony) absztrakciós szinten írjuk le. Az adatállományok több absztrakciós szintre történő általánosítása lehetővé teszi a felhasználónak, hogy megvizsgálja az adatok általános viselkedését. Vegyük például az *ElektroMind* adatbázist. Az értékesítési igazgató ahelyett, hogy megnézne minden vásárlói ügyletet, inkább magasabb szinten általánosított adatokat akar látni, olyasmiket, mint a területenként összesített ügyfélcsoportok, a vásárlás gyakorisága csoportonként és a bevételek ügyfelenként. Az ilyen többdimenziós, többszintű adatáltalánosítás az adattárházak többdimenziós adatelemzéséhez hasonlítható. Ebben az értelemben a fogalomleírás az adattárházak on-line elemző feldolgozására (OLAP) emlékeztet, amit a 2. fejezetben tárgyaltunk.

Mi a különbség a nagy adatbázisokban lévő fogalomleírások és az on-line elemző feldolgozás között? A kettő közti alapvető különbségek a következők.

- **Komplex adattípusok és összesítések** – Az adattárházak és az OLAP-eszközök többdimenziós adatmodellen alapulnak, adatkocka formában tekintik az adatokat, amelyek dimenziókból (vagy attribútumokból) és mértékekből (összesítőfüggvények) állnak. Azonban ezeknek a rendszereknek a legtöbb kereskedelmi forgalomban lévő verziójánál a dimenziók és mértékek lehetséges adattípusai korlátozottak. Sok jelenlegi OLAP-rendszer csak nem numerikus adatokat enged meg a dimenziókban.[1] Hasonlóan, a jelenlegi OLAP-rendszerekben a mértékeket [például `count()`, `sum()`, `average()`] csak numerikus adatokra lehet alkalmazni. Ezzel szemben fogalom képzésekor az adatbázis-attribútumok különféle típusúak lehetnek, beleértve a numerikus, nem numerikus, térbeli, szöveg vagy kép típusokat. Továbbá az adatbázis-attribútumok összesítése is tartalmazhat kifinomult adattípusokat, például nem numerikus adatok gyűjteményét, térbeli részek egyesítését, képek összetételét, beillesztett szövegeket és csoportosított objektummutatókat. Így az OLAP, a maga korlátaival az attribútum és mérték típusokon, egyszerűsített modellt képvisel az adatelemzés számára. A fogalomleírások az adatbázisokban az attribútumok komplex adattípusait és ezek összesítéseit – szükség szerint – képesek kezelni.
- **Felhasználói vezérlés kontra automatizálás** – Az adattárházakon végzett on-line elemző feldolgozás tisztán a felhasználó által vezérelt eljárás. A dimenziók kiválasztása, az OLAP-műveletek alkalmazása, például a lefúrás, a felgörgetés, a szeletelés, a kockázás mind a felhasználó utasításainak, vezérlésének megfelelően folyik. Bár a kezelés a legtöbb OLAP-rendszerben felhasználóbarát, szükséges, hogy a felhasználó jól értse az egyes dimenziók szerepét. Továbbá az adatok kielégítő leírásának eléréséhez a felhasználónak OLAP-utasítások hosszú sorozatát kell megadnia. Ezzel szemben a fogalomleírás az adatbányászatban jobban automatizált eljárásra törekszik, ami segít eldönteni a felhasználónak, mely dimenziókat (vagy attribútumokat) célszerű bevonni az elemzésbe, és hogy milyen mértékben kell általánosítani az adott adathalmazt, hogy az adatok egy érdekes összegzését állítsuk elő.

1 | Megjegyezzük, hogy a 3. fejezetben megmutattuk, hogyan lehet numerikus adatokból automatikusan fogalmi hierarchiát generálni úgy, hogy numerikus dimenziót alkossanak. Ez a lehetőség azonban az adatbányászat újabb kutatásainak eredménye, és a legtöbb kereskedelmi rendszerben még nem elérhető.

Újabban az adattárházas és OLAP-technológia abba az irányba fejlődik, hogy összetettebb adattípusokat tudjon kezelni, és több tudásfeltáró technikát építsen magába, ahogy azt a 2. fejezetben tárgyaltuk. A technológia fejlődésével várható, hogy további adatbányászati leíró lehetőségek épülnek majd be a jövőbeli OLAP-rendszerekbe.

Ebben a fejezetben a fogalomleírás módszereit tanuljuk meg, beleértve a többszintű általánosítást, összegzést, jellemzést és összehasonlítást. Ezek a módszerek adják az alapot az adatbányászat két fő működési területének megvalósításához: *többszintű jellemzés* és *összehasonlítás*. Ezenkívül megvizsgáljuk a fogalomleírások többféle formában való megjelenítésének technikáit, beleértve a táblázatokat, ábrákat, grafikonokat és szabályokat.

5.2. | Adatáltalánosítás és összegzés alapú jellemzés

Az adatbázisban tárolt adatok és objektumok egyszerű fogalmi szinten gyakran részletes információkat tartalmaznak. Például az *árucikk* reláció az *értékesítés* adatbázisban tartalmazhat attribútumokat olyan alacsony szintű árucikk információk leírására, mint például *árucikk_ID, név, márka, kategória, szállító, gyártási_hely* és *ár*. Nagy adathalmaz esetén hasznos lehet, ha az adatokat összegezni tudjuk, és magasabb fogalmi szinten be tudjuk mutatni. Például, a karácsonyi vásár értékesítéseihez kapcsolódó nagy mennyiségű adat összegzése ezeknek az adatoknak az általános leírását adja, ami nagyon hasznos lehet az értékesítési és marketingigazgatóknak. Ehhez az adatbányászat egy fontos funkciója szükséges: az *adatáltalánosítás*.

Adatáltalánosítás során az adatbázisban lévő nagy mennyiségű, a feladat szempontjából lényeges adatokat egy viszonylag alacsony fogalmi szintről magasabb szintre absztraháljuk. Nagy adathalmazok hatékony és rugalmas általánosításának módszereit kétféle megközelítés alapján lehet kategorizálni: (1) adatkocka- (vagy OLAP-) szemlélet, és (2) attribútumorientált indukció. Az adatkocka-szemléletet a 2. fejezetben írtuk le. Ebben a fejezetben az attribútumorientált indukciós megközelítést tárgyaljuk.

5.2.1. | Az attribútumorientált indukció

Az adatáltalánosításnak és összegzés alapú jellemzésnek az attribútumorientált indukciós (attribute-oriented induction, AOI) megközelítését először 1989-ben javasolták, pár évvel az adatkockaszemlélet megjelenése előtt. Az adatkockaszemlélet egy adattárház alapú, előszámítás-orientált, megvalósított nézet szerinti megközelítésnek tekinthető, amely *off-line* összesítéseket hajt végre, mielőtt egy OLAP- vagy adatbányászati lekérdezést feldolgozna. Ezzel szemben az *attribútumorientált indukciós* megközelítés, legalábbis az eredeti javaslat szerint, relációs adatbázis lekérdezésorientált, általánosításon alapuló, *on-line* adatelemző technika. Nincs azonban éles határ a két megközelítés között, vagyis az on-line összesítéseken, illetve az off-line előszámításon alapulók között. Bizonyos összesítések az adatkocka esetén on-line számíthatók, míg többdimenziós térbeli off-line előszámítások felgyorsíthatják az attribútumorientált indukciót is.

Először bemutatjuk az attribútumorientált indukciót. Majd részletesen elemezzük ezt a megközelítést, változataival és kiterjesztéseivel együtt.

Az attribútumorientált indukció alapötlete az, hogy először relációs adatbázis lekérdezésével összegyűjtjük a feladat szempontjából lényeges adatokat, majd végrehajtjuk az általánosítást úgy, hogy a lényeges adathalmazban minden attribútumra megvizsgáljuk a különböző értékek számát. Az általánosítást *attribútumelhagyással* vagy *attribútumáltalánosítással* hajtjuk végre, míg az összesítés úgy történik, hogy az azonos általánosított sorokat összefésüljük, a megfelelő mértékeket pedig halmozzuk. Ezzel csökkentjük az általánosított adathalmaz méretét. Az eredményül kapott általánosított relációt a felhasználónak megjelenítendő, különböző formákba képezhetjük le, például ábrák vagy szabályok segítségével.

A következő példasorozat az attribútumorientált indukció folyamatát mutatja be.

5.1. példa | Jellemzés megadása DMQL adatbányászati lekérdezéssel. Tegyük fel, hogy a felhasználó a *Csúcs_Egyetem* adatbázisban a *másoddiplomások* általános jellemzőit szeretné leírni, adottak a *név, nem, szak, születési_hely, születési_dátum, lakóhely, telefon#* (*telefon_szám*) és *átlag* (*átlag_érdemjegy*) attribútumok. Ehhez a jellemzéshez az adatbányászati lekérdezést a következőképpen lehet DMQL adatbányászati lekérdező nyelven megfogalmazni:

use Csúcs_Egyetem_AB
mine characteristics as "Természettudományi_hallgatók"
in relevance to név, nem, szak, születési_hely, születési_dátum, lakóhely, telefon#, átlag
from hallgató
where státus **in** "másoddiplomás"

Megnézzük, hogyan lehet erre a tipikus adatbányászati lekérdezésre alkalmazni az attribútumorientált indukciót a jellemző leírások bányászatára. ■

Mi az attribútumorientált indukció első lépése? Először az **adatáttekintést** kell végrehajtani, az attribútumorientált indukciót *megelőzően*. Ez a lépés a feladat szempontjából lényeges adatok meghatározásának felel meg, amint azt a 4. fejezetben leírtuk. Az adatokat az adatbányászati lekérdezésben szereplő információk alapján gyűjtjük. Mivel az adatbányászati lekérdezés általában csak az adatbázis egy részét érinti, a megfelelő adathalmazt választva nem csak hatékonyabb lesz a bányászat, de értelmesebb eredményt is kapunk, mintha az egész adatbázisban bányásztunk volna.

A felhasználónak nehéz lehet a lényeges attribútumok halmazának (azaz a kibányászandó attribútumoknak, amelyek a DMQL **in relevance to** tagmondatában szerepelnek) megadása. A felhasználó esetleg csak néhány attribútumot választ, amelyről úgy érzi, hogy lényeges, ezért mások kimaradhatnak, amelyek pedig szintén szerepet játszhatnak a leírásban. Például tegyük fel, hogy a *születési_hely* attribútumot a *város*, a *megye_vagy_tartomány* és az *ország* attribútumok határozzák meg. Ezek közül az attribútumok közül a felhasználó csak a *várost* választotta. A *születési_hely* dimenzió általánosításának lehetővé tételére a többi olyan attribútumot is be kell vonni a választásba, amelyek ezt a dimenziót meghatározzák. Másképpen fogalmazva, a rendszernek automatikusan be kell vennie a *megye_vagy_tartomány* és az *ország* attribútumokat a lényegesek közé, lehetővé téve ezzel a város magasabb fogalmi szintre történő általánosítását az indukciós eljárás során.

Egy másik szélsőséges eset, amikor a felhasználó túl sok attribútumot ad meg, kijelölve az összes lehetséges attribútumot az **in relevance to** * tagmondattal. Ebben az esetben a **from** tagmondatban megadott reláció összes attribútuma részt vesz az elemzésben. Ezen attribútumok közül több valószínűleg nem járul hozzá az érdekes leíráshoz. Az 5.3. alfejezet módszereket mutat az ilyen esetek kezelésére, a statisztikailag lényegtelen vagy kevésbé lényeges attribútumok kiszűrésével a leíró adatbányászati eljárásból.

Mit jelent a **where** *státus* **in** *"másoddiplomás" tagmondat?* Ebből a **where** tagmondatból arra következtethetünk, hogy a *státus* attribútumhoz létezik fogalmi hierarchia. Az ilyen fogalmi hierarchia az alacsony szintű *státus* értékeket, például *"M.Sc."*, *"M.A."*, *"M.B.A."*, *"Ph.D."*, *"B.Sc."*, *"B.A."*, magasabb fogalmi szintekbe szervezi, például *másoddiplomás* és *alapképzéses*. A fogalmi hierarchiák ilyenfajta használata nem fordul elő a hagyományos relációs lekérdező nyelvekben, viszont megszokott az adatbányászati lekérdező nyelvekben.

5.2. példa | Adatbányászati lekérdezés átalakítása relációs lekérdezéssé. Az 5.1. példában bemutatott adatbányászati lekérdezés az alábbi relációs lekérdezéssé alakítható át, amely összegyűjti a feladat szempontjából lényeges adatokat:

use Csúcs_Egyetem_AB
select név, nem, szak, születési_hely, születési_dátum, lakóhely, telefon#, átlag
from hallgató
where státus **in** {"M.Sc.", "M.A.", "M.B.A.", "Ph.D."}

Az átalakított lekérdezést végrehajtva a *Csúcs_Egyetem_AB* relációs adatbázison az 5.1. táblázatban szereplő adatokat kapjuk vissza. Ezt a táblázatot (a feladat szempontjából lényeges) **kezdeti munkarelációnak** hívjuk. Ezeken az adatokon hajtjuk végre az indukciót. Megjegyezzük, hogy minden sor valójában attribútum–érték párokból álló konjunkció. Ezért a reláció egy sorára úgy tekinthetünk, mint egy konjunkciós szabályra, és a reláción alkalmazott indukcióra pedig mint ezeknek a szabályoknak az általánosítására. ■

5.1. táblázat | Kezdeti munkareláció: a feladat szempontjából lényeges adatok gyűjteménye

név	nem	szak	születési_hely	születési_dátum	lakóhely	telefon#	átlag
Jim Woodman	F	informatika	Vancouver, BC, Kanada	8-12-76	3511 Main St. Richmond	687-4598	3,67
Scott Lachance	F	informatika	Montreal, Que, Kanada	28-7-75	345 1st Ave. Richmond	253-9106	3,70
Laura Lee	N	fizikus	Seattle, WA, USA	25-8-70	125 Austin Ave. Burnaby	420-5232	3,83
	...	...	...	...	...	...	...

Az adatok már készen állnak az attribútumorientált indukcióra, de hogyan hajtható végre? Az attribútumorientált indukció alapvető művelete az *adatáltalánosítás*, amelyet a kezdeti munkareláción az alábbi két lehetőség valamelyikét választva hajtható végre: az egyik az *attribútum elhagyása*, a másik az *attribútum általánosítása*.

Az **attribútum elhagyása** a következő szabályon alapul. *Ha egy attribútumnak a kezdeti munkarelációban nagy számú különböző értéke van, de* (1) *nincs általánosító operátor az attribútumon (azaz nincs fogalmi hierarchia definiálva az attribútumhoz)* vagy (2) *annak magasabb szintű fogalma más attribútumokkal van kifejezve, akkor az attribútumot célszerű törölni a munkarelációból.*

Milyen okfejtés rejlik a szabály mögött? Egy attribútum–érték pár az általánosított sor vagy szabály egy konjunkcióját képviseli. Egy konjunkció elhagyása kiiktat egy megszorítást, és így általánosítja a szabályt. Ha – mint az (1) esetben – egy attribútumnak nagyszámú különböző értéke van, de nincs hozzá általánosító operátor, az attribútumot törölni kell, mert nem általánosítható, és amennyiben megtartanánk, az nagyszámú konjunkció megtartását is jelentené, ami ellentmond annak a célnak, hogy tömör szabályokat generáljunk. Másrészt, tekintsük a (2) esetet, ahol az attribútum magasabb szintű fogalma más attribútumokkal kifejezhető. Példaként tegyük fel, hogy a szóban forgó attribútum az utca, amelynek magasabb szintű fogalmát a (*város, megye_vagy_tartomány, ország*) attribútumok képviselik. Az *utca* elhagyása egy általánosító operátor alkalmazásával egyenértékű. Ez a szabály megfelel annak az általánosítási szabálynak, ami *feltételeldobásként* ismert a gépi tanulásról szóló irodalomban a *példákból tanulás* témakörben.

Az **attribútum általánosítása** a következő szabályon alapul: *ha a kezdeti munkarelációban egy attribútumnak nagyszámú különböző értéke van, és létezik az általánosító operátorok egy halmaza az attribútumon, akkor ki kell választani és alkalmazni kell az attribútumra egy általánosító operátort.* Ez a szabály a következő okfejtésen alapul. Egy általánosító operátor használata a munkareláció egy sorában vagy szabályában lévő attribútum értékének általánosítására, a szabályt olyanná teszi, hogy egy sorának több kezdeti adatsor felel meg, és így általánosítja a fogalmat, amit képvisel. Ez megfelel annak az általánosítási szabálynak, amit a *példákból tanulásnál* az *általánosítási fa megmászásaként* vagy a *fogalmi fában való felemelkedésként* ismerünk.

Mindkét szabály, az *attribútum elhagyása* és az *attribútum általánosítása* is igényli, hogy ha nagyszámú különböző érték tartozik egy attribútumhoz, akkor további általánosítást kell alkalmazni. Ez felveti a kérdést: mikor tekinthető soknak egy attribútum esetén *„a nagyszámú különböző érték"*?

Az attribútumoktól vagy az érintett alkalmazástól függően a felhasználó bizonyos attribútumokat alacsony absztrakciós szinten, míg másokat magasabb általánosítási szinten kíván tartani. Annak vezérlése, hogy egy attribútumot milyen magas szintre kell általánosítani, általában erősen szubjektív. Ennek az eljárásnak a vezérlését **attribútum-általánosítási vezérlésnek** nevezzük. Ha egy attribútumot túl magas szintre általánosítunk, az túláltalánosításhoz vezet, és az eredményül kapott szabályok nem lesznek elég informatívak. Másrészt, ha egy attribútum nincs elégséges mértékben általánosítva, aluláltalánosításról beszélünk, és a kapott szabályok szintén nem lesznek informatívak. Ezért az attribútum alapú általánosításban törekedni kell az egyensúlyra.

Az általánosítási eljárás vezérlésére több lehetséges mód is van. Két szokásos megközelítést mutatunk be.

Az első technika, amit **attribútumáltalánosításos küszöbvezérlésnek** hívnak, vagy egy közös küszöbértéket állít be az attribútumok számára, vagy minden egyes attribútumhoz rendel egy küszöbértéket. Ha egy attribútumra a különböző értékek száma nagyobb, mint a megfelelő küszöbérték, további attribútumelhagyást vagy általánosítást kell végre-

hajtani. Az adatbányászati rendszereknek általában van egy alapértelmezett küszöbértékük (rendszerint 2 és 8 közötti érték), és megengedik, hogy a szakértők vagy a felhasználók módosíthassák a küszöbértékeket. Ha a felhasználó úgy érzi, hogy egy bizonyos attribútumnál az általánosítás túl magas szintet ért el, megnövelheti a küszöbértéket. Ez megfelel az attribútum mentén történő lefúrásnak. A reláció további általánosítására a felhasználó csökkentheti is egy bizonyos attribútum küszöbértékét, ami az attribútum mentén történő felgörgetésnek felel meg.

A második technika az **általánosított relációs küszöbvezérlés**, amely az általánosított relációra állít be küszöbértéket. Ha a (különböző) sorok száma az általánosított relációban nagyobb, mint a küszöbérték, akkor további általánosítást kell végrehajtani, egyébként pedig nem. Ilyen küszöbérték szintén előre beállítható az adatbányászati rendszerekben (általában 10 és 30 közötti érték), vagy beállítható szakértő vagy felhasználó által és módosíthatónak kell lennie. Például, ha a felhasználó úgy érzi, hogy az általánosított reláció túl kicsi, akkor megnövelheti a küszöbértéket, ami tulajdonképpen lefúrás. Egyébként a reláció további általánosításával csökkenteni tudja a küszöbértéket, ami egyben felgörgetésnek felel meg.

A két technika egymás után is alkalmazható: először az attribútumos küszöbvezérlő technikával általánosíthatjuk az egyes attribútumokat, majd a relációs küszöbvezérléssel tovább csökkenthetjük az általánosított reláció méretét. Bármelyik általánosítási vezérlőtechnikát alkalmazzuk is, a felhasználó számára meg kell engedni az általánosítási küszöbértékek módosítását a számára érdekes fogalomleírás eléréséhez.

Sok adatbázis alapú indukciós eljárásnál a felhasználó mennyiségi vagy statisztikai információra kíváncsi a különböző absztrakciós szintű adatokról. Ezért az indukciós eljárás során fontos a darabszámok és más összesített értékek gyűjtése. Elméletileg ez a következőképpen történik. Egy speciális mérték vagy numerikus attribútum rendelődik minden egyes adatbázissorhoz, nevezetesen a `count` (darabszám) összesítőfüggvény. Ennek kezdeti értéke 1, a kezdeti munkareláció minden egyes sorában. Az attribútumok elhagyása és általánosítása során a kezdeti munkareláció soraiból az általánosítás eredményeképpen *azonos sorok* csoportjai jönnek létre. Ebben az esetben az egy csoportba tartozó azonos sorokat egyesíteni kell egy sorrá. Ennek az új, általánosított sornak a darabszámértéke a kezdeti munkareláció azon sorainak a teljes száma, amelyeket ez az általánosított sor képvisel (azaz amelyeket egyesítettek). Tegyük fel például, hogy az attribútumorientált indukció során a kezdeti munkareláció 52 adatsorának általánosítása ugyanaz a T sor. Vagyis ennek az 52 sornak az általánosítása a T sor 52 azonos előfordulását eredményezte. Ennek az 52 azonos sornak az egyesítése alkotja a T sor egyetlen előfordulását, amelynek a darabszámértéke 52 lesz. További közkedvelt összesítőfüggvények – többek között – a `sum` és az `avg`. Egy általánosított sor esetén a `sum` a kezdeti munkareláció azon soraira, amelyekből az általánosított sor létrejött, egy adott numerikus attribútum értékének az összegét tartalmazza. Tegyük fel, hogy a T sor tartalmazza a `sum`(*darab_eladott*) összesítőfüggvényt. A T sor `sum` értékét ebben az esetben az 52 sorban szereplő eladott mennyiségek teljes összegére kell állítani. Az `avg` (átlag) összesítőfüggvény értékét az `avg` = `sum`/`count` formula alapján kell kiszámítani.

5.3. példa | Attribútumorientált indukció. Bemutatjuk az attribútumorientált indukciót az 5.2. példából származó, az 5.1. táblázatban bemutatott kezdeti munkareláción. Az általánosítási eljárás a reláció minden egyes attribútumára a következő:

(1) *név*: Mivel a *név*nek sok különböző értéke van és nincs definiálva általánosító operátor, ezt az attribútumot elhagyjuk.
(2) *nem*: A *nem*nek csak 2 különböző értéke van, ezt az attribútumot megtartjuk és általánosítást nem hajtunk végre rajta.
(3) *szak*: Tegyük fel, hogy definiált egy fogalmi hierarchia, ami lehetővé teszi, hogy a *szak* attribútum a {*bölcsészet&természettudomány*, *mérnök*, *üzlet*} értékekre legyen általánosítható. Azt is tegyük fel, hogy az attribútum-általánosítási küszöbérték 5 és több mint 20 különböző *szak* érték van a kezdeti munkarelációban. Az attribútumáltalánosítás és az attribútum-általánosítási vezérlés által a *szak* attribútumáltalánosításra került, felemelkedve az adott fogalmi hierarchiában.
(4) *születési_hely*: Ennek az attribútumnak is nagyszámú különböző értéke van, ezért szeretnénk általánosítani. Tegyük fel, hogy létezik fogalmi hierarchia a *születési_hely*hez, amely a következőképpen van definiálva: *város* <*megye_vagy_tartomány* < *ország*. Ha a kezdeti munkarelációban az *ország* különböző értékeinek száma nagyobb, mint az attribútum-általánosítási küszöbérték, akkor a *születési_hely*et el kell hagyni, mivel bár létezik hozzá általánosító operátor, de arra nem teljesül az általánosítási küszöbfeltétel. Ha ellenben az *ország* különböző értékeinek száma kevesebb, mint az attribútum-általánosítási küszöbérték, akkor a *születési_hely* általánosítható *születési_országgá*.
(5) *születési_dátum*: Tegyük fel, hogy van hierarchia, amely általánosítani tudja a *születési_dátum*ot *életkor*rá és az *életkort életkor_tartomány*ná, és az *életkor_tartomány*ok (vagy intervallumok) száma kicsi az attribútum-általánosítási küszöbhöz viszonyítva. Így a *születési dátum* általánosítása megtörténhet.
(6) *lakóhely*: Tegyük fel, hogy a *lakóhely*et a következő attribútumok határozzák meg: *házszám, utca, lakóhely_város, lakóhely_megye_vagy_tartomány, lakóhely_ország*. A különböző értékek száma a *házszám* és az *utca* esetén valószínűleg nagyon magas, mivel ezek a fogalmak igen alacsony szintűek. Ezért a *házszám* és az *utca* attribútumokat el kell hagyni és a *lakóhely*et általánosítani kell *lakóhely_város*ra, ami kevesebb különböző értéket tartalmaz.
(7) *telefon#*: Akárcsak a korábbi *név* attribútumnak, ennek is túl sok különböző értéke van, ezért el kell hagyni az általánosítás során.
(8) *átlag*: Tegyük fel, hogy van fogalmi hierarchia az *átlag* attribútumon, amely az *átlagot* numerikus intervallumokba csoportosítja, például {3,75–4,0, 3,5–3.75, ...}, amit végül leíró értékekbe csoportosítunk: {*kitűnő, nagyon_jó*,...}. Így az attribútum általánosítható.

Az általánosító eljárás azonos sorok csoportjait fogja eredményezni. Például az 5.1. táblázat első két sora az általánosítás során azonossá válik (nevezetesen az 5.2. táblázat első sorában szereplő sorrá). Az ilyen azonos sorokat összevonjuk, miközben a *darabszám* értékeket összegezzük. Az eljárás az 5.2. táblázatban látható általánosított relációt eredményezi.

Az OLAP szóhasználata szerint a darabszámot tekinthetjük *mértéknek*, a többi attribútumot pedig *dimenziónak*. Megjegyezzük, hogy az olyan összesítőfüggvényeket, mint például a `sum`, numerikus attribútumokra alkalmazhatjuk, például *fizetés* vagy *értékesítés*. Ezekre az attribútumokra *mértékattribútumként* hivatkozhatunk. ■

5.2. táblázat | Általánosított reláció, amit az 5.1. táblázat adataiból az attribútumorientált indukció során kaptunk

nem	szak	születési_ország	életkor_tartomány	lakóhely_város	átlag	darabszám
F	Természettudomány	Kanada	20...25	Richmond	nagyon_jó	16
N	Természettudomány	Külföldi	25...30	Burnaby	kitűnő	22
...	...	...	...	...	...	...

A kapott általánosítás megjelenítésének megvalósítási technikáit és módszereit a következő alfejezetekben tárgyaljuk.

5.2.2. | Az attribútumorientált indukció hatékony megvalósítása

Hogyan lehet az attribútumorientált indukciót ténylegesen megvalósítani? Az előző alfejezetben egy bevezetést adtunk az attribútumorientált indukcióra. Az általános eljárást az 5.1. ábra összegzi. Ennek az algoritmusnak a hatékonyságát az alábbiak szerint elemezhetjük:

- Az algoritmus 1. lépése alapvetően egy relációs lekérdezés, amely összegyűjti a feladat szempontjából lényeges adatokat a W **munkarelációba**. A végrehajtás hatékonysága a lekérdezésfeldolgozó módszertől függ. Az adatbázisrendszerek sikeres megvalósításai és kereskedelmi elterjedtségük miatt ennek a lépésnek várhatóan jó lesz a teljesítménye.
- A 2. lépés statisztikákat gyűjt a munkareláción. Ez a reláció legfeljebb egyszeri végigolvasását igényli. A kívánt minimális szint és a (v, v') leképezéspárok minden egyes attribútumra való meghatározásának költsége függ az egyes attribútumok különböző értékeinek számától és kisebb mint n, ami a kezdeti relációban lévő sorok száma.
- 3. lépés eredményezi a P **elsődleges relációt**. Ez úgy zajlik, hogy beszúrjuk az általánosított sorokat a P-be. Összesen n sor van W-ben és p sor P-ben. A W minden egyes t sorában helyettesítjük az attribútumértékeket az eredményül kapott leképezéspárok alapján. Az eredmény egy t' általánosított sor. Ha az (a) változatot alkalmazzuk minden egyes t' esetén $O(\log p)$-t tesz ki a sorbeszúrásnak vagy a darabszámnövelés helyének a megtalálása. Így az összes általánosított sorra a teljes időbonyolultság $O(n \times \log p)$. Ha a (b) változatot alkalmazzuk, hogy a darabszámnövelés sorát megtaláljuk, az minden egyes t' esetén $O(1)$-t tesz ki. Így az összes általánosított sorra a teljes időbonyolultsága $O(n)$.

Sok adatelemző feladatnak jókora mennyiségű dimenziót vagy attribútumot kell megvizsgálnia. Például egy interaktív adatbányászati rendszernek esetleg *dinamikusan* be kell vennie, és meg kell vizsgálnia további attribútumokat is, nem csak azokat, amelyek az adatbányászati lekérdezésben szerepelnek. Összetettebb leíró adatbányászati feladatok, mint amilyen az *analitikus jellemzés* (amit az 5.3. alfejezetben tárgyalunk), attribútumok igen nagy halmazára igényel attribútumrelevancia-elemzést. Továbbá, egy felhasználó, akinek kevés tudása van az igazán lényeges adatok halmazáról, esetleg egyszerűen az **in**

Algoritmus: Attribútumorientált indukció. Általánosított jellemzők bányászata relációs adatbázisban, adott felhasználói adatbányászati lekérdezés alapján.

Input: (i) *DB* relációs adatbázis; (ii) *DMQuery* adatbányászati lekérdezés; (iii) *a_list* attribútumlista (az a_i attribútumokat tartalmazza); (iv) *Gen*(a_i), fogalmi hierarchiák halmaza vagy általánosító operátorok az a_i attribútumon; (v) *a_gen_thresh*(a_i) attribútum-általánosítási küszöbérték az egyes a_i-kre.

Output: *P, elsődleges_általánosított_reláció.*

Módszer: A módszert az alábbiakban vázoljuk.

1. $W \leftarrow$ **get_task_relevant_data**(*DMQuery,DB*); // Legyen *W* a feladat szempontjából lényeges adatokat tartalmazó munkareláció.
2. **prepare_for_generalization**(*W*); // A következőképpen valósítjuk meg.
   (a) Végigolvassuk *W*-t és kigyűjtjük minden egyes a_i attribútumra a különböző értékeket. (Megjegyzés: Ha *W* nagyon nagy, ez elvégezhető a *W*-ből vett minta megvizsgálásával.)
   (b) Minden egyes a_i attribútumra meghatározzuk, hogy vajon a_i-t el kell-e hagyni, és ha nem, akkor a rá vonatkozó adott vagy alapértelmezett attribútum-küszöbérték alapján ki kell számítani a minimális kívánt L_i szintet, és meg kell határozni a (v,v') leképezéspárokat, ahol v egy érték a különböző *W*-beli a_i értékek közül, és v' az ennek megfelelő általánosított L_i szintű érték.
3. $P \leftarrow$ **generalization**(*W*);
   A *P elsődleges_általánosított_reláció*, úgy keletkezik, hogy minden egyes *W*-beli v értéket lecserélünk a neki a leképezésben megfelelő v'-re, miközben összegyűjtjük a darabszámot, és kiszámolunk minden más összesítőértéket.
   Ezt a lépést az alábbi két változat egyikével lehet hatékonyan megvalósítani.
   (a) Minden egyes általánosított sorra szúrjuk be ezt a sort a *P* rendezett elsődleges relációba bináris keresést használva; ha a sor már létezik *P*-ben, akkor egyszerűen csak növeljük meg megfelelően a darabszámot és az egyéb összesítőértékeket; egyébként pedig szúrjuk be a sort *P*-be.
   (b) Mivel a különböző értékek száma az elsődleges relációs szinten legtöbbször kicsi, ezért az elsődleges relációt lehet egy *m*-dimenziós tömbként kódolni, ahol *m* a *P*-beli attribútumok száma, és minden egyes dimenzió a megfelelő általánosított attribútumértékeket tartalmazza. Mindegyik tömbelem tartalmazza a megfelelő darabszámértéket és a többi összesítőértéket, ha ilyenek léteznek. Az általánosított sor beszúrása a mértékeknek a megfelelő tömbelemben való összesítése mellett történik.

5.1. ábra | Az attribútumorientált indukció alapalgoritmusa

relevance to * tagmondatot adja meg az adatbányászati lekérdezésben. Az ilyen esetekben felgyorsítja a nagyszámú dimenzió vagy attribútum elemzését, ha az összesítőértékeket előre kiszámítjuk. Ezért az adatkockával történő megvalósítás vonzó alternatívája a fent leírt adatbázis-megközelítésnek.

Az attribútumorientált indukció *adatkockával történő megvalósítása* kétféle módon történhet.

- **Menet közben hozunk létre egy adatkockát az adatbányászati lekérdezéshez** – Ez a módszer dinamikusan hoz létre egy adatkockát a feladat szempontjából lényeges adathalmaz alapján. Ez kívánatos lehet, ha a feladat szempontjából lényeges adathalmaz túl különleges, és nem felel meg egyik előre definiált adatkockának sem, vagy ha nem túl nagy. Mivel az ilyen adatkockát csak a lekérdezés futtatása után lehet kiszámítani, egy ilyen adatkocka létrehozásának fő indítéka a hatékony lefúró analízis megkönnyítése. Az ilyen adatkockánál az elsődleges reláció szintje alá való lefúrás egyszerűen csak a kocka adatainak a visszanyerését igényli, vagy csak kisebb általánosítás végrehajtását a kockában tárolt bizonyos köztes szintű adatokon, ahelyett hogy elemi

szintű adatokon kellene általánosítást végezni. Ez felgyorsítja a lefúrást. Viszont, mivel az attribútumorientált adatáltalánosítás magában foglalja a lekérdezéshez kapcsolódó adatkocka kiszámítását is, ez többlet feldolgozással jár, mint ha egyszerűen csak az elsődleges relációt számítanánk ki, és ezért növeli a válaszidőt. A kettő közötti egyensúly megtalálható, ha kiszámítunk egy kocka struktúrájú „al-elsődleges" relációt, amelyikben az általánosított reláció minden egyes dimenziója néhány szinttel mélyebb, mint az elsődleges reláció szintjei. Ez megkönnyíti a lefúrást ezekre a szintekre, elfogadható tárolási és feldolgozási költségek mellett, viszont a további lefúrás ezeken a szinteken túl továbbra is elemi adatokból való általánosítást igényel. Megjegyezzük, hogy az ilyen további lefúrás valószínűleg inkább helyi, semmint az egész kockára kiterjedő.

- **Előre definiált adatkocka használata** – Ennél a módszernél az adatbányászati lekérdezés végrehajtása előtt létrehozunk egy adatkockát, és ezt az előre definiált kockát használjuk a továbbiakban. Ez a kívánatos megoldás, ha a feladat szempontjából lényeges adatok finomsága megfelel az előre definiált adatkockában lévőnek és a feladat szempontjából lényeges adatok halmaza igen nagy. Mivel ilyenkor az adatkocka előre kiszámított, ez megkönnyíti az attribútumrelevancia-elemzést, az attribútumorientált indukciót, a szeletelést és kockázást, a felgörgetést és a lefúrást. Ennek ára a kocka kiszámításának költsége, és a nem jelentéktelen tárolási költség. A kiszámítási/tárolási költség és az elérési sebesség közötti egyensúly úgy érhető el, hogy a lehetséges megvalósítható részkockáknak csak egy kiválasztott részhalmazát számítjuk ki előre, ahogy ez a 2. fejezetben szerepelt.

5.2.3. | A kapott általánosítás megjelenítése

Az attribútumorientált indukció általánosított leírást vagy ezek halmazát állítja elő. Hogyan lehet ezeket a leírásokat megjeleníteni? A leírásokat számos különböző módon lehet a felhasználóknak megjeleníteni. Az attribútumorientált indukciónál eredményül kapott általánosított leírást a leggyakrabban *általánosított reláció* formájában jelenítjük meg.

5.4. példa | Tegyük fel, hogy attribútumorientált indukciót hajtottunk végre az *ElektroMind* adatbázis *értékesítés* relációján, ami az 5.3. táblázatban szereplő általánosított leírást eredményezte az 1999-es eladásokból. A leírást általánosított reláció formájában mutatjuk be. Az 5.3. példa 5.2. táblázata ugyancsak példa az általánosított relációra. ■

A leírások megjeleníthetők **kereszttáblázat** formájában is. Egy kétdimenziós kereszttáblázatban minden egyes sor valamely attribútum értékeit képviseli, és minden oszlopa egy másik attribútum megfelelő értéke. Egy n dimenziós kereszttáblázatban ($n > 2$), az oszlopok több attribútum megfelelő értékeit képviselik az attribútum–érték csoportok részösszegeivel együtt. Ez a megjelenítés a *táblázatkezelőkhöz* hasonló. Az adatkocka struktúra közvetlenül egyszerűen leképezhető kereszttáblázatba.

5.5. példa | Az 5.3. táblázatban szereplő általánosított reláció háromdimenziós kereszttáblázatba transzformálható az 5.4. táblázat szerint. ■

5.3. táblázat | Általánosított reláció az 1999-es értékesítésre

Hely	Árucikk	Értékesítés (millió Ft)	Darabszám (ezer db)
Ázsia	TV	15	300
Európa	TV	12	250
Észak-Amerika	TV	28	450
Ázsia	Számítógép	120	1000
Európa	Számítógép	150	1200
Észak-Amerika	Számítógép	200	1800

5.4. táblázat | Kereszttáblázat az 1999-es értékesítésről

Hely/árucikk	TV		Számítógép		Mindkettő	
	Értékesítés	Darabszám	Értékesítés	Darabszám	Értékesítés	Darabszám
Ázsia	15	300	120	1000	135	1300
Európa	12	250	150	1200	162	1450
Észak-Amerika	28	450	200	1800	228	2250
Összesen	55	1000	470	4000	525	5000

Az általánosított adatokat grafikusan is lehet ábrázolni oszlopdiagram, kördiagram és grafikon felhasználásával. Az adatelemzésben a grafikonos megjelenítés a népszerű. Az ilyen diagramok és grafikonok 2-D (kétdimenziós) és 3-D (háromdimenziós) adatokat is ábrázolhatnak.

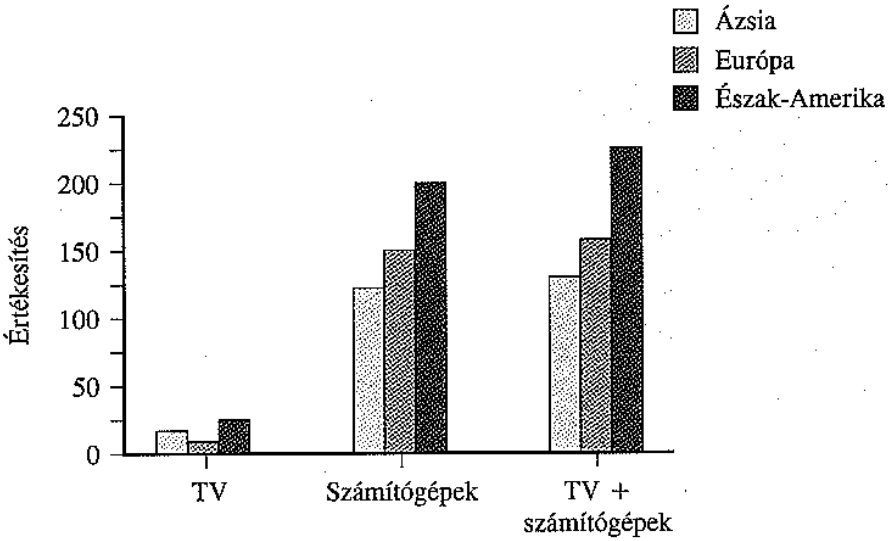
5.6. példa | Az 5.4. táblázatban szereplő eladási adatok kereszttáblázata átalakítható az 5.2. ábrán látható oszlopdiagrammá és az 5.3. ábrán szereplő kördiagrammá. ■

Végül egy háromdimenziós általánosított reláció vagy kereszttáblázat háromdimenziós adatkockaként is ábrázolható. Az ilyen háromdimenziós nézet látványos eszköz a kocka feldolgozásához.

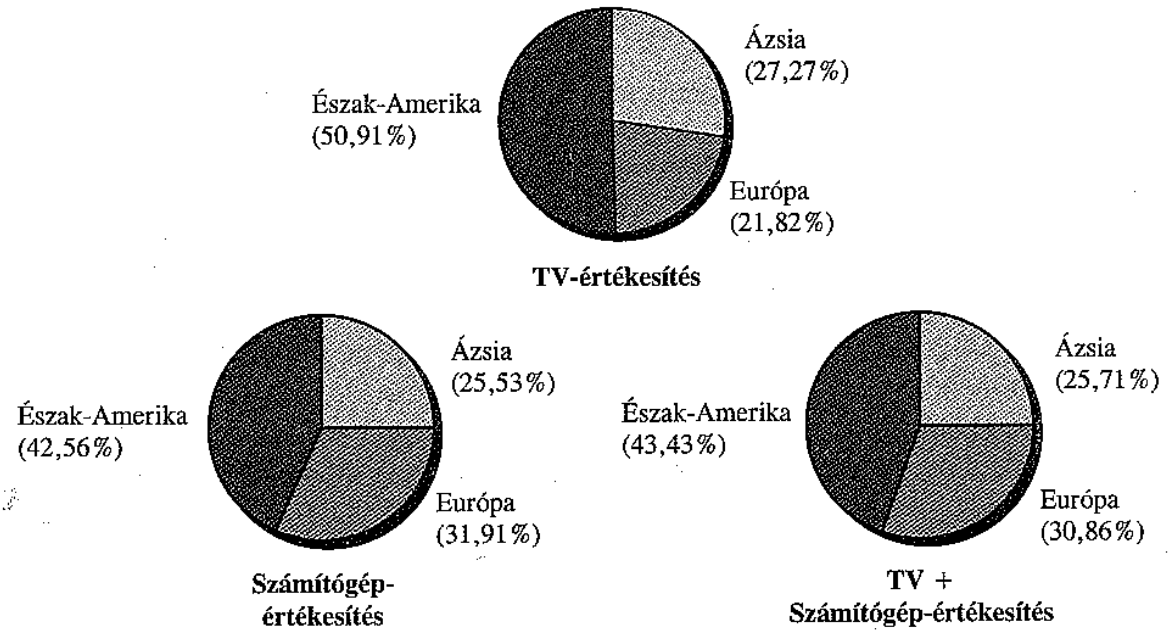
5.7. példa | Tekintsük az 5.4. ábrán szereplő adatkockát az *árucikk, hely, költség* dimenziókra. Egy cella (az ábrán egy kis kockával ábrázolva) *mérete* mutatja a megfelelő cellához tartozó *darabszámot*, míg a cella *árnyalata* használható egy másik mérték megjelenítésére, például a sum(*értékesítés*). A forgatás, fúrás, szeletelés és kockázás műveletek egérkattintással végrehajthatók a kockamegjelenítőn. ■

Egy általánosított relációt logikai szabályokkal is ábrázolhatunk. Tipikusan, minden egyes általánosított sor egy diszjunkciós szabályt képvisel. Mivel nagy adatbázisokban az adatok különböző intervallumokba oszlanak szét, nem valószínű, hogy egyetlen általánosított sor *lefedné*, vagy képviselné a kezdeti munkareláció sorainak vagy *eseteinek* a 100%-át.

Ezért mennyiségi információt kell rendelni minden egyes szabályhoz, például a szabály bal és jobb oldalát kielégítő adatsorok százalékos arányát. Egy logikai szabályt, amelyhez mennyiségi információt rendelünk, **mennyiségi szabálynak** nevezünk.



5.2. ábra | Az 1999-es értékesítés ábrázolása oszlopdiagrammal

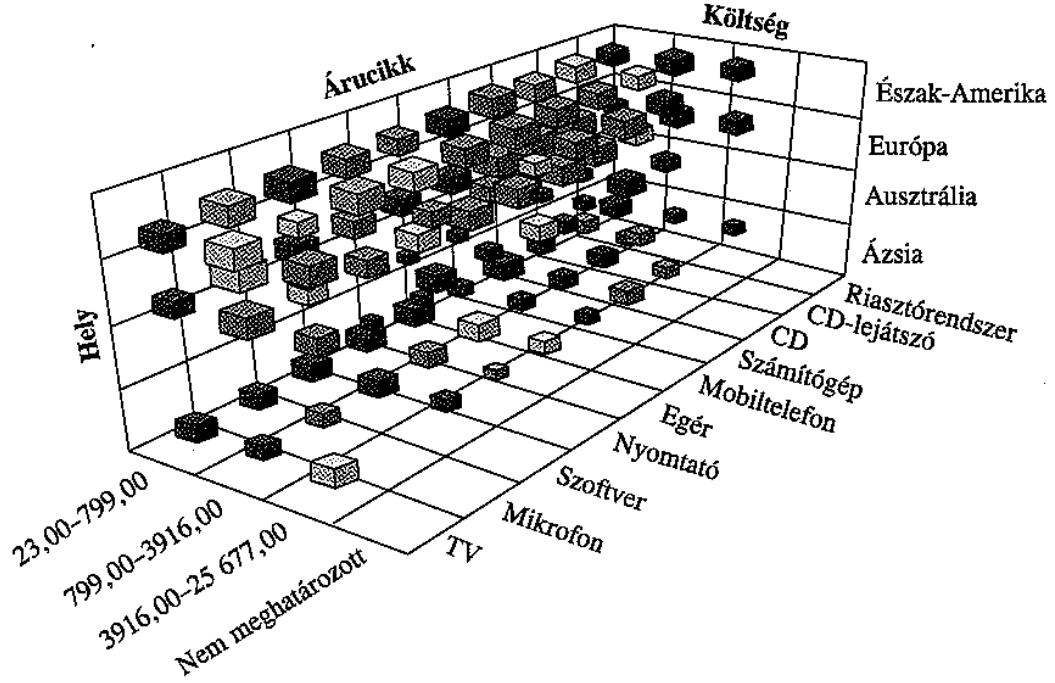


5.3. ábra | Az 1999-es értékesítés ábrázolása kördiagrammal

Mennyiségi jellemző szabály meghatározásához bevezetjük a **t_súly** érdekességi mértéket, amely a szabály minden egyes *diszjunkciójának* vagy a megfelelő általánosított reláció minden egyes *sorának* a *tipikusságát* írja le. A mérték a következőképpen definiálható. Nevezzük *célosztálynak* a jellemzett (vagy szabály által leírt) objektumok halmazát. Legyen q_a a célosztályt leíró általánosított sor. A q_a **t_súlya** a q_a-hoz tartozó kezdeti munkarelációban a célosztálybeli sorok százalékos aránya. Formálisan:

$$t\_súly = count(q_a) / \sum_{i=1}^{n} count(q_i), \tag{5.1}$$

ahol n a célosztályhoz tartozó sorok száma az általánosított relációban; $q_1,\ldots, q_n$ a célosztály sorai az általánosított relációban; és q_a pedig a $q_1,\ldots, q_n$ közül valamelyik. Nyilvánvaló, hogy a t_súly értéke a [0, 1] vagy a [0%, 100%] intervallumba esik.



5.4. ábra | Az 1999-es értékesítés ábrázolása háromdimenziós adatkockanézettel

A **mennyiségi jellemző szabály** ábrázolható akár (1) logikai formában úgy, hogy a megfelelő t_súly értéket hozzárendeljük a célosztályt lefedő minden egyes diszjunkcióhoz; vagy (2) relációs tábla, illetve kereszttáblázat formában úgy, hogy a *darabszám* értékeket ezekben a táblákban lecseréljük a célosztály-sorok megfelelő t_súly értékeire.

A mennyiségi jellemző szabály minden egyes diszjunkciója egy feltételt képvisel. Általánosságban ezen feltételek diszjunkciója a célosztály egy *szükséges* feltételét alkotja, mivel a kapott feltétel a célosztály összes esetén alapul; vagyis a célosztály minden sorának ki kell elégítenie ezt a feltételt. Azonban a szabály esetleg *nem elégséges* feltétele a célosztálynak, mivel azonos feltételt kielégítő sor másik osztályhoz is tartozhat. Ezért a szabályt a következő formában kell kifejezni:

$$\forall X, \textit{cél\_osztály}(X) \Rightarrow \textit{feltétel}_1\ (X)[t : w_1] \vee \ldots \vee \textit{feltétel}_m\ (X)[t : w_m] \qquad (5.2)$$

A szabály azt mutatja, hogy ha X *cél_osztály*-beli, akkor w_i valószínűséggel X kielégíti a *feltétel*$_i$-t, ahol w_i az i feltételhez vagy diszjunkciós taghoz tartozó t_súly érték és i az $\{1, \ldots, m\}$ halmazból való.

5.8. példa | Az 5.4. táblázatban szereplő kereszttáblázatot logikai szabály formára lehet hozni. Legyen a célosztály a számítógép árucikkek halmaza. A megfelelő jellemző szabály logikai formában:

$$\forall X, \textit{árucikk}(X) = \text{“}\textit{számítógép}\text{”} \Rightarrow$$
$$(\textit{hely}(X) = \text{“}\textit{Ázsia}\text{”})[t : 25{,}00\%] \vee (\textit{hely}(X) = \text{“}\textit{Európa}\text{”})[t : 30{,}00\%] \vee$$
$$(\textit{hely}(X) = \text{“}\textit{Észak\_Amerika}\text{”})[t : 45{,}00\%] \qquad (5.3)$$

Megfigyelhető, hogy az első, 25%-os t_súly értéket úgy kapjuk meg, hogy 1000-t, az (*Ázsia, számítógép*) párnak megfelelő darabszámértéket elosztjuk 4000-rel, ami az (összes_terület, *számítógép*) párnak megfelelő darabszámérték. (Vagyis 4000 az összes eladott számítógép számát jelenti.) A másik két diszjunkciós tag t_súlya hasonlóan keletkezett. Más célosztályokra hasonló módon számíthatók a mennyiségi jellemző szabályok. ■

Hogyan tudják az adatbányászati rendszerek felhasználni a t_súlyt és az érdekességi mértékeket általában úgy, hogy csak a tárgyilagosan érdekesnek értékelt fogalomleírásokat jelenítsék meg? Erre a célra beállítható egy küszöbérték. Például, ha egy általánosított sor t_súly értéke kisebb, mint a küszöbérték, akkor a sor az adatbázis elhanyagolható részét képviseli, ezért figyelmen kívül hagyható, mint érdektelen. Az ilyen jelentéktelen sorok mellőzése nem jelenti azt, hogy törölni kellene őket a köztes eredményekből (azaz az elsődleges általánosított relációból vagy az adatkockából a megvalósítástól függően), mivel még a felhasználó által más dimenziókon és absztrakciós szinteken végrehajtott interaktív lefúrások vagy felgörgetések általi rákövetkező további adatfeltárásokhoz hozzájárulhatnak. Az ilyen küszöbértékre úgy hivatkozhatunk, mint **fontossági küszöb** vagy **megalapozottsági küszöb**, ahol az utóbbi kifejezés gyakran használatos a társítási szabály bányászatával kapcsolatban.

5.3. | Analitikus jellemzés: attribútumrelevancia-elemzés

Mi a teendő, ha bizonytalan vagyok, hogy mely attribútumok kerüljenek be az osztályjellemzésbe és az osztály-összehasonlításba? Túl sok attribútum előírása a rendszer jelentős lelassulásához vezethet. Az attribútumrelevancia-elemzés mértékei felhasználhatók a lényegtelen vagy kevésbé lényeges attribútumok azonosítására, amelyeket kihagyhatunk a fogalmi leíró eljárásból. Ha bevesszük ezt az előfeldolgozási lépést az osztályjellemzésbe vagy -összehasonlításba, akkor *analitikus jellemzésként,* illetve *analitikus összehasonlításként* említhetjük. Ez az alfejezet az attribútumrelevancia-elemzés általános módszerét, valamint annak az attribútumorientált indukcióba való beillesztését írja le.

5.3.1. | Miért végzünk attribútumrelevancia-elemzést?

Az adattárházakban és OLAP-eszközökben a többdimenziós adatelemzés számára végzett osztályjellemzés első korlátja a komplex objektumok kezelése. Ezt tárgyaltuk az 5.2. alfejezetben. A második korlát az *automatikus általánosítási eljárás hiánya*: a felhasználónak kifejezetten meg kell mondania a rendszernek, hogy mely dimenziókat kell bevenni az osztályjellemzésbe, és az egyes dimenziókat milyen magas szinten kell az általánosítani. A felhasználónak kell meghatároznia az általánosítás vagy részletezés minden egyes lépését minden egyes dimenzió esetén.

A felhasználó számára általában nem nehéz javaslatot tenni arra, hogz az adatbányászati rendszer milyen magas szintre általánosítsa az egyes dimenziókat. Például beállíthat ehhez attribútum-általánosítási küszöbértéket, vagy előírhatja, hogy az adott dimenziónál milyen szintet kell elérni, például ilyen paranccsal:

generalize dimension *hely* **to the** *ország* **level.**

Határozott felhasználói rendelkezés hiányában az adatbányászati rendszer egy alapértelmezett, például 2 és 8 közötti értéket állíthat be, ami azt teszi lehetővé, hogy az egyes dimenziókat olyan szintre általánosítsuk, amely csak 2 és 8 közötti különböző értéket tartalmaz. Ha a felhasználó nem elégedett az általánosítás aktuális szintjével, akkor megadhat dimenziókat, amelyekre a lefúrás vagy a felgörgetés műveletét kell alkalmazni.

Nem egyszerű azonban meghatározni, hogy mely dimenziókat vegyük be az osztályjellemzők elemzésébe. Az adatrelációk gyakran 50–100 attribútumot is tartalmaznak, és a felhasználónak kevés ismerete van arról, hogy mely attribútumokat vagy dimenziókat kell kiválasztani a hatékony adatbányászathoz. Ha túl kevés attribútumot von be az elemzésbe, az eredményül kapott bányászati leírások hiányosak lesznek. Előfordulhat azonban az is, hogy túl sok attribútumot választunk az elemzéshez (például, az **in relevance to** * jelöléssel, amely az adott reláció összes attribútumát magába foglalja).

Olyan módszereket kell tehát bevezetnünk az attribútum- (vagy dimenzió-) relevancia-elemzés végrehajtásához, amellyel ki tudjuk szűrni a statisztikailag lényegtelen vagy kevésbé lényeges attribútumokat, és meg tudjuk tartani vagy akár rangsorolni a leglényegesebb attribútumokat a folyamatban lévő leíró bányászati feladat számára. Az olyan osztályjellemzést, amely attribútum-/dimenziórelevancia-elemzést tartalmaz, **analitikus jellemzésnek** hívjuk. Az ilyen elemzést tartalmazó osztályösszehasonlítást pedig **analitikus összehasonlításnak** nevezzük.

Intuitívan egy attribútumot vagy dimenziót akkor tekintünk egy adott osztály szempontjából *erősen relevánsnak*, ha az attribútum vagy dimenzió értékei valószínűleg használhatók arra, hogy az egyik osztályt megkülönböztessük a másiktól. Például nem valószínű, hogy az autó színe használható arra, hogy a drága autót megkülönböztessük az olcsótól, de a fajta, gyártmány, típus és a hengerek száma már relevánsabb attribútumoknak tűnnek. Továbbá, még ugyanazon dimenzión belül is a különböző fogalmi szinteknek szignifikánsan eltérő erejük lehet az osztályok egymástól való megkülönböztetésében. Például a *születési_dátum* dimenzióban a *születési_nap* és a *születési_hónap* nem valószínű, hogy erősen releváns az alkalmazottak *fizetése* szempontjából. Viszont a *születési_dekád* (azaz életkor intervallum) lényeges lehet az alkalmazotti *fizetésnél*. Ez azt jelenti, hogy a dimenziórelevancia-elemzést *több absztrakciós szinten* végre kell hajtani, és csak a dimenzió legrelevánsabb szintjét kell bevonni az elemzésbe.

Korábban azt mondtuk, hogy az attribútum-/dimenziórelevancia kiértékelése azon alapul, hogy az attribútum/dimenzió mennyire képes megkülönböztetni az objektumosztályokat egymástól. Amikor osztályösszehasonlítást (vagy megkülönböztetést) bányászunk, a célosztály és a kontrasztosztályok explicit módon adottak az adatbányászati lekérdezésben. A relevanciaelemzést ezen osztályok összehasonlításával kell végrehajtani, mint azt a következőkben látni fogjuk. Amikor azonban osztályjellemzőt bányászunk, csak egyetlen osztályunk van, amit jellemzünk, azaz nincs kontrasztosztály. Ennek következtében nem nyilvánvaló, hogy milyen kontrasztosztályt lehetne használni a relevanciaelemzés során. Ebben az esetben olyan kontrasztosztályt választunk, amely *összehasonlítható adathalmaz az adatbázisban és amely kizárja a jellemzendő adathalmazt*. Például a másoddiplomások jellemzésekor a kontrasztosztály az *alapképzésesek* halmazából képezhető.

5.3.2. | Az attribútumrelevancia elemzésének módszerei

Az attribútumrelevancia-elemzésre több tanulmány létezik a gépi tanulásban, a statisztikában, a fuzzy és a közelítő halmazok elméletében stb. Az attribútumelemzés alapötlete az, hogy kiszámolunk egy bizonyos mértéket, amit egy adott osztállyal vagy fogalommal kapcsolatos attribútum relevanciájának mennyiségi meghatározására használunk. Ilyen mérték lehet az információnyereség, a Gini-index, a bizonytalansági és a korrelációs együtthatók.

Bemutatunk egy módszert, amely az *információnyereséget* elemző technikát (például amilyen a tanuló döntési fák esetén az ID3 és C4.5 algoritmusokban található)[2] a dimenzió alapú adatelemző eljárással egyesíti. Az eredményül kapott módszer elhagyja a kevésbé informatív attribútumokat és összegyűjti az informatívabbakat, hogy a fogalomleírás elemzésnél felhasználhassa.

Hogyan történhet az információnyereség kiszámítása? Legyen S *gyakorló minták* egy halmaza, ahol minden mintára ismert az osztályozási címke. Minden egyes minta valójában egy sor. Az attribútum a gyakorló minták osztályának meghatározására használatos. Például a *státus* attribútum használatos lehet egyes minták osztálycímkéjének meghatározására, ami lehet „*másoddiplomás*" vagy „*alapképzéses*". Tegyük fel, hogy m osztály van. S tartalmazza a C_i osztály s_i mintáit, ahol $i = 1, \ldots, m$. Egy tetszőleges minta s_i/s valószínűséggel tartozik a C_i osztályba, ahol s az S halmazbeli minták száma. Egy adott minta osztályozásához szükséges **várható információ:**

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}. \tag{5.4}$$

Egy A attribútum, amelynek értékei $\{a_1, a_2, \ldots, a_v\}$ használható az S particionálására $\{S_1, S_2, \ldots, S_v\}$ részhalmazokra, ahol S_j azokat az S-beli mintákat tartalmazza, amelyekre az A értéke a_j. Tartalmazza S_j az s_{ij} mintákat a C_i osztályból, ezen A szerinti particionáláson alapuló várható információ az A **entrópiájaként** ismert. Ez az alábbi súlyozott átlag:

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj}). \tag{5.5}$$

Az A szerinti particionálással elérhető **információnyereséget** a következő képlet adja:

$$Nyereség(A) = I(s_1, s_2, \ldots, s_m) - E(A). \tag{5.6}$$

A relevanciaelemzés ezen megközelítésével kiszámíthatjuk az információnyereséget az S-beli mintákat meghatározó minden egyes attribútumra. A legmagasabb információnyereséggel rendelkező attribútumot tekintjük az adott halmaz leginkább megkülönböztető attribútumának. Így tehát minden attribútumra kiszámítva az információnyereséget, az

2 | A **döntési fa** egy folyamatábraszerű fastruktúra, ahol minden egyes csomópont egy attribútumtesztet jelöl, minden egyes ág egy teszteredményt képvisel, és a fa levelei osztályokat vagy osztályeloszlásokat reprezentálnak. A döntési fák hasznosak az osztályozásnál és könnyen átalakíthatók logikai szabályokká. A döntési fa indukciót a 7. fejezetben tárgyaljuk.

attribútumok rangsorolását kapjuk. Ez a rangsor használható a relevanciaelemzés során fogalomleírásban használt attribútumok kiválasztására.

A fogalomleírásnál használt attribútumrelevancia-elemzés a következőképpen hajtható végre.

(1) **Adatgyűjtés** – Lekérdezésfeldolgozással gyűjtsük össze mind a célosztály, mind a kontrasztosztály adatait. Osztály-összehasonlítás esetén mind a célosztályt, mind a kontrasztosztályt a felhasználó adja meg az adatbányászati lekérdezésben. Osztályjellemzés esetén a célosztály a jellemezni kívánt osztály, a kontrasztosztály pedig azoknak az összehasonlítható adatoknak a halmaza, amelyek nincsenek benne a célosztályban.

(2) **Előzetes relevanciaelemzés konzervatív AOI használatával** – Ez a lépés dimenziók és attribútumok egy halmazát azonosítja, amelyekre a választott relevanciamértéket alkalmazni fogjuk. Mivel a dimenziók különböző szintjei egy adott osztály szempontjából nagyon különböző relevanciájúak lehetnek, általában minden attribútumot, amely egy dimenzió fogalmi szintjeit meghatározza, be kell venni a relevanciaelemzésbe. Az attribútumorientált indukciót (AOI) használhatjuk bizonyos előzetes relevanciaelemzés végrehajtására az adatokon a nagyon nagy számú különböző értékkel rendelkező attribútumok törlése vagy általánosítása során (például *név* és *telefon#*). Az ilyen attribútumok valószínűleg nem hasznosak a fogalomleíráshoz. Ahhoz, hogy konzervatív legyen az itt végrehajtott AOI, olyan attribútum-általánosítási küszöböt kell hogy alkalmazzon, amely ésszerűen nagy ahhoz, hogy megengedjen több (de nem az összes) attribútumot figyelembe venni a választott mérték szerinti további relevanciaelemzésben (lásd 3. lépés). Egy ilyen AOI alkalmazásával kapott relációt az adatbányászati feladat **jelölt relációjának** nevezünk.

(3) **Hagyjuk el a lényegtelen és kevésbé lényeges attribútumokat a választott relevanciaelemzési mérték használatával** – A választott relevanciaelemzési mérték használatával értékeljük ki a jelölt reláció minden egyes attribútumát. Ez a relevanciamérték lehet az adatbányászati rendszerbe beépített vagy a felhasználó által megadott. Például mértékként használható a fent leírt információnyereség. Az attribútumok ezután az adatbányászati feladathoz számított relevanciának megfelelően rendezettek (azaz rangsoroltak). Majd a feladat szempontjából lényegtelen vagy kevésbé lényeges attribútumokat elhagyjuk. A „kevésbé lényeges" meghatározásához beállíthatunk küszöbértéket. Ez a lépés egy **kezdeti célosztály munkarelációt** és egy **kezdeti kontrasztosztály munkarelációt** eredményez.

(4) **A fogalomleírás létrehozása AOI felhasználásával** – Az attribútum-általánosítási küszöbértékek egy kevésbé konzervatív halmazát használva hajtsuk végre az AOI-t. Ha a leíró adatbányászati feladat osztályjellemzés, akkor itt csak a kezdeti célosztály munkarelációt használjuk. Ha a leíró adatbányászati feladat osztály-összehasonlítás, akkor egyaránt használjuk a kezdeti célosztály munkarelációt és a kezdeti kontrasztosztály munkarelációt.

Ennek az eljárásnak a bonyolultsága az 5.1. ábra algoritmusához hasonló, mivel az indukciós eljárást kétszer hajtjuk végre, azaz az előzetes relevanciaelemzésben (2. lépés) és a kezdeti munkareláción (4. lépés). A választott mérték szerinti attribútumrelevanciaelemzésben használt statisztikákat (3. lépés) az adatbázis végigolvasása (2.lépés) közben lehet gyűjteni.

5.3.3. | Analitikus jellemzés – példa

Ha a bányászott fogalmi leírások sok attribútumot tartalmaznak, akkor analitikus jellemzést kell végrehajtani. Az eljárás először az általánosítás végrehajtását megelőzően elhagyja a lényegtelen vagy kevésbé lényeges attribútumokat. Vizsgáljunk meg egy példát az ilyen analitikus bányászati eljárásra.

5.9. példa | Tegyük fel, hogy a *Csúcs_Egyetem* adatbázis *másoddiplomásait* leíró általános jellemzőket szeretnénk kibányászni analitikus jellemzés felhasználásával. Adottak a *név, nem, szak, születési_hely, születési_dátum, telefon#* és *átlag* attribútumok.

Hogyan történik az analitikus jellemzés? Az 1. lépésben a célosztály adatait gyűjtjük össze, ami a *másoddiplomások* halmazából áll. A relevanciaelemzés végrehajtásához a kontrasztosztály adataira is szükségünk van. Ezt az *alapképzésesek* halmaza adja.

A 2. lépésben konzervatív attribútum-általánosítási küszöb szerinti attribútumorientált indukció alkalmazásával történő attribútumtörléssel és attribútumáltalánosítással előzetes relevanciaelemzést hajtunk végre. Az 5.3. példához hasonlóan a *név* és a *telefon#* attribútumokat elhagyjuk, mert különböző értékeik száma meghaladja a megfelelő analitikus attribútum küszöbértékét. Mint az 5.3. példában, fogalmi hierarchiákat használunk a *születési_hely* általánosítására *születési_országgá* és a *születési_dátumnak életkor_tartománnyá*. A *szak* és az *átlag* attribútumokat szintén magasabb absztrakciós szintre általánosítjuk az 5.3. példában leírt fogalmi hierarchiák felhasználásával. Ezért a jelölt relációban a *nem, szak, születési_ország, életkor_tartomány, átlag* attribútumok maradnak. Az eredményül kapott relációt az 5.5. és 5.6. táblázat mutatja.

5.5. táblázat | Az analitikus jellemzésnél kapott jelölt reláció: célosztály (*másoddiplomások*)

nem	szak	születési_ország	életkor_tartomány	átlag	darabszám
F	Természettudomány	Kanada	21...25	nagyon_jó	16
N	Természettudomány	Külföldi	26...30	kitűnő	22
F	Mérnök	Külföldi	26...30	kitűnő	18
N	Természettudomány	Külföldi	26...30	kitűnő	25
F	Természettudomány	Kanada	21...25	kitűnő	21
N	Mérnök	Kanada	21...25	kitűnő	18

5.6. táblázat | Az analitikus jellemzésnél kapott jelölt reláció: kontrasztosztály (*alapképzésesek*)

nem	szak	születési_ország	életkor_tartomány	átlag	darabszám
F	Természettudomány	Külföldi	<=20	nagyon_jó	18
N	Üzlet	Kanada	<=20	közepes	20
F	Üzlet	Kanada	<=20	közepes	22
N	Természettudomány	Kanada	21...25	közepes	24
F	Mérnök	Külföldi	21...25	nagyon_jó	22
N	Mérnök	Kanada	<=20	kitűnő	24

A 3. lépésben a jelölt reláció attribútumait kiértékeljük a választott relevanciaelemzési mérték (például információnyereség) felhasználásával. Legyen C_1 a *másoddiplomások* osztálya és C_2 az *alapképzésesek* osztálya. A *másoddiplomások* osztályában 120, az *alapképzésesek* osztályában 130 minta van. Az információnyereség kiszámításához először minden egyes attribútumra az (5.4) egyenlet alkalmazásával kiszámoljuk az adott minta rangsorolásához szükséges várható információt:

$$I(s_1, s_2) = I(120, 130) = -\frac{120}{250}\log_2\frac{120}{250} - \frac{130}{250}\log_2\frac{130}{250} = 0{,}9988.$$

Ezután ki kell számítanunk minden egyes attribútum entrópiáját. Próbáljuk ki ezt a *szak* attribútumra. Meg kell néznünk a *másoddiplomások* és az *alapképzésesek* eloszlását a *szak* minden egyes értékére. Ezen eloszlások mindegyikére kiszámoljuk a várható információt.

Szak = "*természettudomány*" esetén:
$s_{11} = 84 \quad s_{21} = 42 \quad I(s_{11}, s_{21}) = 0{,}9183$

szak = "*mérnök*" esetén:
$s_{12} = 36 \quad s_{22} = 46 \quad I(s_{12}, s_{22}) = 0{,}9892$

szak = "*üzlet*" esetén:
$s_{13} = 0 \quad s_{23} = 42 \quad I(s_{13}, s_{23}) = 0$

Az (5.5) egyenletet alkalmazásával, amennyiben a *szak* attribútum alapján particionáljuk a mintákat, egy adott minta rangsorolásához szükséges várható információra kapjuk:

$$E(szak) = \frac{126}{250}I(s_{11}, s_{21}) + \frac{82}{250}I(s_{12}, s_{22}) + \frac{42}{250}I(s_{13}, s_{23}) = 0{,}7873.$$

Innen az információnyereség, amit ebből a particionálásból kaphatunk:

$$Nyereség(szak) = I(s_1, s_2) - E(szak) = 0{,}2115.$$

Hasonlóan kiszámítható az információnyereség a maradék attribútumokra is. Az egyes attribútumok információnyeresége növekvő sorrendben: *nem* 0,0003; *születési_ország* 0,0407; *szak* 0,2115; *átlag* 0,4490; és *életkor_tartomány* 0,5971. Tegyük fel, hogy a kevésbé lényeges attribútumok azonosítására a 0,1 attribútumrelevancia küszöbértékét használjuk. A *nem* és a *születési_ország* attribútumok információnyeresége a küszöbérték alatt van, ezért kevésbé lényegesnek számítanak, így elhagyjuk őket. A kontrasztosztályt szintén töröljük, eredményül a kezdeti célosztály munkarelációt kapjuk.

A 4. lépésben az attribútumorientált indukciót alkalmazzuk a kezdeti célosztály munkarelációra az 5.1. ábra algoritmusát követve. ■

5.4. | Osztály-összehasonlítások bányászata: különböző osztályok megkülönböztetése

Sok alkalmazásban a felhasználókat nem egy magában álló osztály (vagy fogalom) leírása vagy jellemzése érdekli, hanem inkább olyan leírások bányászatát részesítik előnyben, amelyek összehasonlítanak vagy megkülönböztetnek egy osztályt (vagy fogalmat) más összehasonlítható osztályokkal (vagy fogalmakkal). Az osztály-megkülönböztetés vagy -összehasonlítás (innentől **osztály-összehasonlításként** hivatkozunk rá) során olyan leírásokat bányászunk, amelyek megkülönböztetik a célosztályt annak kontrasztosztályaitól. Megjegyezzük, hogy a célosztálynak és a kontrasztosztályoknak *összehasonlíthatóknak* kell lenniük abban az értelemben, hogy hasonló dimenziókon és attribútumokon kell osztozniuk. Például a *személy, cím* és *árucikk* osztályok nem összehasonlíthatók. Viszont az utolsó három év értékesítési adatai összehasonlítható osztályok és a számítógéptudományi hallgatók, illetve fizikus hallgatók szintén azok.

Az előző alfejezetekben tárgyalt osztályjellemzésben magában álló osztályra vonatkozó többszintű adatösszegzéssel és -jellemzéssel foglalkoztunk. Az ott kifejlesztett technikák kiterjeszthetők több összehasonlítható osztályon keresztüli osztály-összehasonlítás kezelésére. Például az osztályjellemzésre leírt attribútum-általánosítási eljárást tudjuk úgy módosítani, hogy az általánosítás *egyidejűleg* történjen az összes összehasonlítandó osztályon. Ez lehetővé teszi, hogy az összes osztály attribútumait *ugyanarra* az absztrakciós szintre általánosítsuk. Tegyük fel például, hogy adottak az *ElektroMind* adatbázisban az 1998-as és az 1999-es értékesítési adatok, és szeretnénk összehasonlítani ezt a két osztályt. Tekintsük a *hely* dimenziót a *város*, *megye_vagy_tartomány* és *ország* absztrakciós szintekkel. Minden egyes adatosztályt ugyanarra a *hely* szintre kell általánosítani. Azaz egyidejűleg kell mindegyiket általánosítani vagy a *város,* vagy a *megye_vagy_tartomány* vagy az *ország* szintre. Elméletileg ez hasznosabb, mint összehasonlítani mondjuk az 1998-as vancouveri eladásokat az 1999-es Egyesült Államok-beli eladásokkal (azaz, ahol az egyes eladási adathalmazok különböző szintre általánosítottak). A felhasználóknak azonban meg kell adni a lehetőséget, hogy saját választásukkal felülbírálják az ilyen automatikus, egyidejű összehasonlítást, ha úgy látják jónak.

5.4.1. | Osztály-összehasonlítási módszerek és megvalósításuk

Hogyan hajtódik végre az osztály-összehasonlítás? Általában az eljárás a következő:

1. **Adatgyűjtés** – Lekérdezéssel összegyűjtjük az adatbázis releváns adatainak halmazát és egyenként particionáljuk egy *célosztályba* és egy vagy több *kontrasztosztályba*.
2. **Dimenziórelevancia-elemzés** – Ha sok a dimenzió és *analitikus összehasonlításra* van szükség, akkor ezeken az osztályokon az 5.3. alfejezetben leírtak szerint relevanciaelemzést kell végrehajtani, és a további elemzésben csak az erősen releváns dimenziók vesznek részt.
3. **Egyidejű általánosítás** – A célosztályon végrehajtjuk az általánosítást a felhasználó vagy szakértő által megadott dimenzió küszöbértékkel vezérelt szintre, amelynek ered-

ménye az **elsődleges célosztály reláció/kocka**. A fogalmakat a kontrasztosztály(ok)ban ugyanarra a szintre általánosítjuk, mint az elsődleges célosztály relációban/kockában, így alakítva ki az **elsődleges kontrasztosztály(ok) relációt/kockát**.

4. **A kapott összehasonlítás megjelenítése** – Az eredményül kapott osztály-összehasonlítási leírást táblázatok, grafikonok és szabályok formájában lehet megjeleníteni. Ez a megjelenítés általában tartalmaz egy „szembeállító" mértéket (például a `count%`), amely a cél- és kontrasztosztályok közötti összehasonlítást fejez ki. A felhasználó változtatni tudja az összehasonlítási leírást lefúrás, felgörgetés és más OLAP-műveleteket alkalmazva a cél- és a kontrasztosztályokon, ha szükséges.

A fenti tárgyalás egy általános algoritmust vázolt fel az adatbázisokon végzett analitikus összehasonlítások bányászatára. Összevetve az analitikus jellemzéssel, a fenti algoritmus a célosztály és a kontrasztosztályok egyidejű általánosítását is magában foglalja, így ezek az osztályok ugyanazon absztrakciós szinten egyidejűleg kerülnek összehasonlításra.

Megvalósítható-e hatékonyan az osztály-összehasonlítás bányászata adatkockák alkalmazásával? Igen – az eljárás hasonló az 5.2.2. alfejezetben tárgyalt adatjellemzések bányászatához. Egy jelző szolgál annak jelölésére, hogy egy sor beletartozik-e vagy sem egy cél- vagy kontrasztosztályba, ahol ez a jelző kiegészítő dimenzióként jelenik meg az adatkockában. Mivel a cél- és kontrasztosztályok összes többi dimenziója a kocka ugyanazon részén osztozik, az egyidejű általánosítás és részletezés automatikusan megvalósul a kockában történő felgörgetéssel és lefúrással.

A következő példa egy – a *Csúcs_Egyetem* adatbázisban lévő *másoddiplomásokat* és *alapképzéseseket* leíró – osztály-összehasonlítás bányászata.

5.10. példa | **Osztály-összehasonlítás bányászata.** Tegyük fel, hogy a *Csúcs_Egyetem* adatbázisban lévő *másoddiplomás* hallgatók és *alapképzéses* hallgatók általános tulajdonságait szeretnénk összehasonlítani; adottak a *név, nem, szak, születési_hely, születési_dátum, lakóhely, telefon#* és *átlag* attribútumok.

Ez az adatbányászati feladat DMQL-ben a következőképpen fejezhető ki:

use Csúcs_Egyetem_AB
mine comparison as "Másoddiplomások_kontra_alapképzésesek"
in relevance to név, nem, szak, születési_hely, születési_dátum, lakóhely, telefon#, átlag
for "másoddiplomás hallgatók"
where státus **in** "másoddiplomás"
versus "alapképzéses"
where státus **in** "alapképzésesek"
analyze count%
from hallgató

Nézzük meg, hogyan dolgozható fel ez az összehasonlítási leírások bányászatára vonatkozó tipikus adatbányászati lekérdezés.

Először átalakítjuk ezt a lekérdezést két relációs lekérdezéssé, amelyek a feladat szempontjából lényeges két adathalmazt gyűjtik össze: az egyik a *kezdeti célosztály munkareláció*,

a másik a *kezdeti kontrasztosztály munkareláció*, amint az 5.7. és 5.8. táblázatban látható. Ezt tekinthetjük egy adatkocka létrehozásának is, ahol a {*másoddiplomás, alapképzéses*} státus az egyik dimenzió, és a többi attribútum alkotja a maradék dimenziókat.

Másodszor dimenziórelevancia-elemzést hajtunk végre a két adatosztályban. Az elemzés után a lényegtelen vagy kevésbé lényeges dimenziókat, amilyenek a *név, nem, születési_hely, lakóhely és telefon#* elhagyjuk az eredményül kapott osztályokból. Csak az erősen releváns attribútumok vesznek részt a további elemzésben.

5.7. táblázat | Kezdeti célosztály munkareláció (másoddiplomás hallgatók)

név	nem	szak	születési_hely	születési_dátum	lakóhely	telefon#	átlag
Jim Woodman	F	Informatika	Vancouver, BC, Kanada	8-12-76	3511 Main St. Richmond	687-4950	3,67
Scott Lachance	F	Informatika	Montreal, Que, Kanada	28-7-75	345 1st Ave. Richmond	253-9106	3,70
Laura Lee	N	Fizikus	Seattle, WA, USA	25-8-70	125 Austin Ave. Burnaby	420-5232	3,83
...	...	...	...	...	...	...	...

5.8. táblázat | Kezdeti kontrasztosztály munkareláció (alapképzéses hallgatók)

név	nem	szak	születési_hely	születési_dátum	lakóhely	telefon#	átlag
Bob Schumann	F	Vegyész	Calgary, Alt, Kanada	10-1-78	2642 Halifax St. Burnaby	294-4291	2,96
Amy Eau	N	Biológus	Golden, BC, Kanada	30-3-76	463 Sunset Cres. Vancouver	681-5417	3,52
...	...	...	...	...	...	...	...

5.9. táblázat | A célosztályhoz tartozó elsődleges általánosított reláció (másoddiplomás hallgatók)

szak	életkor_tartomány	átlag	count%
Természettudomány	21...25	jó	5,53%
Természettudomány	26...30	jó	5,02%
Természettudomány	30_év_feletti	nagyon_jó	5,86%
...	...	...	...
Üzlet	30_év_feletti	kitűnő	4,68%

Harmadszor egyidejű általánosítást hajtunk végre. Az általánosítás végrehajtódik a célosztályon a felhasználó vagy szakértő által megadott dimenzió küszöbértékkel vezérelt szintre, amelynek eredménye az *elsődleges célosztály reláció/kocka*. A kontrasztosztályt ugyanarra a szintre általánosítjuk, mint az elsődleges célosztály relációban/kockában, kialakítva az *elsődleges kontrasztosztály(ok) relációt/kockát*, amint az 5.9. és 5.10. táblázatban szerepel. Össze-

5.10. táblázat | A kontrasztosztályhoz tartozó elsődleges általánosított reláció (alapképzéses hallgatók)

szak	életkor_tartomány	átlag	count%
Természettudomány	16...20	közepes	5,53%
Természettudomány	16...20	jó	4,53%
...	...	...	...
Természettudomány	26...30	jó	2,32%
Üzlet	30_év_feletti	kitűnő	0,68%

vetve az *alapképzéses* hallgatókkal, a *másoddiplomás* hallgatók jellemzően idősebbek és általában magasabb az átlagérdemjegyük.

Végül az eredményül kapott osztály-összehasonlítást táblázatok, grafikonok és/vagy szabályok formájában mutatjuk be. A megjelenítés magában foglalja az összehasonlító mértéket (például `count%`), amely összehasonlítást tesz a célosztály és a kontrasztosztály között. Például a *másoddiplomások* 5,02%-a természettudományi szakos, 26–30 év közötti korú és „jó" átlagérdemjegyű, míg az *alapképzésesek* 2,32%-a rendelkezik hasonló jellemzőkkel. Ha a felhasználó szükségesnek ítéli, végrehajthat lefúrást és más OLAP-műveleteket a cél- és a kontrasztosztályokon a végső leírás absztrakciós szintjeinek beállítására. ■

5.4.2. | Az osztály-összehasonlítási leírások megjelenítése

Hogyan jeleníthetők meg az osztály-összehasonlítási leírások? Amint az osztályjellemzések, az osztály-összehasonlítások is különféle formában jeleníthetők meg, beleértve az általánosított relációkat, kereszttáblázatokat, oszlop- és kördiagramokat, grafikonokat és szabályokat. A logikai szabályok kivételével, a fenti formákat azonos módon használjuk jellemzésre és összehasonlításra. Ebben az alfejezetben az osztály-összehasonlítások megkülönböztető szabályok formájában való megjelenítését tárgyaljuk.

A jellemzés leírásához hasonlóan, az összehasonlító leírás cél- és kontrasztosztályainak mennyiségi megkülönböztető jegyei *mennyiségi megkülönböztető szabállyal* írhatók le, amelyek egy *d-súly* statisztikai érdekességi mértéket rendelnek a leírás minden egyes általánosított sorához.

Legyen q_a egy általánosított sor és C_j a célosztály, ahol q_a lefedi a célosztály bizonyos sorait. Megjegyezzük, hogy esetleg q_a a kontrasztosztályok bizonyos sorait szintén lefedheti, különösen mivel összehasonlító leírással foglalkozunk. A q_a sor **d-súlya** legyen a q_a által lefedett kezdeti célosztály munkarelációbeli sorok számának és a q_a által lefedett mind a kezdeti célosztály, mind a kezdeti kontrasztosztály munkarelációkhoz tartozó sorok teljes számának az aránya. Formálisan a q_a általánosított sor C_j osztályra vonatkozó **d-súlya**:

$$d\_súly = count(q_a \in C_j) \,/\, \sum_{i=1}^{m} count(q_a \in C_i), \tag{5.7}$$

ahol m a cél- és a kontrasztosztályok száma összesen, C_j a $\{C_1, ..., C_m\}$ közül valamelyik és $count(q_a \in C_i)$ a C_i osztály q_a által lefedett sorainak a száma. A *d_súly* értéke a [0, 1] (vagy a [0%, 100%]) intervallumba esik.

Magas d-súly a célosztályban azt mutatja, hogy az általánosított sor által képviselt fogalom elsősorban a célosztályból ered, míg az alacsony d-súly arra utal, hogy a fogalom elsősorban a kontrasztosztályokból származik. A d-súly alapján, vagy más mértéket használva felállítható egy küszöb a lényeges sorok megjelenítésének vezérlésére, ahogy az 5.2.3. alfejezetben leírtuk.

5.11. táblázat | A másoddiplomás és az alapképzéses hallgatók számának megoszlása egy általánosított sor esetén

státus	szak	életkor_tartomány	átlag	darabszám
másoddiplomás	Természettudomány	21...25	jó	90
alapképzéses	Természettudomány	21...25	jó	210

5.11. példa | Az 5.10. példában tegyük fel, hogy az 5.9. és 5.10. táblázatokból származó *szak* = "*természettudomány*" **and** *életkor_tartomány* = "21...25" **and** *átlag* = "*jó*" általánosított sor darabszám értékének eloszlását az 5.11. táblázat mutatja.

Az adott általánosított sor d-súlya a célosztályra vonatkozóan 90/(90 + 210) = 30%, a kontrasztosztályra nézve pedig 210/(90 + 210) = 70%. Azaz, *ha egy hallgató természettudományi szakos, 21 és 25 év közötti és "jól" átlaga van, akkor az adatok alapján 30%-os valószínűséggel másoddiplomás és 70%-os valószínűséggel alapképzéses hallgató.* Az 5.9. és 5.10. táblázat többi általánosított sorának *d-súlya* hasonlóan kapható meg. ■

Egy adott összehasonlító leírás célosztályára a **mennyiségi megkülönböztető szabály** az alábbi formában írható le:

$$\forall X,\ cél\_osztály(X) \Leftarrow feltétel(X) \qquad [d : d\_súly], \tag{5.8}$$

ahol a feltétel a leírás egy általánosított során alapul. Ez különbözik az osztályjellemzésnél kapott szabályoktól, ahol az „ebből következik" nyíl balról jobbra mutatott.

5.12. példa | Az 5.11. példában szereplő általánosított sor és darabszám-megoszlás alapján a *másoddiplomás_hallgató* célosztályra a mennyiségi megkülönböztető szabály a következőképpen írható fel:

$$\forall X,\ státus(X) = \text{"}másoddiplomás\text{"} \Leftarrow szak(X) = \text{"}természettudomány\text{"} \wedge életkor\_tartomány = \text{"21...25"} \wedge átlag(X) = \text{"}jó\text{"}\ [d : 30\%]. \tag{5.9}$$

■

Vegyük észre, hogy a megkülönböztető szabály egy elégséges, de nem szükséges feltétel arra, hogy egy objektum (vagy sor) a célosztályba tartozzon. Például az (5.9) szabály azt mondja, hogy ha X kielégíti a feltételt, akkor annak valószínűsége, hogy X másoddiplomás hallgató, 30%. Azonban ez nem jelenti annak a valószínűségét, hogy X megfelel a feltételnek, ha tudjuk, hogy X másoddiplomás hallgató. Ez azért van, mert bár a feltételnek megfelelő sorok a célosztályban vannak, más sorok, amelyek nem feltétlenül elégítik

ki ezt a feltételt, szintén a célosztályban lehetnek, mivel a szabály esetleg nem tartalmazza az *összes* célosztálybeli példát az adatbázisból. Tehát a feltétel elégséges, de nem szükséges.

5.4.3. | Osztályleírás: jellemzés és összehasonlítás együttes megjelenítése

Mivel az osztályjellemzés és az osztály-összehasonlítás az osztályleírás kétféle nézőpontját alkotja, meg tudjuk-e jeleníteni mindkettőt ugyanabban a táblában vagy ugyanabban a szabályban? Valójában mindaddig, amíg tisztán értjük a t-súly és a d-súly mértékek jelentését, és helyesen tudjuk tolmácsolni azokat, nem jelent különösebb nehézséget mindkét nézőpontot ugyanabban a táblázatban megjeleníteni. Vizsgáljunk meg egy példát, ahol osztályjellemzést és osztály-összehasonlítást ugyanabban a kereszttáblázatban jelenítünk meg.

5.13. példa | Legyen az 5.12. táblázat az *ElektroMind* adatbázisból az 1999-es TV és számítógép-értékesítés teljes darabszámát (1000 db-ban) mutató kereszttáblázat.

Legyen a célosztály *Európa* és a kontrasztosztály *Észak-Amerika*. A két osztály közötti eladások megoszlásának t-súly és d-súly értékeit az 5.13. táblázat mutatja. A táblázat szerint egy általánosított sor vagy objektum t-súlya (például *árucikk* = "*TV*") egy adott osztályra (például *Európa* célosztályra) azt mutatja, hogy mennyire tipikus egy sor az adott osztályra vonatkozóan (a példában mennyi a TV-eladások aránya az Európai eladásokon belül). Egy sor d-súlya azt mutatja, hogy mennyire megkülönböztető a sor az adott (cél- vagy kontraszt-) osztályban a riválisosztályával összehasonlítva (a példában, hogyan néznek ki az európai TV-eladások az észak-amerikai eladásokkal összehasonlítva).

Például, az "(*Európa*, *TV*)" sor t-súlya 25%, mivel az Európában eladott TV-k száma (80 ezer) csak 25%-a az európai – mindkét árucikket tartalmazó – eladások számának (320 ezer). Az "(*Európa*, *TV*)" sor d-súlya 40%, mivel az Európában eladott TV-k száma (80 ezer), a cél- és a kontrasztosztályban együtt, azaz Európában és Észak-Amerikában értékesített TV-k számának (amely 200 ezer) a 40%-a. ■

5.12. táblázat | Kereszttáblázat az 1999-es TV- és számítógép-értékesítés teljes darabszámáról (`count`) (1000 db-ban)

hely/árucikk	TV	számítógép	mindkét_árucikk
Európa	80	240	320
Észak-Amerika	120	560	680
mindkét_terület	200	800	1000

Vegyük észre, hogy az 5.13. táblázatban lévő kereszttáblázatban a *darabszám* mérték követi a kereszttáblázatok általános tulajdonságát (azaz a *darabszámok* soronkénti és oszloponkénti összege megegyezik a *mindkét_árucikk* és a *mindkét_terület* darabszám által meghatározott celláiban lévő megfelelő összeggel. Azonban ez a tulajdonság nem teljesül a t-súly és a d-súly mértékekre. Ez azért van így, mert ezen mértékek mindegyikének a szemantikus jelentése különbözik a darabszámétól, ahogyan ezt az 5.13. példában kifejtettük.

5.13. táblázat | A két osztály közötti eladások megoszlásának t-súly és d-súly értékei

hely/árucikk	*TV*			*számítógép*			*mindkét_árucikk*		
	darab-szám	t_súly	d_súly	darab-szám	t_súly	d_súly	darab-szám	t_súly	d_súly
Európa	80	25%	40%	240	75%	30%	320	100%	32%
Észak-Amerika	120	17,65%	60%	560	82,35%	70%	680	100%	68%
mindkét_terület	200	20%	100%	800	80%	100%	1000	100%	100%

Kifejezhető-e egy szabályban a mennyiségi jellemző szabály és a mennyiségi megkülönböztető szabály együtt? A válasz igen – mennyiségi jellemző szabály és a mennyiségi megkülönböztető szabály ugyanazon osztály esetén egyesíthető egy mennyiségi leíró szabállyá az adott osztályra, amely a megfelelő jellemző és megkülönböztető szabályokhoz tartozó t-súlyokat és d-súlyokat jeleníti meg. Szemléltetésül gyorsan tekintsük át, hogyan fejezhetők ki a mennyiségi jellemző és megkülönböztető szabályok.

Amint az 5.2.3. alfejezetben tárgyaltuk, a mennyiségi jellemző szabály az adott célosztály szükséges feltételét adja, mivel valószínűségi mértéket nyújt a célosztály minden egyes lehetséges tulajdonságára. Az ilyen szabály a következő alakú:

$$\forall X,\ cél\_osztály(X) \Rightarrow feltétel_1\ (X)[t : w_1]\ \vee \ldots \vee feltétel_m\ (X)[t : w_m], \tag{5.10}$$

ahol minden egyes feltétel a célosztály egy tulajdonságát képviseli. A szabály azt mutatja, hogy ha X a *cél_osztályból* való, akkor a t-súly w_i értéke adja annak valószínűségét, hogy X kielégíti a $feltétel_i$-t, ahol i az $\{1, \ldots, m\}$ halmazból való.

Ahogyan azt az 5.4.1. alfejezetben korábban tárgyaltuk, egy mennyiségi megkülönböztető szabály elégséges feltételt ad a célosztályra, mivel mennyiségi mértéket nyújt azokra a tulajdonságokra, amelyek előfordulnak a célosztályban, szemben a kontraszt-osztályokban előfordulókkal. Az ilyen szabály a következő alakú:

$$\forall X,\ cél\_osztály(X) \Leftarrow feltétel_1\ (X)[d : w_1] \vee \ldots \vee feltétel_m\ (X)[d : w_m].$$

A szabály azt mutatja, ha X kielégíti a $feltétel_i$-t, akkor w_i (a d-súly érték) a valószínűsége annak, hogy X a *cél_osztályban* van, ahol i az $\{1, \ldots, m\}$ halmazból való.

Egy mennyiségi jellemző szabály és egy mennyiségi megkülönböztető szabály egy adott osztályra egy **mennyiségi leíró szabállyá** egyesíthető a következőképpen: (1) minden egyes feltétel esetén mutassuk a hozzárendelt t-súly és d-súly értékeket, és (2) kétirányú nyilat használjunk az adott osztály és a feltételek között. Tehát a mennyiségi leíró szabály a következő alakú:

$$\forall X,\ cél\_osztály(X) \Leftrightarrow feltétel_1\ (X)[t : w_1, d : w_1'] \vee \ldots \vee feltétel_m\ (X)[t : w_m, d : w_m'] \tag{5.11}$$

Ez a képlet azt mutatja, hogy minden i-re az $\{1, \ldots, m\}$ halmazból teljesül, hogy ha X a *cél_osztályban* van, akkor létezik olyan w_i valószínűség, amellyel X kielégíti $feltétel_i$-t; és ha X kielégíti a $feltétel_i$-t, akkor létezik olyan w_i' valószínűség, amellyel X a *cél_osztályban* van.

5.14. példa | Egyszerű az 5.13. példa 5.13. táblázatában lévő kereszttáblázatot átalakítani osztályleírássá mennyiségi leíró szabályok formájában. Például az *Európa* célosztályra a mennyiségi leíró szabály a következő:

$$\forall X, hely(X) = \text{“}Európa\text{”} \Leftrightarrow (árucikk\ (X) = \text{“}TV\text{”})[t : 25\%, d : 40\%] \vee (árucikk(X) = \text{“}számítógép\text{”})\ [t : 75\%, d : 30\%] \quad (5.12)$$

A szabály az *ElektroMind* adatbázis 1999. évi TV- és számítógép-értékesítésről azt fejezi ki, hogy ha ezen árucikkek közül valamelyikből előfordult eladás Európában, akkor 25% annak a valószínűsége, hogy ez az árucikk TV, míg 75% annak a valószínűsége, hogy számítógép. Másrészt, ha összehasonlítjuk ezen árucikkek európai és észak-amerikai eladásait, akkor a TV-k 40%-át adták el Európában (és így kikövetkeztethetjük, hogy a TV-k 60 %-át adták el Észak-Amerikában). Továbbá, a számítógép-eladásokat tekintve, ezek 30%-a történt Európában. ■

5.5. | Leíró statisztikai mértékek bányászata nagy adatbázisokban

Korábban ebben a fejezetben megtárgyaltuk az osztályleírást népszerű mértékekkel kifejezve például `count`, `sum` és `average`. A relációs adatbázisrendszerek öt beépített összesítőfüggvényt nyújtanak: `count()`, `sum()`, `avg()`, `max()` és `min()`. Ezek a függvények hatékonyan kiszámíthatók (inkrementális és osztott módon) adatkockákon is. Ezért nem okoz gondot ezeket az összesítőfüggvényeket mint alapmértékeket belevenni a többdimenziós adatok leíró bányászatába.

Sok adatbányászati feladat esetén azonban a felhasználók többet szeretnének megtudni az adatjellemzőkről, mind az elhelyezkedési mértékek, mind az adatszóródás tekintetében. Az elhelyezkedési mértékek többek között a *középérték*, a *medián*, a *módusz* és a *terjedelemközép*; az adatszóródás mértékei többek között: a *kvartilisek*, a *percentilisek*, a *szélsőséges értékek* és a *szórásnégyzet*. Ezek a leíró statisztikák nagy segítséget nyújtanak az adateloszlások megértésében. Az ilyen mértékeket kiterjedten tanulmányozták a statisztikai irodalomban. Adatbányászati szempontból azt kell megvizsgálnunk, hogyan lehet ezeket hatékonyan kiszámítani nagy, többdimenziós adatbázisokban.

5.5.1. | Az elhelyezkedés mérése

Egy adathalmaz „közepének” legáltalánosabb és leghatékonyabb numerikus mértéke a *(számtani) középérték*. Legyen $x_1, x_2, \ldots, x_n$ egy n értékből vagy mérésből álló halmaz. Ezen értékek halmazának **középértéke**:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \quad (5.13)$$

Ez megfelel a relációs adatbázisrendszerek által nyújtott beépített *átlag* összesítőfüggvénynek (`avg()` az SQL-ben). A legtöbb adatkockában a *sum* és *count* az előszámítások során összegyűjtésre kerül, ezért az *átlag* származtatása egyszerű, az *átlag* = *sum/count* formulát használva.

Időnként, a halmaz minden egyes x_i értékéhez hozzárendelhetünk egy w_i súlyt, $i = 1, \ldots, n$-re. A súlyok a lényegességre, a fontosságra utalnak, vagy a megfelelő értékeikhez előfordulási gyakoriság van kapcsolva. Ilyen esetben kiszámíthatjuk a **súlyozott számtani középértéket** vagy **súlyozott átlagot**:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}. \tag{5.14}$$

A 2. fejezetben egy mértéket *algebrainak* neveztünk, ha disztributív összesítő mértékekkel kiszámítható. Mivel az `avg()` kiszámítható `sum()/count()` alakban, ahol a `sum()` és a `count()` egyaránt disztributív összesítőmértékek, abban az értelemben, hogy disztributív módon számíthatók ki, így `avg()` algebrai mérték. Igazolható, hogy a súlyozott átlag szintén algebrai mérték.

Bár a középérték az önmagában leghasznosabb mennyiség, amit adathalmazok leírására használunk, nem ez az egyetlen vagy mindig a legjobb módszer az adathalmaz közepének mérésére. Ferde adatoknál az adatközép mérésére jobban használható a *medián, M*. Tegyük fel, hogy az adott adathalmazt alkotó értékek numerikusan rendezettek. Ha az értékek száma, *n* páratlan; akkor a **medián** a rendezett halmaz *középső értéke*; egyébként (azaz ha *n* páros) a *két középső érték átlaga*.

A mértékek 2. fejezetbeli kategorizálását használva a *medián* sem nem disztributív mérték, sem nem algebrai mérték, hanem teljes körű mérték abban az értelemben, hogy nem lehet úgy kiszámítani, hogy felosztjuk az adathalmazt tetszőlegesen kicsi részhalmazokra, és ezek mediánját egymástól függetlenül kiszámítjuk, majd az egyes részhalmazok mediánjainak értékét egyesítjük. Ezzel ellenétben a `count()`,`sum()`, `max()` és `min()` kiszámítható ezen a módon (lévén disztributív mértékek), ezért könnyebb kiszámítani őket, mint a mediánt.

Habár egy nagy adatbázisban nem könnyű a medián pontos értékét meghatározni, közelítő medián hatékonyan számítható. Például csoportosított adatokra a medián a következő interpolációval adható meg:

$$\text{medián} = L_1 + \left(\frac{n/2 - \left(\sum f\right)_l}{f_{median}} \right) c, \tag{5.15}$$

ahol L_1 annak az osztálynak az alsó határa (azaz a legkisebb értéke), amely a mediánt tartalmazza, *n* az adatértékek száma, $\left(\sum f\right)_l$ a medián osztályánál kisebb osztályok gyakoriságainak összege, f_{median} a medián osztályának gyakorisága és *c* a medián osztályának intervallummérete.

A másik elhelyezkedési mérték a *módusz*. Egy adathalmaz **módusza** az az érték, amely a leggyakrabban fordul elő a halmazban. Lehetséges, hogy a legnagyobb gyakoriság több különböző értéknek is megfelel, ez egynél több móduszt eredményez. Azokat az adathalmazokat, amelyeknek egy, kettő vagy három módusza van, rendre **egymóduszú**, **kétmóduszú** és **hárommóduszú** adathalmaznak nevezzük. Általában a két vagy több módusszal rendelkező adathalmazt **többmóduszúnak** hívjuk. A másik szélsőséges esetben, amikor minden egyes adatérték csak egyszer fordul elő, akkor nincs módusz.

Az egymóduszú gyakorisági görbékre, amelyek mérsékelten ferdék (aszimmetrikusak) a következő tapasztalati összefüggésünk van:

$$\textit{Középérték} - \textit{módusz} = 3 \times (\textit{középérték} - \textit{medián}). \tag{5.16}$$

Ez azt jelenti, hogy a mérsékelten ferde, egymóduszú gyakorisági görbék esetén a móduszt könnyen ki tudjuk számítani, ha a középérték és a medián már ismert.

Az **osztályközép**, azaz *az adathalmaz legnagyobb és legkisebb értékének az átlaga*, használható az adathalmaz elhelyezkedési mértékeként. A *terjedelemközép* kiszámítása egyszerűen a `max()` és `min()` SQL-összesítőfüggvények felhasználásával történik.

5.5.2. | Az adatszóródás mérése

A numerikus adatok szétterülési hajlamának fokát az adat **szóródásának** vagy **szórásnégyzetének** nevezzük. Az adatszóródás legáltalánosabb mértékei a (*kvartiliseken* alapuló) *5-számos összegzés*, a *kvartilisek közötti távolság* és a *szórás*. A *dobozdiagramok* megjelenítése (amelyek a szélsőséges értékeket mutatják) szintén hasznos grafikus módszer.

Kvartilisek, percentilisek, szélsőséges értékek, dobozdiagramok

Egy numerikusan rendezett adathalmaz **k-adik percentilise** az az x érték, amelyre teljesül, hogy az adatbejegyzések ***k* százaléka** kisebb vagy egyenlő, mint x. Az M *medián* (az előző alfejezetben tárgyaltuk) megfelel az *50. percentilisnek*.

A medián mellett a leggyakrabban használt percentilisek a **kvartilisek**. Az **első kvartilis**, amit Q_1-el jelölünk, a *25. percentilis;* a **harmadik kvartilis**, amit Q_3-al jelölünk, a *75. percentilis*. A kvartilisek, beleértve a mediánt (ami a Q_2, azaz a második kvartilis, illetve 50. percentilis) is, az eloszlás középpontjáról, terjedelméről és alakjáról adnak bizonyos jelzéseket. Az első és a harmadik kvartilis közötti távolság a szóródás egy egyszerű mértéke, amely az adatok középső fele által lefedett intervallum hosszát adja meg. Ezt a távolságot **kvartilisek közötti távolságnak** (*IQR*, InterQuartile Range) nevezzük és a definíciója:

$$IQR = Q_3 - Q_1. \tag{5.17}$$

Figyelemmel kell lennünk arra, hogy a ferde eloszlások leírására nincs egyszerű, jól használható numerikus szóródási mérték, mint amilyen az *IQR*. A ferde eloszlás két

oldalának a szóródása nem egyenlő. Ezért sokkal informatívabb, ha megadjuk a két kvartilist, Q_1-et és Q_3-at, az M mediánnal együtt. Szokásos szabály a gyanúsan **szélsőséges értékek** azonosítására, hogy a legalább (1,5 × *IQR*)-nyivel a harmadik kvartilis fölé vagy az első kvartilis alá eső értékeket választjuk ki.

Mivel Q_1, M és Q_3 nem tartalmaznak információt az adatok végpontjairól (azaz a széleiről), egy eloszlás alakjának teljesebb összefoglalása kapható, ha a legnagyobb és a legkisebb értékeket szintén megadjuk. Ez az *5-számos összegzésként* ismert. Egy eloszlás **5-számos összegzése** tartalmazza az M mediánt, a Q_1 és Q_3 kvartiliseket és a legkisebb, valamint a legnagyobb egyedi méréseket, a következő sorrendben írva: *Minimum*, Q_1, M, Q_3, *Maximum*.

Az eloszlás népszerű vizuális megjelenítése a **dobozdiagram**. Egy dobozdiagram esetén:

- Általában a végei a kvartiliseknél vannak, tehát a dobozdiagram magassága a kvartilisek közötti távolság, az *IQR*.
- A mediánt egy egyenes jelöli a dobozdiagramon belül.
- Két vonal (amiket *pajesznek* nevezünk) a dobozdiagramon kívül a legkisebb *(Minimum)* és legnagyobb *(Maximum)* értékekig nyúlik ki.

Amennyiben korlátozott számú méréssel dolgozunk, érdemes lehet a potenciális szélsőséges értékeket egyesével megjeleníteni. A dobozdiagram esetén ez azt jelenti, hogy a „pajeszokat" *csak akkor* terjesztjük ki a különösen alacsony vagy magas mérésekig, ha ezek az értékek legfeljebb (1,5 × *IQR*)-nyivel vannak túl a kvartiliseken. Egyébként a pajeszok véget érnek a kvartilisektől (1,5 × *IQR*) értéken belül lévő legszélső mérésnél, a maradék eseteket pedig egyesével ábrázoljuk. A dobozdiagram használható több kompatibilis adathalmaz összehasonlítására. Az 5.5. ábra egy adott időszakra vonatkozóan az *ElektroMind* adatbázis négy áruházában eladott árucikkek egységáradatainak dobozdiagramját mutatja. Az 1. áruház esetén, mint látható, az eladott árucikkekre az M medián 80 $, Q_1 60 $, Q_3 pedig 100 $.

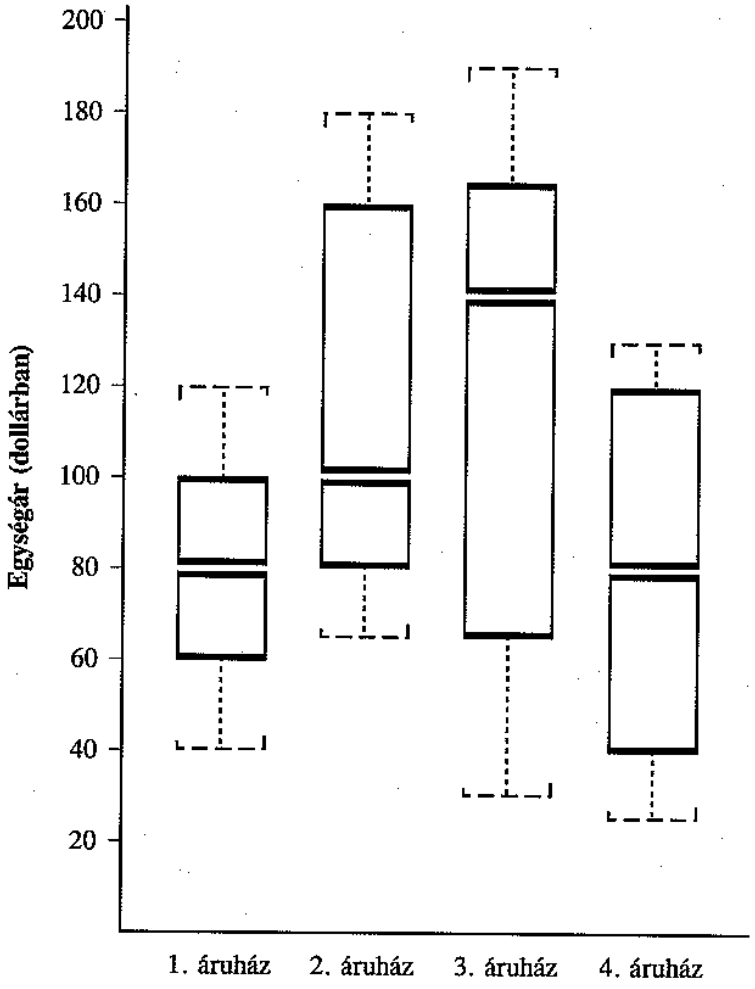
Hasonló okfejtéssel, mint az 5.5.1. alfejezetben a medián elemzésekor, arra a következtetésre jutunk, hogy Q_1 és Q_3 is teljes körű mértékek, akárcsak az *IQR*. A *dobozdiagramok* vagy akár a *megközelítőleg dobozdiagramok* hatékony kiszámítása igen fontos a nagy adathalmazok bányászata szempontjából.

Szórásnégyzet és szórás

Az $x_1, x_2, \ldots, x_n$ n darab mérés **szórásnégyzete** a következő:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - x)^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n}x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n}x_i\right)^2\right]. \tag{5.18}$$

Az **s szórás** a négyzetgyöke az s^2 szórásnégyzetnek. A s szórás mint szóródási mérték alapvető tulajdonságai:



5.5. ábra | Adott időszakra vonatkozóan az *ElektroMind* adatbázis négy áruházában eladott árucikkek egységáradatainak dobozdiagramja

- s a középérték körüli szóródást méri és csak akkor használható, ha a középértéket választottuk a középpont mértékének.
- $s = 0$ csak akkor, ha nincs szóródás, azaz ha minden mérésnek azonos az értéke, egyébként $s > 0$.

Vegyük észre, hogy a szórásnégyzet és a szórás algebrai mértékek, mivel az n (ami `count()` az SQL-ben), a $\sum x_i$ (ami `sum()` az x_i-re) és $\sum x_i^2$ (ami `sum()` az x_i^2-re) kiszámítható tetszőleges részletekben, és egyesítés után behelyettesíthető az (5.18) algebrai egyenletbe. Ezért a két mérték kiszámítása skálázható nagy adatbázisokban.

5.5.3. | Az alapvető statisztikai osztályleírások grafikus megjelenítése

Az oszlop-, kördiagramok és grafikonok mellett, amiket korábban ebben a fejezetben tárgyaltunk, van még néhány további gyakran használt grafikon az adatösszegzések és adateloszlások ábrázolására. Ezek többek között a *hisztogramok*, *kvantilis ábrák*, *q-q ábrák*, *szórásfüggvények* és *loess görbék*.

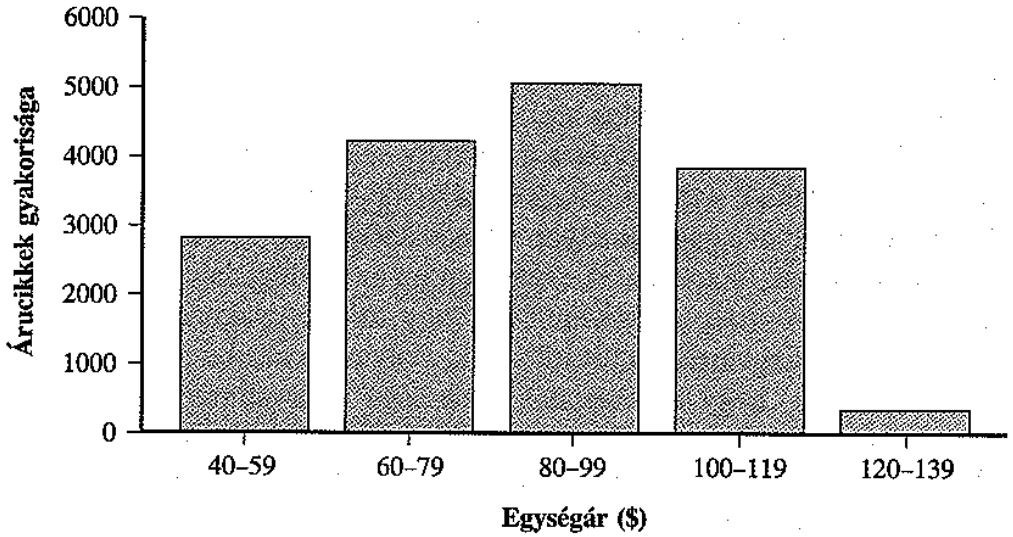
A **hisztogramok** vagy **gyakorisági hisztogramok** ábrázolása egyváltozós grafikus módszer. Egy hisztogram téglalapokból áll, amelyek az adott adatokon belüli osztályok számára vagy gyakoriságára utalnak. Minden téglalap alapja a vízszintes tengelyen van, a közepén „osztály" jelöléssel, és az alap hossza az osztály szélességével egyenlő. Általában az osztály szélessége állandó: egyenlő olyan osztályokkal, amelyeket egy kategorizáló attribútum értékei definiálnak, vagy amelyek azonos szélességű diszkrét intervallumai egy folytonos attribútumnak. Ezekben az esetekben az egyes téglalapok magassága egyenlő az általa reprezentált osztály darabszámával vagy relatív gyakoriságával, és a hisztogramra általában **oszlopdiagramként** hivatkozunk. Folytonos attribútumhoz az osztályok úgy is megválaszthatók, hogy nem azonos szélességű intervallumok határozzák meg őket. Ebben az esetben egy adott osztályra az osztályszélesség egyenlő az intervallum szélességével és a téglalap magassága az osztály sűrűségével (azaz az osztály darabszáma vagy relatív gyakorisága osztva az osztály szélességével). A 3. fejezetben tárgyaltuk a szabályok particionálását hisztogramok létrehozása céljából.

Az 5.14. táblázat adathalmazához tartozó hisztogramot az 5.6. ábra mutatja, ahol az osztályokat azonos szélességű, 20 $-os lépésközű intervallumok határozzák meg, és a gyakoriság az eladott árucikkek száma. A hisztogram legalább százéves és széles körben használt egyváltozós grafikus módszer. Azonban nem feltétlenül olyan hatékony, mint a kvantilis ábra, a *q-q* ábra és a dobozdiagram az egyváltozós mérések csoportjainak összehasonlítására.

5.14. táblázat | Az *ElektroMind* adatbázis áruházaiban eladott árucikkek adatainak egységárhalmaza

Egységár ($)	Eladott árucikkek száma
40	275
43	300
47	250
...	...
74	360
75	515
78	540
...	...
115	320
117	270
120	350

A **kvantilis ábra** egyszerű és hatékony módszer az egyváltozós adateloszlás elsőszintű áttekintésére. Először is megjeleníti az összes adatot (lehetővé téve a felhasználó számára, hogy felbecsülje mind az általános viselkedést, mind a szokatlan előfordulásokat). Másodszor grafikusan ábrázolja a kvantilis információt. Az ebben a lépésben használt eljárás némileg különbözik a percentilisek kiszámításától. Jelölje $x_{(i)}$, ahol $i = 1, ..., n$, a növekvő sorrendbe rendezett adatokat úgy, hogy $x_{(1)}$ a legkisebb és $x_{(n)}$ a legnagyobb mérés. Minden egyes $x_{(i)}$ méréshez tartozik egy f_i százalék, ami azt jelzi, hogy az adatok megközelítőleg $100f_i$%-ára teljesül, hogy kisebb vagy egyenlő, mint az $x_{(i)}$ érték. Azt



5.6. ábra | Az 5.14. táblázat adathalmazának hisztogramja

mondjuk, hogy „megközelítőleg", mivel esetleg nincs olyan érték, amelyre az adatok pontosan f_i törtrésze kisebb vagy egyenlő, mint $x_{(i)}$. Megjegyezzük, hogy a 0,25 kvantilis a Q_1 első kvartilisnek, a 0,50 kvantilis a mediánnak és a 0,75 kvantilis a Q_3 harmadik kvartilisnek felel meg. Legyen:

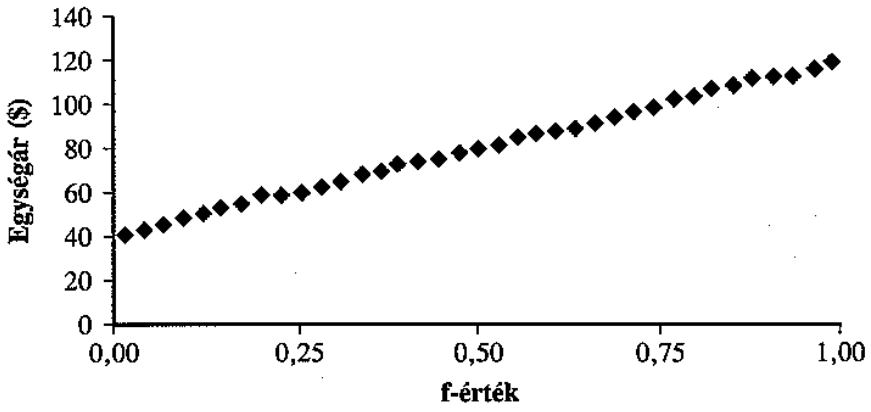
$$f_i = \frac{i - 0{,}5}{n}.$$

Ezek a számok $1/n$-es egyenlő lépésközökkel növekednek $1/n$-től kezdődően (ami közelít 0-hoz) az $1 - 1/2n$ értékig (ami közelít 1-hez). A kvantilis ábra az $x_{(i)}$-t az f_i függvényében ábrázolja. Ez lehetővé teszi számunkra a különböző eloszlások kvantiliseik alapján történő összehasonlítását. Például, ha adott két különböző időszak eladási adatainak kvantilis ábrája, ránézésre összehasonlíthatjuk a Q_1, a medián, a Q_3 és a többi f_i értékeket. Az 5.7. ábra az 5.14. táblázat *egységár* adatainak kvantilis ábráját mutatja.

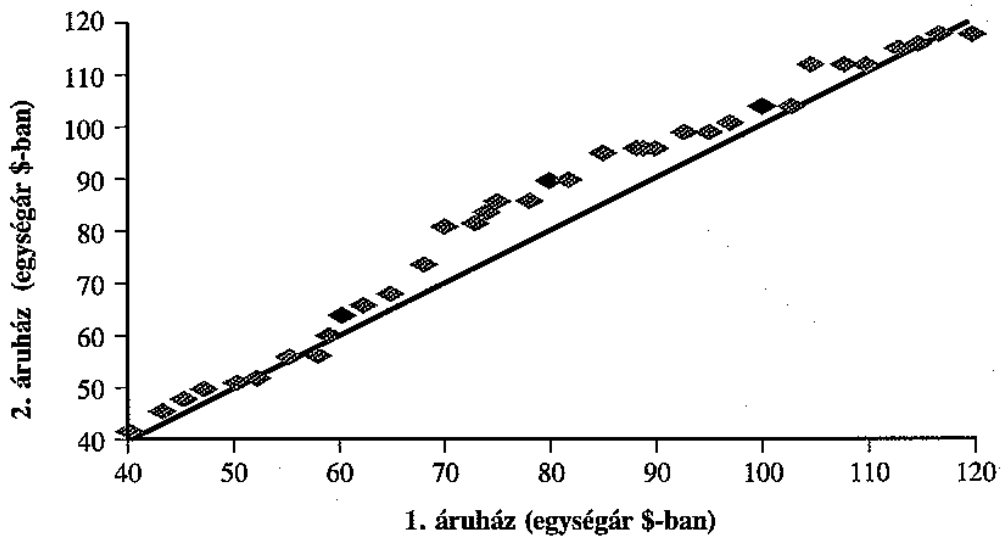
Egy **kvantilis-kvantilis** vagy **q-q ábra** egyváltozós eloszlás kvantiliseit ábrázolja egy másik eloszlás megfelelő kvantiliseinek függvényében. Ez hatásos megjelenítési eszköz, amely lehetővé teszi a felhasználó számára annak megvizsgálását, hogy van-e eltolódás az egyik eloszlás javára a másikkal szemben.

Tegyük fel, hogy van két adathalmazunk az *egységár* változó értékeire, két különböző áruházból. Legyen $x_{(1)}, \ldots, x_{(n)}$ az első és $y_{(1)}, \ldots, y_{(m)}$ a második áruház adatai, ahol mindkét adathalmaz növekvő sorrendben rendezett. Ha $m = n$ (azaz a pontok száma mindkét halmazban azonos), akkor egyszerűen ábrázoljuk $y_{(i)}$-t az $x_{(i)}$ függvényében, ahol $y_{(i)}$ és $x_{(i)}$ a saját megfelelő adathalmazának egyaránt $(i - 0{,}5)/n$ kvantilise. Ha $m < n$ (azaz a második fióknak kevesebb adata van, mint az elsőnek), akkor csak m pont van a q-q ábrán. Itt $y_{(i)}$ az $(i - 0{,}5)/m$ kvantilise az y adatoknak, amely grafikusan ábrázolt az x adat $(i - 0{,}5)/m$ kvantilisének függvényében. Ez a kiszámítás általában interpolációt igényel.

Az 5.8. ábra egy adott időszakra vonatkozóan az *ElektroMind* adatbázis két különböző áruházában eladott árucikkek egységáradatainak kvantilis-kvantilis ábráját mutatja. A bal oldali sarokban lévő legkisebb értékű pont a két adathalmaz azonos, 0,33 értékű



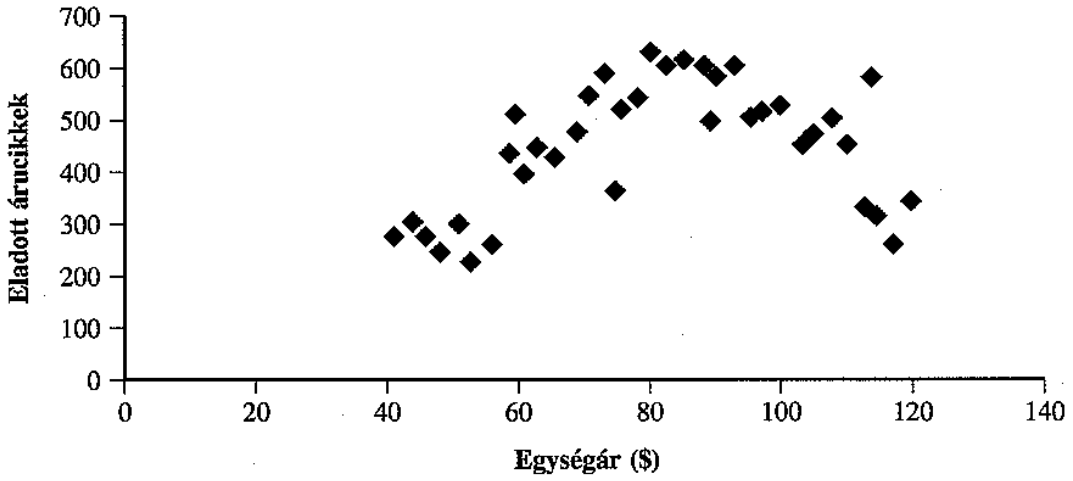
5.7. ábra | Az 5.14. táblázat egységáradatainak kvantilis ábrája



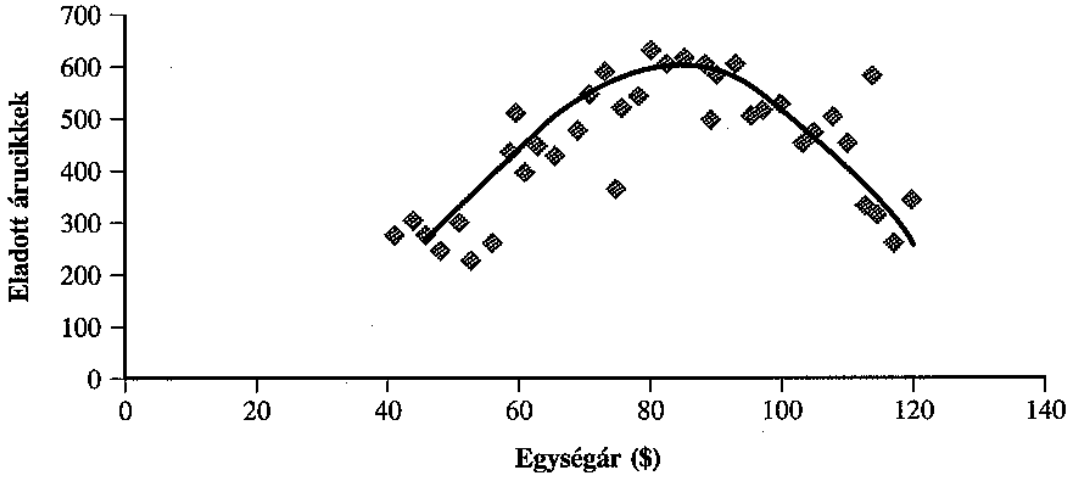
5.8. ábra | Két különböző áruház egységáradatainak kvantilis-kvantillis ábrája

kvantilisének felel meg. (Az összehasonlítás megkönnyítésére bejelölünk egy segédegyenest, amely azokat az eseteket mutatja, amelyeknél minden egyes adott kvantilisre az egyes áruházak egységára azonos. Továbbá a sötétebb pontok rendre a Q_1-hez, a mediánhoz, és a Q_3-hoz tartozó adatoknak felelnek meg.) Például ennél a kvantilisnél azt láthatjuk, hogy az 1. áruházban eladott árucikkek egységára kisebb, mint a 2. áruházban eladottaké. Más szavakkal, az 1. áruházban eladott árucikkek 3%-ának az egységára 40 $ vagy annál kevesebb volt, miközben a 2. áruház árucikkeinek 3%-a 42 $ vagy annál kevesebb. A legmagasabb kvantilisnél azt látjuk, hogy a 2. áruházban eladott árucikkek egységára kissé kevesebb, mint az 1. áruházban. Általánosságban megjegyezhetjük, hogy eltolódás van az 1. áruház eloszlásában a 2. áruház javára, abban az értelemben, hogy az 1. áruház eladott árucikkeinek egységárai inkább alacsonyabbak, mint a 2. áruház esetén.

A **szórásfüggvény** az egyik leghatékonyabb grafikus módszer annak meghatározására, hogy van-e kapcsolat, séma, tendencia két mennyiségi változó között. A szórásfüggvény létrehozásához minden egyes értékpárt algebrai értelemben vett koordinátapárként kezelünk és síkbeli pontként ábrázolunk. A szórásfüggvény hasznos vizsgálati módszert ad kétváltozós adatok síkbeli eloszlásának, valamint a pontok csoportosulásainak, szélső-



5.9. ábra | Az 5.14. táblázat adathalmazának szórásfüggvénye



5.10. ábra | Az 5.14. táblázat adathalmazának loess görbéje

séges értékeinek stb. gyors áttekintésére. Az 5.9. ábra mutatja az 5.14. táblázat adatainak szórásfüggvényét.

A **loess görbe** egy másik fontos grafikus vizsgálati segédeszköz, amely a szórásfüggvényre egy folytonos görbét illeszt azért, hogy jobban érzékelhető legyen az összefüggések sémája. A *loess* szó a „helyi regresszió" (local regression) rövidítése. Az 5.10. ábra az 5.14. táblázat adathalmazának loess görbéjét mutatja.

A loess görbe illesztéséhez két paraméter értékének beállítása szükséges – az α illeszkedési paraméter és a λ, amely a regresszió által illesztett polinom foka. Amíg α egy tetszőleges pozitív szám lehet (tipikusan ¼ és 1 közötti érték), addig λ értéke 1 vagy 2. Az α megválasztásánál a cél egy olyan illeszkedés megvalósítása, amely annyira simul, amennyire csak lehetséges, anélkül hogy túlságosan torzítaná az alapul szolgáló adatmintát. A görbe az α növelésével simábbá válik. Lehet azonban némi zavar az illeszkedésben, jelezvén az adatminták esetleges hiányosságát. Ha α nagyon kicsi, a görbe követi az alapul szolgáló mintát, de azért előfordulhat túlillesztett adat, ahol a görbe helyi tekervényeit az adatok nem támasztják alá. Ha az alapul szolgáló adatmintának helyi maximum

és minimum nélküli „enyhe" görbülete van, akkor a helyi lineáris illesztés általában kielégítő ($\lambda = 1$). Ha azonban van helyi maximuma vagy minimuma, akkor a helyi négyzetes illesztés ($\lambda = 2$) általában jobb megoldás az adatminta-követés és a helyi illeszkedés fenntartása szempontjából.

5.6. | Tárgyalás

Bemutattunk skálázható módszereket a nagy adatbázisok fogalmi vagy osztályleírásainak bányászatára. Ebben az alfejezetben az ilyen leírásokhoz kapcsolódó témákat tárgyaljuk. Ezek tartalmazzák az adatkocka alapú összehasonlításos és az attribútumorientált indukciós megközelítéseket a szokásos gépi tanulási módszerekkel történő adatáltalánosításra, a fogalomleírások inkrementális és párhuzamos bányászatának megvalósítását és a fogalomleírások érdekességi mértékeit.

5.6.1. | Fogalomleírás: tipikus gépi tanulási módszerekkel történő összehasonlítás

Ebben a fejezetben nagy adatbázisok fogalomleírásainak bányászatára adatbázis alapú módszereket tanulmányoztunk. Ezek a módszerek tartalmazzák az adatkocka alapú és az attribútumorientált indukciós megközelítést az adatáltalánosításra a fogalomleírásnál. Más hatásos fogalomleírási módszereket is javasoltak és tanulmányoztak 1980 óta a gépi tanulásról szóló irodalomban. A fogalomleírás tipikus gépi tanulási módszerei a *példák alapján történő tanulás* paradigmáját követik. Az ilyen módszer általában a fogalomhalmazokon vagy osztálycímkézett gyakorló mintákon dolgozik abból a célból, hogy a tanulmányozott osztály leírására hipotézist származtasson vagy tanuljon.

Mi a fő különbség a példák alapján történő tanulás módszerei és az itt bemutatott adatbányászati módszerek között? Először is, különbség van a gépi tanulási és az adatbányászati megközelítések filozófiájában és a fogalomleírás problémájára vonatkozó *alapfeltevésekben*. A legtöbb gépi tanulás kapcsán kifejlesztett példák alapján történő tanulási algoritmusban az elemzett mintahalmaz két részhalmazra oszlik: *pozitív* és *negatív mintákra*, rendre a célosztályt és a kontrasztosztályt reprezentálókra. A tanulási eljárás véletlenszerűen kiválaszt egy pozitív mintát, és azt használja a hozzá tartozó osztály objektumait leíró hipotézis kialakítására. A tanulási eljárás ezután *általánosítást* hajt végre a hipotézisen a maradék pozitív mintákat használva, és *specializációt* a negatív mintákat használva. Általában az eredményül kapott hipotézis az összes pozitív mintát lefedi, de a negatív minták közül egyet sem.

Az adatbázis általában nem tárolja kifejezetten a negatív adatokat. Így specializációra csak nem kifejezetten meghatározott negatív minták használhatók. Ez az, amiért az analitikus jellemzés bányászata és általában az összehasonlító bányászat esetén az adatbányászati módszereknek olyan összehasonlítható adatokat kell összegyűjtenie negatív adatként való felhasználásra, amelyek nincsenek a cél- (pozitív) osztályban (5.3. és 5.4. alfejezet). A legtöbb adatbázis alapú módszer ennek következtében általánosítás alapú is. Annak ellenére, hogy a legtöbb lehetővé teszi a lefúrás (specializáció) műveletét, az valójában az előző állapotba történő visszalépéssel megvalósított általánosítási eljárás.

Másodszor, további fontos különbség a gépi tanulás és a fogalomleírások adatbázis alapú technikái között a *gyakorló minták halmazának méretére* vonatkozik. A hagyományos gépi tanulási módszerek esetén a gyakorló halmaz általában viszonylag kicsi, összehasonlítva az adatbázis alapú technikák által elemzett adatokkal. Ezért a gépi tanulási módszerek esetén könnyebb megtalálni azokat a leírásokat, amelyek az összes pozitív mintát lefedik, de a negatívak közül egyet sem. Azonban a valós élet adatbázisaiban tárolt adatok sokféleségére és hatalmas mennyiségére tekintettel, valószínűtlen, hogy az ilyen adatok elemzése során kapott szabály vagy séma, amely az összes pozitív mintát lefedi, nem fed le a negatívak közül egyet sem. Ehelyett esetleg elvárható, hogy megtaláljuk azon tulajdonságok vagy szabályok halmazát, amelyek a pozitív osztály adatainak *többségét* lefedik, *maximálisan* megkülönböztetve a pozitív példákat a negatívaktól. (Ez *valószínűségi eloszlásként* is leírható.)

Harmadszor, különbség van a gépi tanulási és az adatbázis alapú megközelítések között a *használt általánosítási módszerek* vonatkozásában is. Mindkét megközelítés alkalmazza az attribútumelhagyást és az attribútumáltalánosítást mint fő általánosítási technikákat. Tekintsük a gyakorló minták halmazát sorok halmazaként. A gépi tanulás *sorról sorra* történő összehasonlítást hajt végre, míg az adatbázis alapú megközelítés esetén az általánosítás *attribútumról attribútumra* (vagy a teljes dimenzióra) hajtódik végre.

A gépi tanulás sorról sorra történő stratégiájában a gyakorló minták időben egyesével kerülnek megvizsgálásra azért, hogy az általánosított fogalmakat indukálják. A legjellemzőbb hipotézis (vagy fogalomleírás) kialakításához, amely megfelel az összes pozitív példának, de a negatívak közül egyiknek sem, az algoritmusnak meg kell vizsgálnia minden csomópontot a vizsgálati térben, amely az egyes gyakorló példák általánosításából származó összes lehetséges fogalmat reprezentálja. Mivel a sorok különböző attribútumai esetleg az absztrakció eltérő szintjeire lesznek általánosítva, egy adott gyakorló minta esetén a vizsgált csomópontok száma nagyon sok lehetséges kombinációt foglalhat magában.

Másrészről, az attribútumorientált stratégiát alkalmazó adatbázis megközelítés általánosítást hajt végre az általánosítás *kezdeti* szakaszában minden egyes attribútumon vagy dimenzión, egységesen az összes adatreláció sorra. Az ilyen megközelítés valójában az egyedi attribútumokra összpontosítja a figyelmet, mintsem attribútumcsoportokra. Erre *verziótér faktorizációként* hivatkozhatunk, ahol a **verzióteret** a tanuló mintákkal konzisztens hipotézisek halmazaként definiálhatjuk. A verziótér faktorizációja lényegesen javítani tudja a kiszámítás hatékonyságát. Tegyük fel, hogy k fogalmi hierarchiát használunk az általánosításban, és p csomópontunk van minden egyes fogalmi hierarchiában. A k-faktorú verziótér teljes mérete $p \times k$. Ezzel szemben a gépi tanulási megközelítés által vizsgált faktorizálatlan verziótér mérete ugyanezen fogalmi fa esetén p^k.

Megjegyezzük, hogy azok az algoritmusok, amelyek a kezdeti általánosítási szakaszokban a különböző attribútum–érték feltételeknek több lehetséges kombinációját vizsgálják adott nagyszámú sorra, nem lehetnek hatékonyak, mivel az ilyen kombinációk egyesítése csak a további általánosítások során történik meg. A különböző lehetséges kombinációkat csak akkor kell megvizsgálni, amikor a reláció először kerül általánosításra egy viszonylag kisebb relációba, ahogy az az előző fejezetben leírt adatbázis alapú megközelítésnél történik.

Az attribútumorientált megközelítés további nyilvánvaló előnye sok más gépi tanulási algoritmussal szemben az *adatbányászati eljárás integrálása a halmazorientált adatbázis-*

műveletekkel. A legtöbb létező tanulási algoritmussal ellentétben, amelyek nem nyújtják az adatbázis adta lehetőségek minden előnyét, az attribútumorientált indukciós megközelítés elsődlegesen relációs műveleteket alkalmaz, például: kiválasztás, összekapcsolás, vetítés (feladat szempontjából lényeges adat kiválasztása és attribútumelhagyás), sorhelyettesítés (felemelkedés a fogalmi fákban) és rendezés (osztályok közötti gyakori sorok feltárása). Az ilyen relációs műveletek halmazorientáltak és általában optimalizálva vannak a legtöbb létező adatbázisrendszerben. Továbbá az attribútumorientált megközelítés nemcsak hatékony, hanem könnyen exportálható is más relációs rendszerekbe. Ez a megjegyzés az adatkocka alapú általánosítási algoritmusokra is vonatkozik. Az adatkocka alapú megközelítés több optimalizáló technikát vizsgál, mint a hagyományos adatbázis-lekérdező eljárási technikák, tartalmazva a ritka kocka technikákat, a különféle kockakiszámítási módszereket éppúgy, mint az indexelési és elérési technikákat. Továbbá, nagy adathalmazok kezelésekor magas teljesítménynövekedés várható el az adatbázis alapú algoritmusok esetén, szemben a gépi tanulási technikákkal.

5.6.2. | A fogalomleírás növekményes és párhuzamos bányászata

Nagyon nagy mennyiségű adatot tartalmazó adatbázis esetén lényegesen előnyösebb az adatbányászati eredményeket *növekményesen (inkrementálisan)* módosítani, mintsem az adatbázis minden egyes módosításánál a kiinduló állapotból bányászni. Így az inkrementális adatbányászat vonzó cél a nagy adatbázisokban és adattárházakban a különféle bányászatok számára. Szerencsére az adatbázis alapú fogalomleírást bányászó algoritmusok egyszerűen kiterjeszthetők az inkrementális adatbányászatra.

Hogyan terjeszthető ki az attribútumorientált indukció a növekményes adatbányászat során történő felhasználásra? Tegyük fel, hogy az R általánosított relációt tároljuk az adatbázisban. Amikor az új sorok egy ΔDB halmazát kell beszúrnunk az adatbázisba, az attribútumorientált indukciót végre lehet hajtani a ΔDB halmazon azért, hogy az attribútumokat ugyanazokra a fogalmi szintekre általánosítsuk, mint az általánosított R relációban rendre a megfelelő attribútumok. A megfelelő összesítőinformációk (például `count`, `sum` stb.) kiszámíthatók az általánosítási algoritmust ΔDB-re alkalmazva. Az így a ΔDB-n kapott ΔR általánosított reláció ezután könnyen egyesíthető az R általánosított relációval, mivel R-nek, és ΔR-nek ugyanazok a dimenziói, és az egyes dimenziók ugyanazon az absztrakciós szinten vannak. Az $R \cup \Delta R$ lesz az új általánosított R' reláció. A felhasználó által megadott kisebb változtatások, például dimenzió általánosítás vagy specializáció, végrehajthatók az R' reláción, ha szükséges. Az növekményes törlés hasonló módon hajtható végre. Ennek részletezését gyakorlatra hagyjuk.

Adat-mintavételezési módszerek, párhuzamos algoritmusok, és osztott algoritmusok azonos filozófián alapulóan vizsgálhatók fogalomleírás bányászata esetén. Például az attribútumorientált indukció végrehajtható a feladat szempontjából lényeges igen nagy adathalmazból mintavételezéssel kiválasztott részhalmazon, vagy először párhuzamosan végrehajtva az indukciót a feladat szempontjából lényeges adathalmaz több partícióján, majd azután egyesítve az általánosított eredményeket.

5.7. | Összefoglalás

- Az adatbányászaton belül megkülönböztetünk *leíró adatbányászatot* és *előrejelző adatbányászatot*. A leíró adatbányászat legalapvetőbb formája a **fogalomleírás**, amely tömör és összegző módon írja le a feladat szempontjából lényeges adatok adott halmazát, bemutatva az adatok érdekes, általános tulajdonságait.
- A fogalom- (vagy osztály-) leírás **jellemzésből** és **összehasonlításból** (vagy **megkülönböztetésből**) áll. Az előbbi egy adatgyűjtemény (amit **célosztálynak** nevezünk) összegzése és leírása; míg az utóbbi összegzi és megkülönbözteti az egyik adathalmazt (a **célosztályt**), más adathalmaz(ok)tól, amely(ek)et együttesen **kontrasztosztály(ok)nak** hívunk.
- A fogalmi jellemzés két fő megközelítése: az **adatkocka OLAP alapú megközelítés**, és az **attribútumorientált indukciós szemlélet**. Mindkettő attribútum vagy dimenzió alapú általánosítási eljárás. Az attribútumorientált indukciós szemlélet megvalósítható relációs vagy adatkocka struktúrákkal is.
- Az **attribútumorientált indukciós szemlélet** az alábbi technikákból áll: *adatáttekintés, adatáltalánosítás attribútumelhagyással vagy attribútumáltalánosítással, darabszám és összesítő értékek halmozása, attribútum-általánosítási vezérlés és általánosított adatmegjelenítés*.
- Az általánosított adat többféle formában is **megjeleníthető**, beleértve az általánosított relációkat, kereszttáblázatokat, oszlop-, kördiagramokat, adatkockanézeteket, grafikonokat és szabályokat. A lefúrás és felgörgetés műveletek interaktívan végrehajthatók az általánosított adatokon.
- Az **analitikus jellemzés/összehasonlítás** attribútum- és dimenziórelevancia-elemzést használ a lényegtelen vagy kevésbé lényeges attribútumok kiszűrésére az indukciós eljárást megelőzően.
- A **fogalmi összehasonlítást** a fogalmi jellemzéshez hasonló módon lehet végrehajtani attribútumorientált indukciót vagy adatkocka megközelítést használva. A célosztály és kontrasztosztályok általánosított sorai mennyiségileg összehasonlíthatók és szembeállíthatók.
- A jellemzési és összehasonlítási leírások (amelyek a fogalomleírást alkotják) együttesen megjeleníthetők *ugyanazon* általánosított relációban, kereszttáblázatban vagy mennyiségi szabályban, bár különböző érdekességi mértékekkel ellátva. Ezek a mértékek lehetnek a **t-súly** (a sor tipikusságát jelzi) és a **d-súly** (a sor megkülönböztethetőségét jelzi).
- Leíró statisztikai szempontból további **statisztikai mértékek** használhatók az elhelyezkedés és az adatszóródás leírására. A kvartilisek, szórásnégyzetek és szélsőséges értékek mind hasznos kiegészítő információk, amelyeket adatbázisokban bányászhatunk. A dobozdiagramok, kvantilis ábrák, eloszlásfüggvények és kvantilis-kvantilis ábrák hasznos megjelenítési eszközök a leíró adatbányászatban.
- A gépi tanulási algoritmusokkal összehasonlítva az adatbázis alapú fogalomleírás hatékonysághoz és skálázhatósághoz vezet a nagy adatbázisokban és adattárházakban.
- A fogalomleírás bányászata végrehajtható **növekményesen**, **párhuzamosan** vagy **osztott** módon az alapmódszerekben minimális kiegészítéseket alkalmazva.

5.8. | Feladatok

5.1. Melyek a fontosabb különbségek *osztályjellemzés* esetén az adatkocka alapú megvalósítás és a relációs megvalósítás, például az attribútum alapú indukció között. Vizsgáljuk meg, hogy melyik módszer a leghatékonyabb és milyen feltételek esetén használható.

5.2. Tegyük fel, hogy az alábbi táblázat egy attribútum alapú indukcióból származik.

osztály	születési_hely	darabszám
	Kanada	180
Programozó	mások	120
	Kanada	20
Rendszergazda	mások	80

(a) Alakítsuk át a táblázatot egy kereszttáblázattá, amely a megfelelő t-súly és d-súly értékeket is mutatja.

(b) Képezzük le a *programozó* osztályt egy (kétirányú) *mennyiségi leíró szabállyá*, például:

$$\forall X,\ programozó(X) \Leftrightarrow (születési\_hely(X) = \text{“}Kanada\text{”} \wedge \ldots)$$
$$[t : x\%, d : y\%] \vee \ldots \vee (\ldots)[t : w\%, d : z\%].$$

5.3. Ismertessük, miért szükséges az analitikus jellemzés és hogyan hajtható végre. Hasonlítsuk össze az (1) relevanciaelemzést használó és a (2) relevanciaelemzés nélküli indukciós módszer eredményét.

5.4. Adjunk meg három általánosan használt további statisztikai mértéket (azaz ebben a fejezetben nem leírtat) az adateloszlás jellemzésére és tárgyaljuk meg, hogyan számíthatók ki hatékonyan nagy adatbázisokban.

5.5. Tegyük fel, hogy az elemzés adatai között szerepel az *életkor* attribútum. Az adatsorokban előforduló *életkor* értékek (növekvő sorrendben): 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

(a) Számítsuk ki az adatok középértékét és a mediánt!

(b) Mennyi az adatok módusza? Vizsgáljuk meg az adathalmaz modalitását (kétmóduszú, hárommóduszú stb.)!

(c) Mi az adat terjedelem-közép?

(d) Hol van (nagyjából) az adatok első kvartilise (Q_1) és a harmadik kvartilise (Q_3)?

(e) Adjuk meg az adatok 5-számos összegzését!

(f) Rajzoljuk meg az adatok dobozdiagramját!

(g) Mennyiben különbözik a kvantilis-kvantilis ábra a kvantilis ábrától?

5.6. Adott a *DB* adatbázisból származó *R* általánosított reláció, és tegyük fel, hogy a ΔDB halmaz sorait törölni kell a *DB*-ből. Vázoljunk fel egy növekményes módosító eljárást a szükséges törlésekre, R-re vonatkozóan.

5.7. Vázoljunk fel egy adatkocka alapú növekményes algoritmust az analitikus osztályösszehasonlítások bányászatára.

5.8. Vázoljunk módszert adatkocka környezetben adateloszlások statisztikai mértékeinek (1) párhuzamos és (2) osztott bányászatára.

5.9. | Irodalom

Az általánosítási és összegzési módszereket a statisztikai irodalom már jóval a számítógépek megjelenése előtt tanulmányozta. A statisztikai leíró adatbányászati módszerek jó összefoglalását tartalmazzák Cleveland [Cle93] és Devore [Dev95] könyvei.

Az általánosításon alapuló indukciós technikákat, mint amilyen a példákból tanulás, már az adatbányászati felhasználásuk előtt ajánlották és tanulmányozták a gépi tanulásról szóló irodalomban. Az induktív tanulás elméletét és módszertanát Michalski [Mic83] vezette be. A példákból tanulás módszerét Michalski [Mic83] indítványozta. A verziótér bevezetését Mitchell [Mit77, Mit82] javasolta. Az 5.6.1. alfejezetben leírt verziótér faktorizációs módszert Subramanian és Feigenbaum mutatta be [SF86b]. A gépi tanulási technikákról áttekintés található az alábbiakban: Dietterich és Michalski [DM83], Michalski, Carbonell és Mitchell [MCM86] és Mitchell [Mit97].

Az adatkocka alapú általánosítási technikákat először E. F. Codd, S. B. Codd és Salley javasolta [CCS93], és meg is valósították több OLAP-alapú adattárház rendszerben. Gray, Bosworth, Layman és Pirahesh [GCB+97] ajánlotta a kocka operátort az adatkocka összesítések kiszámítására. Napjainkban számos tanulmány foglalkozik az adatkockák hatékony kiszámításával, amelyek hozzájárulnak az adatáltalánosítás hatékony kiszámításához. A téma részletes áttekintése megtalálható Chaudhuri és Dayal [CD97] cikkében.

A fogalomleírás adatbázis alapú módszerei az adatbázisok és az adattárházak nagy adathalmazai leírásának skálázható és hatékony technikáit vizsgálják. Az ebben a fejezetben leírt attribútumorientált indukciós módszert először Cai, Cercone és Han [CCH91] javasolta, és további kiterjesztése található: Han, Cai és Cercone [HCC93], Han és Fu [HF96], Carter és Hamilton [CH98] és Han, Nishio, Kawano és Wang [HNKW98] cikkekben.

Az attribútumrelevancia meghatározására több módszer létezik, de mindegyiknek megvan a maga korlátja. Az információnyereség mértéke sok értéket tartalmazó attribútumok esetén torzít. Többféle változatot javasoltak, például a nyereségarányt (Quinlan [Qui93]), amely az egyes attribútumértékek valószínűségeit is figyelembe veszi. További relevanciamértékek a Gini index (Breiman, Friedman, Olshen és Stone [BFOS84]), a χ^2 eloszlási tábla statisztika és a bizonytalansági együttható (Johnson és Wichern [JW92]). A döntési fa indukció esetén az attribútum-kiválasztási mértékek összehasonlítására lásd Buntine és Niblett [BN92]. További módszerek találhatók Liu és Motoda [LM98ab], Dash és Liu [DL97], ill. Almuallim és Dietterich [AD91] publikációkban.

A statisztikai alapú adatmegjelenítésre dobozdiagramokat, kvantilis ábrákat, kvantilis-kvantilis ábrákat, eloszlásfüggvényeket és loess görbéket használunk, lásd Cleveland [Cle93] és Devore [Dev95]. Knorr és Ng [KN98] a szélsőséges értékek meghatározásának és kiszámításának egységes megközelítését tanulmányozta.

6. FEJEZET

Társítási szabályok bányászata nagy adatbázisokban

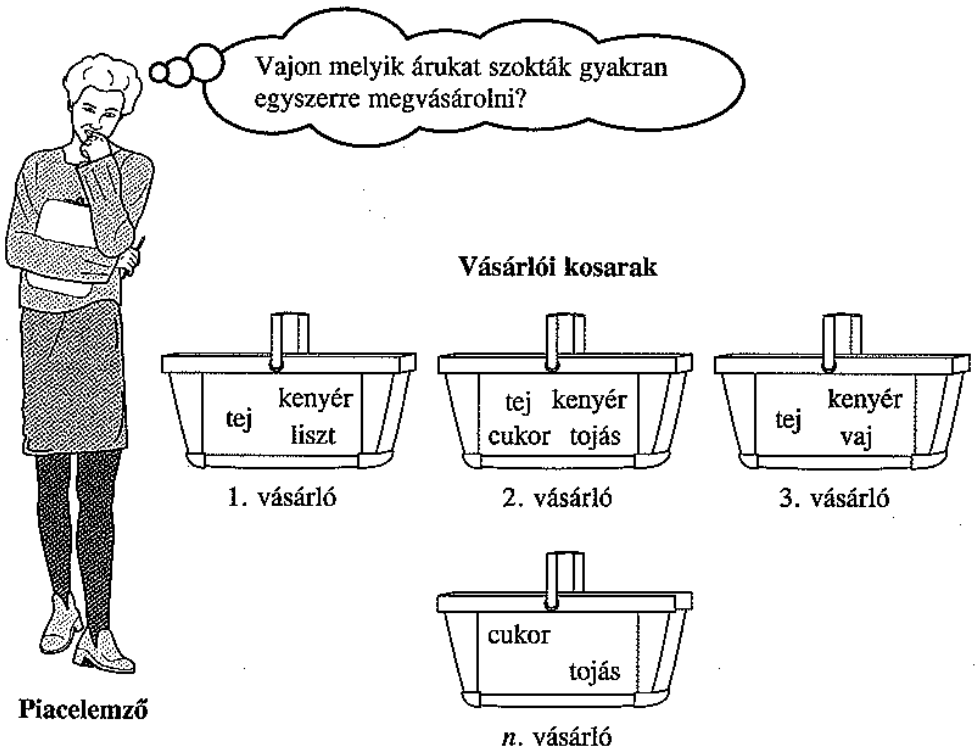
A társítási szabályok bányászata az adatelemek nagy halmazában érdekes társításokat vagy összefüggéseket keres. Nagy adathalmazok gyűjtése és tárolása révén több cégnél felmerült az érdeklődés az adatbázisaikban rejlő összefüggések, társítási szabályok keresése iránt. Üzleti tranzakciók adatainak óriási halmazában az érdekes, fontos összefüggések felfedezése sokféle olyan döntéshozatali helyzetben – például katalógustervezés, keresztértékesítés vagy bevezető árak elemzése – adhat segítséget.

A társítási szabályok bányászatára tipikus példa a vásárlói kosár elemzése. Ebben a feladatban a vásárlói szokások elemzésével társítási szabályokat keresnek a vásárlói kosárba került különböző áruk között (6.1. ábra). Ilyen összefüggések – azaz, hogy mely árukat vásárolják gyakran együtt – felismerése segíti a kereskedő hatékony üzleti stratégiájának kialakítását. Társítási kérdés például, hogy a tejet vásárló mennyire tipikusan vásárol kenyeret is (és éppen milyen fajta kenyeret) ugyanazon bevásárlás alkalmával. Az ilyen információk ötletet adhatnak a helyes üzleti magatartáshoz, a kereskedőt a forgalom növeléséhez segíthetik. A példánkban a tej és a kenyér egymáshoz közeli elhelyezése megnövelheti az egyszerű bevásárlások forgalmát.

Hogyan találhatók meg a társítási szabályok nagy, tranzakciós vagy relációs adathalmazokban? Mely társítási szabályok a valóban érdekesek? Hogyan tudjuk segíteni vagy irányítani a bányászó eljárást a valóban érdekes szabályok felismerésében? A társítási szabályok bányászata szempontjából milyen nyelvi konstrukciók alkalmasak adatbányászati lekérdező nyelvek definiálására? Ebben a fejezetben ezeknek a kérdéseknek ásunk a mélyére.

6.1. | Társítási szabályok bányászata

A társítási szabályok bányászata az adott adathalmazban előforduló elemek közötti érdekes összefüggéseket keresi. Ez a fejezet a társítási szabályok bányászatához ad ismereteket. Kezdésként a 6.1.1. alfejezetben egy vásárlói kosár elemzésének példáján mutatjuk



6.1. ábra | A vásárlói kosár elemzése

be a társítási szabályok keresésének korai módszerét. A társítási szabályok keresésének alapkoncepcióját a 6.1.2. alfejezetben adjuk meg. A 6.1.3. alfejezetben rendszerezzük a bányászással megtalálható különböző típusú társítási szabályokat.

6.1.1. | Vásárlói kosár elemzése – példa, amely motiválta a társítási szabályok bányászatát

Tegyük fel, hogy Ön, mint az *ElektroMind* cég igazgatója, szeretne többet megtudni a vevők vásárlási szokásairól. Konkrétan arra kíváncsi, hogy *egy-egy vásárlás során mely árucikkcsoportokat vagy árucikkcsomagokat vásárolják szívesen a vevők?* A kérdés megválaszolásához elemezni kell a vásárlói kosarat az áruház vásárlói forgalmi adataiból. A kapott eredmény használható a kereskedelmi vagy hirdetési stratégia megtervezésében, például az árukatalógus összeállításában. A vásárlói kosár elemzése segítheti az áruk elrendezésének megtervezését is. Az egyik stratégia szerint a gyakran együtt vásárolt termékeket egymás közelében célszerű elhelyezni, ezzel is elősegítve azok együttes megvásárlását. Ha az elemzés szerint a számítógépet vásárlók pénzügyi programokat is szoktak vásárolni, akkor a hardver- és szoftverkínálat egymáshoz közeli elhelyezése segítheti mindkét áruféle forgalmának növelését. A másik stratégia, mely szerint a hardver- és szoftvertermékeket (a tipikusan együtt vásárolt termékeket) az áruház két szemközti végében kell elhelyezni, elősegítve ezáltal azt, hogy a vásárló az egyik terméktől a másikig tartó útján egyéb árucikkekkel is találkozzon (s esetleg azokból is vásároljon). Például egy drága számítógép megvásárlásának eldöntése után a szoftverkínálat felé haladtában észrevehet egy biztonsági rendszert, s meglehet, hogy úgy dönt, ezt is megvásárolja.

A vásárlói kosár elemzése a kereskedő számára segítséget adhat a kedvezményes akciók megtervezésében is. Ha azt tapasztaljuk, hogy a vásárlók a számítógépeket és nyomtatókat gyakran együtt veszik meg, akkor a nyomtatókra adott engedmény elősegíti a nyomtatók *és* a számítógépek eladását *is*.

Ha univerzumnak az áruházban elérhető árucikkek összességét tekintjük, akkor minden árucikkhez Boole-változót rendelhetünk, jelezvén annak jelenlétét vagy hiányát (az adott vásárlásban). Így minden vásárlói kosár a változókhoz tartozó elemekből álló Boole-vektorral jellemezhető. A vásárlások ezen Boole-vektorainak elemezésével a tipikusan *társított*, együtt vásárolt árucikkek is megtalálhatók. A felismert összefüggések **társítási szabályokkal** írhatók le. Az az információ például, miszerint a számítógépet vásárlók gyakran pénzügyi programokat is vesznek, az alábbi (6.1) társítási szabállyal írható le:

$$Számítógép \Rightarrow pénzügyi\_szoftver[megalapozottság = 2\%, megbízhatóság = 60\%]. \quad (6.1)$$

A szabály megalapozottsága és megbízhatósága a szabály érdekességének két, mértékkel jellemzett tulajdonsága, melyeket korábban a 4.1.4. alfejezetben írtunk le. Ezek a felfedezett szabály hasznosíthatóságát és valószínűségét jellemzik. A (6.1) szabály 2%-os megalapozottsága azt jelenti, hogy az elemezett tranzakciók 2%-a mutatta azt, hogy a számítógépeket és pénzügyi programokat együtt vásárolták meg. A 60%-os megbízhatóság pedig az jelenti, hogy a számítógépet vásárlók 60%-a vásárolt pénzügyi programot is. Általában azt a társítási szabályt tekintjük érdekesnek (fontosnak), amely felette van valamely **minimális megalapozottsági** és **minimális megbízhatósági küszöbszintnek.** Ezeket a küszöbszinteket a felhasználó vagy a szakterület szakértője állítja be.

6.1.2. | Alapfogalmak

Legyen $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ elemek halmaza. Jelölje D a vizsgált adatok halmazát, ez az adatbázis-tranzakciók halmaza, ahol minden T tranzakció elemek halmaza, így $T \subseteq \mathcal{I}$. Minden tranzakcióhoz egy TID-nek (transaction identifier) nevezett azonosítót rendelünk. Legyen A az árucikkek egy halmaza. Akkor és csakis akkor mondjuk, hogy a T tranzakció tartalmazza A-t, ha $A \subseteq T$. A társítási szabály egy $A \Rightarrow B$ implikáció, ahol $A \subset \mathcal{I}$, $B \subset \mathcal{I}$ és $A \cap B = \varnothing$. Az $A \Rightarrow B$ szabály D tranzakcióhalmazra vonatkozó **megalapozottsága** s, ha D tranzakcióinak s százaléka tartalmazza $A \cup B$-t (tehát mind A-t, mind B-t). A megalapozottság tehát $P(A \cup B)$ valószínűségként adódik. Az $A \Rightarrow B$ szabály D tranzakcióhalmazra vonatkozó **megbízhatósága** c, ha D tranzakciói közül c százalékra igaz, hogy ha tartalmazza A-t, akkor tartalmazza B-t is. A megbízhatóság tehát $P(B|A)$ feltételes valószínűség. Így:

$$megalapozottság(A \Rightarrow B) = P(A \cup B), \quad (6.2)$$

$$megbízhatóság(A \Rightarrow B) = P(A|B). \quad (6.3)$$

Azokat a szabályokat, amelyek megfelelnek mind a minimális megalapozottság (*min_sup*), mind a minimális megbízhatóság (*min_conf*) követelményének (azaz megala-

pozottságuk és megbízhatóságuk nagyobb az előírt minimális értékeknél), **erős szabályoknak** nevezzük. Megegyezés szerint a megalapozottságot és a megbízhatóságot 0 és 100% közé eső számmal adjuk meg, nem pedig 0 és 1 közötti (valószínűségi) értékekkel.

Az elemek halmazát a továbbiakban **elemhalmaznak**[1] nevezzük. Egy k elemet tartalmazó elemhalmazt **k-elemhalmazként** említünk. A {*számítógép*, *pénzügyi program*} halmaz egy 2-elemhalmaz. Az **elemhalmaz előfordulási gyakorisága** azoknak a tranzakcióknak a száma, amelyek tartalmazzák az elemhalmazt. Ezt egyszerűen frekvenciának, megalapozottsági számnak vagy az elemhalmazszámnak is nevezik. Az elemhalmaz kielégíti a **minimális megalapozottsági követelményt**, ha az elemhalmaz előfordulási gyakorisága nagyobb vagy egyenlő a D-beli tranzakciók teljes számának és *min_sup*-nak a szorzatánál. Ahhoz, hogy az elemhalmaz kielégítse a minimális megalapozottsági követelményt, megfelelő számú tranzakcióra van szükség. Ezt a számot **minimális megalapozottsági számnak** nevezzük. Ha egy elemhalmaz kielégíti a minimális megalapozottsági követelményt, akkor az **gyakori**[2] elemhalmaz. A gyakori k-elemhalmazok halmazát L_k-val jelöljük.[3]

Hogyan bányásszunk társítási szabályokat nagy adatbázisokban? A társítási szabályok bányászata kétlépéses eljárás:

1. **Megkeressük az összes gyakori elemhalmazt** – A definíciónak megfelelően az összes ilyen elemhalmaznak legalább annyiszor kell előfordulnia, mint az előre meghatározott minimális megalapozottsági.
2. **Előállítjuk a gyakori elemhalmazokra vonatkozó erős társítási szabályokat** – Definíció szerint ezen szabályok kielégítik a minimális megalapozottság és a minimális megbízhatóság követelményeit.

Ha szükséges, további érdekességi mértékeket is alkalmazhatunk (annak eldöntésére, hogy a talált szabályt érdekesnek vagy érdektelennek minősítsük). A két lépésből a második az egyszerűbb. A társítási szabályok keresésének általános hatékonyságát az első lépés határozza meg.

6.1.3. | A társítási szabályok bányászatának feltérképezése

A vásárlói kosár elemzésének módszere csak egyike a társítási szabályok bányászati módszereinek. Valójában sokféle társítási szabály létezik. A társítási szabályokat – a következő szempontokra alapozva – különböző módon osztályozhatjuk:

1 | Az adatbányászattal foglalkozó szakirodalomban az „elemek halmaza" helyett inkább az „elemhalmaz" kifejezést használják.

2 | A korai munkákban a minimális megalapozottsági követelményt kielégítő értékhalmazról mint **nagy** értékhalmazról írtak. Ez a szóhasználat kevésbé egyértelmű, félreértéshez vezethet. Ezért mi a sokkal alkalmasabb **gyakori** megnevezést használjuk.

3 | A 2. megjegyzés szerint a **gyakori** értékhalmazokat korábban **nagy** (large) halmazoknak nevezték, ezen történelmi okból használjuk az L_k jelölést a gyakori k-elemhalmazok halmazára.

- **A szabályban hivatkozott *elemek típusa* alapján** – Ha a szabály olyan társítást jelent, melyben a hivatkozott elemek vagy előfordulnak, vagy nem, akkor **Boole társítási szabályról** beszélünk. Példaként a (6.1) szabály is Boole társítási szabály, melyet a vásárlói kosár elemzés során nyertünk.

  Ha a szabály mérhető elemek vagy attribútumok közötti társítást ír le, akkor az **mennyiségi társítási szabály**. Az ilyen szabályokban az elemek vagy attribútumok mérhető paramétereit intervallumokra osztják. A következő szabály példa a mennyiségi társítási szabályokra, a szabályban az X változó egy vásárlót reprezentál:

$$\textit{életkor}(X, \text{“30...39”}) \wedge \textit{jövedelem}(X, \text{“42 000...48 000”}) \Rightarrow \textit{vásárol}(X, \textit{nagy felbontású TV}) \quad (6.4)$$

  Megjegyezzük, hogy az *életkor* és *jövedelem* mérhető attribútumokat diszkretizáltuk.
- **A szabályban hivatkozott adatok *dimenziója* alapján** – Ha a szabályban előforduló elemek vagy attribútumok csak egydimenziósak, akkor **egydimenziós szabályról** beszélünk. A (6.1) szabály átírható a következő alakba:

$$\textit{vásárol}(X, \textit{“számítógép”}) \Rightarrow \textit{vásárol}(X, \textit{“pénzügyi\_szoftver”}) \quad (6.5)$$

  A (6.1) szabály egydimenziós társítási szabály, minthogy csak az egyetlen *vásárol* dimenziót használja.[4] Az olyan szabályokat, melyekben kettő vagy több dimenziót használunk, például *vásárlás*, *tranzakció_időpontja*, *vásárlói_kategória*, **többdimenziós társítási szabályoknak** nevezzük. A (6.4) szabály tehát többdimenziós társítási szabály, minthogy három dimenzióra: *életkor*, *jövedelem*, *vásárol* hivatkozik.
- **A szabályban használt *absztrakciós szintek* szerint** – A társítási szabályokat kereső eljárásokkal olyan társítási szabályokat találhatunk, amelyekben más-más absztrakciós szintekre hivatkoznak. Tegyük fel, hogy a szabályok között a következőket is megtaláltuk:

$$\textit{életkor}(X, \text{“30...39”}) \Rightarrow \textit{vásárol}(X, \textit{“laptop számítógép”}) \quad (6.6)$$

$$\textit{életkor}(X, \text{“30...39”}) \Rightarrow \textit{vásárol}(X, \textit{“számítógép”}) \quad (6.7)$$

  A (6.6) és (6.7) szabályokban különböző absztrakciós szintekre hivatkozunk. (A *“számítógép”* a *“laptop számítógép”* magasabb szintű absztrakciója.) Ilyen esetekben **többszintű társítási szabályhalmazzal** dolgozunk. Amennyiben a talált társítási szabályokban nem fordulnak elő különböző absztrakciós szintek, akkor **egyszintű társítási szabályról** beszélünk.
- **A társítási szabálykeresés *különböző kiterjesztései* szerint** – A társítási szabályok keresését korrelációanalízissé bővíthetjük. Ennek során korrelált elemek létét vagy nem létét azonosíthatjuk *maxminták* (azaz maximális gyakori minták) és *zárt gyakori*

4 | A többdimenziós adatbázisokban használatos terminológiát követve, a szabály minden különböző predikátumát *dimenziónak* tekintjük.

elemhalmazok keresésével is. A p **maxminta**, ha p bármely szupermintája[5] már nem gyakori. A c elemhalmaz **zárt gyakori elemhalmaz**, ha nincs a c-nek olyan valódi c' szuperhalmaza, melyre igaz lenne, hogy minden c-t tartalmazó tranzakció tartalmazza c'-t is. A maxminták és a zárt gyakori elemhalmazok alkalmazásával lényegesen csökkenthetjük a bányászat során előállított gyakori elemhalmazok számát.

A fejezet további részében valamennyi, a társítási szabályok fent leírt típusának keresési módszerét tanulmányozni fogjuk.

6.2. | Az egydimenziós Boole társítási szabályok bányászata tranzakciós adatbázisokban

Ebben a részben a legegyszerűbb formájú társítási szabályok bányászatának (keresésének) módszerét fogjuk megtanulni. Ezek a vásárlói kosár elemzése kapcsán, a 6.1.1. alfejezetben már bemutatotthoz hasonló, *egydimenziós, egyszintű Boole társítási szabályok*. A 6.2.1. alfejezetben a *gyakori elemhalmazok* keresésére alkalmas **Apriori alapalgoritmus** bemutatásával kezdünk. A gyakori elemhalmazokból az erős társítási szabályok előállítására alkalmas eljárást a 6.2.2. alfejezetben tárgyaljuk. A 6.2.3. alfejezetben az Apriori algoritmus több változatát mutatjuk be, amelyekkel az alapalgoritmus skálázhatósága és hatékonysága növelhető. A 6.2.4. alfejezetben bemutatott társításiszabály-kereső algoritmus eltér az Apriori algoritmustól, nem a gyakori elemhalmaz-„jelöltek" előállításával dolgozik. A 6.2.5. alfejezetben leírjuk, hogy az Apriori algoritmus alapelvei hogyan használhatók a *jéghegy típusú kérdések* megválaszolási hatékonyságának növelésére. A jéghegy típusú kérdés a vásárlói kosár elemzésében általánosan használt kérdéstípus.

6.2.1. | Az Apriori algoritmus: gyakori elemhalmazok keresése jelöltek előállításával

Az **Apriori algoritmus** hatékony módszer a Boole társítási szabályok felállításához szükséges gyakori elemek keresésére. Az algoritmus neve arra utal, hogy – amint a továbbiakban látni fogjuk – az algoritmus a gyakori elemhalmazok tulajdonságaira vonatkozó *előzetes (a priori) ismereteket* használ. Az Apriori algoritmus a *szintenkénti keresés* néven ismert iterációs módszert használja. A k-elemhalmazok felhasználásával állapítja meg a $(k + 1)$-elemhalmazokat. Először a gyakori 1-elemhalmazokat kell meghatározni. Ezt a halmazt L_1-el jelöljük. L_1-et használjuk L_2 megtalálásához (ez a gyakori 2-elemhalmazok halmaza), majd ennek segítségével keressük L_3-at és így tovább, amíg már nem találunk újabb gyakori k-elemhalmazokat. Minden L_k megtalálásához az adatbázis teljes átvizsgálása szükséges.

A gyakori elemhalmazok szintenkénti keresésének hatékonyságát növeli az **apriori tulajdonságnak** nevezett fontos tulajdonság. Az apriori tulajdonság alkalmazásával –

5 | Ha q tartalmazza p-t, akkor q szupermintája p-nek, p részmintája q-nak.

amint később látni fogjuk – tudjuk csökkenteni a keresési teret. Előbb megadjuk a tulajdonság leírását, majd egy példán bemutatjuk az alkalmazását.

Apriori tulajdonság: a gyakori elemhalmazok minden nem üres részhalmazának is gyakorinak kell lennie. Az apriori tulajdonság a következő megfigyelésen alapul: definíció szerint, ha egy I elemhalmaz nem elégíti ki a minimális megalapozottság (*min_sup*) kritériumot, akkor I nem gyakori, azaz $P(I) < min\_sup$. Ha az I elemhalmazt az A elemmel bővítjük, akkor a keletkezett elemhalmaz (vagyis $I \cup A$) nem lehet gyakoribb mint I. Tehát $I \cup A$ sem gyakori, azaz $P(I \cup A) < min\_sup$.

Ez a tulajdonság egy speciális tulajdonságkategóriához tartozik, amelyet **antimonoton** tulajdonságnak nevezünk. Az ilyen tulajdonság azt jelenti, hogy *ha egy halmazra nem teljesül valamely előírás, akkor annak összes szuperhalmazaira sem teljesül ugyanaz az előírás.* Az antimonoton elnevezést az előírás nem teljesítésének monotonsága indokolja.

Hogyan használható az apriori tulajdonság az algoritmusban? Ennek megértéséhez nézzük meg, hogyan használható L_{k-1} az L_k megtalálásához. A következő, kétlépéses eljárást adjuk meg:

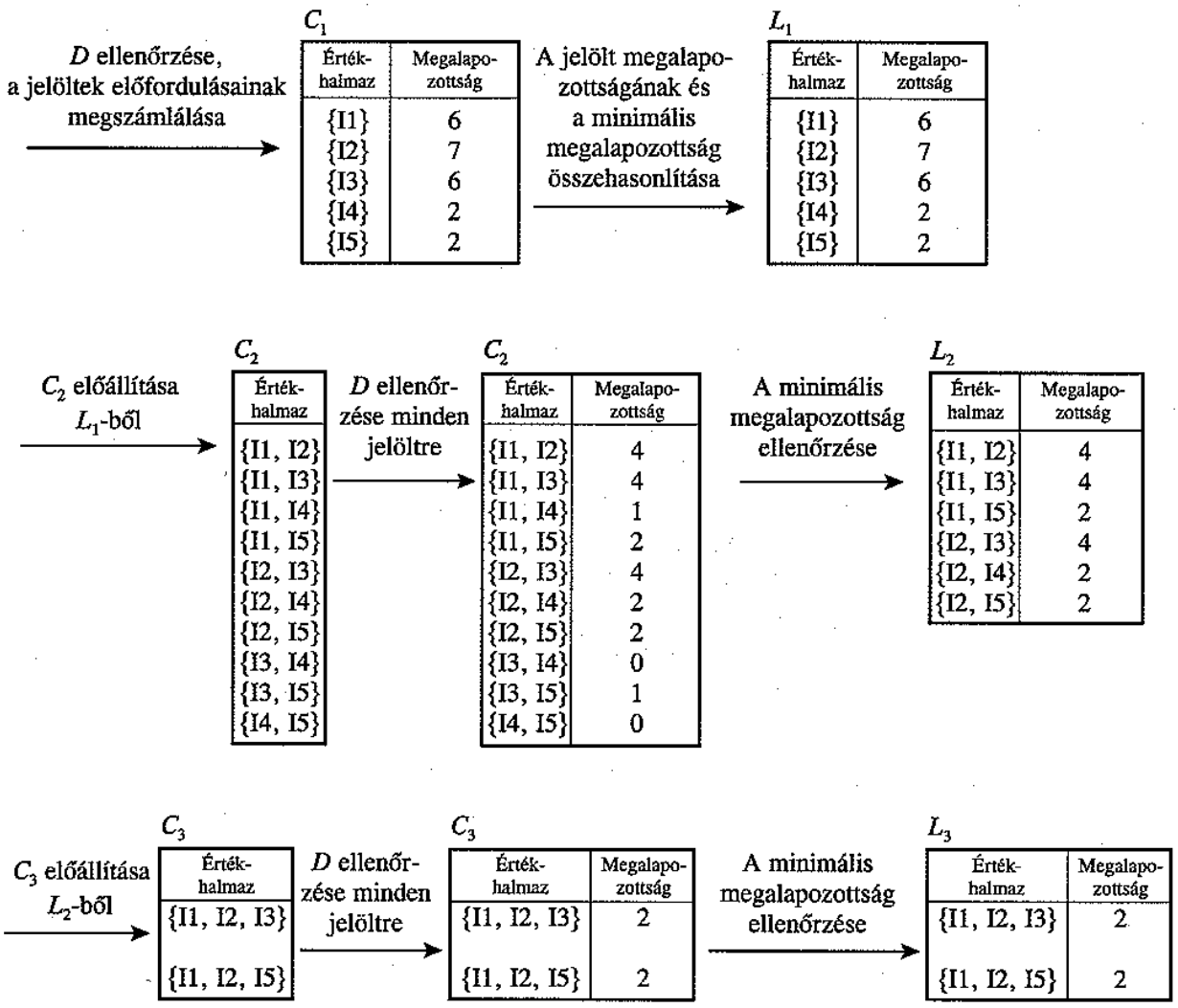
(1) **Összekapcsoló lépés** – L_k megtalálásához az L_{k-1} felhasználásával generáljuk a k-elemhalmaz **jelölteket**. A jelöltek ezen halmazát nevezzük el C_k-nak. Legyenek l_1 és l_2 az L_{k-1}-beli elemhalmazok. Jelöljük $l_i[j]$-vel l_i j-edik elemét (azaz például $l_1[k-2]$ az l_1-nek utolsó előtti elemét jelenti). Megegyezés szerint az Apriori algoritmus az elemeket a tranzakciókban és az elemhalmazokban lexikografikusan rendezettnek tekinti. Az $L_{k-1} \bowtie L_{k-1}$ összekapcsolás úgy értendő, hogy L_{k-1} azon elemei összekapcsolhatók, amelyek első $(k-2)$ elemei megegyezőek. Ez azt jelenti, hogy L_{k-1} két eleme, l_1 és l_2 akkor kapcsolhatók össze, ha $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. Az $l_1[k-1] < l_2[k-1]$ feltétellel egyszerűen biztosíthattuk, hogy duplikátumok ne generálódjanak. Az l_1 és l_2 összekapcsolása eredményeként létrejövő elemhalmaz: $l_1[1]l_1[2] \ldots l_1[k-1]l_2[k-1]$.

(2) **Elhagyás** – C_k halmaz szuperhalmaza L_k-nak, azaz C_k elemei vagy gyakoriak vagy nem, de az összes gyakori k-elemhalmaz előfordul C_k-ban. Az adatbázis átvizsgálásával minden C_k-beli jelöltről eldönthető, hogy gyakorinak tekintendő vagy sem, tehát hogy L_k-ba tartozik-e. (A definíció szerint minden jelölt gyakori, tehát L_k-ba tartozik, amelynek előfordulási száma nem kisebb, mint a minimális megalapozottság, *min_sup* értéke.) C_k igen nagy méretű lehet, feldolgozása nagy számítási teljesítményt igényel. C_k méretének csökkentésére az apriori tulajdonságot a következő módon használjuk: Minden $(k-1)$-elemhalmaz, mely nem gyakori, nem lehet részhalmaza gyakori k-elemhalmaznak. Így, ha valamely k-elemhalmaz jelölt $(k-1)$-részhalmaza nem L_{k-1}-beli, akkor a jelölt nem lehet gyakori sem, s így eltávolíthatjuk C_k-ból. Ha a gyakori elemhalmazokat tördelőfában (hash fában) kezeljük, akkor ezt a részhalmaztesztet gyorsan elvégezhetjük.

6.1. példa | Lássunk egy konkrét példát az Apriori algoritmus használatára. Használjuk az *ElektroMind* cég 6.2. ábrán látható D eladási tranzakciós adatbázisát. Az adatbázisban 9 tranzakció adatai találhatók meg, így $|D| = 9$. A 6.3. ábrát használjuk D-beli gyakori elemhalmazok Apriori algoritmussal való megkeresésének illusztrálására.

TID	Az elemek azonosítóinak listája
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

6.2. ábra | Az *ElektroMind* cégnél az eladási tranzakciók adatai



6.3. ábra | Jelölt elemhalmazok és gyakori elemhalmazok előállítása, a minimális megalapozottsági szint: 2

1. Összekapcsolás: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} = \{\{I1, I2, I3\}, \{I1,I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

2. Elhagyás: az Apriori tulajdonságra alapozottan, mely szerint a gyakori elemhalmaz minden nem üres részhalmazának is gyakorinak kell lennie, ellenőrizzük a tulajdonság fennállását.
   - az {I1, I2, I3} 2-elemű részhalmazai: {I1, I2}, {I1, I3}, {I2, I3}, ezek mind L_2 elemei, így {I1, I2, I3}-at megtartjuk C_3-ban.
   - az {I1, I2, I5} 2-elemű részhalmazai: {I1, I2}, {I1, I5}, {I2, I5}, ezek mind L_2 elemei, így {I1, I2, I5}-öt megtartjuk C_3-ban.
   - az {I1, I3, I5} 2-elemű részhalmazai: {I1, I3}, {I1, I5}, {I3, I5}. Az {I3, I5} nem eleme L_2-nek, tehát nem gyakori, így {I1, I3, I5}-öt eltávolítjuk C_3-ból.
   - az {I2, I3, I4} 2-elemű részhalmazai: {I2, I3}, {I2, I4}, {I3, I4}. Az {I3, I4} nem eleme L_2-nek, tehát nem gyakori, így {I2, I3, I4}-et eltávolítjuk C_3-ból.
   - az {I2, I3, I5} 2-elemű részhalmazai: {I2, I3}, {I2, I5}, {I3, I5}. Az {I3, I5} nem eleme L_2-nek, tehát nem gyakori, így {I2, I3, I5}-öt eltávolítjuk C_3-ból.
   - az {I2, I4, I5} 2-elemű részhalmazai: {I2, I4}, {I2, I5}, {I4, I5}. Az {I4, I5} nem eleme L_2-nek, tehát nem gyakori, így {I2, I4, I5}-öt eltávolítjuk C_3-ból.

3. Az elhagyások eredményeként $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$.

6.4. ábra | Az apriori tulajdonság alkalmazásával a 3-elemhalmaz jelöltek C_3 halmazának generálása L_2-ből

(1) Az algoritmus első iterációs lépésében minden elem a C_1 jelölt 1-elemhalmazok halmazának eleme. Az algoritmus egyszerűen végigvizsgálja az összes tranzakciót azért, hogy megszámolja az elemek előfordulásait.

(2) Tegyük fel, hogy a megkívánt minimális tranzakciós megalapozottsági szint 2 (azaz *min_sup* = 2/9 = 22%). A gyakori 1-elemhalmazok L_1 halmazát kell most meghatároznunk. Ez a minimális megalapozottsági szintet teljesítő jelölt 1-elemhalmazokból áll.

(3) A gyakori 2-elemhalmazok L_2 halmazának felfedezéséhez az algoritmus az $L_1 \bowtie L_1$ összekapcsolást használja a C_2, jelölt 2-elemhalmazok halmazának előállítására.[6] A C_2 halmaz $\left(\frac{|L_1|}{2}\right)$ darab 2-elemhalmazból áll.

(4) A következő lépésben a *D*-beli tranzakciókat kell végigvizsgálni, és meg kell állapítani a C_2-beli jelölt elemhalmazok megalapozottsági szintjét. Ez a 6.3. ábra második sorának középső táblájában látható.

(5) A gyakori 2-elemhalmazok L_2 halmazának meghatározása úgy történik, hogy a jelölt 2-elemhalmazok C_2 halmazának elemei közül azokat tartjuk meg, amelyek teljesítik a minimális megalapozottság követelményét.

(6) A jelölt 3-elemhalmazok C_3 halmazának előállítása a 6.4. ábrán követhető. Először legyen $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$. Az apriori tulajdonság alapján, mely szerint a gyakori elemhalmazok összes részhalmazának is gyakorinak kell lennie, megállapíthatjuk, hogy négy jelölt nem lehet gyakori. Ezért ezeket eltávolítjuk C_3-ból. Ezzel javítjuk *D* átvizsgálásának a hatékonyságát, amikor L_3 elemeit határozzuk meg. Megjegyezzük, hogy az Apriori algoritmus szin-

6 | Az $L_1 \bowtie L_1$ definíciója miatt megegyezik az $L_1 \times L_1$-el, ugyanis $L_k \bowtie L_k$ két, kapcsolható értékhalmazból keletkezik, melyekben $k - 1 = 0$ közös érték van.

Algoritmus: Apriori. Gyakori elemhalmazok keresése, jelöltek előállításán alapuló, szintenkénti iterációs módszerrel.

Bemenet: a tranzakciók D adatbázisa; *min_sup*, a minimális megalapozottság küszöbértéke.

Kimenet: a D-beli gyakori elemhalmazok L halmaza.

Módszer:

(1) L_1 = gyakori_1_elemhalmazok_megtalálása;
(2) **for** (k = 2; $L_{k-1} \neq \varnothing$; k + +) {
(3) C_k = **apriori_gen**(L_{k-1}, *min_sup*);
(4) **for each** tranzakció $t \in D$ { // D végigvizsgálása, az előfordulások összeszámlálása
(5) C_t = részhalmaz(C_k, t); // a t jelölt részhalmazai
(6) **for each** $c \in C_t$ jelöltre
(7) c.count + +;
(8) }
(9) $L_k = \{c \in C_k \mid c.count \geq min\_sup\}$
(10) }
(11) **return** $L = \cup_k L_k$;

procedure apriori_gen(L_{k-1}: gyakori (k – 1)-elemhalmazok; *min_sup*: a minimális megalapozottság küszöbértéke)
(1) **for each** $l_1 \in L_{k-1}$ elemhalmazra
(2) **for each** $l_2 \in L_{k-1}$ elemhalmazra
(3) **if** ($l_1[1] = l_2[1]$) $\wedge$ ($l_1[2] = l_2[2]$) $\wedge$ ... $\wedge$ ($l_1[k-2] = l_2[k-2]$) $\wedge$ ($l_1[k-1] < l_2[k-1]$) **then** {
(4) $c = l_1 \bowtie l_2$; // összekapcsoló lépés: a jelöltek generálása
(5) **if van_nem_gyakori_részhalmaza**(c, L_{k-1}) **then**
(6) **delete** c; // elhagyó lépés: a nem megfelelő jelöltek eltávolítása
(7) **else add** c **to** C_k ;
(8) }
(9) **return** C_k ;

procedure van_nem_gyakori_részhalmaza(c: jelölt k-elemhalmazok; L_{k-1}: gyakori (k – 1)-elemhalmazok);
// az előzetes ismeret felhasználása
(1) **for each** s c-beli (k – 1)-részhalmazra
(2) **if** $s \notin L_{k-1}$ **then**
(3) **return** TRUE;
(4) **return** FALSE;

6.5. ábra | Gyakori elemhalmazok keresése Boole társítási szabályok bányászatához, Apriori algoritmussal

tenkénti működésének következtében a jelölt k-elemhalmazok vizsgálatakor csak ezek (k – 1)-részhalmazait kell ellenőriznünk.

(7) A D-beli tranzakciókat átvizsgálva meghatározhatjuk L_3 halmazt. Ez a C_3 halmaz elemei közül azokból fog állni, amelyek teljesítik a minimális megalapozottság követelményét (6.3. ábra).

(8) A jelölt 4-elemhalmazok C_4 halmazának előállítása $L_3 \bowtie L_3$ képzésével történik. Minthogy az összekapcsolás eredménye {{*I*1, *I*2, *I*3, *I*5}}, s ezen halmaz egyetlen elemét is el kell hagynunk (mert {{*I*2, *I*3, *I*5}} részhalmaza nem gyakori), így $C_4 = \varnothing$. Ezzel az algoritmus befejeződik, megtaláltuk az összes gyakori elemhalmazt. ■

A 6.5. ábra az Apriori algoritmus, és a hozzá tartozó eljárások pszeudokódját mutatja be. Az Apriori algoritmus 1. lépésében előállítja L_1-et, a gyakori 1-elemhalmazokat. A 2–10. lépésekben L_{k-1} felhasználásával előállítja C_k-t, majd L_k-t. Az **apriori_gen** eljárás (3. lépés) generálja a jelölteket (az eljárás leírását lásd alább), majd az összes jelölt előállítása után átvizsgálja az adatbázist. Minden tranzakcióhoz a részhalmaz (**subset**) függvény felhasználásával megkeresi a tranzakcióban előforduló jelölt részhalmazokat (5. lépés), és gyűjti ezeknek a jelölteknek a számlálóit (6. és 7. lépés). Végül (a 9. lépésében) a megmaradt jelöltek közül azok alkotják az L gyakori elemhalmazok halmazát, amelyekre teljesül a minimális megalapozottság követelménye. Az eljárás a társítási szabályok gyakori elemhalmazokból való előállítására alkalmas. Ezt az eljárást a 6.2.2. alfejezetben jellemezzük.

Az **apriori_gen** eljárás két művelettípust hajt végre, nevezetesen az összekapcsolás (join) és az elhagyás (prune) korábban már ismertetett műveleteket. Az összekapcsoló komponens (1–4. lépések) L_{k-1}-et L_{k-1}-gyel összekapcsolja, generálva ezzel a lehetséges jelölteket. Az elhagyó rész (5–7. lépések) használja az apriori tulajdonságot a jelöltek közül azok eltávolítására, melyek részhalmazai nem gyakori elemhalmazok. A részhalmaz nem gyakoriságának ellenőrzése a **van_nem_gyakori_részhalmaza** eljárásban látható.

6.2.2. | Társítási szabályok generálása a gyakori elemhalmazokból

Ha a D tranzakciós adatbázisból előállítottuk a gyakori elemhalmazokat, akkor ezekből egyenesen generálhatók az erős társítási szabályok (ahol az *erős* társítási szabály kielégíti mind a minimális megalapozottság, mind a minimális megbízhatóság követelményeket). Ehhez a következő, a megbízhatóságra vonatkozó egyenlőséget használjuk, ahol a feltételes valószínűséget az elemhalmazbeli előfordulások számával fejezzük ki:

$$megbízhatóság(A \Rightarrow B) = P(B|A) = \frac{előfordulások\_száma(A \cup B)}{előfordulások\_száma(A)}, \tag{6.8}$$

ahol *előfordulások_száma*($A \cup B$) azoknak a tranzakcióknak a száma, amelyek tartalmazzák az $A \cup B$ elemhalmazt, az *előfordulások_száma*(A) pedig azoknak a tranzakcióknak a száma, amelyek tartalmazzák az A elemhalmazt. Erre az egyenletre alapulva a társítási szabályok a következőképpen állíthatók elő:

- Minden l gyakori elemhalmazhoz generáljuk az összes nem üres részhalmazát.
- Ha l egy valódi, nem üres s részhalmazára teljesül, hogy

  $$\frac{előfordulások\_száma(1)}{előfordulások\_száma(s)} \geq min\_conf,$$

  ahol *min_conf* a minimális megbízhatósági küszöbérték, akkor előállítjuk az $s \Rightarrow (l - s)$ szabályt.

Mivel a szabályokat a gyakori elemhalmazokból állítjuk elő, így mindegyik automatikusan kielégíti a minimális megalapozottsági kritériumot. Ha a gyakori elemhalmazokat az előfordulási számukkal együtt tördelőtáblában tároljuk, biztosítható ezeknek az adatoknak a gyors elérése.

6.2. példa | Lássunk egy példát az *ElektroMind* cég 6.2. ábrán látható tranzakciós adatbázisára alapozva. Tegyük fel, hogy az $l = \{I1, I2, I5\}$ gyakori elemhalmaz. Melyek az l-ből előállítható társítási szabályok? Az l valódi nem üres részhalmazai a következők: $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$ és $\{I5\}$. A fentiek szerint előállítható társítási szabályok és megbízhatóságuk:

$I1 \wedge I2 \Rightarrow I5$	*megbízhatóság* = 2/4 = 50%
$I1 \wedge I5 \Rightarrow I2$	*megbízhatóság* = 2/2 = 100%
$I2 \wedge I5 \Rightarrow I1$	*megbízhatóság* = 2/2 = 100%
$I1 \Rightarrow I2 \wedge I5$	*megbízhatóság* = 2/6 = 33%
$I2 \Rightarrow I1 \wedge I5$	*megbízhatóság* = 2/7 = 29%
$I5 \Rightarrow I1 \wedge I2$	*megbízhatóság* = 2/2 = 100%

Ha a minimális megbízhatósági szintet mondjuk 70%-ban állapítjuk meg, akkor csak a fenti második, harmadik és utolsó szabályokat tekintjük erős társítási szabályoknak. ■

6.2.3. | Az Apriori algoritmus hatékonyságának növelése

Hogyan növelhető az Apriori algoritmus hatékonysága? Az Apriori algoritmus eredeti változata hatékonyságának növelésére több módszert is kidolgoztak. Az alábbiakban felsorolunk ezek közül néhányat.

- **Tördelőtechnika** – A tördelőtechnikát arra használjuk, hogy a jelölt k-elemhalmazok C_k (ha $k > 1$) halmazainak méretét csökkentsük. Például, miközben az adatbázis minden tranzakcióját végigvizsgáljuk azért, hogy L_1-et, a gyakori 1-értékhalmazokat előállítsuk C_1 jelölt elemhalmazokból, generálhatjuk az összes tranzakció 2-elemhalmazait. Ezeket szétosztjuk a *tördelőtábla* különböző *kosaraiba,* s egyúttal megnöveljük az adott kosár elemszámlálóját (6.6. ábra). Azok a 2-elemhalmazok, ahol az azokat tartalmazó kosarak számlálója a tördelőtáblában a küszöbérték alatt marad, nem tekintendők gyakorinak, és így a jelöltek elemhalmazából eltávolítandók. A tördelőtechnikával lényegesen csökkenthetjük a megvizsgálandó jelölt k-elemhalmazok számát (különösen ha $k = 2$).
- **Tranzakciók redukálása (a későbbi iterációs lépések során megvizsgálandó tranzakciók számának redukálása)** – Az olyan tranzakció, amely nem tartalmaz egyetlen gyakori k-elemhalmazt, az nem tartalmazhat gyakori $k + 1$ elemhalmazt sem. Így az ilyen tranzakciókat eltávolíthatjuk vagy megjelölhetjük, mert későbbi vizsgálatukra (amikor j-elemhalmazokat keresünk, és $j > k$) már nincs szükség.
- **Particionálás (az adatok felosztása jelölt elemhalmazok megkereséséhez)** – A felosztásos módszerrel az adatbázist csak kétszer szükséges átvizsgálnunk a gyakori elemhalmazok megkeresése céljából (6.7. ábra). A módszer két fázisból áll. Az első fázisban az algoritmus D tranzakcióit n átfedés nélküli partícióra osztja. Ha a tranzakciókra a minimális megalapozottsági küszöb *min_sup,* akkor a partíciók minimális megalapozottsági számlálója: *min_sup* × *a partícióbeli tranzakciók száma* formulával adódik. Minden partícióban minden gyakori elemhalmazt megkeresünk, ezeket **lokálisan gyakori elemhalmazoknak** nevezzük. Az eljárás speciális adatstruktúrát használ, minden elemhalmazhoz tárolja azon tranzakciók TID-jeit, amelyek az elemhalmazhoz tartozó

A H_2 tördelőtáblát megadó tördelőfüggvény:
$h(x, y) = ((x \text{ értéke}) \times 10 + (y \text{ értéke})) \bmod 7$

H_2

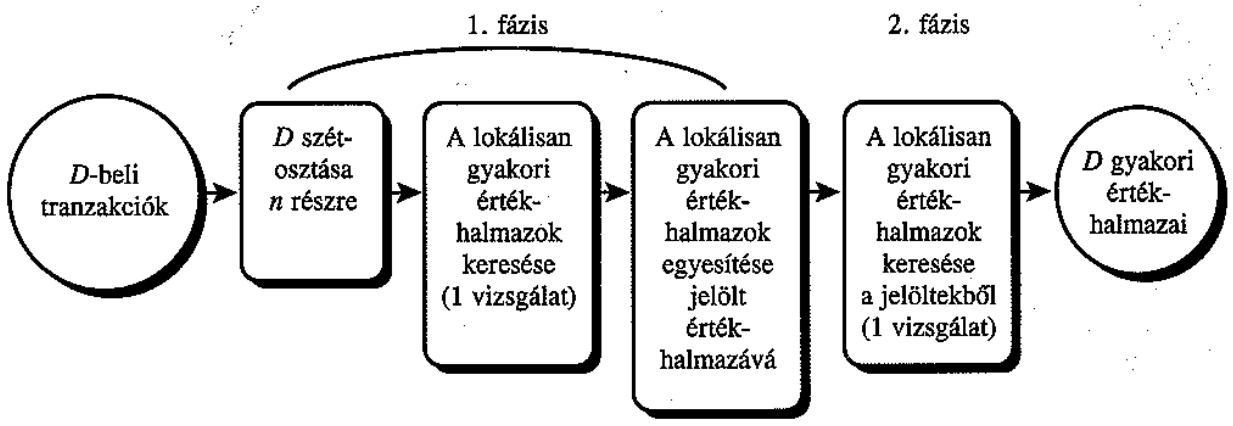
kosár címe	0	1	2	3	4	5	6
kosarak száma	2	2	4	2	2	4	4
kosár tartalma	{I1,I4}	{I1,I5}	{I2,I3}	{I2,I4}	{I2,I5}	{I1,I2}	{I1,I3}
	{I3,I5}	{I1,I5}	{I2,I3}	{I2,I4}	{I2,I5}	{I1,I2}	{I1,I3}
			{I2,I3}			{I1,I2}	{I1,I3}
			{I2,I3}			{I1,I2}	{I1,I3}

6.6. ábra | A jelölt 2-elemhalmazok H_2 tördelőtáblája: ezt a táblát a 6.2. ábra tranzakcióinak vizsgálatával állítottuk elő, L_1-nek C_1-ből történő meghatározása közben. Ha a minimális megalapozottsági szint mondjuk 3, akkor a 0, 1, 3 és 4 kosarakban található elemhalmazokat nem tekintjük gyakorinak, s így ezeket nem adjuk C_2-höz

elemeket tartalmazzák. Ez a módszer az adatbázis egyszeri átvizsgálásával lehetővé teszi az összes lokálisan gyakori k-elemhalmazok ($k = 1, 2, \ldots$) megtalálását.

A lokálisan gyakori elemhalmaz az egész D adatbázisra nézve vagy gyakori, vagy nem. *Minden elemhalmaznak, mely D-re nézve potenciálisan gyakori, elő kell fordulnia legalább egy partícióban mint lokálisan gyakori elemhalmaz.* Így minden lokálisan gyakori elemhalmazt a D-re vonatkozóan jelölt elemhalmaznak tekinthetünk. Az összes partíciók lokálisan gyakori elemhalmazai egyesítése a D-re vonatkozó **globális jelölt elemhalmazt** adja. A második fázisban D második átvizsgálásával tudjuk meghatározni a jelöltek közül a globálisan gyakori elemhalmazokat. A partíciók méretét és számát a rendelkezésre álló memória méretéhez igazítva elérhetjük, hogy bármely partíció beférjen a memóriába, és így minden fázisban minden partíciót csak egyszer kell a háttértárolóról beolvasni.

- **Minta használata (az adatok egy részhalmazán való bányászás)** – A mintavételes megközelítés alapgondolata szerint először az adott D adatokból egy S mintát választunk ki véletlenszerűen, s ezután már csak az S-beli elemek körében keressük a gyakori elemhalmazokat. A D helyett az S-ben keressük a gyakori elemhalmazokat. Ezen a módon a hatékonyságért valamennyit feláldozunk a pontosságból. Az S minta méretét úgy választjuk meg, hogy az S-beli gyakori elemhalmazok keresése a memóriában



6.7. ábra | Bányászás az adatok szétosztásával

végrehajtható legyen. Így az S tranzakcióinak átvizsgálására (háttértárolóról való beolvasására) mindössze egyszer lesz szükség. Minthogy a gyakori elemhalmazokat D helyett S-ben keressük, így előfordulhat, hogy elveszítünk (nem ismerünk fel) néhány globálisan gyakori elemhalmazt. Azért, hogy ennek lehetőségét csökkentsük, az eredetinél alacsonyabb megalapozottsági küszöbértéket használunk az S-beli gyakori elemhalmazok keresésekor. Az S-beli gyakori elemhalmazokat L^S-el jelöljük. Az adatbázis (mintán kívüli) többi részét az L^S-beli elemhalmazok tényleges gyakorisága megállapítására használjuk. E módszer használatos annak eldöntésére, hogy az összes globálisan gyakori elemhalmazok L^S-be kerültek-e. Ha L^S tartalmazza D összes gyakori elemhalmazát, akkor elegendő D egyszeri átvizsgálása. Ha L^S nem tartalmazza D összes gyakori elemhalmazát, akkor D második átvizsgálásakor megtaláljuk azokat a gyakori elemhalmazokat, amelyeket az első menetben nem ismertünk fel. A mintát használó megközelítés különösen olyan esetekben előnyös, amikor a hatékonyság különösen fontos, például ha nagy számításigényű alkalmazásokat kell nagyon intenzíven használt adatbázison működtetni.

- **Elemhalmaz dinamikus kialakítása (a jelölt elemhalmazok a keresési folyamat különböző pontjain kerülnek az eredmény halmazba)** – Az elemhalmaz dinamikus kialakításának technikája olyan esetekben ajánlott, amikor az adatbázis blokkokra van osztva, és minden blokknak saját kezdési pontja van. Ebben a változatban az új jelölt elemhalmazok bármelyik kezdési pontnál hozzáadhatók. Ez eltér az Apriori algoritmustól, amelyben az új jelölt elemhalmazokat közvetlenül az adatbázis teljes átvizsgálása előtt határozza meg. A módszer dinamikus, az elemhalmazok megalapozottságát számolja mindaddig, amíg új jelölt elemhalmazokkal bővíti a jelöltek körét. A bővítésre akkor kerülhet sor, ha a vizsgált jelölt elemhalmaz összes részhalmazait gyakorinak találta. Ez a módszer az Apriori algoritmusnál kevesebb adatbázis-átvizsgálást igényel.

A fejezet hátralevő részében további változatokat tárgyalunk, amelyekben megjelenik a többszintű és többdimenziós társítási szabályok bányászata. A térbeli adatokban, idősoros adatokban és multimédia-adatokban történő, társítási szabályokat kereső bányászatot a 9. fejezetben tárgyaljuk.

6.2.4. | Gyakori elemhalmazok keresése jelöltek generálása nélkül

Mint láttuk, sok esetben az Apriori algoritmus jelölt generálás-ellenőrzés módszere a jelöltek számát erőteljesen csökkenti, ezzel növeli a hatékonyságot. Mindazonáltal jelentkezhet két nem triviális költség.

- **Szükség lehet óriási mennyiségű jelölthalmaz generálására** – Ha a gyakori 1-elemhalmazok száma 10^4, akkor az Apriori algoritmus 10^7-nél több jelölt 2-elemhalmazt fog előállítani, összegyűjteni és meg fogja állapítani előfordulási gyakoriságukat. Túl ezen a 100 méretű, azaz $\{a_1, \ldots, a_{100}\}$ típusú gyakori minták felfedezéséhez **összességében** több mint $2^{100} \approx 10^{30}$ darab jelöltet kell előállítani.
- **Szükségessé válhat az adatbázis többször ismételt átvizsgálása és nagy jelölthalmazok mintaillesztése** – Különösen így van ez hosszú minták bányászata esetében.

Tudunk-e olyan algoritmust tervezni, amely megtalálja a gyakori elemhalmazok teljes halmazát, jelöltek generálása nélkül? Az erre kísérletet tevő érdekes módszer a **gyakori minta bővítése** vagy röviden **FP-bővítés** nevű eljárás (**FP, frequent pattern, gyakori minta**). Az FP-bővítés az *„oszd meg és uralkodj" stratégiát* a következőképpen alkalmazza: tömörítsük az adatbázist a gyakori elemek **FP-fában** (**frequent pattern tree, gyakori minták fája**) való reprezentálásával, de tartsuk vissza az elemhalmaz-társítási információkat. Ezután a tömörített adatbázist osszuk szét *feltétel szerint szétosztott adatbázisok* halmazába (ez a vetítéssel létrejövő adatbázisok speciális fajtája). A feltétel szerint szétosztott adatbázis minden komponenséhez egy-egy gyakori elem tartozik. Minden ilyen részadatbázisban külön-külön bányászunk. Lássunk erre egy példát!

6.3. példa | Vizsgáljuk meg újból a 6.1. példa 6.2. ábráján látható D tranzakciós adatbázist. Használjuk most az FP-bővítést.

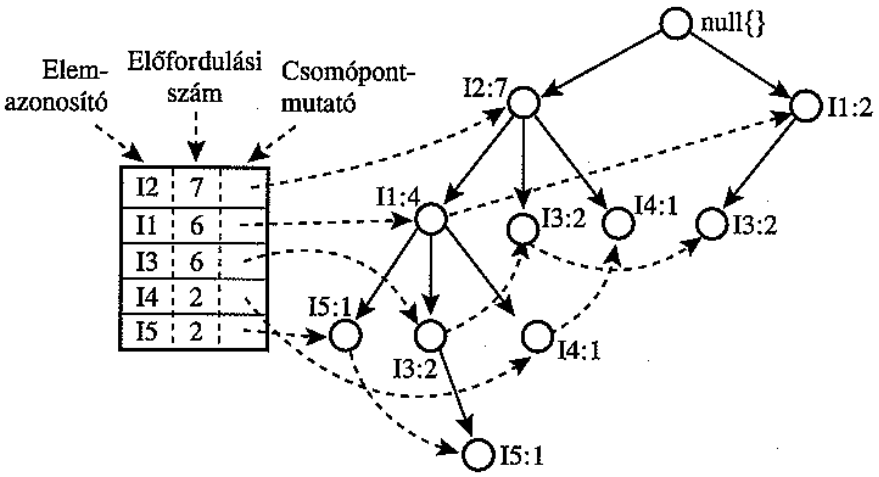
Először vizsgáljuk át az adatbázist ugyanúgy, mint ahogy az Apriori algoritmusban. Ezzel meghatározzuk a gyakori 1-elemhalmazokat és előfordulási számukat (gyakoriságukat). Legyen a minimális megalapozottság küszöbértéke 2. A gyakori elemeket rendezzük sorba a megalapozottságuk szerint csökkenő sorrendben. A kapott lista legyen L. A példában $L = [I2: 7, I1: 6, I3: 6, I4: 2, I5: 2]$.

Az FP-fát a következőképpen konstruáljuk meg: először létrehozzuk a fa gyökerét, megcímkézzük „null" címkével. Másodszor vizsgáljuk végig a D adatbázist. A tranzakciókban előforduló elemeket az L-beli sorrendben (tehát a megalapozottságuk szerint csökkenő sorrendbe rendezve) dolgozzuk fel. Minden tranzakcióhoz egy faágat hozunk létre. Például az első tranzakció (T100: $I1$, $I2$, $I5$) feldolgozásakor, látjuk, hogy három elemet tartalmaz, ezek az L-beli sorrendnek megfelelően: ($I2$, $I1$, $I5$). Létrehozzuk a fa első ágát: három csomóponttal, $\langle I2: 1\rangle$, $(I1: 1)$, $(I5: 1)\rangle$, ahol $I2$ csomópont a gyökérhez, $I1$ csomópont $I2$-höz, $I5$ csomópont az $I1$-hez kapcsolódik. A T200, második tranzakció az L-beli sorrendnek megfelelően $I2$ és $I4$ elemeket tartalmazza. Ennek feldolgozása az $\langle I2, I4\rangle$ ágat hozza létre, ahol $I2$ a gyökérhez, $I4$ pedig $I2$-höz kapcsolódik. Minthogy ezen ág előtagja $\langle I2\rangle$ közös a már T100 miatt létező ággal, ezért az új ág létrehozása helyett 1-gyel megnöveljük $I2$ számlálóját, és létrehozzuk az $(I4: 1)$ csomópontot, és gyerekként $(I2: 2)$ csomóponthoz kapcsoljuk. Általánosságban, ha egy tranzakció miatt létrehozandó ágnak van a már létezők valamelyikével közös előtagja, akkor a közös előtaghoz tartozó csomópontok számlálóját 1-gyel megnöveljük, az előtagot követő csomópontokat pedig létrehozzuk és L-beli sorrendnek megfelelően összekapcsoljuk.

A fa bejárását elősegítendő címketáblát építünk fel, ahol minden elem fabeli elhelyezkedését a hozzá vezető **csomópontlánc** adja meg. Az összes tranzakció feldolgozása után keletkező fát és a hozzá tartozó csomópont-kapcsolatokat a 6.8. ábra mutatja. Ezzel az adatbázisbeli gyakori minták keresése feladatot az FP-fa bányászatára transzformáltuk.

Az FP-fa bányászása a következő eljárással történik: A gyakori 1-hosszú mintákkal (mint elsődleges **utótagmintákkal**) kezdjük. Létrehozzuk a nekik megfelelő **feltételes mintabázisokat** (a részadatbázisokat, melyek az FP-fában utótagmintákkal együtt előforduló **előtag utakból** állnak), majd megkonstruáljuk ezeknek a részadatbázisoknak az FP-fáit. Ezeken a fákon a fenti eljárást rekurzívan folytatjuk. Az FP-bővítés az utótagminták és a részadatbázisok FP-fáiból generált gyakori minták egyesítésével (konkatenációjával) valósul meg.

Az FP-fában való bányászást a 6.1. táblázatban foglaltuk össze. Ennek elemzése: az L-beli első elem helyett először vizsgáljuk meg L utolsó elemét, $I5$-öt. $I5$ a 6.8. ábrán látható FP-fa



6.8. ábra | A gyakori minta információk tömörítése FP-fában

két ágán fordul elő. (*I5* előfordulásai a csomópont-kapcsolatokat megadó táblából kiolvasható láncot követve könnyen megtalálhatók.) Az ezekből az ágakból kialakuló utak ⟨(*I2 I1 I5*:1)⟩ és ⟨(*I2 I1 I3 I5*:1)⟩. Így *I5* utótagnak tekintendő, hozzá két előtag út tartozik, ezek: ⟨(*I2 I1*:1)⟩ és ⟨(*I2 I1 I3*:1)⟩, s ezekhez feltételes mintabázist hozunk létre. Az ehhez tartozó FP-fa egyetlen utat tartalmaz, ez ⟨*I2*: 2, *I1*: 2⟩. *I3* nincs benne, mert megalapozottsága 1, mely nem éri el az előírt küszöbértéket (példánkban ez 2). Ez az egyetlen út generálja a gyakori minták összes kombinációját, ezek: *I2 I5*: 2, *I1 I5*: 2, *I2 I1 I5*: 2.

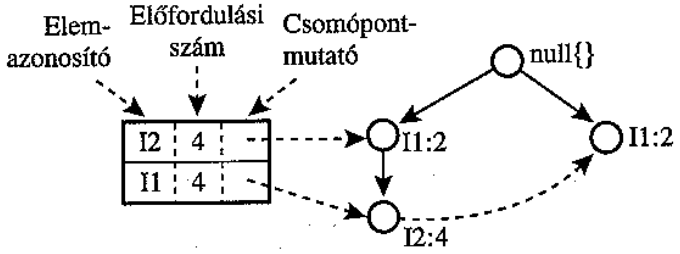
6.1. táblázat | FP-fa bányászása részminták létrehozásával

Elem	Feltételes mintabázis	FP-fa	Generált gyakori minták
I5	{(I2 I1: 1), (I2 I1 I3: 1)}	⟨I2: 2, I1: 2⟩	I2 I5: 2, I1 I5: 2, I2 I1 I5:2
I4	{(I2 I1: 1), (I2: 1)}	⟨I2: 2⟩	I2 I4: 2
I3	{(I2 I1: 2), (I2: 2), (I1: 2)}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	I2 I3: 4, I1 I3: 4, I2 I1 I3: 2
I1	{(I2: 4)}	⟨I2: 4⟩	I2 I1: 4

Az *I4* elemre két előtagútból, az {(*I2 I1*:1), (*I2*:1)} utakból keletkezik feltételes mintabázis. Az ehhez tartozó FP-fa egyetlen csomópontból, ⟨*I2*:2⟩ -ből áll, és egy gyakori mintát lehet belőle megállapítani, ez: *I2 I4*: 2. Megjegyzendő, hogy bár *I5* csomópont *I4*-et követi a fa első ágában, de *I5*-öt már nem kell az elemzésekkor figyelembe venni, mert minden *I5*-öt tartalmazó gyakori mintát már *I5* elemzésekor feldolgoztunk. Ez az oka annak, hogy a feldolgozást *L* utolsó elemével kezdtük, és nem az elsővel.

A fenti elemzéshez hasonlóan az *I3*-hoz tartozó feltételes mintabázis {(*I2 I1*: 2), (*I2*: 2), (*I1*: 2)}. A hozzá tartozó FP-fa kétágú, az ágai: ⟨*I2*: 4, *I1*: 2⟩ és ⟨*I1*: 2⟩, amint a 6.9. ábrán látható, ezek a minták következő halmazát generálják: {*I2 I3*: 4, *I1 I3*: 4, *I2 I1 I3*: 2}. Végül az {(*I2*: 4)} feltételes mintabázis FP-fája az ⟨*I2*: 4⟩ egyetlen csúcsot tartalmazza, ami az egyetlen *I2 I1*: 4 gyakori mintát generálja. Ezt a bányász eljárást a 6.10. ábrán foglaltuk össze. ■

Az FP-bővítés módszere a hosszú gyakori minták keresését rövidebbek rekurzív keresésévé, és végül az utótagok egyesítésévé transzformálja. A legkevésbé gyakori elem utótagként való használata jó szelektivitást biztosít. A módszer lényegesen redukálja a keresés költségét.



6.9. ábra | Az I3 csomóponthoz tartozó FP-fa

Ha az adatbázis nagy, gyakran nincs reális lehetőség a memóriában elférő FP-fa generálására. Érdekes alternatíva előbb részekre bontani az adatbázist, majd minden részre megkonstruálni az FP-fát, és részenként végrehajtani a bányászatot. Ezt rekurzívan ismételhetjük mindaddig, amíg az FP-fa már elfér a memóriában.

Az FP-bővítés hatékonyságának tanulmányozása azt mutatja, hogy hatékony és skálázható mind hosszú, mind rövid gyakori minták keresésében és nagyságrenddel gyorsabb, mint az Apriori algoritmus. Gyorsabb a fa-vetítés algoritmusánál is, amely az adatbázist rekurzívan vetíti a vetített adatbázisok fájába.

Algoritmus: FP_bővítés. Gyakori elemhalmazok keresése, FP-fa használatával.

Bemenet: A tranzakciók D adatbázisa; *min_sup*, a minimális megalapozottság küszöbértéke.

Kimenet: A gyakori minták teljes halmaza.

Módszer:

1. FP-fa generálása a következő lépésekben:
   (a) A tranzakciók D adatbázisának elemzése. A gyakori elemek F halmazának összegyűjtése. F csökkenő előfordulási számok szerint sorba rendezésével L, a gyakori elemek listájának előállítása.
   (b) Az FP-fa gyökerének létrehozása és „null"-ként való megcímkézése. Minden D-beli *Trans* tranzakcióra tegyük a következőket:
   Válogassuk ki és rendezzük az L-beli sorrendnek megfelelően a *Trans* tranzakcióban található gyakori elemeket. Legyen a *Trans* rendezett gyakori elemeinek listája $[p|P]$, ahol p az első elem, P a lista többi része. Hívjuk meg a **fa_beszúrás**($[p|P]$, T) eljárást, mely a következőket teszi: Ha T gyereke N és $N.elem\text{-}név = p.elem\text{-}név$, akkor növeljük meg N számlálóját 1-gyel; különben hozzunk létre új N csomópontot, állítsuk számlálóját 1-re, s gyerekként kapcsoljuk T-hez, végül láncoljuk be az azonos nevű csomópontláncra. Ha P nem üres, akkor hívjuk rekurzívan a **fa_beszúrás**(P, N) eljárást.

2. Az **FP_bővítés**(*FP_fa*, *null*) eljárás meghívásával bányászás az FP-fán. Az **FP_bővítés** eljárás a következő:

```
procedure FP_bővítés(fa, α)
(1) if fa az egyszerű P utat tartalmazza then
(2)     for each a P út csomópontjai (összes) β kombinációira
(3)         β ∪ α mintái generálása a megalapozottság = β-beli csomópontok megalapozottsága
              minimuma figyelembe vételével;
(4) else for each a_i-re, ahol a_i a fa fejrésze {
(5)     β = a_i ∪ α mintái generálása megalapozottság = a_i. megalapozottság figyelembevételével;
(6)     β feltételes mintabázisának és FP_fájának (fa_β) a megkonstruálása;
(7)     if faβ ≠ ∅ then
(8)         call FP_bővítés(fa_β, β);}
```

6.10. ábra | Az FP-bővítés algoritmusa; gyakori elemhalmazok felfedezése jelöltek generálása nélkül

6.2.5. | Jéghegy típusú kérdések

Az Apriori algoritmust arra is használhatjuk, hogy a *jéghegy típusú kérdések* megválaszolásának hatékonyságát növeljük. A jéghegy-kérdések az adatbányászatban általánosan használatosak, különösen a vásárlói kosár elemzésben. A **jéghegy-kérdés** attribútum vagy attribútumok halmaza felett összesített függvényt számít ki, valamilyen küszöbértéket meghaladó összesített értéket keresve. Legyen adott az R reláció, attribútumai a_1, a_2, ... a_n és b, valamint egy összesítőfüggvény, agg_f. A jéghegy-kérdés a következő formájú:

```
select      R.a_1, R.a_2,...,R.a_n, agg_f(R.b)
from        relation R
group by    R.a_1, R.a_2,...,R.a_n
having      agg_f(R.b) > = küszöbérték
```

Nagy számú input adatsor esetén is előfordulhat, hogy viszonylag kevés olyan sor lesz, amely a **having** záradék küszöbfeltételét kielégíti. A lekérdezés eredménye olyannak látszik, mint a jéghegy csúcsa, ahol a „jéghegy" az input adatok halmaza.

6.4. példa | Egy jéghegy típusú kérdés: tegyük fel, hogy adottak vásárlások adatai. Szeretnénk előállítani azon megvásárolt árucikkek és vásárlók listáját, ahol a vásárló az adott árucikkből három vagy több darabot vásárolt. Ez a következő jéghegy-kérdéssel írható le:

```
select      P.vásárló_ID, P.árucikk_ID, SUM(P.mennyiség)
from        Vásárlás P
group by    P.vásárló_ID, P.árucikk_ID
having      SUM(P.mennyiség) > = 3   ■
```

Feltehetjük a kérdést: *Hogyan válaszolhatjuk meg a 6.4. példa kérdését?* Az általános módszer szerint előbb rendezni kell, majd csoportonként ki kell számítani az összesítőfüggvényt, és végül a vizsgált feltétel szerint el kell távolítani az eredményből a nem kívánt részeket (itt azokat a sorokat, amelyekre a vásárolt mennyiség háromnál kisebb). A feltételt kielégítő sorok száma gyakran sokkal kisebb, mint a feldolgozott sorok száma, ez teret ad a hatékonyság növelésére. Másik lehetőségként használhatjuk az apriori tulajdonságot a vásárló–árucikk párok ritkítási lehetőségének megvizsgálására. Azaz ahelyett, hogy minden, a vásárló által vásárolt árucikk mennyiségét megnéznénk, a következőt tesszük:

- Előállítjuk azoknak a *vevőknek a listáját*, akik összességében három vagy annál több árucikket vásároltak, például:

```
select      P.vásárló_ID
from        Vásárlás P
group by    P.vásárló_ID
having      SUM(P.mennyiség) > = 3
```

- Előállítjuk azoknak az *árucikkeknek a listáját*, amelyekből összességében három vagy annál több darabot vásároltak, például:

```
select      P.árucikk_ID
from        Vásárlás P
group by    P.árucikk_ID
having      SUM(P.mennyiség) > = 3
```

Ezzel az előzetes ismerettel a kérdés szempontjából biztosan érdektelen vásárlókat és árucikkeket előzetesen ki tudjuk hagyni, és csak azoknak a jelölteknek a további vizsgálatával kell foglalkoznunk, amelyeknek esélye van a válaszba kerülni. Ez a módszer növeli a hatékonyságot sok páros vagy csoport előzetes ritkításával. A még vizsgálandó párok száma még így is lehet olyan nagy, hogy azok egyszerre nem férnek el a memóriában. A tördelőtechnikák és a mintákból dolgozó módszerek is beépíthetők a folyamatba, segítve az általános hatékonyság növelését.

6.3. | Többszintű társítási szabályok bányászata tranzakciós adatbázisokban

Ebben a fejezetben megismerkedünk a többszintű társítási szabályok keresésére (bányászatára) vonatkozó módszerekkel. A többszintű társítási szabályok az absztrakció különböző szintjein lévő adatelemeket tartalmaznak. A felesleges (redundáns) többszintű szabályok kiszűrésére vonatkozó módszerekkel is foglalkozunk.

6.3.1. | Többszintű társítási szabályok

A többdimenziós térbeli adatok térbeli ritkaságának (szétszórtságának) köszönhetően, az absztrakció alacsony vagy primitív szintjein számos alkalmazás esetén nehezen található erős társítás az adatelemek között. Magasabb fogalmi szinten felfedezett erős társítás általánosabb érvényű ismeretet fogalmazhat meg. Ami általános ismeretet jelent az egyik felhasználó számára, mindaz a másiknak használhatatlannak tűnhet. Ezért adódik az elvárás, hogy az adatbányászati rendszerek kínáljanak lehetőséget többszintű társítási szabályok keresésére, valamint a különböző absztrakciós terek közötti egyszerű átjárásra.

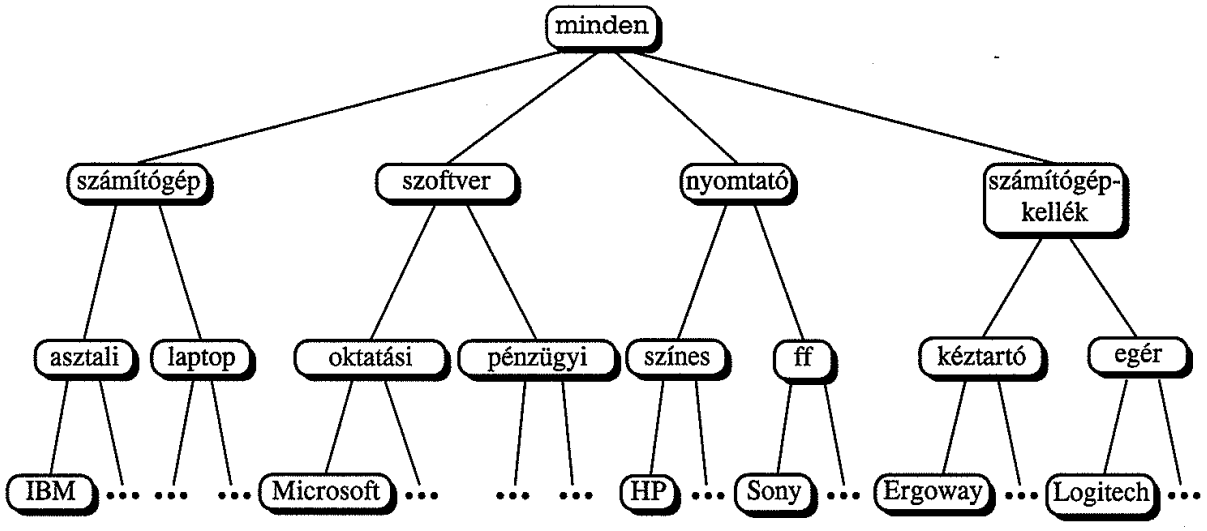
Vizsgáljuk meg a következő példát!

6.5. példa | Tegyük fel, hogy adott az *ElektroMind* cég számítástechnikai részlegének eladási forgalmára vonatkozó, a 6.2. táblázatban látható, fontosabb adatokat tartalmazó tranzakciós adathalmaz. Ez tranzakciónként a TID-hez tartozó eladott áruk megnevezését adja meg. Az áruk fogalomhierarchiáját a 6.11. ábrán mutatjuk be. A fogalomhierarchia az alacsonyabb és a magasabb, általánosabb szintű fogalmak egymáshoz való viszonyát adja meg. Az adat oly módon általánosítható, hogy az alacsonyabb szintű fogalmat a fogalomhie-

rarchiában[7] felette álló valamely magasabb szintűvel helyettesítjük. A 6.11. ábrán bemutatott fogalomhierarchia 4 szintű, a szintekre 0., 1., 2. és 3-ként hivatkozunk. Megállapodás szerint a szinteket felülről lefelé számozzuk, a **minden** (a legáltalánosabb absztrakciós szintet jelentő) gyökérnél a 0. szinttel kezdve. Az 1. szintre a *számítógép, szoftver, nyomtató, számítógépkellék* fogalmak kerültek. A 2. szintre az *asztali számítógép, laptop számítógép, oktatási szoftver, pénzügyi szoftver, ...*, a 3. szintre az *IBM asztali számítógép, ..., Microsoft oktatási szoftver...* stb. kerültek. Ebben a hierarchiában a 3. szint jeleníti meg a legrészletesebb absztrakciós szintet. A fogalomhierarchiát az adatoknak megfelelően a felhasználó is kialakíthatja, vagy az adatok implicite tartalmazhatják.

6.2. táblázat | *D*, az esemény fontos adatai

TID	Vásárolt árucikk
T1	IBM asztali számítógép, Sony ff. nyomtató
T2	Microsoft oktatási szoftver, Microsoft pénzügyi szoftver
T3	Logitech egér (számítógépkellék)
T4	IBM asztali számítógép, Microsoft pénzügyi szoftver
T5	IBM asztali számítógép
...	...



6.11. ábra | Az *ElektroMind* számítógép-részlegében forgalmazott áruk fogalomhierarchiája

A 6.2. táblázatban látható elemek a 6.11. ábrán ábrázolt fogalomhierarchia legalacsonyabb szintjéről valók. Ilyen nyers, alacsony szintű adatokban nehéz megtalálni a fontos/érdekes eladási mintákat. Például az *IBM asztali számítógép* vagy a *Sony ff (fekete-fehér) nyomtató* csak a tranzakciók igen kis részében fordulnak elő, s így nehéz ezen elemeket tartalmazó erős társítási szabályokat felállítani. Kevés vásárló fogja éppen ezeket az árucikkeket együtt megvásárolni, emiatt az {*IBM asztali számítógép, Sony ff nyomtató*} elemhalmaz nem

7 | A fogalomhierarchiákat a 2. és 4. fejezetekben tárgyaltuk. Annak a célnak megfelelően, hogy e könyv fejezetei önállóan komplettek legyenek, a definíciót itt is megadjuk. Általánosítását az 5. fejezetben írtuk le.

valószínű, hogy kielégítené a minimális megalapozottság kritériumát. Ha azonban a *Sony ff nyomtatót* általánosítjuk *ff nyomtatóra*, akkor sokkal inkább találhatunk társítási szabályt az *IBM asztali számítógép* és *ff nyomtató* elemekkel. Hasonlóan, sokkal többen vásárolnak *számítógépet* és *nyomtatót* együtt, mint ahányan pontosan *IBM asztali számítógépet* és *Sony ff nyomtatót* vásárolnak. Más megfogalmazásban: az általánosításokat tartalmazó elemhalmazok, példánkban az {*IBM asztali számítógép, ff nyomtató*} vagy a {*számítógép, nyomtató*}, sokkal nagyobb eséllyel elégítik ki a minimális megalapozottság kritériumát, mint a csak alacsony szintű adatokat tartalmazó elemhalmazok, például az {*IBM asztali számítógép, Sony ff nyomtató*} elemhalmaz. Tehát egyszerűbb érdekes/fontos társításokat találni, ha a fogalomhierarchia magasabb szintű fogalmait is használjuk, mintha csak a legalacsonyabbakat. ■

Az olyan szabályokat, amelyeket a társítási szabályok bányászata közben a fogalomhierarchiát is felhasználva kapunk, **többszörös szintű** vagy **többszintű** szabályoknak nevezzük.

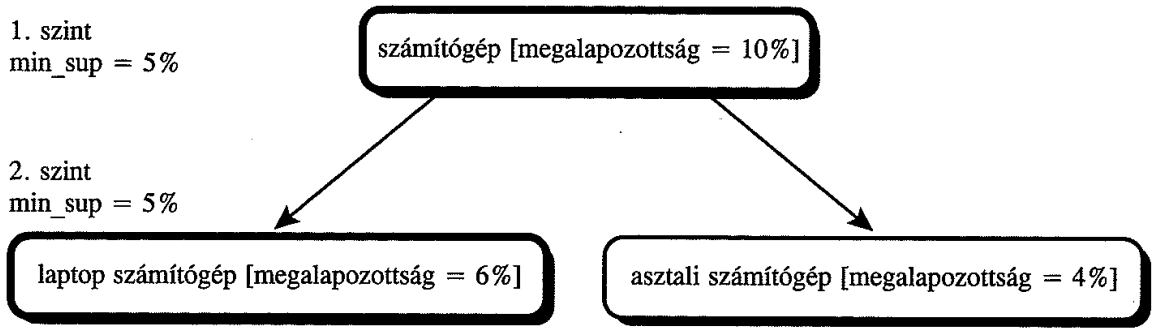
6.3.2. | A többszintű társítási szabályok bányászatáról

Hogyan tudunk többszintű társítási szabályokat keresni a fogalomhierarchia hatékony felhasználásával? Nézzük meg a megalapozottság-megbízhatóság alapon történő bányászó módszereket. Általában a felülről lefelé stratégiát használják, az elemek előfordulásait, a gyakori elemhalmazokat minden fogalomszinten meghatározzák, az 1. szinttel kezdve, majd lefelé haladva az egyre részletezőbb fogalomhierarchia szintjein mindaddig, amíg már nem találnak további gyakori elemhalmazokat. Így tehát meghatározzák az 1. szint összes gyakori elemhalmazát, majd a 2. szint gyakori elemhalmazait és így tovább. Minden szinten a gyakori elemhalmazok felfedezésére alkalmas bármely algoritmus, például az Apriori vagy változatai használhatók. A következőkben megadjuk a feladat néhány lehetséges megközelítését. Szemléltetésre a 6.12–6.16. ábrákat használjuk, melyeken a csomópontok a megvizsgált elemeket vagy elemhalmazokat jelentik, a vastagon keretezett csomópontok pedig azt jelzik, hogy a megvizsgált elem vagy elemhalmaz gyakori.

- **Minden szinten egyforma minimális megalapozottság használata** (**egyenlő megalapozottság** néven hivatkoznak rá) – Ugyanazt a minimális megalapozottsági küszöbértéket használja az absztrakció minden szintjén. Például a 6.12. ábrán a minimális megalapozottsági küszöbérték 5%. Ezt használva minden szinten, adódik, hogy a *számítógép* és a *laptop számítógép* gyakorinak minősül, az *asztali számítógép* pedig nem.

  Ha egységes minimális megalapozottsági szintet használunk, akkor a keresési eljárás egyszerűsödik. A felhasználó számára is az a legegyszerűbb, ha csak egyetlen minimális megalapozottsági szintet kell megadnia. Az optimalizáló technikát is átvehetjük, arra az ismeretre alapozva ugyanis, hogy az ős a leszármazottak szuperhalmaza: a kereséskor kihagyhatjuk az olyan elemhalmazok összes elemének vizsgálatát, amelyek őse nem elégíti ki a minimális megalapozottság követelményét.

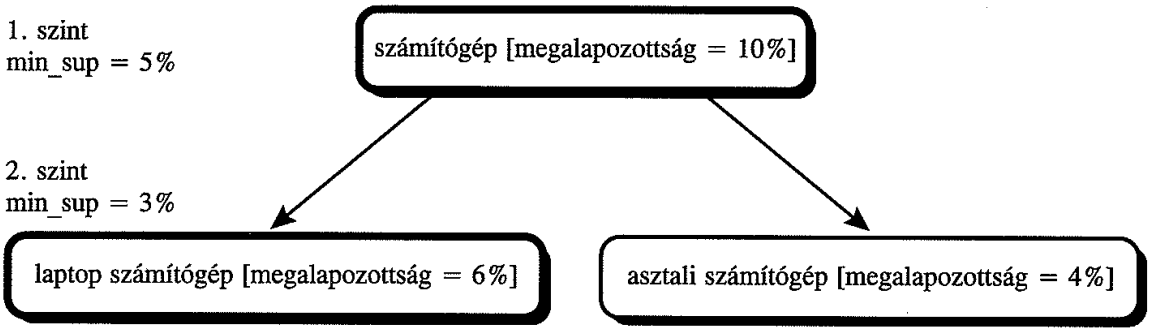
  Az egyenlő megalapozottsági megközelítésnek ugyanakkor vannak problémái is. Lehetetlen, hogy az absztrakció alacsonyabb szintjén lévő elem ugyanolyan gyakran



6.12. ábra | Többszintű bányászás egyenlő megalapozottsággal

forduljon elő, mint az absztrakció magasabb szintjén lévő elem. Ha a minimális megalapozottsági követelmény túl magas, akkor elveszíthetünk több, jelentéssel bíró, az alsóbb szinteken megjelenő társítást. Ha a követelmény túl alacsony, akkor sok érdektelen társítást generálunk a magasabb absztrakciós szinteken. Ez a megfigyelés a következő meggondoláshoz vezet:

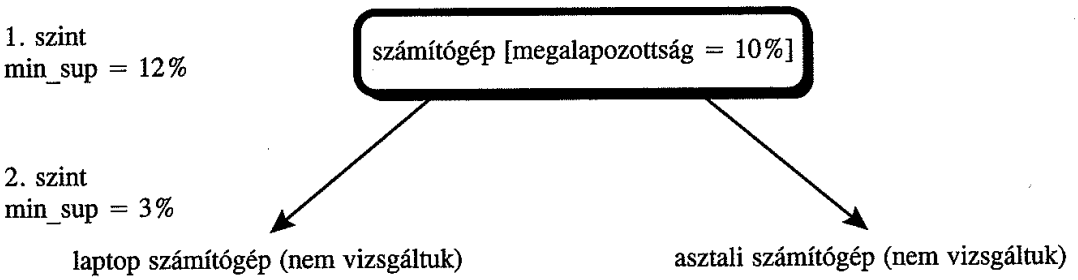
- **Magasabb szinteken használjunk magasabb minimális megalapozottsági küszöbértékeket (megalapozottság redukálása** néven hivatkoznak rá) – Az absztrakció minden szintjéhez tartozzon egy-egy saját minimális megalapozottsági küszöbérték. A magasabb absztrakciós szinthez tartozzon magasabb minimális megalapozottsági érték. Például a 6.13. ábrán az 1. szint megalapozottsági küszöbértéke 5%, a 2. szint megalapozottsági küszöbértéke 3%. Ezzel a *számítógép*, a *laptop számítógép* és az *asztali számítógép* mind gyakorinak minősülnek.



6.13. ábra | Többszintű bányászás a megalapozottság redukálásával

A *megalapozottság redukálása* módszerével többszintű társítási szabályok bányászatára számos keresési stratégia létezik:

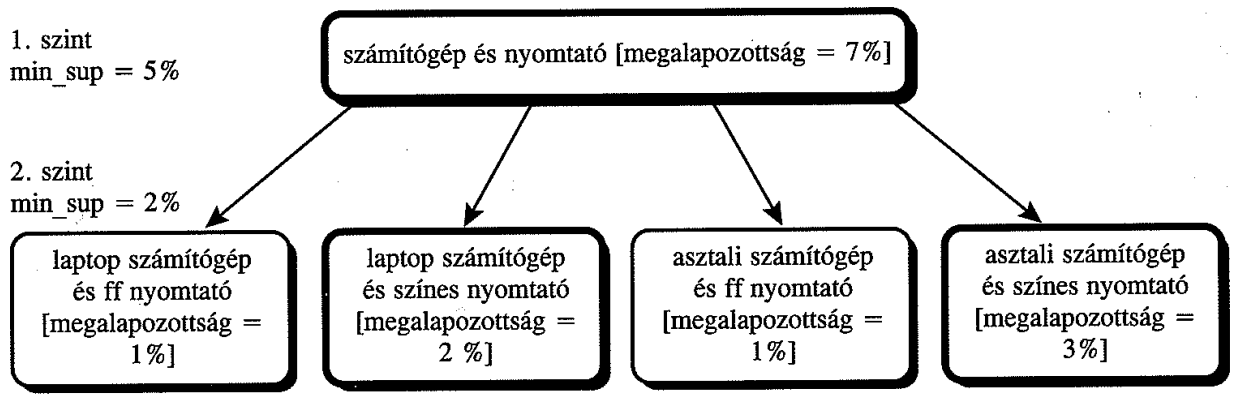
- **Szintenként, egymástól független** – Teljes szélességű keresés, ahol nincs a gyakori elemhalmazokra vonatkozó olyan háttérismeretünk, amellyel bizonyos eseteket eleve kizárhatnánk. Minden csomópont vizsgálatára sor kerül, függetlenül attól, hogy a szülő csomópont gyakorinak minősült-e vagy sem.
- **Többszintű szűrés egyedi elemenként** – Az i-edik szint valamely elemét akkor és csakis akkor vizsgáljuk meg, ha az $(i - 1)$-edik szinten lévő szülője gyakorinak minősült. Más megfogalmazásban, a részletesebb társításokat az általánosabbak alapján



6.14. ábra | Többszintű bányászás a megalapozottság redukálásával, egyedi elemenkénti többszintű szűréssel

keressük. Ha egy csomópont gyakorinak minősül, akkor a gyerekeit megvizsgáljuk, különben a leszármazottait a további vizsgálatok során figyelmen kívül hagyjuk. A 6.14. ábrán látható példában a *számítógép* csomópont leszármazottait már nem is vizsgáljuk, mert a *számítógép* nem gyakori.

- **Többszintű szűrés *k*-elemhalmazokkal** – Az *i*-edik szinten lévő *k*-elemhalmazokat akkor és csakis akkor vizsgáljuk, ha a hozzájuk tartozó, $(i - 1)$-edik szinten lévő, szülő *k*-elemhalmaza gyakori. Például, a 6.15. ábrán a {*számítógép, nyomtató*} 2-elemhalmaz gyakori, ezért a {*laptop, ff nyomtató*}, {*laptop, színes nyomtató*}, {*asztali számítógép, ff nyomtató*} és {*asztali számítógép, színes nyomtató*} csomópontok vizsgálatára sor kerül.



6.15. ábra | Többszintű bányászás a megalapozottság redukálásával, *k*-elemhalmazokkal való többszintű szűréssel (itt $k = 2$)

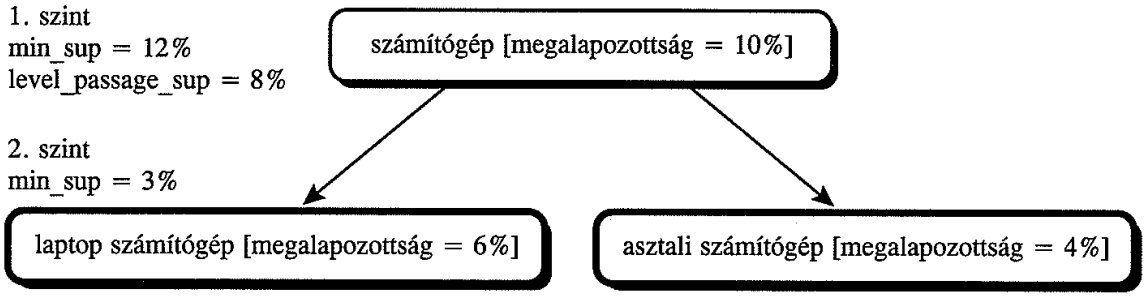
Hogyan hasonlíthatjuk össze ezeket a módszereket? A szintenként, egymástól független módszer nagyon gyenge, az alacsony szinteken sok nem gyakori, érdektelen elemet megvizsgál, ezzel jelentéktelen társításokat is talál. Például a *számítógépbútor* ritkán forgalmazott termék, akkor meglehet, hogy nem célszerű azt vizsgálni, hogy a még specifikusabb *számítógépes szék* társítható-e a *laptop*pal. Mindazonáltal, ha a *számítógépkellék* gyakran forgalmazott termékcsoport, akkor hasznos lehet megnézni például, hogy a *laptop* és az *egér* között milyen erős a társítás.

A *többszintű szűrés k-elemhalmazokkal* stratégiája a bányászó rendszer számára azt teszi lehetővé, hogy csak a gyakori *k*-elemhalmazok gyerekeivel foglalkozzanak. Ez a

megszorítás nagyon erős, többnyire nem sok k-elemhalmazt talál gyakorinak (különösen $k > 2$ esetekben). Így ezzel a megközelítéssel sok értékes minta kiszűrődhet (azaz az algoritmus nem fogja már felismerni).

A *többszintű szűrés egyedi elemenként* módszere kompromisszumot jelent a két „szélsőség" között. Mindamellett, e módszernél előfordulhat, hogy ha a szülő elem nem elégíti ki az ő szintjén érvényes minimális megalapozottság követelményét, akkor ez a módszer elveszíti az alsóbb szintű leszármazott elemei közötti társításokat (nem fedezi fel), bár azok kielégítenék a redukált minimális megalapozottság – adott szinten érvényes – követelményét. (Emlékeztetünk arra, hogy e módszerben a minimális megalapozottság küszöbszintjei szintenként különbözőek.) Például, ha a fogalomhierarchia i-edik szintjén álló *színes monitor* gyakori az i-edik szinten érvényes minimális megalapozottság szerint, de szülője, az $(i - 1)$-edik szinten lévő *monitor* nem gyakori az $(i - 1)$-edik szinten érvényes minimális megalapozottság szerint, akkor az egyébként esetleg erős, *asztali számítógép* ⇒ *színes monitor* társítási szabályt elveszítjük (nem találjuk meg).

A most vázolt probléma megoldására a *többszintű szűrés egyedi elemenként* módszer módosított verziója alkalmas. E módszer **ellenőrzött többszintű szűrés egyedi elemenként** néven ismert. A módszer a következő: bevezeti a **szint-átjárhatósági küszöbértéket**, melyet a **viszonylagosan (részben) gyakori elemek** „lefelé passzolásával" állapít meg. Más megfogalmazásban: ez a módszer lehetővé teszi, hogy a minimális megalapozottsági küszöböt el nem érő elemek gyerekeit megvizsgáljuk, mégpedig akkor, ha kielégítik a szint-átjárhatósági küszöbértéket. A fogalomhierarchia minden egyes szintjéhez saját szint-átjárhatósági küszöbérték tartozik. Egy adott szinthez tartozó szint-átjárhatósági küszöbértéket tipikusan olyan értékre állítunk be, mely a következő alacsonyabb szinthez tartozó minimális megalapozottsági küszöbérték és az adott szint minimális megalapozottsági küszöbértéke közé esik. A felhasználó választhat, hogy a szint-átjárhatósági küszöbértéket lefelé vagy felfelé csúsztatja, befolyásolva ezzel a vizsgálatba bevonandó elemek körét. A szint-átjárhatósági küszöbérték lefelé csúsztatása az alsóbb szint minimális megalapozottsági küszöbértékének közelébe lehetővé teszi minden elem leszármazottainak vizsgálatát. A 6.16. ábrán látható példában az 1. szint szint-átjárhatósági küszöbértékét (*level_passage_sup*) 8%-ra állítva lehetővé teszi a 2. szint *laptop számítógép* és *asztali számítógép* csomópontjainak vizsgálatát és gyakorinak minősítését annak ellenére, hogy szülőcsomópontjuk, *számítógép* nem minősült gyakorinak. Ezzel a módszerrel bővítve az algoritmust, a felhasználó számára a többszintű bányászó



6.16. ábra | Többszintű bányászás az ellenőrzött többszintű szűrés egyedi elemenként módszerével

folyamat vezérlésének rugalmas lehetőségét biztosítjuk, és egyúttal csökkentjük az olyan, jelentéktelen társítási szabályok számát, amelyeket különben vizsgálnánk és előállítanánk.

Eddigi tárgyalásunk középpontjában az olyan gyakori elemhalmazok keresése állt, ahol az elemhalmaz minden eleme a fogalomhierarchia ugyanazon szintjéhez tartozott. Így találtuk a *számítógép* ⇒ *nyomtató* (mind a *számítógép,* mind a *nyomtató* 1-es hierarchiaszintű) és az *asztali számítógép* ⇒ *ff nyomtató* (az *asztali számítógép* és a *ff nyomtató* mindegyike 2-es hierarchia szintű) társítási szabályokat. Tegyük fel, hogy ezek helyett *fogalomhierarchia határokat átlépő*, a *számítógép* ⇒ *ff nyomtató* szabályhoz hasonlókat szeretnénk találni, amelyekben a hivatkozott elemekre nézve nem előírás, hogy a fogalomhierarchia ugyanazon szintjéhez kellene tartozniuk. Az ilyen szabályokat **szintátlépő társítási szabályoknak** nevezzük.

Hogyan lehet szintátlépő társítási szabályokat bányászni? Ha fogalomhierarchia *i*-edik és *j*-edik szintjén keresünk társítási szabályokat (ahol *j* a részletezőbb, azaz alacsonyabb szint), akkor a *j*-edik szint redukált megalapozottsági küszöbértékét használjuk általánosan, és ezzel a *j*-edik szint minden eleme részt vesz a vizsgálatban.

6.3.3. | Többszintű társítási szabályok feleslegességének ellenőrzése

A fogalomhierarchiák azért hasznosak az adatbányászatban, mert – mint a többszintű társítási szabályokban is megfogalmazódik –, különböző absztrakciós szintű tudás felfedezését teszik lehetővé. A többszintű társítási szabályok bányászásakor megtalált szabályok némelyike feleslegesnek tűnhet, az elemek közötti „ős-leszármazott" viszony miatt. Például tegyük fel, hogy a következő szabályokat találtuk:

$$\textit{asztali számítógép} \Rightarrow \textit{ff nyomtató}\ [\textit{megalapozottság} = 8\%,\ \textit{megbízhatóság} = 70\%] \qquad (6.9)$$

$$\textit{IBM asztali számítógép} \Rightarrow \textit{ff nyomtató}\ [\textit{megalapozottság} = 2\%,\ \textit{megbízhatóság} = 72\%] \qquad (6.10)$$

ahol a 6.11. ábra fogalomhierarchiája szerint az *asztali számítógép* az *IBM asztali számítógép* őse.

Lehet, hogy eltűnődünk a következő kérdésen: *Ha a bányászás során mind a (6.9), mind a (6.10) szabályt megkaptuk, akkor az utóbbi mennyire hasznos? Ad-e valóban új információt?* Ha az utóbbi, kevésbé általános szabály nem ad új információt, akkor eldobhatjuk. Nézzük meg, hogyan is lehet ezt eldönteni! Az $R1$ szabály **őse** az $R2$ szabálynak, ha $R1$ előállítható az $R2$-ből, az abban előforduló elemeknek – a fogalomhierarchiának megfelelő – őseikre való kicserélésével. Például a (6.9) őse a (6.10)-nek, mert az *asztali számítógép* őse az *IBM asztali számítógép*nek. E definíció alapján egy szabályt feleslegesnek minősíthetünk, ha a megalapozottsága és megbízhatósága elég közeli a szabály ősében megállapított megfelelő értékekhez. Szemléltetésképpen tegyük fel, hogy a (6.9) szabály megbízhatósága 70%, megalapozottsága 8%, továbbá, hogy a forgalmazott *asztali számítógép*ek nagyjából negyedrésze *IBM asztali számítógép*. A (6.10) szabály

70% körüli megbízhatóságú (hiszen minden *IBM asztali számítógép* egyúttal *asztali számítógép* is), és megalapozottsága 2% körüli (azaz 8% × ¼). Ha valójában ez a helyzet, akkor a (6.10) szabály érdektelen, mert nem biztosít semmilyen további információt, és kevésbé általános, mint a (6.9) szabály.

6.4. | Többdimenziós társítási szabályok bányászata relációs adatbázisokban és adattárházakban

Ebben a fejezetben többdimenziós társítási szabályok bányászatát fogjuk megismerni, azaz olyan szabályok bányászatát, amelyekben egynél több dimenzió vagy predikátum fordul elő. (Ilyen például az a szabály, amelyben a vásárló által *vásárolt* dolgok és a vásárló *életkora* is szerepel.) Ezek a bányászati módszerek a mennyiségi attribútumok kezelésén alapulnak.

6.4.1. | Többdimenziós társítási szabályok

Ebben a fejezetben eddig csak olyan társítási szabályokkal foglalkozunk, melyek egyetlen predikátumból épültek fel, konkrétan a *vásárlás* predikátumból. Például az *ElektroMind* cég adatbázisából felfedeztük az *IBM asztali számítógép* ⇒ *Sony ff nyomtató* Boole társítási szabályt, melyet

$$vásárol(X, \text{“}IBM\ asztali\ számítógép\text{”}) \Rightarrow vásárol(X, \text{“Sony } ff\ nyomtató\text{”}) \qquad (6.11)$$

formában is leírhatunk, ahol X változó olyan vásárlót reprezentál, aki az *ElektroMind* cégnél valamit vásárolt. A többdimenziós adatbázisoknál alkalmazott terminológiát követve a szabály minden különböző predikátumára dimenzióként hivatkozunk. Ennek megfelelően a (6.11) szabályt **egydimenziós** vagy **intradimenzionális szabálynak** nevezzük, hiszen egyetlen predikátumot (konkrétan a *vásárol* predikátumot) tartalmazza, igaz, azt többszörösen. Amint e fejezet korábbi részeiben láttuk, ilyen szabályok a tranzakcionális adatokból bányászhatók.

Mindazonáltal feltételezhető, hogy a kereskedelmi és hasonló információk tárolására inkább relációs adatbázisokat vagy adattárházakat használnak, és csak kevésbé a tranzakcionális adatbázisokat. Ezek az adattárolási formák a definíciónak megfelelően többdimenziósak. Például az eladott árucikk eladási tranzakciókkal való nyomon követése helyett a relációs adatbázisokban rögzíthetők további, az árucikkel kapcsolatos adatok. Tárolhatjuk az eladott mennyiséget, az árat, az eladó részleg helyét, a vevőre vonatkozó adatokat, életkorát, foglalkozását, hitelképességét, jövedelmét, címét stb. Az összes adatbázis-attribútumot vagy adattárház-dimenziót predikátumnak tekintve, érdekes lehet több predikátumot tartalmazó társítási szabályokat keresni. Ilyen társítási szabály például:

$$életkor(X, \text{“20...29”}) \wedge foglalkozás(X, \text{“}diák\text{”}) \Rightarrow vásárol(X, \text{“}laptop\text{”}) \qquad (6.12)$$

Az olyan társítási szabályokat, amelyekben kettő vagy több dimenzió vagy predikátum fordul elő, **többdimenziós társítási szabálynak** nevezzük. A (6.12) szabály három predi-

kátumot (*életkor*, *foglalkozás* és *vásárol*) tartalmaz, ezek mindegyike csak egyszer fordul elő a szabályban, ezért azt mondjuk, hogy ezek **nem ismétlődő predikátumok**. A kizárólag nem ismétlődő predikátumokat tartalmazó többdimenziós társítási szabályokat **interdimenzionális társítási szabályoknak** nevezzük. Természetesen foglalkozhatunk olyan többdimenziós szabályok bányászásával is, amelyekben a predikátumok ismétlődnek, azaz ugyanaz a predikátum többször fordul elő. Az ilyen szabályokat **hibrid-dimenziós társítási szabályoknak** nevezzük. Példa ilyen szabályra, melyben a *vásárol* predikátum az ismétlődő:

$$életkor(X, \text{“20...29”}) \wedge vásárol(X, \text{“}laptop\text{”}) \Rightarrow vásárol(X, \text{„}ff\ nyomtató\text{”}) \qquad (6.13)$$

Megjegyezzük, hogy az adatbázis-attribútumok kategorikusak vagy mennyiségiek. A **kategorikus attribútumok** lehetséges értékeinek száma véges, az elemek között nincs természetes sorrend. Kategorikus attribútum például a *foglalkozás*, a *szín*. A kategorikus attribútumokat **nominális attribútumoknak** is hívják, mert értékeik valamely „dolgok nevei”. A **mennyiségi attribútumok** értékei számok, a lehetséges értékek impliciten rendezettek. Mennyiségi attribútum például az *életkor*, a *jövedelem*, az *ár*. A többdimenziós társítási szabályok keresésének módszereit a mennyiségi attribútumok kezelése három alapmegközelítése szerint osztályozhatjuk.

Az első megközelítés szerint a *mennyiségi attribútumok diszkretizáltak, valamely előre meghatározott felosztásnak megfelelően.* A diszkretizációt a bányászás előtt hajtjuk végre. Például a *jövedelem* eredeti értékét kicseréljük tartományokra, mondjuk a “0...20 000”, “21 000...30 000”, “31 000...40 000” stb. tartományokra. Ekkor a diszkretizáció statikus és előre meghatározott. A diszkretizált numerikus attribútumokat a lehetséges elemhalmazukkal már kategorikus attribútumoknak tekinthetjük. E módszerre **a többdimenziós társítási szabályok bányászata a mennyiségi attribútumok statikus diszkretizációjával** módszerként hivatkozunk.

A második megközelítésben a *mennyiségi attribútumokat az adatok szétosztásán alapuló „kosarakba” osztással diszkretizáljuk.* A kosarak a bányászási folyamat közben alakulnak ki. A diszkretizáció *dinamikus*, és úgy állapítjuk meg, hogy megfeleljen valamely bányászási követelménynek, például annak, hogy maximalizáljuk a bányászott szabályok megbízhatóságát. Ez a módszer a numerikus attribútumértékeket inkább mennyiségi értékként kezeli, s nem mint előre meghatározott értéktartományokat vagy kategóriákat, így az e módszerrel bányászott társítási szabályokat **mennyiségi társítási szabályoknak** nevezzük.

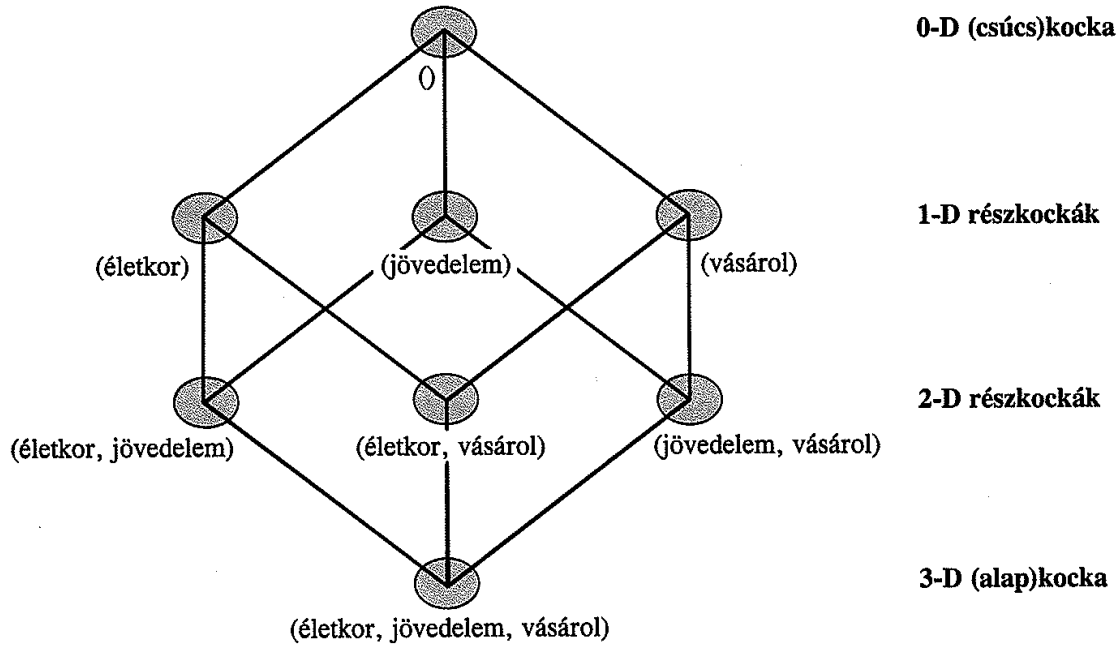
A harmadik megközelítés *a mennyiségi attribútumokat az adatintervallum szemantikus jelentésének megfelelően diszkretizálja.* Ez a dinamikus diszkretizáló eljárás az adatpontok közötti távolságot veszi figyelembe. Az így talált társítási szabályokat a diszkretizáció módszerének megfelelően, **távolság alapú társítási szabályoknak** nevezzük.

A következőkben a többdimenziós társítási szabályok bányászatának fenti módszereit fogjuk tanulmányozni. Az egyszerűség kedvéért a vizsgálatainkat az interdimenziós társítási szabályokra korlátozzuk. Megjegyezzük, hogy gyakori elemhalmazok keresése (amikre az egydimenziós szabályok bányászatánál szükségünk volt) helyett a többdimenziós társítási szabályok bányászatakor gyakori *predikátumhalmazokat* keresünk. A *k*-predikátumhalmaz, *k* darab konjunktív predikátumot tartalmaz. Például a (6.12) szabály predikátumhalmaza {*életkor*, *foglalkozás*, *vásárol*} egy 3-predikátumhalmaz. Az elemhalmazoknál alkalmazott jelölésekhez hasonlóan az L_k jelölést használjuk a gyakori *k*-perdikátumhalmazok halmazára.

6.4.2. | Többdimenziós társítási szabályok bányászata a mennyiségi attribútumok statikus diszkretizációjával

Ebben az esetben a mennyiségi attribútumokat a bányászást megelőzően, előre meghatározott felosztásnak megfelelően diszkretizáljuk. A numerikus értékeket tartományokra cseréljük. A kategorikus értékek, ha szükséges, a fogalomhierarchiával összhangban általánosíthatók. Ha a keletkező adatokat egy relációs táblában tároljuk, akkor az Apriori algoritmus alkalmazásához csak kisebb módosítás szükséges, s így az összes gyakori predikátumhalmazt meg tudjuk találni. (Nevezetesen, egy-egy *vásárol*-szerű attribútum keresése helyett az összes releváns attribútumot kell keresnünk.) Az összes gyakori k-predikátumhalmazok megkeresése céljából a tábla k-szori vagy $k + 1$-szeri átvizsgálására kerül sor. Más stratégiákkal, a tördelő módszerekkel, a particionálással, a mintavételezéssel növelhetjük a keresés hatékonyságát.

Másik lehetőségként adódik a keletkező adatok *adatkockában* való tárolása. A többdimenziós társítási szabályok bányászatára az adatkockák igen alkalmasak, mert eleve többdimenziósak. Az adatkockákat és a velük kapcsolatos algoritmusokat a 2. fejezetben részletesen tárgyaltuk. Emlékeztetőül: az adatkocka többdimenziós adatstruktúrák kockáinak hálójából áll. E struktúrák a megfigyelt folyamatra vonatkozó adatok mellett összesített, csoportosított információkat is tartalmazhatnak. A 6.17. ábrán az *életkor*, *jövedelem* és *vásárol* dimenziók adatkockahálóját látjuk. Az n-dimenziós kockák celláit használjuk a megfelelő n-predikátumhalmazok előfordulási számának tárolására is. Az alapkockák a folyamat adatait, *életkor*, *jövedelem* és *vásárol* tárolják; a (*életkor*, *jövedelem*) kétdimenziós (2-D) részkocka összesít az *életkor* és a *jövedelem* szerint; a nulladimenziós (0-D) kocka (a csúcs) a tranzakciók teljes számát tartalmazza; hasonlóan értelmezzük a többieket is.



6.17. ábra | Egy 3-D adatkocka részkockákból felépülő hálója. Minden részkocka más-más csoportosítást reprezentál. Az alapkocka három predikátumot (*életkor*, *jövedelem* és *vásárol*) tartalmaz

Az adattárházak és az OLAP-technológia „örökös növekedés" tulajdonsága miatt lehetséges, hogy a felhasználó számára érdekes dimenziókat tartalmazó adatkocka már létezik. *Ha így van, akkor mit tegyünk azért, hogy megtaláljuk a gyakori predikátumhalmazokat?* A stratégia hasonló ahhoz, amit az Apriori algoritmusban használtunk. Azon az előzetes ismereten alapul, miszerint *gyakori predikátumhalmaz minden részhalmazának is gyakorinak kell lennie.* Ez a tulajdonság felhasználható a generált jelölt-predikátumhalmazok számának csökkentésére.

Az olyan esetekben, ha nem létezik az éppen vizsgálni kívánt adatok adatkockája, akkor azt elő kell állítani. A 2. fejezetben megadtuk az adatkocka előállításának gyors, hatékony algoritmusát. Ezt módosíthatjuk úgy, hogy a kocka megszerkesztése közben keresse meg a gyakori elemhalmazokat is.

6.4.3. | A mennyiségi társítási szabályok bányászata

A mennyiségi társítási szabályok olyan többdimenziós társítási szabályok, amelyekben a numerikus attribútumokat a bányászási folyamat közben *dinamikusan* diszkretizáljuk oly módon, hogy valamely bányászási feltételnek – például a megbízhatóság maximalizálása vagy a bányászott szabályok tömörsége – megfeleljenek. Ebben az alfejezetben speciálisan az olyan mennyiségi társítási szabályok bányászatára koncentrálunk, amelyek a szabály bal oldalán két mennyiségi attribútumra hivatkoznak, a szabály jobb oldalán pedig egy kategorikus attribútumot tartalmaznak. Ilyen szabály például:

$$A_{quant1} \wedge A_{quant1} \Rightarrow A_{cat},$$

ahol A_{quant1} és A_{quant2} mennyiségi attribútumok intervallumaira vonatkoznak (az intervallumok meghatározása dinamikus), A_{cat} pedig a vizsgált adatok kategorikus attribútumára vonatkozik. Az ilyen szabályokat, mivel két mennyiségi dimenzióra hivatkoznak, **kétdimenziós mennyiségi társítási szabályoknak** nevezzük. Tegyük fel például, hogy mennyiségi attribútumpárok (például a vásárló életkora és jövedelme, valamint a megvásárolni kívánt televízió típusa) társítási viszonyaira vagyunk kíváncsiak. Az ilyen 2-D mennyiségi társítási szabályok egy példája a következő:

$$\textit{életkor}(X, \text{“30...39”}) \wedge \textit{jövedelem}(X, \text{“42 000...48 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.14)$$

Hogyan találhatunk ilyen szabályokat? Vizsgáljuk meg az ARCS-rendszerben (Association Rule Clustering System, társítási szabályokat osztályozó rendszer) használt – a képfeldolgozásból átvett – módszert. A lényegét tekintve ez a módszer a mennyiségi attribútumpárokat a kategorikus attribútumra megadott feltételnek megfelelően, kétdimenziós (2-D) rács rácspontjaiba helyezi el. Ezután a rács vizsgálatával a pontok osztályait keressük meg, ezekből tudjuk majd a szabályokat előállítani. Az ARCS a következő lépéseket hajtja végre:

- **Kosarazás** – A mennyiségi attribútumok értékei az attribútum értéktartományából származó, nagyon változatos értékek lehetnek. Gondoljunk csak arra, hogy milyen

nagy kétdimenziós rács keletkezne, ha a két tengelyen az *életkor* és a *jövedelem* pontos értékeit ábrázolnánk, azaz ha az *életkor* minden lehetséges értékéhez a tengely egy-egy pontját feleltetnénk meg, s ugyanezt tennénk a másik tengelyen a *jövedelem* minden egyedi értékéhez egy-egy pozíciót rendelve. Azért, hogy a rács még kezelhető méretű maradjon, a mennyiségi attribútumok értékeit intervallumokba osztjuk. Az intervallumok dinamikusak, a bányászási folyamat során más-más felosztásokra térhetünk át. A felosztó eljárást **kosarazásnak** is nevezzük, ahol a kosarak a felosztás intervallumai. Általános kosarazási stratégiák:

- **azonos szélességű kosarazás**, ahol az intervallumok mérete minden kosárra azonos;
- **azonos mélységű kosarazás**, ahol minden kosárhoz nagyjából azonos számú elem tartozik;
- **homogenitáson alapuló kosarazás**, ahol a kosarak méretét úgy határozzák meg, hogy a kosarakba az elemek egyenletesen kerüljenek szétosztásra.

Az ARCS-rendszer az azonos szélességű kosarazás módszerét használja, ahol a mennyiségi attribútumokhoz használandó kosarak méretét a felhasználó adja meg. Kétdimenziós tömb generálódik, magában foglalva az összes lehetséges kosárkombinációt. A tömb mindegyik cellája tartalmazza a szabály jobb oldalán lévő kategorikus attribútum lehetséges osztályaihoz tartozó szétosztási számlálókat. Ezen adatstruktúra megkonstruálásához az eredeti adatokat csak egyszer kell feldolgoznunk. Ugyanez a kédimenziós tömb használható a kategorikus attribútumok bármely értékéhez tartozó, az adott mennyiségi attribútumokon alapuló szabályok előállítására. A kosarazást már a 3. fejezetben is tárgyaltuk.

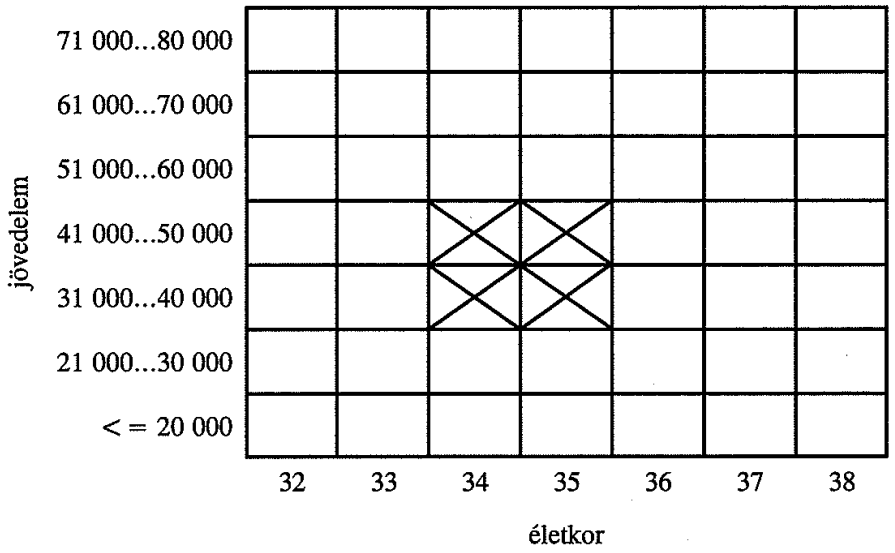
- **A gyakori predikátumhalmazok megkeresése** – Először a fenti kétdimenziós tömböt állítjuk elő, a gyakori (tehát a minimális megalapozottságot kielégítő) predikátumhalmazok megkeresésének érdekében ezt a tömböt fogjuk feldolgozni. A megtalált predikátumhalmazok felhasználásával, a 6.2.2. alfejezetben bemutatott szabályelőállító algoritmushoz hasonló algoritmussal előállíthatjuk az erős társítási szabályokat.
- **A társítási szabályok klaszterezése** – Az előző lépésben megtalált erős társítási szabályokat a kétdimenziós rácshoz kell rendelnünk. A 6.18. ábra azt az *életkor* és *jövedelem,* mennyiségi attribútumokkal megadott kétdimenziós rácsot mutatja be, amely a *vásárol*(*X*, "*nagy felbontású TV*")-t megjósoló mennyiségi társítási szabály jobb oldalának állításához tartozik. A négy *X*-el jelölt cellához a következő szabályok tartoznak:

$$\textit{életkor}(X, 34) \wedge \textit{jövedelem}(X, \text{“31 000...40 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.15)$$

$$\textit{életkor}(X, 35) \wedge \textit{jövedelem}(X, \text{“31 000...40 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.16)$$

$$\textit{életkor}(X, 34) \wedge \textit{jövedelem}(X, \text{“41 000...50 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.17)$$

$$\textit{életkor}(X, 35) \wedge \textit{jövedelem}(X, \text{“41 000...50 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.18)$$



6.18. ábra | Kétdimenziós rács a nagy felbontású televíziót vásárlók jellemző adataival

Találhatunk-e a fenti négy szabályt helyettesítő egyszerűbb szabályt? Észrevehetjük, hogy ezek a szabályok nagyon „közeliek" egymáshoz, a rácsban is láthatóan egy csoportot alkotnak. Valóban, a négy szabályt összekapcsolhatjuk vagy „egy osztályba vonhatjuk", és ezáltal a következő egyszerűbb szabályhoz jutunk, mely szabályban összegződik a fenti négy szabály, azok e szabályra lecserélhetők:

$$\textit{életkor}(X, \text{“34...35”}) \wedge \textit{jövedelem}(X, \text{“31 000...50 000”}) \Rightarrow \textit{vásárol}(X, \text{“}\textit{nagy felbontású TV}\text{”}) \quad (6.19)$$

Az ARCS-rendszer ebből a célból alkalmazza a klaszterezési algoritmust. Az algoritmus a rácsban a fentihez hasonlóan téglalapot képező együttes szabálynégyeseket keres. Ha talál, akkor az később a kosarak összevonásához, más megközelítésben a mennyiségi attribútumok diszkretizációjának dinamikus megváltoztatásához vezethet.

Az itt bemutatott rács alapú technika feltételezi, hogy a kiinduló társítási szabályok zárt, téglalapszerű területre csoportosíthatók. Az osztályozást megelőzően végrehajtott simítási, egyengetési technikák segíthetnek az adatokban lévő zajok és kiugrások (szélsőséges értékek) eltávolítására. A téglalapszerű csoportok (klaszterek) leegyszerűsíthetik az adatokat. Más módszerek, amelyek más formájú területekre alapozottan tovább egyszerűsítik az adatokat, nagy számítási erőforrásokat igényelnek.

Nem rács alapú módszereket is javasolnak a mennyiségi társítási szabályok keresésére. Ezekkel sokkal általánosabb alakú szabályok találhatók, amelyekben a szabály bármelyik oldalán akárhány mennyiségi és kategorikus attribútum előfordulhat. E módszerekben a mennyiségi attribútumokat dinamikusan diszkretizálják az azonos mélységű kosarazás módszerével. A diszkretizációval keletkezett részterületeket a *részleges teljesség* szempontja (ez az újrafelosztás információveszteségét számszerűsíti) szerint kombinálják. Az ARCS ezen változatairól többet az irodalomban hivatkozott anyagokban találhatunk.

6.4.4. | A társítási szabályok távolság alapú bányászata

Az előző fejezetben leírt mennyiségi társítási szabályokban a mennyiségi attribútumokat kezdetben a kosarazási módszerekkel diszkretizáltuk, majd a keletkezett intervallumokat (azok közül bizonyos szomszédosakat) egymással egyesítettük. Ebben a megközelítésben az adatintervallum jelentését nem vizsgáltuk, nem állapítottunk meg az adatelemek közötti vagy az intervallumok közötti távolságot.

Vizsgáljuk meg például a 6.3. táblázatot, mely az *árra* vonatkozó adatokat mutatja be, az azonos szélességű, az azonos mélységű kosarazás és a távolság alapú felosztásokat összehasonlítva. A távolság alapú felosztás olyannak látszik, mint amely sokkal több jelentést hordoz, mert az egymáshoz közeli értékeket egy csoportba sorolja (például a [20, 22] csoportba). Ezzel szemben az azonos mélységű kosarazás egymástól igen távoli elemeket egy csoportba sorolt (például a [22, 50] csoport). Az azonos szélességű kosarazással előfordulhat, hogy közeli adatok különböző intervallumokba kerülnek, és keletkezhetnek üres intervallumok is, amelyekben nincs adat. Világos, hogy a távolság alapú felosztás, amely akkor sorol egy intervallumba adatokat, ha azok „közeliek", elősegíti olyan intervallumok kialakítását, amelyek sokkal inkább bírhatnak valamilyen jelentéssel, mint a másik két módszer által létrehozott intervallumok. A mennyiségi attribútumokra vonatkozó jelentéssel is bíró, távolság alapú intervallumokat az attribútum értékein végrehajtott klaszterelemzés eredményére alapozva tudjuk meghatározni.

6.3. táblázat | Az azonos szélességű és az azonos mélységű kosarazási módszerek nem alkalmasak az adatintervallum jelentéséhez való alkalmazkodásra

Ár ($)	Azonos szélesség (10 $)	Azonos mélység (mélység: 2)	Távolság alapú
7	[0, 10]	[7, 20]	[7, 7]
20	[11, 20]	[22, 50]	[20, 22]
22	[21, 30]	[51, 53]	[50, 53]
50	[31, 40]		
51	[41, 50]		
53	[51, 60]		

A társítási szabályok hátránya, hogy nem alkalmasak az attribútumértékek megközelítő használatára. Megvizsgálva a következő társítási szabályt:

$$\textit{áru\_típusa}(X, \text{"}\textit{elektronika}\text{"}) \wedge \textit{gyártó}(X, \text{"}\textit{külföldi}\text{"}) \Rightarrow \textit{ár}(X, 200\ \$), \qquad (6.20)$$

ahol az X változó az *ElektroMind* cég valamely árucikkét reprezentálja. A valóságban sokkal valószínűbb az, hogy a külföldi elektronikus árucikk ára 200 $-hoz *közeli* vagy *megközelítően* 200 $, és nem pontosan 200 $. Hasznos lenne, ha volnának olyan társítási szabályok, amelyek alkalmasak a hasonló közelítések kifejezésére. Megjegyezzük, hogy a megalapozottság és a megbízhatóság értékeit a közeliség megállapításánál nem vesszük figyelembe. Fenti meggondolások motiválják a **távolság alapú társítási szabályok** bányászatát. Ezek az adatintervallum jelentését figyelembe véve, lehetővé teszik az adatértékekben a megközelítő értékek használatát is. A távolság alapú társítási szabályok bá-

nyászatára kétfázisú algoritmust alkalmazunk. Az első fázis klaszterezést használ az intervallumok, csoportok megállapítására, felhasználva a rendelkezésre álló teljes memóriát. A második fázis, a gyakran együtt előforduló csoportok keresésével, előállítja a távolság alapú társítási szabályokat.

Hogyan lehet az első fázisban kialakítani a csoportokat? A kérdésre először intuitív választ adunk. Az érdeklődő olvasók olvassák el a 8. fejezetet, valamint e fejezet távolság alapú társítási szabályokra vonatkozó irodalmi hivatkozásaiban említett irodalmakat. Legyen N darab sornak az X attribútumhalmazra vetített halmaza $S[X]$ a $t_1, t_2, \ldots, t_N$. Definiáljuk az **átmérő** mértéket a sorok közelségének mérésére. Az $S[X]$ átmérője az X-re vetített sorok páronkénti távolságainak átlaga. A távolság mérésére mind az Euklideszi, mind a Manhattan-távolságot használhatjuk.[8] Minél kisebb $S[X]$ átmérője, annál közelebbiek lesznek a sorok X-re vetített képei. Így az átmérő a csoport (klaszter) *sűrűségének* megállapítására alkalmas. Az X attribútumhalmazon definiált sorok C_X halmaza **csoport (klaszter)**, ha sorai kielégítik a **sűrűségi előírást** éppúgy, mint a **gyakorisági előírást**, mely a csoportbeli sorok számának minimumát határozza meg. A 8. fejezetben leírt klaszterezési módszert lehet úgy módosítani, hogy alkalmas legyen a bányászási folyamat első fázisában való felhasználásra.

A második fázisban a csoportok egyesítésével megkonstruáljuk a távolság alapú társítási szabályokat. Vegyünk egy $C_X \Rightarrow C_Y$ alakú egyszerű, távolság alapú társítási szabályt. Tegyük fel, hogy X az {*életkor*} attribútumhalmaz, Y pedig a {*jövedelem*} attribútumhalmaz. Szeretnénk megbizonyosodni arról, hogy az *életkor*, C_X csoport és a *jövedelem* C_Y csoportok közötti kapcsolat erős. Ez azt jelenti, hogy ha a C_X *életkor* szerint osztályozott sorait vetítjük a *jövedelem* attribútumra, akkor a hozzájuk tartozó *jövedelem* értékek a C_Y csoportba vagy ahhoz közel esnek. A C_X csoport Y attribútumhalmazba való vetített képét $C_X[Y]$-nal jelöljük. Így a $C_X[Y]$ és a $C_Y[Y]$ közötti távolságnak kicsinek kell lennie. Ezt a távolságot használjuk a C_X és C_Y közötti *társítás fokának* mérésére. A $C_X[Y]$ és a $C_Y[Y]$ közötti kisebb távolság a C_X és a C_Y közötti társítás fokának erősödését jelenti. A társítás mértékének definiálásához standard statisztikai mértékeket, például az átlagos klasztertávolságot vagy a centroid Manhattan-távolságot használhatjuk. A csoport centroidjának a sorai átlagából képzett átlagsort tekintjük.

Általánosságban a csoportok összevonásával a következő formájú távolság alapú társítási szabályokat nyerünk:

$$C_{X_1} C_{X_2} \ldots C_{X_x} \Rightarrow C_{Y_1} C_{Y_2} \ldots C_{Y_y},$$

ahol X_i és Y_j páronként diszjunkt attribútumhalmazok, és kielégítik a következő három feltételt: (1) a szabály feltétel oldalán szereplő csoportok mind erősen társítottak a szabály következmény oldalán álló valamennyi csoporttal; (2) a szabály feltétel oldalán előforduló csoportok együttesen fordulnak elő; (3) a szabály következmény oldalán előforduló csoportok együttesen fordulnak elő. A társítás foka lép a nem távolság alapú társítási szabályoknál bevezetett megbízhatóság helyébe, a megalapozottság helyébe pedig a sűrűség kerül.

8 | A $t_1=(x_{11}, x_{12}, \ldots, x_{1m})$ és $t_2=(x_{21}, x_{22}, \ldots, x_{2m})$ sorok Euklideszi távolsága: $\sqrt{\sum_{i=1}^{m}(x_{1i}-x_{2i})^2}$, Manhattan-távolsága: $\sum_{i=1}^{m}|x_{1i}-x_{2i}|$.

6.5. | A társítás bányászásától a korrelációs analízisig

Amikor társítási szabályokat bányászunk, hogyan jelzi az adatbányászrendszer, hogy mely szabályok lesznek érdekesek a felhasználó számára? A társítási szabályokat bányászó rendszerek közül sok alkalmazza a megalapozottság-megbízhatóság vizsgálati módszert. Az érdektelen szabályok kiszórására vagy kizárására a minimális megalapozottsági és minimális megbízhatósági küszöbértékeket használjuk. Ennek ellenére a szabálygeneráló algoritmusok még mindig sok, a felhasználó számára érdektelen szabályt állítanak elő. Ebben a fejezetben először megnézzük, hogy hogyan lehetnek erős társítási szabályok mégis érdektelenek vagy akár félrevezetők, később további, a statisztikai függetlenségen és korrelációs analízisen alapuló mértékeket tárgyalunk, amelyekkel az érdektelen szabályok esetleg jobban kiszűrhetők.

6.5.1. | Az erős szabályok nem feltétlenül érdekesek – példa

Az adatbányászattal megtalált összes erős társítási szabály (azaz azok, amelyek kielégítik mind a minimális megalapozottság, mind a minimális megbízhatóság követelményét) vajon eléggé érdekes-e a felhasználó számára? Nem feltétlenül. Hogy egy szabály érdekes-e vagy sem, az részben szubjektív, részben objektív megítélés kérdése. Végül is csak a felhasználó tudja eldönteni, hogy számára egy szabály érdekes-e vagy sem. Ennek a megítélése szubjektív, az egyik felhasználó véleménye eltérhet a másik felhasználó véleményétől. Mindazonáltal „háttérstatisztikán" alapuló objektív érdekességi mértékek használatával egy lépéssel közelebb jutunk célunkhoz, nevezetesen, hogy kigyomlálhassuk az érdektelen szabályokat, mielőtt azokat a felhasználónak megmutatnánk.

Nos, mely erős társítási szabályok a valóban érdekesek? Vizsgáljuk meg a következő példát!

6.6. példa | Tegyük fel, hogy az *ElektroMind* cég tranzakcióiban a számítógépjátékok és videók eladásait kívánjuk elemezni. Legyen *játék* azoknak a tranzakcióknak a halmaza, amelyben előfordul a számítógépjáték, *videó* azon tranzakciók halmaza, amelyek tartalmaznak videót. 10 000 tranzakciót elemzünk, az adatok azt mutatják, hogy 6000 tranzakcióban fordul elő a számítógépjáték, míg 7500-ban a videó és 4000-ben mindkettő. Tegyük fel, hogy a társítási szabályokat bányászó program 30%-os minimális megalapozottsági és 60%-os minimális megbízhatósági küszöbértékeket használ. A program a következő társítási szabályt fedezi fel:

$$vásárol(X, \text{"számítógépjátékok"}) \Rightarrow vásárol(X, \text{"videók"}) \quad [megalapozottság = 40\%, megbízhatóság = 66\%] \quad (6.21)$$

A (6.21) szabály erős társítási szabály, és ezért bekerül az eredménybe, hiszen a megalapozottsága 4000/10 000 = 40%, megbízhatósága 4000/6000 = 66%, ezzel kielégíti a minimális megalapozottsági és a minimális megbízhatósági követelményeket. Ugyanakkor a (6.21) szabály félrevezető, ugyanis a videó eladásának valószínűsége (önmagában is) 75%,

ami lényegesen magasabb a 66%-nál. Itt az a helyzet, hogy a számítógépjáték és a videó negatív társításban vannak, ugyanis e két árucikk egyikének megvásárlása csökkenti a másik megvásárlásának esélyét. E jelenség teljes megértése nélkül a levezetett szabályra alapozva hibás üzleti döntéseket hozhatnánk. ■

A fenti példa azt illusztrálja, hogy az $A \Rightarrow B$ szabály megbízhatósága megtévesztő lehet abban, hogy ez csak az $A|B$ feltételes valószínűség egy *becslése*. Ez nem méri az A és B közötti összefüggés valódi erejét (vagy erőtlenségét). Mindazonáltal a megalapozottság-megbízhatóság vizsgálat alternatívái hasznosak lehetnek az érdekes adatkapcsolatok bányászásában.

6.5.2. | A társításelemzéstől a korrelációs analízisig

A megalapozottság-megbízhatóság vizsgálattal dolgozó társítási szabály bányászat sok alkalmazásban hasznos. Ugyanakkor a megalapozottság-megbízhatóság vizsgálat félrevezető lehet abban, hogy az $A \Rightarrow B$ szabályt fontosnak minősíti, pedig a valóságban A előfordulása nem implikálja B bekövetkezését. Ebben az alfejezetben az adatelemek közötti érdekes összefüggések keresésére alkalmas, korreláción alapuló alternatív megoldást vizsgálunk meg.

Az A elemhalmaz bekövetkezése **független** a B elemhalmaz bekövetkezésétől, ha $P(A \cup B) = P(A)P(B)$, különben az A és B elemhalmazok mint események **összefüggőek** vagy **korreláltak**. Ez a definíció egyszerűen kiterjeszthető kettőnél több elemhalmazra. Az A és B bekövetkezései közötti korreláció mértékét a következő formula adja meg:

$$\text{corr}_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}. \tag{6.22}$$

Ha a (6.22) szerint kiszámított érték 1-nél kisebb, akkor A bekövetkezése negatívan korrelált B bekövetkezésével. Ha a kiszámított érték 1-nél nagyobb, akkor A és B pozitívan korreláltak, abban az értelemben, hogy egyikük bekövetkezése implikálja a másik bekövetkezését. Ha a kiszámított érték pontosan 1, akkor A és B függetlenek, vagyis nem korrelálnak.

A (6.22) formula ekvivalens $P(B|A)/P(B)$, melyet az $A \Rightarrow B$ társítási szabályhoz tartozó felemelésnek is neveznek. Például, ha A megfelel a számítógépjáték eladásának, B pedig a videó eladásának, akkor az adott piaci feltételeknek megfelelően a (6.22) formulával kiszámított mértékben a játékeladás növeli vagy „emeli" a videóeladások valószínűségét.

Térjünk vissza a 6.6. példa számítógépjáték és videó értékesítési adataihoz.

6.7. példa | A félrevezető, $A \Rightarrow B$ alakú, „erős" társítási szabályok kiszűrésének elősegítésére tanulmányoznunk kell, hogyan függ össze (hogyan korrelál) az A és B elemhalmaz. Jelölje *játék* a 6.6. példa tranzakciói közül azokat, amelyek nem tartalmazzák a számítógépjátékot, *videó* pedig azokat, amelyek nem tartalmazzák a videókat. A tranzakciókat kontingenciatáblázatban összegezhetjük. A 6.6. feladat adatainak kontingenciatáblázata a

6.4. táblázat. A táblázatból láthatjuk, hogy a számítógépjáték-eladás valószínűsége $P(\{játék\}) = 0{,}60$, a videóeladás valószínűsége $P(\{videó\}) = 0{,}75$, mindkettő eladásának valószínűsége $P(\{játék, videó\}) = 0{,}40$. A (6.22) formulával $P(\{játék, videó\})/(P(\{játék\}) \times P(\{videó\})) = 0{,}40/(0{,}60 \times 0{,}75) = 0{,}89$. Minthogy ez az érték 1-nél kisebb, negatív korreláció van a $\{játék\}$ és a $\{videó\}$ bekövetkezése között. A számláló annak valószínűségét adja meg, hogy a vásárló mindkét árut megvásárolja, a nevezőben szereplő érték pedig annak a valószínűsége, hogy a két vásárlás egymástól teljesen független. A megalapozottság-megbízhatóság vizsgálattal az ilyen negatív korreláció nem mutatható ki. ■

6.4. táblázat | A számítógépjáték- és videóeladások tranzakcióit összegző 2 × 2-es kontingenciatáblázat

	játék	$\overline{játék}$	$\sum_{sor}$
videó	4000	3500	7 500
$\overline{videó}$	2000	500	2 500
$\sum_{oszlop}$	6000	4000	10 000

A most vizsgált jelenség motiválja a *korrelációs szabályok* keresését, amelyek valójában korrelációt takarnak. A **korrelációs szabály** $\{i_1, i_2, \ldots, i_m\}$ alakú, ahol az $\{i_1, i_2, \ldots i_m\}$ elemek bekövetkezése korrelált. Legyen a korreláció értéke a (6.22) formulával megadott érték; a χ^2 statisztika használható annak eldöntésére, hogy a korreláció statisztikailag szignifikáns-e (kimutatható-e). A χ^2 statisztikával a negatív implikáció is meghatározható.

A korreláció előnye, hogy *felfelé zárt*. Ez azt jelenti, hogy ha az elemek S halmaza korrelált (azaz S-beli értékek korreláltak), akkor S minden szuperhalmaza is korrelált. Más szavakkal, korrelált értékek halmazához újabb elemeket hozzáadva nem romlik el a már létező korreláció. A χ^2 statisztika a szignifikanciaszinteken szintén felfelé zárt.

Amikor korreláló halmazokat keresünk azért, hogy korrelációs szabályokat tudjunk képezni, akkor a korreláció és a χ^2 statisztika felfelé zárt tulajdonságát ki tudjuk használni. Üres halmazzal indulva, átkutatjuk az elemhalmazok terét (vagy *elemhalmazhálót*), addig bővítve az elemhalmazunkat, amíg **minimális korrelált elemhalmazhoz** (azaz olyanhoz mely már korrelált, de egyetlen részhalmaza sem korrelált) jutunk. Ezek az elemhalmazok **határt** képeznek a hálóban. A zártság miatt ez alatt a határ alatt nincs korrelált elemhalmaz. Mivel a minimális korrelált elemhalmaz minden szuperhalmaza is korrelált, így megállhatunk a felfelé való keresésben. Azt az algoritmust, amely az elemhalmaztérben a fentiekhez hasonló „bolyongások" sorozatát hajtja végre, **véletlen bolyongásos algoritmusnak** nevezzük. Az algoritmust, az érdektelen helyzetek további kihagyása céljából, összevonhatjuk a megalapozottság ellenőrzésével. A véletlen bolyongásos algoritmus adatkockák használatával könnyen elkészíthető. Az itt leírt eljárás nagyon nagy adatbázisokra való adaptálása még nyitott probléma. Másik korlát az is, hogy a χ^2 statisztika kevésbé pontos, ha a kontingenciatáblázat ritka, ez 2 × 2-esnél nagyobb kontingenciatáblázat használatakor félrevezető lehet. A társítási szabályok érdekességének megállapítására javasolt további módszereket az irodalmi összefoglalásában adunk meg.

6.6. | Társítások bányászata megszorításokkal

Az adatbányász eljárás olyan szabályok ezreit fedezheti fel, amelyek közül sok a felhasználó számára érdektelen. A **megszorításokkal történő bányászás** a keresést különböző típusú megszorítások felügyelete alatt hajtja végre. A megszorítások közé a következők tartoznak:

- **Tudás típusú megszorítások** – A keresni kívánt tudás típusát adja meg, például hogy társításokat keresünk.
- **Adatokra vonatkozó megszorítások** – Meghatározza a feladat szempontjából fontos adatok halmazát.
- **Dimenzió-/szintmegszorítások** – Az adatok dimenzióját vagy a felhasználandó fogalomhierarchia szintjeit határozza meg.
- **Érdekességi megszorítások** – A szabályok érdekességének eldöntéséhez, statisztikai módszereken alapuló, küszöbértékeket határoz meg. Ilyenek például a megalapozottsági és megbízhatósági küszöbértékek.
- **Szabálymegszorítások** – Ezek határozzák meg a bányászni kívánt szabályok alakját. Ezek a megszorítások metaszabályokkal (mintaszabályokkal) is kifejezhetők. Megadhatják a szabály feltételi és következmény oldalán előforduló predikátumok minimális és maximális számát, vagy az attribútumok közötti kapcsolatokat, vagy attribútumok értékekeit és/vagy összesítéseket.

A fenti megszorításokat – a 4. fejezetben bemutatott – magas szintű, deklaratív adatbányászati lekérdező nyelveken lehet megadni.

A fenti megszorítások közül az első néggyel már e könyv korábbi fejezeteiben találkoztunk. Ebben a részben a bányászó eljárásra vonatkozó *szabálymegszorítások* használatát elemezzük. A megszorításokra alapozott bányászatnak ez a formája a felhasználó számára lehetővé teszi a szándéka szerint bányászandó szabályok meghatározását. Ezzel az adatbányász eljárást sokkal *hatékonyabbá* teszi. Ezen túl a bányászandó kérdést optimalizáló, kifinomult algoritmust használhatjuk a felhasználó által megadott megszorításban rejlő lehetőségek kiaknázására is. Ezzel a bányászó eljárást még *eredményesebbé* tehetjük. A megszorítás alapú bányászat támogatja, előmozdítja a párbeszédes, kutató bányászatot és elemzést. A 6.6.1. alfejezetben a metaszabály-vezérelt bányászatot tanulmányozzuk, ahol a szabályra vonatkozó szintaktikus megszorítást mintaszabállyal adtuk meg. A 6.6.2. alfejezetben további szabálymegszorítások használatát tárgyaljuk, ezek halmaz/részhalmaz kapcsolatokat, a változók kezdeti értékeit és az összesítő függvényeket határozzák meg.

6.6.1. | Társítási szabályok metaszabály-vezérelt bányászata

Mire használhatók a metaszabályok? A metaszabályok a felhasználó számára lehetővé teszik, hogy a számára érdekes, bányászandó szabályok alakját megadja. A szabály megadott alakját a bányászás során megszorításként használjuk, és ezzel elősegítjük a bányá-

szó eljárás hatékonyságának növelését. A metaszabályok az elemzőnek (ember) az adatokra vonatkozó tapasztalatain, elvárásain vagy elképzelésein alapulnak, vagy az adatbázisséma alapján automatikusan generálódnak.

6.8. példa | Tegyük fel, hogy az *ElektroMind* cég piacelemző munkatársaként hozzáférünk a vásárlókat leíró adatokhoz (például a vásárló korához, címéhez és hitelképessége adataihoz), valamint a vásárlási tranzakciók listájához. Társításokat szeretnénk keresni a vásárlók jellegzetességei és az általuk vásárolt árucikkek között. Az ezeket a kapcsolatokat leíró összes társítási szabály helyett azonban csak azok érdekelnek bennünket, amelyek azt mutatják meg, hogy a vásárlók mely tulajdonság-párosaiból lehet arra következtetni, hogy vélhetően oktatási szoftvereket fognak vásárolni. A keresni kívánt szabály alakját metaszabályként adhatjuk meg. Az ilyen metaszabályok egy példája:

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow \textit{vásárol}(X, \text{"}\textit{oktatási szoftver}\text{"}) \qquad (6.23)$$

ahol P_1 és P_2 **predikátumváltozók,** az adott adatbázis attribútumait szemléltetik, X változó egy vásárlót reprezentál, Y és W a P_1-nek és P_2-nek megfelelő értékeket vehetnek fel. Tipikusan a felhasználó meg szokta adni a P_1-re és P_2-re vonatkozó feltételezett attribútumlistát, ha nem, akkor egy alapértelmezés szerinti halmazt tudunk használni.

A metaszabály általában egy feltételezést, hipotézist fogalmaz meg, és a a felhasználó ennek megerősítését vagy elvetését várja. Az adatbányászrendszer a megadott metaszabálynak megfelelő szabályokat fog keresni. Például a (6.24) szabály **megfelel** a (6.23) metaszabálynak.

$$\textit{életkor}(X, \text{"30...39"}) \wedge \textit{jövedelem}(X, \text{"41 000...60 000"}) \Rightarrow \textit{vásárol}(X, \text{"}\textit{oktatási szoftver}\text{"}) \qquad (6.24)$$

■

Hogyan lehet metaszabályokkal irányítottá tenni a bányászás folyamatát? Vizsgáljuk meg ezt a problémát közelebbről. Tegyük fel, hogy mint a fenti példában is, interdimenzionális társítási szabályokat kívánunk bányászni. A metaszabály a következő alakú mintaszabály:

$$P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r, \qquad (6.25)$$

ahol P_i $(i = 1, 2, \ldots, l)$ és Q_j $(j = 1, 2, \ldots, r)$ mindketten predikátumok, vagy predikátumváltozók. A metaszabályban a predikátumok száma $p = l + r$. A mintának megfelelő interdimenzionális társítási szabályok megkereséséhez:

- meg kell találnunk az összes gyakori p-predikátumhalmazok L_p halmazát;
- meg kell határoznunk az L_p halmaz l-predikátum részhalmazai megalapozottsági számait azért, hogy az L_p-ből levezethető szabályok megbízhatóságát kiszámíthassuk.

Ez a 6.4. alfejezetben leírt többdimenziós társítási szabályok bányászatának tipikus esete. Amint ott láttuk, az adatkockák, az összesített értékek tárolásának képessége folytán, igen alkalmasak többdimenziós társítási szabályok bányászatára. Az OLAP és az adattárházak népszerűségének köszönhetően lehetséges, hogy az adott bányászó feladatnak megfelelő n-dimenziós adatkockák már rendelkezésünkre állnak. Az n szám a predikátumváltozók vizsgálatakor tekintetbe vett attribútumok száma plusz az adott metaszabályban már benne lévő predikátumok száma, továbbá teljesül, hogy $n \geq p$. Az ilyen n-dimenziós kockát tipikusan a 6.17. ábrán láthatóhoz hasonló részkockahálójával reprezentáljuk. Ebben az esetben, L_p megtalálásához, csak a p-dimenziós részkockákat kell feldolgoznunk (a cellaszámlálókat kell összehasonlítanunk a minimális megalapozottsági számmal). Minthogy az l-dimenziós részkockákat már kiszámítottuk, és tartalmazzák az L_p halmaz l-dimenziós predikátum-részhalmazainak számát, így meghívható a szabálygeneráló eljárás, amely az adott metaszabálynak megfelelő erős szabályokat adja vissza. Ezt a megközelítést, minthogy az összes n-dimenziós kocka vizsgálata helyett csak a p-dimenziós és az l-dimenziós részkockákkal foglalkozik, így **korlátozott n-dimenziós kocka keresésének** nevezzük.

Ha a metaszabály-vezérelt bányászathoz a szükséges n-dimenziós adatkocka nem áll rendelkezésünkre, akkor meg kell konstruálnunk és fel kell dolgoznunk. A teljes adatkocka létrehozása helyett elegendő a p-dimenziós és az l-dimenziós részkockákat kiszámítanunk. A kockakonstruáló módszereket a 2. fejezetben tárgyaltuk.

6.6.2. | Kiegészítő szabálymegszorítások által vezérelt bányászás

A szabályokra vonatkozó megszorítások halmaz/részhalmaz viszonyait, a változók kezdeti értékeit és az összesítő függvényeket a felhasználó adhatja meg. E lehetőségek a metaszabály-vezérelt kereséssel együtt, vagy annak alternatívájaként is használhatók. Ebben a fejezetben a szabálymegszorításokat vizsgáljuk meg, abból a szempontból, hogy hogyan használhatók fel a bányászási eljárás hatékonyabbá tételére. Tekintsünk egy példát, melyben szabálymegszorításokat használunk hibrid-dimenziós társítási szabályok bányászatakor.

6.9. példa | Tegyük fel, hogy az *ElektroMind* cég többdimenziós forgalmi adatbázisa a következő relációkból áll:

- *eladás(vásárló_név, árucikk_név, tranzakció_ID)*
- *lakhely(vásárló_név, régió, város)*
- *árucikk(árucikk_név, kategória, ár)*
- *tranzakció(tranzakció_ID, nap, hónap, év)*

ahol a *lakhely*, *árucikk* és *tranzakció* háromdimenziós táblák, az *eladás* három, az adattáblákra mutató kulcsot tartalmaz, ezek rendre a *vásárló_név*, *árucikk_név*, *tranzakció_azonosító*.

A társításbányászó kérdésünk: *Keressük meg azokat az 1999-ben történt eladásokat, melyekben az olcsó áruk (melyek árának összege 100 $-nál kevesebb) eladása elősegíthette ugyanazon kategória valamely drágább árucikkének (melynek ára legalább 500 $) eladását a vancouveri*

vásárlók körében. E kérdést a DMQL adatbányász lekérdező nyelven a következőképp lehet kifejezni (a kérdés sorait a kérdés szöveges kielemzésének elősegítése céljából számoztuk meg):

(1) **mine associations as**
(2) *lakhely*(*C*, _, "*Vancouver*") ∧ *eladás*$^+$(*C*, ?{*I*},{*S*}) ⇒ *eladás*$^+$(*C*, ?{*J*},{*T*})
(3) **from** eladás
(4) **where** S.év = 1999 **and** T.év = 1999 **and** I.kategória = J.kategória
(5) **group by** C, I.kategória
(6) **having** sum(I.ár) ≤ 100 **and** min(J.ár) ≥ 500
(7) **with** support **threshold** = 1%
(8) **with** confidence **threshold** = 50%

Mielőtt a szabálymegszorításokat elemeznénk, nézzük meg közelebbről a fenti kérdést. Az 1. sor tudás típusú megszorítás, megadja, hogy társítási (asszociációs) mintát kívánunk keresni. A 2. sorban adtuk meg a metaszabályt. Ez a következő kevertdimenziós társítási szabály metaszabályának rövidítése (többdimenziós társítási szabály, melyben az ismétlődő predikátum most az *eladás*):

$$\begin{aligned} &\textit{lakhely}(C, \_, \text{"}\textit{Vancouver}\text{"}) \\ &\quad \wedge \textit{eladás}(C, ?I_1, S_1) \wedge \ldots \wedge \textit{eladás}(C, ?I_k, S_k) \wedge I = \{I_1,\ldots,I_k\} \wedge S = \{S_1,\ldots,S_k\} \\ &\quad \Rightarrow \textit{eladás}(C, ?J_1, T_1) \wedge \ldots \wedge \textit{eladás}(C, ?J_m, T_m) \wedge J = \{J_1,\ldots J_m\} \wedge T = \{T_1,\ldots,T_m\} \end{aligned}$$

Ez a szabály azt jelenti, hogy egy vagy több *eladás* az *eladás*(*C*, ?I_1, S_1) ∧ ... ∧ *eladás*(*C*, ?I_k, S_k) formában fordul elő a szabály feltételi (bal) oldalán; a kérdőjel ? azt jelenti, hogy csak az I_1, ..., I_k kerülhet az *árucikk_név* helyébe. Az $I = \{I_1, \ldots, I_k\}$ jelentése az, hogy a szabály feltételi oldalán az összes *I*-k az *I* halmazból valók, s ezenfelül ki kell elégíteniük a 4. sor **where** feltételét, mely hasonló az SQL where feltételéhez. Hasonló megjegyzések érvényesek a szabály következmény (jobb) oldalára nézve is.

A metaszabály az alábbihoz hasonló társítási szabályok generálását teszi lehetővé:

$$\begin{aligned} &\textit{lakhely}(C,\_, \text{"}\textit{Vancouver}\text{"}) \wedge \textit{eladás}(C, \text{"}\textit{Census\_CD}\text{"},\_) \wedge \textit{eladás}(C, \text{"}\textit{MS/Office}\text{"},\_) \\ &\quad \Rightarrow \textit{eladás}(C, \text{"}\textit{MS/SQL server}\text{"}, \_), \qquad [1{,}5\%, 68\%], \end{aligned} \tag{6.26}$$

amely azt jelenti, hogy ha a vancouveri vásárló vásárol "Census_CD"-t és "MS/Office"-t, akkor valószínűsíthető (68%-os valószínűséggel), hogy ez a vásárló vesz "MS/SQL server"-t is, továbbá hogy az összes vásárlók 1,5%-a vásárolja meg mindhárom terméket.

Adatra vonatkozó megszorítást tartalmaz egyrészt a metaszabály *lakhely*(_, _, "*Vancouver*") része (nevezetesen, hogy csak Vancouverben lakókra vonatkozó szabályokat keresünk), másrészt a 3. sor, amely azt az explicit megszorítást eredményezi, hogy csak az *eladás* tábla adatait használjuk. Az ilyen többdimenziós adatbázisban a változókra való hivatkozás egyszerűsíthető. Például az *S.év = 1999* ekvivalens a **from** *eladás S*, *tranzakció R* **where** *S.tranzakció_ID = R.tranzakció_ID* **and** *R.év = 1999* SQL-utasítással. Mindhárom dimenziót (*lakhely, árucikk, tranzakció*) felhasználtuk. Szintmegszorítások a következők: a *lakhely* relációra vonatkozóan: a *vásárló_névvel* csak a *város* = "*Vancouver*" teljesülése ese-

tében foglalkozunk; az *árucikk* relációra vonatkozóan: az *árucikk_név* és *kategória* szintekkel foglalkozunk, mert a kérdésben ezeket használjuk; a *tranzakció* relációra vonatkozóan: csak a *tranzakció_ID*-vel foglalkozunk, mert a *nap* és a *hónap* nem fordulnak elő a kérdésben, az *év*-et is csak a válogatásban használjuk.

Szabálymegszorítások a 4. sor (**where**) és a 6. sor (**having**) több részében is előfordulnak: $S.év = 1999, T.év = 1999, I.kategória = J.kategória, sum(I.ár) \leq 100, min(J.ár) \geq 500$. Végül a 7. és 8. sorok két érdekességi megszorítást (küszöbértékeket) tartalmaznak, nevezetesen a minimális megalapozottságra 1%, a minimális megbízhatóságra 50% értéket határoznak meg. ■

A tudásmegszorításokat és az adatmegszorításokat a bányászat előtt érvényesítjük. A bányászást követően a még fel nem használt megszorításokat a felfedezett szabályok szűrésére használhatjuk. Így a bányászási folyamat esetleg kis hatékonyságúvá és drágává válhat. A dimenzió-/szintmegszorításokat a 6.3.2. alfejezetben már tárgyaltuk, az érdekességi megszorításokkal pedig a fejezet egészében találkozunk. Koncentráljunk most a szabálymegszorításokra.

Milyen típusú megszorításokat használhatunk a bányászási folyamat közben a szabálykeresési tér csökkentésére? Részletesebben, milyen szabálymegszorításokat lehet beépíteni a bányászási folyamat mélyére, biztosítandó a bányász kérdés megválaszolásának teljességét?

A szabálymegszorításokat a gyakori elemhalmazok bányászatának megfelelően a következő öt kategóriába soroljuk: (1) *antimonoton*, (2) *monoton*, (3) *tömörítő*, (4) *konvertálható*, (5) *nem konvertálható*. Valamennyi kategória jellegzetességeit egy-egy példán keresztül mutatjuk be, és megmagyarázzuk, hogy az adott típusú megszorítás hogyan használható a bányászásban.

A megszorítások első kategóriája az *antimonoton*. Vizsgáljuk meg a 6.9. példa megszorító szabályát: $sum(I.ár) \leq 100$. Tegyük fel, hogy az apriori módszert használjuk, amely minden k-adik iterációs lépésben k méretű elemhalmazokkal dolgozik. Ha az elemhalmazban szereplő elemek árainak összege nem kisebb 100-nál, akkor – mivel további elemek hozzávétele az elemhalmazhoz csak tovább növeli annak árösszegét – az elemhalmazt kizárhatjuk a keresési térből. Más szavakkal ha egy elemhalmaz nem elégíti ki ezt a megszorító szabályt, akkor a szuperhalmazainak egyike sem elégítheti ki a megszorítást. Ha a megszorító szabály ilyen tulajdonságnak tesz eleget, akkor **antimonotonnak** nevezzük. A keresési tér antimonoton megszorításokkal való szűkítése minden apriori stílusú iterációs algoritmusban alkalmazható, elősegítve ezáltal a bányászási folyamat általános hatékonyságnövekedését, miközben a bányászás teljessége garantált.

Megjegyezzük, hogy az apriori tulajdonság, mely azt állítja, hogy a gyakori elemhalmazok nem üres részhalmazainak is gyakorinak kell lenniük, szintén antimonoton. Ha adott elemhalmaz nem elégíti ki a minimális megalapozottság követelményét, akkor szuperhalmazai egyike sem fogja. Az Apriori algoritmusban ezt a tulajdonságot használjuk a jelölt elemhalmazok számának iterációs lépésenkénti csökkentésére, ezzel redukálva a társítási szabályok keresési terét.

További példák antimonoton megszorításokra: $min(J.ár) \geq 500$, $count(I) \leq 10$ és így tovább. Minden elemhalmaz, amely megsérti ezen megszorítások valamelyikét, kizárható, mert további elemek hozzávételével keletkező elemhalmaz sosem elégítheti ki a meg-

szorítást. A $avg(I.ár) \leq 100$ megszorítás nem antimonoton. Egy adott elemhalmaz, amely nem elégíti ki ezt a megszorítást, valamely (olcsó) elemmel való bővítést követően, a keletkező halmaz már kielégítheti az adott megszorítást. Így ilyen szabály beépítése a bányászási eljárásba a keresés teljességének elveszítésével jár. Az SQL alapmegszorítási lehetőségeinek jellemzését a 6.5. táblázatban adjuk meg, ennek második oszlopa tartalmazza az antimonotonitásra vonatkozó megjegyzéseket. Az egyszerűség kedvéért csak az $=$, $\in$, $\leq$ és $\subseteq$ operátorokat használjuk.

6.5. táblázat | A gyakran használt SQL alapú megszorítások jellemzése

Megszorítás	Antimonoton	Monoton	Lényegre törő
$v \in S$	Nem	Igen	Igen
$S \supseteq V$	Nem	Igen	Igen
$S \subseteq V$	Igen	Nem	Igen
$min(S) \leq v$	Nem	Igen	Igen
$min(S) \geq v$	Igen	Nem	Igen
$max(S) \leq v$	Igen	Nem	Igen
$max(S) \geq v$	Nem	Igen	Igen
$count(S) \leq v$	Igen	Nem	Gyengén
$count(S) \geq v$	Nem	Igen	Gyengén
$sum(S) \leq v$ $(\forall a \in S, a \geq 0)$	Igen	Nem	Nem
$sum(S) \geq v$ $(\forall a \in S, a \geq 0)$	Nem	Igen	Nem
$range(S) \leq v$	Igen	Nem	Nem
$range(S) \geq v$	Nem	Igen	Nem
$avg(S)\theta\ v$, $\theta \in \{\leq, \geq\}$	Konvertálható	Konvertálható	Nem
$support(S) \geq \xi$	Igen	Nem	Nem
$support(S) \leq \xi$	Nem	Igen	Nem

A megszorítások második kategóriája a *monoton*. Ha a 6.9. példa $sum(I.ár) \geq 100$ megszorítószabályát vizsgáljuk, akkor azt találjuk, hogy a megszorítás alapú feldolgozási módszer egészen más lesz (mint a fentebb tárgyalt). Ha egy I elemhalmaz kielégíti a megszorítást, azaz a halmazban az árak összege nem kisebb mint 100, akkor további elemek I-hez adásával az árak összege csak nőhet, és így mindig ki fogja elégíteni a megszorítást. Ily módon az I elemhalmazon ezen megszorítás további ellenőrzése redundanciához vezet. Más szavakkal, ha egy elemhalmaz kielégíti a vizsgált megszorító szabályt, akkor szuperhalmazai is kielégítik. Ha egy megszorító szabály ennek a tulajdonságnak eleget tesz, akkor azt **monotonnak** nevezzük. Hasonló monoton megszorító szabályok: $min(I.ár) \leq 10$, $count(I) \geq 10$ és így tovább. Az SQL alapmegszorítási lehetőségeit jellemző 6.5. táblázat harmadik oszlopa tartalmazza a monotonitásra vonatkozó megjegyzéseket.

A harmadik kategória a *lényegre törő megszorítások* kategóriája. A megszorítások ezen kategóriája esetén *pontosan (azokat mind, de csak azokat) fel tudjuk sorolni az összes olyan halmazt, mely kielégíti a megszorítást*. Tehát, ha a megszorító szabály **lényegre törő**, akkor közvetlenül, – még a megalapozottság számlálásának kezdete előtt – generálni tudjuk pontosan azokat a halmazokat, amelyek kielégítik a megszorítást. Ezzel a generál-és-tesztel paradigma lényeges költségét tudjuk elkerülni. Más megfogalmazásban az ilyen megszorításokkal előzetesen tudjuk csökkenteni a keresési teret. Például a

6.9. példában előfordult $min(J.ár) \geq 500$ megszorítás lényegre törő. Azért van így, mert expliciten és precízen elő tudjuk állítani azokat az elemhalmazokat, amelyek a megszorító feltételnek eleget tesznek. Konkrétan ezek a halmazok legalább egy olyan elemet kell hogy tartalmazzanak, melynek ára nem kevesebb 500 $-nál. Ezek általános alakja $S_1 \cup S_2$, ahol $S_1 \neq \varnothing$, az olyan elemek halmazának (nem üres) részhalmaza, melyek ára nem kisebb 500 $-nál, S_2, az olyan elemek halmazának (akár üres) részhalmaza, melyek ára alacsonyabb 500 $-nál. Minthogy így pontosan elő tudjuk állítani a lényegre törő megszorításnak eleget tevő összes halmazt, már nem szükséges a bányászási folyamat közben iteratívan ellenőriznünk a megszorító szabály teljesülését. Az SQL alapmegszorítási lehetőségeit jellemző 6.5. táblázatnak a negyedik oszlopa tartalmazza a megszorítás lényegre törő jellegére vonatkozó megjegyzéseket.

A negyedik kategória a **konvertálható megszorítások** kategóriája. Az ilyen megszorítások a fenti három kategória egyikébe sem sorolhatók. Ugyanakkor, ha az elemhalmaz elemeit valamilyen sorrendbe elrendezzük, akkor a megszorítás a gyakori elemhalmazokat bányászó eljárás szempontjából monoton, vagy antimonoton lesz. Például az $avg(I.ár) \leq 100$ szabály sem antimonoton, sem monoton. Ha azonban a tranzakció elemeit az árak növekvő sorrendjében adjuk hozzá az elemhalmazhoz, akkor a megszorítás antimonotonná válik, ugyanis ha egy elemhalmaz már megsérti ezt a megszorítást (tehát az árak átlaga nagyobb 100 $-nál), akkor egyre nagyobb árú elemeket hozzáadva már soha nem fogja tudni kielégíteni a megszorítást. Hasonlóan, ha tranzakció elemeit árcsökkenő sorrendben adogatjuk az elemhalmazhoz, akkor a szabály monotonná válik, ugyanis ha egy elemhalmaz kielégíti a megszorítást (vagyis elemei árának átlaga nem nagyobb 100 $-nál), akkor egyre olcsóbb elemeket az elemhalmazhoz adva, arra mindig teljesülni fog az, hogy az átlagár nem nagyobb 100 $-nál. Amellett, hogy a 6.5. táblázatban csak az $avg(S) \leq v$ és az $avg(S) \geq v$ megszorításoknál szerepel a konvertálható minősítés, sok más konvertálható megszorítás is létezik, ilyenek például a $variance(S) \geq v$, $standard\_deviation(S) \geq v$ és mások.

Megjegyezzük, hogy a fenti meggondolások nem jelentik azt, hogy minden megszorítás konvertálható lenne. Például a $sum(S)\ \theta\ v$, ahol $\theta \in \{\leq, \geq\}$ és S elemeihez bármilyen valós érték tartozhat, nem konvertálható megszorítás. Ezért ez a megszorítások ötödik, **nem konvertálható** kategóriájába tartozik A jó hír az, hogy bár létezik néhány olyan megszorítás, amely nem konvertálható, az egyszerű SQL-kifejezések, még ha SQL-összesítőfüggvények is szerepelnek bennük, többnyire az első négy kategória valamelyikébe tartoznak, melyeket a bányászó eljárásokban hatékonyan tudunk használni.

6.7. | Összefoglalás

- A nagy adathalmazokban felfedezett társítási lehetőségek hasznosíthatók a piaci elemzésekben, a döntéstámogató elemzésekben és az üzleti élet irányításában. Népszerű alkalmazási terület a **vásárlói kosár elemzése**, amely megkeresi a gyakran együtt (vagy egymást követően) megvásárolt termékek csoportjait, és ezzel a vásárlói szándékokat, a vásárlói viselkedést tanulmányozhatjuk. A **társítási szabályok bányászata** először megkeresi a gyakori elemhalmazokat (például A és B elemekből álló elemhalmazokat, amelyek kielégítik a *minimális megalapozottsági küszöb* követelményét, ami az

együttes előfordulások százalékos alsó határa), amelyekből előállítja az **erős**, $A \Rightarrow B$ alakú társítási szabályokat. Ezek a szabályok egyúttal kielégítik a *minimális megbízhatósági küszöbre* vonatkozó előírást is (ez a $P(A|B)$ feltételes valószínűségre előre meghatározott küszöbérték).

- A társítási szabályokat különböző kritériumok szerint kategóriákba oszthatjuk, ilyenek a következők:

  (1) A szabályban előforduló *értékek típusa* alapján osztályozhatjuk Boole és mennyiségi osztályokba. **Boole társítások** diszkrét (kategorikus) objektumok közötti viszonyt mutatnak be. A **mennyiségi társítások** többdimenziós társítások, amelyek a numerikus attribútumokra diszkretizáltan hivatkoznak. A diszkretizáció lehet statikus és dinamikus. A szabályokban természetesen kategorikus attribútumokra is hivatkozhatunk.

  (2) Osztályozhatunk a szabályokban felhasznált adatok *dimenziója* alapján; a társítási szabályok lehetnek **egydimenziósak** és lehetnek **többdimenziósak**. Az egydimenziós társítások egyetlen predikátumot vagy dimenziót tartalmaznak (például a *vásárol* predikátumot), a többdimenziós társítások több (különböző) predikátumot vagy dimenziót használnak fel.

  (3) A társítási szabályokat osztályozhatjuk a szabályban alkalmazott *absztrakciós szintek* alapján. A szabály lehet **egyszintű** vagy **többszintű**. Egyszintű társítási szabályban az elemek vagy predikátumok az absztrakció ugyanazon szintjére tartoznak, a többszintű társítások az absztrakció más-más szintjeire is hivatkozhatnak.

  (4) A társítások bányászatának *különböző kiterjesztései* szerint osztályok. A társítások bányászata kiterjeszthető **korrelációs analízisre**, a **maximális gyakori minták** és a **zárt gyakori elemhalmazok** bányászatára. A korrelációs analízis segít eldönteni, hogy egy szabályt elfogadjunk vagy elutasítsunk. A p elemhalmaz a *maximális gyakori minta*, ha p valódi szuperhalmazainak egyike sem gyakori. A c elemhalmaz *zárt gyakori* elemhalmaz, ha nem létezik c-nek olyan c' valódi szuperhalmaza, hogy minden olyan tranzakció, amely tartalmazza a c egy részmintáját, tartalmazná a c'-t is.

- Az **Apriori algoritmus** hatékony társításiszabály-bányászó algoritmus, a szintenkénti bányászást az apriori tulajdonság használatával bővíti ki. Az apriori tulajdonság szerint: *a gyakori elemhalmazok minden nem üres részhalmazának is gyakorinak kell lennie*. A k-adik iterációs lépésben ($k > 1$-re), a gyakori k-elemhalmazokra alapozottan előállítja a jelölt $(k + 1)$-elemhalmazokat, és az adatbázis ismételt feldolgozásával előállítja az L_{k+1} halmazt, a gyakori $(k + 1)$-elemhalmazok *teljes* halmazát.

  Az Apriori algoritmus tördelő- (hash) módszerekkel vagy tranzakcióredukciós módszerekkel kombinálva hatékonyabbá tehető. Az algoritmus további változatai az adatok felosztását (minden részben külön-külön bányászva, és az eredményeket egyesítve), és a minta használatának módszerét (az adatok részhalmazán történő bányászás módszere) alkalmazzák. Ezekkel a változtatásokkal elérhető, hogy az adatátvizsgálások szükséges száma kettőre vagy egyre is csökkenthető.

- A **gyakori minta bővítésének (FP-bővítés)** módszere a gyakori elemhalmazokat jelöltek előállítása nélkül keresi meg. Az eredeti tranzakciós adatbázis tömörítése céljá-

ból egy nagy tömörítettségű adatstruktúrát *(FP-fát)* alkot meg. Ezt felhasználva, eltérően az apriori típusú módszerek előállít-és-tesztel stratégiájától, a gyakori minták fokozatos bővítésére koncentrál. Elkerüli a jelöltek drága előállítását, s így az algoritmus hatékonyságát növeli.

- **Többszintű társítási szabályok** egy sor stratégiával bányászhatók. Ezek azon alapulnak, hogy a fogalomhierarchia különböző szintjein alkalmazandó minimális megalapozottsági küszöböket hogyan határozzuk meg. Ha **megalapozottság redukálásával** dolgozunk, akkor a következő keresésitér-csökkentő megközelítéseket használjuk: *a többszintű szűrés egyedi elemenként* vagy a *többszintű szűrés k-elemhalmazokkal*. A redundáns (leszármazott) többszintű társítási szabályokat ki tudjuk zárni a felhasználó számára szolgáltatottak közül, ha a megalapozottságuk és megbízhatóságuk a hozzájuk tartozó ősszabály megfelelő értékei közelébe esnek.
- **Többdimenziós társítási szabályok** bányászatának módszereit aszerint kategorizálhatjuk, hogy hogyan kezelik a mennyiségi adatokat. Az első csoportban a mennyiségi adatokat – előre meghatározott fogalomhierarchiára alapozottan – *statikusan diszkretizálják*. E megközelítéshez az adatkockák igen jól alkalmazhatók, hiszen mind az adatkockákra, mind a mennyiségi attribútumokra a fogalomhierarchiát használjuk. A második csoportban a **mennyiségi társítási szabályokat** úgy bányásszuk, hogy a mennyiségi attribútumokat dinamikusan – kosarazásra alapozottan – diszkretizáljuk, majd a „szomszédos" társítási szabályokat csoportképzéssel összevonjuk. A harmadik csoportba a **távolság alapú társítási szabályok** bányászata tartozik. Klaszterezéssel kialakított, és valamilyen jelentéssel bíró intervallumokat használunk a diszkretizációra.
- Nem minden erős társítási szabály érdekes/fontos. **Korrelációs szabályok** bányászatával felfedezhetők a statisztikailag korrelált elemek.
- A **megszorításokkal történő szabálybányászat** lehetővé teszi, hogy a felhasználó metaszabályok (szabályminták) alakjában megadja, hogy a bányászástól milyen típusú eredményeket vár, valamint a bányászatra vonatkozó további megszorításokat is megadhat. Az ilyen bányászatot segíti a DMQL (adatbányász lekérdező nyelv), s nagy kihívás a bányász nyelven megfogalmazott kérdések algoritmikus optimalizálása is. A megszorító szabályokat öt kategóriába soroljuk: **antimonoton**, **monoton**, **lényegre törő**, **konvertálható** és **nem konvertálható**. Az első négy kategóriába tartozó megszorítások a társítások bányászása során a folyamat irányítására használhatók, ezzel sokkal eredményesebbé és hatékonyabbá tehető a bányászás.
- További elemzés és a szakterület ismerete nélkül a társítási szabályokat nem szabad közvetlenül felhasználni előrejelzésre. Nem feltétlenül jelentenek oksági, okozati viszonyt. A társítási szabályok jól használható kiindulópontot adnak a további vizsgálatokhoz, s ezzel az adatelemzés népszerű eszközévé váltak.

6.8. | Feladatok

6.1. Az Apriori algoritmus a részhalmazok megalapozottságára vonatkozó *előzetes ismeretet* használ.

(a) Bizonyítsuk be, hogy a gyakori elemhalmazok összes nem üres részhalmazának is gyakorinak kell lennie!

(b) Bizonyítsuk be, hogy az s elemhalmaz bármely nem üres s' részhalmaza megalapozottságának legalább akkorának kell lennie, mint amekkora s megalapozottsága!

(c) Legyen adott az l gyakori elemhalmaz, és annak s részhalmaza. Bizonyítsuk be, hogy az $s' \Rightarrow (l - s')$ szabály megbízhatósága nem lehet nagyobb mint az $s \Rightarrow (l - s)$ szabály megbízhatósága, ahol s' az s részhalmaza!

(d) Az Apriori algoritmus *felosztásos (particionált)* változata a tranzakciók D adatbázisát n nem átfedő részre osztja szét. Bizonyítsuk be, hogy minden olyan elemhalmaznak, amely gyakori D-ben, gyakorinak kell lennie a felosztott részek legalább egyikében is!

6.2. A 6.2.2. alfejezet leírja annak a módszerét, ahogy a gyakori elemhalmazokból a *társítási szabályok előállíthatók*. Tervezzünk sokkal ügyesebb módszert! Magyarázzuk meg, hogy miért jobb az itt bemutatottnál! (*Javaslat:* vizsgáljuk meg a 6.1. feladat (b) és (c) pontjaiban leírt tulajdonságok felhasználhatóságát az algoritmusunk megtervezésében.)

6.3. Az adatbázisban az alábbi 4 tranzakció van. Legyen $min\_sup = 60\%$, $min\_conf = 80\%$.

TID	dátum	vásárolt_árucikkek
T100	10/15/99	{K, A, D, B}
T200	10/15/99	{D, A, C, E, B}
T300	10/19/99	{C, A, B, E}
T400	10/22/99	{B, A, D}

(a) Az Apriori algoritmussal és az FP-bővítés algoritmussal is keressük meg az összes gyakori elemhalmazokat! Hasonlítsuk össze a két bányászó eljárás hatékonyságát!

(b) Adjuk meg a következő metaszabálynak megfelelő összes *erős* társítási szabályt:

$$\forall x \in tranzakciók,\ vásárol(X, elem_1) \wedge vásárol(X, elem_2) \Rightarrow vásárol(X, elem_3) \quad [s, c]$$

ahol X a vásárlókat reprezentáló változó, $elem_i$ a vásárolt árucikkeket (például A, B stb.) reprezentáló változó, s a szabály megalapozottsága, c a megbízhatósága!

6.4. Az adatbázisban az alábbi 4 tranzakció van.
Legyen $min\_sup = 60\%$, $min\_conf = 80\%$.

vásárló_ID	TID	vásárolt_árucikkek (a cég márkanév-árucikk_kategóriáinak megfelelően)
01	T100	{King's-rák, Sunset-tej, Dairyland-sajt, Best-kenyér}
02	T200	{Best-sajt, Dairyland-tej, Goldenfarm-alma, Tasty-pástétom, Wonder-kenyér}
01	T300	{Westcoast-alma, Dairyland-tej, Wonder-kenyér, Tasty-pástétom}
03	T400	{Wonder-kenyér, Sunset-tej, Dairyland-sajt}

(a) Az *árucikk_kategóriák*-ból (ahol az $elem_i$ lehet például *tej*) állítsuk elő az alábbi mintaszabályt:

$$\forall X \in \textit{tranzakciók}, \textit{vásárol}(X, elem_1) \wedge \textit{vásárol}(X, elem_2) \Rightarrow \textit{vásárol}(X, elem_3) \quad [s, c]$$

adjuk meg a legnagyobb k-ra a gyakori k-elemhalmazokat, és az *összes* olyan, *erős* társítási szabályt (a megalapozottságukkal és megbízhatóságukkal), amely tartalmaz ilyen elemhalmazt!

(b) A *márkanév-árucikk_kategóriák*-ból (ahol az $elem_i$ lehet például *Sunset-tej*) állítsuk elő az alábbi mintaszabályt:

$$\forall X \in \textit{vásárlók}, \textit{vásárol}(X, elem_1) \wedge \textit{vásárol}(X, elem_2) \Rightarrow \textit{vásárol}(X, elem_3)$$

adjuk meg a legnagyobb k-ra a gyakori k-elemhalmazokat! (Megjegyzés: ne írjuk le az összes szabályt.)

6.5. Tegyük fel, hogy egy nagy áruház tranzakciós adatbázisa négy helyre elosztott. A tranzakciók minden komponens-adatbázisban azonos formátumúak, konkrétan: $T_j : \{i_1, \ldots, i_m\}$, ahol T_j a tranzakció azonosítója, $i_k (1 \leq k \leq m)$ a tranzakcióban eladott árucikk azonosítója. Tervezzünk hatékony algoritmust a globális társítási szabályok bányászására (mely a többszintű társítások vizsgálata nélkül dolgozik). Az algoritmust vázlatosan adjuk meg. Olyan algoritmust tervezzünk, amely nem igényli az összes adat egy helyre szállítását, és nem okoz túlzott kommunikációs költségeket.

6.6. Tegyük fel, hogy *DB* egy nagy tranzakciós adatbázis, ennek gyakori elemhalmazait elmentettük. Elemezzük, hogy hogyan lehet hatékonyan bányászni a (globális) társítási szabályokat, ugyanazon minimális megalapozottsági küszöbértékkel, amikor az új tranzakciók ΔDB halmazát *inkrementálisan hozzáadjuk* (*DB*-hez)?

6.7. Tegyük fel, hogy a *Csúcs_Egyetem* hallgatóit leíró adatbázis összesítésével a 6.6. táblázatban látható R relációhoz jutottunk. A fogalomhierarchia a következő:

állapot:	$\{elsőéves, másodéves, harmadéves, végzős\} \in hallgató.$
	$\{M.Sc., M.A., Ph.D.\} \in diplomás.$
szak:	$\{fizika, kémia, matematika\} \in természettudomány.$
	$\{informatikus, mérnök\} \in alkalmazott\_tudomány.$
	$\{francia, filozófia\} \in bölcsészet.$
életkor:	$\{16...20, 21...25\} \in fiatal.$
	$\{26...30, 30\_felett\} \in idős.$
állampolgárság:	$\{Ázsia, Európa, Latin\_Amerika\} \in külföld.$
	$\{Kanada, USA\} \in Észak\_Amerika.$

Legyen a minimális megalapozottsági küszöb 20%, a minimális megbízhatósági küszöb 50% (a szintek mindegyikén).

(a) Rajzoljuk meg az *állapot*, *szak*, *életkor*, *állampolgárság* fogalomhierarchiákat!

(b) Keressük meg R erős, többszintű (de nem szintátlépő) társítási szabályai halmazát, az összes szinten azonos megalapozottsági küszöböt használva, melyek

6.6. táblázat | A 6.7. feladat összesített relációja

szak	állapot	életkor	állampolgárság	tan_átlag	szám
Francia	M.A	30_felett	Kanada	2.8...3.2	3
Informatikus	harmadéves	16...20	Európa	3.2...3.6	29
Fizika	M.S	26...30	Latin_Amerika	3.2...3.6	18
Mérnök	Ph.D	26...30	Ázsia	3.6...4.0	78
Filozófia	Ph.D	26...30	Európa	3.2...3.6	5
Francia	negyedéves	16...20	Kanada	3.2...3.6	40
Kémia	harmadéves	21...25	USA	3.6...4.0	25
Informatikus	negyedéves	16...20	Kanada	3.2...3.6	70
Filozófia	M.S	30_felett	Kanada	3.6...4.0	15
Francia	harmadéves	16...20	USA	2.8...3.2	8
Filozófia	harmadéves	26...30	Kanada	2.8...3.2	9
Filozófia	M.S	26...30	Ázsia	3.2...3.6	9
Francia	harmadéves	16...20	Kanada	3.2...3.6	52
Matematika	negyedéves	16...20	USA	3.6...4.0	32
Informatikus	harmadéves	16...20	Kanada	3.2...3.6	76
Filozófia	Ph.D	26...30	Kanada	3.6...4.0	14
Filozófia	negyedéves	26...30	Kanada	2.8...3.2	19
Francia	Ph.D	30_felett	Kanada	2.8...3.2	1
Mérnök	harmadéves	21...25	Európa	3.2...3.6	71
Matematika	Ph.D	26...30	Latin_Amerika	3.2...3.6	7
Kémia	harmadéves	16...20	USA	3.6...4.0	46
Mérnök	harmadéves	21...25	Kanada	3.2...3.6	96
Francia	M.S	30_felett	Latin_Amerika	3.2...3.6	4
Filozófia	harmadéves	21...25	USA	2.8...3.2	8
Matematika	harmadéves	16...20	Kanada	3.6...4.0	59

megfelelnek az alábbi mintaszabálynak, ahol $P, Q \in \{$*állapot*, *szak*, *életkor*, *állampolgárság*$\}$:

$$\forall S \in R,\ P(S, x) \wedge Q(S, y) \Rightarrow \textit{tan\_átlag}(S, z)\ [s, c]$$

(c) Keressük meg R erős, többszintű (de nem szintátlépő) társítási szabályai halmazát, a *többszintű szűrés egyedi elemenként* módszerrel. Az absztrakció legalacsonyabb szintjén a redukált megalapozottsági küszöb legyen 10%!

6.8. Tervezzük meg és vázoljuk a társítási szabályok **szintosztásos** bányászásának olyan módszerét, melyben minden elem a szintpozíciójával kódolt, és az adatbázis kezdeti átvizsgálása megszámlálja minden elem előfordulásait a *fogalomhierarchia összes szintjén*, ezzel azonosítja a gyakori és nem gyakori elemeket! Hasonlítsuk össze a többszintű társítások ezen módszerrel történő bányászásának költségét az egyszintű társítások bányászásának költségével!

6.9. Mutassuk meg, hogy az olyan H elemhalmaz, – mely a h elemmel együtt, annak $\hat{h}$ ősét is tartalmazza – megalapozottsága megegyezik a $H - \hat{h}$ elemhalmaz megalapozottságával! Tisztázzuk, hogyan lehet ezt a tulajdonságot a társítási szabályok szintátlépő bányászásában felhasználni.

6.10. Amikor szintátlépő társítási szabályokat bányászunk, tegyük fel, hogy a megtalált {*IBM asztali számítógép*, *nyomtató*} elemhalmaz nem elégíti ki a minimális megalapozottság feltételét. Lehet-e ezt az információt a keresési tér csökkentésére használni úgy, hogy az elemhalmaz {*IBM asztali számítógép*, *ff nyomtató*}-hoz hasonló utódait a továbbiakban figyelmen kívül hagyjuk? Adjunk általános szabályt a keresési tér ezen információ felhasználásával történő csökkentésére!

6.11. Tervezzünk a *hibrid-dimenziós* társítási szabályok bányászására alkalmas módszert (többdimenziós társítási szabályok, ismétlődő predikátumokkal)!

6.12. Adjunk rövid példát annak bemutatására, hogy az erős társítási szabályokban előforduló elemek *negatívan korreláltak* is lehetnek!

6.13. A következő kontingenciatáblázat egy szupermarket tranzakciós adataiból keletkezett, ahol a *hotdog* azokra a tranzakciókra vonatkozik, amelyek tartalmaznak hotdogot, a $\overline{hotdog}$ az olyan tranzakciókra vonatkozik, amelyek nem tartalmaznak hotdogot, a *hamburger* azokra a tranzakciókra vonatkozik, amelyek tartalmaznak hamburgert, a $\overline{hamburger}$ az olyan tranzakciókra vonatkozik, amelyek nem tartalmaznak hamburgert.

	hotdog	$\overline{hotdog}$	Σ_{sor}
hamburger	2000	500	2500
$\overline{hamburger}$	1000	1500	2500
Σ_{oszlop}	3000	2000	5000

(a) Tegyük fel, hogy a *hotdog* $\Rightarrow$ *hamburger* társítási szabályt találtuk a bányászás során. Legyen a minimális megalapozottsági küszöb 25%, a minimális megbízhatósági küszöb 50%. Erős-e a fenti társítási szabály?

(b) Az adott adatokra alapozva döntsük el, hogy a *hotdog* eladások függetlenek-e a *hamburger* eladásoktól! Ha nem, akkor milyen típusú korreláció van a kettő között?

6.14. A társítási szabályok bányászására alkalmas módszerekhez hasonló módszerekkel szekvenciális (időtől függő, idősoros) mintákat is lehet bányászni. Tervezzünk hatékony algoritmust mely a tranzakciós adatbázisban **többszintű szekvenciális minták** bányászására alkalmas! Ilyen mintára példa a következő: *„A vásárló, aki PC-t vásárol, három hónapon belül Microsoft szoftvert is fog vásárolni"*, melyből képezhető a szekvenciális minta sokkal finomabb változata, mint például: *„A vásárló, aki Pentium PC-t vásárol, három hónapon belül Microsoft Office szoftvert is fog vásárolni"*.

6.15. Bizonyítsuk be, hogy a gyakori elemhalmazok bányászásának szempontjából, a következő táblázat minden sora helyesen jellemzi a megfelelő megszorító szabályt.

	Megszorító szabály	Antimonoton	Monoton	Lényegre törő
(a)	$V \in S$	nem	igen	igen
(b)	$S \subseteq V$	igen	nem	igen
(c)	$min(S) \leq v$	nem	igen	igen
(d)	$range(S) \leq v$	igen	nem	nem
(e)	$avg(S) \geq v$	konvertálható	konvertálható	nem

6.16. Az áruházban minden árucikk ára nem negatív. Az áruház igazgatója a következő formájú szabályokra kíváncsi: *„egy ingyenes árucikk eredményezheti-e azt, hogy az adott tranzakció teljes forgalma (legalább) 200 $ legyen?"*. Adjuk meg, hogyan lehet ilyen szabályokat *hatékonyan* bányászni!

6.17. Az áruházban minden árucikk ára nem negatív. A következő esetek mindegyikére adjuk meg az adott esetet leíró megszorító szabályt, és röviden ismertessük, hogyan bányászhatók *hatékonyan* az ezeknek megfelelő társítási szabályok! A vásárlás
(a) tartalmazzon legalább egy Nintendo játékot;
(b) olyan árucikkekből álljon, melyek árának összege kevesebb 150 $-nál;
(c) tartalmazzon egy ingyenes elemet és a többiek árának összege legyen legalább 200 $;
(d) (az egy vásárlásban) megvásárolt árucikkek árának átlaga 100 $ és 500 $ közé essen!

6.9. | Irodalom

A társítási szabályok bányászatát először Agrawal, Imielinski és Swami vetették fel [AIS93b]. A 6.2.1. alfejezetben bemutatott Apriori algoritmust Agrawal és Srikant adták meg [AS94]. Az algoritmus változatait, amelyek a keresési tér csökkentésére hasonló heurisztikákat használnak, egymástól függetlenül Mannila, Tiovonen és Verkamo alkották meg [MTV94]. Kapcsolódó és ezeket egyesítő későbbi publikációk Agrawal, Mannila, Srikant, Toivonen és Verkamo munkái [AMS+96]. A társítási szabályok generálásának módszerét Agrawal és Srikant írták le [AS94a]. Az Apriori algoritmus 6.2.3. alfejezetben leírt változatait a következő munkák tartalmazzák. A tördelőtáblák használatát a bányászás hatékonyságának növelésére Park, Chen és Yu tanulmányozták [PCY95a]. A tranzakciókat redukáló módszereket Agrawal és Srikant [AS94b], Han és Fu [HF95], valamint Park, Chen és Yu [PCY95a] írták le. A szétbontásos technikát Savasere, Omiecinski és Navathe ajánlották [SON95]. A mintát használó megközelítést Toivonen elemezte [Toi96]. Az elemhalmaz dinamikus összeszámlálásának megközelítését Brin, Motwani, Ullman és Tsur adták meg [BMUT97]. A társítási szabályok bányászatának vannak különböző kiterjesztései, a szekvenciális minták bányászata (Agrawal és Srikant [AS95]), az epizódkutatás (Mannila, Toivonen és Verkamo [MTV97]), térbeli társítási szabályok bányászata (Koperski és Han [KH95]), ciklikus társítási szabályok bányászata (Özden, Ramaswamy és Silberschatz [ORS98]), negatív társítási szabályok bányászata (Savasere, Omiecinski és Navathe [SON98]), intertranzakcionális társítási szabályok bányászata (Lu, Han és Feng [LHF98]), és a naptári vásárlói kosár elemzése (Ramaswamy, Mahajan és Silberschatz [RMS98]). A maximális minta bányászatát Bayardo írta le [Bay98]. A gyakori közeli elemhalmazok bányászatát Pasquier, Bastide, Taouil és Lakhal javasolták [PBTL99], erre hatékony algoritmust Pei, Han és Mao adtak [PHM00]. A jéghegy kérdések kiszámítását Fang, Shivakumar, Gracia-Molina és mások tanulmányozták [FSGM+98], hatékony jéghegykocka-kiszámítási módszert Beyer és Ramakrishnan fejlesztették ki [BR99]. A gyakori elemhalmazok generálásának depth-first módszerét Agarwal, Aggarwal és Prasad javasolták [AAP00]. A gyakori minták – jelöltek generálása nélküli – megtalálásának módszerét Han, Pei és Yin javasolták [HPY00].

A többszintű társítások bányászatát Han és Fu [HF95], valamint Srikant és Agrawal tanulmányozták [SA95]. Srikant és Agrawal [SA95] munkájukban ugyanezt tanulmányozták a szabályok általánosításának környezetében, valamint a felesleges szabályok eltávolítását szolgáló R-érdekességi mértéket vezették be.

A 6.4.3. alfejezetben említett ARCS-rendszert, mely szabályok osztályozásán alapuló mennyiségi társítási szabályok bányászatára alkalmas Lent, Swami és Widom fejlesztették ki [LSW97]. Mennyiségi társítási szabályoknak x-monoton és egyenes vonalú területeken alapuló bányászatát Fukuda, Morimoto, Morishita és Tokuyama [FMMT96], valamint Yoda, Fukuda, Morimoto és mások [YFM+97] mutatták be. A mennyiségi társítási szabályok nem rács alapú bányászati módszerét, mely a részleges teljesség mérését használja, Srikant és Agrawal javasolták [SA96]. A távolság alapú társítási szabályok bányászatának 6.4.4. alfejezetben leírt megközelítését Miller és Yang javasolták [MY97]. A többdimenziós társítási szabályok bányászatának mennyiségi attribútumok statikus diszkretizációját alkalmazó módszerét, és az adatkockák használatát Kamber, Han és Chiang tanulmányozták [KHC97].

A szabályok statisztikus függetlenségét az adatbányászatban Piatetski-Shapiro tanulmányozták [PS91b]. Az erős társítási szabályok érdekességének/fontosságának problémáját Chen, Han és Yu [CHY96], valamint Brin, Motwani és Silverstein [BMS97], továbbá Aggarwal és Yu [AY99] elemezték. Egy hatékony módszert, mely korrelációkhoz társításokat generál Brin, Motwani és Silverstein adtak [BMS97], ezt a 6.5.2. alfejezeteben röviden összefoglaltuk. A társítási szabályok fontosságának megítélésére a megalapozottság-megbízhatóság módszer alternatíváit Brin, Motwani, Ullman és Tsur [BMUT97] valamint Ahmed, El-Makky és Taha [AEMT00] adták meg. A tranzakciós adatbázisok feletti ok-okozati struktúrák bányászatának problémáját Silverstein, Brin, Motwani és Ullman tanulmányozták [SBMU98].

A metaszabályok használtát az érdekes/fontos szabályok szintaktikus és szemantikus szűrésére Klemettinen, Mannila, Ronkainen és mások [KMR+94] javasolták. A metaszabály-vezérelt bányászatot, ahol a metaszabály a következmény oldalon ad meg a feltételt kielégítő adatokon végrehajtandó tevékenységet (például a Bayes csoportképzés vagy parcellázás), Shen, Ong, Mitbander és Zaniolo javasolták [SOMZ96]. A társítási szabályok metaszabály-vezérelt bányászatának relációs alapú megközelítését Fu és Han tanulmányozták [FH95]. Az adatkockán alapuló megközelítést Kamber, Han és Chiang tanulmányozták [KHC97]. A 6.6.2. alfejezet megszorítás alapú társítási szabály bányászatát Ng, Lakshmanan, Han és Pang [NLHP98], továbbá Lakshmanan, Ng, Han és Pang [LNHP99] valamint Pei és Han [PH00] tanulmányozták. A megszorított, korrelált halmazok bányászatának egy hatékony módszerét Grahne, Lakshmanan és Wang [GLW00] adták meg. A mintaszabályok és predikátum megszorítások, bányászatban való használatára további ötleteket tárgyalnak az [AK93, DT93, HK91, LHC97, ST96, SVA97] anyagok.

Az ebben a fejezetben hivatkozott társításbányászó nyelv a DMQL adatbányász lekérdező nyelv kiterjesztésén alapul, ezt Han, Fu, Wang és mások javasolták [HFW+96], együttműködve az egydimenziós társítási szabályok bányászatára Meo, Psaila és Ceri [MPC96] által javasolt SQL-hez hasonló művelet szellemével. (El)várható, hogy a DMQL későbbi bővítései követni fogják az *OLE DB for Data Mining* Microsoft által javasolt szintaxisát [Mic00].

A bányászott társítási szabályok hatékony utólagos módosítását javasolják Cheung, Han, Ng és Wong [CHNW96]. Az apriori keretekben történő párhuzamos és elosztott társításbányászatot Park, Chen és Yu [PCY95b], Agrawal és Shafer [AS96], valamint Cheung, Han, Ng és mások [CHN+96] tanulmányozták. Másik párhuzamos, társításbányászó módszert, amely az elemhalmaz-csoportosítás módszerét vertikális adatbázis-szerkezet használatával bővíti ki, javasoltak Zaki, Parthasarathy, Ogihara és Li [ZPOL97].

7. FEJEZET

Osztályozás és előrejelzés

Az adatbázisok gazdagok az intelligens üzleti döntéseknél felhasználható rejtett információkban. Az osztályozás és az előrejelzés az adatelemzés kétféle formája, amelyekkel modellek készíthetők az adatok osztályainak leírására, illetve megjósolhatók az adatok jövőbeli változásai. Amíg az *osztályozás* a különböző kategóriák címkéit mondja meg előre, addig az *előrejelzés* folytonos értékű függvényeket modellez. Például kialakítható egy osztályozási modell, ami a banki kölcsönöket osztályozza aszerint, hogy biztonságosnak vagy veszélyesnek ítélhetők-e; ezzel szemben megadható egy előrejelzési modell, amely megjósolja a potenciális számítógép-vásárlók esetében a ráfordítások nagyságát, ha ismerjük jövedelmüket és foglalkozásukat. A gépi tanulás, a szakértői rendszerek, a statisztika, a neurobiológia területén már számos osztályozási és előrejelzési módszert dolgoztak ki a kutatók. A legtöbb algoritmus tipikusan kis mennyiségű adatot feltételez, így csak a memóriát használja. A jelenlegi adatbányászati kutatások ilyen munkákon alapulnak, olyan skálázható osztályozási és előrejelzési technikákat kifejlesztve, amelyek nagy mennyiségű, lemezen tárolt adatot is képesek kezelni. Ezek a módszerek gyakran párhuzamos és megosztott feldolgozást feltételeznek.

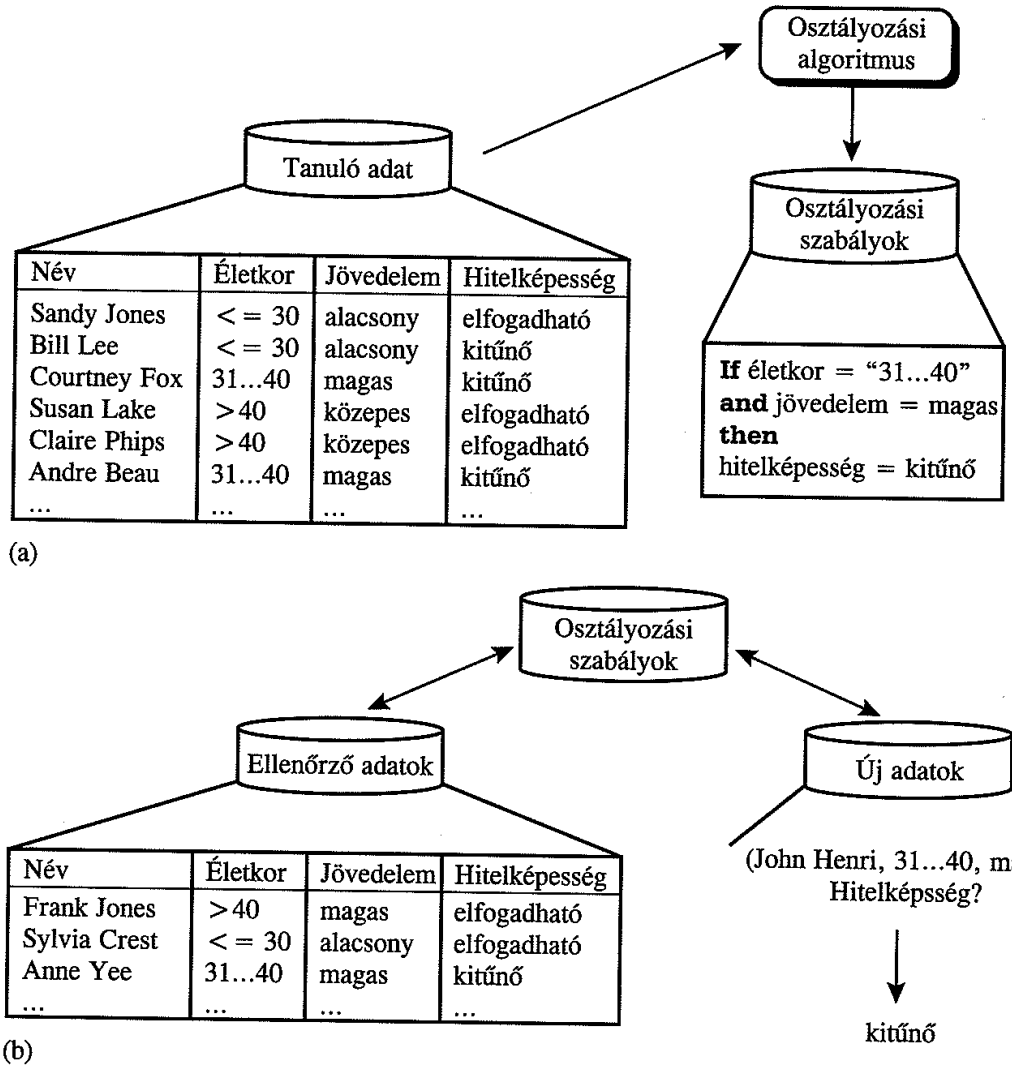
Ebben a fejezetben olyan adatosztályozási alaptechnikák szerepelnek majd, mint például a döntési fa indukció módszere, a Bayes-osztályozás, a Bayes-féle hihetőségi hálók és a neuronhálók. Az adattárházaknál alkalmazott technikáknak az osztályozással való integrálását, valamint a társításon alapuló osztályozását szintén tárgyaljuk majd. Az osztályozás egyéb módszereit is bemutatjuk: a *k*-legközelebbi szomszédon alapuló osztályozást, az eset alapú következtetést, a genetikus algoritmusokat, a közelítő halmazokat és a fuzzy logikát. Az előrejelzési módszereket, lineáris, nemlineáris és általánosított lineáris regressziós technikákat csupán röviden tárgyaljuk. Ahol ez megtehető, ismertetjük ezeknek a technikáknak nagy adatbázisok osztályozási és előrejelzési feladatainál alkalmazható változatait, kiterjesztéseit, optimalizációit.

7.1. | Az osztályozás és az előrejelzés

Az **adatok osztályozása** két lépcsőből álló művelet (7.1. ábra). Először modellt készítünk az adatok osztályain vagy a fogalmakon előre meghatározott halmaz leírására. A modellt az adatbázis attribútumai által megadott sorainak elemzésével építjük fel. Mindegyik sorról feltesszük, hogy egy adott attribútum, az **osztályozási címke attribútum**

által meghatározott, korábban definiált osztályhoz tartozik. Ha osztályozásról beszélünk, az adat sorait szokták még *mintáknak*, *példáknak* vagy *objektumoknak* is nevezni. Azok az adatsorok, amelyeket a modell elkészítésénél végigelemzünk, együttesen alkotják a **tanuló adathalmazt**. Az egyes sorokat, amelyek a tanuló adathalmazt adják, **tanuló mintáknak** nevezzük, és véletlenszerűen választjuk ki őket a mintapopulációból. Mivel minden egyes tanuló minta osztályozási címkéje ismert, ezért ez a lépés a **felügyelt tanulás** néven is ismeretes (azaz, a modell behangolása felügyelet alatt áll abban az értelemben, hogy meg van határozva, hogy az egyes tanuló minták melyik osztályhoz tartoznak). A **felügyelet nélküli tanulás** (vagy **klaszterezés**) esetében a tanuló minták osztályozási címkéje nem ismert, és a megtanulandó osztályok száma vagy halmaza sem feltétlenül ismert előre. A klaszterezés a 8. fejezet témája lesz majd.

A behangolt modell általában osztályozási szabályok, döntési fák vagy matematikai



Név	Életkor	Jövedelem	Hitelképesség
Sandy Jones	< = 30	alacsony	elfogadható
Bill Lee	< = 30	alacsony	kitűnő
Courtney Fox	31...40	magas	kitűnő
Susan Lake	>40	közepes	elfogadható
Claire Phips	>40	közepes	elfogadható
Andre Beau	31...40	magas	kitűnő
...	...	...	...

Név	Életkor	Jövedelem	Hitelképesség
Frank Jones	>40	magas	elfogadható
Sylvia Crest	< = 30	alacsony	elfogadható
Anne Yee	31...40	magas	kitűnő
...	...	...	...

7.1. ábra | Az adatok osztályozásának folyamata: (a) *Tanulás:* a tanuló adatokat egy osztályozási algoritmus elemzi. Jelen esetben a *hitelképesség* az osztályozási címke attribútum, és a behangolt modell vagy osztályozó osztályozási szabályok formájában szerepel. (b) *Osztályozás:* ellenőrző adatokat használunk az osztályozási szabályok pontosságának becslésére. Ha a pontosságot elfogadhatónak találjuk, a szabályokat új sorok osztályozására is használhatjuk

formulák alakjában szerepel. Például adott vásárlók hitelképességét tartalmazó adatbázis esetében olyan osztályozási szabályok taníthatók be, amelyek egy-egy vásárló hitelképességét állapítják meg aszerint, hogy az kiváló vagy elfogadható [7.1.(a) ábra]. A szabályok mind a jövőbeli adatminták kategorizálásánál, mind az adatbázis tartalmának jobb megértésénél használhatók.

A második lépésnél [7.1.(b) ábra] a modellt osztályozásra használjuk. Először a modell (vagy osztályozó) előrejelzési pontosságát becsüljük meg. A 7.9. alfejezet több, az osztályozók pontosságának becslésére alkalmas eljárást is tartalmaz. A **visszatartó módszer** egyszerű eljárás, ami ismert osztályozási címkéjű mintáknak egy **ellenőrző halmazát** használja. Ezeket a mintákat véletlenszerűen választják ki, és függetlenek a tanuló mintáktól. A modell **pontosságát** adott ellenőrző halmaz esetén azoknak az ellenőrző mintáknak az aránya adja, amelyeket a modell helyesen osztályozott. Minden egyes ellenőrző mintánál az ismert osztályozási címke kerül összehasonlításra a behangolt modell által adott eredménnyel. Vegyük észre, hogy ha a modell pontosságát a tanuló halmaz segítségével mérnénk, akkor ez a becslésünk túlzottan optimista lehetne, mivel a behangolt modell hajlamos „túlságosan is jól illeszkedni" az adatokhoz (**túlzott illeszkedés, overfitting**) (vagyis lehet, hogy figyelembe vesz bizonyos, a tanuló adatokra jellemző anomáliákat, amelyek a teljes mintapopulációban már nincsenek jelen). Ezért használják az ellenőrző halmazokat.

Ha a modell pontossága elfogadható, akkor a modell olyan jövőbeli adatsorok vagy objektumok esetében alkalmazható, ahol az osztályozási címke értéke nem ismert. (Az ilyen adatokat a gépi tanulás irodalmában „ismeretlen" vagy „korábban nem látott" adatoknak is nevezik.) Például, a 7.1. ábránál az ismert vásárlók adatainak elemzésével megtanult osztályozási szabályok használhatók új vagy jövőbeli vásárlók hitelképességének becslésére.

Miben különbözik hát az előrejelzés az osztályozástól? Az **előrejelzés** egy olyan modell felépítésének és alkalmazásának tekinthető, amely megbecsüli egy címke nélküli minta osztályát, vagy megbecsüli egy adott minta egy attribútumának várható értékét, vagy értékének várható ingadozási határait. Ebben a megvilágításban az előrejelzési problémák két fő típusa az osztályozás és a regresszió, ahol az osztályozásnál diszkrét vagy nominális értékeket becsülnek, a regresszió esetében pedig folytonos vagy rendezett értékeket. A mi szempontunkból tehát az osztályozási címkék becslésénél használt előrejelzést *osztályozásnak*, míg folytonos értékek becslésénél használt előrejelzést (azaz a regressziós technikák alkalmazását) egyszerűen *előrejelzésnek* fogjuk nevezni. Ez a megközelítés az adatbányászatban általánosan elfogadott.

Az osztályozási és előrejelzési technikákat alkalmazzák többek között hitelek jóváhagyásánál, orvosi diagnózisoknál, teljesítménybecslésnél és a szelektív marketing esetében is.

7.1. példa | Tegyük fel, hogy adott a fogyasztók egy adatbázisa az *ElektroMind* levelezési listán. A levelezési listát az új termékeket bemutató reklámanyagok terjesztésére, illetve a várható árleszállítások beharangozására használják. Az adatbázis a vásárlók következő attribútumait adja meg: *nevük*, *életkoruk*, *jövedelmük*, *foglalkozásuk* és *hitelképességük*. A fogyasztókat aszerint osztályozhatjuk, hogy vásároltak-e már számítógépet az *ElektroMinden* keresztül, vagy nem. Tegyük fel, hogy új fogyasztókat adunk az adatbázishoz, akiket szeretnénk értesíteni a számítógépek nemsokára esedékes árleszállításáról.

Ha az összes új fogyasztónak reklámanyagot küldenénk ki, túlságosan költséges lenne. Tehát költségmegtakarító módszer lehet, ha csupán azokat a fogyasztókat célozzuk meg, akik valószínűleg fognak új számítógépet vásárolni. Erre felépíthető és alkalmazható egy osztályozási modell.

Tegyük fel, hogy ehelyett egy-egy fogyasztónak az *ElektroMinden* keresztül végrehajtott főbb vásárlásainak a számát szeretnénk megbecsülni egy költségvetési éven belül. Mivel itt a becsülendő érték rendezett, előrejelzési modellt készíthetünk erre a célra. ■

7.2. | Témakörök az osztályozásra és előrejelzésre vonatkozóan

Ez a rész az adatok osztályozásra, illetve előrejelzésre való előkészítésével foglalkozó témaköröket mutatja be. Bemutatjuk az osztályozási módszerek összehasonlítására, kiértékelésére vonatkozó kritériumokat is.

7.2.1. | Az adatok előkészítése osztályozásra, illetve előrejelzésre

A következő előfeldolgozási lépések alkalmazhatók annak érdekében, hogy növeljük az osztályozási vagy előrejelzési módszerek pontosságát, hatékonyságát, skálázhatóságát.

- **Adatok tisztítása** – Ez az adatok olyan jellegű feldolgozását jelenti, amelynek során megszüntetjük vagy csökkentjük a *zajt* (például simítási technikákat alkalmazva), vagy kezeljük a *hiányzó értékeket* (például az adott attribútum leggyakoribb értékével vagy a legvalószínűbb értékével helyettesítve ezeket, az utóbbiakat statisztikai módszerekkel határozva meg). Habár a legtöbb osztályozási algoritmus rendelkezik a zajos vagy hiányzó adatok kezelésére alkalmas mechanizmusokkal, ez a lépés segíthet a tanulás folyamán a zavar csökkentésében.
- **Fontossági elemzés** – Sok attribútum *irreleváns* lehet az osztályozási vagy előrejelzési feladat szempontjából. Például a hét azon napjának számontartása, melynek során a banki kölcsönt folyósították, valószínűleg lényegtelen a folyósítás sikerességének szempontjából. Más attribútumok pedig *redundánsak* lehetnek. Így a fontossági elemzés arra használható, hogy eltávolítsuk a fölösleges vagy redundáns attribútumokat a tanulási folyamatból. A gépi tanulásnál ezt a lépést a *főbb jellemzők kiválasztásának* nevezik. Ha ezeket az attribútumokat mégsem hagyjuk figyelmen kívül, az lelassíthatja vagy félrevezetheti a hangolási folyamatot.

  Ideális esetben a fontossági elemzés végrehajtási idejének és a főbb jellemzők csökkentett részhalmazán vett tanulás idejének összege kevesebb kell legyen, mint az az idő, amelyet az eredeti jellemzők halmazán vett tanulás igényelne. Így tehát egy ilyen elemzés javíthatja az osztályozás hatékonyságát és skálázhatóságát.
- **Adatátalakítás** – Az adatokat magasabb fogalmi szinteken *általánosíthatjuk*. Erre fogalmi hierarchiák használhatók. Ez különösen hasznos folytonos értékű attribútumoknál. Például a *jövedelem* attribútum numerikus értékei diszkretizálhatók, ha beve-

zetjük az *alacsony*, *közepes*, *magas* értékeket. Hasonlóan, a nominális[1] értékű attribútumok, mint például *utca*, általánosíthatók magasabb fogalmi szintekre, mint például *város*. Mivel az általánosítás összesűríti az eredeti tanuló adatokat, a tanulás során kevesebb bemeneti/kimeneti műveletet kell végrehajtani.

Az adatokat normalizálhatjuk is, különösen neuronhálók vagy távolsági mértékeket használó módszerek tanulásnál való alkalmazásakor. A **normalizáció** során egy attribútum minden értékét átskálázzuk, hogy egy előre megadott kis intervallumba, például –1 és 1 vagy 0 és 1 közé essenek. Így olyan módszereknél, amelyek távolsági mértékeket használnak, elejét vehetjük annak, hogy a kezdetben nagy értékkészletű attribútumok (mondjuk a *jövedelem*) elnyomják a kezdetben kisebb értékkészlettel rendelkező attribútumokat (például a bináris attribútumokat).

Az adattisztítás, a fontossági elemzés és az adatátalakítás eljárásait nagyobb részletességgel a könyv 3. fejezetében tárgyaljuk. A fontossági elemzés az 5. fejezetben is szerepel.

7.2.2. | Az osztályozási módszerek összehasonlítása

Az osztályozási és előrejelzési módszerek a következő kritériumok alapján hasonlíthatók össze és értékelhetők:

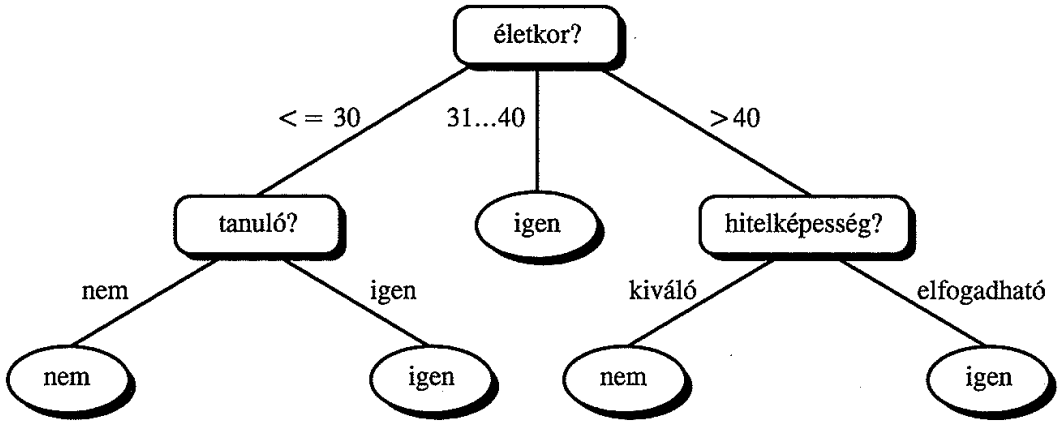
- A **becslési pontosság** a modell azon képességét jelöli, hogy milyen pontosan becsli meg egy új vagy korábban nem látott adat osztályozási címkéjét.
- A **gyorsaság** a modell felépítésének és használatának költségeire vonatkozik.
- A **robusztusság** a modell azon képességét mutatja, hogy mennyire képes pontos becsléseket adni zajos vagy hiányzó adatok esetében.
- A **skálázhatóság** azt mutatja meg, hogy nagy tömegű adat esetében milyen hatékonyan építhető fel a modell.
- Az **értelmezhetőség** a modell nyújtotta érthetőség és betekintés mértékére vonatkozik.

Ezeket a szempontokat tárgyaljuk ebben a fejezetben. Az adatbázis-kutatók közössége az adatbányászat osztályozási és előrejelzési feladatai kapcsán a skálázhatósági szempontot hangsúlyozta, főleg a döntési fa indukciójának esetében.

7.3. | Osztályozás döntési fa indukció segítségével

Mi a döntési fa? A döntési fa egy fejlődési diagramszerű faszerkezet, amelynél minden *belső csúcs* egy attribútumra vonatkozó ellenőrzést jelöl, minden *ág* az ellenőrzés egy-egy kimenetét reprezentálja, és a *levelek* adják az osztályokat vagy az osztályok eloszlását. A fa legfelső csúcsa a *gyökércsúcs*. Tipikus döntési fa látható a 7.2 ábrán, ami a *vesz_számítógépet* fogalmát reprezentálja, vagyis azt becsli meg, hogy egy *ElektroMind-*

1 | A nominális értékű attribútumok értékei névszók lehetnek. *(A fordító megjegyzése)*



7.2. ábra | A *vesz_számítógépet* fogalomra vonatkozó döntési fa, amely előrejelzi, hogy egy fogyasztó az *ElektroMindnél* vásárol-e várhatóan számítógépet vagy sem. Mindegyik belső (nem levél) csúcs egy attribútumra vonatkozó ellenőrzést jelöl. A levelek egy-egy osztályt reprezentálnak (vagy a *vesz_számítógépet = igen* vagy a *vesz_számítógépet = nem*)

beli fogyasztó vélhetően vásárol-e számítógépet vagy sem. A belső csúcsokat téglalapokkal jelöltük, míg a levélcsúcsokat ovális okkal.

Ahhoz, hogy egy ismeretlen mintát osztályozhassunk, a minta attribútumainak értékeit ellenőrizzük a döntési fa szerint. Egy a gyökérből valamelyik levélcsúcsba vezető utat járjuk be, amelynek vége tartalmazza a minta osztályának becslését. A döntési fákat könnyen átalakíthatjuk osztályozási szabályokká.

A 7.3.1. alfejezetben a döntési fák tanulásának alapvető algoritmusát magyarázzuk el. Amikor egy döntési fát készítünk, akkor több elágazás is zaj, vagy az átlagostól nagy mértékben eltérő tanuló minta feldolgozásának eredményeképpen jöhet létre. A *fametszés* során megpróbáljuk felderíteni és eltávolítani ezeket az ágakat, hogy az osztályozási pontosság növekedjen a még feldolgozatlan adatoknál. A fametszést a 7.3.2. alfejezetben mutatjuk be. Az osztályozási szabályoknak a döntési fákból való kikövetkeztetését a 7.3.3. alfejezetben tárgyaljuk. Az alap döntési fa algoritmus javításait a 7.3.4. alfejezet magyarázza. A skálázhatósági kérdéseket a döntési fa indukciójának nagy adatbázisokon való alkalmazásánál a 7.3.5. alfejezetben mutatjuk be. A 7.3.6. alfejezet a döntési fa indukciójának az adattárházas lehetőségekbe, például adatkockákba való beillesztését vizsgálja, amelynek során a döntési fáknak több finomsági szinten történő kibányászása is lehetővé válik. A döntési fákat számos alkalmazási területen használják, kezdve a gyógyszergyártásnál, a játékelméleten át az üzleti életig. Ezek alkotják több kereskedelmi szabálykikövetkeztető rendszer magját is.

7.3.1. | Döntési fa indukció

A döntési fa indukciójának alapalgoritmusa egy mohó eljárás, ami felülről lefelé haladva építi fel a döntési fákat rekurzív módon az „oszd meg és uralkodj" elv alapján. Az algoritmus, amelynek összegzése a 7.3. ábrán látható, az ID3-nak, a széles körben ismert döntési fa indukciós algoritmusnak egyik változata. Az algoritmus kiterjesztéseit a 7.3.2–7.3.6. alfejezetekben tárgyaljuk. Az alapvető stratégia a következő:

- A fa eleinte egyetlen csúcsból áll, amely a tanuló mintát reprezentálja (1. lépés).
- Ha a minták ugyanabból az osztályból valók, akkor ez a csúcs levél lesz, aminek címkéje a szóban forgó osztály (2. és 3. lépés).
- Ellenkező esetben az algoritmus egy entrópián alapuló mértéket használ, az *információnyereséget* mint heurisztikát ahhoz, hogy kiválassza azt az attribútumot, amely a legjobban besorolja a mintákat az egyes osztályokba (6. lépés). Ez az attribútum lesz ebben a csúcsban az „ellenőrző" vagy „döntési" attribútum (7. lépés). Az algoritmusnak ennél a változatánál minden attribútum kategorikus, azaz diszkrét értékű. A folytonos értékű attribútumokat diszkretizálni kell.
- Egy-egy elágazást készítünk az ellenőrző attribútum minden egyes ismert értékének megfelelően, és a mintákat ezek szerint osztjuk szét (8–10. lépések).
- Az algoritmus ugyanazt az eljárást alkalmazza rekurzív módon a döntési fa elkészítésénél a minták minden egyes partíciójánál.[2] Ha egy attribútum megjelent egy csúcsnál, akkor azt a csúcsot egyetlen egy leszármazottjánál sem kell már figyelembe venni. (13. lépés).
- A rekurzív módon való részekre osztás csak akkor állhat le, ha a következő feltételek valamelyike teljesül:
  (a) Egy adott csúcsnál az összes minta ugyanahhoz az osztályhoz tartozik (2. és 3. lépés), vagy

Algoritmus: Döntési_fa_készítés. Az adott tanuló adatokból döntési fa felépítése.

Bemenet: A tanuló minták, *minták,* diszkrét értékű attribútumokkal reprezentálva; a vizsgálandó attribútumok halmaza, *attribútumlista*

Kimenet: A döntési fa.

Módszer:

(1) *N* pont létrehozása
(2) **If** *minták* mind ugyanabból *a C* osztályból valók **then**
(3) legyen *C* osztályú *N* levélcsúcs a visszatérési érték;
(4) **if** *attribútumlista* üres **then**
(5) legyen *N* levélcsúcs a visszatérési érték a *minták*-ban szereplő leggyakoribb osztállyal; // többségi szavazás
(6) legyen az *ellenőrző attribútum* az *attribútumlista* attribútumai közül a legnagyobb információnyereséggel rendelkező
(7) legyen *N* címkéje az *ellenőrző attribútum*;
(8) **for each** ellenőrző attribútumbeli ismert a_i értékére // a minták szétosztása
(9) induljon ki *N*-ből egy ág az *ellenőrző attribútum* = a_i feltétellel;
(10) jelölje s_i azon minták halmazát a *minták*-ban, melyekre *ellenőrző attribútum* = a_i; // egy partíció
(11) **if** s_i üres **then**
(12) illesszünk be egy levélcsúcsot *minták*-ban előforduló leggyakoribb osztály címkéjével;
(13) **else** illesszünk be a Döntési_fa_készítés(s_i, *attribútumlista, ellenőrző attribútum*) által visszaadott eredményt

7.3. ábra | Döntési fa tanuló mintákból való indukciójának alapalgoritmusa

2 | Egy halmaz partíciója alatt a halmaz diszjunkt részhalmazokra való felosztását értjük, a részhalmazoknak az uniója a teljes halmazt adja. *(A fordító megjegyzése)*

(b) Nem maradt olyan attribútum, amelynek alapján a mintát tovább bonthatnánk részekre (4. lépés). Ebben az esetben a **többségi szavazás** elvét alkalmazzuk (5. lépés). Ez azt jelenti, hogy az adott csúcsból levélcsúcs lesz, amelynek címkéje a *minták* között leggyakrabban előforduló osztály címkéje lesz. Alternatív lehetőségként a csúcs mintáinak osztályeloszlását is tárolhatjuk.

(c) Nincsenek olyan *minták*, amelyekre az *ellenőrző attribútum* = a_i elágazás feltétele teljesülne (11. lépés). Ekkor levélcsúcsot készítünk a többségi osztály címkéjével.

Attribútumkiválasztó mérték

Az **információnyereség** mértékét a fa csúcsainál az ellenőrző attribútum kiválasztására használják. Az ilyen mértéket *attribútumkiválasztó mértéknek* vagy a *szétvágás jóságát mérő mértéknek* nevezik. Azt az attribútumot választjuk az aktuális csúcs ellenőrző attribútumaként, amely a legnagyobb információnyereséggel bír (vagy a legnagyobb mértékben csökkenti az *entrópiát*). Ez az attribútum minimalizálja a kialakuló partíció mintáinak osztályozásához szükséges információ nagyságát, valamint a legkisebb véletlenszerűséget vagy „szennyezettséget" eredményezi ennél a partíciónál. Egy efféle információelméleti megközelítés minimalizálja egy objektum osztályozásához szükséges ellenőrzések várható számát, és biztosítja, hogy egy egyszerű fát (habár nem feltétlenül a legegyszerűbbet) találunk majd.

Legyen S egy s adatmintát tartalmazó halmaz. Tegyük fel, hogy egy osztályozási címkeattribútum m különböző értéket vehet fel m különböző osztályt definiálva, C_i ($i = 1, \ldots, m$). Legyen s_i az S-beli minták C_i osztályba eső része. Az adott minta osztályozásához szükséges várható információ a következőképpen adható meg:

$$I(s_1,\ldots,s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i)\,, \tag{7.1}$$

ahol p_i annak a valószínűségét jelöli, hogy egy adott minta a C_i osztályba tartozik, és az s_i/s hányadossal becsüljük. Vegyük észre, hogy kettes alapú logaritmust használtunk, mivel az információt bitekben kódoltuk.

Legyen az A attribútumnak v különböző értéke, $\{a_1, a_2,\ldots, a_v\}$. Az A attribútum alapján S v részhalmazra particionálható, $\{S_1, S_2,\ldots, S_v\}$, ahol S_j azon S-beli mintákat tartalmazza, melyeknek a_j az értéke A-n. Ha A-t választanánk ellenőrző attribútumnak (azaz a legjobb attribútumnak a szétvágás szempontjából), akkor ezek a részhalmazok azon csúcsból induló ágaknak felelnének meg, amely az S halmazt tartalmazza. Legyen s_{ij} az S_j részhalmazban a C_i osztályba eső minták száma. Az entrópia, vagy az A alapján történő partíció várható információja a következőképpen adható meg

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j}+\cdots+s_{mj}}{s} I(s_{1j},\ldots,s_{mj})\,. \tag{7.2}$$

Az $\frac{s_{1j}+\cdots+s_{mj}}{s}$ kifejezés a j. részhalmaz súlyaként fogható fel, és a részhalmazban szereplő minták (azaz a_j az értékük az A-n) számát osztjuk el az S-ben szereplő minták

teljes számával. Minél kisebb az entrópia, annál nagyobb a részhalmaz-partíciók tisztasága (annál nagyobb azoknak a mintáknak a száma egy részhalmazon belül, amelyek egy osztályhoz tartoznak). Vegyük észre, hogy adott S_j részhalmaz esetén:

$$I(s_{1j}, s_{2j}, \ldots, s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}), \tag{7.3}$$

ahol $p_{ij} = \frac{s_{ij}}{|S_j|}$, és ez annak a valószínűsége, hogy egy S_j-beli minta a C_i osztályhoz tartozik.

A következő kódolási információt nyernénk, ha A mentén „ágaztatnánk" el:

$$Nyereség(A) = I(s_1, s_2, \ldots, s_m) - E(A). \tag{7.4}$$

Más szavakkal, *Nyereség*(A) az entrópia várható csökkenését adja A attribútum értékeit ismertnek feltételezve.

Az algoritmus minden attribútum esetében kiszámítja az információnyereséget. A legmagasabb információnyereséggel rendelkező attribútumot választjuk az adott S halmaz ellenőrző attribútumának. Létrehozunk egy csúcsot, és ezzel az attribútummal címkézzük, majd az attribútum minden egyes értékéhez egy-egy ágat rendelünk, és a mintákat ezek szerint particionáljuk.

7.2. példa | Döntési fa indukciója. A 7.1. táblázatban egy az *ElektroMind* fogyasztói adatbázisából származó adatsorokból alkotott tanuló halmaz található. (Az adatok forrása [Qui86].) Osztályozási címke attribútumnak, *vesz_számítógépet*, két különböző értéke lehet (tudniillik {*igen, nem*}); ezért két eltérő osztállyal dolgozunk ($m = 2$). C_1 osztály feleljen meg az *igen*-nek, C_2 a *nem*-nek. 9 minta szerepel az *igen* osztályban, 5 a *nem* osztályban. Ahhoz, hogy kiszámíthassuk az információnyereséget az egyes attribútumok esetében, először a (7.1) egyenlőség felhasználásával kiszámítjuk az adott minta osztályozásához várhatóan szükséges információ nagyságát:

$$I(s_1, s_2) = I(9,5) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0{,}940.$$

Ezután ki kell számítanunk az attribútumokhoz tartozó entrópiákat. Kezdjük az életkor attribútummal. Minden eloszlás esetére kiszámítjuk a várható információ nagyságát:

For *életkor* = "<=30":

$s_{11} = 2 \quad s_{21} = 3 \qquad I(s_{11}, s_{21}) = 0{,}971$

For *életkor* = "31...40":

$s_{12} = 4 \quad s_{22} = 0 \qquad I(s_{12}, s_{22}) = 0$

For *életkor* = ">40":

$s_{13} = 3 \quad s_{23} = 2 \qquad I(s_{13}, s_{23}) = 0{,}971$

7.1. táblázat | Az *ElektroMind* fogyasztói adatbázisból származó tanuló adatsorok

RID	életkor	jövedelem	tanuló	hitelképesség	Osztály: vesz_számítógépet
1	<= 30	magas	nem	elfogadható	nem
2	<= 30	magas	nem	kitűnő	nem
3	31...40	magas	nem	elfogadható	igen
4	>40	közepes	nem	elfogadható	igen
5	>40	alacsony	igen	elfogadható	igen
6	>40	alacsony	igen	kitűnő	nem
7	31...40	alacsony	igen	kitűnő	igen
8	<= 30	közepes	nem	elfogadható	nem
9	<= 30	alacsony	igen	elfogadható	igen
10	>40	közepes	igen	elfogadható	igen
11	<= 30	közepes	igen	kitűnő	igen
12	31...40	közepes	nem	kitűnő	igen
13	31...40	magas	igen	elfogadható	igen
14	>40	közepes	nem	kitűnő	nem

A (7.2) egyenlőség felhasználásával az adott minta osztályozásához szükséges információ nagysága a következő, feltéve, hogy a mintát az életkor attribútum alapján particionáltuk:

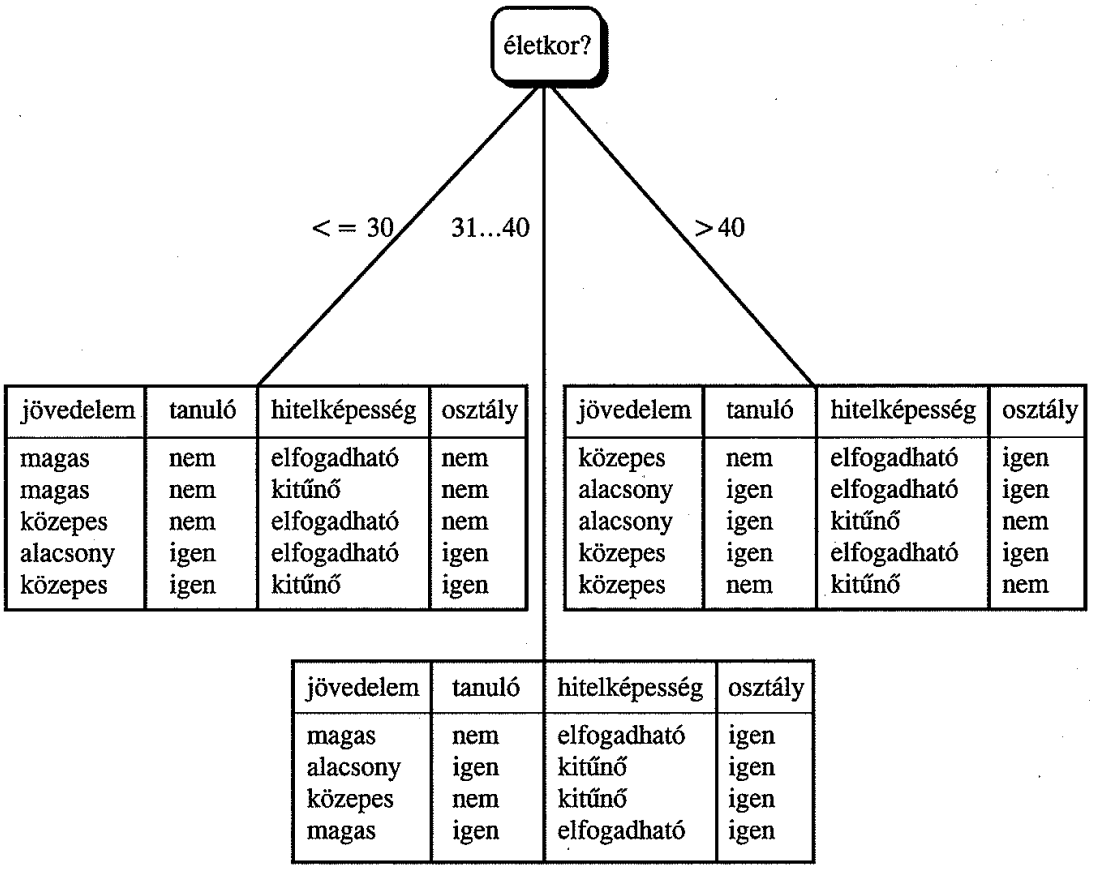
$$I(\text{életkor}) = \frac{5}{14} I(s_{11}, s_{21}) + \frac{4}{14} I(s_{12}, s_{22}) + \frac{5}{14} I(s_{13}, s_{23}) = 0{,}694.$$

Így egy ilyen partíció információnyeresége a következő lenne:

$$\text{Nyereség}(\text{életkor}) = I(s_1, s_2) - E(\text{életkor}) = 0{,}246.$$

Hasonló módon kiszámíthatjuk a következőket: *Nyereség(jövedelem)* = 0,029, *Nyereség(diák)* = 0,151 és *Nyereség(hitelképesség)* = 0,048. Mivel az *életkor* rendelkezik a legmagasabb információnyereséggel az attribútumok között, ezért azt választjuk ellenőrző attribútumnak. Létrehozunk egy csúcsot *életkor* címkével, és az attribútum minden értékéhez egy-egy ágat rendelünk. A mintákat a 7.4. ábrának megfelelő módon particionáljuk. Vegyük észre, hogy azok a minták, amelyek az *életkor* = "31...40" részhalmazba esnek, mind ugyanahhoz az osztályhoz tartoznak. Mivel mindannyian az *igen* osztályból valók, ezért létre kell hoznunk egy levelet ennek az ágnak a végén *igen* címkével. Az algoritmus által felépített végső döntési fa a 7.2. ábrán látható. ■

Összegezve, a döntési fa indukciós algoritmusokat számos alkalmazási területen használják osztályozásra. Az ilyen rendszerek nem használnak az értelmezési tartományra vonatkozó ismereteket. A tanulási és osztályozási lépések a döntési fa indukciójánál általában gyorsan hajthatók végre.



jövedelem	tanuló	hitelképesség	osztály
magas	nem	elfogadható	nem
magas	nem	kitűnő	nem
közepes	nem	elfogadható	nem
alacsony	igen	elfogadható	igen
közepes	igen	kitűnő	igen

jövedelem	tanuló	hitelképesség	osztály
közepes	nem	elfogadható	igen
alacsony	igen	elfogadható	igen
alacsony	igen	kitűnő	nem
közepes	igen	elfogadható	igen
közepes	nem	kitűnő	nem

jövedelem	tanuló	hitelképesség	osztály
magas	nem	elfogadható	igen
alacsony	igen	kitűnő	igen
közepes	nem	kitűnő	igen
magas	igen	elfogadható	igen

7.4. ábra | Az *életkor* attribútum rendelkezik a legmagasabb információnyereséggel, emiatt ez lesz az ellenőrző attribútum a fa gyökerénél. Az *életkor* minden értékének megfelel egy ág. A minták ezek szerint az ágak szerint vannak elrendezve

7.3.2. | Fametszés

Amikor egy döntési fa elkészül, annak több ága is a zaj, illetve az átlagostól nagymértékben eltérő minták által a tanuló adatokban keltett anomáliákat tükrözi. A fanyesési módszerek ezzel a problémával, az adatokhoz való *túlzott illeszkedéssel* foglalkoznak. Az ilyen eljárások általában statisztikai mértékeket használnak a legkevésbé megbízható ágak eltávolításához, gyakran gyorsabb osztályozást és a független ellenőrző adatok pontosabb besorolását eredményezve.

Hogyan működik a fametszés? A fametszésnek két elterjedt megközelítése létezik.

Az **előmetszésnél** a fát úgy metsszük meg, hogy a szerkesztést hamarabb leállítjuk (például úgy döntünk, hogy egy adott csúcsnál nem vágjuk szét, azaz nem particionáljuk tovább a tanuló minták részhalmazát). Az a csúcs lesz a levél, ahol megállunk. A levél tartalmazhatja a részhalmaz mintái közt szereplő leggyakoribb osztályt vagy a minták valószínűségi eloszlását.

Amikor létrehozunk egy fát, az olyan mértékek, mint a statisztikai szignifikancia, a χ^2, az információnyereség és így tovább, használhatók a szétvágás „jóságának" a megbecslésére. Ha a minták partíciója egy csúcsnál olyan szétvágást eredményez, amely egy előre megadott határ alá esik, akkor az adott részhalmazt nem bontjuk tovább. A megfe-

lelő határ megválasztása okozhat nehézségeket. A magas határok túlságosan egyszerű fákat eredményezhetnek, míg az alacsony határok csak kicsiny egyszerűsödéssel járhatnak.

A második megközelítés, az **utómetszés** egy már megszerkesztett fa esetében távolítja el az ágakat. A fa egyik csúcsát úgy metsszük meg, hogy levágjuk az ágait. A *költség bonyolultságán* alapuló metszési algoritmus jó példa az utólagos metszés szerinti megközelítésre. A legalsó le nem nyesett csúcs egy levél, amelynek címkéje a korábbi, a csúcshoz tartozó ágak közt leggyakrabban előforduló osztály. A fa minden egyes, nem levél csúcsára az algoritmus kiszámítja a várható hiba arányát, ami akkor keletkezne, ha a csúcshoz tartozó részfát lemetszenénk. Ezután következik annak a várható hibaaránynak a kiszámítása, amely akkor keletkezik, ha a megfelelő részfát nem vágjuk le; ehhez felhasználjuk az egyes ágak hibaarányait az azokon történő megfigyelések részaránya szerint súlyozva. Ha a csúcs megmetszése nagyobb hibaarányhoz vezet, akkor a részfát megtartjuk, ellenkező esetben levágjuk. Miután létrehoztuk a fokozatosan megmetszett fák egy halmazát, egy független ellenőrző halmazt használunk a fák pontosságának becslésére.

Ahelyett, hogy a fákat a várható hiba arányát véve metszenénk, a kódolásukhoz szükséges bitek száma szerint is metszhetünk. A „legjobban megmetszett fa" az lesz, ami minimalizálja a kódoláshoz szükséges bitek számát. Ez a módszer a *minimális leírási hossz* (MDL, Minimal Description Length) elvét alkalmazza, amely szerint a legegyszerűbb megoldás a legkedvezőbb. A költség bonyolultságán alapuló metszéssel ellentétben, itt nincs szükség a minták egy független halmazára.

További lehetőségként az előmetszést és az utómetszést kombinálhatjuk is. Az utómetszés több számítást igényel, mint az előmetszés, viszont általában megbízhatóbb fákat ad.

7.3.3. | Az osztályozási szabályok kinyerése döntési fákból

Nyerhetünk-e osztályozási szabályokat a döntési fákból és ha igen, hogyan? A döntési fákba foglalt ismereteket kinyerhetjük és megjeleníthetjük IF-THEN formájú osztályozási szabályok segítségével. Minden gyökérből egy adott levélbe vezető úthoz egy szabályt rendelünk. Az adott út mentén szereplő attribútum–érték párok egy konjunkciós láncot[3] alkotnak a szabály előzmény részében (IF rész). A levélcsúcsban szerepel az osztály előrejelzése a szabály következményét alkotva (THEN rész). Az IF-THEN szabályokat az emberek könnyebben megérthetik, különösen, ha a szóban forgó fa nagyon nagy.

7.3. példa | Osztályozási szabályok kinyerése döntési fákból. A 7.2. ábrán szereplő döntési fát követve a gyökérből a fa leveleibe menő utakat osztályozási szabályokká alakíthatjuk át. A 7.2. ábrából kinyert szabályok a következők:

IF *életkor* = "<=30" AND *tanuló* = "*nem*" THEN *vesz_számítógépet* = "*nem*"
IF *életkor* = "<=30" AND *tanuló* = "*igen*" THEN *vesz_számítógépet* = "*igen*"
IF *életkor* = "31...40" THEN *vesz_számítógépet* = "*igen*"

3 | Azaz a szóban forgó ellenőrzéseket logikai ÉS művelettel kapcsoljuk össze. *(A fordító megjegyzése)*

IF *életkor* = "*>40*" AND *hitelképesség* = "*kitűnő*" THEN *vesz_számítógépet* = "*nem*"
IF *életkor* = "*>40*" AND *hitelképesség* = "*elfogadható*" THEN *vesz_számítógépet* = "*igen*"
■

A C4.5, az ID3 algoritmus egy későbbi változata, a tanuló mintákat használja az egyes szabályok pontosságának becslésére. Mivel ez a szabályok pontosságának optimistább közelítését adja, ennek ellensúlyozására pesszimistább becslést alkalmaztak. Alternatívaként az ellenőrző mintaként a tanuló minták halmazától független halmaz is használható.

Egy szabályt úgy „metszhetünk meg", hogy eltávolítunk minden olyan feltételt az előzményből, ami nem javítja a szabály becsült pontosságát. Minden osztálynál az adott osztályhoz tartozó szabályokat rangsorolhatjuk a becsült pontosságuk szerint. Mivel előfordulhat, hogy egy adott ellenőrző minta egyetlen szabály előzményét sem elégíti ki, ezért gyakran az eredményül kapott szabályok halmazát kiegészítik egy szabállyal, amely alapértelmezésben a leggyakoribb osztályt rendeli az adott mintához.

7.3.4. | A döntési fa indukció alapmódszerének javítása

Hogyan javíthatói a döntési fa indukciójának alapmódszere? A 7.3.1. alfejezetben bemutatott döntési fa indukciós algoritmusnak több javítását javasolták. Ebben az alfejezetben számos lényegesebb módosítást is megvizsgálunk, amelyek közül többet a C4.5, az ID3-nál jobb eredményeket adó algoritmus is tartalmaz.

A 7.3.1. alfejezetben a döntési fa indukciós algoritmusánál megköveteltük, hogy minden attribútum kategorizálható vagy diszkretizált legyen. Az algoritmus módosítható úgy, hogy olyan attribútumok is megengedetté váljanak, amelyek diszkrét vagy folytonos értékeknek egy teljes spektrumát tartalmazzák. Egy ilyen A attribútumra vonatkozó ellenőrzés két ágat eredményez az $A \leq V$ és az $A > V$ feltételeknek megfelelően, ahol V az A-nak egy numerikus értéke. A adott v értéke esetén $v - 1$ szétosztást vizsgálhatunk V meghatározásánál. Tipikusan a szomszédos értékek számtani közepét veszik figyelembe. Ha az értékek sorrendben követik egymást, akkor csupán egyszer kell végighaladnunk rajtuk.

Az információnyereségi mérték torzít, mivel azoknak az attribútumoknak a számára kedvező, amelyeknek több értéke van. Több más lehetőséget is javasoltak, például a nyereségrátát, amely figyelembe veszi minden egyes attribútumérték valószínűségét. A kiválasztási mértékeknek több különféle változata létezik még, ilyenek a Gini-index, a χ^2 kontingenciatábla-statisztikák és a G-statisztikák.

Több módszert is kialakítottak a hiányzó attribútumértékek kezelésére. Egy A attribútum hiányzó vagy ismeretlen értékét helyettesíthetjük például az A leggyakoribb értékével. Ehelyett az A attribútum látszólagos információnyereségét is csökkenthetjük azoknak a mintáknak az arányában, amelyeknek A-beli értéke ismeretlen. Ennél a megközelítésnél a hiányzó értékű minta „részleteit" több ág között is szétoszthatjuk. Más módszerek A legvalószínűbb értékét kereshetik meg, vagy az A és más attribútumok közötti ismert összefüggéseket használhatják.

Mivel a döntési fa indukciójánál az adatokat egyre kisebb és kisebb partíciókra osztjuk, szembesülhetünk a *túltöredezettség*, az *ismétlődés* és a *sokszorozódás* problémáival. **Túltöredezettség** esetén egy adott ághoz tartozó minták száma statisztikailag jelentékte-

lenné válik. A probléma egy megoldását jelenheti, ha megengedjük az attribútumértékek kategóriákba való csoportosítását. A fa egy csúcsa azt ellenőrizheti, hogy az attribútum egy értéke az értékeknek egy adott halmazához tartozik-e, $A_i \in \{a_1, a_2, ..., a_n\}$. Másik lehetőség, ha bináris döntési fát készítünk, amelynél minden ág egy bináris logikai ellenőrzést tartalmaz egy attribútumra vonatkozóan. A bináris fák az adat kisebb töredezettségét eredményezik. Néhány, tapasztalatokat összegző tanulmányban azt találták, hogy a bináris fák általában pontosabbak a hagyományos döntési fáknál. **Ismétlődésről** akkor beszélünk, amikor egy attribútumot újra és újra ellenőrzünk a fa egy adott ága mentén. A **sokszorozódásnál** egy fán belül több azonos részfa is szerepel. Ezek a helyzetek csökkenthetik az eredmény pontosságát vagy érthetőségét. Az **attribútum-** (vagy főjellemző-) **szerkesztés** módszere ezt a három problémát igyekszik kiküszöbölni, amellyel az attribútumok egy meghatározott rendszerét javítjuk új, a korábbiakon alapuló attribútumok bevezetésével. A fő jellemzők megszerkesztésének módszerét a 3. fejezetben is tárgyaltuk, mint az adatok átalakításának egyik lehetséges módját.

A döntési fa **növekményes** megközelítésére is érkeztek már javaslatok. Ezeknél az új tanuló minták feldolgozásakor az előző tanuló mintákon alapuló döntési fát szerkesztik át ahelyett, hogy egy teljesen új fát készítenének a semmiből kiindulva.

A döntési fa indukciójának egyéb javításaival, amelyek a skálázhatóságot és az adattárházakhoz kapcsolódó technikákba való beillesztést veszik célba, a 7.3.5. és a 7.3.6. alfejezetben foglalkozunk.

7.3.5. | Skálázhatóság és a döntési fa indukció

Mennyire skálázható a döntési fa indukció? A viszonylag kicsiny adathalmazok esetében a létező döntési fa algoritmusok, úgymint a C4.5 vagy az ID3, hatékonysága elfogadottan jó. A hatékonysági és skálázhatósági kérdések akkor kerülnek az érdeklődés középpontjába, ha ezeket az algoritmusokat nagyon nagy, a való világban megjelenő adatbázisokban végzett adatbányászatra használják. A legtöbb döntési fa algoritmus él azzal a megszorítással, hogy a tanuló mintáknak a fő memóriában kell elhelyezkedniük. Az adatbányászati alkalmazásoknál általában egy-egy tanuló halmaz minták millióit tartalmazhatja. Ezért az előbbi megszorítás behatárolhatja az ilyen algoritmusok skálázhatóságát, így a döntési fa szerkesztése kevésbé hatékonnyá válhat a tanuló mintáknak a fő és a gyorsító memória között történő adatmozgatásának köszönhetően.

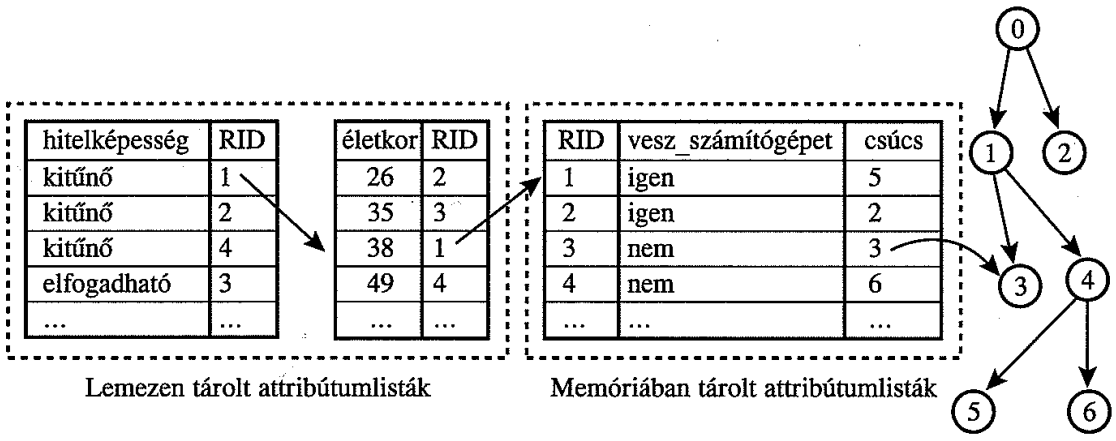
A korai stratégiák, amelyek a döntési fák nagy adatbázisokból való indukciójával foglalkoztak, a csúcsokban diszkretizálták a folytonos attribútumokat és mintavételezték az adatokat. Mindamellett továbbra is éltek azzal a feltételezéssel, hogy a tanuló halmaz befér a memóriába. Egy másik módszer először részhalmazokba osztja az adatokat, amelyek egyenként már befférnek a memóriába, aztán az egyes részhalmazok alapján építi fel a döntési fákat. A végső osztályozó rendszer a részhalmazokból kapott osztályozó rendszereket kombinálja. Habár ennél a megoldásnál a nagy adathalmazok osztályozása is lehetséges, az osztályozási pontosság nem olyan magas, mintha az osztályozó rendszert a teljes adathalmaz felhasználásával készítettük volna el.

Újabb keletű, skálázhatósági kérdéseket figyelembe vevő döntésifa-algoritmusokat javasoltak. A döntési fáknak nagyon nagy tanuló mintából való indukcióját is lehetővé tevő

algoritmusok közé tartozik a SLIQ és a SPRINT, amelyek kategorizált és folytonos értékű attribútumok kezelésére is alkalmasak. Mindkét algoritmus előszortírozó technikákat ajánl az olyan lemezen lévő adathalmazoknál, amelyek nem férnének be a memóriába. Mindkettő új adatszerkezeteket definiál, hogy a fák konstrukcióját megkönnyítse. A SLIQ lemezen tárolt *attribútumlistákat* és egyetlen, memóriában tárolt *osztálylistát* alkalmaz. A 7.2. táblázat mintaadataihoz tartozó SLIQ által megadott attribútumlistákat és osztálylistát a 7.5. ábra mutatja. Mindegyik attribútumhoz kapcsolódik egy attribútumlista, amelyet a *RID* (rekordazonosító) indexel. Minden sort egy kapcsolatlánc, nevezetesen az attribútumlisták egy-egy bejegyzése és az osztályozási lista (amely az adott sor osztályozási címkéjét tartalmazza) egy bejegyzése közti kapcsolódás ábrázol, ahol az osztályozási lista megfelelő bejegyzését még a döntési fa megfelelő csúcsával is összekötik. Az osztályozási lista a memóriában marad, mivel a szerkesztési és metszési fázisoknál gyakran előkerül, illetve módosul. Az osztályozási lista mérete a tanuló halmaz mintáinak gyarapodásával arányosan nő. Amikor az osztályozási lista nem fér el a memóriába, a SLIQ teljesítménye csökken.

7.2. táblázat | A *vesz_számítógépet* osztály mintaadatai

RID	hitelképesség	életkor	vesz_számítógépet
1	kitűnő	38	igen
2	kitűnő	26	igen
3	elfogadható	35	nem
4	kitűnő	49	nem



7.5. ábra | A 7.2. táblázat adatmintáinál a SLIQ attribútum lista és osztály lista szerkezetei

A SPRINT egy másféle *attribútumlista* adatszerkezetet használ, amely az osztályra és *RID*-re vonatkozó információkat a 7.6. ábrán látható módon tárolja. Egy csúcsnál lévő elágazásnál az attribútumlistákat ennek megfelelően particionáljuk és szétosztjuk a gyermekek között. Amikor egy listát particionálunk, a lista rekordjainak sorrendjére is ügyelünk. Így a particionált listáknál már nincs szükség újabb rendezésre. A SPRINT könnyen párhuzamosítható, amely szintén javítja a skálázhatóságot.

Míg a SLIQ és a SPRINT olyan lemezen lévő adathalmazokat kezelnek, amelyek nem

hitelképesség	vesz_számítógépet	RID
kitűnő	igen	1
kitűnő	igen	2
kitűnő	nem	4
elfogadható	nem	3
...	...	...

életkor	vesz_számítógépet	RID
26	igen	2
35	igen	3
38	nem	1
49	nem	4
...	...	...

7.6. ábra | A 7.2. táblázat mintaadatainál az SPRINT attribútumlista szerkezete

férnének be a memóriába, a SLIQ skálázhatóságát behatárolja annak az adatszerkezetnek a használata, aminek be kell férnie a memóriába. A SPRINT esetében megszűnik minden ilyen memóriakorlátozás, de azért olyan tördelőfát (hash tree) használ, amelynek mérete arányos a tanuló halmaz méretével. Ez túlságosan költségessé is válhat, ha a tanuló halmaz mérete növekszik.

A RainForest a döntési fák skálázható indukciójára szolgáló rendszer. A módszer alkalmazkodik az elérhető elsődleges memória méretéhez, és minden döntési fa indukciós algoritmusnál használható. Egy AVC-halmazt (Attribute-Value, Class label, attribútumérték, osztályozási címke) tart fenn, minden attribútumnál jelezve annak osztályok közti eloszlását. A RainForest gyorsulást jelentett a SPRINT-tel szemben.

7.3.6. | Az adattárházas technikák és a döntési fa indukció integrálása

Adatbányászat céljából a döntési fa indukciója integrálható az adattárházas technikákkal. Ebben a részben azt tárgyaljuk, hogy a többdimenziós adatkocka és az attribútumorientált indukció hogyan építhető egybe a döntési fa indukciójával azért, hogy könnyebbé váljon az interaktív többszintű bányászás. Az itt bemutatott technikák általában a tanulás más formáinál is alkalmazhatók.

Az adatkocka-technika egybeszerkeszthető a döntési fa indukciójával, hogy a döntési fák interaktív és többszintű kibányászását kapjuk. A fogalmi hierarchiákban tárolt adatkocka és tudás különböző absztrakciós szinteken teszi lehetővé a döntési fák indukcióját. Mindezen túl, ha egy döntési fa elkészült, a fogalmi hierarchiák segítségével általánosíthatjuk vagy specializálhatjuk a fa egyes csúcsait, lehetővé téve az attribútumok felgörgetését vagy lefúrását és az újonnan megadott absztrakciós szinten az adatok újraosztályozását. Az interaktív jellegnek köszönhetően a felhasználók a fa vagy adat azon területeire irányíthatják a figyelmüket, amelyeket érdekesnek találnak.

Az attribútumorientált indukció (attribute-oriented induction, AOI) az alacsonyabb szintű adatok magasabb szintű fogalmakkal való helyettesítésével, fogalmi hierarchiákat használva általánosítja a tanuló adatokat (5. fejezet). Ha az AOI-t integrálva a döntési fa indukcióval nagyon alacsony (specifikus) fogalmi szinten általánosítunk, egészen nagy és sűrű fákat kaphatunk eredményül. A magas fogalmi szinteken való általánosítás pedig használhatatlan döntési fákat adhat, ahol az érdekes és fontos részfogalmak elvesznek a túlzott általánosításnak köszönhetően. Ehelyett az általánosításnak egy közbülső fogalmi szinten kell megtörténnie, amit a szakterület egy szakértője állíthat be vagy egy felhasználó által definiált küszöb felügyelhet. Így tehát az AOI használata érthetőbb, kisebb és emiatt könnyebben értelmezhető osztályozási fákat eredményezhet, mint a nem általáno-

sított (nagyobb), alacsonyabb szintű adathalmazokon dolgozó módszerek (pédául a SLIQ és a SPRINT) által kapható fák.

A döntési fa generálási eljárás tipikus hibájaként róható fel, hogy a rekurzív partíció eredményeként néhány adatrészhalmaz olyan kicsinnyé válhat, hogy ezek további részekre osztásának nincs statisztikailag szignifikáns alapja. Az ilyen „nem szignifikáns" adathalmazok mérete statisztikailag meghatározható. A probléma megoldásaként megadhatunk egy **kivételi küszöböt**. Ha egy adott részhalmazban a minták száma kevesebb mint a küszöb, a részhalmazt nem particionáljuk tovább. Ehelyett létrehozunk egy levelet, amely tartalmazza részhalmazt és a részhalmaz mintáinak osztályok szerinti eloszlását.

A nagy adatbázisokban az adatok hatalmas tömegének és széles változatosságának köszönhetően nem feltétlenül ésszerű azt feltételezni, hogy minden egyes levél egyetlen osztályba tartozó mintákat tartalmaz majd. A problémát **pontossági** vagy **osztályozási küszöb** alkalmazásával oldhatjuk meg. Egy adott csúcsnál nem particionáljuk tovább az adatok részhalmazát, ha a csúcsban egy adott osztályhoz tartozó minták százalékos aránya túllépi ezt a küszöböt.

Egy adatbányászati lekérdező nyelvet használhatunk a kiterjesztett döntési fa indukció pontos leírására és elindítására. Tegyük fel, hogy adatbányászati feladatként harmincas éveikben járó fogyasztók jövedelmét és foglalkozását figyelembe véve, a hitelezés kockázatosságát szeretnénk megjósolni. Ezt a következő adatbányászati kérdéssel adhatjuk meg:

mine osztályozás
analyze hitelkockázat
in relevance to jövedelem, foglalkozás
from Vevő_ab
where (életkor >= 30) **and** (életkor < 40)
display as szabályok

A fenti DMQL-ben kifejezett kérdés egy relációs lekérdezést hajt végre a *Vevő_ab* adatbázison, hogy összegyűjtse a feladat szempontjából lényeges adatokat. A **where** záradékát nem kielégítő sorokat nem vesszük figyelembe, és csak az **in relevance to** záradékában megadott és az osztályozási címke *(hitelkockázat)* attribútumaihoz tartozó adatokat gyűjtjük egybe. Ezután AOI-t hajtunk végre az adatokon. Mivel a lekérdezés nem adja meg, hogy mely fogalmi hierarchiákat alkalmazzuk, ezért az alapértelmezett hierarchiát használjuk. Egy grafikus felhasználói felület is tervezhető, hogy megkönnyítse a felhasználók számára az ilyen adatbányászati lekérdező nyelveken keresztül megadott adatbányászati feladatok megfogalmazását. Így a felhasználó segíthet az automatikus adatbányászati folyamat irányításában.

7.4. | Bayes-osztályozás

Mik a Bayes-osztályozók? A Bayes-osztályozók statisztikai osztályozók. Egy-egy osztályhoz való tartozásnak a valószínűségét képesek megbecsülni, annak a valószínűségét, hogy egy adott minta egy bizonyos osztályhoz tartozik-e.

A Bayes-osztályozók a később leírásra kerülő Bayes-tételen alapulnak. Osztályozási algoritmusokat összehasonlító tanulmányokban arra a következtetésre jutottak, hogy a *naiv Bayes-osztályozó* néven ismert egyszerű Bayes-osztályozó teljesítménye összehasonlítható a döntési fákon és neuronhálókon alapuló osztályozókéval. A Bayes-osztályozók nagy adatbázisok esetében is nagy pontossággal és sebességgel működtek.

A naiv Bayes-osztályozók feltételezik, hogy egy attribútumértéknek egy adott osztályra gyakorolt hatása független más attribútumok értékeitől. Ezt a feltételezést *feltételes osztályfüggetlenségnek* nevezik. Ez a fellépő számítások egyszerűbbé tételéhez szükséges, és ebből a szempontból naivnak nevezhető. A *Bayes-féle hihetőségi hálók* grafikus modellek, amelyek a naiv Bayes-osztályozóval szemben az attribútumok részhalmazai közti függőségek ábrázolását is lehetővé teszik. A Bayes-féle hihetőségi hálókat szintén használhatjuk osztályozásra.

A 7.4.1. alfejezet végigveszi az alapvető valószínűségi fogalmakat és a Bayes-tételt. Ezután a 7.4.2. alfejezetben a naiv Bayes-osztályozót ismerhetjük meg. A 7.4.3. alfejezetben a Bayes-féle hihetőségi hálókat mutatjuk be.

7.4.1. | Bayes-tétel

Legyen X ismeretlen osztályozási címkéjű adatminta. Legyen H egy olyan hipotézis, amely szerint X egy megadott C osztályhoz tartozik. Az osztályozási feladatokhoz szeretnék meghatározni $P(H|X)$-et, annak a valószínűségét, hogy a H hipotézis igaz, ha X a megfigyelt adatminta.

$P(H|X)$ a H X-et feltételező **utólagos** vagy *a posteriori* **valószínűsége**. Példaként képzeljük el, hogy az adatminták világa a színükkel és alakjukkal megadott gyümölcsökből áll. Tegyük fel, hogy X piros és kerek, és H az a hipotézis, miszerint X alma. Ekkor $P(H|X)$ afelőli bizonyosságunkat mutatja, hogy X alma feltéve, ha láttuk, hogy X piros és kerek. Ezzel szemben $P(H)$ a H **megelőző** vagy *a priori* **valószínűsége**. A példánkban ez annak a valószínűsége, hogy egy adatminta alma, függetlenül attól, hogy néz ki. A $P(H|X)$ posteriori valószínűség több információn alapul (például háttérismereten), mint a $P(H)$ a priori valószínűség, amely független X-től.

Hasonló módon $P(X|H)$ az X-nek H-t feltételező posteriori valószínűsége. Ez annak a valószínűsége, hogy X piros és kerek, ha tudjuk X-ről, hogy alma. $P(X)$ az X a priori valószínűsége. A példánkban ez azt a valószínűséget adja meg, mely szerint egy a gyümölcsök halmazából vett adatminta piros és kerek.

Hogyan becsülhetjük meg ezeket a valószínűségeket? $P(X)$-et, $P(H)$-t, $P(X|H)$-t az adott adatokból is megbecsülhetjük, mint ezt később majd láthatjuk. A Bayes-tétel lehetőséget ad a $P(H|X)$ posteriori valószínűség $P(H)$-ból, $P(X)$-ből és $P(X|H)$-ból való kiszámítására. A Bayes-tétel a következő:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}. \tag{7.5}$$

A következő részben megismerhetjük, miként használja fel a naiv Bayes-osztályozó a Bayes-tételt.

7.4.2. | Naiv Bayes-osztályozó

A naiv Bayes-osztályozó vagy egyszerű Bayes-osztályozó a következőképpen működik:

(1) Mindegyik adatmintát egy $X = (x_1, x_2, \ldots, x_n)$ n dimenziós tulajdonságvektorral jelenítjük meg, amelyek a mintának az $A_1, A_2, \ldots, A_n$ n attribútumon vett értékeit ábrázolják.

(2) Tegyük fel, hogy m osztály létezik: $C_1, C_2, \ldots, C_m$. Adott ismeretlen (azaz még nincs osztályozási címkéje) X adatminta esetén az osztályozó ahhoz az osztályhoz sorolja X-et, amelynek a legnagyobb az X-et feltételező posteriori valószínűsége. Azaz a naiv Bayes-osztályozó akkor és csak akkor rendel egy ismeretlen X mintát a C_i osztályhoz, ha $P(C_i|X) > P(C_j|X)$, $1 \leq j \leq m, j \neq i$ teljesül. Így tehát $P(C_i|X)$-et maximalizáljuk. Azt a C_i osztályt, amelyre $P(C_i|X)$ a legnagyobb, maximális posteriori hipotézisnek nevezzük. A Bayes-tétel alapján [(7.5) egyenlet]:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} . \tag{7.6}$$

(3) Mivel $P(X)$ minden osztály esetében konstans, ezért csupán a $P(X|C_i)P(C_i)$ kifejezést kell maximalizálni. Ha az osztályok *a priori* valószínűségei nem ismertek, akkor általában azzal a feltételezéssel élnek, hogy az osztályok előfordulása ugyanolyan valószínű, azaz $P(C_1) = P(C_2) = \ldots = P(C_m)$, és emiatt elegendő a $P(X|C_i)$-t maximalizálni. Figyeljük meg, hogy az osztályok a priori valószínűségei a $P(C_i) = \frac{s_i}{s}$ kifejezéssel becsülhetők, ahol s_i a C_i osztálybeli tanuló minták száma, míg s az összes minta száma.

(4) Sok attribútummal rendelkező adathalmazok esetén óriási számítási költségekkel járhatna $P(X|C_i)$ kiszámítása. Ahhoz, hogy $P(X|C_i)$ kiértékelésénél csökkentsék a számítási igényt, a **feltételes osztályfüggetlenség** feltételezéssel élnek. Eszerint az attribútumok értékei feltételesen függetlenek egymástól ismerve a minta osztályozási címkéjét, azaz nincsenek függőségi viszonyok az attribútumok között. Így:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) . \tag{7.7}$$

A $P(x_1|C_i)$, $P(x_2|C_i)$, ..., $P(x_n|C_i)$ valószínűségeket a tanuló adatokból becsülhetjük meg, ahol

(a) Ha A_k kategorikus, akkor $P(x_k|C_i) = \frac{s_{ik}}{s_i}$, ahol s_{ik} a C_i-hez tartozó A_k attribútumon x_k értékű tanuló minták száma, s_i pedig a C_i-hez tartozó tanuló minták száma.

(b) Ha A_k folytonos értékű, akkor az attribútumokat tipikusan Gauss-eloszlásúnak feltételezik, így

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x_k-\mu_{C_i})^2}{2\sigma_{C_i}^2}} , \tag{7.8}$$

ahol $g(x_k, \mu_{C_i}, \sigma_{C_i})$ az A_k attribútum **Gauss (normál) sűrűségi függvénye**, míg μ_{C_i}, σ_{C_i} a várható értéket és a szórást jelölik a C_i osztálybeli tanuló minták A_k-n vett értékeire.

(5) Egy ismeretlen X minta osztályozásához minden C_i osztályra kiszámítjuk $P(X|C_i)P(C_i)$-t. Az X minta akkor és csak akkor tartozik a C_i osztályhoz, ha teljesül

$$P(C_i|X) > P(C_j|X),\ 1 \leq j \leq m, j \neq i.$$

Más szavakkal, akkor soroljuk a C_i osztályhoz, ha $P(X|C_i)P(C_i)$ maximális.

Milyen hatékony a Bayes-osztályozó? Elméletileg a többi osztályozóval összehasonlítva a Bayes-osztályozó rendelkezik a legkisebb hibaaránnyal. Mindemellett a gyakorlatban nem mindig ez a helyzet, a használt feltételezések: a feltételes osztályfüggetlenség, pontatlanságának, illetve a valószínűségi adatok hiányának köszönhetően. Mégis több különféle empirikus tanulmányban is úgy találták, hogy bizonyos feladattartományok esetén ez az osztályozó összehasonlítható a döntési fákat vagy neuronhálókat használó osztályozókkal.

A Bayes-osztályozók abból a szempontból is hasznosak, hogy elméletileg alátámasztják azon osztályozók létezésének jogosultságát, amelyek explicit módon nem használják a Bayes-tételt. Például bizonyos feltételek mellett megmutatható, hogy sok neuronhálós vagy görbeilleszkedési algoritmus végeredménye megegyezik a *maximális posteriori* hipotézissel, amit a naiv Bayes-osztályozó ad.

7.4. példa | Az osztályozási címke naiv Bayes-osztályozó használatával való megadása. A 7.2. példában döntési fa indukciójánál használt tanuló adatok mellett szeretnénk megjósolni egy ismeretlen minta osztályozási címkéjét. A tanuló adat a 7.1. táblázatban található. Az adatmintákat az *életkor*, *jövedelem*, *diák*, *hitelképesség* attribútumok határozzák meg. A *vesz_számítógépet* osztályozási címke attribútum két lehetséges értékkel rendelkezik ({*igen*, *nem*}). C_1 feleljen meg a *vesz_számítógépet* = „*igen*" osztálynak, míg C_2 legyen a *vesz_számítógépet* = „*nem*" osztály. Az ismeretlen minta, amit osztályozni szeretnénk, legyen az

X = (*életkor* = "< = 30", *jövedelem* = "*közepes*", *diák* = "*igen*", *hitelképesség* = "*elfogadható*").

Szeretnénk maximalizálni $P(X|C_i)P(C_i)$-t, $i = 1, 2$ esetén. $P(C_i)$-t, az osztályok a priori valószínűségét a tanuló minták alapján kiszámíthatjuk:

P(*vesz_számítógépet* = "*igen*") = 9/14 = 0,643
P(*vesz_számítógépet* = "*nem*") = 5/14 = 0,357

Hogy megadhassuk $P(X|C_i)$-t, $i = 1, 2$ esetén a következő feltételes valószínűségeket számítjuk ki:

P(*életkor* = "<30" | *vesz_számítógépet* = "*igen*") = 2/9 = 0,222
P(*életkor* = "<30" | *vesz_számítógépet* = "*nem*") = 3/5 = 0,600
P(*jövedelem* = "*közepes*" | *vesz_számítógépet* = "*igen*") = 4/9 = 0,444
P(*jövedelem* = "*közepes*" | *vesz_számítógépet* = "*nem*") = 2/5 = 0,400

$P(\text{tanuló} = \text{"igen"} \mid \text{vesz\_számítógépet} = \text{"igen"}) = 6/9 = 0{,}667$
$P(\text{tanuló} = \text{"igen"} \mid \text{vesz\_számítógépet} = \text{"nem"}) = 1/5 = 0{,}200$
$P(\text{hitelképesség} = \text{"elfogadható"} \mid \text{vesz\_számítógépet} = \text{"igen"}) = 6/9 = 0{,}667$
$P(\text{hitelképesség} = \text{"elfogadható"} \mid \text{vesz\_számítógépet} = \text{"nem"}) = 2/5 = 0{,}400$

A fenti valószínűségeket használva kapjuk:

$P(X \mid \text{vesz\_számítógépet} = \text{"igen"}) = 0{,}222 \times 0{,}444 \times 0{,}667 \times 0{,}667 = 0{,}044$
$P(X \mid \text{vesz\_számítógépet} = \text{"nem"}) = 0{,}600 \times 0{,}400 \times 0{,}200 \times 0{,}400 = 0{,}019$
$P(X \mid \text{vesz\_számítógépet} = \text{"igen"})\, P(\text{vesz\_számítógépet} = \text{"igen"}) = 0{,}044 \times 0{,}643 = 0{,}028$
$P(X \mid \text{vesz\_számítógépet} = \text{"nem"})\, P(\text{vesz\_számítógépet} = \text{"nem"}) = 0{,}019 \times 0{,}357 = 0{,}007$

Így a naiv Bayes-osztályozó az X mintára a *vesz_számítógépet* = „*igen*" értéket jósolja. ■

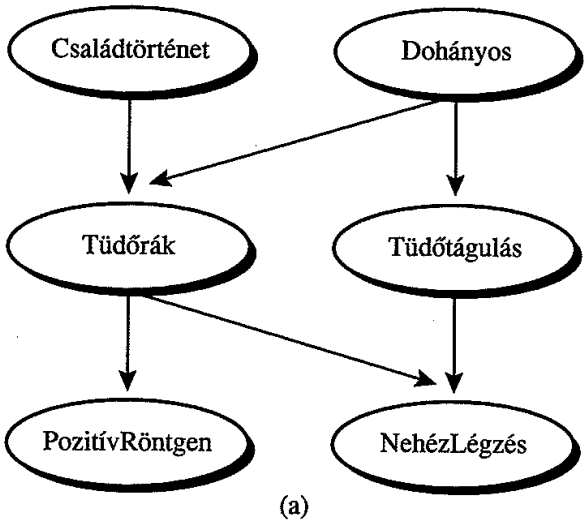
7.4.3. | Bayes-féle hihetőségi hálók

A naiv Bayes-osztályozónál élünk az osztályok feltételes függetlenségének feltételezésével, mely szerint adott osztályhoz tartozó minta esetén az attribútumok értékei feltételesen függetlenek egymástól. Ez a feltételezés egyszerűsíti a számításokat. Ha ez a feltételezés áll, akkor a Bayes-osztályozó a többi osztályozóval összehasonlítva a legpontosabb. A gyakorlatban mindamellett a változók függhetnek egymástól. A **Bayes-féle hihetőségi hálók** együttes feltételes valószínűségi eloszlásokat adnak meg. A változók részhalmazai közt osztályra vonatkozó feltételes függetlenségeket definiálhatunk. Grafikus modellen ábrázolhatjuk a kapcsolatokat, amelyet taníthatunk. Ezek a hálók **hihetőségi hálók**, **Bayes-féle hálók**, illetve **valószínűségi hálók** néven is ismeretesek. A rövidség kedvéért hihetőségi hálóként fogunk hivatkozni rájuk.

Egy hihetőségi hálót két összetevője definiál. Az első egy *irányított körmentes gráf*, ahol mindegyik csúcs egy valószínűségi változónak felel meg, és minden nyíl egy valószínűségi függőséget ábrázol. Ha az Y csúcsból egy nyíl vezet a Z csúcsba, akkor Y a Z **szülője** vagy **közvetlen őse**, és Z az Y **leszármazottja**. Minden változó a szüleire nézve feltételesen független minden olyan csúcstól, amely neki nem leszármazottja. A változók lehetnek diszkrétek vagy folytonosak. Megfelelnek az adatokban lévő valóságos attribútumoknak vagy „rejtett változóknak", amelyek vélhetően kapcsolatokban szerepelnek (például az egészségügyi tünetek orvosi adatok esetében).

A 7.7(a) ábra az [RN95]-ből átvett hat logikai változót tartalmazó egyszerű hihetőségi hálót ábrázol. A nyilak lehetővé teszik az okozati összefüggések megjelenítését. Például a tüdőrák előfordulását befolyásolja mind a személy családjának tüdőrákra vonatkozó története, mind az, hogy a személy dohányos-e vagy sem. Továbbá a nyilakból az is kiderül, hogy a *Tüdőrák* változó a szüleit tekintve (*Családtörténet*, *Dohányos*) feltételesen független a *Tüdőtágulás*-tól. Eszerint ha a *Családtörténet*, *Dohányos* változók értékei ismertek, a *Tüdőtágulás* változó semmilyen további információt nem ad a *Tüdőrák* változóra vonatkozóan.

A hihetőségi háló második összetevője minden változóra tartalmaz egy *feltételes valószínűségi táblázatot* (conditional probability table, CPT). A Z változóhoz tartozó CPT a $P(Z \mid \text{Szülők}(Z))$ feltételes eloszlást adja meg, ahol a *Szülők*(Z) a Z szüleit jelöli. A 7.7(b)



(a)

	CST, D	CST, ~D	~CST, D	~CST, ~D
TR	0,8	0,5	0,7	0,1
~TR	0,2	0,5	0,3	0,9

(b)

7.7. ábra | (a) Egy egyszerű Bayes-féle hihetőségi háló. (b) A *Tüdőrák (TR)* változó összes értékének feltételes valószínűségi táblázata, ami a szülőcsúcsok, *Családtörténet (CST)*, *Dohányzó (D)*, értékeivel vett összes lehetséges kombinációt mutatja. Az ábra az [RN95] munkából származik

ábrán a *Tüdőrák*-ra vonatkozó CPT látható. A *Tüdőrák* minden értékének feltételes valószínűsége a szülei értékeinek összes lehetséges kombinációja mellett szerepel. Például a bal felső sarokban lévő és a jobb alsó sarokban lévő bejegyzésekből sorra a következők láthatók:

$$P(\textit{Tüdőrák} = \text{“}\textit{igen}\text{”} \mid \textit{Családtörténet} = \text{“}\textit{igen}\text{”}, \textit{Dohányos} = \text{“}\textit{igen}\text{”}) = 0{,}8$$
$$P(\textit{Tüdőrák} = \text{“}\textit{nem}\text{”} \mid \textit{Családtörténet} = \text{“}\textit{nem}\text{”}, \textit{Dohányos} = \text{“}\textit{nem}\text{”}) = 0{,}9$$

A $Z_1, Z_2, \ldots, Z_n$ változókhoz, illetve attribútumokhoz tartozó tetszőleges $(z_1, z_2, \ldots, z_n)$ sor együttes valószínűsége a következőképpen számítható ki

$$P(z_1, z_2, \ldots, z_n) = \prod_{i=1}^{n} P(z_i \mid \textit{Szülők}(Z_i)), \tag{7.9}$$

ahol a $P(z_i \mid \textit{Szülők}(Z_i))$ értékei a Z_i-re vonatkozó CPT bejegyzéseknek felelnek meg.

A hálón belül egy csúcsot kimeneti csúcsnak választhatunk, ez felel meg az osztályozási címke attribútumának. Egynél több kimeneti csúcs is szerepelhet. A háló tanításánál következtetési algoritmusokat alkalmazhatunk. Az osztályozási folyamat egyetlen osztálycímke visszaadása helyett az osztályozási címke attribútumának egy valószínűségi eloszlását eredményezheti, megbecsülve minden egyes osztály valószínűségét.

7.4.4. | A Bayes-féle hihetőségi hálók tanítása

Hogyan tanul a Bayes-féle hihetőségi háló? A hihetőségi háló tanulásának, illetve tanításának számos forgatókönyve lehetséges. A háló szerkezete vagy előre adott, vagy az adatokból következtetik ki. A tanuló minták egészénél vagy egy részénél a háló változói vagy *megfigyelhetők*, vagy *rejtettek*. A rejtett adatok esetére *hiányzó értékekként* vagy nem *teljes adatokként* is szoktak hivatkozni.

Ha a háló szerkezete ismert és a változók megfigyelhetők, akkor a háló egyszerűen tanítható. A CPT-bejegyzéseket kell kiszámítani, hasonlóan a naiv Bayes-osztályozónál fellépő valószínűségek kiszámításához.

Amikor adott a háló szerkezete és néhány változó rejtett, a háló tanítására a csökkenő gradiensek módszere használható. A cél a CPT bejegyzéseihez tartozó értékek megtanulása. Jelölje S az s számú $X_1, X_2,...,X_s$ tanuló minta halmazát. Legyen w_{ijk} az $U_i = u_{ik}$ szülőkkel rendelkező $Y_i = y_{ij}$ változó CPT-jének egy bejegyzése. Például, ha w_{ijk} a 7.7(b) ábrán látható CPT-nek a bal felső sarkában lévő bejegyzése, akkor Y_i a *Tüdőrák*-kal azonos, az y_{ij} értéke *"igen"*; az U_i az Y_i szüleit {*Családtörténet, Dohányos*} adja meg, és u_{ik} listázza a szülők értékeit {*"igen", "igen"*}. A w_{ijk}-kat súlyokként kezeljük, hasonlóan a neuronhálók rejtett egységeihez (7.5. alfejezet). A súlyok halmazát együttesen w-vel jelöljük. A súlyokat kezdetben véletlen valószínűségi értékeknek vesszük. A csökkenő gradiens stratégiája mohó hegymászó módszert hajt végre. Minden iterációnál addig módosítjuk a súlyok értékeit, míg végül egy lokális optimumhoz nem konvergálnak.

A módszer az adatokat legjobban modellező w_{ijk} értékeket keresi meg azzal a feltételezéssel, hogy w minden elrendezése azonos valószínűséggel fordul elő. A cél így tehát a $P_w(S) = \prod_{d=1}^{s} P_W(X_d)$ kifejezés maximalizálása. Ez az $ln\ P_w(S)$ gradiensének követésével oldható meg, ami egyszerűsíti a problémát. Adott hálószerkezet és kezdeti értékekkel ellátott w_{ijk} esetén az algoritmus a következőképpen működik:

(1) **Kiszámítjuk a gradienseket** – Minden i, j, k esetén kiszámoljuk a

$$\frac{\partial \ln P_w(S)}{\partial w_{ijk}} = \sum_{d=1}^{s} \frac{P(Y_i = y_{ij}, U_i = u_{ik} \mid X_d)}{w_{ijk}} \tag{7.10}$$

kifejezéseket. A (7.10) egyenlőség jobb oldalán szereplő valószínűséget minden S-beli X_d tanuló minta esetén meg kell adni. A rövidség kedvéért jelöljük ezt a valószínűséget p-vel. Ha az Y_i-nek és U_i-nek megfelelő változók rejtettek egy X_d esetén, akkor a megfelelő p valószínűség a standard Bayes-féle hálókat kikövetkeztető algoritmusok használatával [mint például a kereskedelmi Hugin-programcsomagban találhatókkal *(http://www.hugin.dk)*] az megfigyelhető változókból kiszámítható.

(2) **Kicsit elmozdulunk a gradiens irányába** – A súlyokat a következő kifejezés szerint módosítjuk:

$$w_{ijk} \leftarrow w_{ijk} + (l)\frac{\partial \ln P_w(S)}{\partial w_{ijk}}, \tag{7.11}$$

ahol l a tanulási arány, amely a lépés méretének felel meg, és $\frac{\partial \ln P_w(S)}{\partial w_{ijk}}$ a (7.10) egyenlőség felhasználásával számítható ki. A tanulási arányt egy kis állandónak állítjuk be.

(3) **Újra normalizáljuk a súlyokat** – Mivel a w_{ijk} súlyok valószínűségi értékek, ezért 0 és 1 közé kell esniük, illetve a $\sum_j w_{ijk}$ kifejezésnek 1-gyel kell egyenlőnek lennie minden i és k esetén. Ezt a megkötést a súlyok újra normalizálásával teljesíthetjük, miután módosítottuk őket a (7.11) egyenlőség szerint.

Számos algoritmus létezik, amely az adott megfigyelhető változók mellett a tanuló adatokból megtanulja a háló szerkezetét. A probléma a diszkrét optimalizáció feladatköréhez tartozik. A megoldások megtalálhatók a fejezet végén szereplő hivatkozásokban.

7.5. | Osztályozás hiba-visszaterjesztéssel

Mi jelent a hiba-visszaterjesztés? A hiba-visszaterjesztés egy neuronhálós tanuló algoritmus. A neuronhálókkal eredetileg a pszichológusok és a neurobiológusok kezdtek el foglalkozni, akik a neuronok számítógépes megfelelőit igyekeztek létrehozni és tesztelni. Durván megfogalmazva a **neuronháló** összekötött bemeneti és kimeneti egységek halmaza, ahol minden kapcsolathoz tartozik egy-egy neki megfelelő súly. A tanulási folyamat alatt a neuronháló a súlyok behangolásával tanul, hogy lehetővé váljon a bemeneti minták osztályozási címkéjének a megadása. A neuronhálók tanulására az egységek közötti kapcsolatok miatt *kapcsolati tanulásként* is szoktak hivatkozni.

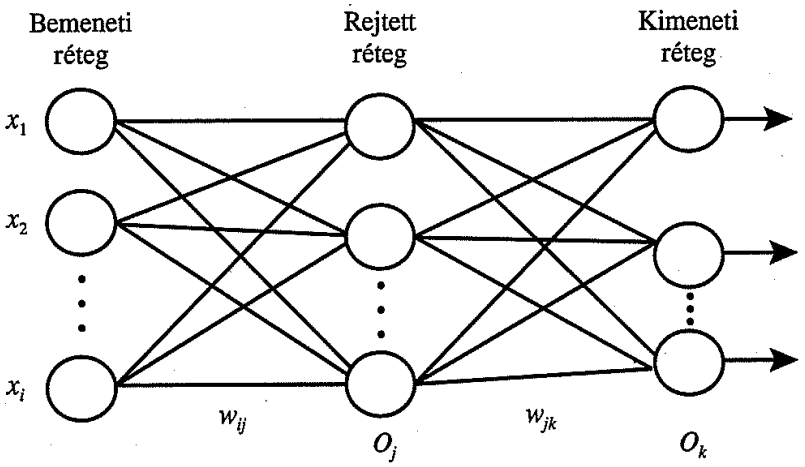
A neuronhálók tanulása hosszú időt vesz igénybe, ezért olyan alkalmazásoknál kedvező, ahol ez megvalósítható. Több paramétert is meg kell adnunk, amelyeket általában tapasztalati úton határozhatunk meg, mint például a háló topológiáját vagy szerkezetét. Sok kritika érte a neuronhálókat, mivel az emberek számára nehezen értelmezhető a megtanult súlyok mögötti szimbolikus jelentés. Ezek a tulajdonságok kezdetben népszerűtlenné tették az adatbányászatban a neuronhálók használatát.

Mindamellett a neuronhálók előnyeihez tartozik, hogy zajos adatokkal is jó eredményt érnek el, illetve hogy olyan minták osztályozására is képesek, amik a tanulásnál nem szerepeltek. Ráadásul mostanában számos algoritmust is kifejlesztettek a behangolt neuronhálókban lévő szabályok kinyerésére. Ezek a tényezők mind alátámasztják a neuronhálók hasznosságát az adatbányászatban.

A legnépszerűbb neuronhálós algoritmus a hiba-visszaterjesztéses algoritmus, amelyet az 1980-as években javasoltak. A 7.5.1. alfejezetben az előrecsatolt többrétegű hálózatokat ismerhetjük meg, ezeken fut a hiba-visszaterjesztéses algoritmus. A 7.5.2. alfejezet tárgyalja a hálók topológiájának definiálását. A hiba-visszaterjesztéses algoritmus a 7.5.3. alfejezetben kerül bemutatásra. A szabályoknak a behangolt hálóból való kikövetkeztetésével a 7.5.4. alfejezet foglalkozik.

7.5.1. | Előrecsatolt többrétegű hálózatok

A hiba-visszaterjesztéses algoritmust az **előrecsatolt többrétegű hálók** tanításánál alkalmazzák. Ilyen hálókra a 7.8. ábrán láthatunk példát. A bemenetek a tanuló mintákhoz vett attribútumoknak felelnek meg. A bemeneteket egyszerre tápláljuk be a **bemeneti réteget** alkotó egységek rétegébe. Ezen egységek súlyozott kimeneteit aztán egyszerre betápláljuk a második, **rejtett réteg** „neuronszerű" egységeibe. A rejtett réteg súlyozott kimenetei ismét bemenetei lehetnek egy másik rejtett rétegnek és így tovább. A rejtett rétegek száma tetszőleges lehet, habár a gyakorlatban általában egyet használnak. Az utolsó rejtett réteg súlyozott kimenetei lesznek a **kimeneti réteget** alkotó egységek bemenetei, s ez adja a háló előrejelzését az adott mintákra.



7.8. ábra | Előrekapcsolt többrétegű hálózat: $X = (x_1, x_2, \ldots, x_i)$ tanuló mintát betápláljuk a bemeneti rétegbe. Mindegyik réteg közt léteznek kapcsolatok, ahol w_{ij} jelöli egy réteg i egysége és a következő réteg j egysége közti súlyt

A rejtett rétegek és a kimeneti réteg egységeit néha **neuronoknak** (a szimbolikus biológiai alapnak köszönhetően) vagy **kimeneti egységeknek** nevezik. A 7.8. ábrán bemutatott többrétegű hálózat a kimeneti egységek két rétegét tartalmazza. Ezért ezt **kétrétegű hálónak** hívjuk. Hasonlóan, a két rejtett réteget tartalmazó hálót *háromrétegű* hálónak mondjuk és így tovább. A háló **előrekapcsolt**, ha a súlyok egyike sem tér vissza a bemeneti rétegbe vagy egy korábbi réteg kimeneti egységébe, illetve **teljesen összekötött**, ha minden egyes egység a következő réteg minden egységének nyújt bemenetet.

A lineáris küszöbfüggvények többrétegű előrekapcsolt neuronhálói elegendő rejtett réteget feltételezve tetszőleges függvényt képesek jól megközelíteni.

7.5.2. | A háló topológiájának definiálása

Hogyan tervezhetem meg egy neuronháló topológiáját? Mielőtt a behangolás elkezdődhetne, a felhasználónak meg kell adnia a bemeneti réteg egységeinek számát, a rejtett rétegek számát (ha több mint egy), minden rejtett réteg egységeinek a számát, a kimeneti réteg egységeinek a számát, és döntenie kell a háló topológiája felől.

A tanító mintákon vett attribútumokhoz tartozó bemeneti értékek normalizálása segít a tanulási szakasz felgyorsításában. Tipikusan úgy normalizálják a bemeneti értékeket, hogy 0 és 1 közé essenek. A diszkrét értékű attribútumokat úgy kódolhatjuk, hogy minden egyes értelmezési tartománybeli értékhez tartozzon egy bemeneti egység. Például, ha az A attribútum értelmezési tartománya $\{a_1, a_2, a_3\}$, akkor három bemeneti egységet rendelhetünk az A megjelenítéséhez. Azaz mondjuk I_0, I_1, I_2 szerepelhetnek bemeneti egységként. Minden egység kezdetben 0 lesz. Ha $A = a_0$, akkor I_0 értéke 1 lesz. Ha $A = a_1$, akkor I_1 értéke lesz 1 és így tovább. Egy kimeneti egységet használhatunk két osztály reprezentálására (ahol az 1 érték jelenti az egyik osztályt, a 0 érték jelenti a másikat). Ha több mint két osztály van, akkor minden egyes osztályhoz tartozik egy kimeneti egység.

A rejtett rétegek „legjobb" kiválasztására nem léteznek világos szabályok. A hálótervezés próbálgatáson alapuló művelet, ami befolyásolhatja a végeredményként létrejött

behangolt háló pontosságát. A súlyok kezdeti értékei szintén hatással lehetnek a pontosságra. Ha egy hálót behangoltak és a pontosságát nem találták elfogadhatónak, akkor gyakran megismétlik a tanítási folyamatot egy eltérő szerkezetű hálón, vagy a kezdeti súlyoknak egy különböző halmazával.

7.5.3. | A hiba-visszaterjesztéses algoritmus

Hogyan működik a hiba-visszaterjesztéses algoritmus? A hiba-visszaterjesztéses algoritmus a tanuló minták halmazának újból és újból való feldolgozásával tanul összehasonlítva a háló becslését minden egyes minta esetében az ismert osztályozási címkével. Minden tanuló mintánál a súlyok úgy módosulnak, hogy a háló becslésének az aktuális osztályozási címkétől vett várható eltérésének a négyzetét minimalizáljuk. Ezeket a módosításokat „hátrafelé" végezzük el, azaz a kimeneti szinttől a rejtett rétegeken át az első rejtett rétegig (innen ered a *hiba-visszaterjesztéses* elnevezés). Habár ezt semmi sem garantálja, általában a súlyok mégis konvergálni fognak, és a tanulási folyamat leáll. Az algoritmust a 7.9. ábrán összegezzük. A következőkben elmagyarázzuk ennek lépéseit.

Hiba-visszaterjesztéses algoritmus. Osztályozás tanulása neuronhálókkal a Hiba-visszaterjesztéses algoritmus használatával.

Bemenet: A tanuló minták, *minták*; a tanulási ráta, *l*, előrecsatolt többrétegű hálózat, *háló*.

Kimenet: A minták osztályozására betanított neuronháló.

Módszer:

(1) Lássunk el minden súlyt és torzítást kezdőértékkel a *háló*-ban.
(2) **while** a leállási feltétel nem teljesül {
(3) **for each** *minták*-beli X tanuló mintára {
(4) // a bemenet előterjesztése
(5) **for each** rejtett vagy kimeneti rétegen levő j egységre {
(6) $I_j = \Sigma_i w_{ij} O_i + \theta_j$; // a j egység *háló* bemenetének kiszámítása az előző i réteg figyelembevételével
(7) $O_j = 1/(1 + e^{-I_j})$; } // a kimenet kiszámítása minden j egységnél
(8) // a hibák visszaterjesztése
(9) **for each** j egységre a kimeneti rétegen
(10) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // hiba kiszámítása
(11) **for each** j egységre a rejtett rétegeken, az utolsótól az első rejtett rétegig
(12) $Err_j = O_j(1 - O_j) \Sigma Err_k w_{jk}$; // a hiba kiszámítása a következő k réteg figyelembevételével
(13) **for each** w_{ij} súlyra a *háló*-ban {
(14) $\Delta w_{ij} = (l) Err_j O_i$; // a súlyok növekményei
(15) $w_{ij} = w_{ij} + \Delta w_{ij}$; } // a súlyok módosítása
(16) **for each** θ_j torzításra a *háló*-ban {
(17) $\Delta \theta_j = (l) Err_j$; // az eltérések növekményei
(18) $\theta_j = \theta_j + \Delta \theta_j$; } // az eltérések módosítása
(19) }}

7.9. ábra | A hiba-visszaterjesztéses algoritmus

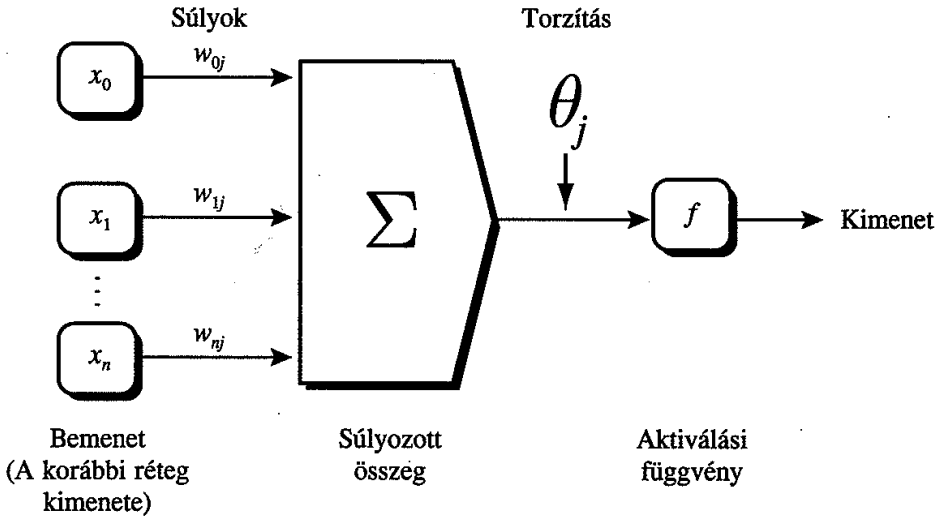
A súlyok inicializálása – A háló súlyait kicsi véletlen számokkal inicializáljuk (például –1 és 1 vagy –0,5 és 0,5 közti számokkal). Minden egységhez tartozik egy *torzítás*, ahogy ezt később elmagyarázzuk. A torzításokat hasonló módon kicsi véletlen számokkal inicializáljuk. Minden X tanuló mintát a következő lépésekben dolgozunk fel:

A bemenet előreterjesztése – Ebben a lépésben minden rejtett és bemeneti rétegbeli egység háló bemenetét és kimenetét számítjuk ki. Először a tanuló mintát betápláljuk a bemeneti rétegbe. A bemeneti réteg j egységénél a bemenet azonos a kimenettel, azaz $O_j = I_j$ a j bemeneti egységnél. A rejtett és a kimeneti rétegek egységeinek nettó bemeneteit a bemenetek lineáris kombinációként számítjuk ki. Ennek illusztrációjára egy rejtett vagy kimeneti rétegbeli egységet mutat be a 7.10. ábra. Az egység bemenetei valójában azon egységek kimenetei is, amelyek az előző réteghez kapcsolják. A nettó bemenet kiszámításához az egységhez kapcsolódó összes bemenetnek vesszük a hozzá tartozó súllyal vett szorzatát, és ezeket összegezzük. Adott j rejtett vagy kimeneti rétegbeli egység esetén a j egység I_j nettó bemenete a következő:

$$I_j = \sum_i w_{ij} O_i + \theta_j \,, \tag{7.12}$$

ahol w_{ij} az előző rétegen lévő i egység és a j egység kapcsolatának a súlya; O_i az előző rétegbeli i egység kimenete; θ_j az egység torzítása. A torzítás egyfajta küszöbként működik abban az értelemben, hogy az egység aktivitásának váltogatásaira szolgál.

Minden rejtett vagy kimeneti rétegbeli egység a nettó kimenetre **aktiválási** függvényt alkalmaz, ahogy az a 7.10. ábrán látható. A függvény az egység által reprezentált neuron aktivitását szimbolizálja. A **logisztikus** vagy **szigmoid** függvényt használják. A j egység adott I_j nettó bemenetére a j egység kimenetét, O_j-t a következőképpen számítjuk ki:



7.10. ábra | Rejtett vagy kimeneti rétegben szereplő j egység: A j egység bemenetei az előző réteg kimenetei. Ezeket összeszorozzuk a megfelelő súlyaikkal, hogy egy súlyzott összeget kaphassunk, melyet a j egység eltéréséhez adunk hozzá. Egy nemlineáris aktiválási függvényt alkalmazunk az így kapott bemenetre

$$O_j = \frac{1}{1+e^{-I_j}}. \tag{7.13}$$

Erre a függvényre *összepréselő* függvényként is szoktak hivatkozni, mivel egy nagy értelmezési tartományt képez le 0 és 1 közötti tartományba. A logisztikus függvény nemlineáris és differenciálható, így a hiba-visszaterjesztéses algoritmus olyan osztályozási problémákat is modellezhet, amelyek lineárisan nem szétválaszthatók.

A hiba visszaterjesztése – A hibát a súlyok és torzítások olyan módosításaival terjesztjük vissza, amelyek tükrözik a háló becslésének a hibáját. A kimeneti rétegen lévő j egységnél az Err_j hibát a következőképpen számítjuk ki:

$$Err_j = O_j(1 - O_j)(T_j - O_j), \tag{7.14}$$

ahol O_j a j egység aktuális kimenete, valamint T_j az adott tanuló minta ismert osztályozási címkéjén alapuló valódi kimenet. Vegyük észre, hogy $O_j(1 - O_j)$ a logisztikus függvény deriváltja.

Egy rejtett rétegbeli egység hibájának kiszámításakor a következő réteg j egységhez kapcsolódó egységeihez tartozó hibák súlyozott összegét kell figyelembe venni. Egy rejtett rétegbeli j egység hibája a következő:

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}, \tag{7.15}$$

ahol w_{jk} a j egység és a következő magasabb rétegbeli k egység közötti kapcsolat súlya, és Err_k a k egység hibája.

A súlyokat és az torzításokat úgy módosítjuk, hogy tükrözzék a terjesztett hibákat. A súlyokat a következő egyenlet alapján módosítjuk, ahol Δw_{ij} a w_{ij} súly változását jelöli:

$$\Delta w_{ij} = (l) Err_j O_i, \tag{7.16}$$

$$w_{ij} = w_{ij} + \Delta w_{ij}. \tag{7.17}$$

Mit jelent az (l) kifejezés a (7.16) *egyenletben?* Az l változó a **tanulási ráta**, egy konstans tipikusan 0 és 1 közötti értékkel. A hiba-visszaterjesztéses algoritmus csökkenő gradiensek elvén alapuló módszert használ azon súlyok halmazának megkeresésekor, amelyek modellezhetik az adott osztályozási feladatot, minimalizálva a minták aktuális osztályozási címkéje és a háló osztályra vonatkozó becslése közti várható torzítás négyzetét. A tanulási ráta segítségével elkerülhetjük, hogy a döntési felület egy lokális minimumában ott ragadjunk (ahová a súlyok láthatóan konvergálnak, viszont nem ez az optimális megoldás), azaz segít a globális minimum megtalálásában. Ha a tanulási ráta túl kicsi, a tanulás nagyon lassúvá válhat. Ha a tanulási ráta túl nagy, nem kielégítő megoldások között oszcillálhatunk. Ökölszabályként a tanulási rátát $1/t$-re állíthatjuk, ahol t a tanuló halmazon eddig vett iterációk száma.

A torzításokat az alábbi egyenletek alapján módosítjuk, ahol $\Delta\theta_j$ a θ_j torzítás változását jelöli:

$$\Delta\theta_j = (l)Err_j, \tag{7.18}$$

$$\theta_j = \theta_j + \Delta\theta_j. \tag{7.19}$$

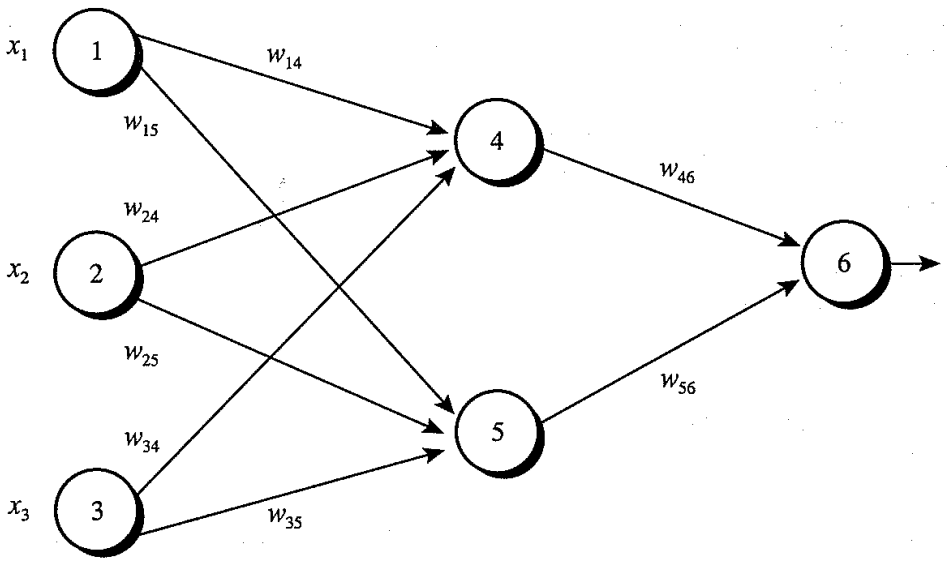
Vegyük észre, hogy itt a súlyokat és torzításokat minden egyes minta sorra vétele után módosítottuk. Erre **eseti módosításként** szoktak hivatkozni. Ehelyett, a súlyok és torzítások növekményeit változókban összegyűjthetjük, és a súlyokat és torzításokat csupán a tanuló halmaz összes mintájának sorra vétele után módosítjuk. Ezt az utóbbi stratégiát **időszakos módosításnak** nevezik, ahol a tanuló halmazon végigfutó iteráció jelenti az **időszakot**. Elméletben a hiba-visszatérítéses algoritmus matematikai származtatásánál az időszakos módosítást alkalmazzák, a gyakorlatban mégis az eseti módosítás a gyakoribb, mivel ez általában pontosabb eredményeket ad.

Leállási feltételek – A hangolás befejeződik, ha

- a Δw_{ij} értékeinek mindegyike az előző időszakban egy megadott küszöb alá esett, vagy
- a rosszul osztályozott minták százaléka az előző időszakban egy küszöb alá került, vagy
- az időszakoknak egy előre meghatározott számát végrehajtottuk.

A gyakorlatban több százezer időszakra van szükség ahhoz, hogy a súlyok konvergálni kezdjenek.

7.5. példa | A hiba-visszaterjesztéses algoritmus általi tanulás számításainak egy mintája. A 7.11. ábra egy előrecsatolt többrétegű hálót mutat. Legyen a tanulási ráta 0,9. A súlyok és torzítások kezdeti értékeit az első tanuló mintával együtt a 7.3. táblázat tartalmazza, $X = (1, 0, 1)$, az osztályozási címke 1.



7.11. ábra | Példa az előrekapcsolt többrétegű neuronhálókra

7.3. táblázat | Kezdeti bemenet, súly és torzítás értékek

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0,2	-0,3	0,4	0,1	-0,5	0,2	-0,3	-0,2	-0,4	0,2	0,1

7.4. táblázat | A nettó bemeneti és kimeneti számításai

Egység, j	Nettó bemenet, I_j	Nettó kimenet, O_j
4	$0,2 + 0 - 0,5 - 0,4 = -0,7$	$1/(1 + e^{0,7}) = 0,332$
5	$-0,3 + 0 + 0,2 + 0,2 = 0,1$	$1/(1 + e^{-0,1}) = 0,525$
6	$(-0,3)(0,332) - (0,2)(0,525) + 0,1 = -0,105$	$1/(1 + e^{0,105}) = 0,474$

Ez a példa az első tanuló mintán a hiba-visszaterjesztéses algoritmus végrehajtásának számításait mutatja be. A mintát betápláljuk a hálóba, és kiszámítjuk minden egység nettó bemenetét és kimenetét. Ezután visszaterjesztéssel megadjuk az egyes egységeknél fellépő hibákat. A hibák értékei a 7.5. táblázatban találhatók. A súlyok és torzítások módosításait a 7.6. táblázat tartalmazza. ■

7.5. táblázat | A hibák kiszámítása a csúcsoknál

Egység, j	Err_j
6	$(0,474)(1 - 0,474)(1 - 0,474) = 0,1311$
5	$(0,525)(1 - 0,525)(0,1311)(-0,2) = 0,0065$
4	$(0,332)(1 - 0,332)(0,1311)(-0,3) = 0,0087$

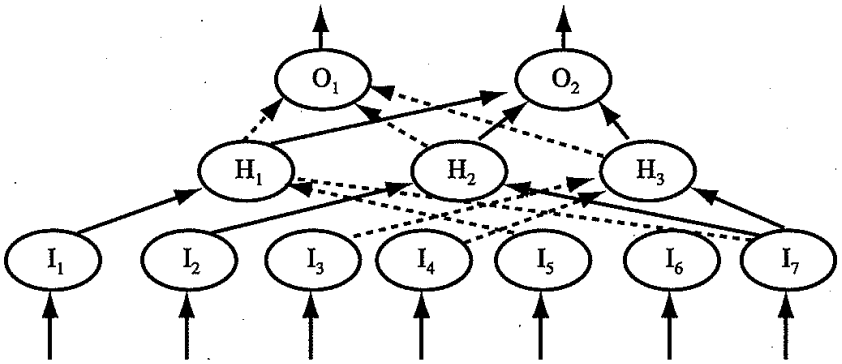
7.6. táblázat | A súlyok és az torzítások módosításainak kiszámítása

Súly vagy torzítás	Új érték
w_{46}	$-0,3 + (0,9)(0,1311)(0,332) = -0,261$
w_{56}	$-0,2 + (0,9)(0,1311)(0,525) = -0,138$
w_{14}	$0,2 + (0,9)(-0,0087)(1) = 0,192$
w_{15}	$-0,3 + (0,9)(-0,0065)(1) = -0,306$
w_{24}	$0,4 + (0,9)(-0,0087)(0) = 0,4$
w_{25}	$0,1 + (0,9)(-0,0065)(0) = 0,1$
w_{34}	$-0,5 + (0,9)(0,0087)(1) = -0,508$
w_{35}	$0,2 + (0,9)(-0,0065)(1) = 0,194$
θ_6	$0,1 + (0,9)(0,1311) = 0,218$
θ_5	$0,2 + (0,9)(-0,0065) = 0,194$
θ_4	$-0,4 + (0,9)(-0,0087) = -0,408$

A hiba-visszaterjesztéses algoritmusnak számos változatát, alternatíváját javasolták a neuronhálóknál osztályozásra. Ezek magukban foglalhatják a háló topológiájának, a tanulási rátának vagy egyéb paramétereknek dinamikus kiigazításait, vagy másféle hibafüggvények használatát.

7.5.4. | A hiba-visszaterjesztéses algoritmus és az értelmezhetőség

Miként érthetném meg, hogy mit tanult meg a hiba-visszaterjesztéses háló? A neuronhálók egyik fő hátránya az ismeretek ábrázolásában rejlik. A súlyokkal összekötött egységek hálójának formájában megszerzett tudás nehezen értelmezhető egy ember számára. Ez a tényező ösztökélte a behangolt neuronhálókba beágyazott tudás kifejtésének, valamint szimbolikus ábrázolásának a kutatását. A módszerek magukba foglalják a szabá-



Minden H_i rejtett csúcshoz a közös aktiválási értékek megtalálása
for H_1: (-1,0,1)
for H_2: (0.1)
for H_3: (-1,0.24,1)

Szabályok származtatása a közös aktiválási értékek és az O_j kimeneti csúcsok összekapcsolásával
IF ($H_2 = 0$ and $H_3 = -1$) OR
($H_1 = -1$ and $H_2 = 1$ and $H_3 = -1$) OR
($H_1 = -1$ and $H_2 = 0$ and $H_3 = -0.24$)
THEN $O_1 = 1$, $O_2 = 0$
ELSE $O_1 = 0$, $O_2 = 1$

Szabályok származtatása az I_i bemeneti csúcsoknak az O_j kimeneti csúcsokkal való összekapcsolásával
IF ($I_2 = 0$ AND $I_7 = 0$) THEN $H_2 = 0$
IF (I_4) = 1 AND $I_6 = 1$) THEN $H_3 = -1$
IF ($I_5 = 0$) THEN $H_3 = -1$

Szabályok létrehozása a bemenetek és a kimeneti osztályok összekapcsolásával
IF ($I_2 = 0$ AND $I_7 = 0$ AND $I_4 = 1$ AND $I_6 = 1$) THEN osztály = 1
IF ($I_2 = 0$ AND $I_7 = 0$ and $I_5 = 0$) THEN osztály = 1

7.12. ábra | A szabályokat kinyerhetjük behangolt neuronhálókból [LSL95]

lyoknak a hálókból való kikövetkeztetésére szolgáló, illetve az érzékenységvizsgálatnál használatos technikákat.

A szabályok kinyerésére különféle algoritmusokat javasoltak. A módszerek általában megszorításokat adnak az adott neuronháló hangolásánál használt eljárásokra, a háló topológiájára és a bemeneti értékek diszkretizációjára vonatkozóan.

A teljesen összekötött hálókat nehéz érthetővé tenni. Így a neuronhálókból a szabályok kikövetkeztetésének első lépéseként gyakran megmetszik a hálót. Ez a háló egyszerűsítését jelenti azoknak a súlyozott éleknek az eltávolításával, amelyek a legkevésbé befolyásolják a behangolt háló működését. Például kitörölhetünk egy súlyozott élt, ha ez nem jár együtt a háló osztályozási pontosságának a csökkenésével.

Ha a háló megmetszése elkészült, egyes eljárások klaszterezik az éleket, egységeket vagy az aktivitásokat. Az egyik módszernél például klaszterezést hajtunk végre azért, hogy megtaláljuk egy adott kétrétegű neuronhálóban (7.12. ábra) minden rejtett egységnél a gyakori aktivitásértékek halmazát. Ezután analizálják az összes egységnél ezeknek az aktivitási értékeknek a kombinációit. A szabályokat az aktivitási értékeknek a megfelelő kimeneti egység értékeivel való összekapcsolásából származtatják. Hasonlóan a bemeneti és aktivitási értékeknek a halmazait vizsgálják olyan szabályok kialakításához, amelyek a bemeneti és a rejtett rétegbeli egységek kapcsolatait tárják fel. Végül a szabályoknak e két halmazát kombinálják, IF-THEN szabályokat kialakítva. Más algoritmusok másféle formájú szabályokat származtathatnak, például *N*-ből *M* szabályokat (ahol a szabály előzményének *N* feltételéből *M*-nek teljesülnie kell ahhoz, hogy a szabály következményét végrehajtsuk), döntési fákat *N*-ből *M* szabályokkal, fuzzy szabályokat vagy véges automatákat.

Annak becslésére, hogy egy adott bemeneti változó mennyiben befolyásolja a háló kimenetét, **érzékenységvizsgálati** technikákat szoktak alkalmazni. A változó bemeneteit változtatgatjuk, míg a többi változó értékét rögzítjük. Mindeközben megfigyeljük a háló kimenetelének viselkedését. Azokat az ismereteket, amelyeket az analízis e formája által nyerünk, *„IF X csökken 5%-kal, THEN Y növekszik 8%-kal"* alakú szabályokkal fogalmazhatjuk meg.

7.6. | A társítási szabályok bányászatán alapuló osztályozás

A társítási szabályok bányászatánál alkalmazott ötleteket fel lehet-e használni osztályozásra is? A társítási szabályok bányászata az adatbányászati kutatásoknak fontos és nagyon aktív területe. A könyv 6. fejezete a társítási szabályok bányászatának számos algoritmusát mutatja be. Ebben az alfejezetben három eljárást tanulmányozunk történeti sorrendben. Az első kettő, az *ARCS* és a *társítási osztályozás*, szabályokat használ az osztályozásra. A harmadik módszer, a *CAEP*, a „felbukkanó mintákat" bányássza ki, amiknél megjelenik a társítások kibányászásánál használt megalapozottság fogalma.

Az első módszer klaszterezésre alapozva bányássza ki a társítási szabályokat, és ezeket használja azután osztályozásra. Az **ARCS (Association Rule Clustering System, társítási szabályokat osztályozó rendszer)** a szabályokat $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$ formában adja meg, ahol az A_{quan1} és az A_{quan2} ellenőrzések mennyiségi attribútumok értékkészletein

(ahol az értékkészleteket dinamikusan határozzák meg), A_{cat} pedig egy, az adott tanuló adatból való kategorikus attribútumhoz rendel osztályozási címkét. A társítási szabályokat kétdimenziós rácson szerkesztik meg. Az algoritmus végigpásztázza a rácsot, szabályok téglalap alakú csoportjait keresve. Így az egy szabálycsoportba eső mennyiségi attribútumok szomszédos értékkészleteit kombinálhatják is. Az ARCS által megadott csoportosított szabályokat alkalmazták osztályozásra, és a pontosságukat összehasonlították a C4.5 algoritmuséval. A tapasztalatok általában azt mutatták, hogy az ARCS kicsit pontosabb, mint a C4.5, ha az adatok közt vannak az átlagtól nagyon eltérő elemek. Az ARCS pontossága összefügg az alkalmazott diszkretizáció mértékével. Skálázhatóság szempontjából az ARCS állandó méretű memóriát igényel, függetlenül az adatbázis méretétől. A C4.5 exponenciálisan több végrehajtási időt igényel, mint az ARCS, ha megköveteljük, hogy a teljes adatbázis mérete, valamilyen tényezővel megszorozva, teljesen beférjen az elsődleges memóriába.

A második módszerre **társítási bányászásként** szoktak hivatkozni. A szabályokat *condset* $\Rightarrow y$ alakban adják meg, ahol a *condset* a *tételek* (vagy attribútum–érték párok) egy halmaza és y egy osztályozási címke. Egy előre meghatározott minimális megalapozottságú szabályt **gyakorinak** nevezünk, ahol egy szabály **megalapozottsága** s, ha az adott adathalmaz mintáinak s százaléka tartalmazza a *condset*-et és az y osztályba tartozik. Egy minimális megbízhatóságú szabályt **precíznek** mondunk, ahol egy szabály c **megbízhatóságú**, ha az adott adathalmaz *condset*-et tartalmazó mintáinak c százaléka tartozik az y osztályhoz. Ha a szabályoknak egy halmaza ugyanazzal a *condset*-tel rendelkezik, akkor a legnagyobb megbízhatóságú szabályt választjuk egy **lehetséges szabálynak** (PR, possible rule), hogy a halmazt reprezentálja.

A társításos osztályozás módszere két lépést tartalmaz. Az első lépésben megkeressük az összes gyakori és precíz PR-ek halmazát. Ehhez egy a 6.2.1. alfejezetben az Apriorinál tárgyalthoz hasonló iteratív megközelítést használnak, ahol előzetes ismeretek alapján szűkítik a szabályok keresési terét. A második lépés egy heurisztikus eljárással elkészíti az osztályozót, amelynek során a felfedezett szabályokat csökkenő precedencia szerint sorrendbe állítják, ami a megbízhatóságukon és a megalapozottságukon alapul. Az algoritmusnak a leghosszabb megtalált szabály hosszától függően akár többször is végig kell mennie az adathalmazon. Egy új minta osztályozásánál a legelső illeszkedő szabály alapján történik az osztályozás. Az osztályozó tartalmaz egy alapértelmezett szabályt is a legkisebb precedenciával azon új minták számára, amelyek az osztályozó egyetlen korábbi szabályát sem elégítik ki. Több adathalmazon is azt tapasztalták, hogy a társításos osztályozási módszer pontosabb eredményt ad, mint a C4.5. Megmutatták, hogy mindkét fenti lépés lineáris növekedési ütemmel rendelkezik.

A harmadik módszer a **felbukkanó minták összegzésével történő osztályozás** (CAEP, classification by aggregating emerging patterns), az elemhalmaz megalapozottság fogalmát alkalmazza a **felbukkanó minták** (EP, emerging patterns) kibányászásához, amiket az osztályozó felépítéséhez használnak. Durván fogalmazva, az EP olyan elemhalmaz (vagy elemek halmaza), aminek a megalapozottsága jelentős módon megnő egy osztályhoz tartozó adatokból egy másikba áttérve. A két megalapozottság arányát az EP **növekedési arányának** nevezik. Tegyük fel például, hogy adott a vásárlóknak egy adathalmaza a *vesz_számítógépet* = "*igen*", C_1, és a *vesz_számítógépet* = "*nem*", C_2 osztályokkal. Az {*életkor* = "$\leq$ 30", *tanuló* = "*nem*"} elemhalmaz például egy jellemző EP, aminek a C_1-beli 0,2%-os megalapozottsága 57,6%-ra nő a C_2-ben, ezzel $\frac{57{,}6\%}{0{,}2\%}$ = 288-szoros

növekedési arányt eredményezve. Vegyük észre, hogy a tétel vagy egy egyszerű egyenlőség teljesülését ellenőrzi egy kategorikus attribútumnál, vagy azt vizsgálja, hogy egy numerikus attribútum egy adott intervallumba esik-e. Mindegyik EP többattribútumos ellenőrzés, és nagyon élesen elkülönítheti az egy osztályhoz tartozó előfordulásokat a többitől. Példának okáért, ha egy új X minta tartalmazza a fenti EP-t, akkor 99,6%-os eséllyel állíthatjuk, hogy X a C_2-höz tartozik. Általában az EP megkülönböztető ereje durván arányos a növekedési arányával és a célosztályhoz tartozó megalapozottságával.

Miként építi fel a CAEP az EP-k segítségével az osztályozót? A CAEP minden C osztályhoz olyan EP-ket talál, amelyek rendelkeznek egy bizonyos megalapozottsággal és növekedési aránnyal, ahol a növekedési arányt az összes nem C-beli minta halmazával szemben vett összes C-beli minta halmazára számítják. *Határoló* algoritmusokat alkalmazhatnak erre a célra. Amikor egy új X mintát osztályoznak, minden C osztályra összesítik az X-ben előforduló összes EP-t, és ez adja meg a C pontszámát, amit aztán normalizálnak. A legnagyobb normalizált pontszámmal rendelkező osztály határozza meg az X osztályozási címkéjét.

A CAEP-et a C4.5-nél és az társításon alapuló osztályozónál is pontosabbnak találták számos adathalmazon. Olyan adatok esetében is jól működött, amelyeknél az érdeklődés központjában álló osztály nem a leggyakoribb volt. Az adatok tömege és kiterjedtsége határozza meg a növekedési ütemét. Egy alternatív osztályozót is javasoltak, a *JEP-osztályozót*, amely az **ugró felbukkanó mintákon** (JEP, jumping emerging patterns) alapul. A JEP az EP-k különleges fajtája, olyan tételhalmaz, aminek a megalapozottsága az egyik adathalmazról a másikra áttérve nulláról hirtelen egy nullától különböző értékre ugrik. A két osztályozót egymás kiegészítőiként tartják számon.

7.7. | Más osztályozási módszerek

Ebben a részben számos egyéb osztályozási eljárást is tárgyalunk röviden. Ezek között a módszerek közt szerepel a k-legközelebbi szomszédon alapuló osztályozás, az eset alapú következtetés, a genetikus algoritmusok, a közelítő halmazokon és a fuzzy logikán alapuló megközelítések. Ezeket az eljárásokat általában kevésbé alkalmazzák kereskedelmi adatbányászati rendszerekben osztályozásra, mint a korábban bemutatottakat. A legközelebbi szomszédokon alapuló osztályozás például az összes tanuló mintát tárolja, ami nehézségeket okozhat, ha a tanulás nagyon nagy adathalmazon történik. Azonkívül az eset alapú következtetési, a genetikus algoritmusos és a közelítő halmazos alkalmazások még kísérleti fázisban vannak. Az eljárások azonban egyre nagyobb népszerűségnek örvendenek, ezért is foglalkozunk velük.

7.7.1. | A k-legközelebbi szomszédon alapuló osztályozás

A legközelebbi szomszédokon alapuló osztályozók az analógiák alapján való tanulás ötletére épülnek. A tanuló mintákat n-dimenziós numerikus attribútumok írják le. Mindegyik minta egy pontnak felel meg az n-dimenziós térben. Ebből a szempontból az összes tanuló mintát egy n-dimenziós sablontérben tároljuk. Adott ismeretlen minta esetén a

***k*-legközelebbi szomszédon alapuló osztályozó** megkeresi a sablontérben azt a k tanuló mintát, amelyek a legközelebb vannak az ismeretlen mintához. Ez a k tanuló minta lesz a tanuló minta k „legközelebbi szomszédja". A „közelséget" az euklideszi távolságra nézve definiáljuk, ahol két pont, $X = (x_1, x_2,..., x_n)$ és $Y = (y_1, y_2,..., y_n)$ euklideszi távolsága a következő:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \ . \tag{7.20}$$

Az ismeretlen mintát a *k-legközelebbi* szomszéd közti leggyakoribb osztályba soroljuk. Ha $k = 1$, az ismeretlen mintát a sablontérben a hozzá legközelebb eső tanuló minta osztályához vesszük.

A legközelebbi szomszédokra épülő osztályozók **előfordulás alapúak** vagy **lusta tanulók** abban az értelemben, hogy az összes tanuló minta tárolásra kerül, és addig nem épül fel az osztályozó, amíg szükség nem lesz egy új minta osztályozására. Ezzel ellentétesen működnek a **buzgón tanuló eljárások**, például a döntési fa indukciója vagy a hiba-visszaterjesztéses algoritmus, amelyek már az osztályozandó új minták megjelenése előtt megszerkesztenek egy általános modellt. A lusta tanulók drága számítási költségekkel járhatnak együtt, ha a lehetséges szomszédok (azaz a tárolt tanuló minták) száma, amelyekkel a címkézetlen mintát összehasonlítjuk, nagy. Ezért hatékony indexelési technikákat igényelnek. Ahogy várható is, a lusta tanuló módszerek gyorsabban tanulnak, mint a buzgó tanuló eljárások, azonban az osztályozást lassabban végzik el, hiszen minden számítást erre a szakaszra halasztottak. A döntési fa indukciójától és a hiba-visszaterjesztéses algoritmustól eltérően, a legközelebbi szomszédokon alapuló osztályozók minden attribútumhoz egyenlő súlyt rendelnek. Ez zavart kelthet, ha az adatok több, a feladat szempontjából érdektelen attribútumot is tartalmaznak.

A legközelebbi szomszédokon alapuló osztályozók előrejelzésre is használhatók, azaz egy valós értékű becslést adnak vissza egy adott ismeretlen minta esetében. Ebben az esetben az osztályozó az ismeretlen mintához tartozó *k-legközelebbi* minta valós értékű címkéinek az átlagértékét adja eredményként.

7.7.2. | Eset alapú következtetés

Az **eset alapú következtetési** (CBR, case-based reasoning) **osztályozók** előfordulás alapúak. Ellentétben a legközelebbi szomszédokon alapuló osztályozókkal, amelyeknél a tanuló mintákat az euklideszi tér pontjaiként kezeljük, a mintákat vagy „eseteket" a CBR összetett szimbolikus leírásokként tárolja. A CBR üzleti alkalmazásai közt szerepel az ügyfélszolgálatok segítő részlegei számára készített problémamegoldó program, ahol az esetek a termékekkel kapcsolatos diagnosztikai problémákat jelentik. A CBR-t olyan területeken is alkalmazták, mint a tervezés vagy a jog, ahol az esetek technikai tervrajzok, illetve törvények.

Egy adott új osztályozandó esetnél az eset alapú következtetés alkalmazásakor először azt vizsgálják, hogy létezik-e ezzel azonos a tanuló esetek közt. Ha találnak ilyet, akkor az ahhoz tartozó megoldást adják vissza. Ha nem találnak, akkor az eset alapú következ-

tetés második lépéseként olyan tanuló eseteket fognak keresni, amelyeknek összetevői hasonlók az új eset bizonyos összetevőihez. Fogalmi szinten ezeket a tanuló eseteket az új eset szomszédjaiként is felfoghatjuk. Ha az eseteket gráfokkal ábrázoljuk, ez az új eset részgráfjaihoz hasonló gráfok keresését jelenti. Ezután megpróbálják kombinálni a szomszédos tanuló esetek megoldásait, hogy eredményt szolgáltathassanak az új esetnél. Ha egyes megoldások között összeférhetetlenség támad, akkor szükségessé válhat más megoldások keresése. Az eset alapú következtetésnél használhatunk háttérismereteket és problémamegoldó stratégiákat, hogy a megoldásoknak egy alkalmas kombinációját javasolhassuk.

Az eset alapú következtetés kihívásai közé tartozik a hasonlóság jó mértékének megtalálása (például az illeszkedő részgráfokra), a tanuló esetek indexelésére szolgáló hatékony technikák kifejlesztése és a megoldások kombinálását végző módszerek megtervezése.

7.7.3. | Genetikus algoritmusok

A **genetikus algoritmusok** a természetes evolúcióból próbálnak meg ötletet meríteni. Általában a genetikus tanulás a következőképpen kezdődik. Vesszük véletlenszerűen megadott szabályok egy kezdeti populációját. Minden szabályt megjeleníthetünk bitekből álló karakterláncként is. Egyszerű példaként tételezzük fel, hogy az adott tanuló halmazban lévő mintákat két logikai attribútum, A_1 és A_2 írja le, és két osztály létezik, C_1 és C_2. Az *IF* A_1 *AND NOT* A_2 *THEN* C_2 szabályt a 100 bitsorozat kódolja, ahol a két bal szélső bit A_1-et és A_2-t reprezentálja és a jobb oldali bit felel meg az osztálynak. Hasonlóan az *IF NOT* A_1 *AND NOT* A_2 *THEN* C_1 szabályt 001-ként kódolhatjuk. Ha egy attribútum k értékkel rendelkezik, ahol $k > 2$, akkor k bitet használhatunk az attribútum értékeinek kódolására. Az osztályokat hasonló módon kódolhatjuk.

A legéletképesebb egyed fennmaradásának elvét követve egy új populációt alakítunk ki a jelenlegi populáció legmegfelelőbb szabályaiból és ezeknek a szabályoknak az utódaiból. Egy szabály életképességét többnyire a tanuló minták egy halmazán vett pontossága határozza meg.

Az utódokat genetikai műveletek – mint a keresztezés és a mutáció – alkalmazásával kapjuk. A keresztezés során a kódolt szabálypárok részsorozatait csereberéljük, hogy egy új kódolt szabálypárt hozzunk létre. A mutációnál pedig egy szabályhoz tartozó sorozat véletlenül kiválasztott bitjeit invertáljuk.[4]

A szabályok kezdeti populációjából kiindulva az új populációk készítésének folyamata addig zajlik, amíg egy olyan P populáció nem alakul ki, amelynél minden P-beli szabály elér egy előre megadott életképességi küszöböt.

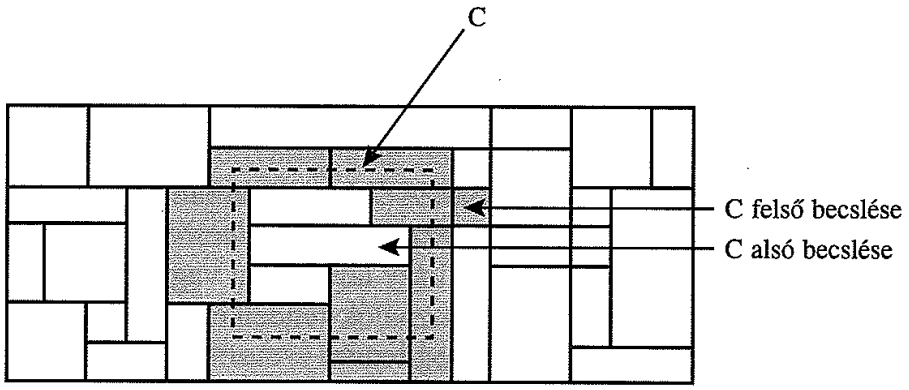
A genetikus algoritmusok könnyen párhuzamosíthatók, és osztályozáson kívül más optimalizálási problémáknál is használhatók. Az adatbányászatban arra is alkalmassá válhatnak, hogy más algoritmusok pontosságát is kiértékeljék.

4 | Azaz 0-ból 1 lesz és 1-ből 0. *(A fordító megjegyzése)*

7.7.4. | Közelítő halmazokon alapuló megközelítés

A **közelítő halmazok elmélete** pontatlan vagy zajos adatok szerkezeti összefüggéseinek felfedezésére használatos. Csak diszkrét értékű attribútumok esetén működik. Emiatt a folytonos értékű attribútumokat diszkretizálni kell az alkalmazása előtt.

A közelítő halmazok elmélete az adott tanuló adatokon belüli **ekvivalenciaosztályok** létesítésén alapszik. Az azonos ekvivalenciaosztályba tartozó adatminták egymástól nem megkülönböztethetők, azaz a minták az adatot leíró attribútumokon azonosak. Valódi adatokat véve gyakran előfordul, hogy néhány osztály nem különíthető el egymástól az elérhető attribútumokban kifejezve. A közelítő halmazok segítségével hozzávetőlegesen vagy „durván" definiálhatjuk az ilyen osztályokat. Adott C osztályra vonatkozó közelítő halmazos definíció alapján C-t két halmazzal közelítik meg – C **alsó** és **felső becslésével**. C alsó becslése azokat az adatmintákat tartalmazza, melyek az ismert attribútumok alapján biztosan C-hez tartoznak. C felső becslése pedig azon mintákból áll, melyekről az ismert attribútumokat figyelembe véve nem jelenthető ki, hogy nem tartoznak C-hez. Egy C osztályhoz tartozó alsó és felső becsléseket mutatja a 7.13. ábra, ahol minden téglalap egy ekvivalenciaosztályt jelent. Minden osztályhoz döntési szabályokat generálhatunk. Tipikusan egy döntési táblát használnak a szabályok ábrázolására.



7.13. ábra | A C osztályhoz tartozó minták halmazának közelítő halmazokkal történő becslése C felső és alsó becslő halmazait használva. A téglalap alakú területek ekvivalenciaosztályokat jelölnek

A közelítő halmazok a jellemzők csökkentésére is használhatók, azaz azoknak az attribútumoknak a megkeresésére és eltávolítására, amelyek nem járulnak hozzá az adott tanuló adatok osztályozásához. Alkalmazhatók emellett még fontossági elemzésre is (relevance analysis), ahol az osztályozási feladat szempontjából becslést adnak minden egyes attribútum jelentőségére vagy a megoldáshoz való hozzájárulására. Az attribútumok azon minimális részhalmazának (**redukáltjának**) a megtalálása, amely az adathalmazban rejlő összes fogalom leírására képes, NP-nehéz. Mindemellett javasoltak algoritmusokat a számítások csökkentésére. Az egyik módszernél például **különbözőségi mátrixot** alkalmaznak, amiben minden egyes adatmintapár esetében tárolják az attribútumok különbségeit. Ahelyett, hogy az egész tanuló halmazt átnéznék, a mátrixban keresik meg a fölösleges attribútumokat.

7.7.5. | Fuzzy halmazos megközelítések

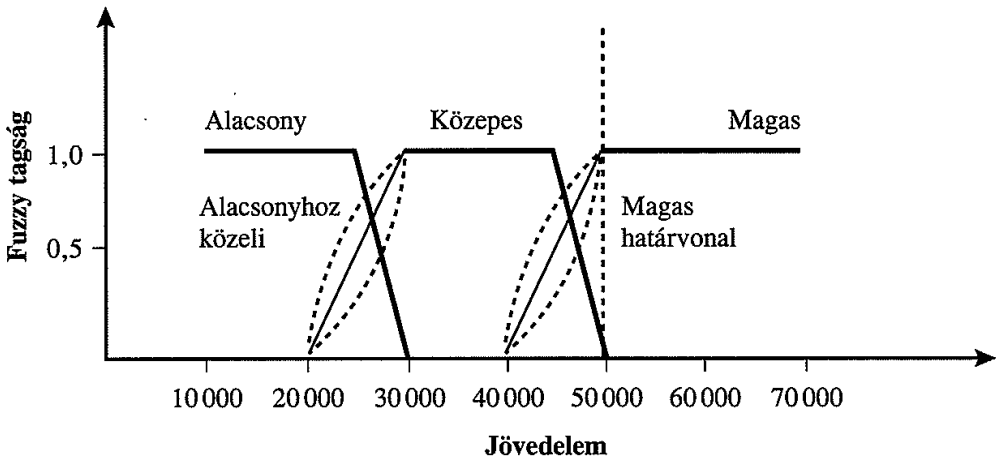
A szabály alapú rendszerek osztályozása azzal a hátránnyal jár, hogy a folytonos értékű attribútumoknál túl éles határokat adnak. Például képzeljük el a következő fogyasztók hitelkérelmének jóváhagyására vonatkozó szabályt. A szabály lényegében azt mondja ki, hogy azoknak a vevőknek a kérvényeit, akiknek legalább két éve van munkájuk és magas jövedelemmel rendelkeznek (azaz legalább 50 000 dollárral), elfogadjuk:

$$IF\ (munkaévek\_száma >= 2) \wedge (jövedelem >= 50\,000)\ THEN\ hitel = \text{“}elfogadva\text{”} \quad (7.21)$$

A (7.21) szabály szerint egy vevő, akinek már legalább két éve van munkája, kap hitelt, ha jövedelme mondjuk 50 000 $, de nem kap, ha 49 000 $. Egy ilyen erős határ meghúzása nem tűnik igazságosnak. Előnyösebb, ha fuzzy logikával bővíthetjük a rendszert, hogy „fuzzy” küszöböket vagy határokat adhassunk meg. Ahelyett, hogy pontos határokat húznánk meg a kategóriák vagy halmazok között, a 0 és 1 közé eső igazságértékek adják meg annak a mértékét, hogy egy bizonyos érték egy adott kategóriához tartozik-e. Így a fuzzy logikával megragadhatjuk a valóságnak azt a szeletét, miszerint a 49 000 $ jövedelem valamekkora mértékben magas, de nem annyira, mint az 50 000 $ jövedelem.

Osztályozást végző adatbányászati rendszerek számára a fuzzy logika kifejezetten hasznos. Az absztrakció magas szintjein való gondolkodásnak az előnyét is magában foglalja. Általában a fuzzy logikának a szabály alapú rendszerekben való alkalmazása a következőket jelenti:

- Az attribútumok értékeit fuzzy értékekké alakítjuk át. A 7.14. ábrán látható, hogy a folytonos értékű *jövedelem* attribútumot miként képezzük le az {*alacsony, közepes, magas*} kategóriákra, és hogy miként számítjuk ki a fuzzy tagságokat vagy igazságértékeket. A fuzzy logikai rendszerek többnyire a felhasználók rendelkezésére bocsátanak grafikai eszközöket, amelyek ebben a lépésben segítenek.
- Egy adott új minta esetében esetleg több mint egy fuzzy szabály is alkalmazható. Mindegyik ilyen szabály egy szavazattal járul hozzá a kategóriákba való tartozáshoz. Tipikusan minden megbecsült kategóriánál a hozzá tartozó igazságértékeket összegzik.



7.14. ábra | A *jövedelem* fuzzy értékei

- A fenti összegeket egy értékké kombinálják össze, és ezt adja vissza a rendszer. Ezt a folyamatot úgy is végrehajthatjuk, hogy az igazságértékével súlyozott összes kategóriát összeszorozzuk a kategória várható igazságértékével. A felmerülő számítások a fuzzy tagságot ábrázoló grafikon bonyolultságától függően összetettek is lehetnek.

A fuzzy logikai rendszereket az osztályozás számos területénél alkalmazzák, például gyógyászatban vagy pénzügyi életben.

7.8. | Előrejelzés

Mi a helyzet akkor, ha nem egy kategorikus címkét, hanem folytonos értéket szeretnénk megadni? A folytonos értékek előrejelzését *regressziós* statisztikai technikákkal modellezhetjük. Például szeretnénk kialakítani egy modellt a 10 éves munkatapasztalattal és főiskolai diplomával rendelkező dolgozók fizetésének becslésére, vagy egy új termék eladásainak a számát szeretnénk megjósolni adott ár mellett. Sok feladatot megoldhatunk lineáris regresszióval, és még több kezelhető a változók olyan transzformációinak alkalmazásával, amelyek eredményeként a nemlineáris probléma lineárissá válik. Helyhiány miatt nem közölhetjük a regresszió teljes részletekbe menő leírását. Ehelyett ez az alfejezet intuitív betekintést ad a témakörbe. A fejezet végéhez érve tisztába kerülünk a lineáris, a többváltozós, a nemlineáris regresszió, valamint az általánosított lineáris modellek fogalmaival.

Számos szoftvercsomag létezik regressziós feladatok megoldására. Például a SAS *(http://www.sas.com)*, az SPSS *(http://www.spss.com)* és az S-Plus *(http://www.mathsoft.com)*.

7.8.1. | Lineáris és többváltozós regresszió

Mi a lineáris regresszió? A **lineáris regressziónál** az adatokat egy egyenes vonal segítségével modellezzük. A lineáris regresszió a regressziószámítás legegyszerűbb formája. A kétváltozós lineáris regresszió egy Y valószínűségi változót, a **válaszváltozót** (response variable) modellezi egy másik X valószínűségi változó, a **prediktor változó** vagy **előrejelzési változó** lineáris függvényeként, azaz

$$Y = \alpha + \beta X, \tag{7.22}$$

ahol az Y szórását állandónak vesszük, az α és β **regressziós együtthatók** pedig sorra megadják az egyenes Y tengellyel vett metszetét és a meredekségét. Ezek az együtthatók a **legkisebb négyzetek módszerével** kereshetők meg, ami minimalizálja az eltérést az aktuális adatok és a becsült egyenes között. Adott (x_1, y_1), (x_2, y_2), ..., (x_s, y_s) formájú s minta vagy adatpont esetén a regressziós együtthatókat a következő egyenlőségeket használó módszer segítségével adhatjuk meg:

$$\beta = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2}, \tag{7.23}$$

$$\alpha = \bar{y} - \beta\bar{x}, \tag{7.24}$$

ahol $\bar{x}$ az $x_1, x_2, \ldots, x_n$ átlaga és $\bar{y}$ az $y_1, y_2, \ldots, y_n$ átlaga. Az α és β együtthatók gyakran jó közelítéseit adják az egyébként bonyolult regressziós egyenleteknek.

7.6. példa | Lineáris regresszió a legkisebb négyzetek elvének alkalmazásával. A 7.7. táblázat adatpárok egy halmazát mutatja, ahol X egy főiskolai diplomás munkában eltöltött éveinek a számát adja meg, Y pedig a fizetését. A 7.15. ábrán az adatokat egy grafikonon ábrázoltuk, ennek alapján lineáris viszonyt sejthetünk a két változó, X és Y között. Úgy modellezzük a kapcsolatot, mintha a fizetés az alábbi egyenlet szerint viszonyulna a munkában eltöltött évekhez: $Y = \alpha + \beta X$.

7.7. táblázat | Fizetési adatok

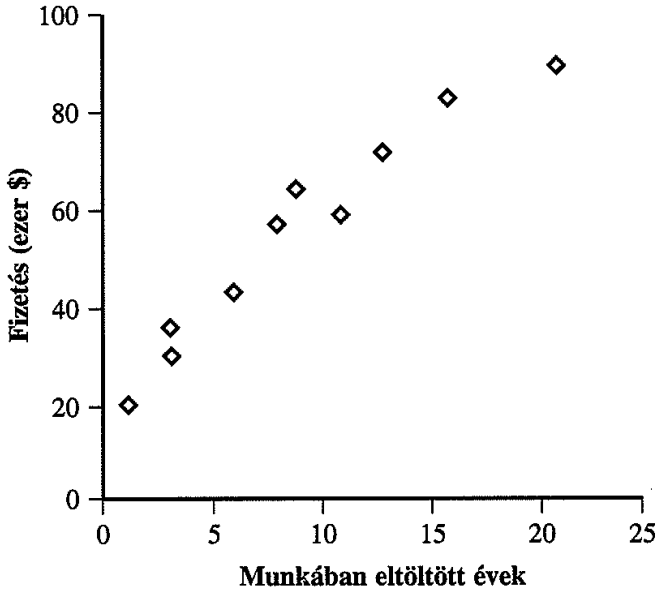
X Munkaévek száma	Y Fizetés (ezer $)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

A fenti adatokból kiszámíthatjuk, hogy $\bar{x} = 9{,}1$ és $\bar{y} = 55{,}4$. Ezeket a (7.23) egyenlőségbe behelyettesítve kapjuk

$$\beta = \frac{(3-9{,}1)(30-55{,}4)+(8-9{,}1)(57-55{,}4)+\cdots+(16-9{,}1)(83-55{,}4)}{(3-9{,}1)^2+(8-9{,}1)^2+\cdots+(16-9{,}1)^2} = 3{,}5$$

$$\alpha = 55{,}4-(3{,}5)(9{,}1) = 23{,}6$$

Így a legkisebb négyzetek egyenletei alapján az egyenest az $Y = 23{,}6 + 3{,}5X$ kifejezéssel becsülhetjük. Ezt az egyenletet felhasználva előre jelezhetjük, hogy egy főiskolai diplomás fizetése mondjuk 10 éves tapasztalattal 58 600 $ lesz. ■



7.15. ábra | A 7.6. példánál a 7.7. táblázat adatainak ábrázolása. Habár a pontok nem egy egyenesre esnek, a teljes kép azt sugallja, hogy az *X* (*munkaévek száma*) és az *Y* (*fizetés*) között lineáris viszony áll fenn

A **többváltozós regressziónál** a lineáris regresszió kiterjesztéseként több mint egy prediktor változónk van. Az *Y* válaszváltozót egy többdimenziós tulajdonságvektor (feature vector) lineáris függvényeként modellezhetjük. A többváltozós regressziós modell egy példája X_1 és X_2 előrejelzési attribútumokkal vagy változókkal a következő:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2. \tag{7.25}$$

A legkisebb négyzetek módszerét itt is alkalmazhatjuk α, β_1 és β_2 megtalálására.

7.8.2. | Nemlineáris regresszió

Hogyan modellezhetnénk olyan adatokat, amelyek úgy tűnik nincsenek lineáris függőségben? Például mi lenne, ha egy adott válaszváltozó és prediktor változó közti viszonyt egy polinomiális függvénnyel modelleznénk? A **polinomiális regressziót** a lineáris alapmodellhez adott polinomiális kifejezésekkel modellezhetjük. A változókat transzformálva a nemlineáris modellt lineárissá alakíthatjuk, amit aztán már megoldhatunk a legkisebb négyzetek módszerével.

7.7. példa | Nemlineáris regressziós modell transzformálása lineáris regressziós modellé. Tételezzünk fel harmadfokú polinomiális viszonyt a következőképpen:

$$Y = \alpha + \beta_1 X + \beta_2 X^2 + \beta_3 X^3. \tag{7.26}$$

Az egyenlőség lineárissá tételéhez új változókat definiálunk:

$$X_1 = X \quad X_2 = X^2 \quad X_3 = X^3. \tag{7.27}$$

A (7.26) egyenlőséget aztán lineáris formára hozhatjuk a fenti hozzárendelések segítségével, és a következő egyenletet kapjuk: $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$, ami a legkisebb négyzetek módszerével már megoldható. ■

A 7.9. feladatban egy nemlineáris hatványfüggvényt tartalmazó modellt lineáris regressziós modellé átalakító transzformációt kell megtalálni.

A modellek egy része minden körülmények között nemlineáris marad (például exponenciális kifejezések összegei), így nem lehet lineáris modellé átalakítani. Az ilyen esetekben összetettebb formulákon végzett kiterjedt számításokkal szintén végezhetünk legkisebb négyzetes becsléseket.

7.8.3. | Más regressziós modellek

A lineáris regresszióval folytonos értékű függvényeket modelleznek. Az egyszerűségének köszönhetően széles körben elterjedt. *Vajon kategorikus címkék megadására is használható?* Az **általánosított lineáris modellek** jelentik az elméleti alapját a lineáris regresszió kategorikus válaszváltozókon való alkalmazásának. Az általánosított lineáris modellek esetében az Y válaszváltozó szórása az Y várható értékének a függvénye szemben a lineáris regressziónál tárgyaltakkal. Az általánosított lineáris modellek gyakori típusai a **logisztikus regresszió** és a **Poisson-regresszió**. A logisztikus regresszió a prediktor változók egy halmazának lineáris függvényeként előálló esemény valószínűségét modellezi. A megszámolt adatok gyakran Poisson-eloszlásúak, és sokszor Poisson-regresszió felhasználásával modellezik ezeket.

A **loglineáris modellek** *diszkrét*, többdimenziós valószínűségi eloszlásokat közelítenek. Az adatkockacellákhoz társított valószínűségi értékek becslésére is alkalmazhatók. Például tegyük fel, hogy a *város*, *árucikk*, *év*, *értékesítés* attribútumokhoz rendelkezünk adatokkal. A loglineáris módszernél minden attribútumnak kategorikusnak kell lennie, ezért a folytonos értékű attribútumokat (például az *értékesítés*) diszkretizálni kell. A módszert ezután a kétdimenziós *város* és *árucikk*, *város* és *év*, *város* és *értékesítés* részkockák, és a háromdimenziós *árucikk*, *év*, *értékesítés* részkocka segítségével az adott attribútumok négydimenziós alapkockájában lévő összes cella valószínűségének becslésére használhatjuk. Így iteratív technikát alkalmazva alacsony rendű részkockákból magasabb rendű adatkockákat építhetünk fel. A technika sok dimenzió esetén is jól működik. Az előrejelzés mellett a loglineáris módszer hasznos az adattömörítésnél (mivel az alacsonyabb rendű részkockák együttesen kevesebb helyet igényelnek, mint az alapkocka), és az adatsimításnál (mivel a cellabecslések az alacsonyabb rendű részkockáknál kevésbé érzékenyek a minták ingadozásaira, mint az alapkocka esetén).

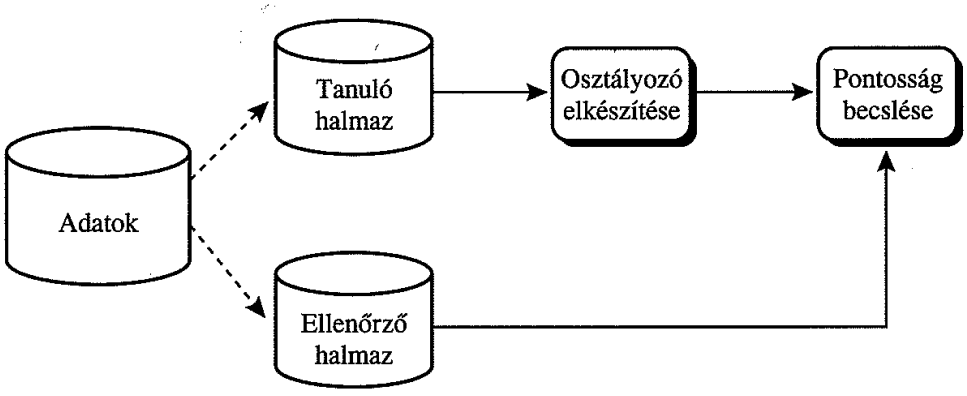
7.9. | Az osztályozó pontossága

Az osztályozó pontosságának megbecslése abból a szempontból fontos, hogy ezáltal kiértékelhetjük mennyire precízen címkézi az adott osztályozó az adatokat a jövőben, olyan adatokat, amelyeket nem használtunk az osztályozó tanításánál. Például, ha a korábbi eladások adataival tanítottuk az osztályozót a fogyasztók vásárlási szokásainak előrejelzésére, szeretnénk egy becslést kapni arra nézve, hogy az osztályozó mennyire pontosan képes megbecsülni a jövőbeli vásárlók szokásait. A pontosságra vonatkozó becslések különböző osztályozók összehasonlításánál is segíthetnek. A 7.9.1. alfejezetben olyan osztályozók pontosságát becslő eljárásokat ismertetünk, mint a *visszatartó* és a *k-szoros kereszt-megerősítéses* módszerek. A 7.9.2. alfejezet a *többszörös halmazoló és megfejelő* stratégiákat magyarázza el, amelyek az osztályozók pontosságát növelik. A 7.9.3. alfejezetben az osztályozó pontosságához és a kiválasztáshoz kapcsolódó egyéb témaköröket mutatunk be.

7.9.1. | Az osztályozó pontosságának becslése

A tanuló adatoknak az osztályozó felépítésére, valamint pontosságának a becslésére való egyidejű használata félrevezető, túlságosan optimista eredményeket adhat, mivel a tanuló algoritmus (vagy modell) túlságosan is jól illeszkedhet az adatokhoz. A visszatartó és kereszt-megerősítéses algoritmusok az adott adatok véletlenszerűen kiválasztott felosztásain alapuló elterjedt technikák az osztályozó pontosságának becslésére.

A **visszatartó módszernél** az adatokat véletlenszerűen két, egymástól független halmazra: a *tanuló halmazra* és az *ellenőrző halmazra* osztjuk. Többnyire az adatok kétharmada kerül a tanuló halmazba és a maradék egyharmada az ellenőrző halmazba. A tanuló halmaz felhasználásával készítik el az osztályozót, aminek a pontosságát az ellenőrző halmaz alapján becslik (7.16. ábra). A becslés pesszimista, hiszen a kezdeti adatoknak csak egy részét használtuk az osztályozó származtatására. A mintából való **véletlenszerű mintavételezés módszere** a visszatartó algoritmus egy változata, amelynél a visszatartó algoritmust ismétlik k alkalommal. A végső pontossági becslésként az iterációnként kapott pontosságok átlagát veszik.



7.16. ábra | Az osztályozó pontosságának becslése visszatartó módszerrel

A ***k*-szoros kereszt-megerősítéses módszernél** a kezdeti adatokat véletlenszerűen k páronként diszjunkt, nagyjából azonos méretű S_1, S_2,..., S_k halmazba vagy „tartóba" osztják. A tanítást és ellenőrzést k alkalommal végzik el. Az i. iterációnál az S_i részhalmaz marad meg ellenőrző halmazként, a többi részhalmazzal hangolják az osztályozót. Azaz az osztályozót az első lépésben az S_2,..., S_k részhalmazokon tanítják, és az S_1-en ellenőrzik, a második iterációhoz tartozó osztályozót az S_1, S_3,..., S_k halmazokon hangolják, és az S_2-n ellenőrzik és így tovább. A pontosság becslését a k iteráción vett helyes osztályzások számának és az összes minta számának a hányadosa adja. A **rétegezett kereszt-megerősítésnél** a tartókat úgy rétegezzük, hogy minden tartóban a minták osztályok közötti eloszlása közel azonos a kezdeti adatokon vett eloszlással.

A többi osztályozók pontosságát becslő módszerek között szerepel az **öntöltő algoritmus**, ami az adott tanuló előfordulásokat egyenletesen mintavételezi *visszahelyezéssel*, és a **„hagyjunk-ki-egyet" algoritmus**, ami egy k-szoros kereszt-megerősítéses módszer k halmazzal és s kezdeti mintával. Az osztályozó pontosságának becslésére általában rétegezett 10-szeres kereszt-megerősítéses módszert javasolnak (még ha a számítási erőforrások több tartó használatát is lehetővé tennék) a viszonylagos kicsi torzítása és szórása miatt.

Az ilyen jellegű technikák alkalmazása az osztályozó pontosságának becslésére növelik a teljes számítási időt, mégis hasznosak, ha különböző osztályozók közt szeretnénk választani.

7.9.2. | Az osztályozó pontosságának növelése

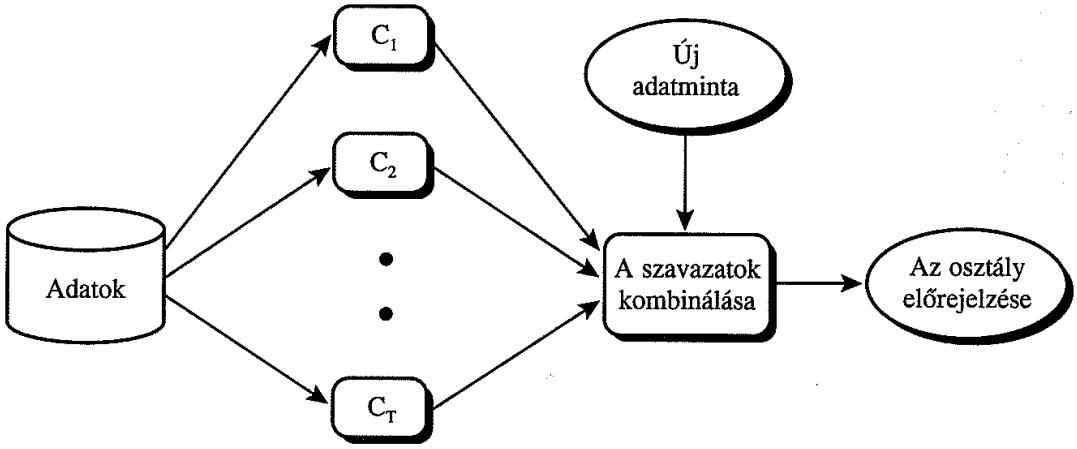
Az előző részben az osztályozók pontosságát becslő eljárásokat tanulmányoztunk. A 7.3.2. alfejezetben láttuk, hogy miként használható a metszés a döntési fa indukciójánál az eredményül kapott döntési fák precizitásának növelésére. Léteznek-e *általános technikák* az osztályozó pontosságának javítására?

A válasz igen. A **többszörös halmazolás**[5] (vagy *öntöltő felhalmozás*) és a **megfejelés** két ilyen technika (7.17. ábra). Mind a kettő C_1, C_2,..., C_T behangolt osztályozók sorozatának a kombinációt veszi azzal a céllal, hogy egy javított, ezekből összerakott C^* osztályozót készítsen.

Hogyan működnek ezek a módszerek? Tegyük fel, hogy betegek vagyunk, és szeretnénk a tüneteinken alapuló diagnózist kapni. Egyetlen doktor megkérdezése helyett több doktor véleményére leszünk kíváncsiak. Ha egy bizonyos diagnózis többször fordul elő, mint a többi, akkor azt választhatjuk a végső vagy legjobb diagnózisnak. Most helyettesítsük az összes orvost egy-egy osztályozóval, és megkapjuk a többszörös halmazolás mögött meghúzódó intuíciót. Képzeljük el ehelyett, hogy minden orvos diagnózisának értékéhez rendelünk egy súlyt, amely a korábbi diagnózisaik használhatóságán alapszik. A végső diagnózis a súlyozott diagnózisok kombinációjaként áll elő. Ez a megfejelés lényege. Vizsgáljuk meg közelebbről is ezt a két technikát.

Adott S halmazát véve az s mintáknak a többszörös halmazolás a következőképpen működik. A t. iterációra ($t = 1, 2,..., T$) egy S_t tanuló halmazt[6] vételezünk visszahelye-

5 | Többszörös halmaz alatt olyan „halmaz" értendő, melynek elemei ismétlődhetnek. *(A fordító megjegyzése)*
6 | A halmaz elnevezés itt pongyola a lehetséges ismétlődő elemek miatt. *(A fordító megjegyzése)*



7.17. ábra | Az osztályozó pontosságának növelése: többszörös halmazolás és megfejelés az osztályozók a C_1, C_2,..., C_T halmazát állítják elő. Egy ismeretlen osztályú minta megfelelő osztályba sorolása az osztályozók előrejelzéseiből többségi szavazati elven vagy valamilyen súlyozás alapján történik

zéssel a minták eredeti S halmazából. Mivel a válogatás visszahelyezéssel történik, S_t esetleg nem tartalmaz néhány eredeti S-beli mintát, míg mások többször is előfordulhatnak. Minden egyes S_t tanuló halmazzal egy C_t osztályozót hangolunk be. Egy ismeretlen X minta osztályozásánál az összes C_t osztályozó visszaadja az osztályra vonatkozó becslését, ami egy szavazatnak számít. A többszörösen halmazolt C^* osztályozó megszámlálja a szavazatokat, és X-hez a legtöbb szavazattal rendelkező osztályt rendeli. A többszörös halmazolást folytonos értékek előrejelzésénél is használhatjuk minden osztályozó becslésének az átlagos értékét véve.

A megfejelés esetében minden tanuló mintához egy súlyt rendelünk. Az osztályozók sorozatait hangoljuk. Miután a C_t osztályozót behangoltuk, a súlyokat úgy módosítjuk, hogy a következő C_{t+1} osztályozó „nagyobb figyelmet szentelhessen" C_t félreosztályozási hibáinak. C^*, a végső megfejelt osztályozó, minden egyes osztályozó szavazatának egy kombinációját veszi, ahol az osztályozók szavazatainak súlya a pontosságuknak függvénye. A megfejelő algoritmust is kiterjeszthetjük valós értékek előrejelzésére.

7.9.3. | Elegendő csupán a pontosság ismerete egy-egy osztályozó megítéléséhez?

A pontosság mellett az osztályozókat a sebességük, robusztusságuk (azaz a zajos adatokon való pontosságuk), a skálázhatóságuk és az értelmezhetőségük szerint is összehasonlíthatjuk. A skálázhatóságot egy adott osztályozási algoritmus egyre nagyobb adathalmazokon történő I/O-műveletei számának becslésével értékelhetjük ki. Az értelmezhetőség szubjektív, habár objektív mérőszámokat is használhatunk, például az eredményül kapott osztályozó bonyolultsága (azaz döntési fáknál a csúcsok száma vagy a rejtett egységek száma neuronhálóknál és így tovább).

Léteznek a pontossági mértéknek alternatívái? Tegyük fel, hogy egy osztályozót arra tanítottunk be, hogy a betegek adatmintáit aszerint osztályozza, hogy azok *rákosak*-e

vagy *nem_rákosak*. Mondjuk egy 90%-os pontossági arány elfogadhatónak tűnik, de mi van akkor, ha jelenleg az adatmintáknak éppenséggel csak 3-4%-a *rákos*? Nyilvánvalóan nem biztos, hogy a 90%-os hibaarány elfogadható – például az osztályozó esetleg csak a *nem_rákos* mintákat osztályozza helyesen. Ehelyett azt szeretnénk felmérni, hogy az osztályozó mennyire jól ismeri fel a *rákos* mintákat (**pozitív mintákat**, ahogy hivatkozni szoktak rájuk), és mennyire jól ismeri fel a *nem_rákos* mintákat (**negatív mintákat**). Erre a célra az **érzékenységi**, illetve a **sajátságossági mértékeket** használhatjuk. Emellett a **precizitást** is alkalmazhatjuk azon *rákosnak* besorolt minták arányának a becslésére, amelyek valóban *rákosak*. Ezeket a mértékeket a következőképpen definiáljuk:

$$Érzékenység = \frac{t\_pos}{pos}, \tag{7.28}$$

$$Sajátságosság = \frac{t\_neg}{neg}, \tag{7.29}$$

$$Precizitás = \frac{t\_pos}{(t\_pos + f\_pos)}, \tag{7.30}$$

ahol *t_pos* a **valódi pozitív minták** száma (azon *rákos* mintáké, amelyeket jól osztályoztunk), *pos* a pozitív (*rákos*) minták száma, *t_neg* a **valódi negatív minták** száma (azon *nem_rákos* mintáké, amelyeket jól osztályoztunk), *neg* a negatív (*nem_rákos*) minták száma és *f_pos* a **téves pozitív minták** száma (azon *nem_rákos* mintáké, amelyeket helytelenül *rákosnak* ítéltünk). Megmutatható, hogy a pontosság az érzékenység és a sajátságosság függvénye:

$$Pontosság = érzékenység\frac{pos}{(pos + neg)} + sajátságosság\frac{neg}{(pos + neg)}. \tag{7.31}$$

Léteznek-e egyéb esetek, amikor a pontosság nem megfelelő? Az osztályozási feladatoknál általánosan elfogadott feltételezés szerint minden mintát egyértelműen osztályozhatunk, azaz minden tanuló minta csupán egyetlen osztályhoz tartozhat. Mégis, a nagy adatbázisokban lévő adatok változatosságának köszönhetően nem mindig ésszerű azt feltételeznünk, hogy az összes minta egyértelműen osztályozható. Helyesebb azzal a feltételezéssel élnünk, hogy egy minta nem csupán egy osztályhoz tartozhat. Akkor mégis miként mérhetjük az osztályozók pontosságát nagy adatbázisok esetében? A pontossági mérték nem megfelelő, hiszen nem veszi figyelembe a minták több osztályhoz való tartozásának lehetőségét.

Ahelyett, hogy egy osztályozási címkét adnánk vissza, hasznosabb egy valószínűségi osztály eloszlást visszaadnunk. A pontossági mértékek ekkor egy **második próba heurisztikát** alkalmazhatnak, amikor is egy osztályelőrejelzést helyesnek ítélünk, ha megegyezik az első vagy második legvalószínűbb osztállyal. Habár itt bizonyos fokig figyelembe vettük a minták nem egyértelmű osztályozását, ez nem teljes megoldás.

7.10. | Összefoglalás

- Az osztályozás és előrejelzés az adatelemzés két formája, amelyeket fontos adatosztályokat leíró vagy az adatok jövőbeli változásának irányvonalait becslő modellek készítésére használhatunk. Amíg az **osztályozás** kategorikus címkéket (osztályokat) ad meg, addig az **előrejelzés** folytonos értékű függvényeket modellez.
- Az adatoknak az osztályozásra és előrejelzésre való előkészítéséhez végzett előfeldolgozás magában foglalhatja az **adattisztítást**, a zaj csökkentését és a hiányzó értékek kezelését, a **fontossági elemzést**, a fölösleges vagy redundáns attribútumok eltávolítását és az **adatátalakítást**, például az adatok magasabb fogalmi szintekre való általánosítását vagy normalizálását.
- A becslési pontosság, számítási gyorsaság, robusztusság, skálázhatóság és értelmezhetőség az osztályozó és előrejelző módszerek kiértékeléséhez öt **kritériumot** ad.
- Az **ID3** és a **C4.5** mohó algoritmusok döntési fák indukciójára. Mindkét algoritmus egy információelméleti mértéket használ a fa levéltől különböző csúcsaihoz tartozó ellenőrző attribútumként. A **metszési algoritmusok** az adatban lévő zajt tükröző ágak eltávolításával próbálják meg javítani a pontosságot. Valamennyi korai döntésifa-algoritmus feltételezte, hogy az adatok elférnek a memóriában – ami korlátozza a nagy adatbázisokban történő adatbányászást. Azóta több olyan skálázható algoritmust is javasoltak, például a SLIQ-t, a SPRINT-et és a RainForestet, amelyek ezt a problémakört veszik célba. A döntési fák könnyen IF-THEN szabályokká alakíthatók.
- A **naiv Bayes-osztályozás** és a **Bayes-féle hihetőségi hálók** a *posteriori* eloszlás Bayes-tételére épülnek. A naiv Bayes-osztályozással szemben (ahol feltesszük az osztályok feltételes függetlenségét), a Bayes-féle hihetőségi hálóknál lehetőség nyílik a változók részhalmazai közötti feltételes függetlenségek definiálására.
- A **hiba-visszaterjesztéses algoritmus** neuronhálós algoritmus az osztályozásra, amely a csökkenő meredekségek módszerét alkalmazza. A súlyok olyan halmazát keresi, amelyek úgy modellezhetik az adatokat, hogy az átlagos négyzetes eltérés a háló osztályra vonatkozó becslése és az adatminták valódi osztályozási címkéje között minimális lesz. Szabályokat is kaphatunk a behangolt neuronhálókból, amelyekkel a megtanult hálózat érthetőségét növeljük.
- A **társítási bányászati technikák**, amelyek az adatokban gyakran előforduló mintázatokat keresik, osztályozási feladatokra is átalakíthatók.
- A legközelebbi szomszédokon alapuló osztályozók és az eset alapú következtetési osztályozók az osztályozás **előfordulás alapú** módszerei abban az értelemben, hogy az összes tanuló mintát a mintatérben tárolják. Emiatt mind a kettő hatékony indexelési technikákat igényel. A **genetikus algoritmusoknál** a szabályok populációi fejlődnek keresztezési és mutációs műveletek segítségével, amíg egy populáción belüli összes szabály el nem ér egy bizonyos küszöböt. A **közelítő halmazok elméletének** segítségével hozzávetőlegesen adhatunk meg osztályokat, amelyek a rendelkezésre álló attribútumok alapján nem különböztethetők meg egymástól. A **fuzzy halmazos** megközelítések a folytonos függvényeknél a küszöbök adta éles határokat helyettesítik fokozatos tagsági függvényekkel.

- Előrejelzéshez a **regresszió** lineáris, nemlineáris és általánosított lineáris modelljei használhatók. Sok nemlineáris probléma lineáris feladattá alakítható át a prediktor vagy előrejelzési változókon alkalmazott transzformációkkal.
- Az adattárházas technikákat, úgymint az attribútumorientált indukciót és a többdimenziós adatkockák használatát, egybefoglalhatjuk az osztályozási módszerekkel a gyors, **több szinten történő bányászás** érdekében. Az osztályozási feladatokat egy adatbányászati lekérdezőnyelv segítségével is megadhatjuk támogatva az interaktív adatbányászást.
- A **rétegezett *k*-szoros kereszt-megerősítéses** algoritmus az osztályozó pontosságának becslésére ajánlott eljárás. A **többszörös halmazolási** és **megfejelési módszereket** egyedi osztályozók egy sorozatának a behangolásával és kombinálásával a teljes osztályozási pontosság növelésére használhatjuk. Az **érzékenységi**, **sajátságossági** és **precizitási mértékek** hasznos alternatívái a pontossági mértéknek, különösen akkor, ha az érdeklődés középpontjában álló osztály kevesebb elemet tartalmaz.
- A különböző osztályozási eljárásokat számtalanszor összehasonlították, a problémakör azonban továbbra is kutatási terület. Egyetlen módszer sem teljesít jobban az összes többinél tetszőleges adathalmaz esetében. Olyan szempontokat kell figyelembe venni, mint a pontosság, hangolási idő, robusztusság, értelmezhetőség, skálázhatóság, ami kompromisszumokkal jár, és ez tovább bonyolítja az általánosan legjobb módszer keresésének a kérdéskörét. Kísérleti úton igazolták, hogy több módszer pontossága elegendően hasonló ahhoz, hogy az eltérés statisztikailag jelentéktelen legyen, ugyanakkor a hangolási idők lényegesen különbözőek lehetnek. Általánosságban a legtöbb neuronhálós, statisztikai osztályozó modell, beleértve a spline-okat, több számítást igényel, mint a legtöbb döntésifa-eljárás.

7.11. | Feladatok

7.1. Röviden ismertessük a *döntési fákon alapuló osztályozás* főbb lépéseit.

7.2. Miért hasznos a *fametszés* a döntési fa indukciónál? Mi a hátránya annak, ha egy külön mintahalmazt használunk a metszés kiértékelésére?

7.3. Adott döntési fa esetében a következő lehetőségekkel élhetünk: (a) a döntési fát először szabályokká alakítjuk, aztán a szabályokat metsszük; (b) metsszük a fát, az eredményt pedig szabályokká alakítjuk. Milyen előnyei vannak (a)-nak (b)-vel szemben?

7.4. Miért nevezzük a *naiv Bayes-osztályozót* naivnak? Röviden vázoljuk a naiv Bayes-osztályozás gondolatmenetét.

7.5. Vessük össze a *buzgó* osztályozás (például döntési fák, neuron- és Bayes-hálók) és a *lusta* osztályozás (például *k*-legközelebbi szomszéd, eset alapú következtetés) előnyeit és hátrányait.

7.6. A következő táblázat alkalmazotti adatbázisból származó tanuló adatokat tartalmaz. Az adatokat általánosítottuk. Adott sorra a *darab* oszlop azt adja meg, hogy hány olyan sor létezik, amelyek a *részleg*, *beosztás*, *életkor*, *fizetés* attribútumokon ugyanazzal az értékkel rendelkeznek, mint a szóban forgó sor.

részleg	beosztás	életkor	fizetés	darab
értékesítés	régi alkalmazott	31...35	46 000...50 000	30
értékesítés	kezdő	26...30	26 000...30 000	40
értékesítés	kezdő	31...35	31 000...35 000	40
rendszerek	kezdő	21...25	46 000...50 000	20
rendszerek	régi alkalmazott	31...35	66 000...70 000	5
rendszerek	kezdő	26...30	46 000...50 000	3
rendszerek	régi alkalmazott	41...45	66 000...70 000	3
marketing	régi alkalmazott	36...40	46 000...50 000	10
marketing	kezdő	31...35	41 000...45 000	4
titkárság	régi alkalmazott	46...50	36 000...40 000	4
titkárság	kezdő	26...30	26 000...30 000	6

Legyen a *beosztás* az osztályozási címke attribútum.

(a) Hogyan módosítanánk az ID3 algoritmust, hogy figyelembe vegye minden egyes általánosított adatsor *darab* attribútumát?

(b) Használjuk az ID3 módosított változatát a döntési fa elkészítéséhez.

(c) Adott a *részleg*, *fizetés*, *életkor* attribútumokon sorra *rendszerek*, 46 000...50 000 és 26...30 értékekkel rendelkező adatminta, mi lenne a *kezdő* naiv Bayes-osztályozása?

(d) Tervezzünk egy előrekapcsolt neuronhálót az adott adatokon. Címkézzük meg a bemeneti és kimeneti rétegek csúcsait.

(e) Az előbbi előrekapcsolt többrétegű neuronhálót használva mutassuk meg a hibavisszaterjesztéses algoritmus egy iterációja után a súlyok értékeit a következő tanuló sor esetén: (*értékesítés*, *régi alkalmazott*, 31...35, 46 000...50 000). Adjuk meg az általunk inicializált súlyértékeket, torzításokat és a tanulási rátát.

7.7. Adott k és n, ahol n a mintákat megadó attribútumok száma. Írjunk egy algoritmust a *k-legközelebbi* szomszéd alapján történő osztályozásra.

7.8. A következő táblázat tanulóknak az adatbázis tantárgyból évközi és év végi vizsgákon elért jegyeit tartalmazza.

X Évközi vizsga	Y Év végi vizsga
72	84
50	63
81	77
74	78
94	90
86	75
59	49
83	79
65	77
33	52
88	74
81	90

(a) Ábrázoljuk az adatokat. Ránézésre X és Y lineáris viszonyban állnak-e?

(b) A *legkisebb négyzetek módszerét* alkalmazva találjunk egy egyenlőséget, melyet használva előrejelezhetjük az év végi vizsgák jegyeit a tanulók évközi vizsgáit figyelembe véve.

(c) Becsüljük meg annak a tanulónak az év végi osztályzatát, aki 86-ost kapott év közben.

7.9. Némely *nemlineáris regressziós* modell lineárissá alakítható a prediktor változókon végzett átalakításokkal. Mutassuk meg, hogy az $Y = \alpha X^{\beta}$ egyenlőség lineárissá alakítható, amit a legkisebb négyzetek módszerével megoldhatunk.

7.10. Mi az a *megfejelés*? Mutassuk be, hogy miért javíthatja ez a döntési fa indukciós algoritmusát?

7.11. Mutassuk meg, hogy a pontosság az érzékenységnek és a sajátságosságnak a függvénye, vagyis bizonyítsuk a (7.31) egyenlőséget.

7.12. Az olyan esetekben, amikor egy adatobjektum egynél több osztályhoz is tartozhat, nehéz megbecsülni az osztályozás pontosságát. Ebben a helyzetben milyen kritériumokat alkalmazna ugyanazt az adatot modellező, egymástól különböző osztályozók összehasonlításánál?

7.12. | Irodalom

Több könyv is, például Weiss és Kulikowski [WK91], Michie, Spiegelhalter és Taylor [MST94], Langley [Lan96], Mitchell [Mit97] foglalkozik az osztályozásnak a gépi tanulás szempontjai szerinti bemutatásával több terület osztályozási és előrejelzési módszereit is összehasonlítva az osztályozók teljesítményének kiértékelésére szolgáló gyakorlati eljárásoknak az ismertetése mellett. E könyvek közül több is elmagyarázza az ebben a fejezetben tárgyalt osztályozási alapmódszereket. Michalski, Carbonell és Mitchell [MCM83, MCM86], Kodratoff és Michalski [KM90], Shavlik és Dietterich [SD90], illetve Michalski és Tecuci [MT94] gyűjteményekben a gépi tanulás területén íródott „ösztönző" hatású cikkek találhatók. A gépi tanulásnak adatbányászati alkalmazásokat szem előtt tartó bemutatását Michalski, Bratko és Kubat [MBK98] művében találhatják meg az érdeklődők.

A C4.5-öt Quinlan könyve [Qui93] magyarázza el. A könyv, ahogy Murthy [Mur98] döntési fák indukciójáról írt széles körű áttekintése is, a döntési fák indukciójához tartozó témakörök kiváló bemutatását adja. Más döntési fa indukciós algoritmusok a következők, kezdve a C4.5 ősével, az ID3-mal: (Quinlan [Qui86], CART [Breiman, Friedman, Olshen és Stone [BFOS84]), FACT (Loh és Vanichsetakul [LV88]), QUEST (Loh és Shih [LS97]), PUBLIC (Rastogi és Shim [RS98]), CHAID (Kass [Kas80] és Magidson [Mag94]). Az ID3 növekményes verziói a következők: ID4 (Schlimmer és Fisher [SF86a]), ID5 (Utgoff [Utg88]). Emellett az INFERULE (Uthurusamy, Fayyad, és Spangler [UFS91]) a döntési fákat nem túl jó adatok esetén is behangolja. A KATE (Manago és Kodratoff [MK91]) összetett szerkezetű adatok alapján hangolja a döntési fákat. A következő döntési fa algoritmusok az adatbányászat skálázhatóságát tartják szem előtt: SLIQ (Mehta, Agrawal és Rissanen [MAR96]), SPRINT (Shafer, Agrawal és Mechta [SAM96]), RainForest (Gehrke, Ramakrishnan és Ganti [GRG98]), BOAT (Gehrke, Ganti, Ramakrishnan és Loh [GGRL99]), Kamber, Winstone, Gong és mások [KWG+]. A [Cat91, CS93a, CS93b]

korábbi megközelítések leírásait adják. A döntési fa indukcióhoz javasolt attribútum kiválasztó mértékek összehasonlításához javasoljuk a következő cikkeket: Buntime és Niblett [NB92], Murthy [Mur98] és Shih [Shi00]. Az ilyen mértékek egy részletes tárgyalását adja Konnonenko és Hong [KH97] tanulmánya. Az attribútum (vagy főbb jellemző) szerkesztést Liu és Motoda [LM98a, LM98b] cikke írja le. Attribútumszerkesztést tartalmazó rendszerekre példák a következők: BACON (Langley, Simon, Bradshaw, Zytko [LSBZ87]), Stagger (Schlimmer [Schl87]), FRINGE (Pagallo [Pag89]), AQ17-DCI (Bloedorn és Michalski [BM98]).

A döntési fák metszésére rengeteg algoritmus létezik, beleértve ezekbe a költség bonyolultságán alapuló metszést (Breiman, Friedman, Olshen és Stone [BFOS84]), a csökkentett hiba metszést (Quinlan [Qui87]) és a pesszimista metszést (Quinlan [Qui86]). A PUBLIC (Rastogi és Shim [RS98]) a döntésifa-szerkesztést összekombinálja a metszéssel. MDL alapú metszési módszerek a következő munkákban találhatók: Quinlan and Rivest [QR89], Mehta, Agrawal és Rissanen [MRA95], Rastogi és Shim [RS98]. A többi eljárás között szerepelnek: Niblett és Bratko [NB86], Hosking, Pednault és Sudan [HPS97]. A metszési módszerek tapasztalati összehasonlítását megtaláljuk Mingers [Min89] és Malerba, Floriana és Semeraro [MFS95] munkákat. Az egyszerűsítő döntési fák áttekintéséhez olvassuk el Breslow és Aha [BA97] cikkét.

A szabályok döntési fákból való kikövetkeztetéséhez ajánljuk Quinlan [Qui87, Qui93] munkáit. A döntési fákból történő kikövetkeztetés helyett, szabályokat közvetlenül a tanuló adatokból is kinyerhetünk. Az ilyen algoritmusok között szerepelnek a következők: CN2 (Clark és Niblett [CN89]), AQ15 (Hong, Mozetic és Michalski [HMM86]), ITRULE (Smyth és Goodman [SG92]), FOIL (Quinlan [Qui90]), Swap-1 (Weiss és Indurkhya [WI98]). Szabályokat finomító stratégiákat, amelyek egy adott szabályhalmazból a legérdekesebbeket találják meg, Major és Mangano [MM95] cikkében láthatunk. Az attribútumorientált indukció és a döntési fa indukciójának egybeszerkesztésére a Kamber, Winstone, Gong és mások [KWG+] munkában tettek javaslatot. A 7.3.6. alfejezetben tárgyalt pontossági vagy osztályozási küszöböt Agrawal, Ghosh, Imielinski és mások [AGI+92] és Kamber és mások [KWG+] alkalmazták.

A Bayes-osztályozás részletes bemutatását találhatjuk Duda és Hart [DH73] mintafelismerési klasszikus tankönyvében, ugyanúgy a gépi tanulási tankönyvekben, mint például a Weiss és Kulikowski [WK91] és a Mitchell [Mit97] könyvekben. A naiv Bayes-osztályozó előrejelzési képességét vizsgálja Domingos és Pazzani [DP96] könyve abban a helyzetben, amikor is a feltételes függetlenség feltétele sérül. John [Joh97] munkájában a Gauss becslés helyett mag sűrűségi becslést (kernel density estimation) alkalmazott a folytonos értékű attribútumoknál. A Bayes-féle hihetőségi hálók témakörébe való bevezetéshez lapozzuk át Heckerman [Hec96] cikkét. Russell és Norvig [RN95] és Jensen [Jen96] munkáiban hihetőségi hálókon belüli következtető algoritmusokat találhatunk. A 7.4.4. alfejezetben tárgyalt csökkenő gradiensek módszerének Bayes-féle hihetőségi hálók tanítására való alkalmazását Russell, Binder, Koller és Kanazawa [RBKK95] adja meg. A 7.8. ábrán szereplő példát Russel és mások [RBKK95] cikkéből vettük. Azon alternatív stratégiák között, melyek a hihetőségi hálókat rejtett változókkal tanítják, megemlítendő az EM-algoritmus (Lauritzen [Lau95]). Adott megfigyelhető változók mellett a hihetőségi hálók szerkezetének a tanuló adatokból való tanulására szolgáló megoldásokat a [CH92, Bun94, HGC95] cikkekben javasoltak.

A hiba-visszaterjesztéses algoritmust Rumelhart, Hinton és Williams [RHW86] mutatta be. Azóta több változatot is javasoltak, többek között például: alternatív hibafüggvényeket (Hamson és Burr [HB88]), a háló szerkezetének dinamikus kiigazításait (Fahlman és Lebiere [FL90] és Le Cun, Denker, Solla [LDS90]), és a tanulási ráta és a momentum paraméterek dinamikus kiigazításait (Jacobs [Jac88]). Más változatokat tárgyal Chauvin és Rumelhart [CR95] cikke. A neuronhálókról írott könyvek közt szerepelnek a következők: [RM86, HN90, HKP91, Fu94, CR95, Bis95, Rip96]. Több, gépi tanulásról szóló könyvben [WK91, Mit97], szintén jó leírásai találhatók a hiba-visszaterjesztéses algoritmusnak. Számos technika létezik szabályoknak a neuronhálókból történő kikövetkeztetésére, ezekre példák a következők [SN88, Gal93, TS93, Fu94, Avn95, LSL95, CS96b, LGT97]. A 7.5. alfejezetben leírt szabály-kikövetkeztetési módszer Lu, Setiono és Liu [LSL95] cikkén alapult. A neuronhálókból való szabály-kikövetkeztetési technikák kritikái Craven és Shavlik [CS97] munkájában találhatók. Roy [Roy00] megfigyelése szerint a neuronhálók elméleti alapjai tévesek azon feltevéseket illetően, melyek arra vonatkoztak, hogy a kapcsolatokon alapuló tanulás miként modellezi az agyat. A neuronhálóknak ipari, üzleti, tudományos alkalmazásainak egy széles áttekintését adja Widrow, Rumelhart és Lehr [WRL94] cikke.

A 7.6. alfejezetben leírt ARCS-rendszert Lent, Swami és Widom [LSW97] javasolták, és a 6. fejezetben szintén bemutatásra került. A társítási tanulás módszerét Liu, Hsu és Ma [LHM98] javasolták. A felbukkanó mintákat használó CAEP-osztályozót Dong és Li javasolták. A JEP-osztályozót az ugró, felbukkanó mintákkal Li, Dong és Ramamoharanao [LDR00] mutatták be. Meretakis és Wüthrich [MW99] hosszú listák kibányászásához naiv Bayes-osztályozó létrehozását ajánlották.

A legközelebbi szomszédokon alapuló módszereket több statisztikai szöveg is tárgyalja, mint például Duda és Hart [DH73] valamint James [Jam85] könyvei. További információk találhatók a Cover és Hart [CH67] és a Fukonaga és Hummels [FH87] művekben. Eset alapú következtetésre való hivatkozásokat találhatunk a következő szövegekben: [RS89, Kol93, Lea96, AP94]. Genetikus algoritmusokról szóló írások a következők [Gol89, Mic92, Mit96]. A közelítő halmazokat Pawlak [Paw91] vezette be. A közelítő halmazok adatbányászati szerepének a tömör összegzését adják többek között a [Zia91, CPS98] művek. Több alkalmazásnál is az összetevők csökkentésénél és a szakértői rendszerek tervezésénél közelítő halmazokat használtak, mint például a [Zia91, LP97, Swi98] munkákban is. Az [SR92]-ben a redukáltak megtalálásának számítási igényeit csökkentő algoritmusokat javasoltak. A fuzzy logika általános leírásait találhatjuk meg a [Zad65, BS97, CPS98] művekben.

Sok jó tankönyv van, amely a regressziós technikák témakörét fedi le, például [Jam85, Dob90, JW92, Dev95, HC95, NKNW96, Agr96]. Press, Teukolsky, Vetterling és Flannery [PTVF96] könyve és az azzal együtt járó forráskód sok statisztikai eljárást tartalmaz, többek között a legkisebb négyzetek módszerét mind lineáris, mind többváltozós regresszió esetén. A jelenlegi nemlineáris regressziós modellek között megemlítendő a projekciókövetés és a MARS (Friedman [Fri95]). A loglineáris modellek a számítástudomány irodalmában multiplikatív modellek elnevezés alatt is ismeretesek. A loglineáris modellek számítástudományi szempontból vett bemutatásához tekintsük meg Pearl [Pea88] művét. A regressziós fák (Breiman, Friedman, Olshen és Stone [BFOS84]) teljesítménye sokszor összehasonlítható más regressziós módszerekével, különösen, ha a prediktor változók között sok magasabb rendű függőség is fennáll.

Adattisztítási és adatátalalakítási eljárásokat tárgyalnak: Kennedy, Lee, Van Roy és mások [KLV+], Weiss és Indurkhya [WI98], Pyle [Pyl99] munkái és e könyv 3. fejezete. Az osztályozók pontosságának becsléséhez tartozó témaköröket Weiss és Kulikowski [WK91] cikkében találhatunk. A 10-szeres kereszt-megerősítéses módszert az osztályozó pontosságának becslésére Kohavi [Koh95] elméleti és tapasztalati megfontolások alapján jobbnak találta a visszatartó, kereszt-megerősítéses, „hagyjunk-ki-egyet" (Stone [Sto74]) és az öntöltő (Efron és Tibshirani [ET93]) módszereknél. A halmazolást Breiman [Bre96] ajánlotta. Freund és Schapire öntöltő technikáját több különböző osztályozó esetében is alkalmazták, többek között: a döntési fa indukciónál (Quinlan [Qui96]) és a naiv Bayes-osztályozásnál. Az érzékenységi, sajátsági, precizitási mértékeket Frakes és Baeza-Yates [FBY92] tárgyalta.

Az Irvine-i Californiai Egyetem (University of California at Irvine UCI) az osztályozási algoritmusok kifejlesztéséhez és ellenőrzéséhez az adathalmazoknak egy Gépi Tanulás Adattárát tartja fenn. Az adattárról való információszerzéshez tekintsük meg a *http://www.ics.uci.edu/~mlearn/MLRepository.html* oldalt.

Egyik osztályozási módszer sem jobb a többinél mindenféle adattípust és értelmezési tartományt figyelembe véve. Az osztályozási módszerek tapasztalati összehasonlításait tartalmazzák a következő írások [Qui88, SMT91, BCP93, CM94, MST94, BU95, LLS00].

8. FEJEZET
Klaszterelemzés

Képzeljük el, hogy rendelkezésünkre áll adatelemek egy halmaza, amit elemezni szeretnénk, most azonban – az osztályozási feladattal ellentétben – nem ismerjük az egyes elemekhez tartozó osztályokat. A *klaszterezés* az adatoknak osztályokba, más szóval *klaszterekbe* való csoportosítását jelenti oly módon, hogy az egy klaszterbe tartozó elemek közötti hasonlóság nagy legyen, a különböző klaszterekbe tartozó elemek között pedig jelentősen különbözzenek egymástól. A hasonlóságot és a különbözőséget az elemeket jellemző attribútumok értékei alapján értelmezzük, leggyakrabban valamilyen mértéket véve alapul. A klaszterezés témaköre több önálló területből fejlődött ki, például az adatbányászatból, a statisztikából, a biológiából és a gépi tanulásból.

Ebben a fejezetben megismerjük majd azokat a követelményeket, amelyek nagy adatmennyiségre alkalmazott klaszterezési módszerek esetén merülnek fel. Azt is megvizsgáljuk, hogy hogyan lehet az elemek közötti különbözőséget kiszámítani különböző típusú attribútumok esetén. Bemutatunk néhány klaszterezési technikát, amelyeket a következő kategóriák valamelyikébe sorolhatunk: *particionáló módszerek*, *hierarchikus módszerek*, *sűrűségen alapuló módszerek*, *rács alapú módszerek*, illetve *modell alapú módszerek*. A klaszterezést a *szélsőséges értékek felismerésére* is használhatjuk, amiről a fejezet legutolsó részében lesz majd szó.

8.1. | A klaszterelemzés

Klaszterezésnek nevezzük azt a folyamatot, amikor absztrakt elemeket *hasonló* elemekből álló osztályokba csoportosítunk. Egy **klaszter** olyan adatelemek gyűjteménye, amelyek *hasonlóak* az ugyanazon klaszteren belüli elemekhez, és *különböznek* a többi klaszterben lévő elemtől. Számos alkalmazásban egy adatelemekből álló klasztert egy csoportként kezelhetünk.

A klaszterelemzés fontos emberi tevékenység. A korai gyermekkorban az ember úgy tanulja meg megkülönböztetni a macskákat a kutyáktól vagy a növényeket az állatoktól, hogy folyamatosan javítja a meglévő tudatalatti klaszterező sémáit. A klaszterelemzést széles körben alkalmazzák számos területen, például a mintafelismerés, az adatelemzés, a képfeldolgozás vagy a piackutatás területén. A klaszterezés segítségével azonosítani tudjuk a sűrű és a ritka területeket, ezáltal általános eloszlási mintákat, valamint az adatattribútumok között fennálló érdekes korrelációkat fedezhetünk fel.

Melyek a klaszterezés tipikus alkalmazási területei? Az üzleti életben a klaszterezés segíthet a piaci szakembereknek abban, hogy a vásárlók tömegében különálló csoportokat fedezzenek fel, és ezeket a csoportokat jellemezni tudják a vásárlási minták alapján. A biológiában a klaszterezést a növények és állatok rendszerezéséhez, a hasonló funkciójú gének osztályozásához használják, továbbá ennek segítségével mélyebb betekintés nyerhető a populációk belső, meglévő struktúráiba. A klaszterezés segíthet a hasonló felhasználású földterületek azonosításában egy földterület megfigyelő adatbázis esetén, a gépjármű-biztosítással rendelkezők közül a magas kárigényű ügyfelek csoportjainak azonosításában, egy városban a családi házak típus, érték és elhelyezkedés szerinti csoportjainak kialakításában, vagy éppen a weben lévő dokumentumok információtartalom szerinti osztályozásában. Adatbányászati funkcióját tekintve a klaszterelemzés önálló eszköz lehet, aminek segítségével betekintést nyerhetünk az adatok eloszlásába, jobban megismerhetjük az egyes klaszterek jellemzőit és a klaszterek speciális csoportját további elemzéseknek vethetjük alá. Másik lehetőségként a klaszterelemzést előkészítő lépésként használhatjuk olyan algoritmusok előtt, mint a karakterizáció vagy az osztályozás, amiket majd a kiválasztott klaszterekre fogunk alkalmazni.

Az adatok klaszterezése lendületesen fejlődő terület manapság. Olyan kutatási területek járulnak hozzá a fejlődéséhez, mint az adatbányászat, a statisztika, a gépi tanulás, a téradatbázisok, a biológia vagy a marketing. Az adatbázisokban összegyűjtött hatalmas mennyiségű adatnak köszönhetően az utóbbi időben a klaszterelemzés az adatbányászati kutatások egyik legaktívabb területévé vált.

A klaszterelemzést mint a statisztika egyik ágát már évek óta széles körben tanulmányozzák, főleg a *távolság alapú klaszterelemzésre* koncentrálva. A *k*-átlag, *k*-medoid és ehhez hasonló módszereken alapuló klaszterelemző eszközök be vannak építve sok statisztikai elemző szoftvercsomagba, mint például az S-Plus, az SPSS vagy a SAS. A gépi tanulás területén a klaszterezés jó példája a **felügyelet nélküli tanulásnak**. Az osztályozással ellentétben a klaszterezés és a felügyelet nélküli tanulás nem előre definiált osztályokra és betanító példákra alapoz. Éppen ezért a klaszterezés a **megfigyeléses tanulásnak** és nem a *példák alapján történő tanulásnak* egy formája. A **fogalmi klaszterezésben** elemeknek egy csoportja csak akkor alkot egy osztályt, ha az osztály leírható egy fogalommal. Ez eltér a hagyományos klaszterezéstől, ahol a hasonlóság mértéke a geometriai távolságon alapul. A fogalmi klaszterezés két lépésből áll: (1) először kialakítjuk a megfelelő osztályokat, majd (2) leírást készítünk minden osztályhoz, mint az osztályozás esetén. Az az alapelv, hogy az osztályokon belüli hasonlóság nagy legyen, az osztályok közötti pedig kicsi, itt is érvényes.

Az adatbányászat területén az egyik legfőbb cél olyan módszerek keresése, amelyek hatékony klaszterelemzést tesznek lehetővé *nagy adatbázisok* esetén is. Az aktuális kutatások a *skálázhatóságra* összpontosítanak, valamint arra, hogy *bonyolult alakzatok* és *adattípusok* esetén, illetve *többdimenziós* klaszterezés esetén is megfelelően hatékony módszereket találjanak. További probléma a nagy adatbázisokban előforduló *numerikus* és *nem numerikus adatok* egyidejű klaszterezése.

A klaszterezés izgalmas kutatási terület, ahol a lehetséges alkalmazások számos további követelményt állítanak elénk. A következőkben felsorolunk néhányat ezekből a követelményekből, mégpedig azokat, amelyek tipikusan az adatbányászati környezetben alkalmazott klaszterezés esetén merülnek fel.

- **Skálázhatóság** – Sok olyan klaszterező algoritmus létezik, amelyik kis adatmennyiség (kevesebb, mint 200 adatelem) esetén jól működik, egy nagy adatbázisban azonban nem ritkán több millió adatelem fordul elő. Ha egy nagy adathalmaznak csak egy *kiválasztott kis részére, mintájára* végezzük el a klaszterezést, téves eredményhez vezethet. Ezért van szükség jól skálázható klaszterező algoritmusokra.
- **Különböző típusú attribútumok kezelése** – Sok olyan algoritmust terveztek, amelyik intervallum alapú (numerikus) adatok klaszterezésére alkalmas. Az alkalmazások azonban gyakran egyéb adattípusok klaszterezését is megkívánják. Ilyen további adattípusok lehetnek például a bináris adatok, a felsorolás típusú vagy rendezett adatok, vagy ezek valamilyen keveréke.
- **Tetszőleges alakzattal rendelkező klaszterek felismerése** – Számos klaszterező algoritmus a klasztereit az euklideszi vagy a Manhattan-távolság alapján határozza meg. Az ilyen távolságon alapuló algoritmusok általában gömb alakú, hasonló méretű és sűrűségű klasztereket alakítanak ki. Egy klaszter azonban tetszőleges alakú lehet, ezért fontos olyan algoritmusok kifejlesztése, amelyek ilyenek felismerésére is képesek.
- **A bemenő paraméterek megadása minél kevesebb előismeretet feltételezzen** – Számos klaszterező algoritmus a felhasználótól vár bizonyos bemenő paramétereket, mint például a kívánt klaszterek száma. A klaszterezés eredménye jelentős mértékben függhet ezektől a paraméterektől. Ezeket a paramétereket azonban sokszor nem könnyű meghatározni, különösen olyan adathalmazok esetén, amelyek magas dimenziószámú elemeket tartalmaznak. Ez a probléma egyrészt gondot okoz a felhasználónak, másrészt nehezen ellenőrizhetővé teszi a klaszterezés minőségét.
- **Zajos adatok kezelése** – A legtöbb valós adatbázis tartalmaz szélsőséges értékeket, hiányzó, ismeretlen vagy hibás adatokat. Néhány klaszterező algoritmus érzékeny az ilyen adatokra, és ezek esetén gyenge minőségű klasztereket hoz létre.
- **A bemenő rekordok sorrendjétől való függetlenség** – Vannak olyan klaszterező algoritmusok, amelyek érzékenyek a bemenő adatok sorrendjére. Az ilyen algoritmus például ugyanahhoz az adathalmazhoz jelentősen eltérő klasztereket hoz létre, ha az adatokat más sorrendben kapja. Ezért fontos olyan algoritmusok létrehozása, amelyek nem érzékenyek a bemenő adatok sorrendjére.
- **Magas dimenziószám** – Egy adatbázis vagy adattárház jó néhány dimenziót, illetve attribútumot tartalmazhat. Számos klaszterező algoritmus jól kezeli az alacsony dimenziószámú adatokat, például a két- vagy háromdimenziós adatokat. Az emberi szem három dimenzióig jól tudja értékelni egy klaszterezés minőségét. A többdimenziós térbeli adatok klaszterezése már jóval nehezebb feladat, különös tekintettel arra, hogy az ilyen adatok nagyon ritkák és aszimmetrikusak is lehetnek.
- **Megszorításokon alapuló klaszterezés** – A valós életbeli alkalmazásoknak gyakran olyan klaszterezést kell alkalmazniuk, amelyek bizonyos megszorításoknak is megfelelnek. Képzeljük el például, hogy az a feladatunk, hogy egy városban adott számú bankjegykiadó automatát kell elhelyeznünk. A döntéshez klaszterezznénk a háztartásokat, miközben olyan típusú megszorításokat is figyelembe kellene vennünk, mint a város folyóinak és közútjainak elhelyezkedése, vagy az egyes körzetek vásárlói igényei. Érdekes feladatot jelent olyan adatcsoportok megkeresése, amelyek jól viselkednek klaszterezési szempontból, és kielégítik a megadott megszorításokat is.

- **Értelmezhetőség és felhasználhatóság** – A felhasználók azt várják a klaszterezés eredményétől, hogy értelmezhető, áttekinthető és használható legyen. Más szavakkal kifejezve, a klaszterezésnek szoros kapcsolatban kell állnia bizonyos szemantikai értelmezésekkel és alkalmazásokkal. Fontos annak tanulmányozása, hogy egy alkalmazás célja hogyan befolyásolhatja a klaszterező módszer kiválasztását.

A fenti követelményeket szem előtt tartva a klaszterelemzés tanulmányozását a következő lépésekben fogjuk elvégezni. Először áttekintjük a különböző adattípusokat, majd azt, hogy ezek hogyan befolyásolják a klaszterező módszereket. Ezt követően a módszerek általános osztályozását adjuk meg. Ezután minden egyes módszert megvizsgálunk részletesen. A módszerek között szerepelni fognak particionáló módszerek, hierarchikus módszerek, sűrűségen alapuló módszerek, rács alapú módszerek és modell alapú módszerek. Végül áttekintjük a magas dimenziószámú terek klaszterezését, valamint a szélsőséges értékek elemzését.

8.2. | Adattípusok a klaszterelemzés során

Ebben az alfejezetben áttekintjük azokat az adattípusokat, amelyek gyakran fordulnak elő a klaszterelemzés során, és azt is megnézzük, hogy ezek milyen előfeldolgozást igényelnek az elemzéshez. Tegyük fel, hogy a klaszterezendő adathalmaz n elemet tartalmaz, ahol az elemek lehetnek személyek, házak, dokumentumok, országok... stb. A memóriában dolgozó klaszterező algoritmusok általában a következő két adatszerkezet valamelyikét használják.

- **Adatmátrix** (más szavakkal *objektum–változó szerkezet*) – Ez a szerkezet n elemet reprezentál (például személyeket), mindegyiket p **változóval** (szokás ezeket *attribútumoknak* is nevezni). Az attribútumok lehetnek például életkor, magasság, szélesség, nem, fajta... stb. Az adatszerkezet formája egy relációs tábla, vagy más szóval egy $n \times p$ elemű mátrix (n elem $\times$ p változó):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}. \tag{8.1}$$

- **Különbözőségi mátrix** (más szavakkal *objektum–objektum szerkezet*) – Ez a szerkezet az n elemből alkotott párok közelségeit (távolságait) tárolja. Általában egy $n \times n$-es táblázattal ábrázoljuk, ahol $d(i, j)$ jelöli az i és j elemek közötti **különbözőséget** (távolságot):

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & \cdots & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}, \tag{8.2}$$

ahol $d(i, j)$ általában nem negatív szám, ami nagyon hasonló (illetve közeli) i és j elemek esetén közel van 0-hoz, és annál nagyobb lesz, minél jobban különböznek az elemek egymástól. Mivel $d(i, j) = d(j, i)$ és $d(i, i) = 0$, ezért kapjuk a (8.2) képletben látható mátrixot. A különbözőség mértékeit a szakasz további részében fogjuk tárgyalni.

Az adatmátrixot gyakran **két szempontú** mátrixnak is szokás nevezni, a különbözőségi mátrixot pedig **egy szempontúnak**, mivel az előbbinek a sorai és oszlopai különböző jelentéssel bírnak, az utóbbi esetén pedig azonos jelentésűek. A legtöbb klaszterező algoritmus a különbözőségi mátrixot használja. Ha az adataink egy adatmátrix formájában állnak rendelkezésre, akkor ezt először át kell alakítanunk különbözőségi mátrixszá, hogy ezeket az algoritmusokat alkalmazni tudjuk.

Jogosan merül fel a kérdés, hogy *hogyan tudjuk a d(i, j) különbözőséget megállapítani*. A szakasz további részében bemutatjuk, hogy hogyan lehet az elemek közötti különbözőséget kiszámolni olyan elemek esetében, amelyeket leíró változók típusa *intervallumváltozó*, *bináris változó*, *felsorolás típusú*, *rendezett* vagy *arányskálázott változó*, vagy a fentiek keveréke. Az így kiszámolt különbözőséget fogjuk később használni a klaszterek kialakításánál.

8.2.1. | Intervallumváltozók

Ebben a részben az *intervallumváltozókról* és azok standardizálásáról lesz szó. Ezután azokat a leggyakoribb távolsági mértékeket mutatjuk be, amelyeket ilyen változókkal jellemzett elemek különbözőségeinek kiszámolásához szoktak használni. E mértékek a következők: *euklideszi*, *Manhattan-*, illetve *Minkovszki-távolság*.

Nézzük először, hogy *mik is ezek az intervallumváltozók*? Az **intervallumváltozók** egy nagyjából lineáris skálán történő folytonos mérés eredményei. Tipikus példájuk a súly, a magasság, a szélességi és hosszúsági koordináták (például házak klaszterezése esetén) vagy az időjárási hőmérséklet.

A méréseknél használt mértékegység befolyásolni tudja a klaszterelemzés eredményét. Ha például méterről hüvelykre változtatjuk a magasság mérésénél használt mértékegységet, vagy kilogrammról fontra a súly méréséhez használt mértékegységet, teljesen más klaszterező struktúrához juthatunk. Általában igaz az, hogy ha egy változót kisebb mértékegységben fejezünk ki, akkor a változó által felvehető értékek értékhatára szélesebb lesz, és így a változónak nagyobb hatása lesz az eredményül kapott klaszterező struktúrára. Hogy a mértékegységektől való ilyen függőséget elkerüljük, az adatokat

standardizálni kell. A standardizálás megpróbál minden változónak azonos súlyt adni. Ez különösen fontos lehet akkor, ha semmilyen előismeretünk nincs az adatokról. Néhány esetben azonban a felhasználók szándékosan adhatnak nagyobb hangsúlyt egyes változóknak, mint a többinek. Ha például kosárlabdajátékos jelölteket kell klasztereznünk, akkor a testmagasság változónak valószínűleg nagyobb jelentőséget fogunk tulajdonítani.

Hogyan standardizáljuk egy változó adatait? A mérési eredmények standardizálásának egyik lehetséges módja, hogy átalakítjuk az eredeti értékeket egy mértékegység nélküli változóba. Ha adottak az f változó értékei, akkor ezt a következőképpen tehetjük meg.

(1) Kiszámoljuk az **átlagos abszolút eltérést**, s_f-et:

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|\right), \tag{8.3}$$

ahol x_{1f}, ... x_{nf} az f változó n darab értéke, m_f pedig f átlaga, vagyis:
$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf})$.

(2) Kiszámoljuk a **standardizált értékeket**, amit más szóval **z-értéknek** is nevezünk.

$$z_{if} = \frac{x_{if} - m_f}{s_f}. \tag{8.4}$$

Az s_f átlagos abszolút eltérés kevésbé érzékeny a szélsőséges értékekre, mint a σ_f hagyományos szórás, mivel annak kiszámolásakor az átlagtól való eltéréseket (vagyis $|x_{if} - m_f|$-eket) nem emeljük négyzetre. Ezzel a szélsőséges értékek hatását némiképp csökkentjük. Vannak további mértékek, amelyek még kevésbé érzékenyek a szétszóródásra, ilyen például a *mediántól való abszolút eltérés*. Az átlagos abszolút eltérésnek azonban megvan az az előnye, hogy a szélsőséges értékekre vonatkozó z-értékek nem lesznek túlságosan kicsik, és így a szélsőséges értékek még azonosíthatók maradnak.

A standardizálás lehet hasznos vagy kevésbé hasznos egy adott alkalmazásban, így annak eldöntése, hogy éljünk-e vele, és ha igen, melyik módszert használjuk, a felhasználó feladata kell legyen. A standardizálási módszereket a 3. fejezetben már tárgyaltuk, amikor a normalizálásról és az előfeldolgozási módszerekről volt szó.

Rendben van, most már *standardizáltuk az adatokat, de ezután vajon hogyan tudjuk kiszámolni az elemek közötti különbözőségeket?* A standardizálás után, vagy esetleg bizonyos alkalmazásokban e nélkül az intervallumváltozókkal megadott elemek közötti különbözőséget általában az elempárok közötti távolság alapján számítjuk ki. A legelterjedtebb távolsági mérték az **euklideszi távolság**, amelyet a következőképpen definiálunk:

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2}, \tag{8.5}$$

ahol $i = (x_{i1}, x_{i2}, \ldots x_{ip})$ és $j = (x_{j1}, x_{j2}, \ldots x_{jp})$ két p-dimenziós adatelem. Egy másik jól ismert mérték a **Manhattan-távolság**, amit a következő képlet definiál:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|. \tag{8.6}$$

Az iménti két távolságdefiníció mindegyike kielégíti a távolságfüggvényekre vonatkozó következő matematikai elvárásokat:

(1) $d(i, j) \geq 0$: A távolság nem negatív.
(2) $d(i, i) = 0$: Egy elem önmagától vett távolsága egyenlő 0-val.
(3) $d(i, j) = d(j, i)$: A távolság szimmetrikus függvény.
(4) $d(i, j) \leq d(i, h) + d(h, j)$: Az i-ből j-be vezető közvetlen út nem lehet hosszabb az i-ből h-ba és h-ból j-be vezető utak összegénél *(háromszög egyenlőtlenség)*.

A **Minkovszki-távolság** tulajdonképpen az euklideszi és a Manhattan-távolság általánosítása. A definíciója a következő:

$$d(i,j) = \left(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{ip} - x_{jp}|^q\right)^{1/q}, \tag{8.7}$$

ahol q pozitív egészet jelöl. $q = 1$ esetén a Manhattan-távolságot kapjuk, míg $q = 2$ esetén az euklideszi távolságot.

Ha az egyes változókhoz súlyokat rendelünk a fontosságuknak megfelelően, akkor az alábbi úgynevezett *súlyozott* euklideszi távolságot kapjuk:

$$d(i,j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \cdots + w_p|x_{ip} - x_{jp}|^2}. \tag{8.8}$$

A súlyozást a Manhattan-távolságra és a Minkovszki-távolságra is alkalmazhatjuk.

8.2.2. | Bináris változók

Ebben az alfejezetben azt vizsgáljuk meg, hogy hogyan lehet a különbözőséget kiszámolni *szimmetrikus* vagy *aszimmetrikus bináris változók* által leírt elemek esetén.

Egy **bináris változónak** csak két állapota lehet: 0 vagy 1. A 0 azt jelenti, hogy a változó nincs beállítva, az 1 pedig azt, hogy be van állítva. Ha például egy páciens jellemzői között szerepel a *dohányzik* nevű változó, akkor az 1 érték azt jelenti, hogy az illető dohányzik, a 0 pedig azt, hogy nem dohányzik. Ha a bináris változókat úgy kezelnénk, mintha intervallumváltozók lennének, félrevezető klaszterezési eredményekhez vezethet. Ezért a bináris változók esetén más módszerek szükségesek a különbözőségek kiszámításához.

Nézzük meg tehát, hogy *hogyan számolhatjuk ki a különbözőséget két bináris változó esetén*. Az egyik megközelítés egy különbözőségi mátrix kiszámolását igényli a megadott bináris adatok alapján. Ha minden bináris változóhoz ugyanazt a súlyt rendeljük, akkor a 8.1. táblázatban látható 2 × 2-es kontingenciatáblázatot kapjuk. A táblázatban q jelöli azoknak a változóknak a számát, amelyek az i és j elemekre is 1 értékkel rendelkeznek,

r jelöli azoknak a számát, amelyek i elem esetén 1, j esetén pedig 0 értékkel rendelkeznek, s jelöli azoknak a számát, amelyek i esetén 0, j esetén 1 értékkel rendelkeznek, végül pedig t jelöli az olyan változók számát, amelyek mindkét elem esetén 0 értékkel rendelkeznek. Az összes változó száma p, így $p = q + r + s + t$.

8.1. táblázat | Kontingenciatáblázat bináris változókra

		j elem		
		1	0	Összesen
i elem	1	q	r	$q + r$
	0	s	t	$s + t$
	Összesen	$q + s$	$r + t$	p

Nézzük meg, hogy *mi a különbség a szimmetrikus és az aszimmetrikus változók között*. Egy bináris változót **szimmetrikusnak** nevezünk, ha mindkét állapota azonos értékű és egyforma súllyal rendelkezik, vagyis nincs jelentősége, hogy melyik állapotot kódoljuk 1-gyel és melyiket 0-val. Ilyen például a *nem* változó, amelynek értékei férfi és nő. Az olyan hasonlóságot, amelyik szimmetrikus bináris változókon alapul, **invariáns hasonlóságnak** nevezzük. Ilyen esetben az eredmény nem változik, ha a változók egy részének kódolását megváltoztatjuk. Invariáns hasonlóság esetén az i és j elem közötti különbözőség kiszámítására a legismertebb az úgynevezett **egyszerű egyezőségi tényező**, amit a következőképpen definiálunk:

$$d(i,j) = \frac{r+s}{q+r+s+t}. \tag{8.9}$$

Egy bináris változót **aszimmetrikusnak** nevezünk, ha a lehetséges állapotok nem egyformán fontosak. Ilyen például egy betegségi teszt *pozitív* vagy *negatív* eredménye. Szokás szerint a legfontosabb állapotot (ami általában a legritkább is) kódoljuk 1-gyel (például *HIV-pozitív*) és a másikat 0-val (például *HIV-negatív*). Két aszimmetrikus bináris változó esetén sokkal jelentősebb a számunkra, ha mindkettő értéke 1 (pozitív egyezőség), mintha mindkettő értéke 0 lenne (negatív egyezőség). Ezért az ilyen bináris változókat gyakran unárisnak tekintjük, mintha csak egy állapotuk lenne. Az ilyen változók közötti hasonlóságot **nem invariáns hasonlóságnak** nevezzük. Nem invariáns hasonlóság esetén a legismertebb képlet az alábbi úgynevezett **Jaccard-tényező**, amelynél a negatív egyezőségek számát, t-t nem tekintjük fontosnak, és a számításnál nem is vesszük figyelembe:

$$d(i,j) = \frac{r+s}{q+r+s}. \tag{8.10}$$

Ha az adathalmazban szimmetrikus és aszimmetrikus változók is szerepelnek, akkor a 8.2.4. alfejezetben leírt, vegyes változók esetén használható megközelítést alkalmazhatjuk.

8.1. példa | **Különbözőség bináris változók esetén.** Tegyük fel, hogy egy pácienseket tartalmazó tábla (8.2. táblázat) a következő attribútumokat tartalmazza: *név*, *nem*, *láz*, *köhögés*, *teszt_1*, *teszt_2*, *teszt_3*, *teszt_4*, ahol a *név* az elem azonosítója, a *nem* szimmetrikus attribútum, a többi attribútum pedig bináris aszimmetrikus.

8.2. táblázat | Relációs tábla, amely főként bináris attribútumokat tartalmaz

név	nem	láz	köhögés	teszt_1	teszt_2	teszt_3	teszt_4
Jack	F	I	N	P	N	N	N
Mary	N	I	N	P	N	P	N
Jim	F	I	I	N	N	N	N
...	...	...	...	...	...	...	...

Az aszimmetrikus attribútum értékekre legyen az *I (igen)* és a *P (pozitív)* értékek kódolása 1, a *N (nem* vagy *negatív)* kódolása pedig 0. Tegyük fel azt is, hogy az elemek (páciensek) közötti távolságok kiszámolásánál csak az aszimmetrikus változókat vesszük figyelembe. A Jaccard-tényező képlete (8.10) alapján a három páciens, Mary, Jack és Jim közötti páronkénti távolságok a következőképpen adódnak:

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0{,}33, \tag{8.11}$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0{,}67, \tag{8.12}$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0{,}75. \tag{8.13}$$

A fenti értékek azt sugallják, hogy nem valószínű, hogy Jimnek és Marynek hasonló betegsége lenne, mivel rájuk a legnagyobb a különbözőségi érték. Ugyanilyen meggondolásból látható, hogy Jack és Mary esetén a legvalószínűbb a hasonló betegség. ■

8.2.3. | Felsorolás típusú, rendezett és arányskálázott változók

Ebben a részben azt vizsgáljuk meg, hogy hogyan lehet az olyan elemek közötti különbözőséget kiszámolni, amelyeket leíró változók *felsorolás típusúak*, *rendezettek* vagy *arányskálázottak*.

Felsorolás típusú változók

A felsorolás típusú változó a bináris változó általánosítása, de egy ilyen változó több mint két állapotú lehet. Például a *térkép_szín* olyan felsorolás típusú változó, amelyik mondjuk a következő öt értéket veheti fel: *piros*, *sárga*, *zöld*, *rózsaszín*, *kék*.

Legyen egy felsorolás típusú változó állapotainak száma M. Az állapotokat jelölhetjük

betűkkel, szimbólumokkal vagy egész számokkal, például 1, 2, ..., M. Fontos megjegyezni, hogy a fenti számok semmilyen speciális sorrendet nem jelölnek.

Nézzük meg, hogy *hogyan számolhatjuk ki a különbözőséget felsorolás típusú változók által leírt elemek esetén*. Az i és j elemek közötti különbözőséget kiszámolhatjuk az **egyszerű egyezőség elve** alapján:

$$d(i,j) = \frac{p-m}{p}, \tag{8.14}$$

ahol m az *egyezőségek* száma, vagyis azoknak a változóknak a száma, amelyekre i és j *megegyezik*, p pedig az összes változónak a száma. Az m hatását súlyok hozzárendelésével növelhetjük, de akár azokhoz a változókhoz is rendelhetünk nagyobb súlyt, amelyeknek több állapota van.

A felsorolás típusú változókat kódolhatjuk aszimmetrikus bináris változókkal úgy, hogy az M állapot mindegyikéhez egy bináris változót hozunk létre. Egy adott állapottal rendelkező elem esetén az ezt az állapotot jelölő bináris változó értéke 1 lesz, a többi bináris változó értéke pedig 0. Például ha a *térkép_szín* nevű változót kódoljuk, akkor a fent említett öt szín mindegyikéhez egy bináris változót hozhatunk létre. A *sárga* színnel rendelkező elem esetén a *sárga* változó értéke 1 lesz, a többi négy értéke pedig 0. Ilyen kódolás esetén a különbözőségek a 8.2.2. alfejezetben tárgyalt módszerekkel számíthatók ki.

Rendezett változók

Egy **diszkrét, rendezett változó** hasonlít a felsorolás típusú változóra, de most az M állapot egy értelmes, jelentéssel bíró sorba van rendezve. A rendezett változók nagyon jól használhatók olyan szubjektív értékelések rögzítésére, amelyek nem mérhetők objektívan. A szakmai rangokat például gyakran adják meg szekvenciális sorrendben, mint például adjunktus, docens, egyetemi tanár. A **folytonos, rendezett változó** olyan, mint egy ismeretlen skálával rendelkező, folytonos adatokból álló halmaz. Ez azt jelenti, hogy az értékek relatív sorrendje alapvető fontosságú, de a tényleges nagyságuk mellékes. Például a relatív sorrend (arany, ezüst, bronz) egy adott sportágban általában sokkal fontosabb, mint a ténylegesen mérhető, elért eredmények. Rendezett változókat kaphatunk az intervallumváltozók diszkretizálásával is úgy, hogy a lehetséges intervallumot véges sok osztályra bontjuk. Egy rendezett változó értékeihez hozzárendelhetünk rangszámokat is, például ha egy ilyen valtozónak M_f lehetséges állapota van, akkor a rangszámok a következők: 1, ..., M_f.

Nézzük meg, hogy *hogyan kezelhetjük a rendezett változókat*. Ezeknek a változóknak a kezelése nagyon hasonlít az intervallumváltozókéhoz a különbözőség kiszámítását tekintve. Tegyük fel, hogy f az egyike az n elemet leíró rendezett változóknak. Ekkor az f-re vonatkozó különbözőség kiszámítása a következő lépések végrehajtását igényli:

(1) f értéke az i-edik elemre legyen x_{if}, és f-nek M_f rendezett állapota van, amelyekhez tartozó rangszámok: 1, ..., M_f. Helyettesítsük mindegyik x_{if}-et a megfelelő rangszámmal, $r_{if} \in \{1, ..., M_f\}$.

(2) Mivel az egyes sorrendi változók különböző számú állapottal rendelkeznek, ezért a változók értékeinek tartományát le kell képezni a [0, 1] intervallumra, hogy a változók azonos súlyt kapjanak. Ezt úgy érhetjük el, hogy az r_{if} rangszámot lecseréljük a következővel:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \tag{8.15}$$

(3) Ezután a különbözőséget úgy számolhatjuk ki, hogy a 8.2.1. alfejezetben az intervallumváltozókra leírt bármelyik távolsági mértéket alkalmazzuk úgy, hogy most már az f változó i-edik elemhez tartozó értékét z_{if} jelöli.

Arányskálázott változók

Az arányskálázott változók egy nemlineáris skálán (például exponenciális skálán) történő pozitív mérést jelentenek és közelítőleg az alábbi képlettel adhatók meg:

$$Ae^{Bt} \text{ vagy } Ae^{-Bt}, \tag{8.16}$$

ahol A és B pozitív konstansok. Jó példa lehet ilyen változóra egy baktériumtenyészet növekedése, vagy egy radioaktív elem lebomlása.

Vizsgáljuk meg, *hogyan lehet a különbözőséget kiszámolni ilyen változók által leírt elemek esetén*. Erre három módszer is ismeretes.

(1) Kezeljük az arányskálázott változókat úgy, mint az intervallumváltozókat. Ez azonban sok esetben nem túl jó választás, mert a skála torzított is lehet.
(2) Alkalmazzunk **logaritmikus transzformációt** az f, arányskálázott változóra. Ha f eredeti értéke az i elemre x_{if} volt, akkor az új érték $y_{if} = \log(x_{if})$. Az y_{if} értékeket már intervallumváltozóként kezelhetjük a 8.2.1. alfejezetben leírt módon. Megjegyezzük, hogy az alkalmazástól függően bizonyos arányskálázott változókra log-log vagy más hasonló transzformációt kell alkalmazni.
(3) Kezeljük az x_{if}-eket úgy, mint folytonos, rendezett adatokat, és tekintsük a rangszámait intervallum értékűnek.

A két utóbbi módszer tekinthető a leghatékonyabbnak, bár a módszer megválasztása minden esetben az adott alkalmazástól függ.

8.2.4. | Vegyes típusú változók

A 8.2.1–8.2.3. alfejezetekben azt vizsgáltuk, hogy hogyan lehet az elemek közötti különbözőséget kiszámolni olyan esetekben, amikor a leíró változók azonos típusúak. Ezek a típusok lehettek *intervallum alapúak*, *szimmetrikus* vagy *aszimmetrikus bináris*, *felsorolás*, *rendezett* vagy *arányskálázott* típusok. A valós adatbázisokban azonban az elemek sokszor vegyes típusú változókkal vannak megadva. Egy adatbázis általában mind a hat fenti típusú változót tartalmazza.

Hogyan lehet tehát a vegyes típusú változókkal megadott elemek közötti különbözőséget kiszámolni? Az egyik lehetséges megközelítés, hogy csoportosítsuk össze az azonos típusú változókat, és végezzünk külön-külön klaszterelemzést az egyes változótípusokra. Ez akkor megvalósítható, ha az egyes elemzések összeegyeztethető eredményre vezetnek. A valós alkalmazásokban azonban eléggé valószínűtlen, hogy a változótípusonkénti különálló klaszterelemzések egymással összhangba hozható eredményt adnának.

Talán jobb megközelítés, hogy az összes változótípust együtt dolgozzuk fel, egyetlen klaszterelemzést végezve. Az egyik alkalmazható technika úgy működik, hogy a különböző változókat egyetlen különbözőségi mátrixba tesszük be úgy, hogy az összes változót a [0, 1] intervallumnak egy közös részére képezzük le.

Tegyük fel, hogy az adathalmazhoz p darab változó tartozik, amelyek vegyes típusúak. Az i és j elemek közötti különbözőséget a következő képlettel definiáljuk:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}, \tag{8.17}$$

ahol a $\delta_{ij}^{(f)}$ indikátor értéke 0-val egyenlő, ha (1) x_{if} és x_{jf} valamelyike hiányzik, azaz f nem tartalmaz értéket i és j valamelyikére vonatkozóan; vagy (2) $x_{if} = x_{jf} = 0$ és f aszimmetrikus bináris változó. Minden más esetben $\delta_{ij}^{(f)} = 1$. Az f változónak a hozzájárulását az i és j elemek közötti különbözőséghez $d_{ij}^{(f)}$ jelöli, amit a következőképpen kell kiszámítani (f típusától függően):

- Ha f bináris vagy felsorolás típusú: $d_{ij}^{(f)} = 0$ ha $x_{if} = x_{jf}$, egyébként $d_{ij}^{(f)} = 1$.
- Ha f intervallum alapú: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$, ahol h azokon az elemeken fut végig, amire f-nek van értéke.
- Ha f rendezett vagy arányskálázott: számoljuk ki az r_{if} rangszámokat, és a $z_{if} = \frac{r_{if} - 1}{M_f - 1}$-eket és tekintsük z_{if}-eket intervallum alapúnak.

A fenti módszerrel az elemek közötti különbözőséget akkor is ki tudjuk számolni, ha a leíró változók típusa különböző.

8.3. | A legfontosabb klaszterező módszerek osztályozása

A szakirodalomban számos klaszterező algoritmus ismeretes. Az algoritmus kiválasztása függ az adatok típusától, az alkalmazástól és az elérni kívánt céltól. Ha a klaszterelemzés leíró vagy feltáró célokat szolgál, akkor célszerű lehet több módszert is kipróbálni ugyanazokra az adatokra, hogy lássuk, mi rejlik bennük.

A főbb klaszterező módszereket a következő kategóriák valamelyikébe sorolhatjuk:

- **Particionáló módszerek** – Ha adott egy n elemből (adatsorból) álló adatbázis, akkor egy particionáló módszer k partíciót fog létrehozni az adatokból, ahol $k \leq n$, és minden partíció egy klaszternek felel meg. Ez azt jelenti, hogy az adatokat k csoportba osztjuk úgy, hogy teljesüljenek a következő feltételek: (1) minden csoport tartalmaz legalább egy elemet, és (2) minden elem pontosan egy csoportba tartozik. Megjegyezzük, hogy ez utóbbi feltételt szokás enyhíteni néhány, úgynevezett fuzzy particionálási technika esetén. Ilyen technológiákra majd az irodalomjegyzékben adunk utalásokat.

  Ha adott a létrehozandó partíciók száma k, akkor egy particionáló módszer először létrehoz egy kezdeti particionálást. Ezután valamilyen **iteratív áthelyezési technika** segítségével megpróbál javítani a particionáláson úgy, hogy elemeket helyez át egyik csoportból a másikba. Egy particionálást általában akkor tekintünk jónak, ha az egy klaszterbe eső elemek „közeliek" (vagy hasonlók) egymáshoz, míg a különböző klaszterbeliek távoliak (vagy különbözők). Egy particionálás minőségének elbírálásához vannak további szempontok is.

  Ahhoz, hogy globális optimumhoz jussunk egy particionálás alapú klaszterezésnél, az összes lehetséges particionálást fel kellene sorolnunk. Ehelyett a legtöbb alkalmazás a következő két heurisztikus módszer egyikét alkalmazza: (1) k-átlag algoritmus, amelynél minden klasztert a klaszterbeli elemek átlaga reprezentál; (2) k-medoid algoritmus, amelynél minden klasztert egy olyan elem reprezentál, amelyik közel van a klaszter közepéhez. Ezek a heurisztikus módszerek jól működnek olyan esetekben, amikor gömb alakú klasztereket kell megtalálnunk kicsi vagy közepes méretű adatbázisokban. Bonyolult alakú klaszterek megkereséséhez, és nagyon nagy adathalmazok klaszterezéséhez a particionáláson alapuló módszereket valamilyen módon ki kell terjesztenünk. A particionáláson alapuló módszereket részletesebben a 8.4. alfejezetben tárgyaljuk.
- **Hierarchikus módszerek** – A hierarchikus módszerek az adathalmaznak egy hierarchikus dekompozícióját hozzák létre. A hierarchikus módszereket *egyesítő*, illetve *felosztó* módszerekre oszthatjuk aszerint, hogy hogyan történik a hierarchikus dekomponálás. Az *egyesítő megközelítés*, amelyet szoktak még *lentről fölfelé* módszernek is nevezni, úgy működik, hogy kezdetben minden elem egy önálló csoportot alkot. Ezután folyamatosan egyesítjük az egymáshoz közeli csoportokat addig, amíg az összes csoport eggyé nem olvad (ez a hierarchia legfölső szintje), vagy amíg a befejeződési feltétel nem teljesül. A *felosztó megközelítés*, amelyet szoktak még *föntről lefelé* módszernek is nevezni, úgy működik, hogy kezdetben minden elem ugyanabban a klaszterben van. A további iteratív lépésekben egy klasztert mindig kisebb klaszterekre bontunk addig, amíg minden elem külön klaszterbe nem kerül, vagy amíg a befejeződési feltétel nem teljesül.

  A hierarchikus módszereknek megvan az a hátránya, hogy ha egy műveletet (egyesítést vagy felosztást) elvégeztünk, akkor azt a későbbiekben már nem tudjuk semmissé tenni. Ez a merevség annyiban hasznos, hogy ez kisebb számítási költséget eredményez, és nem kell a választási lehetőségek kombinatorikusan nagy száma miatt aggódnunk. Egy nagy problémája azonban az ilyen technikáknak, hogy a rossz döntéseket nem tudják javítani. A hierarchikus klaszterezés minőségének javítására két megközelítés ismeretes: (1) az elemek közötti kapcsolatokat alaposan elemezzük minden hierarchikus particionáló lépés előtt, mint például a CURE és Chameleon algoritmusok-

ban; (2) a hierarchikus egyesítést és az iteratív áthelyezést együttesen alkalmazzuk úgy, hogy először a hierarchikus egyesítő algoritmust használjuk, majd az eredményt az iteratív áthelyezés segítségével finomítjuk, mint például a BIRCH algoritmusban. A hierarchikus klaszterező módszereket a 8.5. alfejezetben tárgyaljuk.

- **Sűrűségen alapuló módszerek** – A legtöbb particionáló módszer az elemek közötti távolság alapján klaszterezi az adatokat. Az ilyen módszerek csak a gömb alakú klasztereket képesek megtalálni, és nehézségekbe ütköznek a másfajta alakú klaszterek felismerésekor. Olyan klaszterező algoritmusokat is kifejlesztettek, amelyek a *sűrűség* fogalmán alapulnak. Ezeknek az algoritmusoknak az az alapötlete, hogy addig növelik az adott klasztert, amíg a környezetének a sűrűsége (az ott lévő elemek száma) meghalad egy bizonyos küszöböt. Ez azt jelenti, hogy egy klaszter minden pontjának adott sugarú környezetében legalább minimálisan megadott adatpontnak kell lennie. Az ilyen módszerek képesek a zajok (szélsőséges értékek) kiszűrésére és tetszőleges alakú klaszterek felismerésére.

  A DBSCAN egy jellemző példája a sűrűség alapú módszereknek, ez is egy sűrűségi küszöb alapján növeli a klasztereket. Az OPTICS szintén sűrűség alapú módszer, amelyik egy megnövelt klaszterező elrendezést számol ki automatikus és interaktív esetben egyaránt. A sűrűségen alapuló módszereket a 8.6. alfejezetben tárgyaljuk majd.
- **Rács alapú módszerek** – A rács alapú módszerek felosztják az elemek terét véges számú cellára, amelyek egy rácsszerkezetet alkotnak. Az összes klaszterező műveletet erre a rácsszerkezetre hajtjuk végre. Ennek a megközelítésnek a legnagyobb előnye a gyors végrehajtási idő, amely független az adatelemek számától, és csak az egyes dimenziókban megadott cellák számától függ.

  A STING algoritmus egy jó példája a rács alapú módszereknek. A CLIQUE és a Wave-Cluster algoritmusok mindegyike egyidejűleg rács alapú és sűrűség alapú. A rács alapú módszereket a 8.7. alfejezetben tárgyaljuk.
- **Modell alapú módszerek** – A modell alapú módszerek minden klaszterről egy modellt feltételeznek, és megkeresik az adott modellhez legjobban illeszkedő adatokat. Egy modell alapú algoritmus úgy helyezi el a klasztereket, hogy egy sűrűség függvényt hoz létre, amelyik az adatpontok térbeli eloszlását tükrözi. Ez a módszer egyben automatikusan meg is határozza a klaszterek számát a hagyományos statisztikákra alapozva, a zajokat vagy szélsőséges értékeket is figyelembe véve, és így egy robusztus klaszterező módszert eredményez. A modell alapú klaszterezést a 8.8. alfejezetben ismertetjük majd.

Vannak olyan klaszterező algoritmusok, amelyek több módszer ötletét is egyesítik, így néha nehéz egy adott algoritmust egyértelműen csak az egyik klaszterező módszerhez sorolni. Ezenkívül bizonyos alkalmazásoknak lehetnek olyan elvárásaik a klaszterezéssel kapcsolatosan, amelyek több technika egyidejű alkalmazását igénylik.

A következő alfejezetekben az imént ismertetett öt klaszterező módszer mindegyikét kicsit részletesebben fogjuk bemutatni. Olyan algoritmusokat is bemutatunk majd, amelyek több módszer ötletét is egyesítik. A szélsőséges értékek elemzését, ami szintén a klaszterezéssel kapcsolatos téma, a 8.9. alfejezetben fogjuk tárgyalni.

8.4. | Particionáló módszerek

Ha adott egy n elemből álló adatbázis, és a létrehozandó klaszterek száma k ($k \leq n$), akkor egy particionáló algoritmus az elemeket k partícióba szervezi, ahol minden partíció egy klaszternek felel meg. A klasztereket úgy alakítjuk ki, hogy egy objektív particionáló feltételnek, amit gyakran *hasonlósági függvénynek* neveznek, az optimumát keressük. Ez a függvény olyan, mint a távolság. Az optimumot úgy keressük, hogy az egy klaszteren belüli elemek „hasonlóak" legyenek, míg a különböző klaszterekben levő elemek „különbözőek" legyenek az adatbázis attribútumaival kifejezve.

8.4.1. | Klasszikus particionáló módszerek: *k*-átlag és *k*-medoid

A legismertebb és leggyakrabban alkalmazott particionáló módszerek a *k*-átlag és a *k*-medoid módszer, illetve ezek különböző változatai.

Súlypont alapú technika: a *k*-átlag módszer

A ***k*-átlag algoritmus** bemenő paramétere k, és az algoritmus n elemet particionál k klaszterbe úgy, hogy az eredményben a klasztereken belüli hasonlóság nagy legyen, a klaszterek közötti hasonlóság pedig kicsi legyen. A klaszterhasonlóságot a klaszterbeli elemek *átlagához* képest mérjük. Ezt az átlagot úgy is tekinthetjük, mint a klaszter *súlypontját*.

Nézzük meg, *hogyan működik a k-átlag algoritmus*. Az algoritmus működése a következő lépésekből áll. Először véletlenszerűen kiválasztunk k elemet, kezdetben ezek fogják jelölni a klaszterek átlagát (illetve középpontját). A fennmaradó elemek mindegyikét hozzárendeljük ahhoz a klaszterhez, amelyikhez a leginkább hasonló. A hasonlóságot az elem és a klaszter középpontja közötti távolság jelenti most. Ezután kiszámítjuk a klaszterek új átlagát (középpontját). Ez a folyamat ismétlődik mindaddig, amíg a kritériumfüggvény konvergál. Leggyakrabban a **négyzetes hiba kritériumfüggvényt** használjuk, aminek a definíciója a következő:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} | p - m_i |^2, \tag{8.18}$$

ahol E az összes adatbázisbeli elemre vonatkozó hibanégyzet összege, p egy adott elemet reprezentáló térbeli pont, m_i pedig a C_i klaszter átlaga (p és m_i is többdimenziósak). Ez a kritérium megpróbálja az eredményül kapott klasztereket a lehető legtömörebbé és legjobban elkülönülővé alakítani. A *k*-átlag algoritmus összefoglalását a 8.1. ábrán láthatjuk.

Az algoritmus megpróbál k darab olyan partíciót meghatározni, amelyek minimalizálják a hibanégyzet függvényt. Ez akkor működik jól, ha a klaszterek tömör felhők, amelyek jól elkülönülnek egymástól. A módszer elég jól skálázható és nagy adathalmazokra is hatékony, mert az algoritmus számítási bonyolultsága $O(nkt)$, ahol n az elemek száma,

Algoritmus: *k*-**átlag**. A klaszterekbeli elemek átlagán alapuló *k*-átlag algoritmus a particionálás elvégzésére.

Bemenet: A klaszterek száma *k*, és az *n* elemből álló adatbázis.

Kimenet: *k* darab klaszter, amely a négyzetes hiba kritériumot minimalizálja.

Módszer:

(1) válasszunk tetszőlegesen *k* elemet, ezek lesznek a kezdeti klaszterközéppontok;
(2) ismételjük
(3) rendeljünk minden elemet ahhoz a klaszterhez, amelyikhez az elem a leginkább hasonló, úgy, hogy ehhez a klaszterek korábbi átlagát vesszük figyelembe;
(4) frissítsük a klaszterek átlagát, vagyis számoljuk ki az új átlagot;
(5) mindaddig, amíg nincs változás;

8.1. ábra | A *k*-átlag algoritmus

k a klaszterek száma, *t* pedig az iterációk száma. Általában $k \ll n$ és $t \ll n$. A módszer gyakran egy lokális optimumban fejeződik be.

A *k*-átlag módszert azonban csak akkor alkalmazhatjuk, ha a klaszterek átlagát definiálni lehet. Néhány alkalmazás esetén ez nem így van, például ha az adatok felsorolás típusú attribútummal is rendelkeznek. A felhasználók számára hátrányt jelenthet, hogy a klaszterek számát előre meg kell adniuk. A *k*-átlag módszer nem alkalmas olyan klaszterek felismerésére, amelyeknek nem konvex az alakja, vagy nagyon eltérő méretű klaszterek felismerésére. Továbbá a módszer érzékeny a zajra és a szélsőséges értékekre, mivel néhány ilyen elem jelentős hatással van az átlagra.

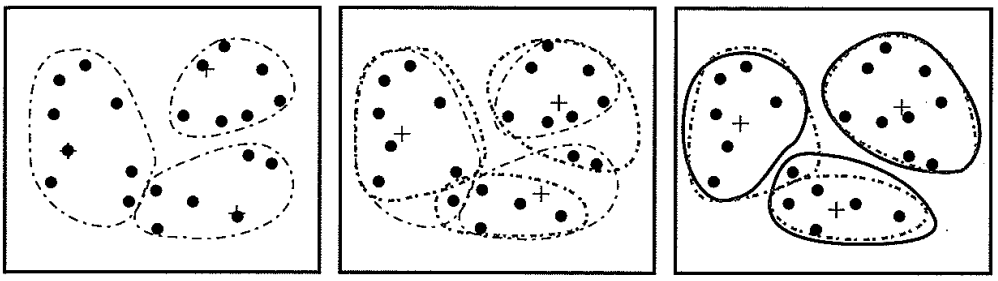
8.2. példa | Tegyük fel, hogy van egy elemekből álló halmazunk a térben, ahogyan az a 8.2(a) ábrabeli téglalapon látható. Legyen $k = 3$ a kívánt klaszterek száma.

A 8.1. ábra algoritmusa szerint először tetszőlegesen kiválasztunk három elemet, amelyek a kezdeti klaszterközéppontok lesznek. A klaszterek középpontjait + jel jelöli (lásd 8.2. ábra). Minden elemet hozzárendelünk valamelyik klaszterhez, méghozzá ahhoz, amelyiknek a középpontja a legközelebb esik az elemhez. Az így kialakult klaszterek körvonalait jelöltük pontozott vonallal a 8.2(a) ábrán.

Ez a csoportosítás maga után vonja a klaszterek középpontjainak módosulását. A csoportosítás után a módosult klaszterbe tartozó összes elem alapján ki is számoljuk ezeket az új középpontokat. Ezután az elemeket ismét hozzárendeljük a klaszterek valamelyikéhez, most már az új középpontoktól való távolságuk alapján. Az újonnan kialakult klaszterek körvonalait szaggatott vonal jelöli a 8.2(b) ábrán.

Ez a folyamat ismétlődik, amíg el nem jutunk a 8.2(c) ábrán látható állapothoz. Most már a klaszterek egyikéből sem kellett elemet máshová áthelyeznünk, így az algoritmus befejeződik. A klaszterező folyamat az eredményül kapott klasztereket adja meg kimenetként. ■

A *k*-átlag algoritmusnak számos különböző változata ismeretes. Ezek a kezdeti átlagok kiválasztásában, a különbözőség számítási módjában vagy a klaszterközéppontok kiszámítási módjában különbözhetnek az eredeti változattól. Figyelemre méltó és gyakran jó eredményt adó stratégia, hogy először hierarchikus egyesítő algoritmussal meghatározzuk a klaszterek számát és kezdeti elhelyezkedését, majd ezután alkalmazzuk az iteratív áthelyezést az eredmény javítására.



(a) (b) (c)

8.2. ábra | Elemek klaszterezése a *k*-átlag módszerrel. A klaszterek középpontjait + jelöli

A *k*-átlag módszer egy másik változata az úgynevezett ***k*-módusz módszer**, amelyik a *k*-átlag módszert úgy általánosítja, hogy az felsorolás típusú adatokra is alkalmazható legyen. Ezt úgy teszi, hogy a klaszterek átlaga helyett azok móduszát veszi figyelembe. Ehhez persze másfajta különbözőségi mértéket használ, és a klaszterek móduszát is egy gyakoriságon alapuló módszerrel aktualizálja. A *k*-átlag és a *k*-módusz módszereket egyesíthetjük is olyan esetekben, amikor az adatok között numerikus és felsorolás típusúak is szerepelnek, és az így kapott módszert nevezzük ***k*-prototípus módszernek**.

Az **EM (elvárások maximalizálása)** algoritmus más módon terjeszti ki a *k*-átlag módszert. Ahelyett hogy az elemeket egyetlen klaszterhez rendelné, az elemeket adott súllyal rendeli a klaszterekhez, ahol a súly annak valószínűségét jelöli, hogy az elem az adott klaszterhez tartozik. Így nem lesznek éles határvonalak a klaszterek között. Éppen ezért a módszer másfajta átlagokat számol ki, amelyek súlyozott mértékeken alapulnak.

Nézzük meg, hogyan tehetjük skálázhatóbbá a k-átlag algoritmust. Egy viszonylag új kísérlet azon az ötleten alapul, hogy az adatokat három különböző régióba osztjuk. Lesznek olyan régiók, amelyek tömöríthetők, lesznek olyanok, amelyeket a memóriában kell tartanunk, és lesznek eldobhatók. Azok az elemek lesznek *eldobhatók*, amelyeknek a klaszterhez való tartozását már véglegesen megállapítottuk. Azok az elemek lesznek *tömöríthetők*, amelyek ugyan nem eldobhatók, de egy szűk részklaszterhez tartoznak. Az eldobható és tömöríthető elemekre vonatkozó információkat egy úgynevezett *klaszterezési jellemzők* nevű adatszerkezetben tároljuk. Ha egy elem sem nem eldobható, sem nem tömöríthető akkor azt a *memóriában kell tartanunk*. A skálázhatóságon úgy javíthatunk, hogy az iteratív klaszterező algoritmus csak a tömörítendő és a memóriában tartandó elemek klaszterezési jellemzőivel foglalkozik, ezáltal egy merevlemez alapú algoritmusból memória alapú algoritmust kapunk.

Egy jellegzetes elemen alapuló technika: a *k*-medoid módszer

A *k*-átlag módszer érzékeny a szélsőséges értékekre, mivel egy kiugróan nagy értékkel rendelkező elem jelentősen eltorzíthatja az adatok hozzárendelését.

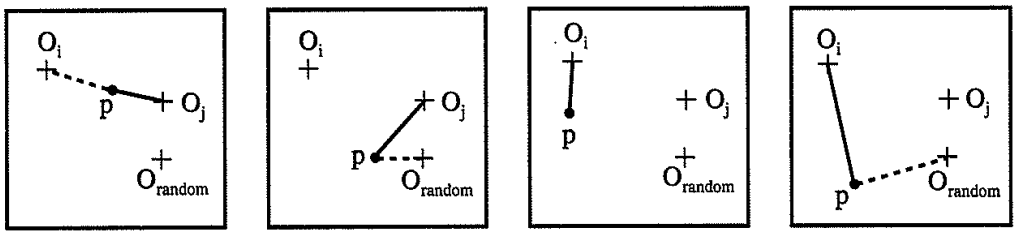
Nézzük meg, hogyan lehet az algoritmus módosításával ezt az érzékenységet csökkenteni. Vegyük a klaszterbeli elemek átlaga helyett azok **medoidját**, ami a leginkább középen elhelyezkedő elem, és ezt használjuk referenciapontként. Ezzel a módosítással a particionáló módszer ugyanúgy alkalmazható, mint korábban. Az alapelv most is az, hogy az elemek és a referencia pontok közötti különbözőségek összegét próbáljuk meg minimalizálni. A fentiek képezik a ***k*-medoid módszer** alapját.

A k-medoid algoritmus n elemhez k klasztert keres a következő módon. Először tetszőlegesen válasszunk ki minden klaszterhez egy őt képviselő elemet, a medoidot. A többi elemet osszuk szét a klaszterek között, a hozzá legközelebb lévő medoid alapján. A módszer ezután folyamatosan lecseréli a medoidokat egy másik, nem medoid elemmel, egészen addig, amíg a klaszterezés minősége javítható. A klaszterezés minőségét egy költségfüggvénnyel becsüljük, ami az elemek és a hozzájuk tartozó klaszter medoidjának átlagos különbözőségét méri. Annak eldöntése, hogy egy nem medoid elem, o_{random}, jó lesz-e az aktuális medoid o_j helyettesítésére, a következő négy eset megvizsgálásával történik minden p, nem medoid elem esetén.

- **1. eset:** p jelenleg az o_j medoidhoz tartozik. Ha o_j-t o_{random}-mal cseréltük le és p legközelebb valamelyik o_i-hez van, ahol $i \neq j$, akkor p-t hozzárendeljük o_i-hez.
- **2. eset:** p jelenleg az o_j medoidhoz tartozik. Ha o_j-t o_{random}-mal cseréltük le és p legközelebb o_{random}-hoz van, akkor p-t hozzárendeljük o_{random}-hoz.
- **3. eset:** p jelenleg az o_i medoidhoz tartozik $i \neq j$. Ha o_j-t o_{random}-mal cseréltük le és p még mindig o_i-hez van a legközelebb, akkor p hozzárendelésén nem változtatunk.
- **4. eset:** p jelenleg az o_i medoidhoz tartozik $i \neq j$. Ha o_j-t o_{random}-mal cseréltük le és p legközelebb o_{random}-hoz van, akkor p-t hozzárendeljük o_{random}-hoz.

A fenti négy esetet a 8.3. ábrán szemléltetjük. Minden alkalommal, amikor a hozzárendelés megváltozik, az E hibanégyzetben változás lesz, ami a költségfüggvény változását vonja maga után. Ezért a költségfüggvény a hibanégyzetben bekövetkező *változást* számolja ki, amikor egy medoidot lecserélünk egy nem medoid elemmel. A csere összköltsége megegyezik az összes nem medoid elem költségváltozásainak összegével. Ha ez az összköltség negatív, akkor o_j-t felcseréljük o_{random}-mal, hiszen ezáltal az aktuális hibanégyzet, E, csökkenni fog. Ha az előbbi összköltség pozitív, akkor az aktuális medoid, o_j megfelelő, és így nem cseréljük le. A k-medoid algoritmust a 8.4. ábrán láthatjuk.

A **PAM** (particionálás medoidokkal) volt az egyik első k-medoid módszeren alapuló algoritmus. Ez az algoritmus n elemhez k darab partíciót próbál meg létrehozni. A k-medoid kezdeti véletlenszerű kiválasztása után ismétlődően megpróbál jobb medoidokat találni. Minden lehetséges elempárt megvizsgál, miközben az egyik elemet medoidnak



1. O_i-hez rendelés | 2. O_{random}-hoz rendelés | 3. nincs változás | 4. O_{random}-hoz rendelés

- • adatelem
- + klaszter közepe
- — csere előtt
- --- csere után

8.3. ábra | A költségfüggvény négy esete a k-medoid klaszterezéskor

Algoritmus: *k*-medoid. Az elemek particionálására szolgáló *k*-medoid algoritmus, amely a medoidon vagy központi elemen alapul.

Bemenet: A klaszterek száma *k*, és az *n* elemből álló adatbázis.

Kimenet: *k* darab klaszter, amely az elemek legközelebbi medoidtól vett különbözőségeit minimalizálja.

Módszer:

(1) válasszunk tetszőlegesen *k* elemet, ezek lesznek a kezdeti medoidok;
(2) ismételjük
(3) rendeljünk minden további elemet a legközelebbi medoid klaszteréhez;
(4) válasszunk ki véletlenszerűen egy nem medoid elemet, o_{random}-ot;
(5) számoljuk ki az *S* összköltségét az $o_j \rightarrow o_{random}$ cserének;
(6) **if** S < 0 **then** cseréljük le o_j-t o_{random}-mal;
(7) mindaddig, amíg nincs változás;

8.4. ábra | A *k*-medoid algoritmus

tekinti, a másikat pedig nem medoidnak. A kapott klaszterezés minőségét minden ilyen kombinációra kiszámolja. Az o_j elemet azzal az elemmel fogja lecserélni, amelyiknek a hatására a legnagyobb mértékben csökken a hibanégyzet. A klaszterenként így kapott legjobb elemek alkotják a medoidokat a következő iterációban. Nagy *n* és *k* értékekre ez a számolási mód nagyon költséges lehet.

Nézzük meg, hogy melyik módszer a robusztusabb (ellenállóbb): a k-átlag vagy a k-medoid. A *k*-medoid módszer kevésbé érzékeny a zajokra és a szélsőséges értékekre, mivel a medoidra kisebb hatással vannak ezek a kiugró értékek, mint az átlagra. A végrehajtása azonban költségesebb ennek a módszernek, mint a *k*-átlagnak. A felhasználónak mindkét módszer esetén meg kell adnia *k*-t, a klaszterek elvárt számát.

8.4.2. | Particionáló módszerek nagy adatbázisokban: *k*-medoidtól a CLARANS-ig

Nézzük meg, hogy mennyire hatékony a k-medoid algoritmus nagy adathalmazok esetén. A *k*-medoid algoritmusok, mint amilyen például a PAM, hatékonyak kis adathalmaz esetén, de nem jól skálázhatók nagy adatmennyiség esetén. Nagyobb adathalmazokra valamilyen *mintavételezésen* alapuló módszert lehet alkalmazni, mint amilyen például a **CLARA** (Clustering LARge Applications, klaszterezés nagy alkalmazások esetén).

A CLARA a következő alapötletet használja: nem a teljes adathalmazt veszi figyelembe, hanem annak csak egy kis részét, ami jól reprezentálja a tényleges adatokat. A medoidokat ebből a mintából választja ki, a PAM algoritmust használva. Ha a mintát megfelelően véletlenszerűen választjuk ki, akkor az jól fogja reprezentálni az eredeti adathalmazt. A kiválasztott reprezentáló elemek (medoidok) vélhetően hasonlóak lesznek azokhoz, amiket a teljes adathalmazból választottunk volna ki. A CLARA algoritmus több mintát vesz az adatokból, mindegyikre alkalmazza a PAM algoritmust és a legjobb klaszterezést adja vissza kimenetként. Ahogyan ez várható, a CLARA nagyobb adathalmazok kezelésére alkalmas, mint a PAM. Az egyes iterációk bonyolultsága ez esetben $O(ks^2 + k(n - k))$, ahol *s* a minta mérete, *k* a klaszterek száma, *n* pedig az elemek száma.

A CLARA algoritmus eredményessége a minta méretétől függ. Vegyük észre, hogy a PAM algoritmus az egész adathalmazból keresi meg a k legjobb medoidot, míg a CLARA algoritmus csak a kiválasztott mintából. Ezért a CLARA nem találhatja meg a legjobb klaszterezést, ha a k legjobb medoid valamelyike nincs benne a mintában. Ha például az o_i elem a k legjobb medoid egyike, de a mintavételezésnél nem választottuk ki, akkor a CLARA soha nem fogja a legjobb klaszterezést megtalálni. Ez az ára a hatékonyságnak. Egy jó klaszterezés, ami mintákon alapul, nem jelenti feltétlenül az egész adathalmaz jó klaszterezését is, főleg akkor nem, ha a minta nem eléggé reprezentatív.

Hogyan javíthatunk a CLARA algoritmus minőségén és skálázhatóságán? Erre egy másik k-medoid típusú algoritmust javasoltak, aminek **CLARANS** (Clustering Large Applications based upon RANdomized Search, nagy alkalmazások klaszterezése véletlen kereséssel) a neve, és amelyik a mintavételezést kombinálja a PAM-mal. A CLARA-val ellentétben azonban a CLARANS egyik időpontban sem szorítkozik egy rögzített mintára. A CLARA a keresés egyes fázisaiban rögzített mintával rendelkezik, ezzel szemben a CLARANS minden lépésnél új mintát választ valamilyen véletlen tényező segítségével. A klaszterező folyamatot elképzelhetjük úgy, mint egy gráfban való keresést, amelynél minden csúcspont egy lehetséges megoldást, vagyis k darab medoidot jelent. Egy medoid cseréjével előálló klaszterezést az aktuális klaszterezés *szomszédjának* nevezünk. A véletlenszerűen kipróbált szomszédok számát egy felhasználó által megadott paraméter korlátozza. Ha egy jobb szomszédot találunk (amelyre kisebb a hibanégyzet), a CLARANS átmegy ebbe a szomszédos csomópontba, és a folyamat kezdődik elölről. Ellenkező esetben az aktuális klaszterezés egy lokális optimumot jelent. A CLARANS ha egy ilyen lokális optimumot talált, akkor véletlenszerűen kiválaszt egy új csúcsot, hogy másik lokális optimumot keressen.

A CLARANS-ról a tapasztalatok azt mutatják, hogy eredményesebb a PAM-nál és a CLARA-nál is. Ezzel az algoritmussal megtalálhatjuk a klaszterek számára vonatkozó „legtermészetesebb" értéket. Ehhez egy úgynevezett *körvonaltényezőt* használhatunk, ami egy elemnek olyan tulajdonsága, ami megmondja, hogy az adott elem mennyire tartozik valóban a klaszterhez. A CLARANS a szélsőséges értékek felismerésére is alkalmas. A számítási bonyolultsága azonban $O(n^2)$ körül van, ahol n az elemek száma. A klaszterezés minősége is nagyban függ az alkalmazott mintavételezéstől. A CLARANS teljesítményét tovább javíthatjuk olyan térbeli adatstruktúrák felfedezésével, mint az R* fák, valamint egyes fokuszáló technikák segítségével.

8.5. | Hierarchikus módszerek

Egy hierarchikus klaszterező módszer úgy működik, hogy az adatelemeket egy klaszterekből álló fába csoportosítja. Ezeket a módszereket tovább osztályozhatjuk *egyesítő* és *felosztó* módszerekre attól függően, hogy a hierarchikus dekomponálást alulról fölfelé vagy föntről lefelé végzik. Egy tisztán hierarchikus klaszterezésen alapuló módszernek az a hátránya, hogy ha az egyesítés vagy felosztás megtörtént, akkor a módszer nem képes ennek javítására. A legújabb kutatások a megoldást a hierarchikus egyesítés és az iteratív áthelyezés egyesítésében látják.

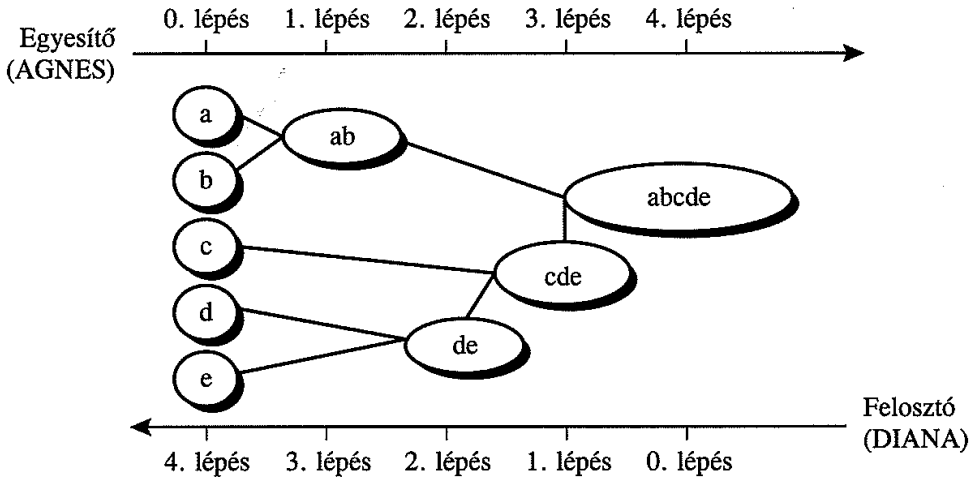
8.5.1. | Egyesítő és felosztó hierarchikus klaszterezés

A hierarchikus klaszterezésnek két alapvető módját különböztetjük meg.

- **Egyesítő hierarchikus klaszterezés** – Ez a lentről fölfelé haladó stratégia először minden elemet egy különálló klaszterbe tesz, majd ezeket az atomi klasztereket egyre nagyobb klaszterekké egyesíti, amíg az összes elem egyetlen klaszterbe nem kerül, vagy amíg valamilyen befejeződési feltételek nem teljesülnek. A legtöbb hierarchikus klaszterező módszer ebbe a kategóriába tartozik. A különbségek általában csak a klaszterek közötti hasonlóság definiálásában jelentkeznek.
- **Felosztó hierarchikus klaszterezés** – Ez a föntről lefelé haladó módszer épp a fordítottját végzi az egyesítő hierarchikus klaszterezésnek. Kezdetben az összes elemet egyetlen klaszterbe teszi, majd ezt egyre kisebb és kisebb részekre bontja, amíg minden elem külön klaszterbe nem kerül, vagy amíg bizonyos leállási feltételek nem teljesülnek. Ilyen feltétel lehet például, hogy elérjük a klasztereknek egy adott darabszámát, vagy hogy a két legközelebbi klaszter közötti távolság egy megadott távolsági küszöbnél nagyobb lesz.

8.3. példa | A 8.5. ábrán az **AGNES** (Aglomerative NESting, egyesítő beágyazás) egyesítő hierarchikus módszernek és a **DIANA** (Divisive ANAlysis, felosztó elemzés) felosztó módszernek az alkalmazását láthatjuk az $\{a, b, c, d, e\}$ ötelemű adathalmazra. Az AGNES először minden elemet egy külön klaszterbe tesz, majd ezeket lépésenként egyesíti bizonyos feltételek alapján. Például a C_1 és C_2 klaszterek akkor egyesíthetők, ha van olyan C_1-beli és C_2-beli elem, amelyek közötti távolság minimális a különböző klaszterekbe tartozó elempárok közötti távolságokat tekintve. Ezt **egyetlen kapcsolat** megközelítésnek tekinthetjük, mivel a klasztert a benne lévő összes elem képviseli, és a klaszterek közötti hasonlóságot a bennük lévő legközelebbi elempárok hasonlóságával mérjük. A klasztereket egyesítő folyamat addig ismétlődik, amíg végül az összes elem egyetlen klaszterbe nem kerül.

A DIANA algoritmusnál az összes elem együttesen alkot egy kiindulási klasztert. A klasztert valamilyen elv alapján kettéosztjuk, például vesszük a klaszteren belüli legközelebbi szom-



8.5. ábra | Egyesítő és felosztó hierarchikus klaszterezés az $\{a, b, c, d, e\}$ elemekre

szédok közül azt a kettőt, amelyek között a legnagyobb az euklideszi távolság. A klaszterek felosztása addig folytatódik, amíg végül minden klaszter már csak egy elemet tartalmaz. ■

Mind az egyesítő, mind a felosztó hierarchikus klaszterezésnél a felhasználó megadhatja a klaszterek kívánt számát, ami a befejeződési feltételként szolgál.

Az alábbiakban megadunk négy elterjedt mértéket, amit a klaszterek közötti távolságok méréséhez szoktak használni. A jelölések a következőket jelentik: $|p - p'|$ a p és p' elemek közötti távolság, m_i a C_i klaszter átlaga, n_i pedig a C_i klaszterbeli elemek száma.

- **Minimális távolság:** $d_{min}(C_i, C_j) = min_{p \in C_i, p' \in C_j} |p - p'|$
- **Maximális távolság:** $d_{max}(C_i, C_j) = max_{p \in C_i, p' \in C_j} |p - p'|$
- **Átlagok távolsága:** $d_{mean}(C_i, C_j) = |m_i - m_j|$
- **Átlagos távolság:** $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$

Nézzük meg, hogy melyek a hierarchikus klaszterezés gyenge pontjai. Ez a módszer ugyan meglehetősen egyszerű, mégis problémákat okozhat számára a megfelelő egyesítések, illetve felosztások megtalálása. Ezek a döntések azért kritikusak, mert miután az elemek egy csoportját felosztottuk vagy egyesítettük, a folyamat a következő lépésben ezeken az újonnan létrejött klasztereken dolgozik tovább. Sem visszacsinálni nem fog egy korábbi lépést, sem nem fogja az elemeket a klaszterek között áthelyezni. Ezért a nem megfelelően kiválasztott felosztások és egyesítések gyenge minőségű klaszterezéshez vezetnek. A módszer ezenkívül nem túl jól skálázható, mivel a felosztó és egyesítő lépések eldöntéséhez nagyon sok elemet, illetve klasztert kell megvizsgálni.

Egy ígéretes iránynak tűnik a hierarchikus módszerek javítására, hogy a hierarchikus klaszterezést valamilyen más módszerrel ötvözzük, és így több fázisú klaszterezést kapunk. A következő részben néhány ilyen módszert mutatunk be. Az első, a BIRCH módszer először fastruktúrák segítségével hierarchikusan particionálja az elemeket, majd más klaszterező módszerek segítségével finomítja a kapott klasztereket. A második, a CURE nevű, minden klasztert egy előre rögzített számú elemével reprezentál, majd ezeket zsugorítja a klaszter közepe felé valamilyen arányban. A harmadik, a ROCK algoritmus a klasztereket az összekapcsoltságaik alapján egyesíti. A negyedik, a Chameleon nevű a dinamikus modellezést használja a hierarchikus klaszterezésnél.

8.5.2. | BIRCH: kiegyensúlyozott iteratív csökkentés és klaszterezés hierarchiák segítségével

A **BIRCH** egy integrált hierarchikus klaszterezési módszer. Két új fogalmat vezet be, a **klaszterezési jellemzőt (CF)**, és a **klaszterezési jellemző fát (CF-fa)**, amelyeket a klaszterek tömör reprezentálására használ. Ezek a struktúrák lehetővé teszik, hogy a módszer gyors és skálázható legyen nagy adatbázisokban is. A BIRCH módszer eredményes folyamatosan bejövő adatok dinamikus klaszterezése esetén is. Nézzük meg az imént említett struktúrákat kissé részletesebben. A **klaszterezési jellemző (CF)** egy hármas, ami az elemek részklasztereiről tárol információkat. Ha adott N darab d-dimenziós adat-

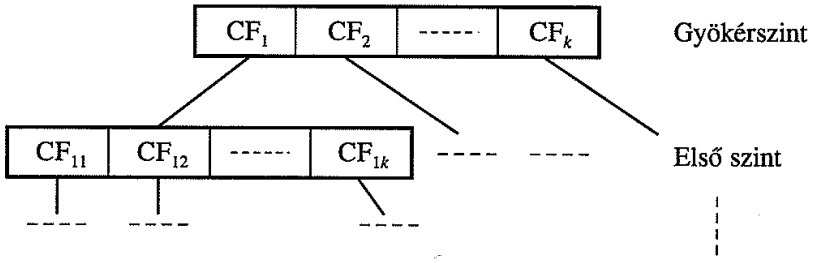
pont vagy adatelem $\{o_i\}$ egy részklaszterben, akkor a részklaszterre vonatkozó CF-et a következőképpen definiáljuk.

$$CF = (N, \vec{LS}, SS), \tag{8.19}$$

ahol N a részklaszterbeli pontok száma, $\vec{LS}$ az N pont lineáris összege $\left(\sum_{i=1}^{N} \vec{o}_i\right)$, SS pedig az adatpontok négyzetes összege $\left(\sum_{i=1}^{N} \vec{o}_i^2\right)$.

A klaszterezési jellemző tulajdonképpen az adott részklaszterre vonatkozó statisztikák összegzése, vagyis statisztikai szempontból nézve a részklaszter nulladik, első és második momentumai. Ez a klaszterek kiszámolásához alapvető fontosságú információkat tárol, és a tároló helyet is hatékonyan használja, mivel a részklaszterre vonatkozó összesített információkat tárol, az összes elem tárolása helyett.

A **CF-fa** egy kiegyensúlyozott fa, amely a klaszterezési jellemzőket tárolja egy hierarchikus klaszterezéshez. A 8.6. ábrán láthatunk erre egy példát. Definíció szerint egy fában a nem levél csúcsoknak vannak leszármazottai (gyerekei). Most a nem levél csúcsok a gyermekeik CF-jeinek összegét tárolják, vagyis a gyermekeikre vonatkozó klaszterezési információkat. Egy CF-fának két paramétere van, a *B elágazási faktor* és a *T küszöb*. Az elágazási faktor azt adja meg, hogy egy nem levél csúcsnak maximum hány gyereke lehet. A küszöb pedig megadja, hogy a levél csúcsokban tárolt részklaszteреknek maximum mekkora lehet az átmérője. Ez a két paraméter az eredményül kapott fa méretét befolyásolja.



8.6. ábra | Egy CF-fa szerkezete

Nézzük meg, hogyan működik a BIRCH algoritmus. Az algoritmus két fázisból áll:

- **1. fázis:** A BIRCH beolvassa az adatbázist, és felépít belőle a memóriában egy kezdeti CF-fát. Ezt az adatok többszintű tömörítésének tekinthetjük, amelyik megpróbálja az adatokban rejlő belső klaszterezési szerkezetet megőrizni.
- **2. fázis:** A BIRCH egy (kiválasztott) klaszterező algoritmust alkalmaz a CF-fa levél csomópontjaira.

Az 1. fázisban a CF-fát dinamikusan, az objektumok folyamatos beszúrásával építi az algoritmus. Az elemeket a hozzájuk legközelebb lévő levél csomópontba (részklaszterbe) teszi be. Ha a levélben tárolt részklaszter átmérője a beszúrás után nagyobb lesz, mint a küszöb érték, akkor a levél csomópontot és esetlegesen más csomópontokat is kettéoszt az algoritmus. Az új elem beszúrása után a rá vonatkozó információ a fa gyökere felé

halad. A CF-fa mérete megváltoztatható a küszöb módosításával. Ha a CF-fa tárolásához szükséges memória nagyobb a tényleges memóriánál, akkor egy kisebb küszöböt adhatunk meg, és a fát újraépítjük. Az újraépítés úgy történik, hogy egy új fát építünk a régi fa leveleiből. Ezáltal az újraépítési folyamatot anélkül végezhetjük el, hogy ismét beolvasnánk az összes elemet. A fenti folyamat hasonlít a B+-fák létrehozásakor alkalmazott beszúráshoz és csomópont felosztáshoz. Így a fa építéséhez csak egyszer kell az adatokat beolvasni. Néhány módszer és heurisztika ismeretes a szélsőséges értékek kezelésére, valamint arra, hogy további beolvasások segítségével hogyan lehet javítani a CF-fa minőségén. A CF-fa felépítése után bármelyik klaszterező algoritmus (például valamelyik particionáló algoritmus) alkalmazható a 2. fázisban.

A BIRCH algoritmus a legjobb klaszterezést próbálja meg létrehozni a rendelkezésre álló erőforrások segítségével. Ha a memória mérete korlátozott, akkor egy fontos szempont, hogy minimalizáljuk az I/O-műveletekhez szükséges időt. A BIRCH többfázisú klaszterezési technikát használ. Az adathalmaz egyszeri beolvasásával egy megközelítően jó klaszterezést ad, majd egy vagy több (opcionális) beolvasással tovább javítja a minőséget. Az algoritmus számítási bonyolultsága $O(n)$, ahol n a klaszterezendő elemek száma.

Mennyire hatékony a BIRCH algoritmus? A tapasztalatok azt mutatják, hogy az algoritmus lineárisan skálázható az elemek számának függvényében, és az adatoknak jó minőségű klaszterezését adja. Mivel azonban a CF-fa csomópontjai a méretük miatt csak korlátozott számú bejegyzést tartalmazhatnak, ezért ezek a csomópontok nem mindig annak felelnek meg, amit a felhasználó egy természetes és jó klaszternek tekint. Ezenkívül, ha a klaszterek nem gömb alakúak, akkor a BIRCH algoritmus nem túl jó eredményt ér el, mivel ez a klaszterek határainak megállapításánál a sugár, illetve átmérő fogalmával dolgozik.

8.5.3. | CURE: klaszterezés reprezentáló elemek segítségével

A legtöbb klaszterező algoritmus vagy a gömb alakú és hasonló méretű klasztereket részesíti előnyben, vagy pedig érzékeny a szélsőséges értékekre. A **CURE** (Clustering Using Representatives, klaszterezés reprezentáló elemekkel) algoritmus elég jól kezeli a szélsőséges értékeket, és nem feltétlenül eredményez gömb alakú, hasonló méretű klasztereket sem.

A CURE egy új hierarchikus klaszterezési algoritmust alkalmaz, amely köztes megoldást jelent a középpont alapú és a reprezentáló elemen alapuló megközelítések között. A klaszterek reprezentálására nem egyetlen középpontot vagy egyetlen elemet használ, hanem rögzített számú pontot választ ki a térben. Ezeket a reprezentáló pontokat úgy hozza létre, hogy először jól szétszórt elemeket választ ki, majd ezeket zsugorítja, illetve mozgatja a klaszter közepe felé egy bizonyos *zsugorító tényezővel*. Az algoritmus minden lépésben azt a két klasztert egyesíti, amelyekben a két legközelebbi reprezentáló elempár található. (Az elempárok természetesen különböző klaszterekből értendők.)

A CURE azáltal, hogy egynél több reprezentáló elemet használ klaszterenként, jól alkalmazkodik a nem gömb formájú alakzatokhoz is. A klaszterek tömörítése segít a szélsőséges értékek hatásainak csökkentésében. Az előbbiekből adódóan, a CURE kevésbé érzékeny a szélsőséges értékekre, és jó eredménnyel azonosítja be a nem gömb alakú

és eltérő méretű klasztereket is. Jól skálázható nagy adatbázisok esetén is, és ez nem megy a klaszterezés minőségének rovására.

A nagy adatbázisok kezelésére a CURE a véletlen mintaválasztás és a particionálás kombinációját alkalmazza. Először egy véletlen mintát particionál, majd az egyes partíciókat részlegesen klaszterezi. Ezeket a részleges klasztereket egy második menetben ismét klaszterezi, és így jut a kívánt végeredményhez.

A CURE algoritmus lényegét a következő lépések vázolják fel.

(1) Kiválasztunk egy S véletlen mintát az eredeti elemekből.
(2) S-et partíciók egy halmazába particionáljuk.
(3) Részlegesen klaszterezzük az egyes partíciókat.
(4) Véletlen mintaválasztással eltüntetjük a szélsőséges értékeket. Ha egy klaszter túl lassan nő, kitöröljük.
(5) A részleges klasztereket tovább klaszterezzük. A újonnan alakult klaszterekbe eső reprezentáló pontokat a klaszter közepe felé „zsugorítjuk" egy a felhasználó által megadott, α zsugorító tényezővel. Az így kialakuló pontok határozzák meg a klaszter alakját.
(6) Megjelöljük az adatokat a megfelelő klasztercímkékkel.

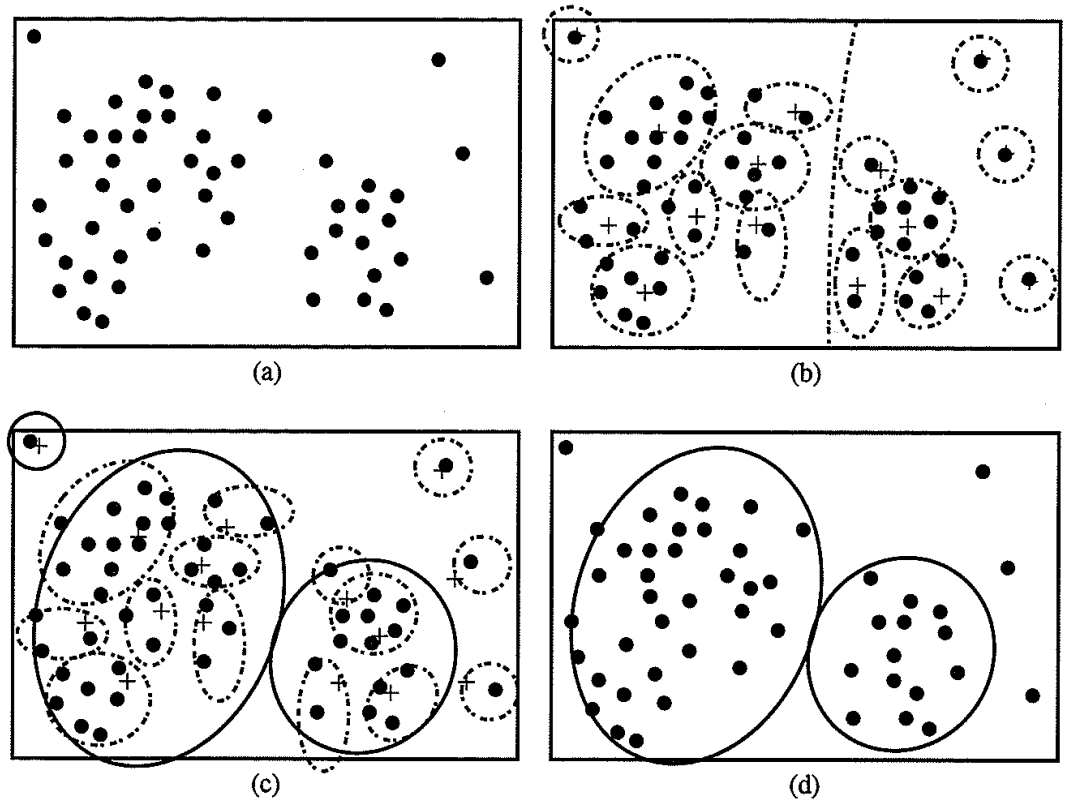
Nézzünk egy példát.

8.4. példa | Képzeljük el, hogy adott egy ponthalmaz, amely egy téglalap alakú területen belül helyezkedik el. A pontoknak egy véletlenül kiválasztott mintáját láthatjuk a 8.7(a) ábrán. Ezeket az elemeket két partícióba osztjuk, amelyeket részlegesen klaszterezünk az átlagok minimális távolsága alapján. A részleges klasztereket szaggatott vonal jelöli a 8.7(b) ábrán. A klasztereket képviselő pontokat + jelöli. A részleges klasztereket tovább klaszterezzük; az eredményül kapott két klasztert folytonos vonal jelöli a 8.7(c) ábrán. Az új klasztereket „zsugorítjuk" oly módon, hogy a reprezentáló pontokat a klaszter közepe felé mozgatjuk egy α tényezővel. A reprezentáló pontok határozzák meg az egyes klaszterek alakját. Végeredményül az eredeti elemeket két klaszterbe particionáljuk, úgy hogy a szélsőséges értékeket kizárjuk. Ezt láthatjuk a 8.7(d) ábrán. ■

A CURE jó minőségű klasztereket hoz létre szélsőséges értékek létezése esetén is, és megengedi a különböző méretű és alakú klaszterek létrejöttét. Az algoritmushoz egyszer kell beolvasni a teljes adatbázist. n elem esetén a CURE bonyolultsága $O(n)$.

Nézzük meg, hogy mennyire érzékeny a CURE a felhasználó által megadott paraméterekre. Ilyen paraméterek a minta mérete, a klaszterek elvárt száma, és az α zsugorító tényező. Az erre vonatkozó vizsgálatok azt mutatták, hogy noha bizonyos paraméterek változtatása nincs hatással a klaszterezés minőségére, az összes paraméter együttes beállítása mégis jelentősen befolyásolja a végeredményt.

A CURE nem képes a felsorolás típusú attribútumok kezelésére. A **ROCK** egy másik egyesítő hierarchikus algoritmus, amelyik alkalmas a felsorolás típusú attribútumok klaszterezésére. Ez az algoritmus úgy méri két klaszter hasonlóságát, hogy a két klaszter *összesített összekapcsoltságát* hasonlítja össze egy a felhasználó által megadott, statikus *összekapcsoltsági modellel*. A C_1 és C_2 klaszter közötti **összekapcsoltságot** a közöttük



8.7. ábra | Elemek klaszterezése a CURE algoritmussal. (a) Az elemekből választott véletlen minta. (b) Az elemek particionálása és részleges klaszterezése. A klaszterek reprezentáló pontjait + jelöli. (c) A részleges klaszterek további klaszterezése. Az új klaszterek reprezentáló pontjait a klaszter közepe felé mozgatjuk. (d) A végleges, nem gömb alakú klaszterek

meglevő keresztkapcsolatok száma definiálja, ahol a $link(p_i, p_j)$ keresztkapcsolat a p_i és p_j pontok közös szomszédainak a száma. Másként megfogalmazva, a klaszterek közötti hasonlóság a különböző klaszterekben lévő, közös szomszéddal rendelkező pontpárok számán alapul.

A ROCK algoritmus először egy ritka gráfot hoz létre egy adott hasonlósági mátrixból, amihez a hasonlósági küszöb és a közös szomszéd fogalmát használja. Ezután egy hierarchikus klaszterező algoritmust alkalmaz a ritka gráfra.

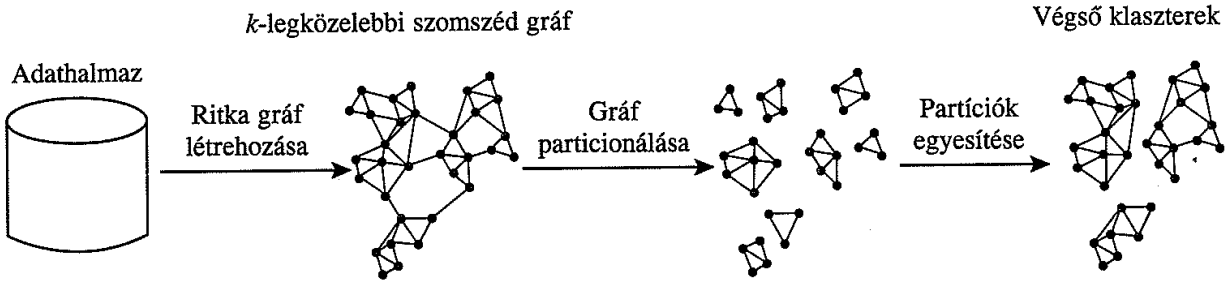
8.5.4. | Chameleon: dinamikus modellezést használó hierarchikus klaszterező algoritmus

A **Chameleon** olyan klaszterező algoritmus, amelyik a dinamikus modellezést használja a hierarchikus klaszterezéshez. Ez az algoritmus a klaszterező részében akkor egyesít két klasztert, ha a közöttük lévő összekapcsoltság (illetve közelség) szoros kapcsolatban van az elemeik közötti összekapcsoltsággal (illetve közelséggel). A dinamikus modellen alapuló egyesítés elősegíti a természetes és homogén klaszterek felismerését, és minden adattípusra alkalmazható, feltéve, hogy adott valamilyen hasonlósági függvény.

A Chameleon algoritmus úgy született meg, hogy két másik algoritmusnak, a CURE-nek és a ROCK-nak a gyengeségeit próbálták meg kijavítani. A CURE és a hozzá hason-

ló algoritmusok figyelmen kívül hagyják a különböző klaszterekbeli elemek közötti összesített összekapcsoltságra vonatkozó információt, míg a ROCK típusú algoritmusok csak az összekapcsoltságra összpontosítanak, és figyelmen kívül hagyják a két klaszter közelségét.

Hogyan működik a Chameleon algoritmus? Az algoritmus működésének vázlatát a 8.8. ábrán láthatjuk. A Chameleon először egy gráfparticionáló algoritmussal az elemeket viszonylag nagy számú, relatív kicsi részklaszterekbe teszi. Ezután egy egyesítő hierarchikus klaszterező algoritmussal, a részklaszterek ismételt egyesítésével megkeresi a tényleges klasztereket. A leginkább hasonló részklaszterekből álló párok megkereséséhez figyelembe veszi a klaszterek összekapcsoltságát és közelségét is, különös figyelmet fordítva a klaszterek belső jellegzetességeire. Ezáltal nem függ egy statikus, a felhasználó által megadott modelltől, és automatikusan tud alkalmazkodni az egyesítendő klaszterek specialitásaihoz.



8.8. ábra | Chameleon: a *k*-legközelebbi szomszédon és a dinamikus modellezésen alapuló hierarchikus klaszterezés [KHK99] alapján

Nézzük meg a Chameleon algoritmust részletesebben. Amint azt a 8.8. ábrán is láthatjuk, a Chameleon az elemeket a gyakran használt *k-legközelebbi szomszéd gráf* segítségével reprezentálja. A gráfnak minden csúcspontja egy elemnek felel meg, és két csúcs között akkor van él, ha az egyik elem a másik elemre legjobban hasonlító k darab elem között van. Ez a gráf (jelöljük G_k-val) a szomszédság fogalmát dinamikusan kezeli, ami azt jelenti, hogy egy elem szomszédsági sugara az elem körüli terület sűrűségétől függ. Egy sűrű területen a szomszédság definíciója is szorosabb közelséget jelent, egy ritka területen viszont kevésbé szorosat. Ez a megközelítés természetesebb klasztereket eredményez a sűrűség alapú módszerekhez képest, például a 8.6. alfejezetben ismertetendő DBSCAN-hez képest, amelyik egy globális szomszédságdefinícióval dolgozik. Ráadásul a gráf a területek sűrűségét is tárolja az élekhez rendelt súlyok segítségével, így egy sűrűbb terület élei nagyobb súllyal rendelkeznek, mint egy ritka területé.

A Chameleon a C_i és C_j klaszterek közötti hasonlóságot a közöttük meglévő *relatív összekapcsoltság* $RI(C_i, C_j)$ és *relatív közelség* $RC(C_i, C_j)$ alapján határozza meg.

- A C_i és C_j klaszterek közötti **relatív összekapcsoltság** $RI(C_i, C_j)$ definíciója a következő: vesszük a klaszterek közötti abszolút összekapcsoltságot és normalizáljuk a két klaszter közötti belső összekapcsoltsággal.

$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}, \tag{8.20}$$

ahol $EC_{\{C_i, C_j\}}$ a C_i-t és C_j-t is tartalmazó klaszter *élvágásainak* a száma, ha az C_i-re és C_j-re esik szét. Hasonlóan EC_{C_i} (illetve EC_{C_j}) a *kétszektoros minimális vágás*, vagyis azoknak az éleknek a súlyozott összege, amelyek a gráfot két, nagyjából egyforma részre particionálják.

- A C_i és C_j klaszterek közötti **relatív közelség** $RC(C_i, C_j)$ definíciója a következő: vesszük a klaszterek közötti abszolút közelséget és normalizáljuk a két klaszter közötti belső közelséggel.

$$RI(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}, \tag{8.21}$$

ahol $\overline{S}_{EC_{\{C_i,C_j\}}}$ azon élek átlagos súlya, amelyek C_i-beli és C_j-beli csúcsokat kötnek össze, $\overline{S}_{EC_{C_i}}$ (illetve $\overline{S}_{EC_{C_j}}$) pedig azon élek átlagos súlya, amelyek a C_i (illetve C_j) klaszter kétszektoros minimális vágásához tartoznak.

Megmutatták, hogy a Chameleon algoritmus hatékonyabb a tetszőleges alakú klaszterek jó minőségű felismerésében, mint a CURE és a DBSCAN. A feldolgozás költsége azonban magas dimenziószámú elemek esetén $O(n^2)$ lehet legrosszabb esetben, n elemre vonatkozóan.

8.6. | Sűrűség alapú módszerek

A tetszőleges alakú klaszterek felismerésére vezették be a sűrűség alapú klaszterezést. Ezek az algoritmusok általában a sűrű területeket tekintik klaszteroknak, amiket zajokat jelentő ritka régiók választanak el egymástól.

8.6.1. | DBSCAN: megfelelően sűrű és összefüggő területekre alapozó, sűrűség alapú klaszterezési módszer

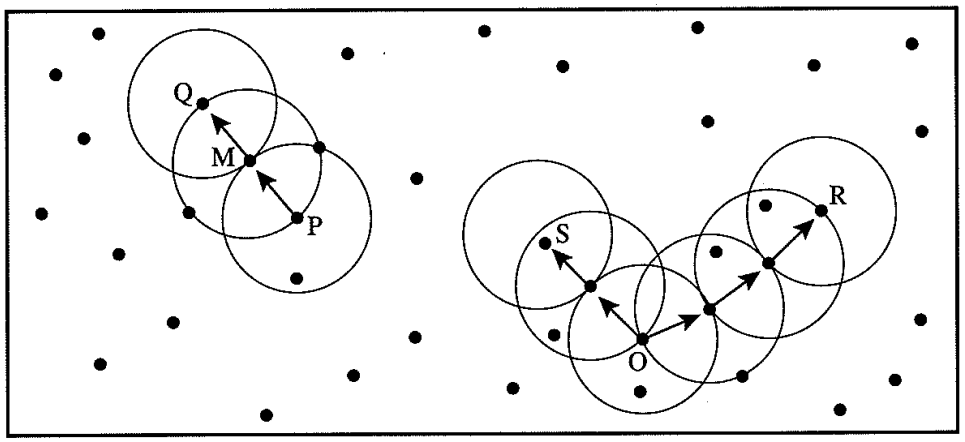
A **DBSCAN** (Density-Based Clustering of Applications with Noise, zajos alkalmazások sűrűség alapú térbeli klaszterezése) egy sűrűség alapú klaszterező módszer. Az algoritmus a megfelelően sűrű területeket növeli klaszterekké, és tetszőleges alakú klasztereket felismer téradatbázisokban, zajok megléte esetén is. A klasztereket úgy definiálja, mint *sűrűn összekötött* pontok maximális halmazát.

A sűrűség alapú klaszterezés alapötlete számos új definíciót vezet be. Először ezek intuitív magyarázatát adjuk meg, majd egy példával illusztráljuk őket.

- Egy adott elem ε sugarú környezetén belüli részt az elem **ε-környezetének** nevezünk.
- Ha egy elem ε-környezete legalább *MinPts* (minimális küszöbszám) darab elemet tartalmaz, akkor az elemet **belső elemnek** nevezzük.
- Adott D elemhalmaz esetén azt mondjuk, hogy a p elem **közvetlenül sűrűn elérhető** a q elemből, ha q belső elem, és p a q ε-környezetében van.

- Egy p elem **sűrűn elérhető** a q elemből a D adathalmazban ε-ra és *MinPts*-re vonatkozóan, ha létezik elemeknek olyan $p_1, \ldots, p_n$ láncolata, hogy $p_1 = q$, $p_n = p$ és p_{i+1} közvetlenül sűrűn elérhető p_i-ből ε-ra és *MinPts*-re vonatkozóan, minden i-re $1 \leq i < n, p_i \in D$.
- Egy p elem **sűrűn összekötött** a q-val a D adathalmazban ε-ra és *MinPts*-re vonatkozóan, ha létezik olyan o elem a D-ben, amelyből p és q is sűrűn elérhető ε-ra és *MinPts*-re vonatkozóan.

A sűrűn elérhetőség a közvetlenül sűrűn elérhetőségnek a tranzitív lezártja, és ez előbbi reláció nem szimmetrikus. Csak a belső elemekre lehet a sűrűn elérhetőség kölcsönös. A sűrűn összekötöttség viszont szimmetrikus reláció.



8.9. ábra | Sűrűn elérhetőség és sűrűn összekötöttség a sűrűség alapú klaszterezésnél [EKSX96] alapján

8.5. példa | Nézzük meg a 8.9. ábrát egy adott ε-ra – ami egyenlő a körök sugarával – és legyen most *MinPts* = 3. Ekkor a fenti definíciók alapján a következők igazak:

- A betűvel jelzett pontok közül *M*, *P*, *O* és *R* belső pontok, mivel mindegyiknek az ε-környezete tartalmaz legalább 3 pontot.
- *Q* közvetlenül sűrűn elérhető *M*-ből. *M* közvetlenül sűrűn elérhető *P*-ből és fordítva.
- *Q* (nem közvetlenül) sűrűn elérhető *P*-ből, mivel *Q* közvetlenül sűrűn elérhető *M*-ből, és *M* közvetlenül sűrűn elérhető *P*-ből. *P* azonban nem sűrűn elérhető *Q*-ból, mivel *Q* nem belső pont. Hasonló megfontolással látható, hogy *R* és *S* sűrűn elérhetők *O*-ból, és *O* is sűrűn elérhető *R*-ből.
- *O*, *R* és *S* sűrűn összekötöttek. ■

Egy **sűrűség alapú klaszter** sűrűn összekötött elemek olyan halmaza, amelyik maximális a sűrűn elérhetőségre vonatkozóan. Minden klaszteren kívüli elemet zajnak tekintünk.

Nézzük meg, hogyan találja meg a DBSCAN a klasztereket. A DBSCAN úgy keresi meg a klasztereket, hogy megvizsgálja az adatbázis minden pontjának ε sugarú környezetét. Ha egy p pont ε-környezete több mint *MinPts* darab elemet tartalmaz, akkor egy új klasztert hoz létre, p belső ponttal. A DBSCAN ezután iteratív módon összegyűjti az

ezekből a belső pontokból közvetlenül sűrűn elérhető elemeket, ami egyben néhány sűrűn elérhető klaszter összevonásával is jár. A folyamat akkor ér véget, amikor már nem tud egyik klaszterhez sem új elemet hozzáadni.

Térbeli index használatával a DBSCAN számítási bonyolultsága $O(n \log n)$, ahol n az adatelemek száma. Index nélkül a bonyolultság $O(n^2)$. Az algoritmus érzékeny a felhasználó által megadott paraméterekre. A következő alfejezetben még szót ejtünk majd a DBSCAN-ről, ahol összehasonlítjuk majd egy másik sűrűség alapú módszerrel, az OPTICS-szal.

8.6.2. | OPTICS: a pontok rendezése a klaszterező struktúra azonosításához

A 8.6.1. alfejezetben tárgyalt DBSCAN algoritmus klaszterezi ugyan az adatokat, ha megadjuk az ε és *MinPts* bemenő paramétereket, de a megfelelő paraméterek megválasztása, amelyek elfogadható klaszterezéshez vezetnek, a felhasználó felelősségére van bízva. Ez számos más klaszterező algoritmus esetén is problémát jelent. Ezeket a paraméter-beállításokat általában tapasztalati úton állapítják meg, és nehéz őket jól megadni, különösen a valós, magas dimenziószámmal rendelkező adathalmazok esetén. A legtöbb algoritmus nagyon érzékeny ezekre a paraméterértékekre, és azok kis mértékű megváltoztatása az adatok egészen eltérő klaszterezéséhez vezethet. Ráadásul a sokdimenziós, valós adathalmazok gyakran nagyon aszimmetrikus eloszlásúak, és így a bennük rejlő klaszterezettségi szerkezetet nem lehet globális sűrűségi paraméterekkel jellemezni.

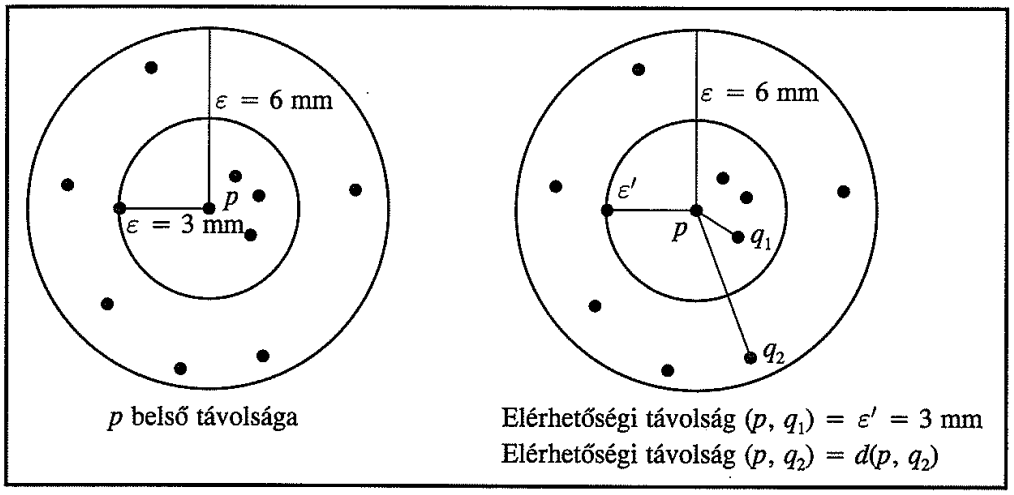
Ennek a nehézségnek az orvoslására vezették be az **OPTICS** (Ordering Points To Identify the Clustering Structure, pontok rendezése klaszterező struktúra megtalálásához) klaszterelemző módszert. Ez az algoritmus nem egy konkrét klaszterezést ad meg, hanem klaszterezések egy rendezett sorozatát, automatikus és interaktív esetben egyaránt. Ez a rendezés jól tükrözi az adatok sűrűség alapú klaszterezéseinek a szerkezetét. Ez azzal egyenértékű információt tartalmaz, mintha a sűrűség alapú klaszterezést sok fajta paraméter beállítással végeztük volna el.

Ha megvizsgáljuk a DBSCAN algoritmust, láthatjuk, hogy konstans *MinPts* érték esetén a nagyobb sűrűségre (vagyis kisebb ε értékre) vonatkozóan megkapott klaszterek *teljesen részét képezik* azoknak a klasztereknek, amiket kisebb sűrűség esetén kapnánk. Ne feledjük, hogy az ε paraméter tulajdonképpen egy távolság, méghozzá a szomszédsági sugár. Így ahhoz, hogy a sűrűség alapú klaszterek egy rendezését kapjuk, kiterjeszthetjük a DBSCAN algoritmust oly módon, hogy egyidejűleg több távolságparaméterrel dolgozzon. Ahhoz, hogy a különböző klasztereket egyidejűleg létre tudjuk hozni, az elemeket egy speciális sorrendben kell feldolgozni. Ez a sorrend először azt az elemet választja ki, amelyik a legkisebb ε mellett sűrűn elérhető, és így a sűrűbb klaszterek (kisebb ε-ra vonatkozók) lesznek készen először. Ennek az ötletnek az alkalmazásához minden elemre vonatkozóan két értéket kell tárolnunk: a *belső távolságot* és az *elérhetőségi távolságot*.

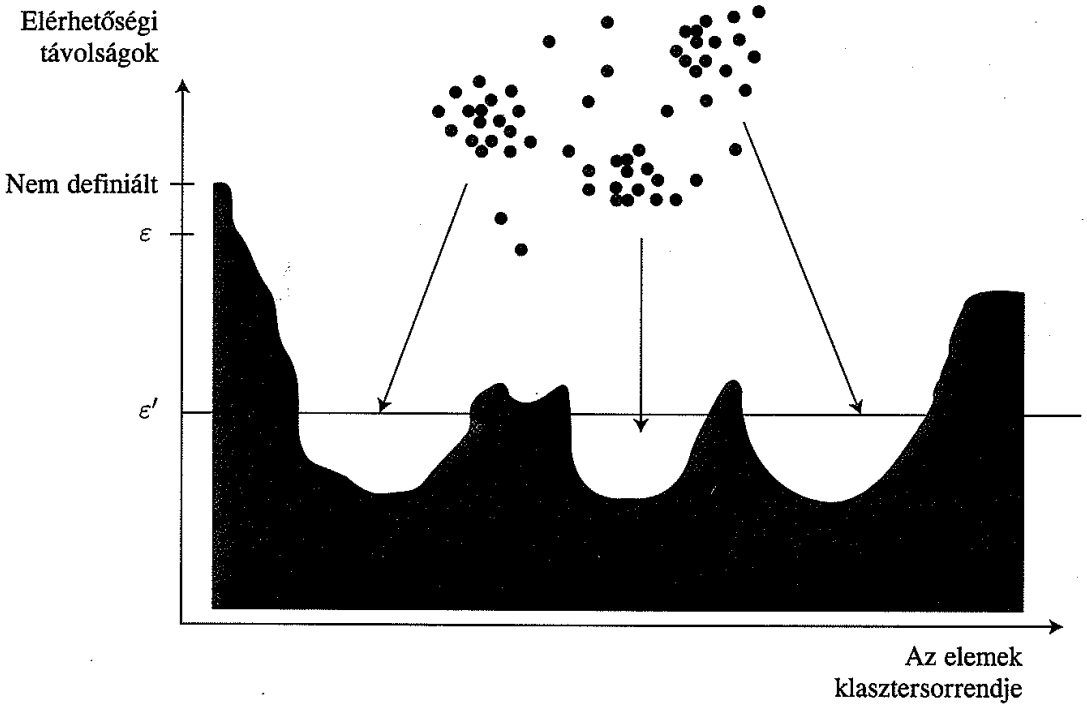
- Egy p elem **belső távolsága** az a legkisebb ε' érték, amely mellett p belső elem lesz. Ha p nem belső elem, akkor p-re a belső távolság nincs definiálva.

- A q elem **elérhetőségi távolsága** a p elemre vonatkozóan, a következő két érték közül a nagyobbik: a p elem belső távolsága, illetve a p és q közötti euklideszi távolság. Ha p nem belső elem, akkor a p és q közötti elérhetőségi távolság nincs definiálva.

8.6. példa | A 8.10. ábra a belső távolságot és az elérhetőségi távolságot illusztrálja. Tegyük fel, hogy $\varepsilon = 6$ mm és *MinPts* $= 5$. A p elem belső távolsága, ε', a hozzá negyedik legközelebb lévő elemtől való távolságával egyenlő. A q_1 elem elérhetőségi távolsága p-re vonatkozóan a p belső távolságával egyenlő (vagyis $\varepsilon' = 3$ mm), mivel ez nagyobb, mint a p és q_1 közötti euklideszi távolság. A q_2 elem elérhetőségi távolsága p-re vonatkozóan a p és q_2 közötti euklideszi távolság, mivel ez nagyobb, mint p belső távolsága. ■



8.10. ábra | Az OPTICS terminológiája [ABKS99] alapján



8.11. ábra | Klaszterek rendezése az OPTICS esetén (az ábra [ABKS99] alapján készült)

Nézzük meg, hogy hogyan használható fel az előző két érték. Az OPTICS algoritmus az elemek egy sorrendjét készíti el, továbbá tárolja minden elem belső távolságát és egy megfelelő elérhetőségi távolságát. Egy önálló algoritmust vezettek be, amelyik az OPTICS által létrehozott sorrendinformáció alapján előállítja a klasztereket. Ez az információ elegendő az összes olyan sűrűség alapú klaszter előállításához, amelyekhez tartozó ε' kisebb, mint az az ε, ami alapján a sorrendet meghatároztuk.

Egy adathalmazhoz tartozó klaszterek rendezését grafikusan is ábrázolhatjuk, ami segít a jobb megértésben. A 8.11. ábra például egyszerű kétdimenziós adathalmaz elérhetőségi térképét ábrázolja. Ez általános képet ad az adatok szerkezetéről és klaszterezéséről is. Kidolgoztak arra vonatkozó módszereket is, hogy a többdimenziós adatok szerkezetét is meg lehessen nézni, többféle részletezettségi szinten.

Mivel felépítését tekintve az OPTICS algoritmus ekvivalens a DBSCAN algoritmussal, így a bonyolultsága is azonos a DBSCAN-évél, vagyis $O(n \log n)$ ha térbeli indexet használunk.

8.6.3. | DENCLUE: klaszterezés sűrűségeloszlás-függvények alapján

A DENCLUE (DENsity-based CLUstEring, sűrűség alapú klaszterezés) olyan klaszterező módszer, amelyik sűrűségeloszlás-függvények egy halmazán alapul. A módszer a következő ötleteken nyugszik: (1) Az egyes adatpontok hatását formálisan modellezhetjük egy matematikai függvény segítségével, amelyet *hatásfüggvénynek* nevezünk; ez a függvény az adatpont hatását írja le a környezetén belül. (2) Az adatokból álló tér teljes sűrűségét modellezhetjük az összes adatponthoz tartozó hatásfüggvények összegével. (3) A klasztereket ezután úgy határozhatjuk meg matematikailag, hogy megkeressük a *sűrűségi csomópontokat*, ahol ezek a pontok a teljes sűrűség lokális maximumai.

Legyenek x és y a d dimenziós F^d tér elemei. Az y elem **hatásfüggvénye** x-re egy $f_B^y : F^d \rightarrow R_0^+$ függvény, amelyet egy f_B alaphatásfüggvény segítségével definiálunk:

$$f_B^y(x) = f_B(x, y). \tag{8.22}$$

A hatásfüggvény tulajdonképpen bármilyen függvény lehet, amelyet két közeli pont közötti távolság egyértelműen meghatároz. A $d(x, y)$ távolságfüggvénynek reflexívnek és szimmetrikusnak kell lennie, ilyen például a 8.2.1. alfejezetben említett euklideszi távolság. Ennek segítségével adhatjuk meg az úgynevezett *szögletes hullám hatásfüggvényt*,

$$f_{Square}(x, y) = \begin{cases} 0 \text{ ha } d(x, y) > \sigma \\ \text{egyébként pedig } 1; \end{cases} \tag{8.23}$$

vagy a Gauss-hatásfüggvényt,

$$f_{Gauss}(x, y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}. \tag{8.24}$$

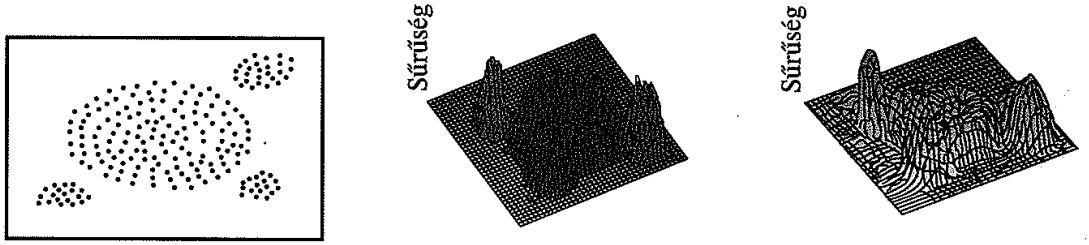
Egy $x \in F^d$ pontban a **sűrűségfüggvény** értéke egyenlő az összes adatpont hatásfüggvényeinek összegével. Ha adott n adatelem, $D = \{x_1, ..., x_n\} \subset F^d$ akkor a sűrűségfüggvény értéke x-ben a következőképpen van definiálva:

$$f_B^D(x) = \sum_{i=1}^{n} f_B^{x_i}(x). \tag{8.25}$$

Például a Gauss-hatásfüggvényen (8.24) alapuló sűrűség függvény a következő:

$$f_{Gauss}^D(x) = \sum_{i=1}^{n} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}. \tag{8.26}$$

A sűrűségfüggvény segítségével definiálhatjuk a függvény *gradiensét* és a *sűrűségi csomópontokat*, vagyis a teljes sűrűségfüggvény lokális maximumait. Egy x pontról azt mondjuk, hogy *sűrűn tart* az x^* sűrűségi csomóponthoz, ha léteznek olyan $x_0, x_1, ..., x_k$ pontok, amelyre $x_0 = x$, $x_k = x^*$, és az x_{i-1} gradiense x_i irányába mutat $0 < i < k$. Egy folytonos és deriválható hatásfüggvényre a gradiens módszerrel meghatározhatjuk az adatpontok sűrűségi csomópontjait. A 8.12. ábrán egy kétdimenziós adathalmazt, a hozzá tartozó Gauss-sűrűségfüggvényt és a sűrűségi csomópontokat láthatjuk.

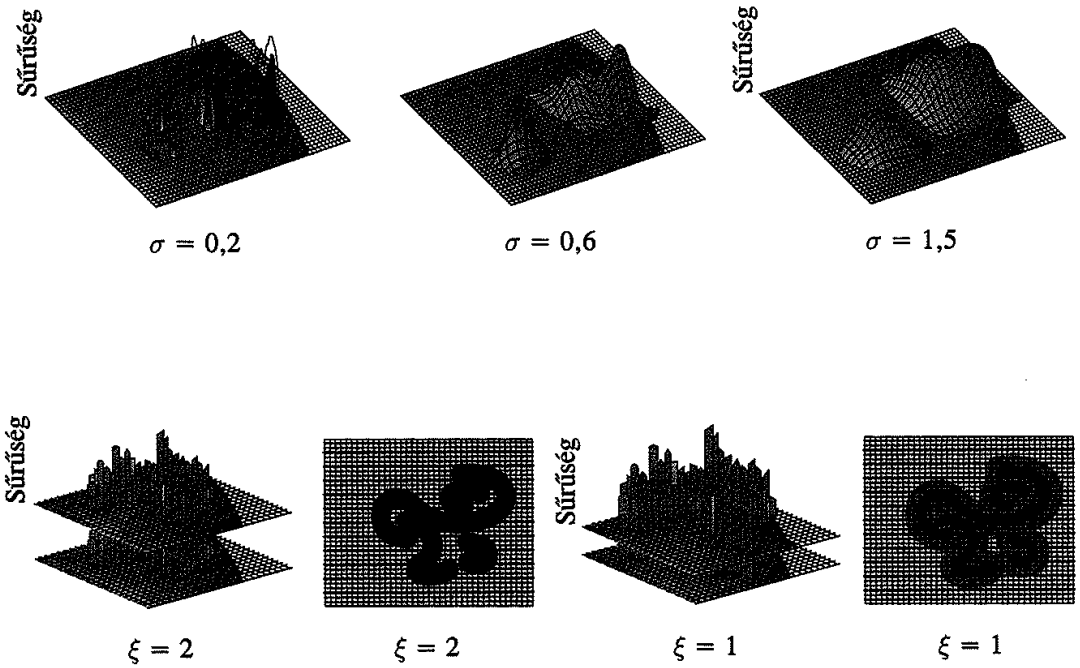


(a) Adathalmaz (b) Szögletes hullám (c) Gauss

8.12. ábra | Lehetséges sűrűségi függvények kétdimenziós adatokra [HK98]

A fenti fogalmak segítségével formálisan is definiálhatjuk a *középponttal rendelkező klasztert*, és a *tetszőleges alakú klasztert*. Egy x^* sűrűségi csomópont esetén **középponttal rendelkező klaszternek** nevezzük azt a C részhalmazt, amelynek pontjai sűrűn tartanak x^*-hoz, és még az is teljesül, hogy a sűrűségi függvény értéke x^*-ban nagyobb vagy egyenlő, mint egy ξ küszöbérték. Ha ez utóbbi feltétel nem teljesül, akkor x^*-ot szélsőséges értéknek tekintjük. A fenti tulajdonságú C-k egy halmazát **tetszőleges alakú klaszternek** nevezzük, ha létezik P út az egyes régiók között úgy, hogy a sűrűségfüggvény értéke az út menti pontokban nagyobb vagy egyenlő, mint ξ. A középponttal rendelkező klaszterekre, és a tetszőleges alakú klaszterekre a 8.13. ábrán láthatunk példákat.

Nézzük meg, hogy a DENCLUE milyen előnyökkel rendelkezik, más klaszterező algoritmusokhoz képest. Felsorolunk néhányat: (1) Szilárd matematikai alapokkal rendelkezik, és olyan egyéb klaszterező módszereket általánosít, mint a particionáláson alapuló módszer, a hierarchikus módszer, vagy a lokális keresésen alapuló módszerek. (2) Nagy mennyiségű zajt tartalmazó adathalmazokra is jól működik. (3) Tömör matematikai le-



8.13. ábra | Példák középponttal rendelkező klaszterekre (felső sor), és a tetszőleges alakú klaszterekre (alsó sor) [HK98]

írást ad meg a tetszőleges alakú klaszterekre, akár sokdimenziós adathalmazok esetén is. (4) Rács cellákat használ, de csak azokról a cellákról tárol információt, amelyekben van adatpont. Ezeket a cellákat egy fa-struktúrába szervezi, és így lényegesen gyorsabb, mint más algoritmusok (a DBSCAN-nél például akár 45-ször). A módszer azonban a σ sűrűségi paraméternek és a ξ zajküszöbnek a gondos megválasztását igényli, mivel ezek a paraméterek jelentős hatással vannak a klaszterezés minőségére.

8.7. | Rács alapú módszerek

A rács alapú klaszterező módszerek egy többszörös felbontású rács adatszerkezetet használnak. A teret véges sok cellára osztják fel, ebből alakul ki a rácsszerkezet, és minden műveletet ezen végeznek. A legnagyobb előnye ennek a módszernek a gyors végrehajtási idő, amely általában független az adatelemek számától, és csak az egyes dimenziókban megadott cellák számától függ.

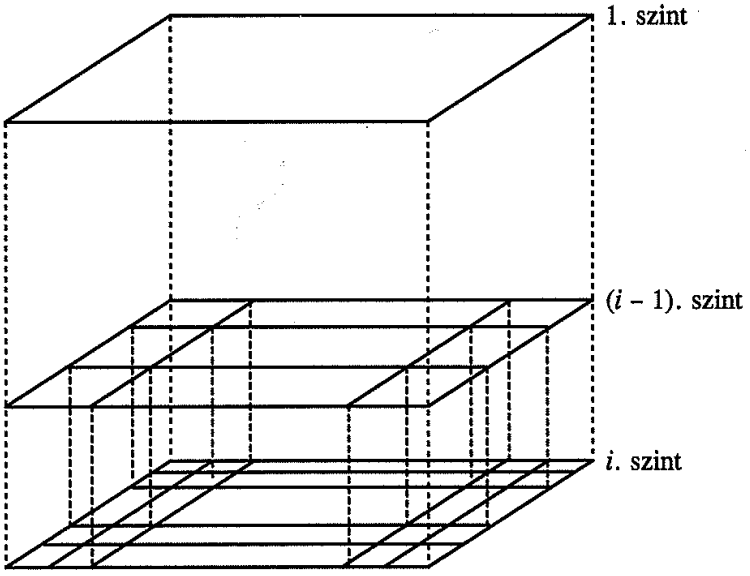
Nézzünk néhány tipikus példát rács alapú algoritmusokra. Ilyen például a STING algoritmus, amelyik a rács cellákban tárolt statisztikai információk vizsgálatán alapul. A WaveCluster algoritmus hullámtranszformációs módszerrel klaszterezi az elemeket. Végül a CLIQUE algoritmus egy rács és sűrűség alapú módszer, amely sokdimenziós adatok klaszterezésére alkalmas.

8.7.1. | STING: statisztikai információs rács

A STING egy rács alapú, többszörös felbontást használó klaszterező módszer, amely a teret téglalap alakú cellákra osztja. Általában több szintje létezik ezeknek a celláknak egyidejűleg, amelyek különböző felbontási szinteknek felelnek meg. A cellák így hierarchikus szerkezetet alkotnak oly módon, hogy minden magasabb szinten lévő cella további cellákra van osztva a következő, alsóbb szinten. A cellákban levő adatokra vonatkozó statisztikai információkat (átlag, maximum és minimum értékek) folyamatosan kiszámoljuk és tároljuk. Ezek a statisztikai adatok hasznosak lesznek az alább ismertetendő lekérdezésekhez.

A 8.14. ábra a STING algoritmushoz tartozó hierarchikus szerkezetet mutatja be. A magasabb szinten levő cellák statisztikai adatai (paraméterei) könnyen kiszámolhatók az alsóbb szintű cellák adataiból. Ilyen adatok lehetnek például a következők: az attribútum típusától független paraméter, a *szám* (számosság); az attribútumtól függő paraméterek, az *m* (átlag), *s* (szórás), *min* (minimum), *max* (maximum), és a cellabeli értékek *eloszlásának* típusa. Ez az eloszlás lehet például *normális*, *egyenletes*, *exponenciális* vagy *ismeretlen*. Amikor az adatokat betöltjük az adatbázisba, a legalsó szintű cellákra vonatkozó *darabszám, m, s, min, max* paraméterek közvetlenül kiszámolhatók az adatokból. Az *eloszlás* értékét vagy a felhasználó adhatja meg, ha ezt előre ismeri, vagy valamilyen teszt segítségével (mint amilyen például a χ^2 teszt), állapíthatjuk meg. Egy magasabb szintű cella eloszlását az alsóbb szintű celláinak az eloszlástípusai alapján számolhatjuk ki, egy úgynevezett küszöb alapú szűrő folyamattal együttműködve. Ha az alsóbb szintű cellák eloszlásai nagyon eltérőek, és egy küszöbvizsgálatnak nem felelnek meg, akkor a magasabb szintű cellának az *ismeretlen* eloszlásértéket adjuk.

Hogyan használhatók fel ezek a statisztikai információk lekérdezések megválaszolásához? A statisztikai információkat föntről lefelé haladva használhatjuk fel a következőképpen. Először meghatározzuk a hierarchikus szerkezeten belül azt a szintet, ahonnan a lekérdezések megválaszolását kezdeni fogjuk. Ez a szint általában kevés számú cellát



8.14. ábra | Hierarchikus struktúra a STING módszerhez

tartalmaz. Az aktuális szint minden cellájára kiszámoljuk a konfidenciaintervallumot (a becsült valószínűségtartományt), ami a cellának az adott lekérdezéshez való tartozását (relevanciáját) tükrözi. A nem a kérdéshez tartozó cellákat a további vizsgálatokban figyelmen kívül hagyjuk. Az eggyel lejjebbi szint vizsgálata már csak a megmaradó releváns cellákkal foglalkozik. Ez a folyamat addig ismétlődik, amíg a legalsó szintet el nem érjük. Ekkor amennyiben a lekérdezés feltételének megfelelnek, a releváns cellák adatait adjuk meg a lekérdezésre válaszul. Ellenkező esetben a releváns cellák adatait további feldolgozásoknak vetjük alá, amíg azok meg nem felelnek a lekérdezés követelményeinek.

Milyen előnyökkel rendelkezik a STING más módszerekhez képest? Az alábbiakban felsorolunk néhányat: (1) A rács alapú számolás független a lekérdezéstől, mivel az egyes cellákban tárolt statisztikai információk a lekérdezéstől független, összegző információkat tartalmaznak a cellabeli adatokra vonatkozóan. (2) A rács struktúra elősegíti a párhuzamos feldolgozást, és a növekményes adatfrissítést. (3) A módszer hatékonysága nagyon fontos előny. A STING algoritmus egyszer megy végig az adatokon a statisztikai információk kiszámolásához, és így a klaszterek létrehozásának időbonyolultsága $O(n)$, ahol n az elemek száma. A hierarchikus struktúra létrehozása után a lekérdezés feldolgozási ideje $O(g)$, ahol g a legalsó szintű cellák száma, ami általában sokkal kisebb, mint n.

Mivel a STING algoritmus többszörös felbontást használ, a klaszterezés minősége a legalsó szint finomságától függ. Ha a felbontás túlságosan finom, a feldolgozási idő jelentősen megnő, ha viszont a legalsó szintű rács túl durva, az a klaszterelemzés minőségének romlásához vezethet. A STING ráadásul nem veszi figyelembe a gyermek cellák és a velük szomszédos cellák közötti térbeli kapcsolatot, amikor a szülő cellát létrehozza. Ennek eredményeképpen a keletkező klaszterek alakja olyan lesz, hogy a klaszterek határai vagy vízszintesek, vagy függőlegesek lesznek, de átlósak nem. Ez ronthatja a klaszterek minőségét és pontosságát, a gyors végrehajtási idő ellenére.

8.7.2. | WaveCluster: klaszterezés hullámtranszformáció segítségével

A WaveCluster egy többszörös felbontást használó klaszterező algoritmus, amelyik először összegzi az adatokat oly módon, hogy egy többdimenziós rácsszerkezetet helyez el az adatokat tartalmazó térben. Ezután egy *hullámtranszformációt* alkalmaz az eredeti térre, és sűrű területeket keres a transzformált térben.

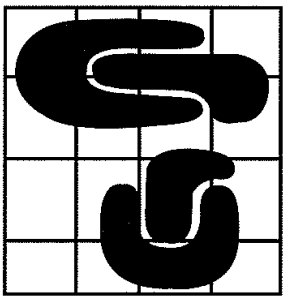
Ennél a módszernél minden cella azon adatok csoportjáról tárol összegző információkat, amelyek a cellába képződnek le. Ezek az összegző információk általában beférnek abba a memóriarészbe, amit a hullámtranszformáció és a további klaszterelemzés használ.

Nézzük meg, hogy mi is az a hullámtranszformáció. A **hullámtranszformáció** egy jelfeldolgozási technika, ami egy jelet különböző frekvenciájú részsávokba bont szét. A hullám modellt úgy tudjuk n dimenziós jelekre alkalmazni, hogy egy egydimenziós hullámtranszformációt n-szer alkalmazunk. A hullámtranszformációt úgy alkalmazzuk, hogy az adatok közötti relatív távolság megőrződjön a felbontás különböző szintjein. Ez lehetővé teszi az adatokban rejlő természetes klaszterek könnyebb megkülönböztethetőségét. A klasztereket ezután úgy azonosítjuk, hogy sűrű területeket keresünk az új tarto-

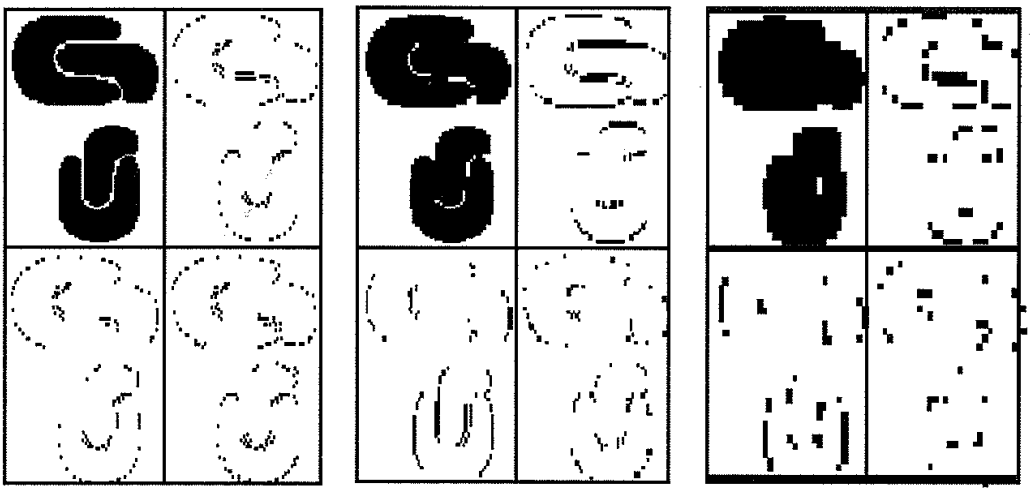
mányban. A hullámtranszformációkról a 3. fejezetben már volt szó, ahol tömörítéssel történő adatcsökkentésre használtuk őket. Erre a technikára vonatkozó további utalásokat az irodalomjegyzékben találhatunk.

Miért hasznos a hullámtranszformáció a klaszterezéshez? A módszernek a következő előnyei vannak:

- Felügyelet nélküli klaszterezést biztosít. Négyszög alakú szűrőket használ, amelyek kihangsúlyozzák azokat a területeket, ahol a pontok sűrűsödnek, és egyidejűleg csillapítják a klaszter határain kívüli gyengébb információkat. Ezáltal az eredeti térbeli sűrűbb régiók sűrűségi csomópontként viselkednek a közeli pontokkal szemben, és taszítják a távoli pontokat. Ez azt jelenti, hogy az adatok klaszterei automatikusan kirajzolódnak, és „kitisztítják" a körülöttük levő területeket. Így egy másik előnye a hullámtranszformációnak, hogy automatikusan eltünteti a szélsőséges értékeket.
- A többszörös felbontásnak köszönhetően a klaszterek felismerését különböző pontossági szinteken végezhetjük el. Például a 8.15. ábrán egy mintát láthatunk a kétdimenziós térben, amelynél a kép minden pontja a térbeli adathalmaz egy elemének attribútum értékeit reprezentálja. A 8.16. ábra a hullámtranszformáció eredményét mutatja különböző felbontások mellett, egy finomabb felbontástól [(a) ábra] a durva felbontá-



8.15. ábra | Egy minta a kétdimenziós térben [SCZ98]



(a) (b) (c)

8.16. ábra | A 8.15. ábrán szereplő tér többszörös felbontása. (a) nagy felbontás; (b) közepes felbontás; (c) alacsony felbontás [SCZ98]

sig [(c) ábra]. Mindegyik szinten azt a négy részsávot láthatjuk, amelyekbe az eredeti adatokat szétbontottuk. A bal fölső negyedben látható részsáv a pontok körüli átlagos szomszédságot hangsúlyozza ki. A jobb fölső negyedben látható részsáv az adatok vízszintes éleit hangsúlyozza. A bal alsó negyedben látható részsáv az adatok függőleges éleit hangsúlyozza, végül a jobb alsó negyedbeli részsáv a sarkokat hangsúlyozza.

- A hullámtranszformáción alapuló klaszterezés nagyon gyors. A számítási bonyolultsága $O(n)$, ahol n az adatbázisbeli elemek száma. Az algoritmus megvalósítása párhuzamosítható.

A WaveCluster egy rács alapú és egyben sűrűség alapú algoritmus. Megfelel számos olyan követelménynek, amiket egy jó klaszterező algoritmustól elvárunk: hatékonyan kezeli a nagy adathalmazokat, tetszőleges alakú klasztereket felfedez, sikeresen kezeli a szélsőséges értékeket, nem érzékeny a bemenő adatok sorrendjére, és nem kell a számára megadni a bemenő paramétereket, például a klaszterek számát vagy a szomszédsági sugarat. A kísérleti tanulmányok a WaveCluster algoritmust jobbnak találták, mint a BIRCH, CLARANS vagy DBSCAN algoritmust, mind hatékonyságban, mind a klaszterezés minőségét tekintve. A vizsgálat a WaveCluster algoritmust alkalmasnak találta akár 20-dimenziós adatok kezelésére is.

8.7.3. | CLIQUE: sokdimenziós terek klaszterezése

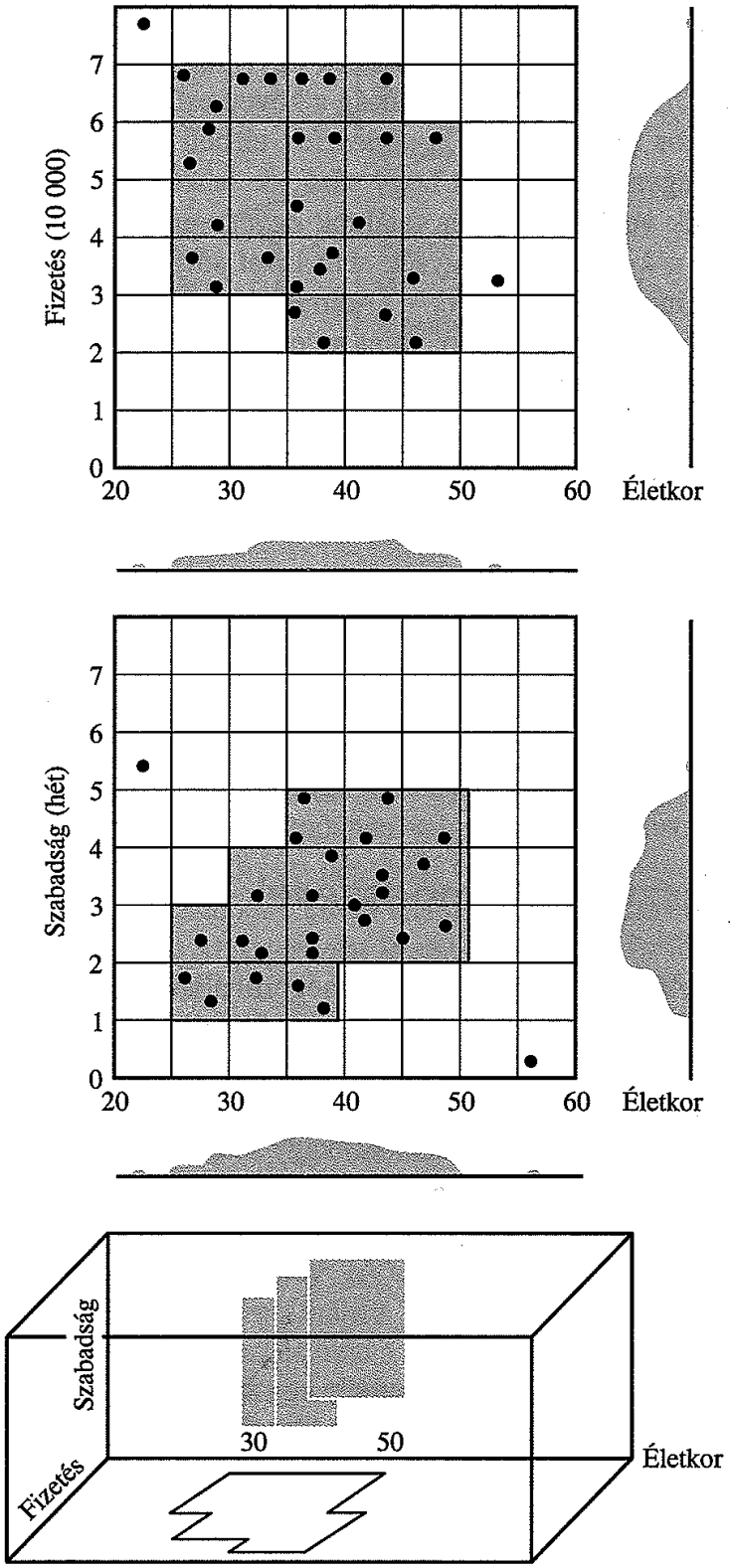
A CLIQUE klaszterező algoritmus a sűrűség alapú és a rács alapú klaszterezést egyesíti. Nagy adatbázisokban lévő, sokdimenziós adatok klaszterezésénél hasznos. A CLIQUE algoritmus a következőkre alapul:

- Ha adva van többdimenziós adatpontoknak egy nagy halmaza, akkor a pontok általában nem egyenletesen töltik ki a teret. A CLIQUE algoritmus felfedezi a ritka és a zsúfolt területeket a térben (illetve egységekben), és ezáltal felismeri adathalmaz mintáinak általános eloszlását.
- Egy egységet sűrűnek mondunk, ha az általa tartalmazott összes adatpont hányada meghalad egy bemeneti paramétert. A CLIQUE algoritmusnál a klasztert úgy definiáljuk, mint összefüggő sűrű egységek maximális halmaza.

Nézzük meg, hogyan működik a CLIQUE algoritmus. A CLIQUE többdimenziós klaszterezést végez két lépésben.

Az első lépésben a CLIQUE particionálja az n dimenziós adatteret egymást nem átfedő téglalap alakú egységekre, majd azonosítja közülük a sűrű egységeket. Ezt minden dimenzióra elvégzi. Például a 8.17. ábra azokat a sűrű, téglalap alakú egységeket mutatja, amelyeket az *életkorra* vonatkozóan találtunk a *fizetés* és *szabadság* dimenziókra. Ezen sűrű egységeket reprezentáló alterek metszete fog egy lehetséges keresési térként szolgálni, amelyben magasabb dimenziójú sűrű egységek lehetnek.

Vajon miért korlátozza a CLIQUE a magasabb dimenziójú sűrű egységek keresését az alterek sűrű egységeinek metszetére? A lehetséges keresési tér azonosítása az *apriori*



8.17. ábra | Az életkorra vonatkozóan megtalált sűrű egységek a fizetés és szabadság dimenziókra. Ezek metszete adja meg a lehetséges keresési terét a magasabb dimenziójú sűrű egységeknek

tulajdonságon alapul, amit a társítási szabályok keresésénél használnak.[1] Ez a tulajdonság a keresési térbeli elemekre vonatkozó előzetes tudást alkalmazza úgy, hogy a tér egyes részei elhagyhatók legyenek. A CLIQUE esetén ez a tulajdonság a következőt mondja ki: ha egy *k-dimenziós* egység sűrű, akkor sűrűk a $(k - 1)$-*dimenziós* térbeli vetületei is. Vagyis ha adva van egy k-dimenziós egység, ami lehet hogy sűrű, de lehet, hogy nem, akkor elég ellenőriznünk a $(k - 1)$-edik vetületeit, és ha találunk egyet, ami nem sűrű, akkor biztosak lehetünk benne, hogy a k-dimenziós egység sem lehet sűrű. Ezért a k-dimenziós lehetséges sűrű egységeket megkaphatjuk a $(k - 1)$-dimenziós tér sűrű egységeiből. Általában az így kapott keresési tér jóval kisebb, mint az eredeti. Ezután a sűrű egységeket kell megvizsgálnunk a klaszterek meghatározásához.

A második lépésben a CLIQUE egy minimális leírást generál minden klaszterhez a következő módon. Minden klaszterre meghatározza azt a sűrű egységekből álló maximális régiót, ami lefedi a klasztert. Ezután minden klaszternek meghatározza a minimális lefedését.

Nézzük meg, hogy mennyire hatékony a CLIQUE algoritmus. A CLIQUE automatikusan megtalálja a legmagasabb dimenziókban is azokat az altereket, amelyekben nagyon sűrű klaszterek vannak. Az algoritmus nem érzékeny a bemenő sorok sorrendjére, és nem feltételez semmilyen szokásos adateloszlást. Lineárisan skálázható a bemenő adatok méretével, és jól skálázható, ha az adatok dimenziószáma növekszik. A klaszterezés eredményének a pontosságát azonban csökkentheti a módszer egyszerűsége.

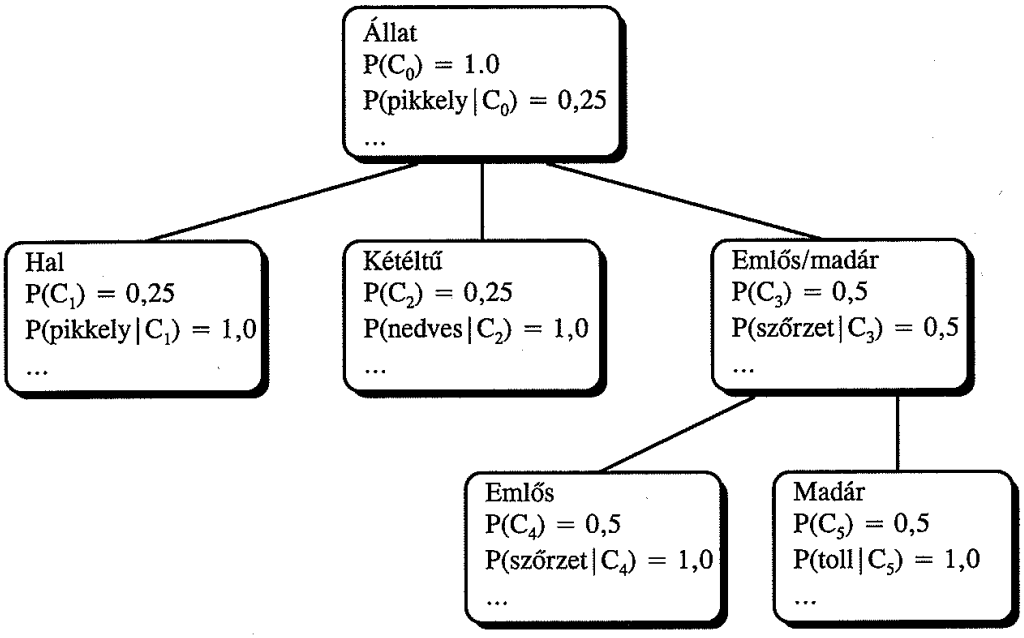
8.8. | Modell alapú klaszterező módszerek

A modell alapú klaszterezési módszerek úgy működnek, hogy megpróbálják a tényleges adatokat valamilyen matematikai modellhez illeszteni, lehetőleg a hozzájuk legjobban passzolóhoz. Ezek a módszerek gyakran azzal a feltételezéssel élnek, hogy az adatok valamilyen valószínűségeloszlások keverékeként állnak elő. A modell alapú klaszterező módszerek általában vagy statisztikai megközelítést használnak, vagy neuronhálókon alapulót. Ebben az alfejezetben mindkét megközelítésre mutatunk példákat.

8.8.1. | Statisztikai megközelítés

A **fogalmi klaszterezés** a gépi tanulás klaszterezési formája, amely egy adott elemhalmazra az elemeknek egy osztályozási sémáját adja meg. A hagyományos klaszterezéstől eltérően, amely elsősorban a hasonló elemek csoportjait azonosítja be, a fogalmi klaszterezés egy lépéssel továbbmegy, és az egyes csoportokhoz jellemző leírásokat is keres. A csoportok itt a klaszternek vagy osztálynak felelnek meg. Vagyis a fogalmi klaszterezés kétlépcsős folyamat; az első lépés a klaszterezés, a második pedig a karakterizáció. Ebben az esetben a klaszterezés minősége nem csupán az egyes elemektől függ, hanem olyan tényezőktől is, hogy az egyes fogalomleírások mennyire általánosak vagy mennyire egyszerűek.

1 | A társítási szabályok keresését részletesen a 6. fejezet tartalmazza. Az apriori tulajdonság ezen belül a 6.2.1. alfejezetben található.



8.18. ábra | Egy osztályozó fa. A kép [Fis87] alapján készült

A legtöbb fogalmi klaszterező módszer statisztikai megközelítést alkalmaz, és valamilyen valószínűségi mérték alapján határozza meg a fogalmakat, illetve klasztereket. A kapott fogalmakat általában valószínűségi leírások jellemzik.

A **COBWEB** egy népszerű és nem túl bonyolult, növekményes fogalmi klaszterező módszer. Az input elemeit felsorolás típusú attribútum-érték párok írják le. A COBWEB hierarchikus klaszterezést készít **osztályozó fa** formájában.

Nézzük meg, hogy mi is az osztályozó fa. Mennyiben hasonlít ez a döntési fára? A 8.18. ábrán egy osztályozó fát láthatunk, amelyen állatokra vonatkozó információk szerepelnek. Az ábra [Fis87] alapján készült. Az osztályozó fa különbözik a döntési fától. Minden csomópontja egy fogalomra utal, és ennek a fogalomnak egy valószínűségi leírását tartalmazza. A fogalom az adott csomóponthoz sorolt elemek összefoglalása. A valószínűségi leírás a fogalom valószínűségét és a következő alakú feltételes valószínűségeket tartalmazza: $P(A_i = V_{ij} \mid C_k)$, ahol $A_i = V_{ij}$ egy attribútum–érték páros, C_k pedig a fogalomhoz tartozó osztály. (A számosságokat a csomópontokban gyűjtjük és tároljuk a valószínűségek kiszámolásához.) Az imént ismertetett szerkezet eltér a döntési fától, amely az ágakat címkézi, és nem a csomópontokat, továbbá logikai leírókat használ a valószínűségiek helyett.[2] Az osztályozó fa egy adott szintjén lévő testvérekre azt mondjuk, hogy azok egy **partíciót** alkotnak. Egy elem osztályozása az osztályozó fa használatával úgy történik, hogy lefelé haladunk a fában a legjobban illeszkedő csomópontok mentén, miközben egy részleges egyezés függvényt alkalmazunk.

A COBWEB a fa felépítéséhez egy heurisztikus értékelési módszert használ, a *kategóriahasznosságot*. A **kategóriahasznosság** (CU) definíciója a következő:

$$\frac{\sum_{k=1}^{n} P(C_k)[\Sigma_i \Sigma_j P(A_i = V_{ij} \mid C_k)^2 - \Sigma_i \Sigma_j P(A_i = V_{ij})^2]}{n}, \tag{8.27}$$

2 | A döntési fákról a 7. fejezetben volt szó.

ahol n azon csomópontok vagy fogalmak vagy „kategóriák" száma, amelyek a fa egy adott szintjén egy $\{C_1, C_2, ..., C_n\}$ partíciót alkotnak. Másképp kifejezve, a kategóriahasznosság az a szám, amennyivel több attribútumértéket tudunk helyesen megbecsülni egy partíció ismeretében ahhoz képest, mintha a becslést ezen információ nélkül végeznénk. $P(C_k)\Sigma_i\Sigma_j P(A_i = V_{ij} | C_k)^2$ felel meg a helyes becslések várható számának egy adott partíció esetén, míg $\Sigma_i\Sigma_j P(A_i = V_{ij})^2$ a partíció ismerete nélküli helyes becslések száma). Noha hely hiányában nem bizonyítjuk, de azért megemlítjük, hogy a kategóriahasznosság az osztályokon belüli hasonlóságot, és a különböző osztályok esetén a különbözőséget „díjazza". E két fogalom a következőképpen értendő.

- Az **osztályon belüli hasonlóság** a $P(A_i = V_{ij} | C_k)$ valószínűség. Minél nagyobb ez az érték, az osztályon belüli elemeknek annál nagyobb hányada rendelkezik ezzel az attribútum-érték párral, és annál inkább előre jelezhető, hogy ez az értékpár egy osztálybeli elemhez tartozik.
- Az **osztályok közötti különbözőség** a $P(C_k | A_i = V_{ij})$ valószínűség. Minél nagyobb ez az érték, annál kevesebb különböző osztálybeli elem van, amelyek ezzel az értékpárral rendelkeznek, és annál inkább előre jelezhető, hogy ez az értékpár az osztályhoz tartozik.

Nézzük meg, hogy hogyan is működik a COBWEB. A COBWEB folyamatos beszúrásokkal helyezi el az elemeket egy osztályozó fába.

Egy adott új elem esetén vajon hogyan dönti el a COBWEB algoritmus, hogy hová tegye azt a fában? A COBWEB lefelé halad a fában egy megfelelő útvonal mentén, azt a csomópontot keresve, ami a legmegfelelőbb hely lenne az elem osztályozásához. Eközben az érintett csomópontokban levő számlálókat mindig frissíti. A döntést úgy hozza meg, hogy átmenetileg beteszi az elemet minden csomópontba, és kiszámolja a keletkező partíció kategóriahasznosságát. Ahol a legmagasabb lesz a kategóriahasznosság értéke, az lesz a megfelelő hely az elem számára.

De vajon mi történik akkor, ha az elem nem igazán tartozik bele egyik olyan fogalomba sem, amelyek eddig a fában szerepeltek? Mi lenne, ha egy új csomópontot hoznánk létre az elem számára? Ez egy jó ötletnek tűnik. Valójában a COBWEB kiszámolja annak a partíciónak a kategóriahasznosságát is, ami úgy keletkezne, ha egy új csomópontot hoznánk létre az elem számára. Ennek eredményét összehasonlítja a létező csomópontokra elvégzett iménti számításokkal. Majd az elemet attól függően teszi be egy létező osztályba, illetve készít számára egy újat, hogy melyik partíció rendelkezik a legmagasabb kategóriahasznossággal. Érdemes megfigyelni, hogy a COBWEB automatikusan képes arra, hogy megváltoztassa a partíción belüli osztályok számát. Így nem kell a felhasználóra támaszkodnia, hogy ő adjon meg ilyen bemenő paramétert.

Az előbb említett két operátor nagyon érzékeny az elemek beérkezési sorrendjére. A COBWEB két további operátorral is rendelkezik, amelyek segítenek abban, hogy ez az érzékenység csökkenjen. Ez a két operátor az **összevonás** és a **kettéosztás**. Amikor egy elemet beszúrunk, az algoritmus mérlegeli a két legjobb csomópont összevonását is egyetlen osztályba. Ezenkívül a COBWEB azt is számításba veszi, hogy a legjobb csomópont gyerekeit kettéossza a már létező kategóriák között. Ezek a döntések mind a kategóriahasznosságon alapulnak. Az összevonás és a kettéosztás lehetővé teszik a COBWEB

számára, hogy kétirányú keresést végezzen. Például egy összesítés semmisé tehet egy korábbi kettéosztást.

Nézzük meg, hogy mik a COBWEB algoritmus korlátai. Az első, hogy az algoritmus azon a feltételezésen alapul, hogy a különböző attribútumokra vonatkozó valószínűség eloszlások statisztikailag függetlenek egymástól. Ez a feltételezés azonban nem mindig igaz, hiszen az attribútumok között gyakran van korreláció. Egy másik probléma, hogy a klaszterek reprezentálása valószínűség eloszlásokkal, meglehetősen költségessé teszi azok tárolását és frissítését. Ez különösen így van, ha az attribútumok sok különböző értékkel rendelkeznek, hiszen azok idő- és tárbonyolultsága nem csak az attribútumok számától, hanem azok értékeinek számától is függ. Ráadásul az osztályozó fa aszimmetrikus bemenő adatok esetén nem kiegyensúlyozott, ami az idő- és tárbonyolultság drámai romlásához vezethet.

A CLASSIT, ami a COBWEB egy kiterjesztése, folytonos (valós értékű) adatok növekményes klaszterezésére szolgáló algoritmus. Ez az algoritmus minden csomópontban minden attribútumra vonatkozóan tárol egy folytonos normál eloszlást (vagyis átlagot és szórást), és egy módosított kategóriahasznosságot használ. Ez most a folytonos attribútumokra vonatkozóan egy integrál, a COBWEB-beli diszkrét attribútumokra vonatkozó összeg helyett. Ennek az algoritmusnak azonban ugyanazok a gyengéi, mint a COBWEB-nek, és így ez sem alkalmas nagy adatbázisok adatainak klaszterezésére.

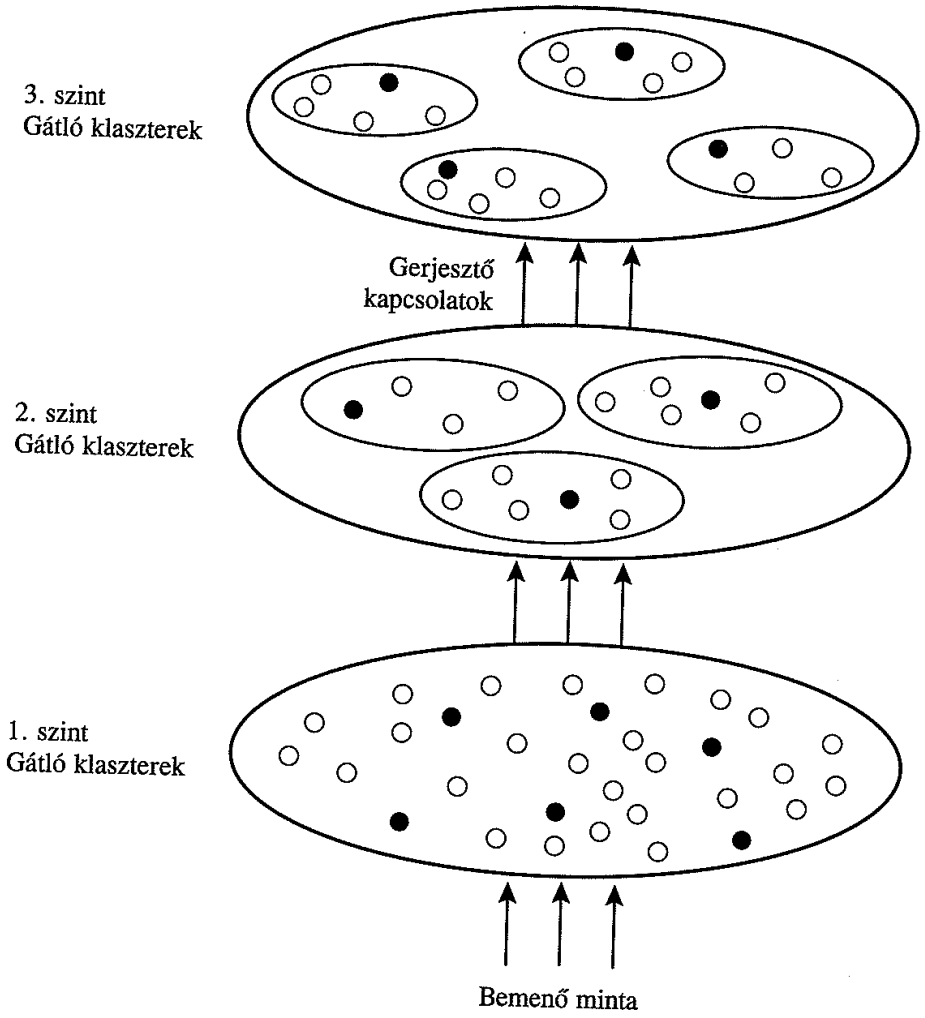
Az iparban az AutoClass népszerű klaszterezési módszer, amely Bayes-statisztikai elemzést használ a klaszterek számának becslésére. A fogalmi klaszterezési módszerek adatbányászati felhasználásának kérdésében még további kutatások szükségesek. Az irodalomjegyzékben további utalásokat adunk meg a témára vonatkozóan.

8.8.2. | Neuronhálós megközelítés

A neuronhálós megközelítés az egyes klasztereket *mintapéldányokként* kezeli. A mintapéldány úgy viselkedik, mint a klaszter prototípusa, de nem kell feltétlenül megfelelnie egy konkrét adatelemnek. Az új elemeket úgy osztjuk szét a klaszterekbe, hogy abba tesszük az elemet, amelyiknek a mintapéldánya a leginkább hasonlít hozzá. Itt a távolságot valamilyen távolsági mértékre alapozzuk. Egy klaszterhez hozzárendelt elem attribútumai nagyjából előre jelezhetők a mintapéldány alapján.

Ebben az alfejezetben két kitűnő példát mutatunk be a klaszterezés neuronhálós megközelítésére. Az első az úgynevezett *versenyző tanulás*, a második pedig az *önszerveződő tulajdonságtérkép*. Mindkét módszer versenyző neuronokkal dolgozik.

A **versenyző tanulás** néhány egység (mesterséges „neuron") hierarchikus rendszeréből áll, és ezek az egységek „győztes mindent visz" alapon versengenek a következő beérkező elemért. A 8.19. ábra egy ilyen elven működő rendszerre mutat be egy példát. Az ábrán minden kör egy egységnek felel meg. Egy klaszteren belül a győztes egység *aktívvá* válik (ezt satírozott kör jelöli), míg a többiek *inaktívak* (ezt üres kör jelöli). A szintek közötti kapcsolatok gerjesztő jellegűek, egy adott szinten lévő egység az összes egy szinttel lejjebbi egységtől fogadhat bemenetet. Egy adott szint aktív elemeinek konfigurációja jelenti a bemeneti mintát a következő, eggyel magasabb szint számára. A klaszteren belüli egységek egymással versenyeznek, hogy az eggyel lejjebbi szint kimenő



8.19. ábra | A versenyző tanulás szerkezete; a szintek száma tetszőleges lehet [RZ85] alapján

mintájára reagáljanak. A szinteken belüli kapcsolatok gátló jellegűek, és egy klaszteren belül csak egy egység lehet aktív. A győztes egység módosítja a klaszteren belüli egységekkel meglévő kapcsolatain szereplő súly értékeket, hogy még erőteljesebben tudjon reagálni a jövőben, az aktuális elemhez hasonló elemekre. Ha úgy tekintjük a súlyokat, mint a mintapéldányok definiáló értékeit, akkor az új elemet ahhoz a klaszterhez kell majd rendelni, amelyik a legközelebbi mintapéldánnyal rendelkezik. A klaszterek száma és a klaszteren belüli egységek száma az algoritmus bemenő paraméterei.

A klaszterezés végén úgy gondolhatunk az egyes klaszterekre, mint egy új „tulajdonságra", ami valamilyen szabályosságot állapít meg az elemek között. Ezért a végeredményül kapott klasztereket tekinthetjük olyan leképezésnek is, ami az alsóbb szintű tulajdonságokat a magasabb szintű tulajdonságokra képezi le.

Az **önszerveződő tulajdonságtérkép** (**SOM**, self-organizing feature maps) szintén úgy működik, hogy néhány egység verseng az aktuális elemért. Az az egység lesz a győztes (aktív), amelynek a súlyvektora a legközelebb van az aktuális elemhez. Annak érdekében, hogy még közelebb kerülhessen a bemenő elemhez, a győztes elem megváltoztatja a súlyértékeit, és a legközelebbi szomszédjainak a súlyértékei is megváltoznak. A SOM algoritmus azt feltételezi, hogy a bemenő elemek között van valamiféle rendezés

vagy topológia, és hogy az egységek végül ezt a térbeli szerkezetet fogják felvenni. Az egységek szerveződéséről azt mondjuk, hogy az egy tulajdonságtérképet alkot. A SOM algoritmusról azt gondolják, hogy működése hasonlít az agy működéséhez. Az algoritmussal sokdimenziós adatokat lehet láthatóvá tenni 2- vagy 3-dimenziós térben.

A klaszterezés neuronhálós megközelítésének erős elméleti kapcsolata van a tényleges agykutatással. Ahhoz, hogy nagy adatbázisokra is valóban alkalmazható lehessen a módszer, még további kutatásokra van szükség. A problémát a hosszú feldolgozási idő és az összetett adatok bonyolultsága okozza.

8.9. | Szélsőséges értékek elemzése

Mik a szélsőséges értékek? Nagyon gyakran előfordul, hogy léteznek olyan adatelemek, amelyek nem felelnek meg az adatok általános viselkedésének, vagy a felállított modellnek. Az ilyen adatelemeket, amelyek durván eltérnek az adathalmaz többi részétől, **szélsőséges értékeknek** nevezzük.

A szélsőséges értékeket mérési vagy végrehajtási hiba is okozhatja. Például, ha egy személy életkoraként -999-et jelenítünk meg, azt egy alapértelmezés szerinti beállítás is okozhatja, ha a program ismeretlen életkorral találkozik. Más esetekben a szélsőséges értékek lehetnek csupán az adatok változatosságának a következményei. Például egy vállalat vezérigazgatójának a fizetése természetes szélsőséges érték lehet a többi dolgozó fizetése között.

A legtöbb adatbányászati algoritmus megpróbálja minimalizálni a szélsőséges értékek hatását, vagy megpróbálja azokat teljesen eltüntetni. Ez persze fontos rejtett információk elvesztéséhez vezethet, hiszen *ami az egyik ember számára zaj, az a másik ember számára fontos jel lehet*. Más szóval a szélsőséges értékek különös jelentőséggel rendelkezhetnek, például a csalások felderítésében, ahol azok a gyanús eseményeket jelezhetik. Ezért a szélsőséges értékek felfedezése és elemzése érdekes adatbányászati feladat, amelyet általában **szélsőséges értékek bányászataként** szoktak emlegetni.

A szélsőséges értékek bányászatának széles körű alkalmazási területe van. Ahogyan már említettük, használják a csalások kiderítéséhez, például a hitelkártyák vagy a telekommunikációs szolgáltatások szokatlan használatának felderítésével. Hasznos továbbá a marketing területén, a kiugróan alacsony vagy kiugróan magas jövedelmű vevők vásárlási szokásainak felderítésében, vagy orvosi elemzésekben, a különböző gyógyászati beavatkozásokra való szokatlan reagálások megtalálásánál.

A szélsőséges értékek bányászata a következőképpen definiálható: Adott egy n elemű adathalmaz és k, a szélsőséges értékek várható száma, találjuk meg azt a k elemet, amelyik leginkább különböző, kivételes vagy eltérő a fennmaradó adatoktól. A szélsőséges értékek bányászata két alproblémára bontható: (1) definiáljuk, hogy mely adatok tekinthetők inkonzisztensnek egy adott adathalmaz esetén, (2) találjunk egy hatékony módszert az így definiált értékek megtalálására.

A szélsőséges értékek definiálása nem triviális. Ha regressziós modellt használunk az adatmodellezéshez, a rezíduumok elemzése egy jó becslés lehet a szélsőségek megállapításához. A feladat azonban jóval nehezebbé válik, ha idősoros adatok között kell a szélsőséges értékeket megtalálnunk, mivel a trendek, szezonális vagy egyéb ciklikus változá-

sok elfedhetik őket. Ha többdimenziós adatokat elemzünk, lehet hogy nem az egyes dimenzióbeli értékek lesznek szélsőségesek, hanem azok egy adott kombinációja. Nem numerikus (például felsorolás típusú) adatok esetén a szélsőséges értékek definiálása különös figyelmet igényel.

Nem használhatnánk-e az adatok vizualizálásának módszerét a szélsőséges értékek felderítéséhez? Ez elég kézenfekvő ötlet, hiszen az emberi szem nagyon gyorsan és hatékonyan ismeri fel az inkonzisztenciákat. Ez azonban nem vonatkozik olyan adatokra, amelyek ciklikus értékeket tartalmaznak, hiszen itt a látszólag szélsőséges értékek teljesen normálisak lehetnek a valóságban. A vizualizálással kapcsolatos módszerek nem túl jók felsorolás típusú attribútumok vagy többdimenziós adatok esetén, hiszen az emberi szem csak két- vagy háromdimenziós adatok felismerésére jó.

Ebben az alfejezetben a fentiek helyett inkább a szélsőséges értékek számítógépes módszerekkel történő keresésével fogunk foglalkozni. Az ilyen módszerek három különböző megközelítés valamelyikén alapulnak, ezek a *statisztikai megközelítés*, a *távolság alapú megközelítés* és az *eltérés alapú megközelítés*. Az alábbiakban mindhárom megközelítést áttekintjük. Érdemes megjegyezni, hogy mivel a klaszterező algoritmusok általában eldobják a szélsőséges értékeket, és zajnak tekintik azokat, kis módosításukkal a szélsőséges értékek megkeresését is elvégeztethetjük velük, mintegy melléktermékként. A felhasználónak általában mindig érdemes ellenőriznie, hogy a fenti módszerekkel megtalált szélsőséges értékek valóban „szélsőségesek"-e.

8.9.1. | Szélsőséges értékek statisztikai alapú keresése

A statisztikai megközelítés az adott adathalmazról feltételez valamilyen eloszlást, vagy valószínűségi modellt (például normális eloszlás), és a szélsőséges értékeket e modellhez képest keresi meg valamilyen *diszkordancia teszt* alkalmazásával. A teszt alkalmazásához szükség van az adathalmaz paramétereinek ismeretére (például a feltételezett eloszlásra), az eloszlás paramétereinek ismeretére (például átlag és szórás), és a szélsőséges értékek várható számára.

Hogyan működik a diszkordancia teszt? Egy statisztikai diszkordancia teszt két hipotézist vizsgál meg, egy *munkahipotézist* és egy *alternatív hipotézist*. Egy H **munkahipotézis** egy olyan állítás, hogy az egész n elemű adathalmaz egy kezdeti F eloszláshoz tartozik, vagyis:

$$H : o_i \in F, \qquad \text{ahol } i = 1, 2, \ldots, n.$$

A hipotézist megtartjuk, ha nincs statisztikailag szignifikáns alapja az elvetésének. Egy **diszkordancia teszt** azt ellenőrzi, hogy az o_i elem szigninfikánsan nagy (vagy kicsi)-e az F eloszláshoz viszonyítva. Diszkordancia tesztként többféle statisztikát szoktak használni attól függően, hogy milyen ismereteink vannak az adatokra vonatkozóan. Tegyük fel, hogy valamilyen T statisztikát kiválasztottunk diszkordancia tesztként, és a statisztika értéke az o_i elemre v_i, ekkor megkonstruáljuk a T eloszlását, majd kiszámoljuk az $SP(v_i) = Prob(T > v_i)$ szignifikancia valószínűséget. Ha valamelyik $SP(v_i)$ kellően kicsi, akkor az o_i elem disszonáns, és a munkahipotézist elvetjük. Egy $\bar{H}$ **alternatív hipo-**

tézist fogadunk el, amely azt mondja ki, hogy o_i egy másik, G eloszláshoz tartozik. Az eredmény nagymértékben függ attól, hogy melyik F-et választottuk, hiszen az o_i szélsőséges érték lehet az egyik eloszlás számára, és teljesen érvényes egy másik számára.

Az alternatív eloszlás nagyon fontos a teszt erejének meghatározásában, ami annak a valószínűsége, hogy a munkahipotézist elvetjük, és o_i tényleg szélsőséges érték. Többféle alternatív eloszlást szoktak használni.

- **Benne rejlő alternatív eloszlás** – Ebben az esetben a munkahipotézist, vagyis hogy az összes elem az F eloszláshoz tartozik, azzal az alternatív hipotézissel szemben vetjük el, hogy az összes elem egy másik, G eloszláshoz tartozik:

  $\bar{H}$: $o_i \in G$, ahol $i = 1, 2, ..., n$.

  F és G lehetnek különböző eloszlások, vagy ugyanannak az eloszlásnak különböző paraméterű változatai. A G eloszlás alakjára vonatkozóan vannak bizonyos megszorítások, amennyiben annak lehetővé kell tennie a szélsőséges értékeket. Lehet például más átlaga, diszperziója, vagy hosszabb farka.
- **Vegyes alternatív eloszlás** – A vegyes alternatív eloszlás azt mondja, hogy a disszonáns értékek nem szélsőségesek az F populációban, hanem szennyeződések egy másik, G populációból. Ez esetben az alternatív hipotézis a következő:

  $\bar{H}$: $o_i \in (1 - \lambda)F + \lambda G$, ahol $i = 1, 2, ..., n$.

- **Csúszó alternatív eloszlás** – Ez az alternatíva azt mondja, hogy az összes elem (kivéve néhány előre megadott elemet) egymástól függetlenül a kezdeti, megadott paraméterekkel rendelkező F eloszláshoz tartozik, míg a többi elem független megfigyelések F-nek egy módosított változatából, ahol a paramétereket elcsúsztattuk.

A szélsőséges értékek felismerésére alapvetően két különböző eljárás létezik.

- **Blokkeljárások** – Ebben az esetben vagy az összes gyanús elemet szélsőséges értékként kezeljük, vagy az összes elemet elfogadjuk normális értéknek.
- **Egymást követő eljárások** – Ilyen eljárásra jó példa az úgynevezett *belülről kifelé* eljárás. Ennek az az alapötlete, hogy először azt az elemet teszteljük, amelyikről a legkevésbé valószínű, hogy szélsőséges érték. Ha ez szélsőséges érték lenne, akkor a többi ennél szélsőségesebb értéket mindet szélsőséges értéknek tekintjük. Ellenkező esetben vesszük a következő legkevésbé valószínű elemet és így tovább. Ez az eljárás hatékonyabbnak tűnik, mint a blokkeljárások.

Nézzük meg, hogy mennyire eredményes a statisztikai megközelítés a szélsőséges értékek felfedezésében. A legnagyobb hátránya, hogy a legtöbb teszt egyetlen attribútumot vizsgál, míg sok adatbányászati probléma esetén a szélsőséges értékeket többdimenziós térben kell megkeresni. Egy további hátránya, hogy a statisztikai megközelítéshez ismernünk kell az adathalmaz bizonyos paramétereit, például az eloszlását, ami sok esetben

nem ismert. A statisztikai módszerek nem garantálják, hogy az összes szélsőséges értéket megtalálják azokban az esetekben, amelyekre nem fejlesztettek ki speciális módszert, vagy ha az adatok megfigyelt eloszlása nem megfelelően modellezhető valamilyen ismert eloszlással.

8.9.2. | Távolság alapú szélsőséges értékek keresése

A távolság alapú szélsőséges értékek fogalmát azért vezették be, hogy a statisztikai módszerek korlátait túlléphessük.

Mi is egy távolság alapú szélsőséges érték? Az S adathalmaz o elemét **távolság alapú (DB, distance based) szélsőséges értéknek** nevezzük p és d paraméterekkel, ha az S-beli elemeknek legalább p része d-nél nagyobb távolságra van o-tól. Ennek jelölése $DB(p, d)$. Más szavakkal kifejezve, a statisztikai tesztekre való támaszkodás helyett, a távolság alapú szélsőséges értékekre úgy gondolhatunk, mint azokra az elemekre, amelyeknek nincs „elegendő" szomszédjuk. A szomszédságot most az adott elemtől való távolság alapján definiáljuk. A statisztikai módszerekkel összehasonlítva, a távolság alapú szélsőséges értékek keresése általánosítja a diszkordancia tesztek ötletét különböző standard eloszlásokra. A távolság alapú keresés elkerüli azokat a kimerítő számításokat, amik a megfigyelt eloszlásnak valamilyen standard eloszláshoz való illesztésével, és a diszkordancia tesztek kiválasztásával járnak.

Számos diszkordancia teszt esetén megmutatható, hogy ha egy o elem szélsőséges érték az adott teszt alapján, akkor o egy $DB(p, d)$ is egyben megfelelően választott p-re és d-re. Például ha azok az elemek tekintendők szélsőséges értéknek, amelyek legalább háromszoros szórásnyira vannak az átlagtól, normál eloszlás esetén, akkor ezt a definíciót általánosíthatjuk a $DB(0.9988, 0.13\sigma)$ értékkel.[3]

A távolság alapú szélsőséges értékek megkeresésére számos hatékony algoritmust fejlesztettek ki. A következőkben ezeket ismertetjük.

Index alapú algoritmus – Adott adathalmaz esetén az index alapú algoritmus többdimenziós indexstruktúrákat használ, mint például az R-fák vagy a $k - d$ fák, és ezek segítségével keresi meg minden egyes o elem d sugarú környezeten belüli szomszédjait. Legyen M a szélsőséges értékek körüli d sugarú környezetben maximálisan megengedett elemek száma. Így ha egy o elemnek $M + 1$ szomszédját találtuk meg, akkor világos, hogy o nem szélsőséges érték. Ez az algoritmus a legrosszabb esetben $O(k \times n^2)$ bonyolultságú, ahol k a dimenziók száma, n pedig az adathalmaz elemeinek száma. Az index alapú algoritmus jól skálázható k növekedésével. Az előbbi bonyolultság azonban csak a keresési időt veszi figyelembe, pedig magának az indexnek a felépítése is sok számítást igényelhet.

Egymásba ágyazott ciklusú algoritmus – Az egymásba ágyazott ciklusú algoritmusnak ugyanaz a számítási bonyolultsága, mint az index alapúnak, de ez az algoritmus nem épít

3 | A p és d paramétereket a normál görbe sűrűségfüggvényével számoljuk ki úgy, hogy a következő feltétel teljesüljön: $(P|X - 3| \leq d) < 1 - p$, azaz $P(3 - d \leq X \leq 3 + d) < 1 - p$. (Nem biztos, hogy a megoldás egyértelmű). A 0,13 sugár körüli d-szomszédság jelöli a 3σ jel körüli $\pm 0{,}13$ egységet (azaz [2.87, 3.13]). A teljes bizonyítást lásd [KN97]-ben.

fel indexstruktúrákat, és megpróbálja minimalizálni az I/O-műveletek számát. A memóriát két részre, az adathalmazt pedig néhány logikai blokkra osztja. Ezután gondosan megválasztja azt a sorrendet, ahogyan a blokkokat az egyes memóriarészekbe beolvassa, ezáltal éri el az I/O-műveletekre vonatkozó hatékonyságot.

Cella alapú algoritmus – Az $O(n^2)$ számítási bonyolultság elkerülésére fejlesztették ki a cella alapú algoritmust, olyan adathalmazokra, amelyek a memóriában tarthatók. Ennek az algoritmusnak a bonyolultsága $O(c^k + n)$, ahol a c a cellák számától függő konstans, k pedig a dimenziók száma. Ennél a módszernél az adatok terét cellákra osztjuk úgy, hogy a cellák oldala $\frac{d}{2\sqrt{k}}$ hosszú legyen. Minden cellát két réteg vesz körül. Az első réteg egy cella vastagságú, a második réteg vastagsága pedig $\left\lfloor 2\sqrt{k} - 1 \right\rfloor$ (felső egészrész) cellányi. Az algoritmus a szélsőséges értékeket *cellánként* számolja, és nem elemenként. Egy adott cella esetén három számot tart nyilván: a cellabeli elemek számát, a cellában és az első rétegben együttesen elhelyezkedő elemek számát, valamint a cellában és a két rétegben összesen meglévő elemek számát. Jelöljük ezeket a számokat rendre a következőkkel: *cellánkénti_szám*, *cella+1_rétegbeli_szám*, *cella+2_rétegbeli_szám*.

Nézzük meg, hogy e módszer hogyan találja meg a szélsőséges értékeket. Legyen M a szélsőséges értékek körüli d sugarú környezetben maximálisan megengedett elemek száma.

- Az aktuális cella egy o elemét akkor tekintjük szélsőséges értéknek, ha *cella+1_rétegbeli_szám* kisebb vagy egyenlő, mint M. Ha e feltétel nem teljesül, akkor a cella összes elemét kivehetjük a további vizsgálatokból, mivel azok nem lehetnek szélsőséges értékek.
- Ha *cella+2_rétegbeli_szám* kisebb vagy egyenlő, mint M, akkor a cella összes elemét szélsőséges értéknek tekintjük. Ellenkező esetben, ha ez a szám nagyobb, mint M, akkor lehetséges, hogy a cella néhány eleme szélsőséges érték. Ezek megkeresését elemenként végezzük úgy, hogy a cella minden egyes o elemére az o-hoz tartozó második réteg elemeit vizsgáljuk meg. A cella elemei közül csak azok a szélsőséges értékek, amelyek d sugarú környezetében legfeljebb M elem van. Egy elem d sugarú környezete az elem cellájából, a teljes első rétegből, és a második réteg egy részéből áll.

Az algoritmus egy módosított változata lineáris az n függvényében, és garantálja, hogy legfeljebb háromszor kell végigmenni az adatokon. Ezt lemezen tárolt, nagy adathalmazokra is használhatjuk, viszont nem túl jól skálázható magasabb dimenziószámok esetén.

A távolság alapú szélsőséges értékek keresése megköveteli a felhasználótól a p és d paraméterek megadását. Ezen paraméterek megfelelő értékeinek megtalálása sok kísérletezéssel és hibalehetőséggel járhat.

8.9.3. | Szélsőséges értékek eltérés alapú keresése

A szélsőséges értékek eltérés alapú keresése nem használ statisztikai teszteket vagy távolság alapú mértékeket a kivételes elemek azonosításához. Ehelyett az egyes csoportok elemeinek fő jellemzőit vizsgálja meg, és ennek segítségével azonosítja a szélsőséges értékeket. Azokat az elemeket tekinti szélsőséges értéknek, amelyek „eltérnek" az előbbi

jellemzőktől. Ezért ebben a megközelítésben általában az *eltérő* szót használjuk a szélsőséges értékekre. Az alábbiakban két eltérés alapú módszert mutatunk be a szélsőséges értékek keresésére. Az első egymás után összehasonlítja egy halmaz elemeit, míg a második egy OLAP-adatkocka megközelítést használ.

Egymás utáni kivételek módszere

Az egymás utáni kivételek módszere azt az eljárást szimulálja, ahogyan az emberek a szokatlan elemeket választják ki a feltételezhetően hasonló elemek egy sorozatából. Az algoritmus kihasználja az adatok implicit redundanciáját. Adott n elemű S halmazra létrehozza részhalmazoknak egy $\{S_1, S_2, \ldots, S_m\}$ sorozatát, ahol $2 \leq m \leq n$, és teljesülnek a következők:

$$S_{j-1} \subset S_j, \qquad \text{ahol } S_j \subseteq S.$$

A különbözőségek értékelését a sorozatbeli részhalmazokra végezzük el. A módszerhez be kell vezetnünk a következő fogalmakat.

- **Kivételek halmaza** – Ez az eltérő vagy szélsőséges értékek halmaza. Ezt úgy definiáljuk, mint elemeknek azt a legkisebb halmazát, amelynek eltávolítása a legnagyobb csökkenést eredményezi a megmaradó halmaz (alább definiált) különbözőségében.[4]
- **Különbözőségi függvény** – Ez a függvény nem igényli valamiféle metrikus távolság meglétét az elemek között. Ez bármilyen függvény lehet, ami egy adott elemhalmazra alacsony értéket ad vissza, ha az elemek hasonlóak egymáshoz. Minél nagyobb a különbözőség az elemek között, a függvény annál nagyobb értéket ad vissza. Egy részhalmaz különbözőségét a sorban előtte levő részhalmaz alapján számítjuk ki, induktív módon. Ha adott egy n darab számból álló halmaz, akkor egy lehetséges különbözőségi függvény a számok varianciája, azaz

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2, \tag{8.28}$$

  ahol $\bar{x}$ a halmazbeli n darab szám átlaga. Karakterláncok esetén a különbözőségi függvény lehet egy minta karakterlánc (ez joker karaktereket is tartalmazhat), ami illeszkedik az összes korábbi mintára. A különbözőség növekszik, ha az S_{j-1}-beli karakterláncokra illeszkedő minta nem illeszkedik valamely S_j-beli karakterláncra, ami S_{j-1}-ben nem volt benne.
- **Számosság függvény** – Ez általában egy halmazbeli elemek száma.
- **Simító tényező** – Ezt a függvényt a sorozat minden részhalmazára kiszámoljuk. A függvény azt becsüli meg, hogy mennyivel csökkenthető a különbözőség, ha az adott részhalmazt elhagyjuk az eredeti elemhalmazból. Ez az érték a halmaz számosságával

4 | Az érdeklődő olvasók számára megemlítjük, hogy ez ekvivalens azzal az értékkel, amit úgy kapunk, hogy vesszük az eldobott elemek Kolmogorov-bonyolultságának lehetséges legnagyobb csökkenését.

együtt növekszik. Az a részhalmaz lesz a kivételek halmaza, amelynek a simító tényezője a legnagyobb.

A kivételek halmazának megtalálása általános esetben NP-nehéz is lehet. A fenti sorozatot használó megközelítés a számításigény szempontjából még megvalósítható, és lineáris algoritmussal implementálható.

Nézzük meg, hogy hogyan működik a módszer. Ahelyett, hogy az aktuális részhalmaz különbözőségét becsülnénk meg a komplementer halmazának figyelembevételével, az algoritmus részhalmazok egy sorozatát választja ki a vizsgálandó halmazból. Minden részhalmazra meghatározza a különbözőség megváltozását a sorozatban előtte levő részhalmazhoz képest.

Jogosan vetődik fel a kérdés, hogy vajon a részhalmazok sorrendje nem befolyásolja-e az eredményt. A bemenet sorrendjének a végeredményre gyakorolt hatását úgy csökkenthetjük, hogy a fenti folyamatot többször megismételjük, minden esetben véletlenül megválasztva a részhalmazok sorrendjét. Végül az a halmaz lesz a kivételek halmaza, amelynek a simító tényezője az összes iterációt figyelembe véve a legnagyobb.

OLAP adatkocka megközelítés

Az eltérések keresésének OLAP-megközelítése adatkockákat használ a rendellenes területek azonosítására, többdimenziós, nagy adathalmazok esetén. Ezt a megközelítést a 2. fejezetben részletesen ismertettük. A hatékonyság további növelésének céljából, az eltéréseket kereső folyamat a kockaszámításokkal egyidejűleg, egymást átfedő módon is történhet. Ez a megközelítés a *felfedezésvezérelt feltárás* egyik formája, ahol a kivételeket jelző, előre kiszámolt értékek segítik a felhasználót az elemzésben, az összesítések minden szintjén. A kockában egy cella értékét kivételnek tekintjük, ha az jelentősen eltér egy statisztikai modell alapján elvárt értéktől. A módszer vizuális figyelmeztetéseket használ, mint például a háttérszín, és ezzel jelzi az egyes cellák kivételességének mértékét. A felhasználó az így megjelölt cellákra választhatja a lefúrás műveletet. A cella értéke olyan kivételeket is megmutathat, amelyek a kockának részletezettebb vagy *alacsonyabb szintjén* vannak, és amelyek az adott szintről nem láthatók.

Ez a modell az *összes olyan dimenzió* értékeinek változásait és mintáit figyelembe veszi, amely dimenziókhoz a cella tartozik. Tegyük fel például, hogy egy olyan adatkockánk van, amelyben eladási információk vannak, és éppen a havonkénti összesített eladásokat vizsgáljuk. A vizuális jelekből látjuk, hogy december hónapban növekedtek az eladások a többi hónaphoz viszonyítva. Ez kivételnek tűnhet az idő dimenzióra vonatkozóan. Ha azonban lefúrunk a december hónapban és a tételes eladásokat is megvizsgáljuk, akkor azt láthatjuk, hogy decemberben más tételek eladásai is növekedtek. Ezért az összes eladás december havi növekedése nem kivétel, ha a tételek dimenziót vesszük figyelembe. A modell olyan kivételeket is figyelembe vesz, amelyek az adatkocka minden összesítési szintjén rejtve maradnak. Az ilyen kivételek manuális felderítése nagyon nehéz, mivel a keresési tér általában nagyon nagy. Ez különösen igaz olyan esetekben, amikor sok dimenzió több szintű fogalomhierarchiával rendelkezik.

8.10. | Összefoglalás

- A **klaszter** olyan adatelemek gyűjteménye, amelyekre igaz, hogy az azonos klaszteren belüli elemek *hasonlóak* egymáshoz, míg a különböző klaszterbeli elemek különböznek egymástól. Azt a folyamatot, amikor tényleges vagy absztrakt elemeket *hasonló* elemekből álló osztályokba csoportosítunk, **klaszterezésnek** nevezzük.
- A klaszterelemzések széles **felhasználási területe** van, ilyenek többek között a piac- vagy vevőszegmentálás, a mintafelismerés, a biológiai kutatások, a térbeli adatok elemzése vagy a webdokumentumok elemzése. A klaszterelemzés önálló adatbányászati eszközként használható, aminek a segítségével betekintést nyerhetünk az adatok eloszlásába, illetve más adatbányászati algoritmusok előfeldolgozó lépéseként, amik majd a kapott klasztereken hajtanak végre műveleteket.
- A klaszterezés minőségét az elemek **különbözőségének** mértéke alapján állapíthatjuk meg. Ezt **különböző típusú adatokra** számolhatjuk ki, ilyenek az *intervallum alapú*, *bináris*, *felsorolás típusú*, *rendezett*, *arányskálázott adatok* vagy ezek valamilyen kombinációja.
- A klaszterezés az adatbányászatnak egy dinamikus kutatási területe. Számos klaszterező algoritmust fejlesztettek ki. Ezeket a következő **kategóriákba sorolhatjuk**: *particionáló módszerek*, *hierarchikus módszerek*, *sűrűség alapú módszerek*, *rács alapú módszerek* és *modell alapú módszerek*.
- Egy **particionáló módszer** először létrehoz egy kezdeti, k partícióból álló halmazt, ahol k a létrehozandó partíciók száma. Ezután valamilyen *iteratív áthelyezési technika* segítségével megpróbál javítani a particionáláson úgy, hogy elemeket helyez át egyik csoportból a másikba. Jellegzetes particionáló módszerek a k-átlag, a k-medoid, a CLARANS, illetve ezek javításai.
- A **hierarchikus módszer** az adatelemek hierarchikus dekompozícióját hozza létre. A módszereket *egyesítő (lentről fel)* és *felosztó (föntről le)* módszerekre osztjuk aszerint, hogy hogyan végzik a hierarchikus dekompozíciót. Az egyesítés és a felosztás lépések merevségét kompenzálhatjuk, és így a klaszterezés minőségét javíthatjuk az elemek közötti kapcsolatok minden hierarchikus particionáló lépés előtti elemzésével (mint például a CURE és Chameleon algoritmusokban), vagy más klaszterező módszerek egyidejű használatával (mint például az iteratív áthelyezés a BIRCH algoritmus esetén).
- Egy **sűrűség alapú módszer** az elemeket a sűrűség fogalma alapján klaszterezi. A klaszterek növelését vagy a szomszédos elemek sűrűsége alapján végzi (mint a DBSCAN algoritmus), vagy valamilyen sűrűségfüggvény alapján (mint például a DENCLUE algoritmus). Az OPTICS egy olyan sűrűség alapú módszer, amely az adatok klaszterezési szerkezeteinek egy növekvő rendezését hozza létre.
- Egy **rács alapú módszer** először felosztja az elemek terét véges sok cellára, amelyek egy rácsszerkezetet alkotnak, majd a klaszterezést a rácsszerkezetre hajtja végre. A STING egy jellemző példája a rács alapú módszereknek, amely statisztikai információkat tárol a rács celláiban. A CLIQUE és a WaveCluster két olyan klaszterező módszer, amelyek egyidejűleg rács alapúak és sűrűség alapúak.

- Egy **modell alapú módszer** minden klaszterről egy modellt feltételez, és megkeresi az adott modellhez legjobban illeszkedő adatokat. A jellemző modell alapú módszerek vagy statisztikai megközelítést használnak (mint a COBWEB, a CLASSIT és az AutoClass), vagy valamilyen neuronhálós megközelítést, mint amilyen például a versenyző tanulás, vagy az önszerveződő tulajdonságtérkép.
- *Ami az egyik ember számára zaj, az a másik ember számára jel lehet.* A **szélsőséges értékek felismerése és elemzése** nagyon hasznos a csalások felderítésénél, a marketing területén, az orvosi elemzésekhez és más további feladatok esetén. A szélsőséges értékek számítógépes elemzési módszerei általában a *statisztikai megközelítést*, a *távolság alapú megközelítést* vagy az *eltérés alapú megközelítést* használják.

8.11. | Feladatok

8.1. Röviden vázoljuk, hogy hogyan lehet a különbözőséget kiszámítani az alábbi típusú változókkal megadott elemek esetén:
(a) aszimmetrikus bináris változók,
(b) felsorolás típusú változók,
(c) arányskálázott változók,
(d) numerikus (intervallum alapú) változók.

8.2. Adottak a következő értékek az *életkor* változóra:
18, 22, 25, 42, 28, 43, 33, 35, 56, 28,
standardizáljuk a változót a következőképpen:
(a) Számoljuk ki az *életkor* átlagos abszolút eltérését.
(b) Számoljuk ki az első négy értékre vonatkozó z-értéket.

8.3. Adott két elem, amelyeket a következő sorok (rekordok) reprezentálnak: (22, 1, 42, 10) és (20, 0, 36, 8).
(a) Számoljuk ki a két elem közötti euklideszi távolságot.
(b) Számoljuk ki a két elem közötti Manhattan-távolságot.
(c) Számoljuk ki a két elem közötti Minkovszki-távolságot $q = 3$ használatával.

8.4. Az alábbi táblázat a következő attribútumokat tartalmazza: *név*, *nem*, *jellemző_1*, *jellemző_2*, *jellemző_3*, *jellemző_4*, ahol a *név* az elem azonosítója, a *nem* egy szimmetrikus attribútum, a többi attribútum pedig aszimmetrikus, és olyan személyek jellemzőit adják meg, akik levelezőtársat keresnek. Tegyük fel, hogy van egy olyan szolgáltatás, amelyik megpróbál megfelelő levelezőtársakból álló párokat találni.

név	nem	jellemző_1	jellemző_2	jellemző_3	jellemző_4
Kevan	F	N	P	P	N
Caroline	N	N	P	P	N
Erik	F	P	N	N	P
⋮	⋮	⋮	⋮	⋮	⋮

Az aszimmetrikus attribútumokra legyen a *P* érték 1-re állítva, az *N* érték pedig 0-ra.

Tegyük fel, hogy az elemek (potenciális levelezőtársak) közötti távolság csupán az aszimmetrikus változók alapján számítódik.

(a) Mutassuk meg a *kontingenciamátrixot* minden elem párra. Kevan, Caroline és Erik adottak.
(b) Számítsuk ki az *egyszerű egyezőségi tényezőt* minden párra.
(c) Számítsuk ki a *Jaccard-tényezőt* minden párra.
(d) Mit javasol, kik lennének a legmegfelelőbb levelezőtársak? Melyik pár lenne a legkevésbé megfelelő?
(e) Tegyük fel, hogy bevonjuk a *nem* szimmetrikus változót az elemzésünkbe. A Jaccard-tényező alapján kik lennének a legmegfelelőbb pár, és miért?

8.5. Mi a klaszterezés? Röviden jellemezzük a következő klaszterezési megközelítéseket: *particionáló módszerek*, *hierarchikus módszerek*, *sűrűség alapú módszerek*, *rács alapú módszerek* és *modell alapú módszerek*. Adjunk példákat mindegyik esetre.

8.6. Tegyük fel, hogy a következő nyolc pontot kell három klaszterbe klasztereznünk [(x, y) elhelyezkedést jelölnek]:

$A_1(2, 10)$, $A_2(2, 5)$, $A_3(8, 4)$, $B_1(5, 8)$, $B_2(7,5)$, $B_3(6, 4)$, $C_1(1, 2)$, $C_2(4, 9)$.

A távolságfüggvény az euklideszi távolság. Tegyük fel, hogy kezdetben a három klaszter középpontja rendre A_1, B_1, és C_1. Használjuk a k-átlag algoritmust és adjuk meg:

(a) a klaszterek középpontjait az első végrehajtási lépés után,
(b) a végeredményül adódó három klasztert.

8.7. Egy ábra segítségével mutassuk meg, hogy konstans *MinPts* értékre, a nagyobb sűrűséghez (vagyis kisebb ε szomszédsági sugárhoz) tartozó *sűrűség alapú klaszterek* teljesen részét képezik a kisebb sűrűséghez tartozó sűrűn összekötött halmazoknak.

8.8. Az emberi szem gyorsan és hatékonyan értékeli a klaszterezés minőségét kétdimenziós adatok esetén. Tervezzünk egy vizualizációs módszert, ami segíthet háromdimenziós adatok klaszterezésének értékelésében. Mi a helyzet még magasabb dimenziószám esetén?

8.9. Adjunk példát arra, amikor különböző klaszterezési módszereket *együttesen* lehet használni. Például az egyik módszer előfeldolgozó lépése a másiknak.

8.10. A klaszterezés fontos adatbányászati feladat, amely széles alkalmazási területtel rendelkezik. Adjunk meg egy-egy példaalkalmazást az alábbi esetekre:

(a) egy olyan alkalmazásra, amelynek a klaszterezés a fő adatbányászati feladata;
(b) egy olyan alkalmazásra, amely a klaszterezést előfeldolgozó lépésként használja valamilyen más adatbányászati feladathoz.

8.11. Az adatkockák és a többdimenziós adatbázisok a felsorolás típusú, rendezett és numerikus adatokat általában hierarchikus és összesített formában tartalmazzák. Az eddigiekben tanultak alapján tervezzünk egy klaszterezési módszert, amelyik a nagy adatkockákban hatékonyan és eredményesen találja meg a klasztereket.

8.12. Tegyük fel, hogy el kell helyeznünk néhány pénzkiadó automatát (ATM) egy adott körzetben úgy, hogy bizonyos megszorításoknak is eleget tegyünk. A háztartásokat és a munkahelyeket úgy kellene klasztereznünk, hogy általában egy automata jusson egy klaszterre. A klaszterezést azonban korlátozhatják olyan tényezők, mint a hidak, folyók és országutak elhelyezkedése, ami befolyásolhatja az automaták elér-

hetőségét. További megszorításokat jelenthet az automaták darabszámának kerületenkénti korlátozása, ahol ezek a kerületek alkotják a körzetet. Ilyen megszorítások mellett hogyan módosíthatók a klaszterező algoritmusok úgy, hogy azok alkalmasak legyenek a *megszorítások melletti klaszterezésre*?

8.13. Miért fontos a szélsőséges értékek bányászata? Röviden jellemezzük a különböző megközelítéseket, amelyek a következő módszerek mögött vannak: *szélsőséges értékek statisztikai alapú keresése*, *szélsőséges értékek távolság alapú keresése* és *szélsőséges értékek eltérés alapú keresése*.

8.12. | Irodalom

A klaszterező módszereket több könyv is tárgyalja, mint például Hartigan [Har75], Jain és Dubes [JD88], Kaufman és Rousseeuw [KR90]. A klaszterezésről egy friss áttekintést adott Jain, Murty és Flynn [JMF99]. Hogy a különböző típusú változókat hogyan lehet egyetlen különbözőségi mátrixban egyesíteni, erre vonatkozó módszereket Kaufman és Rousseeuw mutattak be [KR90].

A particionáló módszerek közül elsőként a *k*-átlag algoritmust mutatta be Mac-Queen [Mac67]. A PAM és CLARA *k*-medoid algoritmusokat Kaufman és Rousseeuw javasolták [KR90]. A *k*-módusz (felsorolás típusú adatokra) és *k*-prototípus (hibrid adatokra) algoritmusokat Huang javasolta [Hua98], az EM (elvárások maximalizálása) algoritmust pedig Lauritzen [Lau95]. A CLARANS algoritmust Ng és Han vezették be [NH94]. A CLARANS teljesítményének további javítására vonatkozó eljárásokat javasolt Ester, Kriegel és Xu [EKX95]. A javaslatukban hatékony térbeli elérési módszerek szerepeltek, mint amilyen például az R* fa, vagy a fokuszáló technikák. Egy másik skálázható, *k*-átlag alapú módszert javasolt Bradley, Fayyad és Reina [BFR98].

Az egyesítő hierarchikus klaszterezést, mint például az AGNES, és a felosztó hierarchikus klaszterezést, mint például a DIANA, Kaufman és Rousseeuw vezették be [KR90]. Egy érdekes irányzat a hierarchikus klaszterezési módszerek minőségének javítására, azok egyidejű használata a távolság alapú iteratív áthelyezéssel, vagy más nem hierarchikus módszerekkel. Például a Zhang, Ramakrishnan és Livny [ZRL96] által ajánlott BIRCH először hierarchikus klaszterezést végez a CF-fa segítségével, majd más módszereket alkalmaz. A hierarchikus klaszterezést kiegészíthetjük kifinomult kapcsolatelemzéssel, átalakítással, vagy legközelebbi szomszéd elemzéssel, mint a Guha, Rastogi és Shim [GRS98] által javasolt CURE algoritmusban, az ugyanezen szerzők által javasolt ROCK algoritmusban [GRS98], és a Karypis, Zhan és Kumar által javasolt Chameleon algoritmusban [KHK99].

A sűrűség alapú algoritmusok közül a DBSCAN-t Ester, Kriegel, Sander és Xu javasolták [EKSX96]. Ankerst, Breunig, Kriegel és Sander [ABKS99] az OPTICS algoritmusban kifejlesztettek egy klaszter rendezési módszert, amely lehetővé teszi a sűrűség alapú klaszterezést anélkül, hogy a paraméterek megadásával kellene törődnünk. A DENCLUE algoritmust, amely sűrűség eloszlás függvények egy halmazán alapul, Hinneburg és Keim ajánlották [HK98].

A rács alapú, többszörös felbontást használó STING módszert, amely statisztikai információkat gyűjt a cellákban, Wang, Yang és Muntz ajánlották [WYM97]. A

WaveCluster, amelyet Sheikholeslami, Chatterjee és Zhang fejlesztettek ki [SCZ98], egy többszörös felbontásos klaszterező módszer, amely az eredeti adatteret egy hullámtranszformációval átalakítja. Az Agrawal, Gehrke, Gunopulos és Raghavan által kifejlesztett CLIQUE algoritmus [AGGR98] egyidejűleg sűrűség alapú és rács alapú módszer, amely sokdimenziós adatok klaszterezésére alkalmas.

A modell alapú klaszterezést elindító cikkek közül érdemes megemlíteni Shavlik és Dietterich cikkét [SD90]. A fogalmi klaszterezést először Michalski és Stepp vezette be [MS83]. További példák a statisztikai megközelítésre a Fisher által bevezetett COBWEB [Fis87], a Gennari, Langley és Fisher által bevezetett CLASSIT [GLF89], és a Cheeseman és Stutz által bevezetett AutoClass [CS96a]. A neuronhálós megközelítésre vonatkozó tanulmányok közül megemlítjük Rumelhart és Zipser versenyző tanulásra vonatkozó cikkét [RZ85], valamint Kohonen önszerveződő tulajdonságtérképekre vonatkozó cikkét [Koh82].

A felsorolás típusú adatok skálázható klaszterezésére vonatkozó módszereket Gibson, Kleinberg és Raghavan [GKR98], Guha, Rastogi és Shim [GRS99], valamint Ganti, Gehrke és Ramakrishnan tanulmányozták [GGR99]. Számos további klaszterezési paradigma ismeretes még. Például a fuzzy klaszterező módszereket tárgyalja Kaufman és Rousseeuw [KR90], valamint Bezdek és Pal [BP92].

A szélsőséges értékek keresését és elemzését három megközelítés valamelyikébe sorolhatjuk, ezek a statisztikai megközelítés, a távolság alapú megközelítés és az eltérés alapú megközelítés. A statisztikai megközelítést és a diszkordancia teszteket Barnett és Lewis ismerteti [BL94]. A távolság alapú módszereket Knorr és Ng írják le [KN97, KN98]. A szélsőséges értékek eltérés alapú keresésének szekvenciális megközelítését Arning, Agrawal és Raghavan vezették be [AAR96]. Sarawagi, Agrawal és Megiddo egy felfedezésvezérelt módszert javasoltak a kivételek azonosítására, OLAP-adatkockákat használó, nagy adathalmazok esetére [SAM98]. Jagadish, Koudas és Muthukrishnan egy hatékony módszert vezettek be az eltérések bányászatára idősoros adatbázisokban [JKM99].

9. FEJEZET

Komplex adattípusok bányászata

Az adatbányászat módszerei terén folytatott korábbi vizsgálódásaink a relációs, a tranzakciós adatbázisokban, valamint a strukturált adatok átalakításával és integrálásával kialakított adattárházakban végzett bányászatra koncentráltak. Az adatgyűjtő eszközök gyors fejlődésének, a fejlett adatbázisrendszereknek és a Word Wide Web (WWW) technológiáknak köszönhetően szinte robbanásszerűen óriási adatmennyiség gyűlt össze. Így az adatbányászat területén egyre fontosabb feladat a komplex adattípusok, beleértve a *komplex objektumokat,* a *tér-, multimédia-, idősor-, szöveges adatok* és a *World Wide Web* bányászata (a komplex adatokról lásd 1.3.4. alfejezet).

Ebben a fejezetben megvizsgáljuk a fontosabb adatbányászati technikák (karakterizálás, társítás, osztályozás és klaszterezés) fő fejlődési irányait, illetve hogy milyen új módszereket kell kifejleszteni ahhoz, hogy meg tudjunk birkózni a komplex adattípusokkal és a komplex információtárolókban eredményes adatbányászatot hajthassunk végre. A fejezet felépítése a következő. A 9.1. alfejezet a komplex adatobjektumok többdimenziós elemzésével és leíró bányászatával foglalkozik. A 9.2. alfejezet a téradatbázisok, a 9.3. alfejezet a multimédia-adatbázisok, a 9.4. alfejezet az idősor-adatbázisok, a 9.5. alfejezet a szöveges adatbázisok, és a 9.6. alfejezet a Word Wide Web adatbányászatát írja le. Mivel a komplex adatbázisok bányászatával kapcsolatos kutatások rohamléptékkel folynak, áttekintésünk csak néhány előzetes témát érint. Várható, hogy sok olyan könyv jelenik még meg, amely a komplex adattípusok bányászatával foglalkozik.

9.1. | Komplex adatobjektumok többdimenziós analízise és leíró bányászata

Sok üzleti adattárház és multidimenziós adatbázis analízisét szolgáló OLAP-eszköz fő korlátja a dimenziók és a mértékek adattípusainak szegényes készlete. A legtöbb adatkocka létrehozásánál a dimenziókhoz nem numerikus adatok, a mértékekhez egyszerű, aggregált értékek kötődnek. Hogy bevezethessük a komplex adatobjektumok többdimenziós analízisét és bányászatát, ebben az alfejezetben megvizsgáljuk, hogyan kell elvégezni a komplex, strukturált objektumok általánosítását, és az OLAP, valamint az objektum-adatbázisok bányászata céljából objektumkockákat létrehozni.

A **komplex strukturált adatok** tárolásának, az adatokhoz történő hozzáférésnek a témakörét már az objektumrelációs és az objektumorientált adatbázisrendszerek kereté-

ben feldolgoztuk. Ezek a rendszerek a komplex adatobjektumok nagy csoportját *osztályokba* szervezik, amelyek azután *osztály–alosztály* hierarchiákba szerveződnek. Egy osztály mindegyik **objektuma** rendelkezik (1) *objektumazonosítóval* és (2) *attribútumkészlettel*, ahol az attribútumok tartalmazhatnak komplex adatstruktúrákat, halmazértékű vagy listaértékű adatokat, osztályokból szervezett hierarchiákat, multimédia-adatokat és így tovább; valamint (3) *metóduskészlettel*, ami specifikálja az objektumosztályra vonatkozó számítási rutinokat vagy szabályokat.

Hogy megkönnyítsük az objektumrelációs és objektumorientált adatbázisok általánosítását és indukcióját, meg kell vizsgálni, hogyan kell általánosítani az ilyen adatbázisok minden egyes komponensét, valamint azt, hogyan lehet az általánosított adatokat felhasználni többdimenziós analízis és adatbányászat céljából.

9.1.1. | Strukturált adatok általánosítása

Az objektumrelációs és az objektumorientált adatbázisok fontos tulajdonsága, hogy **komplex struktúraértékű adatokat** (halmazértékkel, valamint listaértékkel, beágyazott struktúrákkal rendelkező adatokat) rendezni, elérni és modellezni lehet.

Hogyan végezhető el az ilyen adatokon az általánosítás? Vessünk először egy pillantást a halmazértékű és a listaértékű attribútumok általánosítására.

Egy **halmazértékű attribútum** lehet homogén vagy heterogén típusú. Halmazértékű adatok tipikusan a következő módon általánosíthatók: (1) *a halmaz minden értékének a megfelelő magasabb szintű szempontba történő általánosításával* vagy (2) *a halmaz általános viselkedésének* (a halmaz elemszáma, típus, értékszóródás a halmazban, numerikus adatok súlyozott átlaga) *kinyerésével*. Sőt, általánosítás véghajtható *különböző általánosító operátorok segítségével az alternatív általánosítási utak felfedése céljából*. Ebben az esetben az általánosítás eredménye heterogén halmaz lesz.

9.1. példa | Tegyük fel, hogy egy ember *hobbija* egy halmazértékű attribútum, amely a következő értékek halmaza: {*tenisz, hoki, sakk, hegedű, nintendo_játékok*}. Ez a halmaz egy magas szintű szemponthalmazra általánosítható, például {*sport, zene, videojátékok*}, vagy egy számértékre – például az 5 (a hobbik száma a halmazban). Sőt, egy *számláló* érték összekapcsolható az általánosított értékkel, ami megmutatja, hogy hány elemet általánosítottunk ehhez az értékhez, például {*sport(3), zene(1), videojátékok(1)*}. ■

Egy halmazértékű attribútum halmazértékű vagy egyszerű értékű attribútumra általánosítható. Az egyszerű értékű attribútumot pedig – ha az értékek hálós vagy hierarchikus szerkezetet alkotnak, vagy az általánosítás különböző útvonalakat jár be – halmazértékűre általánosíthatunk. Az ilyen általánosított halmazértékű attribútumon végrehajtott további általánosításoknak követniük kell a halmaz minden értékének általánosítási útvonalát.

Egy **listaértékű vagy szekvenciaértékű tulajdonság** *általánosítása a halmazértékű attribútumok általánosításával azonos módon hajtható végre, azzal az eltéréssel, hogy tartani kell az elemek sorrendjét.* A lista minden értékét általánosítani kell a megfelelő magasabb szintű szempontra. Egy lista még az általános viselkedésének (lista hossza, listaelemek típusa, értékszóródás, numerikus adatok súlyozott átlaga vagy a lista lényeg-

telen elemeinek eldobása) figyelembevételével is általánosítható. Egy lista általánosítható listára, halmazra vagy egyszerű értékre.

9.2. példa | Nézzük egy személy képzettségi adatainak a következő listáját (vagy szekvenciáját): *„((B.Sc. Elektromérnök, U.B.C., 1990, dec.), (M.Sc. Számítógépes mérnök, Maryland Egyetem, 1993, máj.), (Ph.D. Számítástudomány, UCLA, 1997, aug.))"*. Ez általánosítható az adathármasokban található kevésbé lényeges leírások (alattribútumok) eldobásával, mint a *„((B.Sc., U.B.C., 1990),...)"* és/vagy csak a listában a legfontosabb sor meghagyásával, mint amilyen a *„(Ph.D. Számítástudomány, UCLA, 1997)"*. ■

Egy komplex struktúraértékű attribútum tartalmazhat halmazokat, sorokat, listákat, fákat, rekordokat és így tovább, illetve ezek kombinációit, ahol egy struktúra bármely szinten be lehet ágyazva egy másikba. A struktúraértékű attribútum általában többféleképpen általánosítható: (1) mindegyik attribútumát általánosíthatjuk, megtartva a struktúra szerkezetét; (2) a struktúrát kifejtjük és azután a kifejtett struktúrát általánosítjuk; (3) az alacsony szintű struktúrákat a magasabb szintű szempontoknak vagy aggregációknak megfelelően összegezzük; végül (4) visszaadjuk a struktúra típusát vagy egy leírását..

9.1.2. | Az összesítés és approximáció a térbeli és a multimédia-adatok általánosításában

Az összesítést és az approximációt az általánosítás további fontos eszközeinek kell tekintenünk, amelyek különösen hasznosak a nagy értékhalmazokkal rendelkező attribútumok, komplex struktúrák, térbeli vagy multimédia-adatok általánosítása esetén.

Vegyük példaként a **térbeli adatokat**. Részletes, kimerítő geográfiai pontokat szeretnénk általánosítással régiókra klaszterezni: üzleti, lakó-, ipari negyedekre vagy hasznosítás szerinti mezőgazdasági területekre. Ilyenkor gyakran szükség van a geográfiai területcsoportok térműveletekkel (térbeli unió vagy térklaszterezés) történő összefésülésre. Az ilyen általánosítások esetén fontos technika az összesítés és az approximáció. A **térbeli összefésülés** esetén nemcsak az a fontos, hogy a hasonló típusú régiókat ugyanabba az általános osztályba összefésüljük, hanem hogy kiszámítsuk a teljes területet, az átlagos sűrűséget vagy egyéb aggregációs függvényt, mialatt figyelmen kívül hagyjuk az egyes, a vizsgálatunk szempontjából lényegtelen, különböző típusú szétszórt régiókat. Más téroperátorok, például a *térunió*, a *térátfedés* és a *térmetszet*, amelyeknél szükség lehet a kis, szétszórt területek nagy, klaszterelt területekké történő összefésülésére, szintén használhatják a téraggregációt és approximációt mint adatáltalánosító operátorokat.

9.3. példa | Tegyük fel, hogy különböző mezőgazdasági hasznosítás céljából (gabonatermesztés, zöldség- vagy gyümölcstermesztés) több földdarabunk van. Ezek a darabkák térbeli összefésüléssel egy nagy mezőgazdasági földdarabba *gyűjthetők*. Viszont az ilyen mezőgazdasági földdarab magába foglalhat országutakat, házakat, kisebb raktárakat és így tovább. Ha a földterület döntő része mezőgazdasági hasznosítású, akkor a más célú szétszórt területek figyelmen kívül hagyhatók, és az egész terület az *approximáció* módszere alapján mezőgazdasági területnek tekinthető. ■

A **multimédia-adatbázis** komplex szövegeket, grafikákat, képeket, videorészleteket, térképeket, hanganyagot, zenét, valamint az audiovizuális információ egyéb formáit tartalmazhatja. A multimédia-adatokat jellemzően különböző hosszúságú bájtsorozatként tárolják, és ezek az adatszegmensek hivatkozásokkal vannak összekapcsolva, vagy a könnyű elérés céljából többdimenziós indexelést kapnak.

Multimédia-adatok általánosítása az ilyen adatok lényeges tulajdonságainak és/vagy általános mintáinak felismerésével és kikeresésével végezhető el. Az ilyen információ kikeresésének sok különböző módja van. Egy *kép* esetében a méret, a szín, az alak, a szöveges elemek, az elhelyezkedés iránya, valamint a bennfoglalt objektumok vagy képterületek relatív helyzete és struktúrája vonható ki aggregáció és/vagy approximáció segítségével. Egy *zenei* részlet esetén a dallama az ismétlődően előforduló hozzávetőleges minták alapján, míg a stílusa a hangszín, a tempó vagy a főbb hangszerek alapján értékelhetők. Egy *publikált cikknél* annak tartalma vagy általános felépítése (tartalomjegyzék, téma, gyakran előforduló fogalmak kivonata stb.) szolgálhatnak az általánosítás alapjául.

Általában véve, a téradatok és a multimédia-adatok általánosítása – amelynek segítségével az adatokban implicit módon rejtőző érdekes ismereteket nyerhetünk ki – igazi kihívás. Ahhoz, hogy kielégítő eredményeket érjünk el, a térinformatikai és multimédia-adatbázisoknál kifejlesztett technológiákat – például téradatok elérése, analízismódszerek, tartalom alapú képvisszakeresés, többdimenziós indexelési módszerek – integrálni kell az adatáltalánosítással és az adatbányászati technológiákkal. Az ilyen adatok adatbányászati technikáit a következő alfejezetek tárgyalják.

9.1.3. | Az objektumazonosítók és az osztály–alosztály hierarchiák általánosítása

Hogyan általánosíthatók az objektumazonosítók, ha az a szerepük, hogy egyértelműen azonosítsák az objektumokat? Első ránézésre lehetetlennek tűnik egy objektumazonosító általánosítása. Ha strukturálisan át is szervezzük az adatot, az azonosító változatlan marad. Ellenben, mivel az objektumorientált adatbázisokban az objektumok osztályokba, az osztályok pedig osztály–alosztály hierarchiába szervezettek, egy objektum általánosítása végrehajtható a neki megfelelő hierarchiahivatkozás alapján, a következőképpen: először az objektumazonosítót ahhoz a *legalacsonyabb szintű alosztályhoz* általánosítjuk, amelyhez az objektum tartozik. Ennek az alosztálynak az azonosítója sorban általánosítható a magasabb szintű osztály–alosztály azonosítóhoz, így felfelé haladva az osztály–alosztály hierarchián. Hasonló módon, egy osztály vagy egy alosztály általánosítható a felette lévő szülő(k)höz, felfelé haladva a megfelelő osztály–alosztály hierarchián.

Általánosíthatók-e az öröklött objektumtulajdonságok? Mivel az objektumorientált adatbázisok osztály–alosztály hierarchiába szervezettek, egyes attribútumok vagy metódusok nem magában az objektumban vannak explicit módon meghatározva, hanem az objektum magasabb szintű objektumaitól öröklődnek. Egyes objektumorientált adatbázisrendszerek megengedik a **többszörös öröklődést**, ahol – amennyiben az osztály–alosztály „hierarchia" háló formájú – a tulajdonságok több mint egy szülőosztálytól öröklődhetnek. Az objektumorientált adatbázisban egy objektum öröklött tulajdonságai lekérdező eljárással megkaphatók. Az adatáltalánosítás szempontjából nem szükséges különbséget

tenni az osztályban tárolt és a szülőosztálytól örökölt adatok között. Ha a releváns adatokat lekérdező eljárással gyűjtötték ki, az adatbányászati eljárás az öröklött adatokat az objektumosztályban tárolt adatokkal azonos módon kezeli, és az általánosítás is ennek megfelelően lesz végrehajtva.

Az objektumorientált adatbázisok fontos részei a *metódusok*. Az objektumok viselkedéséről sok adat vonható ki a metódusok használatával. Mivel a metódust számítási eljárás/függvény vagy dedukciós szabálykészlet határozza meg, lehetetlen magukon a metódusokon általánosítást végezni. Viszont a metódus *működése eredményeként kapott adatokon* már elvégezhető az általánosítás. Mivel a metódus meghívásával feladat szempontjából releváns adatok keletkeznek, ezeket az adatokat lehet általánosítani.

9.1.4. | Az osztályszerkezet-hierarchiák általánosítása

Egy objektum attribútuma más objektumokból állhat vagy más objektum írhatja le azt. Ezeknek az objektumoknak az attribútumai további objektumokból állhatnak vagy további objektumok írhatják le azokat, így ezek **osztályszerkezet-hierarchiát** alkotnak. Az osztályszerkezet-hierarchia általánosítása úgy tekinthető, mint egymásba ágyazott strukturált adatok általánosítása (amely lehet akár végtelen mélységű, ha az egymásba ágyazás rekurzív).

Elvileg az összetett objektumra hivatkozás egy hosszú, a megfelelő osztályszerkezet-hierarchián végighúzódó hivatkozási láncon vonulhat végig. Az esetek többségében viszont minél hosszabb a hivatkozási lánc, annál gyengébb a szemantikus kapcsolat az eredeti objektum és a hivatkozott összetett objektum között. Például, a *diák* objektumosztály *tulajdonában_lévő_jármű* attribútuma hivatkozhat egy másik objektumosztályra: *autó*ra, amelynek lehet egy *autó_kereskedő* attribútuma, amely hivatkozhat a kereskedő *főnök*ére és *gyerek*ére. Nyilvánvalóan nem valószínű, hogy bármilyen érdekes kapcsolat létezik a diák és az autókereskedő főnöke, valamint a gyereke között. Következésképpen egy objektumosztály általánosítását a leíró attribútumértékeken és metódusokon a közeli relációban lévő komponensekre korlátozott hivatkozásokon kell végrehajtani, az osztályszerkezet-hierarchián közelre mutató hivatkozások mentén. Ez azt jelenti, hogy az érdeklődésre számot tartó tudás feltárásához az általánosítást az osztályszerkezet-hierarchia azon objektumain kell elvégezni, amelyek *szemantikailag közeli kapcsolatban* állnak egymással, és nem azokon, amelyeknek csak távoli és elég gyenge szemantikai kapcsolatuk van a vizsgálat tárgyát képező osztállyal vagy osztályokkal.

9.1.5. | Objektumkockák létrehozása és bányászata

Egy objektum-adatbázis estén az adatáltalánosítást és a többdimenziós analízist nem az egyes objektumokra, hanem az objektumok osztályaira alkalmazzák. Mivel egy objektumosztályon belül az objektumok egy csoportja sok attribútumot és metódust megoszt egymás között, és minden egyes attribútum és metódus általánosítása általánosító operátorok sorozatát hívhatja elő, a fő kérdés az, hogy hogyan lehet az osztály(ok) különböző attribútumain és metódusain működő általánosító processzek közötti együttműködést létrehozni.

Hogy lehet nagy objektumhalmazon végrehajtani az osztály alapú általánosítást? Az 5. fejezetben bemutatott, a relációs adatbázisok jellemzőinek kibányászására kifejlesztett osztály alapú általánosítás, az *objektumorientált indukciós módszer* kiterjeszthető az objektum-adatbázisokon adatjellemzők kibányászására. Az általánosítás alapú adatbányászat folyamata úgy tekinthető, mint különböző attribútumokon az osztály alapú általánosítás operátoraiból álló sorozat végrehajtása. Az általánosítás addig folytatható, amíg az eredmény osztály kis számú általánosított objektumot tartalmaz, amelyek tömör, magasabb szintű fogalmakkal megadott szabályként összegezhetők. A hatékony megvalósítás céljából egy komplex objektumosztály többdimenziós attribútumainak általánosítása az egyes attribútumok (vagy dimenziók) vizsgálatával hajtható végre, ahol minden attribútumot egyszerű adatértékre általánosítunk. Így egy többdimenziós adatkocka jön létre, amit objektumkockának nevezünk. Amint létrejött egy objektumkocka, – a relációs adatkockákhoz hasonló módon – többdimenziós analízis és adatbányászat hajtható végre rajta.

Vegyük észre, hogy felhasználói szempontból nem mindig kívánatos egy értékhalmazt egyszerű adatértékre általánosítani. Nézzük meg a *kulcsszó* attribútumot, amely egy könyvet leíró kulcsszóhalmazt tartalmazhat. Nincs sok értelme ezt a kulcsszóhalmazt egy egyszerű értékre általánosítani. Ebben az értelemben nehéz *kulcsszó* dimenziót tartalmazó objektumkockát létrehozni. A következő alfejezetben, amelyben a téradatkocka létrehozását tárgyaljuk, bemutatunk ebben az irányban elért bizonyos eredményeket. Kihívó téma marad viszont a halmazértékű adatok objektumkocka-szerkezeteken végzett hatékony kezelésének és az objektum alapú adatbányászat technikáinak kutatása.

9.1.6. | Tervadatbázisok általánosítás alapú bányászata az „oszd meg és uralkodj" módszerrel

Ahhoz, hogy bemutassuk, milyen fontos szerepet játszik az általánosítás a komplex adatbázisokon történő adatbányászatban, megvizsgáljuk az „oszd meg és uralkodj" módszerrel történő „sikeres akciók szignifikáns mintái" bányászatát egy tervadatbázison.

Egy **ütemterv** *akciók* változó szekvenciáját tartalmazza. Egy **tervadatbázis**, vagy egyszerűen egy **tervbázis** egy nagy tervgyűjtemény. A **tervbányászat** olyan feladat, melynek során a tervbázisból kibányásszuk a fontos mintákat vagy tudást. Használható például üzletemberek utazási mintáinak repülési adatbázisból történő felderítésére vagy az autójavítási tevékenység szekvenciái alapján szignifikáns minták feltárására. A **tervbányászat** különbözik a szekvenciális minta bányászatától, ahol a gyakran előforduló, nagyszámú szekvenciák nagyon részletesen kerülnek kibányászásra. A tervbányászat esetén ehelyett a fontos vagy szignifikáns *általánosított* (szekvenciális) minták kibányászása történik a tervbázisból.

Vizsgáljuk meg a tervbányászat folyamatát egy légi utazási példa felhasználásával.

9.4. példa | Légi utazási tervbázis. A 9.1. táblázatban látható légi utazási tervbázis tárolja az ügyfél repülési szekvenciáit, ahol minden egyes rekord a szekvenciális adatbázisban egy *akciónak* – tevékenységnek – felel meg, és az azonos tervsorszámmal (terv#) rendelkező rekordok *szekvenciáját* egy akciósorozatból álló tervnek tekintjük. Az *indulás* és az

érkezés oszlopok meghatározzák az indulási és az érkezési repülőterek kódját. A 9.2. táblázat tartalmazza a repülőterek adatait.

9.1. táblázat | Utazási terv adatbázis: utazási tervbázis

terv#	akció#	indulás	indulási_idő	érkezés	érkezési_idő	társaság	...
1	1	ALB	800	JFK	900	TWA	...
1	2	JFK	1000	ORD	1230	UA	...
1	3	ORD	1300	LAX	1600	UA	...
1	4	LAX	1710	SAN	1800	DAL	...
2	1	SPI	900	ORD	950	AA	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

9.2. táblázat | Repülőterek információs táblája

repülőtér_kód	város	állam	megye	repülőtér_méret	...
ORD	Chicago	Illinois	Midwest	100 000	...
SPI	Springfield	Illinois	Midwest	10 000	...
LAX	Los Angeles	California	Pacific	80 000	...
ALB	Albany	New York	Atlantic	20 000	...
⋮	⋮	⋮	⋮	⋮	⋮

Sok mintát lehet kibányászni egy a 9.1. táblázathoz hasonló tervbázisból. Például kideríthetjük, hogy az USA keleti parti városaiból a középnyugati városokba tartó repülőutak nagy része megáll Chicago ORD repülőterén, ami annak tudható be, hogy több nagy repülőtársaság az ORD repülőteret csomópontként használja. Látható, hogy a csomópontként szolgáló repülőterek (például LAX – Los Angeles, ORD – Chicago, JFK – New York) könnyen kinyerhetők a 9.2. táblázat *repülőtér_méret* oszlopa alapján. Viszont egy utazási adatbázisban több száz csomópont lehet. Megkülönböztetés nélküli adatbányászat nagyszámú, alapvető segítséget nem nyújtó „szabályt" eredményezhet anélkül, hogy a teljes kép tisztábbá válna.

Tehát, hogyan kell hozzáfogni a tervbázis adatbányászatához? Amit meg szeretnénk találni, az kis számú általános (szekvenciális) minta, amely a tervek alapvető részét lefedi, és ezután az ilyen kibányászott szekvenciák alapján megoszthatjuk kereső erőfeszítéseinket. Az ilyen minta bányászatának kulcsa a tervek egy lényegesen magasabb szintre történő általánosítása. Egy többdimenziós adatbázismodell – amilyen a 9.1. ábrán látható légi utazási tervbázis – felhasználható egy ilyen terv általánosításának támogatására. Mivel a tömör tervek kialakításához az alacsony szintű információ soha nem osztozik elegendő közös vonásban, a következőket kell tennünk: (1) a tervbázist a többdimenziós modellt használva különböző irányokban általánosítani; (2) figyelni, hogy az általánosított tervek mikor tartalmaznak közös, alapvetően érdekes, szekvenciális mintákat; és (3) kinyerni magas szintű tömör terveket.

Vizsgáljuk meg ezt a tervbázist! Az ugyanazzal a tervsorszámmal rendelkező sorokat kombinálva az akciószekvenciák (repülőtérkódokkal leírva) a következőképpen alakulnak:

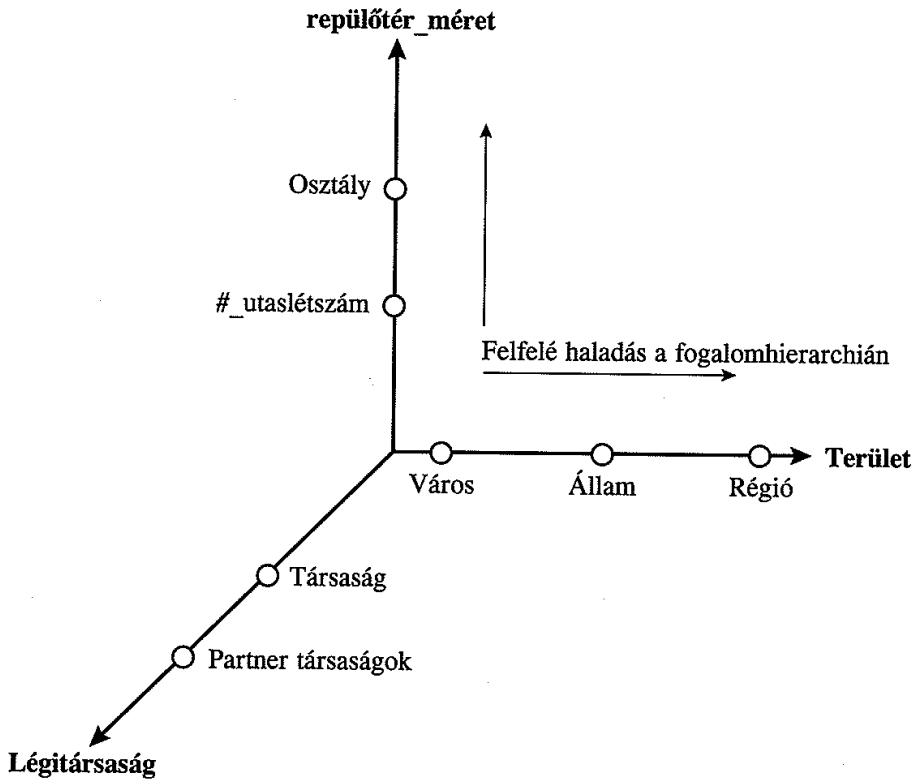
ALB – JFK – ORD – LAX – SAN
SPI – ORD – JFK – SYR
...

Ezek a szekvenciák teljesen különbözők lehetnek, ugyanakkor többdimenziósra általánosíthatók. Ha a *repülőtér_méret* dimenzió alapján általánosítjuk, akkor néhány érdekes szekvenciális mintát – például *S-L-L-S*, ahol *L* nagyméretű repülőteret (vagyis csomópontot), *S* kis regionális repülőteret jelent – figyelhetünk meg, ahogy ezt a 9.3. táblázatban láthatjuk.

9.3. táblázat | Tervbázis többdimenziós általánosítása

terv#	*terület-szekvencia*	*méret-szekvencia*	*állam-szekvencia*	*megye-szekvencia*	...
1	ALB-JFK-ORD-LAX-SAN	*S-L-L-L-S*	N-N-I-C-C	E-E-M-P-P	...
2	SPI-ORD-JFK-SYR	*S-L-L-S*	I-I-N-N	M-M-E-E	...
⋮	⋮	⋮	⋮	⋮	⋮

Nagyszámú légi utazási terv általánosítása néhány elég általános, de erősen szabályos mintához vezet. Gyakran ez a helyzet, ha az általánosított szekvenciákra a **merge** (összefésülés) és az **optional** (opcionális) operátorokat alkalmazzuk. Az első operátor a tranzitív + jel használata segítségével egy szimbólumba fésüli össze (vagy egyesíti) az egymás utáni azonos szimbólumokat, hogy egy azonos típusú tevékenységekből álló szekvenciát egy szimbólummal lehessen reprezentálni. A második operátor a [] jelölést használja annak jelzésé-



9.1. ábra | Az adatbázis többdimenziós nézete

re, hogy a szögletes zárójelekben álló objektum vagy akció opcionális. A 9.4. táblázat mutatja a **merge** operátor 9.3. táblázat terveire történt alkalmazásának eredményét.

9.4. táblázat | A tervek egymásutáni, azonos tevékenységeinek összefésülése

terv#	méret-szekvencia	állam-szekvencia	megye-szekvencia	...
1	S-L^+-S	N^+-I-C^+	E^+-M-P^+	...
2	S-L^+-S	I^+-N^+	M^+-E^+	...
⋮	⋮	⋮	⋮	⋮

Hasonló tevékenységek összefésülésese és összevonása segítségével a 9.1. képletnek megfelelő általánosított szekvenciális mintákat kaphatunk:

$$[s] - I^+ - [s] \quad [98{,}5\%] \tag{9.1}$$

A minta azt mutatja, hogy az utazási tervek 98,5%-a az $[S] - L^+ - [S]$ mintájú, ahol $[S]$ mutatja, hogy az S tevékenység opcionális és L^+ az L^+egyszeri vagy többszöri ismétlését jelzi. Más szóval, az utazási minta először esetlegesen egy kis repülőtérről indul, egy vagy néhány nagy repülőtéren keresztül egy nagy (vagy esetleg egy kis) repülőtéren ér véget.

Miután elegendő megalapozottsággal találtunk egy mintát, felhasználhatjuk a tervbázis felosztására. Ezután a közös jellemzők kikeresése céljából mindegyik részen végezhetünk adatbányászatot. Például egy leválasztott tervbázisból a következőt találhatjuk:

$$járat(x, y) \wedge repülőtér\_méret(x, S) \wedge repülőtér\_méret(y, L) \Rightarrow régió(x) = régió(y) \; [75\%] \tag{9.2}$$

Ez azt jelenti, hogy egy x kis repülőtérről egy y nagy repülőtérre tartó közvetlen repülőútnál 75% annak valószínűsége, hogy x és y repülőterek ugyanahhoz a régióhoz tartoznak. ■

A példa az *„oszd meg és uralkodj" stratégiát* mutatja be, amely a tervbázis többdimenziós általánosítása segítségével először az érdekes, magas szintű, tömör tervszekvenciákat találja meg, ezután a kibányászott minták alapján felosztja a tervbázist, hogy felfedje a rész-tervbázisok megfelelő jellemzőit. Ez a bányászó közelítés sok más alkalmazásnál is használható. Például a webnapló adatbányászatnál, még mielőtt leásnánk a részletes, mélyebb rangú mintákba, az általános webelérési minták vizsgálatával meghatározhatjuk a népszerű, keresett webportálokat és webutakat.

A tervbányászat technikáit különböző szempontok szerint fejleszthetjük tovább. Például a *minimális megalapozottság*, amely hasonló a társítási szabállyal történő bányászatban lévő küszöbhöz, felhasználható az általánosítás szintjének meghatározására, és arra, hogy megbizonyosodjunk arról, hogy a minta az esetek jelentős hányadát lefedi. A tervbányászatban további operátorok találhatók, mint például *kisebb_mint*. Más változatokhoz tartoznak olyanok, amelyek a rész-szekvenciákból kapcsolatokat fednek fel, vagy amelyek többdimenziós attribútumokat tartalmazó szekvenciamintákat bányásznak ki, például olyan mintát, amely mind a repülőtér méretét, mind az országot tartalmazza. Ilyen kombinált dimenziók bányászatához a kombinált szekvenciaminták vizsgálata előtt mindegyik dimenzió magas szintre történő általánosítására van szükség.

9.2. | Téradatbázisok bányászata

Téradatbázis nagy mennyiségű térre vonatkozó adatot tárol, például térképeket, előfeldolgozott távérzékelő adatokat vagy orvosi képadatokat, valamint VLSI lapkaelrendezési adatokat. A téradatbázisok sok tulajdonságban különböznek a relációs adatbázisoktól. Topológiai és/vagy távolsági információt hordoznak, az adatok szervezéséhez bonyolult többdimenziós térindex struktúrát használnak, az adatokat térelérési módszerekkel lehet elérni, és gyakran szükség van a tér értelmezésére, geometriai számításra, valamint a térbeli ismereteket megjelenítő technikákra.

Téradatbányászat célja olyan tudás, térrelációk vagy más érdekes minták kikeresése, amelyeket a téradatbázis explicit módon nem tárol. Ilyen adatbányászat számára szükség van az adatbányászat és a téradatbázis-technológiák integrálására. Ez arra használható, hogy megértsük a téradatokat, feltárjuk a térrelációkat, a téradatok és a nem téradatok közötti relációkat, létrehozzunk tértudásbázisokat, átszervezzük a téradatbázisokat, valamint optimalizáljuk a térlekérdezéseket. Széles körű alkalmazásra lehet számítani a térképészeti információs rendszerek, a geo-piackutatás, a távérzékelés, a képadatbázis-feltárás, az orvosi képfeldolgozás, a navigáció, a közlekedésirányítás, a környezetvédelmi vizsgálatok és sok egyéb területen, ahol téradatokat használnak. Döntő kihívás a téradatbányászattal szemben *hatékony* téradatbányászati technikák kidolgozása, hiszen a téradatok mennyisége hatalmas, szerkezetük és elérési módjuk pedig igen bonyolult.

Mi a helyzet a téradatbányászatnál a statisztikai módszerek használatával? A téradatok analízisének népszerű módszere volt a statisztikai analízis. A módszer jól kezeli a numerikus adatokat, és általában a térbeli jelenségek reális modelljét kínálja. Viszont a módszer tipikusan statisztikai függetlenséget tételez fel a térben elosztott adatok között, a valóságban azonban a térbeli objektumok gyakran kölcsönös kapcsolatban állnak egymással. Sőt mi több, a legtöbb statisztikai modellezést csak a tárgykörben kellő ismerettel és statisztikai szaktudással rendelkező szakember tudja elvégezni. Továbbá a statisztikai módszerek rosszul kezelik a szimbolikus értékeket, a nem teljes vagy nem meggyőző adatokat, és nagy adatbázisok esetén a számítási költség is magas. A téradatbányászat módot ad a tradicionális térelemző módszerek kibővítésére – a hangsúlyt a hatékonyságra, a skálázhatóságra, az adatbázisokkal történő együttműködésre, a felhasználói interakció javítására, új tudás feltárására helyezve.

9.2.1. | Téradatkocka létrehozása és a tér-OLAP

Létre tudunk-e hozni téradattárházat? Igen, ugyanúgy, mint a relációs adatoknál, a téradatok integrálásával létrehozható olyan adattárház, amely lehetővé teszi a téradatbányászatot. Egy téradattárház a tér- és nem téradatoknak *tárgyorientált, integrált, időfüggő* és *tartós,* a téradatbányászatot és a téradatokkal kapcsolatos döntéshozási folyamatokat segítő gyűjteménye.

Nézzük a következő példát.

9.5. példa | Körülbelül 3000 meteorológiai mérőszonda található Brit Kolumbiában (BK), mindegyik egy kijelölt, kis terület napi hőmérsékletét, valamint a csapadékot méri, és a megyei meteorológiai állomásra küldi az adatokat. A felhasználó egy tér-OLAP-ot támogató téradattárház segítségével meg tudja nézni a térképen a meteorológiai mintákat havi, regionális, különböző hőmérséklet-csapadék kombinációs bontásban, bármely dimenzió mentén le tud fúrni vagy felgörgetést tud végezni, hogy megtalálja a szükséges mintát, például „nedves és forró régiók a Fraser-völgyben 1999 nyarán". ■

A téradattárház létrehozását és használatát illetően van néhány probléma. Az első a heterogén forrásokból és rendszerekből származó téradatok integrálása. A téradatok általában különböző ipari cégeknél, kormányhivataloknál, sokféle adatformátumban találhatók. Az adatformátum nemcsak struktúraspecifikus (raszter, illetve vektor alapú téradat, objektumorientált, illetve relációs modellek, eltérő téradattárolási módok és indexelési struktúrák stb.), hanem gyártóspecifikus is (ESRI, MapInfo, Intergraph stb.). Sok munkát fektettek a heterogén téradatok integrációjának és cseréjének kidolgozásába. Ez kövezte ki az utat a téradatok integrációjához és a téradattárház létrehozásához.

A második kihívás a gyors és rugalmas on-line analitikus feldolgozás megvalósítása a téradattárházakban. A 2. fejezetben bemutatott csillagséma modell jó választás a téradattárházak modellezésére, mivel a tömör és szervezett adattárház-struktúra elősegíti az OLAP-műveleteket. A téradattárházakban viszont mind a dimenziók, mind a mértékek térbeli komponenseket tartalmazhatnak.

Egy téradatkockában három *dimenziótípus* található:

- A **nem térbeli dimenzió** csak nem térbeli adatot tartalmaz. A 9.5. példában a hőmérséklet és a csapadék nem térbeli dimenziókat lehet létrehozni, mivel mindegyik nem térbeli adatot tartalmaz, amelyek általánosítása nem térbeli adatokat eredményez (mint a *„forróság"* a *hőmérséklet,* a *„nedvesség"* a *csapadék* esetén).
- Egy **térbeli–nem térbeli dimenzió** primitív szintjén téradatok vannak, amelyeknek egy bizonyos, magas szinttől indított általánosítása nem térbeli adatokat eredményez. Például a *város* térbeli dimenzió földrajzi adatokat helyez az USA térképére. Tegyük fel, hogy mondjuk Seattle dimenzió térbeli reprezentációja, általánosítása az *óceáni_északnyugati* szöveg. Az *óceáni_északnyugati* térbeli fogalom, reprezentációja viszont nem térbeli (mivel példánkban szöveg). Tehát nem térbeli dimenzióként szerepel.
- Egy **térbeli–térbeli dimenzió** olyan dimenzió, amelynek primitív szintje és az összes magasabb szintre általánosított adata téradat. Például *az izotermikus_régió* téradatot tartalmaz, valamint az összes többi általánosított leszármazottja, például *0–5_fok* (Celsius), *5–10_fok*-ot lefedő területek és így tovább.

A téradatkockában két *mértéktípust* különböztetünk meg.

- **Numerikus mérték** csak numerikus adatokat tartalmaz. Például, a téradattárházban egy mérték lehetne a régió *havi_bevétele*, így felgörgetés segítségével kiszámíthatjuk a megyére, az évre jutó teljes bevételt, és így tovább. Numerikus mértékeket később

disztributív, algebrai és *holisztikus* osztályokba lehet sorolni, ahogy azt a 2. fejezetben tárgyaltuk.

- **Térbeli mérték** egy, a térobjektumokra irányuló mutatók gyűjteményét tartalmazza. Például, a 9.5. példa téradatkocka általánosításakor (vagy felgörgetésekor), az izotermikus és az azonos csapadékszintű régiókat ugyanabba a cellába csoportosítjuk, és az így kialakított mérték ezekre a régiókra irányuló mutatók gyűjteményét tartalmazza.

Egy nem térbeli adatkocka csak nem térbeli dimenziókat és numerikus mértékeket tartalmaz. Ha egy téradatkocka tartalmaz térbeli dimenziókat, de nem tartalmaz nem térbeli mértékeket, a rajta értelmezett OLAP-műveletek, például lefúrás vagy az adatkocka elforgatása (pivoting), végrehajtása hasonló módon történhet, mint a nem térbeli adatkockáknál.

Mi a helyzet akkor, ha térbeli mértékeket kell használnom egy téradatkockában? Ez a megjegyzés, ahogy a következő példán látni fogjuk, a hatékony implementálással kapcsolatban néhány izgalmas kérdést vet fel.

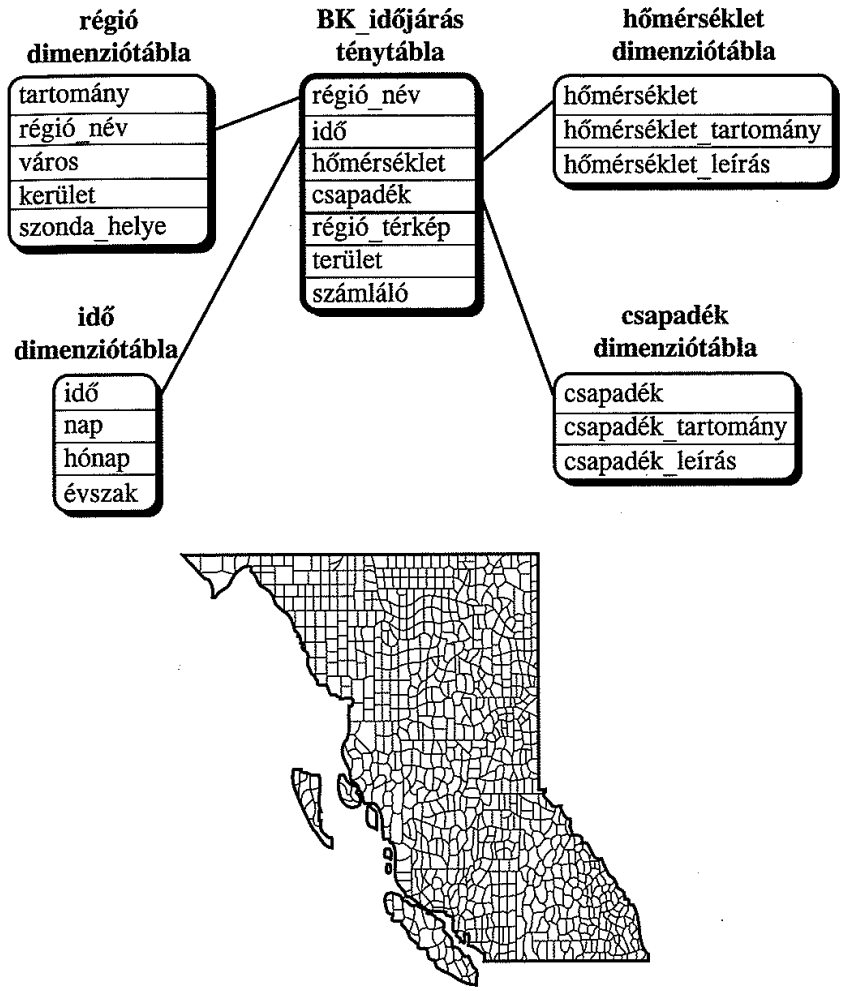
9.6. példa | A 9.5. példán bemutatott *BK_időjárás* adattárház csillagsémája a 9.2. ábrán látható, négy dimenzióval: *régió, hőmérséklet, idő* és *csapadék*, valamint három mértékkel: *régió_térkép, térség* és *számláló*. A dimenziónkénti fogalomhierarchiát vagy a felhasználók, illetve szakemberek tudják előállítani, vagy klaszterelemzés segítségével automatikusan generálható. A 9.3. ábra a *BK_időjárás* adattárház minden egyes dimenziójához bemutatja a hozzá tartozó hierarchiát.

A három mérték közül a *terület* és a *számláló – numerikus* mértékek, ezeket a nem térbeli adatkockák mértékeihez hasonló módon lehet kiszámítani, a *régió_térkép* pedig térbeli mérték, amely a megfelelő régiókra mutató térbeli pointerhalmazt tartalmazza. Mivel különböző OLAP-műveletek a *régió_térkép*-ről különböző térbeli objektumgyűjteséket eredményeznek, a fő feladat rugalmasan és dinamikusan kiszámolni egy nagyszámú régió összefésülését. Például a BK időjárástérkép adatain végrehajtott két különböző aggregáció két különböző általánosított régiótérképet eredményezhet, – ahogy ezt a 9.4. ábrán láthatjuk –, ahol mindegyik régió a 9.2. ábrán lévő nagyszámú kis (szonda) régió összefésülése. ■

Ki tudjuk-e számolni előre az összes lehetséges térbeli összefésülést és az eredményt tárolhatjuk-e egy téradatkocka részkockáinak megfelelő cellákban? A válasz: valószínűleg nem. Ellentétben a numerikus mértékekkel, ahol minden aggregált értéknek néhány bájtnyi területre, a BK összefésült régiótérképének néhány megabájtnyi tárterületre van szüksége. Ily módon szembenézünk az on-line számítás költsége és a kiszámított mértékek tárolásának többletköltsége közötti kiegyensúlyozás dilemmájával: a téraggregáció alapvető, dinamikus igény szerinti, menet közben történő számítási költsége miatt szükség van előfeldolgozásra, de az aggregált térértékek igen nagy tárolási többletköltsége ebben elbátortalanít minket.

A téradatkocka létrehozásakor a térbeli mértékeinek kiszámításával kapcsolatban legalább három válasz lehetséges.

- *A megfelelő térobjektum-mutatók összegyűjtése és tárolása anélkül, hogy a téradatkockán a térbeli mértékek előfeldolgozását elvégeznénk.* Ez úgy hajtható végre, hogy az adat-



9.2. ábra | *BK_időjárás* téradattárház csillagsémája és a megfelelő BK időjárási szondatérképe

régió_név dimenzió:
szonda_helye < *kerület* < *város* < *régió* < *tartomány*

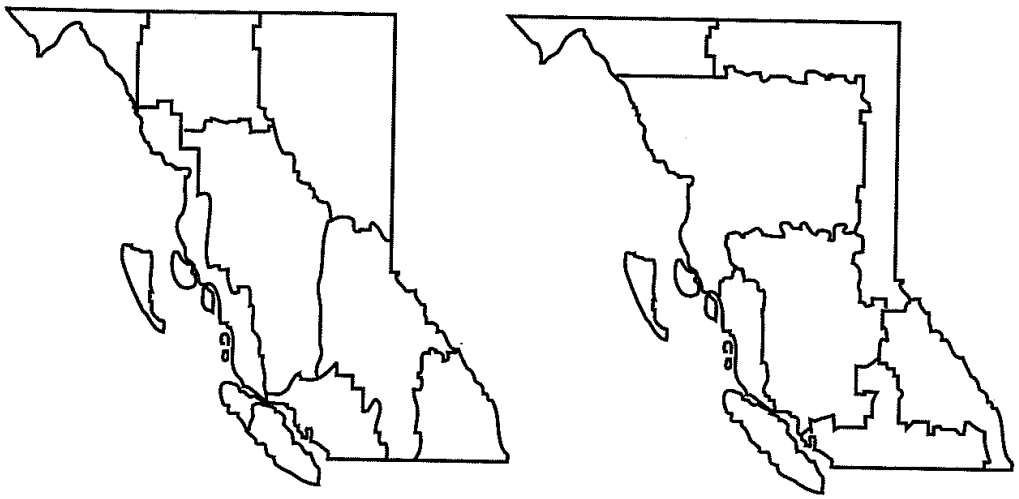
hőmérséklet dimenzió:
(*hideg, enyhe, meleg*) ⊂ **minden**(*hőmérséklet*)
(–20_alatt, –20 ... –11, –10 ... 0) ⊂ *hideg*
(0 ... 10, 11 ... 15, 16 ... 20) ⊂ *enyhe*
(20 ... 25, 26 ... 30, 30 ... 35, *35_felett*) ⊂ *meleg*

idő dimenzió:
óra < *nap* < *hónap* < *évszak*

csapadék dimenzió:
(*száraz, derült, nedves*) ⊂ **minden**(*csapadék*)
(0 ... 0,05, 0,06 ... 0,2) ⊂ *száraz*
(0,2 ... 0,5, 0,6 ... 1,0, 1,1 ... 105) ⊂ *derült*
(1,5 ... 2,0, 2,1 ... 3,0, 3,1 ... 5,0, *5,0_felett*) ⊂ *nedves*

9.3. ábra | A *BK_időjárás* téradattárház dimenzióinak hierarchiái

kocka megfelelő cellájában tárolunk egy pointert, amely a térbeli objektummutatókra mutat, a megfelelő térbeli objektumokon a térösszefésülést (vagy más számítást) csak akkor hívjuk meg és hajtjuk végre, amikor szükséges. Ez a módszer jó választásnak bizonyul akkor, ha csak térmegjelenítés szükséges (nem kell tényleges térösszefésülést végrehajtani), vagy ha bármely régiópointer-halmazban kevés régiót kell összefésülni (az on-line összefésülés nem nagyon drága), vagy ha az on-line térösszefésülési számítás gyors (korábban néhány hatékony térösszefésülés került kifejlesztésre a gyors tér-



9.4. ábra | Különböző aggregációs operációkkal kapott általánosított régiók

OLAP számára). Mivel az OLAP-eredményeket gyakran on-line téranalízisre és adatbányászatra használják, ezért az ilyen analízis felgyorsítására ajánlott a térbeli kapcsolatban lévő régiókat előre feldolgozni.

- *A térbeli mértékek nyers approximációjának előfeldolgzása és téradatkockában tárolása.* Ez a választás alkalmas a térösszefésülés eredményének durva megtekintésére, nyers értékelésére, azzal a feltevéssel élve, hogy kis tárigény merül fel. Például a két ponttal ábrázolt *minimális határoló téglalap* (minimal bounding rectangle, MBR), egy összefésült régió durva értékelésének tekinthető. Az ilyen előfeldolgozási eredmény kicsi és gyorsan ábrázolható a felhasználó számára. Ha egyes konkrét cellák esetén nagyobb pontosság szükséges, az alkalmazás előkeresi az előfeldolgozás minőségi eredményeit (amennyiben azok rendelkezésre állnak), vagy szükség szerint kiszámítja azokat.
- *Egyes térbeli mértékek téradatkockában történő szelektív előfeldolgozása.* Ez szellemes megoldás lehet. A kérdés így vetődik fel: „A kocka mely részét kell kiválasztani a megvalósításhoz?" A kiválasztás részkocka szinten történhet, ami azt jelenti, hogy egy kiválasztott részkocka *összes* cellájának *összes* összefésülhető térbeli régiócsoportjára elvégezzük az előfeldolgozást, vagy – ha a részkocka nincs kiválasztva – egyre sem végezzük el. Mivel egy részkocka általában nagyszámú térbeli objektumot tartalmaz, az előfeldolgozásba és a tárolásba sok összefésülhető térbeli objektum kerülhet, amelyek közül egyeseket esetleg ritkán használnak fel. Következésképpen ajánlott egy finomabb felbontási szinten kiválasztást végezni: annak eldöntésére, hogy végre kell-e hajtani egy ilyen összefésülést, a részkockában minden összefésülhető térbeli objektumcsoportot meg kell vizsgálni. A döntés alapjául a használhatóságot (hozzáférési gyakoriság vagy prioritás), az összefésült területek megoszthatóságát, a tárhely és az on-line számítás kiegyensúlyozott teljes költségét kell figyelembe venni.

A téradatkockák és a tér-OLAP hatékony implementálásával az általánosítás alapú leíró téradatbányászat, mind a térbeli karakterizáció és diszkrimináció hatékonyan végrehajtható.

9.2.2. | Térbeli társításelemzés

Mi a helyzet a térbeli társítási szabályok bányászatával? A tranzakciós és a relációs adatbázisok társítási szabályainak bányászatához hasonlóan lehet a téradatbázisokban a *térbeli társítási szabályokat* bányászni. A **térbeli társítási szabály** egy $A \Rightarrow B[s\%, c\%]$ szerkezet, ahol A és B térbeli vagy nem térbeli predikátumok halmaza, $s\%$ a szabály megalapozottsága, a $c\%$ pedig a szabály megbízhatósága. Térbeli társítási szabály például a következő:

$$van\_egy(X, \text{“}iskola\text{”}) \wedge közel\_valamihez(X, \text{“}sport\_centrum\text{”}) \Rightarrow közel\_vmihez(X, \text{“}park\text{”}) \quad [0{,}5\%, 80\%]$$

A fenti szabály szerint a sportcentrumokhoz közel lévő iskoláknak a 80%-a valamely parkhoz szintén közel van; ez az összes iskolák 0,5%-a.

Egy térbeli társítási szabályban különféle térpredikátumok szerepelhetnek. Például távolsági információt tartalmazhatnak (*közel_vmihez*, *távol_vmitől*), vagy topológiai relációkat (*metszet*, *átfedés*, *diszjunkció*), valamint térirányultságokat (*jobbra_vmitől*, *nyugatra_vmitől*).

Mivel a térbeli társításbányászatnál nagyszámú térbeli objektum közötti reláció kiértékelésére van szükség, az adatbányászat folyamata elég költséges lehet. A térbeli társítási analízishez egy érdekes adatbányászati optimalizálás, az ún. **fokozatos finomítási módszer** alkalmazható. A módszer először egy gyors algoritmus segítségével a nagy adathalmazokon durva adatbányászatot végez, majd az így megtisztított adathalmazban – sokkal költségesebb algoritmust használva – javítja az adatbányászat minőségét.

Azért, hogy biztosítsuk azt, hogy a megtisztított adathalmaz a későbbi fázisban történő minőségi adatbányászat alkalmazásához a lehetséges teljes válaszkészletet tartalmazza, a korábbi fázisban alkalmazott durva adatbányászati algoritmussal szemben támasztott fontos követelmény a **szuperszet lefedési tulajdonság**, ami azt jelenti, hogy az algoritmus megőrzi az összes potenciális választ. Más szóval, meg kell engednie a *hamis pozitív tesztet*, ami tartalmazhat a válaszhalmazhoz nem tartozó adathalmazokat is, de nem engedheti meg a *hamis negatív tesztet*, amely akár potenciális választ is kizárhat.

A *közel_vmihez* predikátummal kapcsolatos térbeli társítási adatbányászat számára először a következő módon kell kigyűjteni azokat a jelölteket, amelyek átjutnak a *minimális megalapozottsági küszöbön*:

- bizonyos durva térkiértékelő algoritmusok használatával, például minimális határoló téglalap használatával (amely csak két térbeli pontot ír le, szemben egy sokszög halmazzal), és
- a gyengített *g_közel_vmihez* térpredikátum kiértékelésével, amely általánosított *közel_vmihez* és szélesebb értelmű, a *közel_vmihez*, *érintkező* és a *metsző* predikátumokat fogja össze.

Ha két térbeli objektum egymáshoz közel helyezkedik el, a bennfoglaló minimális határoló téglalapoknak is – a *g_közel_vmihez* predikátumnak eleget téve – közel kell elhelyezkedniük egymáshoz. Viszont az ellenkezője nem mindig igaz: ha a bennfoglaló minimális határoló téglalapok egymáshoz közeliek, a két térbeli objektum közel is lehet egymáshoz,

meg nem is. Így a minimális határoló téglalapos tisztítás a közelség szempontjából hamis pozitív tesztelő eszköz: csak azt kell a költségesebb algoritmussal további vizsgálatnak alávetni, ami átjut a *durva* teszten. Az ilyen előfeldolgozás használata mellett csak a közelítő szinten gyakran előforduló mintákat kell részletesebb, finomabb, de jóval drágább számítással vizsgálni.

9.2.3. | Térbeli klaszterezési módszerek

A térbeli adatok klaszterezése többdimenziós, nagy adathalmazon értelmezett valamilyen távolsági mértéknek megfelelően meghatározza a klasztereket vagy sűrű népességű területeket. A térbeli klaszterezési módszereket a 8. fejezet alaposan megvizsgálta, mivel a példákban és az alkalmazásokban a klaszterelemzés általában téradat-klaszterezést tételez fel. Így a térbeli klaszterezés után érdeklődő olvasó forduljon a 8. fejezethez.

9.2.4. | Térbeli osztályozás és térbeli trendanalízis

Abból a célból, hogy meghatározott térbeli tulajdonságokkal összefüggésben (például körzet, országút vagy folyó *szomszédsága*) osztályozási sémákat nyerjünk ki, térbeli osztályozási analízisnek vetjük alá a térobjektumokat.

9.7. példa | Térosztályozás. Tegyük fel, hogy az átlagos családi jövedelem alapján osztályozni akarjuk a területeket *gazdag* és *szegény* osztályokra. Ez alatt meg szeretnénk határozni a lényeges, a térrel összefüggésben lévő, a területi osztályozást befolyásoló tényezőket. Sok térobjektumokkal kapcsolatos tulajdonság van, például egyetem jelenléte, városközi utak megléte, tóhoz vagy óceánhoz való közelség stb. Ezek a tulajdonságok felhasználhatók relevanciaanalízis céljára, valamint arra, hogy érdekes osztályozási sémákhoz jussunk. Ilyen osztályozási sémák döntési fákkal vagy szabályokkal ábrázolhatók, például úgy, ahogy ez a 7. fejezetben szerepel. ■

A **térbeli trendanalízis** más témával foglalkozik: egy térdimenzió mentén megjelenő változások és trendek felismerésével. A trendanalízis jellemzően az időbeli változásokat fedi fel, például az idősoradatokban történő időbeli mintaváltozásokat. A térbeli trendanalízis az időt térrel váltja fel, és a nem térbeli és a térbeli adatok térben történő változásának trendjét vizsgálja. Például vizsgálhatjuk a város központjától távolodva a gazdasági helyzet változását vagy a klíma és a vegetáció változásának trendjét az óceántól való távolság növekedésével. Ilyen analízisekhez gyakran alkalmaznak regressziós és korrelációs elemző módszereket a téradatstruktúrák és térbeli elérési módszerek segítségével.

Sok olyan alkalmazás van, ahol a minták *mind időben, mind térben* változnak. Például az országutak és a városi utak forgalma egyaránt idő- és térfüggő. Az időjárási minták szintén szoros kapcsolatban vannak az idővel és a térrel. Annak ellenére, hogy már néhány érdekes tanulmány készült a térbeli osztályozás és térbeli trendanalízis témakörében, a tér–idő adatbányászat kutatása még gyerekcipőben jár. Még több módszer és alkalmazás kutatására van szükség a térbeli osztályozás és térbeli trendanalízis területén, különösen az idővel összefüggésben.

9.2.5. | Raszteres adatbázisok bányászata

A téradatbázis-rendszerek általában vektoros adatokat kezelnek, amelyek pontokat, vonalakat, poligonokat (régiókat[1]) és ezek kompozícióit – hálózatokat vagy felosztásokat – tartalmaznak. Jellegzetes példák ilyen adatokra a térképek, a hálótervek, a proteinmolekula-láncok háromdimenziós ábrázolása. Viszont igen sok térre vonatkozó adat **digitális raszter (kép) formában** található, például szatellitképek, távérzékelőtől kapott adatok, valamint CT- (komputertomográf) adatok. Fontos a raszter- és képadatok bányászatának kutatása. A raszter- vagy képadatok adatbányászati módszereit a következő, multimédia-adatok adatbányászatával kapcsolatos alfejezet tárgyalja.

9.3. | Multimédia-adatbázisok bányászata

Mi a multimédia-adatbázis? Egy **multimédia-adatbázisrendszer** a *multimédia-objektumok* (audio-, kép-, video-, szekvencia, valamint hipertextadat, amely szöveget, szövegkijelölőket és hivatkozásokat tartalmaz) nagy gyűjteményét tárolja és kezeli. A multimédia-adatbázisrendszerek az audio/videoberendezések, CD-ROM-ok és az internet népszerűségének köszönhetően egyre elterjedtebbekké válnak. A jellegzetes multimédia-adatbázisrendszerekhez tartoznak a NASA EOS-t (Earth Observation System, földmegfigyelő rendszer), különböző fajta kép- és audio/video-, humángenetikai (genome) adatbázisok, valamint az internet-adatbázisok.

Ebben a részben a multimédia-adatbányászat keretén belül a képadatbányászatra fordítjuk figyelmünket. Ami a szekvenciális adatokat illeti, adatbányászatukat a 9.4. alfejezet vizsgálja, a bioinformatikai alkalmazások adatainak adatbányászatával pedig a 10. fejezet foglalkozik. A hipertextadatok adatbányászatát a Word Wide Web adatbányászatával foglalkozó 9.6. alfejezet vizsgálja. Itt a multimédia-adatbányászat módszereit mutatjuk be, beleértve a multimédia-adatokon történő hasonlóságkeresést, a többdimenziós analízist, az osztályozást és előrejelzést, valamint a multimédia-adatokban a kapcsolatok bányászatát.

9.3.1. | Hasonlóság keresése a multimédia-adatokban

Elvégezhető-e a multimédia-adatokban a hasonlóságok keresése mind az adatleírás, mind az adattartalom alapján? Igen. A multimédia-adatokban történő hasonlóságkereséshez a multimédia indexelés és visszakereső rendszerek két fő családját vesszük figyelembe. Ezek (1) a **leírás alapú visszakereső** rendszerek, amelyek indexeket hoznak létre és a képleírásokon – például kulcsszavak, feliratozások, méret, létrehozás ideje – alapuló elérést hajtanak végre; (2) a **tartalom alapú visszakereső** rendszerek, amelyek a képtartalom – például a színkép, alkat, forma, objektumok és hullámtranszformáció – alapján elősegítik a visszakeresést. A leírás alapú adatvisszakeresés – ha manuálisan végzik – munkaigényes. Ha automatizált, akkor viszont tipikusan gyenge a minősége; például a

1 | Poligonok által határolt régiókat. *(A fordító megjegyzése)*

kulcsszavak hozzárendelése a képekhez bonyolult és önkényes feladat lehet. Tartalom alapú adatvisszakeresés a képek indexelésére vizuális jellegzetességeket használ, és a jellegzetességi hasonlóság alapján történik az objektumok visszakeresése, ez sok alkalmazásnál javasolt módszer.

A tartalom alapú visszakereső rendszereknél gyakran kétfajta lekérdezés létezik: *képminta alapú lekérdezés* és *képjellegzetesség lekérdezése*. A **képminta alapú lekérdezés** megtalálja az összes olyan képet, amely hasonlít a megadott képmintához. Ez a keresés összehasonlítja a mintából kivont **jellegzetességvektort** (vagy **szignót**) a képekből előre kivont és a képadatbázisban indexként használt jellegzetességekkel. Az ilyen összehasonlítás alapján a mintához közeli képek válogatódnak ki. A **képjellegzetesség alapú lekérdezés** képjellegzetességeket (szín, szerkezet vagy forma) határoz meg vagy vázol fel, amiket az adatbázisban lévő képek jellegzetességvektoraival történő összehasonlítás céljából egy jellegzetességvektorrá transzformál. A tartalom alapú lekérdezést széles körben alkalmazzák, például az orvosi diagnosztikában, időjárás-előrejelzésnél, tv-műsor készítésénél, webes képkeresőknél és az elektronikus kereskedelemben. Egyes rendszerek, például a *QBIC (Query By Image Content, képtartalom szerinti lekérdezés)*, mind a képminta alapú lekérdezést, mind a képjellegzetesség specifikációs lekérdezést lehetővé teszik. Vannak még olyan rendszerek is, amelyekben lehetséges a tartalom alapú és a leírás alapú visszakeresés is.

A kép szignóját használó, a képadatbázisban történő hasonlóság alapú visszakeresésre többféle megközelítést javasoltak és kutattak:

- **Színkép alapú szignó** – Ebben a szemléletben egy kép szignója a kép színösszetételén alapuló színképeket tartalmazza, függetlenül annak skálázásától vagy irányultságától. Mivel ez a módszer nem tartalmaz semmilyen formai, elhelyezkedési vagy szerkezeti információt, két hasonló színkompozíciójú kép nagyon eltérő formával vagy alkattal rendelkezhet, és így lehet, hogy szemantikusan semmi kapcsolat nincs közöttük.
- **Többjellemzős szignó** – Ebben a szemléletben a kép szignója több jellegzetesség (színkép, forma, elhelyezkedés, szerkezet) kompozícióját tartalmazza. Sok esetben minden egyes jellegzetességre külön távolságfüggvényt lehet megadni, azután pedig ezek kombinálásával kapjuk az általános eredményt. A többdimenziós tartalom alapú keresés gyakran használ egy vagy több próbajellegzetességet, hogy megtaláljuk az ilyen (hasonló) jellegzetességeket tartalmazó képeket. Következésképpen ezt hasonló képek keresésére lehet használni.
- **Wavelet alapú szignó** – Ez a megközelítés a kép domináns wavelet-együtthatóit használja szignóként. A waveletek az alak, a szerkezet és a helyzet információt egy egyszerű, egységes keretbe foglalják.[2] Ez javítja a hatékonyságot és (szemben a többjellemzős módszerrel) csökkenti a többszörös keresési primitívek biztosításának igényét. Azonban, mivel ez a módszer az egész képre egy egyszerű szignót számít ki, a hasonló, de elhelyezkedésben vagy méretben különböző objektumokat tartalmazó képeket esetleg nem ismeri fel.
- **Régió szerint bontott (granuláris) wavelet alapú szignó** – Ebben a megközelítésben a szignó kiszámítása és összehasonlítása régió szerinti bontásban, nem pedig a teljes

2 | A wavelet-analízist a 3.4.3. alfejezet írja le.

képre nézve történik. Ez azon a megfigyelésen alapul, hogy a hasonló képek hasonló régiókat tartalmazhatnak, de az egyik képen lévő régió a másik képen lévő illeszkedő régió transzformáltja vagy skálázottja lehet. Így a lekérdező kép Q és a célkép T közötti hasonlóság mértékét az illeszkedő régiók által kapott területeknek a két képen elfoglalt arányával lehet meghatározni.

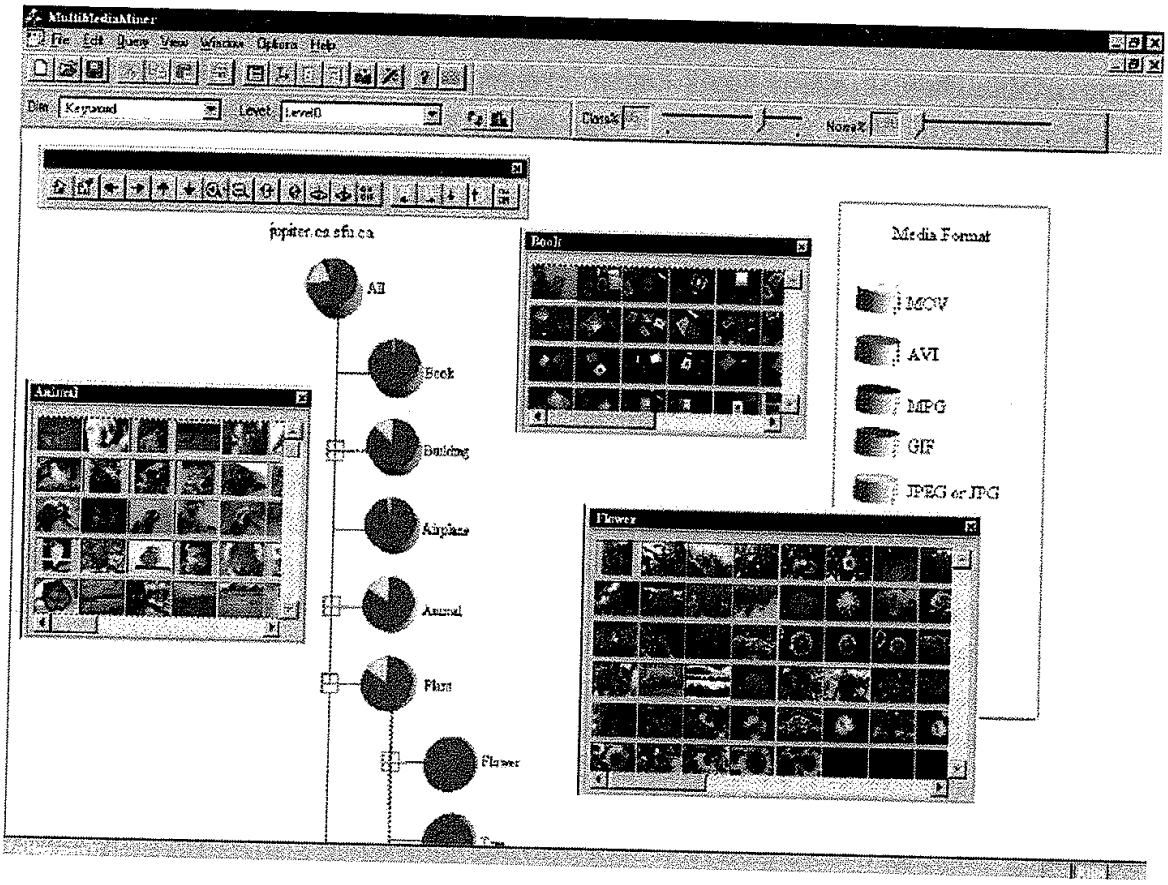
9.3.2. | Multimédia-adatok többdimenziós analízise

Létrehozhatunk-e adatkockát a multimédia-adatok analízise céljából? A nagy multimédia-adatbázisok többdimenziós analízisének végrehajtására a multimédia-adatkockákat a relációs adatok tradicionális adatkockáihoz hasonló módon lehet tervezni és létrehozni. A **multimédia-adatkocka** a multimédia-információhoz kiegészítő dimenziókat és mértékeket (szín, szerkezet és forma) tartalmazhat.

Vizsgáljuk meg a multimédia-adatbányászati rendszer MultiMediaMiner nevű prototípusát, amely multimédia-adatok kezelésével bővíti ki az ún. DBMinert (adatbázis adatbányászati rendszert). A MultiMediaMiner rendszer által tesztelt példaadatbázis a következőképpen készült. Minden képnek két leírója van: *jellegzetességleíró* és *elrendezésleíró*. Az adatbázis nem tárolja közvetlenül az eredeti képet, hanem csak a leíróit. A leíró információ olyan mezőket tartalmaz, mint a képfájl neve, kép URL, képtípus (gif, jpeg, bmp, avi, mpeg stb.), az összes ismert, a képre hivatkozó weblap URL-jét, (szülő URL-ek), a kulcsszavak listáját, valamint egy ikont, amelyet a kép- és videoböngészésre szolgáló felhasználói interfész használ. A **jellegzetességleíró** minden egyes vizuális jellemzőt külön-külön leíró vektorok készlete. A lényeges vektorok: 512 színre digitalizált ($8 \times 8 \times 8$ az $R \times G \times B$-nek megfelelően) színképet tartalmazó színvektor, egy MFC- (Most Frequent Color, leggyakoribb szín) vektor, MFO- (Most Frequent Orientation, leggyakoribb irányultság) vektor. Az MFC öt szín centroidot, az MFO öt peremirány centroidot tartalmaz az öt leggyakoribb szín, illetve az öt leggyakoribb peremirányultság számára. A használt peremirányultságok fokokban: 0°, 22,5°, 45°, 67,5°, 90° és így tovább. Az **elrendezésleíró** színelrendezés vektort és peremelrendezés vektort tartalmaz. Az összes képhez – eredeti méretétől függetlenül – hozzá van rendelve egy 8×8-as rácsháló. A 64 cellára külön-külön vonatkozó leggyakoribb szín a színelrendezés vektorban, a cellánként és irányultságonként a peremek száma pedig a perem-elrendezés vektorban kerül eltárolásra. Más méretű rácshálók (4×4, 2×2 és 1×1) könnyen generálhatók.

A MultiMediaMiner *Image Excavator* (képexkavátor) komponense a kép szövegkörnyezeti információit – például a weblapok HTML-elemeit – használja fel kulcsszavak kikeresésére. On-line könyvtárszerkezetek, például a Yahoo! könyvtár bejárásával létrehozható a képet tartalmazó könyvtárakkal összerendelt kulcsszavak hierarchiája. Az ilyen gráfokat a multimédia-adatkockában a *kulcsszó* dimenzió számára fogalmi hierarchiaként használják.

Milyen típusú dimenziókkal rendelkezhet egy multimédia-adatkocka? Egy multimédia-adatkocka sok dimenzióval rendelkezhet. Íme néhány példa: a kép vagy video mérete bájtokban; a keretek (vagy kép) két dimenziót adó szélessége és magassága; létrehozás (vagy a legutolsó módosítás) dátuma; formátum típusa; a képsorozat időtartalma másodpercekben; internet domain; a hivatkozó weblapok internet domainjai (szülő URL); kulcs-



9.5. ábra | A MultiMediaMiner *Osztályba soroló* moduljának outputja

szavak, szín és peremelrendezés dimenzió és így tovább. Sok numerikus dimenzió számára a fogalmi hierarchia automatikusan meghatározható. Más dimenziók esetén (például internet domain vagy szín) előre megadott hierarchiák használhatók.

A multimédia-adatkocka létrehozása elősegíti az elsődlegesen vizuális tartalmú multimédia-adatok többdimenziós analízisét, és a különböző típusú tudásbányászatot – az összegzés, hasonlítás, osztályozás, társítás alkalmazásával. A MultiMediaMiner *Osztályba soroló* modulja és a kimeneti képe a 9.5. ábrán látható.

A multimédia-adatkocka a multimédia-adatok többdimenziós analízisének érdekes modellje. Meg kell azonban jegyeznünk, hogy a dimenziók nagy száma esetén nehéz hatékonyan létrehozni egy adatkockát. A dimenziók átka különösen súlyos multimédia-adatkockák esetén. A színt, irányultságot, szerkezetet, kulcsszavakat stb. szeretnénk a multimédia-adatkocka (multi)dimenzióiként modellezni. Viszont ezek közül az attribútumok közül sok halmazorientált, mintsem egyszerű értékkel bír. Például egy kép megfelel egy kulcsszóhalmaznak. Tartalmazhat egy objektumhalmazt, ahol mindegyik objektumhoz egy színhalmaz van rendelve. Ha az adatkocka tervezésénél mindegyik kulcsszót vagy mindegyik részletezett színt külön dimenzióként használjuk, akkor nagyon sok dimenziónk keletkezik. Másfelől, ha nem így teszünk, akkor ez a kép elég durva, korlátozott, pontatlan modellezéséhez vezethet. A multimédia-adatkocka tervezése területén még további kutatásra van szükség, hogy elérjük a hatékonyság és a megjelenítési képesség közötti egyensúlyt.

9.3.3. | Multimédia-adatok osztályozása és előrejelzés-analízise

Az osztályozás és előrejelzés modellezése a multimédia-adatok adatbányászatának területén, különösen a tudományos kutatásoknál – mint a csillagászat, szeizmológia, földtudomány – használatos. A képadatok adatbányászatában alapvető adatbányászati módszer[3] a döntési fa szerinti osztályozás.

9.8. példa | A csillagászok által gondosan osztályozott égi felvételeket kísérleti készletként véve, a galaxisok, csillagok és más égi objektumok felismerésére különböző tulajdonságokon (nagyság, térség, erősség, képmomentum és irányultság) alapuló modelleket hozhatunk létre. Új égitestek azonosítása céljából nagyszámú, teleszkópok vagy űrszondák által készített égi felvételt lehet a létrehozott mintákkal összevetni. Hasonló sikeres vizsgálatok történtek a Vénusz vulkánjainak azonosítására. ■

Az ilyen képadatok adatbányászata esetén fontos az *adat-előfeldolgozás,* amely magába foglalhat adattisztítást, adat-összpontosítást és adatjellemzők kikeresését. A mintafelismerésnél használt alapvető módszerek – például a peremfelismerés és Hough-transzformációk – mellett a bizonytalanság kezelésére fel lehet tárni olyan technikákat, mint a sajátvektorra történő képdekompozíció vagy a bizonytalanságot kezelni tudó valószínűségi modellek adoptálása. Mivel a képadat gyakran nagy méretű és jelentős feldolgozó teljesítményt igényel, hasznos a párhuzamos és osztott feldolgozás.

A kép típusú adatok adatbányászata során használt osztályozás és klaszterezés szorosan kapcsolódik a képanalízishez és a tudományos adatok adatbányászatához. Így a képi adatok adatbányászatánál sok képelemzési technika és tudományos elemzési módszer alkalmazható.

9.3.4. | Társításbányászat a multimédia-adatokban

Milyen típusú társítások bányászhatók a multimédia-adatokban? A kép- és video-adatbázisokban a multimédia-objektumokra vonatkozó társítási szabályokat lehet bányászni. Legalább három kategóriát lehet megfigyelni:

- **A képi tartalom és a nem képi tartalom jellemzői közötti kapcsolatok** – Például a következő szabály: *„Ha a kép felső részének legalább 50%-a kék, akkor valószínű, hogy eget ábrázol"* ebbe a kategóriába tartozik, mivel a képi tartalomhoz hozzáköti az „ég" kulcsszót.
- **Térrelációban nem lévő képtartalmak közötti kapcsolatok** – Nézzük a következő szabályt: *„Ha a kép tartalmaz két kék négyzetet, akkor valószínű, hogy tartalmaz egy piros kört is"* ebbe a kategóriába tartozik, mivel a kapcsolatok a képtartalommal vannak összefüggésben.

3 | Az osztályozási módszereket, a döntési fákat is magában foglaló osztályozást is beleértve, a 7. fejezet vizsgálja.

- **Térrelációban lévő képtartalmak közötti kapcsolatok** – A következő szabály: *„Ha egy vörös háromszög van két sárga négyzet között, akkor valószínű, hogy egy nagy ovális objektum van alul"* ebbe a kategóriába tartozik, mivel a képen lévő objektumokat térösszefüggésben vizsgálja.

A multimédia-objektumok közötti kapcsolatok bányászatához minden egyes képet tranzakcióként kezelhetünk, és megtalálhatjuk a különböző képek között a gyakori mintákat.

Mi a különbség a multimédia-adatbázison és a tranzakciós adatbázison végzett társítási szabályok bányászata között? Néhány finom különbség található. Először, egy kép sok objektumot tartalmazhat, ahol mindegyik egy sor jellemzővel bír (szín, forma, textúra, kulcsszó, térbeli elhelyezkedés), így nagyszámú lehetséges kapcsolat fordulhat elő. Egy bizonyos felbontási szinten sok esetben két képnél azonosnak vehetünk egy jellemzőt, viszont finomabb felbontási szinten különbözőnek kell értelmeznünk. Következésképpen lényeges, hogy támogassuk a **felbontás fokozatos finomítása** módszerét. Ez azt jelenti, hogy a gyakran előforduló mintákat először viszonylag durva felbontási szinten bányászhatjuk, és aztán már a finomabb felbontási szinteken azokra összpontosíthatunk, amelyek átjutottak a minimális megalapozottsági küszöbön. Ezt azért tehetjük, mert a durva felbontási szinten nem gyakori minták nem lehetnek gyakoriak finomabb felbontási szinteken. Ilyen sokszoros – iteratív felbontási – adatbányászati stratégia lényegesen csökkenti a teljes adatbányászat költségeit anélkül, hogy az adatbányászat eredményének minőségét és teljességét csökkentené. Ez a nagy multimédia-adatbázisokban a gyakori tételcsoportok és kapcsolatok hatékony bányászati módszertanához vezet.

Másodszor, mivel egy többszörösen ismétlődő objektumokat tartalmazó kép fontos a képanalízis során, azonos objektumok ismétlődését nem lehet a társításelemzés során figyelmen kívül hagyni. Például, két arany karikát tartalmazó képet teljesen másképp kezelnek, mint azt, amely csak egyet tartalmaz. Ez teljesen eltér a tranzakciós adatbázisoknál történő kezeléstől, ahol azt a tényt, hogy valaki egy kanna vagy két kanna tejet vesz, gyakran úgy lehet kezelni, hogy *„vesz_tejet"*. Következésképpen, a multimédia-kapcsolatokat és annak mértékeit – a megalapozottságot és a megbízhatóságot – megfelelően kell beállítani.

Harmadszor, gyakran léteznek a multimédia-objektumok közötti lényeges térviszonyokat leíró relációk, például *fent, alatt, mellett, balra_tőle* és így tovább. Ezek a jellemzők nagyon hasznosak az objektumkapcsolatok és korrelációk felfedéséhez. Térbeli viszonyok, más tartalom alapú multimédia-jellemzőkkel együtt (szín, forma, szerkezet és kulcsszavak) érdekes kapcsolatokat alkothatnak. Így a téradatbányászati módszerek és a topologikus térbeli viszonyok tulajdonságai eléggé fontosakká váltak a multimédia-adatbányászatban.

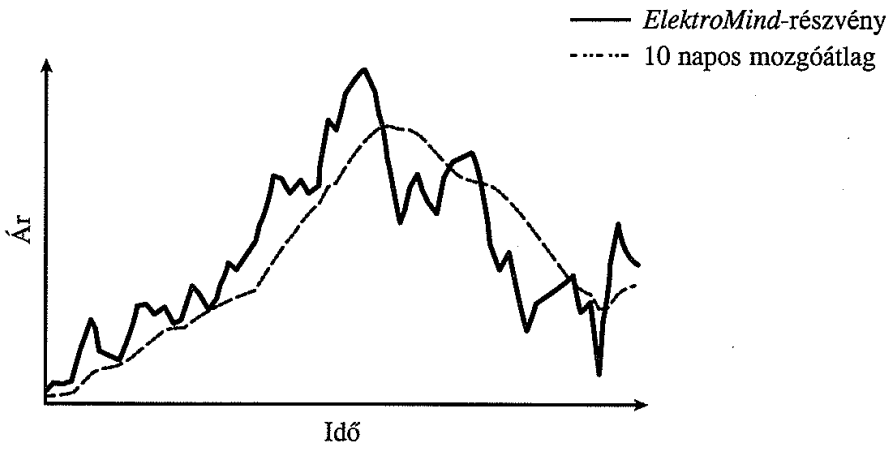
9.4. | Idősorok és szekvenciális adatok bányászata

Mi az idősor-adatbázis és a szekvencia-adatbázis? Az **idősor-adatbázis** időben változó értékek vagy események sorozatát tartalmazza. Az értékeket jellemzően egyenlő időintervallumokon mérik. Az idősor-adatbázisok sok felhasználási területen népszerűek, például az értékpapírpiac napi ingadozásainak vizsgálatánál, a dinamikus termelési folyama-

tok, tudományos kísérletek, orvosi kezelések stb. nyomon követésénél. Egy idősor-adatbázis egyben szekvencia-adatbázis is. Viszont a **szekvencia-adatbázis** lehet bármely olyan adatbázis, amely – időmegjelöléssel ellátott vagy anélküli – rendezett eseményeket tartalmaz. Például weblapokon történő szörfözés, bejárás sorozatszekvencia-adat, de nem szükségszerű, hogy idősoradat legyen.

9.4.1. | Trendanalízis

Az Y változót (amely mondjuk, az értékpapírpiacon egy részvény napi záró árát jelenti) tartalmazó idősort az idő (t) függvényeként tekinthetjük, vagyis $Y = F(t)$. Egy ilyen függvényt idősorgrafikonként ábrázolhatunk, ahogy azt a 9.6. ábra mutatja, amely az idő haladásával mozgó pontot ábrázol.



9.6. ábra | Idősoradatok: az *ElektroMind*-részvény árfolyama az idő függvényében

Hogyan vizsgálhatjuk az idősoradatokat? Négy fő összetevőt (vagy mozgást) használnak az idősoradatok karakterizációjára:

- **Hosszú távú vagy trendmozgások** – Megmutatják azt az általános irányt, amely mentén az idősorgrafikon hosszú időszak során mozog. Ezt a mozgást egy **trendgörbe** vagy **trendvonal** ábrázolja. Például a 9.6. ábrán a trendgörbét szaggatott vonal jelzi. A trendgörbe vagy trendvonal meghatározásának tipikus módszere többek között a súlyozott mozgóátlag és a legkisebb négyzetek módszere, amelyekről később szó lesz.
- **Ciklikus mozgások vagy ciklikus változások** – Ezek a mozgások *ciklusoknak* felelnek meg, vagyis a hosszú periódusú, a trendvonal körüli ingadozásoknak, amelyek lehetnek akár periodikusak, akár aperiodikusak. Ez azt jelenti, hogy a ciklusoknak azonos időintervallumokban nem kell feltétlenül pontosan hasonló mintát követniük.
- **Szezonális mozgások vagy szezonális változások** – Ilyen mozgásokat az évente ismétlődő események váltják ki, például a Valentin nap előtti csokoládé- és virágvásárlás, vagy a karácsonyi vásár alatt az áruházi cikkek forgalmának hirtelen megugrása. Más szóval a szezonális mozgások évről évre a megfelelő hónapokban azonos vagy közel azonos idősormintákat követnek.

- **Szabálytalan vagy véletlen mozgások** – Ezeket a véletlen vagy alkalmi eseményeknek (munkaügyi viták, árvizek, vállalatokon belüli személyi változások) köszönhetően szórványos, esetleges mozgás jellemzi.

A fent említett trend-, ciklikus, szezonális és szabálytalan mozgásokat megfelelően a *T, C, S, I* változók jelölik. Az idősorok elemzése megfelel az idősor négy alapvető mozgására történő lebontásnak, **dekompozíciónak**. Az *Y* idősorváltozót a négy változó szorzataként ($Y = T \times C \times S \times I$) vagy összegeként modellezhetjük. A választásnak jellemzően gyakorlati oka van.

Hogyan tudjuk meghatározni az adatok trendjét, ha adva van az Y egy értéksora $(y_1, y_2, y_3, \ldots)$*?* A trend kiszámításának általános módszere az ***n*-ed rendű mozgóátlag** kiszámítása, amely a következő számtani közepekből álló sorozat:

$$\frac{y_1 + y_2 + \ldots + y_n}{n}, \frac{y_2 + y_3 + \ldots + y_{n+1}}{n}, \frac{y_3 + y_4 + \ldots + y_{n+2}}{n}, \ldots \tag{9.3}$$

A mozgó átlag igyekszik csökkenteni az adatsorban lévő ingadozások mennyiségét. Ily módon az idősor felváltása mozgóátlagokkal kiküszöböli a fölösleges fluktuációkat, következésképpen az **idősor simításának** felel meg. Ha súlyozott számtani közepet használunk a sorozatban (9.3), akkor az eredményként adódó szekvencia az ***n*-ed rendű súlyozott mozgóátlag**.

9.9. példa | Egy kilenc értékből álló szekvenciánk van. Kiszámíthatjuk a harmadrendű mozgóátlagát, valamint a harmadrendű súlyozott mozgóátlagát az (1, 4, 1) súlyok felhasználásával. Ez az információ táblázatos formában ábrázolható, ahol a mozgóátlag mindegyik értéke a közvetlenül felette lévő három érték középértéke, és a súlyozott mozgóátlag mindegyik értéke a közvetlenül felette lévő három érték súlyozott középértéke.

Eredeti adatok:	3	7	2	0	4	5	9	7	2
Harmadrendű mozgóátlag:		4	3	2	3	6	7	6	
Súlyozott (1, 4, 1) harmadrendű mozgóátlag:		5,5	2,5	1	3,5	5,5	8	6,5	

Az első súlyozott átlagot a következőképp kaptuk: $\frac{1\times 3 + 4\times 7 + 1\times 2}{1+4+1} = 5{,}5$. A súlyozott átlagnál a simító hatás csökkentése céljából a középső értékek nagyobb súlyt kapnak. ■

A mozgóátlagnál a sorozat első és utolsó értékének elvesztése néha ciklusokat vagy más olyan mozgásokat generálhat, amelyek nincsenek jelen az eredeti adatban, és erre az extrém értékek jelenléte is erős hatással lehet. Megjegyezzük, ahogy a 9.9. példa mutatja, hogy az extrém értékek hatása a megfelelő súlyokkal rendelkező súlyozott mozgóátlag alkalmazásával csökkenthető.

Az adatokban levő ciklikus, szezonális és szabálytalan minták a megfelelő rendű súlyozott mozgóátlag használatával kiküszöbölhetők, és ezzel csak a trend mozgását nyerjük.

Léteznek más módok is a trend kiértékelésére? Igen, egy ilyen módszer a **szabad kéz**

módszere, ahol a közelítő görbe vagy vonal a felhasználó ítélete alapján úgy lesz berajzolva, hogy illeszkedjen az adatsorhoz. Ez a módszer drága, és nem elég megbízható a nagymérvű adatbányászat számára. Alternatív módszer a **legkisebb négyzetek módszere**. A legjobban illeszkedőnek tartjuk azt a legkisebb négyzetes C görbét, amelyre $\sum_{i=1}^{n} d_i^2$ minimális, ahol d_i az eltérés vagy a hiba, azaz az (x_i, y_i) éppen y_i értékének és a C görbén meghatározott megfelelő értéknek a különbsége.

Van valamilyen mód arra, hogy beállítsuk a szezonális ingadozás adatait? Sok üzleti tranzakciónál várható rendszeres szezonális ingadozás, például a karácsonyi időszakban a magasabb vásárlási forgalom. Érdemes tehát az ilyen szezonális változásokat felfedni, a trendadatokat „szezonmentesíteni" a trend- és ciklusadat-analízis számára. Erre a célra szolgál a **szezonális index**, ami egy változónak az év hónapjai alatti relatív értékeit mutató számsor. Például ha októberben, novemberben és decemberben a forgalom az egész évre vett havi átlagos forgalomnak sorban 80%, 120%, illetve 140%-a, akkor az év **szezonális indexszámai** 80, 120, 140. Ha az eredeti havi adatot osztjuk a megfelelő szezonális indexszámmal, az eredményül kapott adatot **szezonmentesítettnek**, vagyis a *szezonális változáshoz igazítottnak* tartjuk. Az ilyen adatok még mindig tartalmaznak trendet, ciklikus és szabálytalan mozgásokat.

A szezonmentesített adatok a megfelelő trendértékekkel történő osztással a trendhez igazíthatók. Ráadásul, a megfelelő mozgóátlag a további elemzés számára kisimítja a szabálytalan ingadozásokat, és csak a ciklikus ingadozásokat hagyja meg. Ha a ciklusok periodikus vagy közelítően periodikus módon jelennek meg, a szezonális indexekhez hasonlóan ciklikus indexeket lehet létrehozni.

Végül a szabálytalan vagy véletlen mozgásokat az adatoknak a trendhez, a szezonális és ciklikus változásokhoz igazításával értékelhetjük ki. Általában, a normális eloszlást követve, a kis eltérések nagy gyakorisággal fordulnak elő, míg a nagy eltérések előfordulása ritka.

A gyakorlatban hasznos először grafikonon megjeleníteni az idősort, és minőségileg kiértékelni a hosszú távú trendeket, a szezonális és ciklikus változásokat. Ez segíthet a megfelelő elemző módszer kiválasztásában és az eredmények megértésében.

A *trend*, a *ciklikus*, a *szezonális* és a *szabálytalan* komponensek szisztematikus elemzésével elfogadható minőségű hosszú és rövid távú jóslást (**idősorok előrejelzését**) végezhetünk.

9.4.2. | Hasonlóság keresése az idősorok elemzésében

Mit jelent a hasonlóságkeresés? A normál adatbázis-lekérdezésekkel szemben, ahol azok az adatok kerülnek kiválasztásra, amelyek pontosan illeszkednek az adott lekérdezéshez, a **hasonlóságkeresés** azokat az adatszekvenciákat találja meg, amelyek *csak kismértékben különböznek* az adott lekérdezési szekvenciától. Adott idősorszekvenciák egy halmazához kétfajta hasonlóság keresés létezik. A **részszekvencia-illeszkedés** azokat az adatszekvenciákat találja meg, amelyek hasonlóak a megadottal, míg a **teljes szekvenciailleszkedés** az egymáshoz hasonló szekvenciákat találja meg. Az idősorok elemzése során végzett hasonlóságkeresés a pénzpiaci (részvényadatok elemzése), az orvosdiagnosztikai (kardiogram-analízis), a tudományos vagy mérnöki adatbázisok elemzése terén hasznos.

Adattranszformáció: időtartományról a frekvenciatartományra

A idősoradatok hasonlósági elemzésére a hasonlóság mértékéül tipikusan az euklideszi távolság használatos.

Transzformálnom kell az adatokat? A jelanalízis sok módszeréhez frekvenciatartományban reprezentált adatokra van szükség. Következésképpen a távolságtartó ortonormált transzformációk gyakran használatosak az adatok időtartományból frekvenciatartományba történő transzformációjához. Általában adatfüggetlen transzformációt alkalmaznak, ahol a transzformáció mátrixa az inputadattól függetlenül, előre kerül meghatározásra. Két népszerű adatfüggetlen transzformáció a *diszkrét Fourier-transzformáció (discrete Fourier transform, DFT)* és a *diszkrét wavelet-transzformáció (discrete wavelet transform, DWT)*.[4] Mivel két jel között az időtartományban vett távolság azonos a frekvenciatartományban vett euklideszi távolsággal, a DFT az első néhány együtthatóban megőrzi a lényeget. A DFT első néhány (a „legerősebb") együtthatójának megtartásával a tényleges távolság alsó határát tudjuk kiszámítani.

Ha az adatokat például DFT-vel transzformáltuk, hogyan lehet végrehajtani a hasonlóságkeresést? A hatékony hozzáférés érdekében az első néhány Fourier-együttható segítségével többdimenziós index hozható létre. Amikor a rendszer egy hasonlóság-lekérdezési kérést kap, az index arra használható, hogy kikeressük azokat a szekvenciákat, amelyek legfeljebb egy meghatározott kis távolságra vannak a lekérdezési szekvenciától. Az utófeldolgozás kiszámítja az időtartományban lévő tényleges távolságokat, és eldobja a hamis illeszkedéseket.

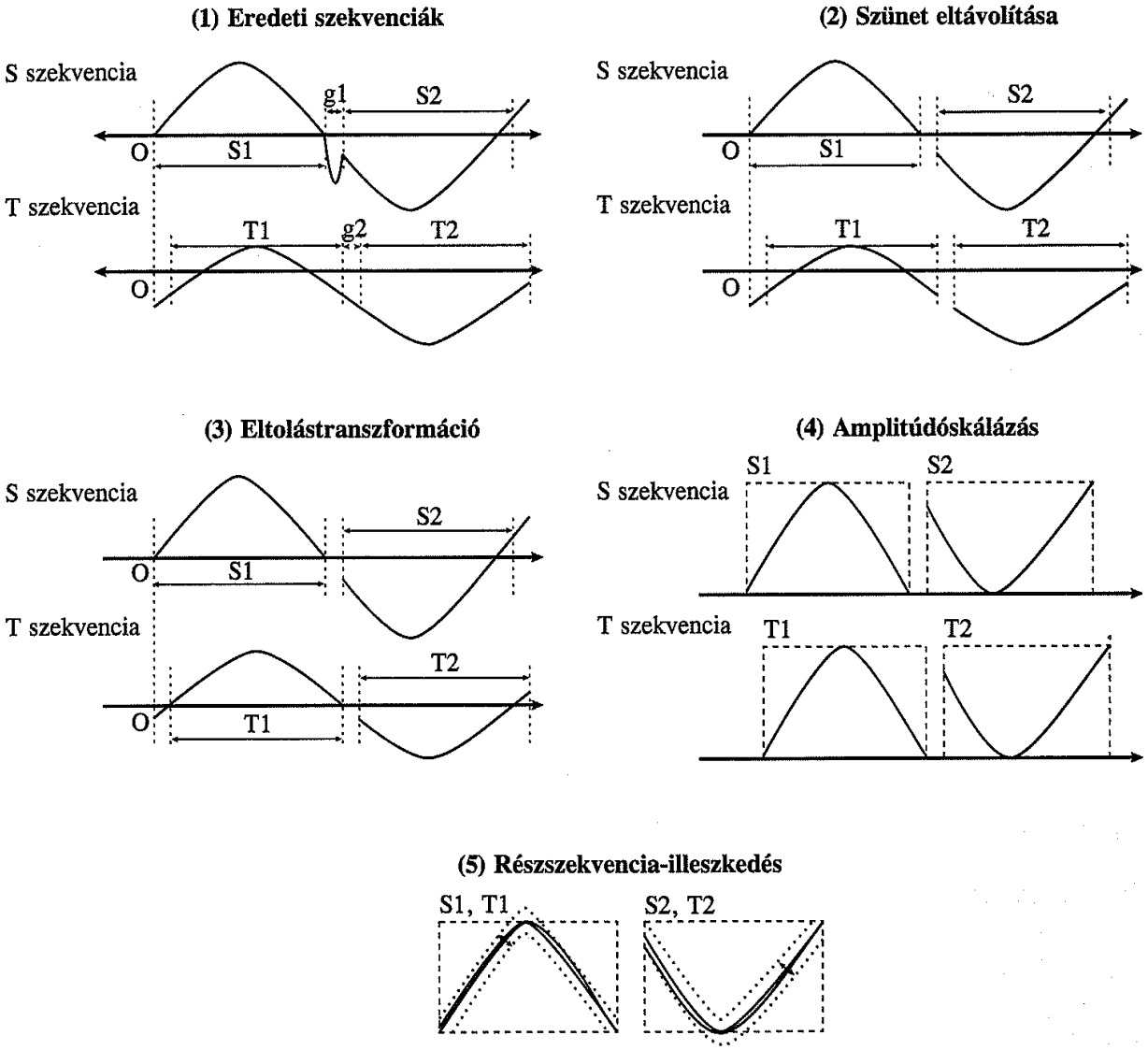
Mi a helyzet a részszekvencia-illeszkedéssel? Részszekvencia-illeszkedés esetén mindegyik szekvencia először *w* hosszúságú *ablakdarabkákból álló* csoportra darabolódik fel. Ezután nyerjük ki a részszekvenciák jellemzőit. Minden egyes szekvenciához hozzá tartozik a tulajdonságtérben egy nyom. A részszekvencia-elemzés számára minden egyes szekvencia nyoma résznyomokra lesz osztva, ahol mindegyik résznyomot egy minimálisan határolt téglalap ábrázol. Ezután a sok apró darabot összerakó algoritmus alkalmazható a hosszabb szekvenciailleszkedések megkeresésére.

Fejlett hasonlóságkeresési módszerek a szünetek, valamint az eltolás és amplitúdó-különbségek kezelésére

A valós élet alkalmazásainak nagy részében nincs szükség arra, hogy az illeszkedő részszekvenciák az időtengely mentén sorban, egymáshoz tökéletesen igazodva helyezkedjenek el. Más szóval, meg kell engedni, hogy a részszekvencia-párokat akkor is illeszkedőknek tekinthessük, ha ugyan *formájuk azonos*, de a szekvencián belüli szünetek, érték-eltolásban vagy amplitúdóban meglévő eltérések miatt különböznek egymástól. Ez különösen hasznos sok hasonlósági szekvenciaelemzés esetén, például a részvénypiac és a kardiogramok analízisénél.

Hogyan hajtható végre az ilyen különbségeket megengedő, fejlett hasonlóságkeresés? A fejlett hasonlósági modellnél a felhasználók vagy szakemberek olyan paramétereket határozhatnak meg, mint a mozgó ablak mérete, a hasonlósági burkoló szélessége, a

4 | A diszkrét Fourier- és a diszkrét hullámtranszformációt röviden a 3.4.3. alfejezet ismerteti.



9.7. ábra | Részszekvencia-illeszkedés az idősoradatoknál. Az eredeti szekvenciák azonos alakúak, de igazításokra van szükség, hogy megbirkózzunk a szünetek, az eltolások és az amplitúdók különbözőségével. Ezek az igazítások módot adnak arra, hogy a részszekvenciák összeilleszthetők legyenek ε szélességű burkolón belül

maximális szünet, az illeszkedési tényező és így tovább. Nézzük a 9.7. ábrát, amely két időszekvenciát mutat be, amelyekből a szünetet eltávolítjuk. Az így kapott szekvenciát az eltolás és az amplitúdómérték szempontjából normalizáljuk, így mód nyílik a különböző amplitúdóértékű és eltolású részszekvenciák összehasonlítására. Két részszekvenciát *hasonlónak* tekinthetünk, és illeszkedhetnek egymáshoz, ha az egyik egy, a másikat körülvevő, ε szélességű burkolón belül fekszik, a szélsőséges értékeket nem vesszük figyelembe (ahol ε egy felhasználó vagy szakértő által meghatározott kis szám). Két szekvencia akkor *hasonló,* ha elég sok hasonló részszekvencia-párt tartalmaz, amelyek időben nem fedik egymást.

A fentiek alapján a szünet, az eltolás és az amplitúdó különbözőségét kezelni tudó hasonlóságkeresés az alábbi lépésekben hajtható végre:

(1) **Atomi illeszkedés** – Az összes olyan kisméretű, szünetet nem tartalmazó, rövid ablakpár megkeresése, amelyek hasonlóak.
(2) **Ablakok összefűzése** – A hasonló ablakok összefűzése, hogy kialakíthassunk nagy, hasonló részszekvencia-párokat, ahol az atomi illeszkedések között már megengedhető a szünet.
(3) **Részszekvencia rendezése** – A részszekvencia-illeszkedések lineáris sorbarendezése, hogy eldönthessük, van-e elegendő számú hasonló rész.

Hasonló formájú, de szünetekkel, eltolás vagy amplitudó eltérésekkel rendelkező szekvenciák ilyen feldolgozása segítségével meg lehet találni az egymáshoz vagy a lekérdező szekvenciához illeszkedő szekvenciákat.

A hasonlóságkeresés indexelési módszerei

Létezik valamilyen stratégia a hatékony megvalósításra? A nagy adatbázisokban történő hasonlóságkeresés hatékony végrehajtása érdekében különböző indexelési módszereket vizsgáltak. Például, hogy meggyorsítsuk a hasonlóságkeresést, a minimálisan érintkező téglalapok tárolásánál R-fákat és R^*-fákat alkalmaztak. Emellett, a magas dimenziós pontoknál gyorsabb térbeli hasonlósági összekapcsolások érdekében az *ε-kdB fákat* valamint a *suffix fákat* fejlesztették ki.

Az idősorok lekérdező nyelvei

Hogy adhatok meg egy utasítást hasonlóságkeresésre? Szükség van az idősorokban történő keresés leírását/specifikálását támogató hathatós nyelv tervezésére és kifejlesztésére. Egy **idősorlekérdező nyelvvel** nemcsak olyan egyszerű hasonlóságlekérdezéseket kell tudni megadni, mint *„Keresd meg a megadott részszekvenciához hasonló összes szekvenciát!"*, hanem olyan bonyolult lekérdezéseket is mint *„Keresd meg az összes olyan szekvenciát, amely hasonló valamely A osztálybeli szekvenciához, de nem hasonló egyetlen B osztálybeli szekvenciához sem!"* Sőt, különböző típusú (tartomány, összes pár, legközelebbi szomszéd) lekérdezéseket is lehetővé kell tennie.

Az idősorlekérdező nyelv egy másik érdekes fajtája az alakleíró nyelv. Ez módot ad a felhasználóknak, hogy a specifikus részletek mellőzése mellett *szekvenciaátmenetek* vagy makrók olvasható sorozatával írják le és kérdezzék le az idősorok *teljes lefutási görbéjének alakját*.

9.10. példa | A *fel, Fel, FEL* minta használható egy emelkedő lejtő növekvő meredekségi fokának megadására. Egy *hegy* makró megfelelhet a következő szekvenciának: (*Emelkedő, lapos, Ereszkedő*), ahol az *Emelkedő* a következőképpen van megadva: ({*Fel, FEL*}, {*Fel, FEL*}, {*Fel, FEL*}). Ez azt jelenti, hogy egy *Emelkedő* három felmenő rézsűt jelent, és mindegyik vagy *Fel*, vagy *FEL*. Az *Ereszkedő* hasonlóan adható meg. ■

Egy ilyen alakleíró nyelv segítségével a felhasználó nagyobb rugalmassággal tudja a szekvenciahasonlóság keresésénél a kívánt alakú időlefutás-lekérdezést megadni.

9.4.3. | Szekvenciális minták bányászata

Mit takar a szekvenciális minták bányászata? Szekvenciális minták bányászata az idősorban vagy más sorozatban gyakran előforduló minták bányászata. Példa a szekvenciális mintára: *„Az a vevő, aki kilenc hónappal ezelőtt vett egy Pentium PC-t, bizonyára egy hónapon belül megrendel egy új CPU chipet."* Mivel sok üzleti tranzakció, távközlésforgalmi adat, időjárási adat, termelési folyamat idősorrendű adat, ezért a szekvenciális minták bányászata az ilyen adatok elemzésében hasznos a célzott marketing, az ügyfelek megtartása, az időjárás előrejelzése stb. céljából.

Esetek és paraméterek a szekvenciaminta-bányászat számára

A szekvenciaminta-bányászattal foglalkozó tanulmányok többsége a *szimbolikus mintákra* koncentrál, mialatt a numerikus görbeminták általában a trendanalízis és a statisztikai idősorok elemzésével végzett előrejelzés felhasználási területéhez tartoznak, amit a 9.4.1. alfejezet tárgyal.

Van néhány paraméter, amelyek megadása, beállítása erősen befolyásolhatja a szekvenciaminta-bányászat eredményét. Az első ilyen paraméter az idősor **időtartama**, *T*. Az időtartam átfoghatja az adatbázisban lévő teljes elérhető szekvenciát vagy a felhasználó által megadott részszekvenciát, például azt, amely 1999-nek felel meg. Így a szekvenciaminta-bányászat a megadott időtartam alatti adatokra korlátozódhat. Az időtartam különálló szekvenciák halmazaként is definiálható, például minden hét, vagy az összes tőzsdei összeomlás utáni hét, vagy az összes vulkánkitörés előtti és utáni két-két hét. Ilyen esetekben *periodikus mintákat* tárhatunk fel.

A második paraméter az **eseménybefoglaló ablak**, *w*. Bizonyos elemzésekben egy meghatározott időszak alatti eseményeket együttesen bekövetkezett eseményeknek tekinthetjük. Ha *w*-t akkorára állítjuk, mint a *T* időtartam, akkor időérzéketlen gyakori mintákat kapunk – ezek alapvetően illeszkedési minták, például *„Azok a vevők, akik 1999-ben PC-t vettek, digitális kamerát is vettek"* (nem tekintve azt, hogy melyiket vették előbb). Ha a *w* értéke 0 (nincs eseményösszefogó tárolás), akkor olyan szekvenciamintákat kapunk, ahol az események különböző időpillanatban történnek, például *„Az a vevő, aki egy PC-t, majd azután egy memóriachipet vett, valószínű, hogy később egy CD-ROM-ot is venni fog."* Ha a *w* a két érték között van (az ugyanabban a hónapban vagy egy 24 órás mozgó ablakban előforduló tranzakciók számára), akkor ezeket a tranzakciókat ugyanabban az időszakaszban előfordulóknak tekintjük, ezeket a szekvenciákat az elemzés alatt összefogjuk.

A harmadik paraméter a feltárt mintában az események közötti **időintervallum**, *int*. Ennek a paraméternek a következő beállításai lehetnek:

- *int* = 0: Ez azt jelenti, hogy semmilyen időköz nincs megengedve, a szorosan egymás után következő mintákat találja meg, mint amilyen az $a_{i-1}a_ia_{i+1}$ minta, ahol a_i az *i* időben bekövetkezett eseményt jelöli. Ilyen esetben figyelembe vehető az eseménybefoglaló ablak, *w*. Például, ha az eseménybefoglaló ablak egy hétre van beállítva, akkor a közvetlenül egymás utáni hetekben bekövetkezett gyakori mintákat fogjuk meg-

találni. DNS-elemzés esetén gyakran van szükség intervallumszünet nélküli egymás utáni szekvenciák feltárására.

- *min_interval* $\leq$ *int* $\leq$ *max_interval*: Ez azt jelenti, hogy olyan mintákat szeretnénk találni, amelyek legalább *min_interval*, de legfeljebb *max_interval* távol vannak egymástól. Például, *„Ha egy személy kikölcsönözi az A filmet, akkor valószínű, hogy 30 napon belül kikölcsönzi a B filmet"* maga után vonja, hogy *int* $\leq$ 30 (nap).
- *int* = *c* $\neq$ 0: A felhasználók olyan mintákat szeretnének találni, amelyek pontosan *int* távolságra vannak egymástól. Például, a következő lekérdezés: *„Mindig, amikor a Dow Jones több mint 5%-ot esik, mi fog történni pontosan két nap múlva?"* az *int* = 2 (nap) értékkel rendelkező szekvenciamintákat keres.

A felhasználó *egymás utáni történések*, illetve *párhuzamos történések* formájú „mintasablonok" megadásával a bányászathoz használt szekvenciaminták típusaira megszorításokat határozhat meg. Az **egymás utáni történés** kötött sorrendű események csoportja, míg a **párhuzamos történés** olyan események csoportja, ahol a sorrend lényegtelen. (E, t) jelölje a *t időben* bekövetkezett *E típusú eseményt*. Tekintsük az $(A, 1)$, $(C, 2)$ és $(B, 5)$ adatokat és egy 2 szélességű eseménybefoglaló ablakot, ahol mind az $A \rightarrow B$ egymás utáni történés, mind az A & C párhuzamos történés előfordul. A felhasználó reguláris kifejezéssel is megadhatja a megszorításokat, mint például $(A \mid B)C * (D \mid E)$, ami olyan keresést ír elő, ahol A és B következik be előbb (egymás közötti sorrend lényegtelen), ezt követi a C esemény vagy a C egy csoportja, ezután jön a D és az E (ahol D az E előtt is, vagy akár utána is előfordulhat). Vegyük észre, hogy a reguláris kifejezésben megadott események bekövetkezése között más események is előfordulhatnak.

Szekvenciaminta bányászatának módszerei

A társítási szabályok bányászatában alkalmazott *Apriori tulajdonság* alkalmas a szekvenciaminták bányászatára, mivel ha a k hosszúságú szekvenciaminta ritka előfordulású, akkor ennek egy szuperszetje ($k + 1$ hosszú) is ritka. Így a legtöbb szekvenciaminta-bányászati módszer az Apriori jellegű algoritmusok különböző változatait használja, habár eltérő paraméter-beállításokat és korlátozásokat is figyelembe vehet. Az ilyen minták bányászatának egy másik megközelítése a vetített adatbázis alapú szekvenciaminta-bővítési eljárás, ami hasonló a jelölt generálása nélküli gyakori minták bányászatához (a 6.2.4. alfejezetben tárgyalt) használt FP bővítési eljáráshoz (gyakori minta bővítés).

9.4.4. | Ismétlődésanalízis

Mi az ismétlődésanalízis? Az ismétlődésanalízis az ismétlődő minták bányászata, periodikusan visszatérő minták keresése az idősor-adatbázisokban. Az ismétlődésanalízis sok fontos területen alkalmazható. Például az évszakok, vízállások, égitestpályák, napi energiafogyasztás, napi közlekedési minták, heti tv-programok mind rendelkeznek ismétlődő mintákkal.

Ahogy az előző részben megjegyeztük, az ismétlődő minták bányászatát tekinthetjük szekvenciaminták bányászatának, ahol az időszakokat különálló szekvenciacsoportként

értelmezzük, például a naptári évek, bizonyos eseményeket követő vagy megelőző időszakaszok és így tovább. A ismétlődő minták bányászatának problematikáját három csoportra bonthatjuk:

- **Teljes ismétlődő mintabányászat** – Az idő minden egyes pontja hozzájárul (pontosan vagy közelítőleg) az idősor ciklikus viselkedéséhez, például az év minden napja *közelítőleg* hozzájárul az évben lezajló évszak ciklikussághoz.
- **Részleges ismétlődő mintabányászat** – Megadja az idősorok néhány, de nem az összes időpontjának ismétlődő viselkedését. Például Sandy a hét minden napján 7:00 és 7:30 között olvassa a *New York Times*-ot, de a más időben végzett tevékenysége nem mutat nagy rendszerességet. A részleges ismétlődés a teljes ismétlődés gyengébb formája, és sokkal gyakrabban is fordul elő a valóságban.
- **Ciklikussági vagy ismétlődéstársítási szabálybányászat** – Olyan szabályokról van szó, melyek az ismétlődően előforduló események közötti kapcsolatokat adják meg. Az ismétlődéstársítási szabályra példa: *„A mindennapi forgalom alapján, ha 15:00 és 17:00 óra között jól fogy a tea, akkor a hétvégeken jó vacsorai forgalom lesz 19:00 és 21:00 óra között".*

A teljes ismétlődésanalízis technikái a jelanalízis és a statisztika területén kerültek tanulmányozásra. Az ilyen elemzéseket segítik az olyan módszerek alkalmazása, mint például az FFT (Fast Fourier Transformation, gyors Fourier-transzformáció), amellyel az adatokat az időtartományból a frekvenciatartományba transzformáljuk.

Lehet-e a teljes ismétlődő mintákat kereső módszereket a részleges ismétlődő mintabányászathoz használni? A részleges ismétlődés hatékony bányászata az adatbányászati kutatásokban mostanában került tanulmányozásra. A teljes ismétlődő mintabányászat legtöbb módszere alkalmazhatatlan és elfogadhatatlanul drága a részleges ismétlődő minták bányászatához, ahol is ugyanabban az időperiódusban periodikus és nem periodikus események keverednek. Például FFT nem használható részleges ismétlődés bányászatához, mivel az az idősorokat szétválaszthatatlan értékfolyamként kezeli. Egyes ismétlődésdetektáló módszerek csak akkor képesek feltárni néhány részleges mintát, ha bizonyos viselkedéssel rendelkező részleges mintákban a szegmens periódusa, hossza és üteme explicit módon van megadva. Az újságolvasási példánál részleteket kell megadni, például „Keresd Sandy 7:00 óra után fél órán keresztül végzett rendszeres tevékenységét 24 órás napi időszakaszban!" Az ilyen módszereknek a részleges ismétlődő mintabányászati problémához történő laikus alkalmazása elfogadhatatlanul drága lehet, mivel a három paraméter – ismétlődés, hossz, ütemezés – nagyszámú kombinációját kell használni.

A legtöbb részleges ismétlődő minta- és ciklikustársítási szabálybányászattal foglalkozó munka az Apriori tulajdonságot heurisztikusan alkalmazza, és az Apriori típusú bányászati módszerek valamilyen változatát használja. Korlátok is elhelyezhetők a szekvenciális és ismétlődő mintákat bányászó folyamat mélyébe. Az Apriori tulajdonságot, az Apriori algoritmus változatait és a bányászó megszorítások használatát a 6. fejezet tárgyalja.

9.5. | Szöveges adatbázisok bányászata

Az adatbányászattal foglalkozó korábbi tanulmányok a strukturált – relációs, tranzakciós, valamint adattárház – adatokra koncentráltak. A valóságban a rendelkezésre álló információ jelentős része azonban **szöveges adatbázisokban** (vagy **dokumentum-adatbázisokban**) van tárolva, amelyek különböző forrásból származó dokumentumok nagy gyűjteményeiből állnak, mint amilyenek az újságcikkek, kutatási dolgozatok, könyvek, digitális könyvtárak, e-mail üzenetek és weblapok. A szöveges adatbázisok az elektronikus formában elérhető információ – például elektronikus publikációk, e-mail, CD-ROM-ok és a World Wide Web (amit egy nagy, összekapcsolt, dinamikus szöveges adatbázisnak is tekinthetünk) – egyre gyarapodó mennyiségének köszönhetően gyors ütemben növekednek.

A szöveges adatbázisok legtöbbjében a tárolt adat félig strukturált, ahol az adat nem teljesen strukturálatlan, de nem is teljesen strukturált. Például, egy dokumentum tartalmazhat néhány strukturált mezőt, például cím, szerzők, kiadási dátum, méret, téma és így tovább, de szintúgy tartalmazhat néhány nagymértékben strukturálatlan szöveges összetevőt, például kivonat, tartalomjegyzék. Sok tanulmány született a korábbi adatbáziskutatások során a félig strukturált adatok modellezésével és megvalósításával kapcsolatban. Ezenfelül a strukturálatlan dokumentumok kezelésére információvisszakeresési technikák – szöveges indexelési módszerek – kerültek kifejlesztésre.

A tradicionális információvisszakeresési technikák alkalmatlanná váltak az egyre növekvő mennyiségű szöveges adat kezelésére. Jellemző, hogy a sok elérhető dokumentumnak csak egy kis hányada érdekes egy adott személy vagy felhasználó számára. Az elemzés és a hasznos információ kinyerése céljából hatékony lekérdezést nehéz megfogalmazni anélkül, hogy tudnánk, mit is tartalmazhat egy dokumentum. A felhasználóknak olyan eszközökre van szükségük, amelyekkel különböző dokumentumokat összehasonlítanak, rangsorolják a dokumentumok fontosságát, relevanciáját, vagy több dokumentumban mintákat és tendenciákat keresnek. Így a szöveges bányászat egyre növekvő népszerűségű és alapvető témája az adatbányászatnak.

9.5.1. | Elemzés és információvisszakeresés szöveges adatokon

Mi is az információvisszakeresés? Az **információvisszakeresés** olyan terület, amely hosszú évek alatt az adatbázisokkal párhuzamosan fejlődött. Eltérően az adatbázisrendszerektől – amelyeknél a strukturált adatok lekérdezésén és tranzakciós feldolgozásán volt a hangsúly – az információvisszakeresés nagyszámú szöveges alapú adatbázisból történő információszervezéssel és kinyeréssel foglalkozik. Tipikus információvisszakeresési probléma a releváns dokumentumoknak a felhasználó által megadott információ – kulcsszó vagy mintadokumentum – alapján történő kiválasztása. A tipikus információvisszakereső rendszerek tartalmaznak on-line könyvtári katalógusrendszereket és on-line dokumentumkezelő rendszereket.

Mivel az információvisszakereső rendszerek és az adatbázisrendszerek különböző típusú adatokat kezelnek, vannak olyan adatbázisrendszereket érintő problémák, amelyek általában nem jellemzőek az információvisszakereső rendszereknél, például konkuren-

ciakezelés, helyreállítás, tranzakciókezelés, adatfrissítés. Szintén vannak olyan általános információvisszakeresési problémák, amelyekkel általában nem találkozunk a tradicionális adatbázisoknál, például strukturálatlan dokumentumok, kulcsszavakon alapuló közelítő keresés, a relevancia fogalma.

A szöveges visszakeresés alapvető mérőszámai

Tegyük fel, hogy a lekérdezési formában beadott kérdésemre a szöveges lekérdező rendszer számos dokumentumot adott vissza. Hogy tudjuk megállapítani azt, hogy mennyire volt „pontos" vagy „helyes" a rendszer? Legyen a lekérdezésnek megfelelő dokumentumhalmaz a $\{Releváns\}$, a visszanyert dokumentumhalmaz a $\{Visszanyert\}$. Mint ahogy a 9.8. ábrán a Venn-diagram mutatja, az egyaránt releváns és visszanyert dokumentumok halmaza a $\{Releváns\} \cap \{Visszanyert\}$. A szöveges visszakeresés minőségének megállapítására két mérőszám áll rendelkezésünkre:

- **Pontosság** – A visszanyert dokumentumokban a lekérdezésnek ténylegesen megfelelők („helyes" válaszok) százalékos aránya:

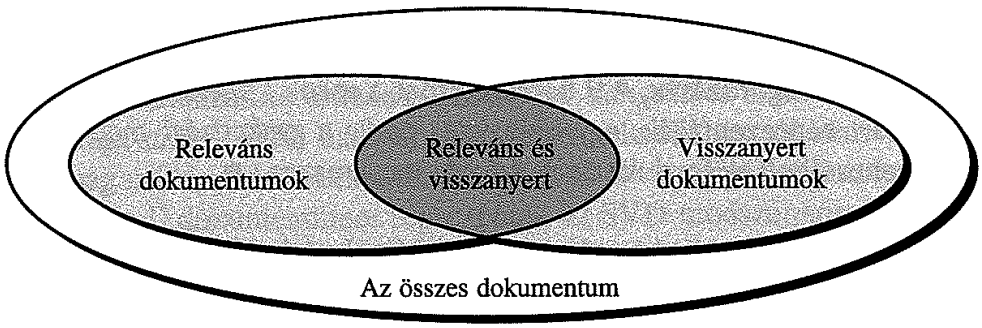
$$Pontosság = \frac{|\{Releváns\} \cap \{Visszanyert\}|}{|\{Visszanyert\}|}. \tag{9.4}$$

- **Teljesség** – Az összes, a lekérdezésnek ténylegesen megfelelő dokumentumban a helyesen visszanyertek százalékos aránya:

$$Teljesség = \frac{|\{Releváns\} \cap \{Visszanyert\}|}{|\{Releváns\}|}. \tag{9.5}$$

Kulcsszó alapú és hasonlóság alapú visszakeresés

Milyen információvisszakereső módszerek léteznek? A legtöbb információvisszakereső rendszer alkalmas a *kulcsszó alapú* és/vagy a *hasonlóság alapú visszakeresésre*. A **kulcsszó alapú információvisszakeresésnél** a dokumentum szöveges (sztring) formában van ábrázolva, és egy kulcsszócsoport segítségével ismerhető fel. A felhasználó egy kulcsszót vagy egy kulcsszavakból képzett kifejezést ad meg, például *„autó* **and** *javítóműhely"*,



9.8. ábra | A releváns és a visszanyert dokumentumcsoportok kapcsolata

„tea **or** *kávé"* vagy *„adatbázisrendszer* **but not** *Oracle"*. Egy jó információvisszakereső rendszernek az ilyen lekérdezésekre válaszolva figyelembe kell vennie a szinonimákat. Például ha adott az *autó* kulcsszó, figyelembe kell venni a keresésnél az *automobil* és *jármű* szinonimákat is. A kulcsszó alapú visszakeresés olyan egyszerű modell, ahol két fő nehézséggel találkozunk. Az első a **szinonima probléma**: egy kulcsszó, például *szoftvertermék*, annak ellenére, hogy a dokumentum szorosan kapcsolódik a *szoftvertermék*hez, nem is fordul elő a dokumentumban. A második a **többértelműség problémája**: ugyanaz a kulcsszó, például a *bányászat*, különböző összefüggésben különböző jelentéssel bír.

A **hasonlóság alapú visszakeresés** a hasonló dokumentumokat egy közös kulcsszócsoport alapján keresi. Egy ilyen visszakeresés kimenetének a *relevancia mértékén* kell alapulnia, ahol a relevancia mértéke a kulcsszavak közelsége, viszonylagos gyakorisága stb. alapján kerül megállapításra. Megjegyezzük, hogy a relevancia pontos mértékét sok esetben nehéz megadni egy kulcsszócsoportban, például az *„adatbányászat"* és az *„adatelemzés"* közötti távolság értékét.

Hogyan működik a kulcsszó alapú és a hasonlóság alapú visszakereső rendszer? A szöveges visszakereső rendszerben a dokumentumok egy csoportjához gyakran egy *tiltólista* van hozzárendelve. A **tiltólista** az „irrelevánsnak" tartott szavak gyűjteménye. Például az *egy, az, és, illetve, ahol* és így tovább **tiltott szavak**, még akkor is, ha gyakran fordulnak elő. A tiltólisták dokumentumcsoportról dokumentumcsoportra változhatnak. Például egy újságban az „adatbázisok" fontos kulcsszó lehet. De egy adatbázis-konferencia tudományos dolgozataiban kizárt szónak lehet tekinteni.

Különböző szavak egy csoportja közös **szótővel** rendelkezhet. Egy szöveges visszakereső rendszernek fel kell tárnia olyan szócsoportokat, amelyekben a szavak szintaktikusan kis mértékben térnek el egymástól, és egy csoporton belül a közös szótövű szavakat kell összegyűjtenie. Például, *kábítószerezik, elkábult, kábítószerek* közös szótővel rendelkeznek, ugyannak a szónak a különböző megjelenési formáiként tekinthetők.

Hogyan modellezhető az információvisszakeresés céljából egy dokumentum? Egy d dokumentumokból álló dokumentumcsoport és egy t szóból álló szócsoportból kiindulva minden egyes dokumentumot a t dimenziós $\mathfrak{R}^t$ térben v vektoraként modellezhetünk. A v-nek a j-ik koordinátája a j-ik szónak az adott dokumentumhoz való kötésének számszerű mértéke: általában ha a dokumentum nem tartalmazza a szót, akkor az értéke nulla, különben pedig nem nulla. Sokféleképpen lehet meghatározni a szósúlyozást a vektor nem nulla bejegyzései számára. Például, egyszerűen $v_j = 1$ értéket adhatunk, ha a j-ik szó előfordul a dokumentumban, vagy v_j lehet a **szógyakoriság**, azaz, hányszor fordul elő a t_j szó a dokumentumban, vagy a **relatív szógyakoriság**, vagyis a szó előfordulási száma az összes szó összesített előfordulási számához viszonyítva.

9.11. példa | Szógyakorisági mátrix. A 9.5. táblázatban a sorok a szavakat, az oszlopok a dokumentumvektort reprezentálják, egy bejegyzés pedig a *gyakorisági_mátrix*(i, j) eleme, a t_i szó d_j dokumentumbeli előfordulásainak számát tartalmazza. ■

Hogyan dönthető el két dokumentumról, hogy hasonlóak? Mivel várható, hogy a hasonló dokumentumok hasonló relatív szógyakorisággal rendelkeznek, egy dokumentumcsoporton belüli, vagy egy dokumentum és egy lekérdezés (amit gyakran egy kulcsszócsoporttal

9.5. táblázat | Szógyakorisági mátrix, amely mutatja a szavak dokumentumonkénti gyakoriságát

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
t_1	321	84	31	68	72	15	430
t_2	354	91	71	56	82	6	392
t_3	15	32	167	46	289	225	17
t_4	22	143	72	203	51	15	54
t_5	74	87	85	92	25	54	121

azonosítunk) közötti hasonlóságot a gyakorisági táblában megtalálható hasonló relatív szóelőfordulások alapján értékelhetjük ki.

A dokumentum hasonlóság szempontjából történő kiértékelésekor sok különböző mérték is szóba került. Egy tipikus mérték az ún. **koszinuszmérték**, amit a következőképpen határoztak meg. Legyen v_1 és v_2 két dokumentumvektor. A koszinuszhasonlóságukat a következő képlet adja meg:

$$sim(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|} \tag{9.6}$$

ahol a $v_1 \times v_2$ szorzat a szokásos skalárszorzat: $\sum_{i=1}^{t} v_{1_i} \cdot v_{2_i}$, a nevezőben levő norma pedig $|v_1| = \sqrt{v_1 \cdot v_2}$.

Hogyan használhatjuk fel a hasonlóság mértékét? A dokumentumok hasonlósági mértékének mérőszámai segítségével az ilyen dokumentumok hasonlóság alapú indexét hozhatjuk létre. A szöveges alapú lekérdezéseket vektorként tekinthetjük, amit a dokumentumgyűjteményben a legközelebbi szomszédok kikeresésére használhatunk fel. Viszont bármely fontos dokumentum-adatbázis esetén általában a szavak száma, T és a dokumentumok száma, D elég magas. Az ilyen magas dimenziószám alacsony hatékonyságú számításhoz vezet, mivel a kapott gyakorisági tábla mérete $T \times D$. Ráadásul a magas dimenziószám alacsony feltöltöttségű vektort eredményez, és megnehezíti a szavak közötti kapcsolatok (szinonimitás) felismerését és feltárását. Ezen problémák kiküszöbölése érdekében fejlesztették ki a *rejtett szemantikus indexelés* módszerét, ami hatékonyan csökkenti az elemzés számára létrehozott gyakorisági tábla méretét.

Rejtett szemantikus indexelés

Hogyan csökkenti a rejtett szemantikus indexelés a szógyakorisági mátrix méretét? A **rejtett szemantikus indexelési módszer** *szinguláris érték dekompozíciót (singular value decomposition, SVD)* használ. Ez a mátrixelméletben jól ismert technika, amivel a gyakoriság mátrix méretét csökkenthetjük. Adva van egy $T \times D$ szógyakoriság mátrix, amely T szót és D dokumentumot reprezentál, az SVD-módszer sorokat és oszlopokat távolít el, hogy $K \times K$-ra csökkentse a mátrix méretét, ahol K értékét a nagy dokumentumgyűjtemények esetében általában néhány százra (például 200) választják. Az információveszteség csökkentése érdekében a gyakorisági mátrixnak csak a legkevésbé lényeges részeit hagy-

ják el. Az SVD-mátrix transzformációja meglehetősen bonyolult, és túlmutat a fejezet keretein. Viszont szoftvercsomagokban ingyenesen érhetők el az ismert SVD-algoritmusok – például MATLAB *(www.mathworks.com)* és LAPACK *(www.netlib.org/lapack++)*.

A rejtett szemantikus indexelés módszere általában a következő alapvető lépéseket tartalmazza:

(1) A gyakorisági mátrix megvalósítása: *gyakorisági_mátrix*.
(2) A *gyakorisági_mátrix* szinguláris érték dekompozíciójának kiszámítása a mátrix három kisebb mátrixra – U, S, V – darabolásával, ahol U és V ortogonális mátrixok ($U^TU = I$), az S pedig diagonális mátrix szinguláris értékekkel. Az S mátrix mérete $K \times K$ és ez az eredeti gyakorisági mátrix redukált változata.
(3) Minden egyes d dokumentum dokumentumvektorának lecserélése olyanra, amelyből hiányoznak az SVD által eltávolított szavak.
(4) Az összes vektort tartalmazó csoport tárolása, és a fejlett multidimenziós indexelési technikákkal indexeik létrehozása.

A szinguláris érték dekompozíció és a multidimenziós indexelés segítségével a transzformált dokumentumvektorokat fel lehet használni két dokumentum közötti hasonlósági fok összevetésére vagy egy lekérdezés első N illeszkedésének megtalálására.

Egyéb indexelési technikák a szöveg-visszakeresésben

Van néhány népszerű indexelési technika a szöveg-visszakeresésben, beleértve az invertált indexeléseket és a szignófájlokat.

Az **inverz index** olyan indexszerkezet, amely két tördelő-indexelt vagy B+-fa indexelt táblát kezel, ezek a *dokumentum_ tábla* és a *szó_ tábla*, ahol

- a *dokumentum_tábla* két mezővel bíró dokumentumrekordokból áll. A mezők: *dok_id* (dokumentumazonosító), *hivatkozás_lista* (a dokumentumban előforduló valamilyen relevanciamérték szerint rendezett szavak vagy azokra mutató pointerek) listája;
- *a szó_tábla* kétmezős szórekordok halmaza. A mezők *szó_id* (szóazonosító), *hivatkozás_lista* (azon dokumentumok azonosító listája, amelyekben előfordul a szó).

Ilyen szervezés mellett könnyű olyan lekérdezésekre válaszolni, mint *„Keresd ki azokat a dokumentumokat, amelyek kapcsolatba hozhatók az adott szócsoporttal"* vagy *„Keresd ki azokat a szavakat, amelyek kapcsolatba hozhatók az adott dokumentumcsoporttal"*. Például, ahhoz hogy kikeressük az adott szócsoporttal kapcsolatba hozható dokumentumokat, először a *szó_táblából* minden szóhoz kigyűjtjük a dokumentumazonosítók listáját, majd e listák metszetét képezve megkapjuk a releváns dokumentumok listáját. A gyakorlatban széles körben használatos az inverz index. Könnyű létrehozni, viszont nem kezeli kielégítően a szinonimitást és a többértelműséget. A *hivatkozás_lista* elég nagyméretű is lehet, amihez nagy tárterület szükséges.

A **szignófájl** olyan fájl, amely dokumentumonként egy szignórekordot tartalmaz. A szignórekord b bit méretű, ahol mindegyik bit egy szót reprezentál. Egy egyszerű kódo-

lási séma a következő: egy dokumentum szignójának mindegyik bitje 0 kezdőértéket kap. A bit értéke akkor lesz 1, ha a dokumentumban előfordul a bit által képviselt szó. Az S_1 szignó akkor illeszkedik egy másik, S_2 szignóhoz, ha minden bit, amely az S_2-ben 1 értékű, az S_1-ben is 1 értéket kap. Mivel általában több szó van, mint ahány bit rendelkezésre áll, ugyanahhoz a bithez több szó is hozzá lehet rendelve. Az ilyen, sok az egyhez bitösszerendelés költségessé teszi a keresést, mivel egy dokumentum, amelynek a szignója megegyezik a lekérdezés szignójával, nem feltétlenül tartalmazza a lekérdezés összes kulcsszavát. A dokumentumot elő kell venni, gyakorisági elemezést kell végezni, le kell vágni a szavak toldalékait, és ellenőrizni kell. Javulást lehet elérni először gyakorisági elemzés, a szótőkeresés, a tiltott szavak kiszűrésének elvégzésével, ezután a tördelőtechnika alkalmazása és a fölé helyezett – a szólistát bit reprezentációba kódoló – kódolási technika segítségével. Mindezek ellenére a *sok az egyhez bitösszerendelés* továbbra is megmarad, ami a módszer legnagyobb hátránya.

9.5.2. | Szövegbányászat: kulcsszó alapú társítás és dokumentumosztályozás

Mi a helyzet a szöveges adatbázisokban a társítások bányászatával? Tudunk-e generálni dokumentumosztályozási sémákat? Az alfejezet ezzel a két kérdéssel foglalkozik.

Kulcsszó alapú társításelemzés

Mi a kulcsszó alapú társításelemzés? Az ilyen analízis összegyűjti azokat a kulcsszavakat, melyek gyakran együtt fordulnak elő, és feltárja a közöttük lévő kapcsolatokat és korrelációs viszonyokat.

Mint a legtöbb szöveges adatbázison végzett analízis, a társításelemzés először – elemzéssel, szótőkereséssel, a tiltószavak eltávolításával stb. – előfeldolgozást végez, és ezután indítja a társításbányászati algoritmusokat. Egy dokumentumokat tartalmazó adatbázisban minden egyes dokumentum egy tranzakciónak tekinthető, mivel a kulcsszavakból álló halmazt a tranzakció tételcsoportjaként lehet tekinteni. Így az adatbázis formája a következő:

$\{dokument\_id, kucsszavak\_csoportja\}$.

A dokumentum-adatbázisokban a kulcsszótársítás kérdése ily módon a tranzakciós adatbázisban lévő tételtársítás-bányászattal hozható kapcsolatba, ahol már sok érdekes módszert fejlesztettek ki, ahogy ez a 6. fejezetben látható.

Észrevehetjük, hogy a nagy gyakorisággal szorosan egymás után előforduló vagy egymáshoz közel lévő kulcsszavak szókapcsolatot vagy kifejezést alkothatnak. A társításbányászati tevékenység segít az *összetett kapcsolatok* felismerésében, mint amilyenek a tartományfüggő szótkapcsolatok vagy kifejezések, például [ELTE, Egyetem], vagy [Főváros, Pest, Budapest], vagy *nem összetett kapcsolatok*, például [dollár, részvény, árfolyam, végösszeg, jutalék, kockázat, kötvények].

Az ilyen kapcsolatokon alapuló bányászat a *fogalom szintű társításbányászat* (szem-

ben az egyedi szavakon történő bányászattal). Fogalomfelismerésnek és fogalom szintű társítások bányászatának a szöveges elemzésben két előnye van: (1) a fogalmakat és a kifejezéseket automatikusan határozzuk meg, így nincs szükség emberi beavatkozásra; (2) az értelmetlen találatok száma jelentősen csökken, így a bányászati algoritmus végrehajtási ideje is.

Ilyen fogalom és kifejezés felismerés segítségével fogalom szintű bányászat hozható létre a társítások keresésére a feltárt fogalmak és a kulcsszavak halmazában. Egyes felhasználók szívesen keresik az adott kulcsszó vagy kifejezés halmazában kulcsszó- vagy fogalompárok kapcsolatát, míg mások szeretnék megtalálni az együtt előforduló kifejezések maximális halmazát. Így, a felhasználó bányászati igénye alapján, a standard társításbányászó vagy a max-minta bányászó algoritmus vehető igénybe.

Dokumentumok osztályozása

Az automatikus dokumentumosztályozás egy fontos szövegbányászati feladat, mivel a rengeteg rendelkezésre álló on-line dokumentum miatt nehéz, de azért mégis alapvető, hogy az ilyen dokumentumokat a dokumentum-visszakeresés és további elemzés céljából képesek legyünk automatizált módon osztályokba szervezni.

Hogyan lehet az automatizált osztályozást végrehajtani? Az általános eljárás a következő: Először egy előosztályozott dokumentumcsoportot tanuló halmaznak vesszük. Ezután ezt a tanuló csoportot elemezzük, hogy kinyerjünk egy osztályozási sémát. Az osztályozási sémát egy tesztelési folyamatban gyakran finomítani kell. Az ily módon kapott osztályozási séma felhasználható más on-line dokumentumok osztályozására.

Ez az eljárás a relációs adatosztályozáshoz hasonlónak tűnik. Viszont van egy alapvető különbség. A relációs adat jól strukturált: Minden sort egy attribútum–érték pár csoport határoz meg. Például a {*napos, meleg, száraz, szélmentes, teniszezik*} sorban a *„napos"* érték az *időjárási_kilátások* attribútumnak, a *„meleg"* a *hőmérséklet* attribútumnak felel meg és így tovább. Az osztályozásanalízis meghatározza, hogy mely attribútum–érték párokból álló csoport bír a legnagyobb megkülönböztető erővel annak a kérdésnek eldöntésében, hogy valaki elmegy-e teniszezni vagy nem. Ezzel szemben a dokumentum-adatbázisok strukturáltsága nincs az attribútum–érték pároknak megfelelő strukturáltságuk. Azaz, egy dokumentumcsoporttal kapcsolatos kulcsszócsoport nincs rögzített attribútum- vagy dimenziócsoportba szervezve. Következésképpen, az általánosan használt relációs adatorientált osztályozási módszereket – mint döntésifa-analízis – nem lehet használni dokumentum-adatbázisok osztályozására.

A dokumentumosztályozás egyik hatékony módszere a társítás alapú osztályozás, ami a dokumentumokat a gyakran előforduló, egymással kapcsolatban levő szövegminták csoportja alapján osztályozza. A társítás alapú osztályozási módszer a következőképpen zajlik: először, a kulcsszavakat és kifejezéseket információvisszakereső és egyszerű társításelemző technikákkal kinyerjük. Másodszor, a rendelkezésre álló szóosztályok – például a WordNet – felhasználásával, szakértői ismeretekre vagy valamely kulcsszó-osztályozási rendszerre támaszkodva a kulcsszavak és a szavak fogalmi hierarchiáit kapjuk meg. A tanuló dokumentumcsoport dokumentumai ugyancsak osztályhierarchiába rendezhetők. Ezután szókapcsolat-bányászati módszer alkalmazható, amellyel feltárjuk a kapcsolatban levő szavak csoportjait, melyek segítségével a legélesebben elkülöníthetünk egy doku-

mentumosztályt a többitől. Ez minden egyes dokumentumosztályhoz származtat egy társítási szabálycsoportot. Ezek az osztályzási szabályok az előfordulási gyakoriságuk, megkülönböztető erejük szerint rendezhetők, és felhasználhatók új dokumentumok osztályozására. Az ilyen típusú társítás alapú dokumentumosztályozók hatékonyaknak bizonyultak. A webdokumentumok osztályozása számára a weblaplink-analízis használható a dokumentumosztályok megállapításának további segítésére. A weblaplink-analízis módszereit a következő alfejezet tárgyalja.

9.6. | A World Wide Web bányászata

A World Wide Web hatalmas, széles körben osztott, globális információszolgáltató központ, amely híreket, hirdetéseket, felhasználói információkat, pénzügyi, oktatási, kormányzati, elektronikus kereskedelmi és más egyéb információt szolgáltat. A web ezenfelül tartalmazza még a hiperlink információk gazdag és dinamikus gyűjteményét, továbbá a weblapok elérési és felhasználási információit, így az adatbányászat számára gazdag információforrást nyújt. Viszont a következő megjegyzések alapján a web a hatékony erőforrás- és tudásfeltárás területén még nagy feladatokat állít elénk.

- *Úgy tűnik, hogy a web túl nagy a hatékony adattárház és adatbányászat céljaira.* A web százas nagyságrendű terabájt méretű, és még továbbra is gyorsan növekszik. Sok szervezet és társaság a közérdekű adatai nagy részét a webre helyezi. Aligha lehetséges olyan adattárház felállítása, amely a web összes adatát tükrözi (replikálja), tárolja vagy integrálja.[5]
- *A weblapok sokkal bonyolultabbak, mint bármely hagyományos szöveges dokumentumgyűjtemény.* A weblapoknál hiányzik az egységes szerkezet. Sokkal bővebb szerzői stílussal és tartalmi változatossággal rendelkeznek, mint bármely könyvkészlet vagy egyéb hagyományos szöveges alapú dokumentum. A webet óriási digitális könyvtárnak tekintjük, de ebben a könyvtárban ez a hatalmas mennyiségű dokumentum semmilyen szempont szerint sincs rendezve. Kategória, cím, fedőlap, tartalomjegyzék stb. egyike szerint sem létezik indexelés. Esetenként nagy kihívás lehet a kívánt információ keresése.
- *A web nagyfokú dinamizmussal rendelkező információs forrás.* A web nemcsak hogy gyors léptekkel növekszik, hanem a benne elérhető információ is állandóan frissítésre kerül. A hírlapok, a tőzsdék, a hirdetőcégek és a webszolgáltatók rendszeresen frissítik weblapjaikat. A linkinformációk és a hozzáférési bejegyzések szintén gyakran változnak.
- *A web a felhasználói csoportok széles skáláját szolgálja ki.* Az internethez jelenleg körülbelül 50 millió munkaállomás csatlakozik, és a felhasználói csoportok is gyorsan bővülnek. A felhasználók jelentősen eltérő háttérrel, érdeklődési körrel, felhasználói szándékkal rendelkeznek. A legtöbb felhasználónak nincs megfelelő ismerete az információs hálózat felépítéséről, nincs tisztában egy-egy keresés magas költségével, könnyen

5 | Korábban voltak a web összes adatának tárolására vagy integrálására irányuló kísérletek. Például, egy tíz terabájt nagyságrendű internetarchívum érhető el a *http://www.archive.org/indexl.html* oldalon.

eltévedhet a hálózat „sötétjében" tapogatózva, megunhatja a sok „ugrást" az interneten és a türelmetlen várakozást egy kis információra.

- *A weben lévő információnak csak egy kis része valóban releváns vagy hasznos.* Azt mondják, hogy a webinformáció 99%-a a webhasználók 99%-a számára haszontalan. Habár nem magától értetődő, de igaz, hogy egy konkrét személyt általában a web egy kis része érdekli, míg a web többi része érdektelen számára, és tönkreteheti a keresés kívánt eredményét. Hogyan lehet a web érdeklődési körünknek megfelelő területét lokalizálni? Hogy tudunk egy meghatározott témához minőségi weblapokat megtalálni?

Ezek a kihívások mozdították elő az interneten lévő erőforrások hatékony és hatásos feltárását és használatát célzó kutatást.

Sok index alapú **webkeresőmotor** létezik, amelyek keresést végeznek a weben, weblapokat indexelnek, nagyméretű kulcsszó alapú indexeket építenek fel és tárolnak, amelyek segítenek a megadott kulcsszavakat tartalmazó weblapcsoportok lokalizálásában. Ilyen keresések segítségével a gyakorlott felhasználó szorosan körülhatárolt kulcsszavak vagy kifejezések megadása mellett gyorsan megtalálja a dokumentumokat. Viszont a jelenlegi kulcsszó alapú keresőmotoroknak van néhány hiányossága. Először, egy tetszőlegesen széles körű témával akár a dokumentumok százai vagy ezrei foglalkozhatnak. Ez oda vezethet, hogy a keresőmotor óriási mennyiségű dokumentumot ad vissza, amelyek nagy része a téma szempontjából érintőlegesen releváns vagy rossz minőségű anyagot tartalmaz. Másodszor, sok dokumentum, amely a témával kapcsolatosan jelentős relevanciával rendelkezik, esetleg nem tartalmazza a keresési kulcsszót. Ez a *többértelműség* problémája, amit a szövegbányászattal foglalkozó előző alfejezetben már tárgyaltunk. Például az *adatbányászat* kulcsszó kihozhat sok, egyéb bányászati iparra vonatkozó weblapot, viszont nem találja a tudáskeresés, statisztikai analízis vagy a gépi tanulás terén releváns dolgozatokat, mivel ezek nem tartalmazzák az *adatbányászat* kulcsszót. Egy másik példa: a *keresőmotor* kulcsszó alapján végzett keresés még az olyan népszerű webkeresőmotorokat sem találja meg, mint a Yahoo!, AltaVista vagy az America On-Line, ha ezek a szolgáltatások a weblapjukon nem nevezik magukat „keresőmotornak". Ez azt mutatja, hogy a jelenlegi webkeresőmotorok nem elégségesek a web erőforrások feltárására.

Ha a webkeresőmotorok nem elégségesek a weberőforrások feltárására, akkor hogyan gondolhatunk a webbányászat végrehajtására? Még izgalmasabb kihívás a webelérési mintákat, webstruktúrákat, webtartalmak szabályosságát és dinamikáját kereső **webbányászat**. A webbányászati feladatok általában három kategóriába sorolhatók: *webtartalom-bányászat*, *webstruktúra-bányászat* és *webhasználat-bányászat*. A webstruktúrákat a webtartalom részeként lehet kezelni, így a webbányászatot a fenti besorolás helyett a webtartalom-bányászat és a webhasználat-bányászat osztályokba lehet besorolni.

A további alfejezetekben a webbányászatot érintő néhány fontos témával foglalkozunk: a weblinkstruktúra bányászata (9.6.2. alfejezet), a webdokumentumok automatikus osztályozása (9.6.2. alfejezet), többrétegű webinformációs bázis építése (9.6.2. alfejezet), és a webnapló-bányászat (9.6.2. alfejezet).

9.6.1. | A weblink struktúráinak bányászata az autentikus weblapok lokalizálására

Mit értünk az autentikus weblapok alatt? Tegyük fel, hogy keresni akarunk adott témával – például pénzügyi befektetéssel – foglalkozó weblapokat. A releváns lapok megtalálása mellett reméljük, hogy a kapott lapok a téma szempontjából jó minőségűek, vagyis *autentikusak.*

De hogyan tudja egy keresőmotor az adott téma szempontjából autentikus weblapokat megkeresni? Érdekes módon, az autentikusság titka a weblapok kapcsolataiban rejlik. A web nemcsak lapokat, hanem az egyik lapról a másikra mutató *hiperhivatkozásokat* (hiperlinkeket) is tartalmaz. Ezek a hiperhivatkozások óriási mennyiségű rejtett emberi megjegyzést tartalmaznak, ami segíthet az autentikus fogalom automatikus kialakításában. Amikor egy weblaptervező a saját weblapján más weblapokra mutató hiperlinkeket helyez el, ezt úgy lehet tekinteni, hogy azt a weblapot (hivatkozásával) megerősíti. Egy adott lap ilyen, különböző weblaptervezők általi kollektív megerősítése fokmérője lehet a lap fontosságának, és természetes módon vezethet az autentikus weblapok feltárásához. Ily módon, a hatalmas mennyiségű weblink információ gazdag anyagot szolgáltat a webtartalom relevanciájáról, minőségéről és szerkezetéről, és így a webbányászat számára gazdag forrás.

Ez az elgondolás néhány érdekes, a web autentikus lapjainak bányászatával foglalkozó tanulmányt eredményezett. Az 1970-es években az információ-visszakereséssel foglalkozó kutatók javasolták, hogy a folyóiratokban megjelenő tudományos publikációk minőségét idézetek (hivatkozások) alapján értékeljék. Szemben a folyóirathivatkozásokkal, a web linkfelépítésének sajátos jellege van. Először, számunkra nem minden hiperlink érdemleges. Egyes linkek más – nem hivatkozási – célokat szolgálnak, például navigálásra vagy fizetett hirdetések elhelyezésére. Általában viszont, ha a hiperlinkek többsége érdemleges, akkor érvényesül a „kollektív vélemény". Másodszor, a kereskedelmi vagy a versenyszférában egy hirdető sem fog a saját weblapjáról a versenytársra hivatkozni. Például, a *CocaCola* nem helyez el *PepsiCola*-ra mutató linket, így nem nyújt támogatást a konkurenciának. Harmadszor, az autentikus lapok nem különösen beszédesek. Például, a Yahoo! honlapjának nem szükséges explicit módon tartalmaznia a „webkereső" önmeghatározást.

A weblink felépítésének ilyen tulajdonságai azt eredményezték, hogy a kutatók a weblapok egy másik fontos kategóriáját vezették be, az ún. *csomópontot*. A **csomópont** egy weblap vagy több weblapból álló csoport, amely az autentikus helyekre mutató hivatkozásokat fogja össze. A csomópontlapok önmagukban nem érdemiek, vagy kevés link mutat rájuk, viszont olyan hivatkozásokat biztosítanak, amelyek egy általános témában fontos weblapcsoport tagjaira mutatnak. Az ilyen lapok az egyes honlapokra mutató, ajánlott hivatkozások listái lehetnek, mint amilyenek egy tanfolyam honlapján ajánlott referencia helyek, vagy szakszerűen összeállított forráslisták egy kereskedelmi webhelyen. A csomóponti lap szerepe, hogy természetes módon tájékoztasson az autentikus lapokról egy érdekelt témában. Általában, egy jó csomóponti lap sok jó autentikus lapra mutat, egy jó autentikus lapra pedig sok jó csomópontról mutatnak. A csomópontok és az autentikus lapok ilyen, egymás kölcsönös megerősítése segít az autentikus weblapok bányászásában és a jó minőségű webszerkezetek és erőforrások feltárásában.

Tehát, hogyan használhatjuk a csomóponti lapokat az autentikus lapok előkeresésére? Egy, a csomópontokat felhasználó algoritmus, az ún. HITS (Hiperlink Induced Topic Search, hiperlink alapú témakeresés) kifejlesztése a következőképpen történt.

Először, a HITS lekérdező szavakat használ arra, hogy egy index alapú keresőmotorról egy kiinduló készletet gyűjtsön össze, mondjuk olyat, ami 200 weblapból áll. Ezek a lapok alkotják a **gyökérkészletet**. Mivel e lapok közül sokról feltehető, hogy a keresett témában releváns, némelyiknek tartalmaznia kell egy linket a kiemelkedően releváns weblapok legtöbbjéhez. Ennek következtében a gyökérkészletet ki lehet bővíteni a megadott mérethatárig, például 1000-től 5000 lapig, egy **alapkészletté** oly módon, hogy bevesszük az összes olyan lapot, amelyre a gyökérkészlet lapjai mutatnak, és az összes olyan lapot, amelyek a gyökérkészlet lapjaira mutatnak.

Másodszor, elindul egy súlyterjedési fázis. Ez egy iteratív folyamat, amelynek során meghatározzuk a csomóponti és az autentikus súlyok becsléseit. Megjegyezzük, hogy mivel két azonos webtartományon (domain) levő lap (az URL-jeik első szintje azonos) közötti linkek gyakran navigációs funkcióként szolgálnak, ezért ezek nem erősítik az autentikusságot, így az ilyen linkek ki lesznek zárva a súlyterjedésből.

Elsőként az alapkészlet minden p lapjához hozzárendelünk egy nem negatív a_p **autentikus súlyértéket**, egy nem negatív h_p **csomóponti súlyértéket**, és az összes a, valamint h egy egységes konstans kezdőértéket kap. Az értékeket normalizáljuk, veszünk egy olyan invariánst, ami szerint az összes súly négyzetösszege 1 lesz. Az autentikus és a csomóponti súlyokat a következő egyenletek szerint újraszámoljuk:

$$a_p = \sum_{(q:q \to p)} h_q, \tag{9.7}$$

$$h_p = \sum_{(q:q \leftarrow p)} a_q. \tag{9.8}$$

A (9.7) egyenlet az sugallja, hogy ha egy lapra sok jó csomópont mutat, akkor az autentikus súlyát meg kell növelni (ez a jelenleg a lapra mutató összes lap csomóponti súlyainak összege). A (9.8) egyenlet az sugallja, hogy ha egy lap sok jó autentikus lapra mutat, akkor a csomóponti súlyát meg kell növelni (ez azon lapok autentikus súlyértékeinek összege, amelyekre a lapunk mutat).

Ezeket az egyenleteket a következő módon írhatjuk mátrix alakba. Számozzuk be a lapokat $\{1, 2, \ldots, n\}$, határozzuk meg az $n \times n$ méretű **szomszédsági mátrixot**, A-t. $A(i, j) = 1$, ha az i lap a j lapra mutató linket tartalmaz, különben az értéke 0. Hasonlóképpen, megadjuk az **autentikus súly vektort** $a = (a_1, a_2, \ldots a_n)$, és a **csomóponti súly vektort** $h = (h_1, h_2, \ldots h_n)$. Így a következő egyenleteket kapjuk:

$$h = A \times a, \tag{9.9}$$

$$a = A^T \times h, \tag{9.10}$$

ahol A^T az A mátrix transzponáltja. Ezt a két egyenletet k-szor kibontva:

$$h = A \times a = AA^T h = (AA^T)h = (AA^T)^2 h = \ldots = (AA^T)^k h, \tag{9.11}$$

$$a = A^T h = A^T A a = (A^T A)a = (A^T A)^2 a = \ldots = (A^T A)^k a. \tag{9.12}$$

A lineáris algebrának megfelelően, ez a két iteráció-szekvencia normalizálás után egyenként konvergálnak az AA^T és az A^TA legnagyobb sajátértékhez tartozó sajátvektoraihoz. Ez is azt igazolja, hogy az autentikus és a csomópont súlyok az összegyűjtött, hivatkozási kapcsolatban lévő lapok belső, lényegi tulajdonságai, amit nem befolyásol a súlyok kezdeti értékkiosztása.

Végül, a HITS-algoritmus kiadja az adott keresési témához tartozó nagy csomópont súlyú és nagy autentikusság súlyú lapok rövid listáját. Sok kísérlet megmutatta már, hogy a HITS a kérések széles skáláján meglepően jó keresési eredményt ad.

Habár a linkekre széles körben építve biztató eredmények érhetők el, a szöveges tartalom figyelmen kívül hagyása miatt szembekerülhetünk néhány problémával. Például a HITS néha, ha a csomópont több témát tartalmaz, *témasodródáshoz* vezet. Témasodródást okozhat az is, ha egy webhelyről sok lap mutat ugyanarra a népszerű webhelyre, így a webhely túl nagy részarányú autentikus értéket kap. Az ilyen problémákat úgy lehet megoldani, hogy a (9.7) és a (9.8) egyenletekben az összegeket súlyozott összegekre cseréljük, ahol leértékeljük az egy webhelyről mutató többszörös hivatkozások súlyát. E célból felhasználjuk az ún. horgony szöveget (a weblapon a hiperlinket körülvevő szöveg), hogy beállítsuk az autentikusságot továbbító hivatkozások súlyát, és a nagy csomóponti lapokat kisebb egységekre daraboljuk fel.

A HITS-algorimuson alapuló rendszer a Clever. A Google is hasonló elven működik. A beszámolók szerint az olyan rendszerek, amelyek egyaránt elemzik a weblinkeket és a tartalmi információkat, jobb keresési eredményeket érhetnek el, mint azok, amelyek a szóindex alapján működnek, például az AltaVista és azok, amelyeket ontológusok készítettek, például a Yahoo!.

9.6.2. | Webdokumentumok automatikus osztályozása

Az webdokumentumok automatikus osztályozásakor minden dokumentum egy előre megadott kategóriakészletből az előre osztályozott dokumentumok példakészlete alapján kap egy osztálycímkét. Például hogy megkapjuk a webdokumentum-osztályozási sémát, a Yahoo! osztályozása és a hozzá kapcsolódó dokumentumok kísérleti és tesztcsoportként használhatók fel. Ez a séma ezután az ugyanabból az osztályozásból vett kategóriák hozzárendelésével új dokumentumok osztályozására használható fel.

Kulcsszó alapú dokumentumosztályozási módszereket, valamint a kulcsszó alapú dokumentumtársítás-analízist a 9.5.2. alfejezetben tárgyaltuk. Ezek a módszerek felhasználhatók a webdokumentumok osztályozására. Egy ilyen szó alapú osztályozási séma a weblaposztályozásnál jó eredményeket mutatott. Mivel a hiperhivatkozások a lapok témáihoz jó minőségű szemantikus kulcsokat tartalmaznak, előnyös az ilyen szemantikus információt a tisztán kulcsszó alapú osztályozásnál a még nagyobb pontosság elérésére felhasználni. Viszont a dokumentumot körülvevő hiperlinkek elég zajosak lehetnek, ezért a dokumentum a hiperlinkjét körülvevő szavak naiv felhasználása a pontosságot még le is ronthatja. A robusztus statisztikai modellek – mint a Markov véletlen mezők (Markov random fields, MRF) – együttes alkalmazása a feloldó (relaxáló) címkézéssel segít a helyzeten. Kísérletileg kimutatott, hogy egy ilyen módszer alapvetően megjavítja a webdokumentum-osztályozás pontosságát.

9.6.3. | Többrétegű webinformációs bázis létrehozása

Ahogy arról a 2. fejezetben szó volt, egy relációs adatbázisból többdimenziós és hierarchikus adatatokat biztosító adattárház alakítható ki.

Ki lehet-e alakítani egy, a web többdimenziós, hierarchikus nézetét biztosító többrétegű webinformációs bázist? Hogy megtudjuk, hogy ez reális-e vagy előnyös-e egyáltalán, próbáljunk megtervezni egy ilyen többrétegű webinformációs bázist.

Először, olyan webadattárház létrehozásának realitása, amely a web összes lapjáról másolatot tárol, hiszen ekkor a WWW egy óriási másolatához jutunk. Ez azt mutatja, hogy egy ilyen többrétegű webinformációbázis legalacsonyabb (legrészletesebb) rétegének magának a webnek kellene lennie. Ez a réteg nem lehet különálló adattárház. Ez lesz a *0. réteg*.

Másodszor, az *1. réteget weblapleíró rétegként* definiáljuk, amely a weblapleíró információt tárolja. Ennélfogva az *1. réteg* a *0. réteg* absztrakciója. Ez lényegesen kisebb, mint a *0. réteg*, de elég bő ahhoz, hogy a kulcsszó alapú vagy a többdimenziós keresés vagy bányászat számára megőrizze az érdeklődésre számot tartó, általános információ többségét.

A weblaptartalom változatossága alapján az *1. réteg* néhány tucat félig strukturált osztályba szervezhető, mint például *dokumentum, személy, szervezet, hirdetés, címtár, vevők, szoftver, játék, értékpapír, könyvtár_katalógus, földrajzi_adat, tudományos_adat* és így tovább. Például a *dokumentum* osztályt a következőképp adhatjuk meg:

dokumentum(*fájl_cím, dokumentum_kategória, fontossági_fok, kulcsszavak, szerzők, cím, publikáció, publikációs_dátum, kivonat, nyelv, tartalomjegyzék, kategória_leírás, index, linkek_kifelé, csatolt_multimédia, oldalszám, forma, dok_méret, időbélyeg, ..., hozzáférési_gyakoriság*),

ahol minden bejegyzés a *dokumentum* weblap absztrakciója. Az első attribútum, *fájl_cím* megadja a fájlnevet és az URL hálózati címet. A *dokumentum_kategória* és a *fontossági_fok* fontos információt tartalmaznak, amit az előző két alfejezetben tárgyalt weblinkanalízis és a dokumentumosztályozás módszerekkel lehet kinyerni. Sok attribútum tartalmaz a dokumentumra vonatkozó fontosabb szemantikus információt. Ilyenek például a *kulcsszavak, szerzők, cím, publikáció, publikációs_dátum, kivonat, nyelv, tartalomjegyzék, index, linkek_kifelé, csatolt_multimédia, oldalszám.* Más attribútumok formátuminformációt írnak le, például a *forma,* amely a fájl formátumát adja meg (.ps, .pdf, .tex, .html, text, tömörített, uu-kódolt stb.). Néhány attribútum a fájllal közvetlenül kapcsolatos információt ír le, például a *dok_méret* (a dokfájl mérete) és az *időbélyeg* (az utolsó módosítás ideje). A *hozzáférési_gyakoriság* attribútum megadja, hogy milyen gyakran fordulnak a dokumentumhoz.

Harmadszor, többdimenziós, alkalmazásspecifikus szolgáltatások biztosítása érdekében az *1. réteg* felett különböző, *magasabb rétegű* webkatalógusok szolgáltatásai hozhatók létre. Például, az adatbázisrendszer területen folyó kutatások számára szaknévsor jellegű szolgáltatást szervezhetünk. Egy ilyen címtár néhány dimenziós hierarchikus struktúrát tartalmazhat, mint például témakategória, földrajzi hely, publikációs dátum és így tovább.

Tényleg minden weblapról akarunk információt gyűjteni? A weblaprangsoroló és a lap- vagy dokumentumosztályozó szolgáltatások igénybevételével módunk van arra, hogy csak azokat az információkat válasszuk ki az információbázis *1. rétegben* és/vagy a magasabb rétegekben, amelyek a viszonylag jó minőségű és nagy relevanciával bíró weblapokhoz szükségesek.

Várható, hogy a strukturált weblapleíró nyelv, az XML népszerűvé válásával a jövőben egyre több weblap készül XML nyelven, és amelyek közösen használnak lehetőleg egy jó DTD (Docoment Type Declaration, dokumentumtípus leírás) készletet.[6] A nyelvi szabványosítás – például az XML-lel – nagymértékben támogatni fogja a különböző webhelyek közötti információcserét, és a többrétegű webinformációbázis létrehozásához szükséges adatkinyerést. Ráadásul erre a célra web alapú információkereső és tudásfeltáró nyelveket lehet tervezni és létrehozni.

A fentieket összefoglalva, meg kell legyen az interneten történő erőforrás-feltárás, a többdimenziós analízist és az adatbányászatot megalapozó többrétegű webinformációs bázis létrehozásának lehetősége. Várható, hogy a web alapú többdimenziós analízis és adatbányászat fogják az internet alapú információszolgáltatások jelentős részét kitenni a jövőben.

9.6.4. | Webhasználat bányászata

Mi a webhasználat-bányászat? Webtartalom és weblink szerkezeteinek bányászata mellett a webbányászat másik fontos feladata a webhasználat-bányászat, ami a webnapló-bejegyzéseket bányássza abból a célból, hogy feltárja a weblapok hozzáférési mintáit. A webnapló-bejegyzések szabályosságainak analízisével és feltárásával azonosíthatjuk az elektronikus kereskedelem potenciális vásárlóit, javíthatunk a végfelhasználó számára nyújtott internetes információszolgáltatások minőségén, javíthatjuk a webszerverrendszerek teljesítményét.

Egy webszerver általában minden elérés esetén **webnapló-bejegyzést** készít. Ez tartalmazza a kért URL-t, a kérés származási helyeinek IP-címét és egy időbélyeget. A web alapú elektronikus kereskedelem szerverei számára nagyszámú naplórekord gyűlik össze. A népszerű webhelyek naponta több száz megabájt méretben rögzítenek naplórekordokat. A webnapló-adatbázisok bő információt szolgáltatnak a web dinamikájáról. Így fontos a bonyolult webnapló-bányászati technikák kifejlesztése.

A webhasználat-bányászat technikáinak kifejlesztésénél a következőket vehetjük figyelembe. Először, annak ellenére, hogy bátorító és izgalmas dolog a webnaplófájl-analízis különböző lehetséges alkalmazásait elképzelni, fontos szem előtt tartani azt, hogy az ilyen alkalmazások sikere a nagy mennyiségű nyers naplóadatokból nyerhető tudás érvényességétől és megbízhatóságától függ. Azért, hogy fontos és hasznos információt nyerjünk ki, a nyers webnapló adatokat gyakran *tisztítani, tömöríteni* és *transzformálni* kell. Ezek az előfeldolgozási módszerek elvileg hasonlítanak a 3. fejezetben tárgyalt módszerekhez, de gyakran a webnaplóhoz igazított előfeldolgozásra van szükség.

6 | A DTD egy, az elemek, attribútumok és entitások megadását szabályzó szabálykészlet vagy „nyelvtan". Megadja, hogy milyen elemek, milyen sorrendben megengedettek, hogyan lehet őket egymásba ágyazni.

Másodszor, a rendelkezésre álló URL, idő, IP-cím és a weblaptartalom segítségével a webadatbázison többdimenziós nézetet lehet létrehozni, és *többdimenziós OLAP-analízist* lehet végrehajtani ahhoz, hogy megtaláljuk az első *N* felhasználót, az első *N* elért weblapot, a legforgalmasabb időszakokat és így tovább, ami segít a potenciális ügyfelek, felhasználók, piacok és egyebek felderítésében.

Harmadszor, a társítási minták, szekvenciaminták és trendek megkeresésére webnaplórekordok *adatbányászatát* lehet elvégezni. A részletes webnaplóanalízis végrehajtásához a webnaplóminta-bányászatnál gyakran szükség van további mértékekre, hogy a felhasználói barangolásról kiegészítő információt kapjunk. Ilyen kiegészítő információ tartalmazhatja a felhasználónak webszerver-pufferben lévő weblapokon történő szörfözési szekvenciáit és így tovább.

Az ilyen webnaplófájlok felhasználásával rendszerteljesítményt elemző vizsgálatok végezhetők, így a webpufferelléssel, weblapok előolvasásával, weblapcserével, a webforgalom, felhasználói reakció és motiváció megértésével javítható a rendszerek tervezése. Például néhány tanulmány adaptív webhelyeket javasolt: a weboldalak a felhasználói elérési mintákból szerzett ismeretek alapján önmagukat javítják. A webnaplóanalízis segíthet az egyes felhasználókra szabott webszolgáltatások kialakításában.

Mivel a webnaplóadatok arról nyújtanak információt, hogy mely típusú felhasználók mely típusú weblapokat érnek el, a weblaprangsorolás, webdokumentum-osztályozás, többrétegű webinformációs bázis létrehozása céljaira a webnaplóinformációt integrálni lehet a webtartalom és a weblinkszerkezet bányászatába.

9.7. | Összefoglalás

- Rengeteg adatot tárolunk változatos, bonyolult formában, például strukturált vagy strukturálatlan adatokat, hipertextet, multimédiát. Így az adatbányászatban a **bonyolult adattípusok bányászata** – ideértve az *objektum-, tér-, multimédia-, idősor, szöveg- és webadatokat* – egyre nagyobb jelentőségre tett szert.
- Többdimenziós analízis és adatbányászat a következő módokon hajtható végre az **objektumrelációs és az objektumorientált adatbázisokon**: (1) *bonyolult objektumok osztály alapú általánosításával*, beleértve a halmazértékű, listaértékű és más bonyolult adattípusokat, osztály–alosztály hierarchiákat és osztályfelépítési hierarchiákat; (2) *objektum-adatkockák* létrehozásával; (3) *általánosítás alapú bányászattal*. Egy **tervadatbázis** a különböző absztrakciós szinten érdekes általános minták keresésére az általánosítás alapú, „oszd meg és uralkodj" szemlélettel bányászható.
- **A téradatbányászat** nem más, mint a nagy geográfiai téradatbázisokban érdekes minták feltárása. Létre lehet hozni **téradatkockákat**, amelyek térdimenziókat és mértékeket tartalmaznak. Létrehozható *tér-OLAP* a *többdimenziós téradatanalízis* elvégzéséhez. A téradatbányászat magában foglalja a *téradatleírást, osztályozást, társítást, klaszterezést* és a *tértrend-analízist és szélsőséges értékek analízisét.*
- **Multimédia-adatbányászat** a multimédia-objektumok – beleértve audio, kép, video típusú adatokat, szekvenciákat, szöveget, szövegkijelöléseket és hivatkozásokat tartalmazó hipertextadatokat – nagy gyűjteményét tároló és kezelő multimédia-adatbázisok érdekes mintáinak feltárása. A multimédia-adatbányászattal foglalkozó dolgozatok té-

mái többek között a *tartalom alapú visszakeresés és hasonlóságkeresés, általánosítás és többdimenziós analízis, osztályozás és előrejelzés-analízis* és *társításbányászat* a multimédia-adatokon.

- Egy idősor-adatbázis időben változó értékek vagy események szekvenciáit tartalmazza. Ilyenek az értékpapírpiac, üzleti tranzakció-szekvenciák, dinamikus termelési folyamatok, orvosi kezelések, weblapok elérési szekvenciái és így tovább. Az **idősor és szekvencia jellegű adatok bányászatával** foglalkozó kutatások lefedik a *trendanalízist,* az *idősor-analízisben hasonlóság keresését*, valamint az *idővel kapcsolatos adatok szekvenciaminta és ismétlődő minta bányászatát.*
- A rendelkezésre álló információ jelentős része szöveges vagy dokumentum-adatbázisokban van tárolva. Ezek az adatbázisok a dokumentumok (új cikkek, technikai jelentések, könyvek, digitális könyvtárak, elektronikus levélüzenetek, weblapok) nagy gyűjteményeiből állnak. Ily módon a szöveges adatbányászat egyre növekvő jelentőségre tett szert. A **szöveges bányászat** kutatása egy lépéssel el van maradva a kulcsszó és hasonlóság alapú információ-visszakereséstől, és olyan módszerek segítségével, mint *kulcsszó alapú társítás* és *dokumentumosztályozás* tárja fel a félig strukturált adatokból a tudást.
- A World Wide Web hatalmas, széles területre osztott, a hírek, hirdetések, vásárlói információ, pénzügyi vezetés, oktatás, államapparátus, elektronikus kereskedelem és sok más területen szolgáltatást nyújtó globális információs központként működik. Tartalmazza még a hiperlink-információ, a webhozzáférés és felhasználás adatainak gazdag és dinamikus gyűjteményét is, és ezzel az adatbányászat számára gazdag információforrást biztosít. A **webbányászat** a *weblinkszerkezetek*, *webtartalom* és a *webelérési minták* bányászatát foglalja magában. Ez maga után vonja az *autentikus weblapok lokalizálását*, a webdokumentumok *automatikus osztályozását*, *többrétegű webinformációs bázis* felépítését, valamint a *webnaplóbányászatot* elősegítő *weblinkszerkezet-bányászatot*.

9.8. | Feladatok

9.1. Egy *heterogén adatbázisrendszer* sok, független módon létrehozott adatbázisrendszerből áll, melyeknek egymás között adatcserét és adat-transzformációt kell végezni, valamint lokális és globális lekérdezésekre kell választ adni. Fejtsük ki, hogy az általánosítás alapú szemlélet segítségével egy ilyen rendszerben hogyan lehet leíró bányászati lekérdezést lefolytatni.

9.2. *Objektumkocka* egy objektumorientált adatbázis többdimenziós általánosítását megelőzően történő, viszonylag strukturált adatokká történő általánosításával hozható létre. Fejtsük ki, hogyan kell kezelni az objektumkockában a halmazorientált adatot.

9.3. *Tértársítás-bányászatot* legalább kétféleképpen lehet megvalósítani: (1) különböző térobjektumok közötti térbeli társítási relációk bányászata lekérdezésen alapuló dinamikus kiszámítással; (2) a térobjektumok közötti térbeli távolságok előre történő kiszámolásával, ezen eredmények felhasználásával a társításbányászat végrehajtása. Fejtsük ki (1) hogyan kell a két módszert hatékonyan megvalósítani és (2) melyik helyzetben melyik módszer az előnyösebb.

9.4. A városi közlekedési osztály adatanalízist akar végezni a főútvonalak forgalmán, hogy a főúthálózatot a nap különböző óráiban gyűjtött forgalmi adatok alapján tervezze meg.

(a) Tervezzen egy térbeli adattárházat, amely a főútvonalak forgalmát tárolja, és amelynek segítségével könnyen nyomon követhető a főútvonalak napszakokra, a hét napjaira vett átlagos és csúcsidő forgalma, valamint a nagy balesetek forgalmi helyzetei.

(b) A tervezők segítésére milyen információt tudunk bányászni egy ilyen térbeli adattárházból?

(c) Ez a téradattárház egyaránt tartalmaz tér- és időadatokat. Javasoljon olyan bányászat technikát, amellyel hatékonyan bányászhatunk egy ilyen tér-idő adattárházból!

9.5. A *hasonlóságkeresés a multimédiában* a multimédia-adatvisszakereső rendszerek fejlesztésének fő témája. Viszont sok multimédia-adatbányászati módszer az elkülönült, egyszerű multimédia-tulajdonságok (szín, forma, leírás, kulcsszó stb.) analízisén alapul.

(a) Meg tudjuk-e mutatni, hogy a multimédia-adatokban történő hasonlóság alapú keresés és az adatbányászat integrálása jelentős fejlődést válthat ki? Példaként mutathatunk bármilyen bányászati feladatot, például a multimédia-analízis, osztályozás, társítás vagy klaszterezés.

(b) Vázoljon egy hasonlóság alapú keresést megvalósító technikát, amely a multimédiaadatok klaszterezési minőségét megjavítja.

9.6. Egy erőmű a következő adatokat tárolja: energiafogyasztás idő és területi bontásban, területenként az egyes fogyasztók energiafelhasználása. Ismertessük, hogyan kell megoldani egy ilyen *idősor-adatbázisban* a következő problémákat:

(a) Egy adott terület péntekenkénti energiafogyasztási görbéjében hasonló szakaszok kikeresése.

(b) Mi történik minden egyes alkalommal 20 percen belül az után, hogy a fogyasztási görbe meredeken megugrik?

(c) Hogyan tudjuk feltárni azt a tulajdonságot, amely legjobban hat a fogyasztás stabilitására?

9.7. Egy éttermi lánc szeretné kibányászni a vendégek fő sporteseményekkel összefüggő fogyasztói szokásait, például *„Mindig, amikor a Canucks jégkorong mérkőzés megy a tv-n, a mérkőzés előtt egy órával a Kentucky Fried Chicken eladása 20%-kal megnő"*.

(a) Adjuk meg az ilyen minták feltárásának hatékony módszerét.

(b) A legtöbb, idővel kapcsolatos társításbányászati algoritmus az ilyen minták feltárásához apriori tulajdonságú algoritmusokat használ. Egy másik, a 6.2.4. alfejezetben bemutatott vetített adatbázis alapú FP- (frequent pattern, gyakori minta) bővítés módszere hatékony a gyakori tételcsoportok bányászatnál. Ki tudjuk-e bővíteni az FP-bővítés módszerét, hogy az ilyen idővel kapcsolatos mintákat hatékonyan lehessen feltárni?

9.8. Egy *e-mail adatbázis* nagyszámú e-mail üzenetet tárol. Úgy tekinthetjük, mint egy, főként szöveges adatot tartalmazó, félig strukturált adatbázis. Fejtsük ki a következőket:

(a) Milyen struktúrát kell adni egy ilyen e-mail adatbázisnak, hogy támogassa a többdimenziós keresést, feladó, címzett, tárgy, idő stb. szerint?

(b) Mi bányászható ki egy ilyen e-mail adatbázisból?

(c) Tegyük fel, hogy a korábbi e-mail üzeneteknek van egy nyers osztályozású *(kiszórandó, lényegtelen, normál, fontos)* gyűjteménye. Adjuk meg, hogyan használhatja fel egy adatbányászati rendszer ezt kísérleti csoportként az új vagy osztályba nem rendezett levelek automatikus osztályozására.

9.9. A World Wide Web dinamikus jellege és az óriási tárolt adatmennyiség miatt nehéz felépíteni számára egy globális adattárházat. Viszont érdekes és hasznos az internet összegzett, lokalizált többdimenziós információja számára adattárházakat építeni. Egy internetszolgáltató cég szeretne felállítani a turisták számára egy internet alapú adattárházat, amely segít a helyi szállodák és éttermek kiválasztásában.

(a) Lehet-e egy, ezt a szolgáltatást támogató web alapú turista-adattárházat tervezni?

(b) Tegyük fel, hogy mindegyik szálloda és étterem rendelkezik saját weblappal. Ismertessük, hogyan találjuk meg ezeknek a weblapoknak a helyét, és milyen módszereket kell alkalmazni, hogy a lapokról kinyerjük az információt, amellyel feltöltjük az adattárházat!

(c) Részletezzük, hogyan kell létrehozni olyan bányászat módszert, amely mindig, amikor a keresés visszaad egy új weblapot, olyan kiegészítő információval tud szolgálni, mint: *„A Downtown szállóban megszálló vendégek 90%-a legalább kétszer az Emperor Garden Éteremben vacsorázik".*

9.10. Mindegyik tudományos vagy mérnöki terület saját témakör szerinti indexosztályozási szabvánnyal rendelkezik, amelyet gyakran az adott terület dokumentumainak osztályba sorolására használnak.

(a) Tervezzünk egy webdokumentum-osztályozási módszert, amely az ilyen témaindexelést fel tudja használni a webdokumentumok automatikus osztályozására.

(b) Fejtsük ki, hogyan kell felhasználni a web linkinformációt egy ilyen osztályozás jobbá tételére.

(c) Fejtsük ki, hogyan kell felhasználni a webhasználat-információt egy ilyen osztályozás jobbá tételére.

9.11. A webnaplórekordok az adatbányászat számára bő webfelhasználási információt nyújtanak.

(a) A webnaplóból az elérési szekvenciák bányászata segíthet bizonyos weblapoknak a webszerver pufferébe történő előre olvasásában. Ilyen lapok azok, melyekre a következő néhány egérkattintás során szükség lesz. Tervezzük meg az ilyen elérési szekvenciák bányászatának hatékony végrehajtási módszerét.

(b) A webnaplóból az elérési bejegyzések bányászata segíthet a felhasználók csoportosításában, ami lehetővé teszi a személyre szabott marketing kialakítását. Részletezzük, hogyan kell kifejleszteni egy hatékony végrehajtási módszert, amely segíti a felhasználók csoportosítását.

9.9. | Irodalom

A bonyolult adattípusok bányászata egy gyorsan fejlődő, népszerű kutatási terület, sok kutatási dolgozattal és bevezető oktatási anyaggal, amelyek az adatbányászat és adatbázisrendszer témájú konferenciákon jelentek meg. Ez a fejezet néhány fontos témát fed le, többek között a bonyolult adatobjektumok többdimenziós analízisét és bányászatát, téradatbányászatot, multimédia-adatbányászatot, idősor és más időkapcsolatban lévő adatok bányászatát, szöveges bányászatot és webbányászatot.

Zaniolo, Ceri, Faloutsos és mások [ZCF+97] a bonyolult adatobjektumokat kezelő fejlett adatbázisrendszerek egy rendszerezett bevezetését adják. A bonyolult adatobjektumok többdimenziós analízise és bányászata terén Han, Nishio, Kawano és Wang [HNKW98] javasolnak egy módszert a többdimenziós általánosítással történő objektumkockák tervezésére és felépítésére, valamint azoknak az objektumorientált és objektumrelációs adatbázisok bonyolult adattípusainak bányászatában való felhasználására. Han, Ng, Fu és Dao [HNFD98] egy módszert javasoltak a szemantikus heterogenitás kezelése céljából, általánosítás alapú adatbányászattal történő többrétegű adatbázis felépítésére. A tervadatbázisok bányászatára szolgáló „oszd meg és uralkodj" módszert Han, Yang és Kim [HYK99] javasolta.

A téradatbázisról szóló néhány bevezető anyag megtalálható a következő helyeken: Maguire, Goodchild és Rhind [MGR92], Güting [Güt94] és Egenhofer [Ege98]. A geotérbeli adatbányászattal kapcsolatban a téradatbányászati módszerekről teljes körű tanulmány található: Ester, Kriegel és Sander [EKS93]. A földrajzi adatbányászat és tudásfeltárás kutatási eredményeinek egy gyűjteménye megtalálható: Miller, Han [MH00]. Lu, Han és Ooi [LHO93] javasolt egy általánosítás alapú attribútum orientált adatanalízises téradatbányászat módszert. Ng és Han [NH94] szerint a leíró téradatanalízist az előre definiált fogalmi hierarchiák helyett klaszterezési eredményeken kell elvégezni. Han, Stefanovic és Koperski [HSK98] a téradatkockák tervezésével és felépítésével kapcsolatos problémákat vizsgálta. Zhou, Truffet és Han az on-line többdimenziós téranalízis és a téradatbányászat számára hatékony poligonösszerakó módszereket ajánlott [ZTH99]. Koperski és Han [KH95] a tértársítási szabályok számára javasolt egy fokozatos finomítási módszert. Knorr és Ng [KN96] bemutatott egy módszert a téradatbázisokban a csoportos szomszédságfeltárást célzó bányászatra. Ester, Kriegel, Sander és Xu [EKSX97], valamint Ester, Frommelt, Kriegel és Sander [EFKS98] térosztályozás és trendanalízises módszereket dolgoztak ki. A téradatosztályozásra Koperski, Han és Stefanovic [KHS98] kétlépéses módszert ajánlottak. Az utóbbi kutatások egy igen aktív területe volt a földrajzi adatbányászatot támogató térklaszter. A térklaszterezési módszerek részletes hivatkozási listáját a 8. fejezet publikációs ismertetésében lehet megtalálni. A téradatbányászat GeoMiner elnevezésű prototípusát Han, Koperski és Stefanovic fejlesztették ki [HKS97].

A multimédia-adatbázisrendszerek elmélete és gyakorlata sok könyvben és kutatási tanulmányban került bemutatásra, többek között Subramanian [Sub98], Yu és Meng [YM97] által. Az IBM QBIC (Query by Image and Video Content) rendszerét Flickner, Sawhney, Niblack, Ashley és mások [FSN+95] mutatták be. A hagyományos és multimédia-adatállományok indexelésére, struktúrabányászatára és megjelenítésére Faloutsos és Lin [FL95] egy FastMap elnevezésű gyors algoritmust fejlesztettek ki. Natsev, Rastogi

és Shim [NRS99] kifejlesztették a WALRUS elnevezésű képadatbázisokból visszanyerő hasonlósági algoritmust, amely terület finomsággal tárja fel a hullám jelalakokat. Fayyad és Smyth [FS93] a Vénusz vulkánjainak azonosítására szolgáló, a nagy felbontású radarképek analízisét végző osztályozási módszert dolgoztak ki. Fayyad, Djorgovski és Weir [FDW96] a Palomar Observatory Sky Survey (POSS-II) projekt keretén belül a galaxisok, csillagok és más csillagobjektumok osztályozására döntési fa alapú módszereket alkalmaztak. Stoloz és Dean [SD96] egy, a távoli érzékelőkről érkező képek alapján földrengésészlelő adatbányászati rendszert, a Quakefindert fejlesztették ki. Zaïane, Han és Zhu [ZHZ00] egy, a nagyméretű multimédia-adatbázisok objektumainak és tulajdonság társításoknak bányászatára szolgáló fokozatosan leásó (deepening) módszert ajánlottak. Zaïane, Han, Li és mások [ZHL+88] egy multimédia-adatbányászati rendszer prototípusát – MultiMediaMiner – fejlesztették ki.

Többek között Chatfield [Cha84] és Shumumway [Shu88] az idősoranalízis statisztikai módszereit javasolták, és azokat széleskörűen tanulmányozták. Szekvenciaadatbázisok hatékony hasonlóságkeresését tanulmányozta Agrawal, Faloutsos és Swami [AFS93]. Az idősor-adatbázisokban gyors részszekvencia-illeszkedési módszert mutatott be Faloutsos, Ranganathan és Monolopoulos [FRM94]. Az idősor-adatbázisokban zaj, skálázás és transzláció jelenléte mellett történő gyors hasonlóságkeresést fejlesztett ki Agrawal, Lin, Sawhney és Shim [ALSS95]. A történésformák lekérdezésére nyelvi primitíveket ajánlott Agrawal, Psaila, Wimmers és Zait [APWZ95]. A hasonlóság alapú idősoradat-kereséssel foglalkozó további munkák: Rafiei és Mendelzon [RM97], Yi, Jagadish és Faloutsos [YJF98], Park, Chu, Yoon és Hsu [PCYH00], valamint Perng, Wang, Zhang és Parker [PWZP00].

Szekvenciális minták bányászatára Agrawal és Srikant [AS95, SA96] egy apriori tulajdonságú technikát és egy általános szekvenciaminta-bányászati algoritmust dolgoztak ki. Mannila, Toivonen és Verkamo [MTV97] figyelembe veszik a szekvenciákban a gyakori történéseket, ahol a történések alapvetően nem ciklikus (esemény) történésgráfok, melyek élei az időbeli „előtte–utána", időintervallum-korlátozás nélküli kapcsolatok. Lu, Han és Feng [LHF98] együttműködési (intertransaction) társítási szabályokat javasoltak, amik implikációs szabályok, melyek oldala időintervallum-korlátokkal bíró, mindkét oldalukon teljes rendezettségű (a történéseken belül és mindkét oldalukon lévő események szerint). Bettini, Wang és Jajodia [BWJ98] figyelembe veszi az együttműködési társítási szabályok általánosítását. Özden, Ramaswamy és Silberschatz [ORS98] ciklikus társítási szabályok bányászati módszereit tanulmányozta. Tervhibák szekvenciaminta-bányászatát javasolta Zaki, Lesh és Ogihara [ZLO98]. Han, Dong és Yin [HDY99] a parciális periodicitás bányászatára egy maximális részmintatalálati módszert javasol. Garofalakis, Rastogi és Shim [GRS99] egy hatékony, korlátozás alapú szekvenciaminta-bányászatot javasol. Han, Pei, Mortazavi-Asl és mások [HPMA+00] kidolgoztak egy FreeSpan elnevezésű, adatbázisvetítésen és partcionált bányászaton alapuló hatékony szekvenciaminta-bányászati módszert. Yi, Sidiropoulos, Johnson, Jagadish és mások [YSJ+00] bevezettek az együtt fejlődő idősorok on-line bányászatára egy módszert.

Az információvisszakeresés és szöveges analízis területén a szöveges adat analízis széles körű vizsgálat tárgya volt. A témában sok jó könyv és kutatási tanulmány született, többek között Salton és McGrill [SM83], Faloutsos [Fal85], Salton [Sal89], van Rijsbergen [vR90], Yu és Meng [YM97], Raghavan [Rag97], Subramanian [Sub98], valamint Kleinberg és Tomkins [KT99]. A korábbi információvisszakeresési módszereket Baeza-

Yates és Riberio-Neto [BYRN99] könyve rendszerezve mutatja be. Deerwester, Dumais, Furnas és mások [DDF+90] kidolgozták a dokumentum hasonlóságelemzésének rejtett szemantikus indexelési módszerét. A szignófájlok használatát Tsichritzis és Christodoulakis [TC83] írta le. Feldman és Hirsh [FH98] tanulmányozta a szöveges adatbázisokban a társítási szabályok bányászatát. A társításbányászaton alapuló automatizált dokumentum-osztályozási módszert Wang, Zhou és Liew ajánlotta [WZL99].

Sok munka készült a webadat-modellezés és a weblekérdező rendszerek témakörben. Többek között a W3QS Konopnicki és Shmuelitől [KS95], a WebSQL Mendelzontól, Mihailatól és Milotól [MMM97], a Lorel Abitboultól, Quasstól, McHughtól és másoktól [AQM+97], a webnapló Lakshmanantól, Sadritól és Subramaniantól [LSS96], a WebOQL Arocenatól és Mendelzontól [AM98], valamint a NiagraCQ Chentől, DeWitttől, Tiantól és Wangtól [CDTW00]. Florescu, Levy és Mendelzon [FLM98] a webadatbázisok kutatásáról egy teljes körű áttekintést mutat be.

Az autentikus weblapok felismerésére szolgáló weblink-szerkezetek bányászatát Chakrabarti, Dom, Kumar és mások [CDK+99] mutatták be. A HITS algoritmust Kleinberg dolgozta ki [Kle99]. A weblaprangsoroló algoritmust Brin és Page dolgozta ki [BP98]. Weblaposztályozást Chakrabarti, Dom és Indyk [CDI98], valamint Wang, Zhou és Liew [WZL99] vizsgálta. A WebML – webbányászat nyelvet Zaïne és Han javasolta [ZH98]. A webadattárház felépítését szolgáló többszintű adatbázis megközelítést Zaïne és Han [ZH95] vizsgálta. A hipertext és a webadatbányászatáról Chakrabartinál [Cha00] találunk gyakorlati áttekintést.

A webhasználat bányászatát sok szoftvercég támogatta és megvalósította. Perkowitz és Etzioni [PE99] javasolta a webnaplók anyagán alapuló, a felhasználói elérési mintákon tanuló adaptív weblapok létrehozását. Tauscher és Greenberg [TG97] a web használhatóságát feltáró webnapló-elérési minták használatát tanulmányozta. A WebLogMiner kutatási prototípusról számolt be Zaïne, Xin és Han [ZXH98]. Srivastava, Cooley, Deshpande és Tan [SCDT00] a webhasználat bányászatáról és felhasználásáról adott áttekintést.

10. FEJEZET

Alkalmazások és irányzatok az adatbányászatban

Melyek az adatbányászat jellemző alkalmazásai a tudomány és az üzleti élet területén? Merre fejlődik? A könyv korábbi fejezetei után ilyen és hasonló kérdések merülhetnek fel az olvasóban. Ebben az utolsó fejezetben áttekintjük az adatbányászat alkalmazási területeit, és javaslatokat adunk az adatbányászati szoftverrendszerek beszerzésénél figyelembe veendő szempontokra. További területeket mutatunk be a témakörből, így például kitérünk az audiovizuális adatbányászatra, a statisztikai módszerekre, az elméleti alapokra és az adatbányászati módszerek segítségével megvalósított intelligens válaszadásra. Megvitatjuk továbbá az adatbányászat jövőbeli trendjeit és társadalmi hatásait.

10.1. | Az adatbányászat felhasználási területei

A könyv eddigi fejezeteiben a relációs adatbázisok, adattárházak és összetett adattípusok (így például földrajzi, multimédiás adathalmazok, idősorok, szöveges és web alapú adatok) bányászatával kapcsolatos alapelveket, módszereket tanulmányoztuk. Mivel az adatbányászat fiatal, sok és sokféle területet felölelő alkalmazási lehetőségekkel bíró tudományág, ezért az általános elvi alapvetések és a konkrét, gyakorlati feladatokra adaptált hatékony adatbányászati eszközök között egyelőre jelentős szakadék húzódik. Ebben a részben megvizsgáljuk, hogy néhány alkalmazási területen milyen testreszabott adatbányászati eszközökre van igény.

10.1.1. | Orvostudományi és DNS-adatok bányászata

Az elmúlt évtizedben az orvostudomány robbanásszerű fejlődésének lehettünk tanúi, ideértve például az újabb és újabb gyógyszerkészítmények kifejlesztését, a különböző rákos megbetegedések kezelésében tapasztalható előrelépéseket, valamint az emberi génállományban fellelhető génszakaszok szerepének meghatározását. Mivel a kutatások jelentős része foglalkozik a DNS-ben tárolt adatok elemzésével, a jelen fejezetben ezek adatbá-

nyászati hátterére térünk ki részletesebben. A DNS-vizsgálatokban véghezvitt legújabb kutatások lehetővé tették számos betegség és fogyatékosság genetikai okainak feltárását, új gyógyszerek kifejlesztését, valamint a betegségek diagnosztizálásával, megelőzésével és kezelésével kapcsolatos, az eddigiektől eltérő szemléletmódok kialakítását.

Az emberi genom kutatásának központjában a DNS-szekvenciák tanulmányozása áll, hiszen ez minden egyes élőlény genetikai kódjának alapja. Minden DNS-szakasz 4 alapvető építőkockából *(nukleotidból)* áll. Ezek az adenin (A), a citozin (C), a guanin (G) és a timin (T). Ennek a négy nukleotidnak a kombinációi hosszú, spirál alakban tekeredő, létrára emlékeztető láncokat alkotnak.

Mi emberek megközelítőleg 100 000 génnel rendelkezünk. Egy gén általában több száz, egyedileg elrendezett nukleotidból áll. A nukleotidok sorrendje és az abból alkotott szekvenciák, gének lényegében végtelen változatosak lehetnek, így a különböző betegségekben szerepet játszó gének meghatározása óriási kihívást jelent. Mivel az adatbányászatban számos, szekvenciális minta kinyerésére és hasonlóság keresésére szolgáló módszer létezik, az adatbányászat a DNS kutatásával foglalkozó tudósok kezében az egyik leghatékonyabb eszközzé lépett elő, ami az alább felsorolt módokon járul hozzá a DNS elemzéséhez.

- **Heterogén, elosztott génállomány-adatbázisok szemantikus integrációja** – A DNS-állomány kutatásával számos intézmény foglalkozik, így a kutatási folyamatok során előálló adatok elosztottan, nem egységesített formában kerülnek rögzítésre. Ezen heterogén, elosztott DNS-adatbázisok módszeres, összehangolt elemzéséhez a génállományra vonatkozó adathalmazokat szemantikusan egységesíteni kell. Ez a törekvés mozdította előre az olyan integrált adattárházak és elosztott, „föderatív" jellegű adatbázisok kifejlesztését, amelyekkel az elsődleges és a származtatott genetikai adatok is tárolhatók és kezelhetők. Az adatbányászatban kifejlesztett adattisztítási és adatintegrációs módszerekkel megoldhatóvá válik a genetikai adatok integrálása, valamint az adatok összetett elemzését lehetővé tevő adattárházak építése.
- **Hasonlóságkeresés, DNS-szekvenciák összehasonlítása** – Az idősorok adatbányászatával foglalkozó részben bemutattunk néhány, hasonlóság keresésére használható eljárást. A gének elemzése során az egyik legfontosabb keresési feladat a hasonló DNS-szekvenciák keresése és ezeknek a szekvenciáknak az egymással történő összevetése. A megbetegedett és egészséges szövetekből kivont génrészletek összehasonlításával felfedhetők a két géntípus közötti kritikus különbségek. Ehhez először a megfelelő génrészleteket kell kinyerni a két összehasonlítandó szövetállományból, majd megtalálni és összevetni az egyes szövetekben gyakran előforduló mintákat. Az egészségeshez képest a megbetegedett szövetekben nagyobb arányban előforduló szekvenciák az adott betegség genetikailag kódolt mivoltára utalhatnak. Ugyanakkor az egészséges állományban jellemzően gyakrabban megjelenő génrészletek az adott betegség megelőzésében játszhatnak fontos szerepet. Fontos azonban megjegyezni, hogy a genetikai adatok elemzésénél használt hasonlóságkereső módszerek jelentősen különböznek az idősorok elemzésénél felhasznált módszerektől. Így például az idősorok elemzésénél széles körben használt módszerek, például a skálázás, a normalizálás és az ablakokra bontás a nem numerikus jellegű genetikai adatok esetében hatástalanok, tekintve, hogy a különböző nukleotidok között meghúzódó sokféle kölcsönhatás szin-

tén meghatározhatja az egyes génszakaszok által betöltött funkciót. Másfelől a gyakori sorrendi minták elemzése rendkívül fontos a génrészletek hasonlóságának és különbözőségének vizsgálatánál.

- **Társítási szabályok keresése: együtt előforduló génrészletek azonosítása** – Jelenleg számos tanulmány a gének egymással történő összehasonlítását helyezi a kutatások fókuszába. Ugyanakkor a legtöbb betegségért nem egy gén felelős, hanem különböző gének egy kombinációja. A társítási szabályok kutatásában már bevált módszerek segíthetnek az olyan géntípusok kimutatásában, amelyek nagy valószínűséggel együttesen fordulnak elő bizonyos vizsgált mintákban. Az ilyen típusú vizsgálatok lehetővé tennék a géncsoportok felfedezését, és a közöttük lévő kölcsönhatások, kapcsolatok vizsgálatát.
- **Epizódkutatás: gének kapcsolata a betegségek egymást követő stádiumaival** – Miközben egyes géncsoportok hozzájárulhatnak egy betegség kialakulásához, annak különböző fázisaiban a csoporton belül mindig más és más gén befolyásolhatja aktívan a betegség további alakulását. A betegség lefolyását vezérlő genetikai aktivitássorozat megismerése hozzájárulhatna olyan gyógyszeres terápiás módszerek kifejlesztéséhez, amelyek a betegség minden egyes fázisában a lehető leghatékonyabb kezelését valósítanák meg. Az ehhez hasonló epizódkutatások várhatóan igen fontos szerephez jutnak majd a genetikában.
- **Adatszemléltető eszközök a genetikai adatbányászatban** – A gének összetett, szekvenciális mintákkal tarkított felépítése hatékonyan szemléltethető a legkülönbözőbb vizuális adatmegjelenítő eszközökben alkalmazott gráfok, fák, adatkockák és láncok segítségével. Ezek a szemet gyönyörködtető vizuális megoldások segíthetnek a minta- és tudáskinyerésben, valamint az interaktív adatfeltárásban. A különböző vizuális szemléltető technikák tehát fontos szerepet játszhatnak az orvosbiológiai adatbányászat terén.

10.1.2. | Adatbányászat a pénzügyekben

A bankok és pénzintézetek legtöbbje számos banki (számlavezetés, megtakarítások kezelése, üzleti és egyéni ügyfelek részére végzett tranzakciók stb.), hitelezési (üzleti, jelzálog, gépjármű-hitelezési) és befektetési (például befektetésialap-kezelés) szolgáltatást nyújt. Néhányuk ezen túlmenően biztosítási és tőzsdei befektetési szolgáltatásokkal is foglalkozik.

A banki és pénzügyi szektorban rendelkezésre álló adatokat a legtöbbször hiánytalanul, megbízhatóan és jó minőségben tárolják, ami elősegíti a szisztematikus adatelemzési és adatbányászati munkát. A következőkben néhány jellemző esetet mutatunk be.

- **Többdimenziós elemzésre és adatbányászatra alkalmas adattárházak tervezése és építése** – Más alkalmazási területekhez hasonlóan a pénzügyi és banki adatokon végzett többdimenziós adatelemzések végrehajtása is igényli az adattárházak által nyújtott lehetőségeket. Érdekes lehet a bevételek és kiadások hónapokra, régiókra, üzletágakra lebontott alakulása, de szükséges lehet egyéb statisztikai jellemzők, például a maximum, minimum, összeg, átlag, trend kinyerésére is. Az adattárházak, többdimenziós és felfedezésvezérelt adatkockák, a leíró és összehasonlító statisztikák, a szélsőségek

elemzése mind-mind jelentős szerepet töltenek be a pénzügyi adatok elemzésében és bányászatában.

- **Hiteltörlesztési hajlandóság előrejelzése és hitelbírálati irányelvek elemzése** – Üzleti szempontból a bankok egyik legkritikusabb tevékenységei közé sorolhatók a hiteltörlesztési hajlandóság előrejelzéséhez és a hitelbírálathoz kapcsolódó ügyviteli folyamatok. Az ügyfél törlesztő- és hitelképességét számos tényező befolyásolhatja kisebb-nagyobb mértékben. Az olyan adatbányászati módszerek, mint például a jellemző tulajdonságok kiválasztása és a tulajdonságok fontosság szerinti rangsorolása, nagyban hozzájárulhatnak az ilyen szempontból lényeges tényezők azonosításához és az irrelevánsak figyelmen kívül hagyásához. Így a hitelvisszafizetési kockázatot befolyásoló faktorok között említhetjük például a hitelösszeg–érték arányt, a hitelvisszafizetési periódus hosszát, a kiadási hányadot (havi összbevétel és kiadás aránya), törlesztőrészlet–bevétel arányt, az ügyfél jövedelemszintjét, az iskolai végzettséget, lakóhelyet, hiteltörténetet stb. Az ügyfélkör hiteltörlesztési adatait vizsgálva kiderülhet, hogy például a törlesztőrészlet–bevétel arány alapvető fontosságú, míg az iskolai végzettség és a kiadási hányad elhanyagolható. Ennek megfelelően a bank módosíthatja hitelfolyósítási politikáját, elfogadva az olyan, korábban elutasított kérelmezők hiteligényét, akiknél a kritikus faktorok analízise alacsony hitelezési kockázatra utal.
- **Fogyasztók célzott marketing szempontú osztályozása és klaszterezése** – A klaszterezési és osztályozási eljárások hatékonyan alkalmazhatók fogyasztói csoportok és direkt marketing célcsoportjainak meghatározásához. Így például a bankügyletek és hiteltörlesztési szokások szempontjából hasonló viselkedésű ügyfeleket többdimenziós klaszterezési eljárások segítségével vonhatjuk össze egy nagyobb csoportba. A hatékony klaszterezési és együttműködő szűrési módszerek (azaz egyszerre több információkiszűrési technika használata, mint például a legközelebbi szomszéd módszerére építő osztályozás, döntési fák stb.) elősegítheti a fogyasztói csoportok meghatározását, új ügyfelek csoportba sorolását és a célzott marketingtevékenységek hatékonyabbá tételét.
- **Pénzmosás és más pénzügyi bűncselekmények felderítése** – A pénzmosással és más egyéb pénzügyi bűncselekményekkel kapcsolatos tevékenységek felderítésében fontos az olyan, több adatbázisból (például banki tranzakciós rendszerek, szövetségi és állami bűnügyi nyilvántartások) származó információk integrálása, amelyek potenciálisan összefüggésbe hozhatók a vizsgálatokkal. Számos adatelemzési eszköz használható az olyan, megszokottól eltérő pénzügyi tevékenységi minták kiszűrésére, mint például a nagy összegű, bizonyos időpontokhoz és embercsoportokhoz köthető pénzforgalmak. Az ilyen feladatok megoldására alkalmas eszközök között említhetők a vizuális adatszemléltető alkalmazások (például banki tranzakciós tevékenységek időpontok és ügyfélcsoportok szerint felépített gráfjának megjelenítése), a kapcsolatanalízis (különböző emberek és tevékenységek közötti összefüggések felderítése), az osztályozás (a vizsgálattal összefüggésbe nem hozható tényezők kiszűrése, és a relevánsak rangsorolása), a klaszterezési eljárások (különböző esetek csoportosítása), a szélsőségek elemzése (kiugróan magas összegű átutalások és egyéb tevékenységek felderítése), valamint a szekvenciális mintakinyerés (pénzfelvételek szokványostól eltérő időbeli sorozata). Az itt felsorolt módszerek a bűnügyek felderítése szempontjából fontos összefüggésekre, tevékenységekre deríthetnek fényt, így a nyomozók ezekre koncentrálva a gyanús ügyeket további vizsgálatnak vethetik alá.

10.1.3. | Adatbányászat a kiskereskedelemben

A kiskereskedelem alkotja az adatbányászat egyik legfőbb alkalmazási területét, hiszen napról napra óriási mennyiségű adat halmozódik fel az eladásokról, a vásárlói szokásokról, az áruszállításról, a fogyasztásról, a kínálatról és számos egyéb tényezőről. A felhalmozott adatok mennyisége igen gyors növekedést mutat, hiszen egyre könnyebben elérhetővé és népszerűbbé válnak az e-kereskedelem vívmányai, így például a weben keresztül lebonyolítható vásárlások. Manapság számos üzlet kínálja portékáit on-line formában is, és nem ritkák a kizárólag az interneten keresztül látogatható áruházak, mint amilyen például az *Amazon.com* is. Ilyenformán a kiskereskedelem az adatbányászatra használható adathalmazok gazdag tárháza.

A kiskereskedőknél felhalmozott adatokból megállapíthatjuk a fogyasztók vásárlási szokásait, gyakori vásárlási mintákat, trendeket fedezhetünk fel, javíthatjuk a szolgáltatások színvonalát, könnyebben megtarthatjuk vevőkörünket és növelhetjük elégedettségüket. Az eddigieknél hatékonyabb áruszállítási és áruelosztási rendszereket tervezhetünk, csökkentve üzletünk költségeit.

Az alábbiakban néhány példát említünk az adatbányászat lehetséges kereskedelmi felhasználására.

- **Az adatbányászat előnyeit kiaknázó adattárházak tervezése és építése** – Mivel a kiskereskedelemben elérhető adatok széles tevékenységi kört fednek le (így az eladások, vásárlók, alkalmazottak, áruszállítás, kereslet és kínálat), számtalan típusú adattárház tervezhető. A tárolásra kerülő adatok részletessége szintén változó lehet. Tekintve, hogy az adattárházak legfőbb célja az adatbányászat hatékony támogatása, a tervezéssel és implementálással párhuzamosan végzett adatbányászati tesztek eredményeinek visszacsatolásával az adattárház felépítése folyamatosan finomítható a későbbi igények eredményesebb kiszolgálása érdekében. Így például a tervezési folyamat során könnyebben születhet döntés a tárolandó adatoknál figyelembe veendő dimenziókról, azok részletességéről, valamint a jó minőségű és hatékony adatbányászat megvalósításához szükséges adat-előfeldolgozási, adattisztítási eljárásokról.
- **Többdimenziós analízis** – A kiskereskedelem számára nélkülözhetetlenek a fogyasztói igényeket, a termékeladásokat, a trendeket és divatokat, valamint a minőséget, a költségszintet, a profitot és az árukínálatot jellemző folyamatos, kellő időben elérhető információk. Ezért nagyon fontosak a hatékony, többdimenziós analízist és vizuális adatszemléltetést megvalósító eszközök, amelyek az adatelemzés igényeihez alkalmazkodó kifinomult adatkockák felépítésére is alkalmasak. Így például a 2. fejezetben bemutatott *többtulajdonságú adatkockák* jól illeszkednek a kiskereskedelemben felmerülő adatelemzések által támasztott követelményekhez, megkönnyítve bonyolultabb feltételek szerint összesített adatok elemzését is.
- **Reklámhadjáratok hatékonyságának vizsgálata** – Amint azt nap mint nap tapasztalhatjuk, a kínálati piac résztvevői által indított reklámhadjáratokban hirdetésekkel, szórólapokkal, különböző árengedményekkel és kedvezményekkel próbálják elnyerni a vevők kegyeit, és előremozdítani termékeik eladását. A marketingtevékenység körültekintő hatékonyságvizsgálata hozzájárulhat a profit növekedéséhez. Többdimenziós elemzések segítségével összehasonlíthatjuk az eladások alakulását a reklámkampányt

megelőző és követő időszakokban. Társítási szabályok keresésével felfedhetjük, hogy melyek azok a termékek, amiket az akciós termékekkel együtt a reklámok hatására esetleg szintén fokozottabban vásárolnak.

- **Vevőmegtartás: a vevőkör lojalitásának vizsgálata** – A manapság népszerű vásárlói „hűségkártyák" segítségével nyomon követhetők az egyes kuncsaftok bevásárlásai. Ezen információk birtokában a vevőkör lojalitása, vásárlási szokásai könnyen és módszeresen elemezhetővé válnak. A vevő által különböző időszakokban vásárolt termékeket sorozatoknak tekintve szekvenciális mintakereséssel vizsgálhatjuk a fogyasztói szokásokban bekövetkezett változásokat, így a meglévő vevők megtartása és újabbak elcsábítása érdekében módosításokat javasolhatunk a termékkínálat és az árstratégia kialakításában.
- **Vásárlási javaslatok és árucikkek kereszthivatkozása** – Az eladási adatokból a vásárolt termékekre vonatkozóan nyert társítási szabályok felhasználásával kimutatható például, hogy egy adott márkájú parfümöt megvásárló vevő a parfümmel összefüggésben más termékeket is hajlamos megvenni. Ezek az információk különböző vásárlási tanácsok formájában reklámként közzétehetők a weben, ingyenes hirdetési újságokban, bevásárlási számlák hátoldalán. Így javítható az ügyfélszolgálat színvonala, ami segíthet a vásárlóknak a megfelelő termék kiválasztásában és nem utolsó sorban növelheti az eladási mutatókat. Hasonlóképpen a társítási szabályokból vett információk akciós termékek és egyéb vonzó ajánlatok formájában is előmozdíthatják különböző termékek forgalmát.

10.1.4. | Adatbányászat a távközlésben

A hagyományos helyi és távhívásokra koncentráló telefonszolgáltatásból kiindulva a telekommunikációs ipar rendkívül gyors fejlődésen ment keresztül, és ma már a hangszolgáltatás mellett a fax, személyhívó, mobiltelefon, képküldés, elektronikus levelezés, adatátvitel és internetkapcsolatok stb. szolgáltatások nyújtása is a kommunikáció szinte minden területére kiterjedő kínálatuk részét képezi. Jelenleg a telekommunikációs és számítógépes hálózatok, az internet és további kommunikációs rendszerek egyre szorosabb integrációjának lehetünk tanúi. Az iparág világszerte tapasztalható deregulációjának, valamint a számítógépes és távközlési technológiák fejlődésének köszönhetően a távközlési piacot a rohamos bővülés és erős verseny jellemzi. Ezért jelentős igény mutatkozik az olyan adatbányászati technológiák iránt, amelyek segítenek a jellemző telekommunikációs felhasználói viselkedésminták megismerésében, a visszaélések felderítésében, a rendelkezésre álló erőforrások hatékonyabb felhasználásában, valamint a szolgáltatás színvonalának emelésében. A továbbiakban néhány olyan területet mutatunk be, ahol az adatbányászat hozzájárulhat a távközlési szolgáltatások fejlesztéséhez.

- **Távközlési adatok többdimenziós vizsgálata** – A távközlési adatok alapvetően több dimenzióval jellemezhetők: hívás időpontja és időtartama, hívó és hívott fél elhelyezkedése, hívás típusa. Az adatok többdimenziós elemzésével meghatározható és összehasonlítható az adatforgalom nagysága, a rendszer terhelése, az erőforrások kihasználtsága, az előfizetői csoportok viselkedése, a profit stb. Az iparági elemzők kíváncsiak lehetnek például a hívásokban a hívó és hívott végpontok területi eloszlását, a forgalom nagyságát vagy a hívásintenzitás egy napon belüli változásait leíró adatokat bemutató diagramokra.

Ezért a távközlési adatokat gyakran nagyméretű adattárházakban vonják össze, és ezeken OLAP- és vizuális adatszemléltető eszközök segítségével végeznek vizsgálatokat.

- **Visszaélésekre utaló motívumok elemzése és szokatlan minták felderítése** – A telekommunikációs ipar számára évente több millió dolláros nagyságrendű veszteséget okoznak a különböző típusú visszaélések. Ezért fontos a potenciális csalók és a csalásra utaló szokatlan felhasználói viselkedésmotívumok meghatározása; más előfizetők számlájára történő tisztességtelen károkozás kimutatása. Hasonlóan lényeges az olyan, normálistól eltérő, különös figyelmet igénylő események felfedezése, mint a csúcsidei hívások sikertelenségének hirtelen megemelkedése, a hálózati kapcsolók és útvonalak túlterheltsége vagy rosszul beállított végberendezések (például faxok) által generált sűrű, automatikus újrahívások. A többdimenziós adatelemzés, a klaszterelemzés és a kiugró értékek vizsgálata használható az ilyen problémák felderítésére.
- **Többdimenziós társítási szabályok elemzése és a szekvenciális mintakinyerés** – A társítási szabályok és szekvenciális minták több dimenzió mentén történő vizsgálata hozzájárulhat a távközlési szolgáltatások előrelendítéséhez. Tegyük fel például, hogy egy távközlési szolgáltatáshalmaz felhasználását jellemző modellt keresünk előfizetői csoportokra, valamint havi és napi szintre lebontva. A hívások jellemzőit tároló adatbázis-rekordok előfizetők szerint csoportosítva a következőképpen néznek ki:

⟨*előfizető_azonosító*, *lakhely*, *munkahely*, *időpont*, *dátum*, *szolgáltatás*_1, *szolgáltatás*_2, ...⟩

Szekvenciális minta lehet például a következő: *„Ha egy Los Angeles környékén élő női előfizető egy lakhelyétől távolabb eső városban dolgozik, akkor hétköznapokon először várhatóan délután 5 óra körül a két város között távhívás(oka)t kezdeményez, majd a következő órában legalább 30 percet beszél a mobiltelefonján"*. Egy ilyen szekvenciális mintát további vizsgálatnak vethetünk alá, meghatározva, hogy a benne foglalt megállapítás érvényes-e különböző városokra, különböző előfizető csoportokra (például mérnökök, orvosok stb.). Az ehhez hasonló elemzések elősegíthetik például speciális, kedvezményes távhívási és mobiltelefon-szolgáltatások kombinációjából felépített, összetett csomagok kidolgozását, és bizonyos szolgáltatások adott régión belüli hozzáférhetőségének javítását.
- **Vizuális adatszemléltető eszközök a távközlésben** – Az OLAP-eszközökhöz kapcsolódó, valamint társításokat, kapcsolatanalízisek eredményét, klasztereket, szélsőséges adatokat szemléletesen megjelenítő szoftverek igen hasznosnak bizonyulnak a távközlési adatok elemzésében.

10.2. | Adatbányászati termékek és kutatási fázisban lévő prototípusok

Bár az adatbányászat viszonylag fiatal tudományágnak mondható – számos további kutatásra váró részterülettel – már ma is széles a készen megvásárolható adatbányászati rendszertermékek és az adott felhasználási területekre specializált célszoftverek választéka. Fiatal tudományágként az adatbányászat viszonylag rövid történeti háttérrel és töretlen

fejlődéssel jellemezhető. Így nem csoda, hogy évről évre újabb adatbányászati rendszerek jelennek meg a piacon, melyek folyamatosan új funkciókkal és vizuális szemléltető eszközökkel bővítik a korábbi rendszereket, miközben az adatbányászatra használható programozási nyelv szabványosítása még gyerekcipőben jár. Éppen ezért nem célunk a piacon fellelhető adatbányászati rendszerek részletes bemutatása. Inkább arra törekszünk, hogy felsoroljuk az adatbányászati rendszerek kiválasztásakor figyelembe veendő lényegesebb szempontokat, és röviden bemutassunk néhány tipikus rendszert. Az adatbányászati rendszerekről szóló fontosabb cikkeket, weboldalakat és a legfrissebb tanulmányokat lásd az irodalomjegyzékben.

10.2.1. | Az adatbányászati rendszer megválasztása

A piacon hozzáférhető adatbányászati rendszerek sokaságát tekintve az olvasóban joggal merülhet fel a kérdés, hogy vajon melyik rendszert érdemes választani. Néhányan úgy gondolhatják, hogy a relációs adatbázis-kezelő rendszerekhez hasonlóan az adatbányászati rendszerek is ugyanazokat a jól definiált eljárásokat valósítják meg, rendelkeznek szabványosított lekérdező nyelvvel, és a megszokott funkciókat hasonlóan látják el. Amennyiben ez valóban így lenne, a választás sokkal inkább az egyes rendszerek által támogatott hardverplatformoktól, kompatibilitásuktól, robusztusságuktól, skálázhatóságuktól, az áruktól és a nyújtott szolgáltatásoktól függne. Sajnos a dolog nem ilyen egyszerű. Számos olyan kereskedelmi forgalomban lévő adatbányászati eszköz létezik, amelyek igen kevés közös tulajdonsággal rendelkeznek, már ami az adatbányászati funkciókat vagy az alkalmazott metodológiát illeti. Ráadásul ezek gyakran teljesen eltérő típusú adathalmazokat képesek csak feldolgozni.

Az adott feladathoz leginkább illeszkedő adatbányászati rendszer kiválasztásánál fontos, hogy több dimenziót is figyelembe vegyünk. Általánosságban elmondható, hogy az adatbányászati rendszereknél a következő szempontokra kell tekintettel lennünk.

- **Adattípusok** – A piacon megtalálható rendszerek nagyobb része formázott, rekord alapú, relációs jellegű, numerikus, kategória és szimbolikus változókat tartalmazó adathalmazok kezelésére képes. Az adatokat ASCII szövegformátumból, relációs adatbázisokból vagy adattárházakból is elérhetik. Fontos, hogy pontosan megismerjük a kiszemelt rendszer által támogatott formátumokat. Néhány adattípus vagy alkalmazás a mintakeresésekhez speciális algoritmusokat igényelhet, amit a készen kapható, általános adatbányászati rendszerek esetleg nem képesek kezelni. Ilyen esetekben specializált rendszerek alkalmazása válhat szükségessé, amelyek képesek szöveges, földrajzi, multimédia-adatok, idősorok, DNS-szekvenciák, webszervernaplók stb. feldolgozására. Előfordulhat, hogy adott alkalmazási területre (pénzügyi, kiskereskedelmi, távközlési) specializálódott rendszereket kell keresni. Sok adatbányászattal foglalkozó cég szállít egyedi igényekre szabott megoldásokat, amelyek alapvető adatbányászati funkciókkal és módszerekkel rendelkeznek.
- **Rendszerkövetelmények** – Egy adatbányászati rendszer több operációs rendszer felett is működőképes lehet, de az is elképzelhető, hogy csak egyet támogat. A Unix és a Microsoft Windows (95, 98, 2000, NT) operációs rendszerek felett futtatható a legtöbb

adatbányászati szoftver. Léteznek OS/2, Macintosh és Linux felett működő rendszerek is. A nagy, ipari jellegű felhasználásra készülő adatbányászati rendszerek lehetőleg a kliens/szerver architektúrát követik, ahol a kliens általában Microsoft Windows alapú PC, míg a szerver szerepét nagyteljesítményű, párhuzamosan működő számítógépekből álló Unix-szerverfarm tölti be. Manapság terjedőben vannak a webes interfészen keresztül, XML ki- és bemeneti formátummal kommunikáló adatbányász rendszerek is.

- **Adatforrások** – Ebben a bekezdésben az adatbányászati rendszerek által kezelt adatformátumokat tekintjük át. Néhány rendszer csak ASCII szövegfájlok feldolgozására képes, míg mások többféle, relációs adatbázisokból származó adatokat is be tudnak olvasni. Fontos, hogy az adatbányászrendszer támogassa az ODBC- vagy az OLE DB for ODBC kapcsolatokat. Ezek biztosítják a nyílt, szabványos módon történő adatbázis-elérést, azaz gyakorlatilag bármilyen relációs adatbáziskezelő rendszerben (DB2, Informix, Microsoft SQL Server, Microsoft Access, Microsoft Excel, Oracle, Sybase stb.) tárolt és a formázott ASCII szöveges adatok elérését is. Az adattárházakhoz kapcsolódó adatbányászrendszereknek támogatnia kell az OLE DB for OLAP szabványt, ami biztosítja a Microsoft SQL Server 7.0, és más, a szabványra épülő adattárház-rendszerekben raktározott adatok elérését.
- **Adatbányászati funkciók és módszerek** – Az adatbányászati funkciók képezik az adatbányászati rendszerek velejét. Néhány rendszer csak egyetlen ilyen feladatot valósít meg, például az osztályozást. Mások több ilyen funkcióval is bírnak, mint például a leíró adatbányászat, felfedezésvezérelt OLAP-elemzés, társítási szabályok,[1] osztályozás, előrejelzés, klaszterezés, szélsőségek elemzése, hasonlóságkeresés, szekvenciális mintakinyerés, valamint a vizuális jellegű adatbányászat. Az egyes adatbányászati funkciókhoz tartozó feladatokat (például az osztályozást) néhány rendszer csak egyetlen módszer segítségével, míg mások több alternatív algoritmussal is meg tudják oldani ugyanazt a feladatot (például döntési fák, Bayes-hálók, neurális hálózatok, genetikus algoritmusok, eset alapú következtetések stb.). Az utóbbi csoportba tartozó rendszerek a felhasználók számára nagyobb rugalmasságot és nagyobb analitikai teljesítményt képesek nyújtani. Sok esetben egy adott probléma megoldásához több különböző funkciót kell kipróbálni, esetleg azokat kombinálni. A különböző módszerek más és más jellegű adatokon nyújtanak jobb teljesítményt. A felhasználó részéről persze sok tapasztalatra és gyakorlásra van szükség, hogy képes legyen kihasználni a nagyobb rugalmasságot biztosító rendszerek előnyeit. Éppen ezért az ilyen eszközök a kezdő felhasználók részére a legnépszerűbb funkciókhoz alapbeállításokat, egyszerűsített és közvetlen hozzáférést nyújtanak.
- **Adatbányászati rendszerek és adatbázis-kezelők, adattárházak összekapcsolása** – Az adatbányászati rendszertől elvárható, hogy adatbázis-kezelő és/vagy adattárház-rendszerrel együtt kerüljön kialakításra úgy, hogy az egymáshoz kapcsolódó rendszerek egy egységes információfeldolgozási környezetbe integrálódjanak. Általánosságban négyféle csatolás létezik: *csatolás nélküli, lazán, közepesen és szorosan csatolt.* Néhány rendszer csak ASCII adatfájlokkal dolgozik, és így *egyáltalán nem csatlakozik* adatbázis-kezelőhöz vagy adattárházhoz. Ezek a rendszerek csak nehezen birkóz-

1 | Számos kereskedelmi forgalomban lévő termékben a társításelemzés helyett a kapcsolatelemzés kifejezést használják.

nak meg a nagyobb, esetleg adatbázisokban tárolt adathalmazokkal. A *lazán csatolt* rendszerek esetében az adatokat először az adatbázisból egy átmeneti tárolóba másolják, és ezen végzik el az adatbányászati feladatokat. Az ilyen rendszerek nehezen skálázhatók, és bizonyos adatbányászati lekérdezések esetében igen rossz hatásfokkal működnek. A *közepesen csatolt* rendszereknél az adatbázis/adattárház és az adatbányász-szoftver közötti kapcsolat néhány hatékonyan implementált, alapvető adatbányászati primitívre épül (például rendezés, indexelés, aggregálás, hisztogramanalízis, többutas illesztés és néhány statisztikai jellemző előzetes kiszámítása). Ideális körülmények között az adatbányászati rendszer *szorosan kapcsolt* az adatbázishoz, így az adatbányászati és adatlekérési folyamatokat integrálják és együttesen optimalizálják. Szintén kívánatos az adatbányászati eszközök OLAP alapú adattárházakhoz történő szoros csatolása OLAP alapú adatbányászati funkciók megvalósításához.

- **Skálázhatóság** – Az adatbányászatban kétféle skálázhatóság létezik: a *sor- (adatbázis mérete)* és *oszlop- (adatok dimenziója) skálázhatóság*. Az adatbányászrendszer sorskálázhatónak tekinthető, ha az adatbázist 10-szeres méretűre növelve a bányászati lekérdezések futási ideje nem több, mint 10-szeres. Hasonlóképpen oszlopskálázhatónak nevezzük a rendszert, ha a lekérdezések futási ideje az oszlopok számával lineárisan nő. A dimenzióproblémák miatt jóval nagyobb kihívást jelent oszlopskálázható, mint sorskálázható rendszert építeni.
- **Vizuális megjelenítő eszközök** – *„Egy kép többet ér ezer szónál"* – tartja a közmondás, és ez alól az adatbányászat sem kivétel. A vizuális megjelenítés adatbányászati szempontból több csoportra osztható: *adatmegjelenítés, az adatbányászati eredmények szemléltetése* és a 10.3.1. alfejezetben tárgyalásra kerülő *vizuális adatbányászatra*. A vizualizációs elemek sokszínűsége, minősége és rugalmassága nagymértékben befolyásolhatja az adatbányászrendszer használhatóságát, értelmezhetőségét és vonzerejét.
- **Adatbányászati lekérdező nyelv és grafikus felhasználói felület** – Az adatbányászat egy feltáró jellegű folyamat. A könnyen használható, magas színvonalú grafikus felhasználói felület alapvető fontosságú a felhasználó által irányított, interaktív jellegű adatbányászat előremozdítása érdekében. A legtöbb rendszer rendelkezik ilyen felhasználóbarát felülettel. Mégis, eltérően az SQL-alapokra (ami egy szabványosított, jól megtervezett adatbázis-lekérdező nyelv) helyezett adatbázis-kezelő rendszerektől, a legtöbb adatbányászrendszer nem rendelkezik közös adatbányászati nyelvvel. A szabványosított adatbányászati nyelv hiányában az adatbányászati rendszerek nehezen szabványosíthatók, és az egyes rendszerek közötti együttműködés is sok akadályba ütközik. A 4. fejezetben már szóltunk az adatbányászati nyelv definiálására és szabványosítására tett erőfeszítésekről. Egy ilyen szabványosított nyelv az A) függelékben részletezett Microsoft OLE DB for DM.

10.2.2. | Néhány adatbányászati termék bemutatása

Amint azt korábban is említettük, könyvünknek nem célja az egyelőre gyermekcipőben járó, de gyorsan fejlődő adatbányászati piacon fellelhető számtalan termék bemutatása. Ehelyett röviden felvázolunk néhány tipikus adatbányászati rendszert, ami alapján az olvasó fogalmat alkothat a jelenleg hozzáférhető termékek képességeiről.

A legtöbb adatbányászati rendszer egy adatbányászati funkció ellátására képes, mint például az osztályozás, sőt léteznek egy funkciót csak egy bizonyos módszertannal (például döntési fa alapú osztályozás) megoldó rendszerek is. Léteznek természetesen az adatbányászati feladatok széles spektrumát támogató eszközök is. Ebben a fejezetben bemutatunk néhány, több feladat ellátását is megoldó adatbányászati szoftvert, és betekintést adunk az ezekben megvalósított tudáskinyerő eljárások működésére.

- Az **Intelligent Miner** az IBM által forgalmazott termék, mely kiterjedt adatbányászati algoritmusgyűjteménnyel bír a társításelemzés, az osztályozás, a regresszió, a prediktív modellezés, az eltérésfigyelés, a szekvenciális mintakinyerés és a klaszterezés területéről. A termékhez kapcsolódik egy külön eszköztár, ami neurális hálózati algoritmusokat, statisztikai módszereket, adatelőkészítést támogató és vizuális adatábrázoló programokat tartalmaz. Az **Intelligent Miner** legfőbb megkülönböztető jegyei az alkalmazott algoritmusok skálázhatóságában és az IBM DB2 relációs adatbáziskezelő rendszerrel való szoros integrációban rejlenek.
- Az **Enterprise Minert** a SAS Institute Inc. fejlesztette ki. Többféle adatbányászati algoritmust kínál a regresszió és az osztályozás témakörében, és rendelkezik statisztikai programcsomagokkal is. Az **Enterprise Miner** erőssége statisztikai elemző eszközeinek sokszínűsége, amelyek a SAS statisztikai elemzések piacán méltán megalapozott múltjára építenek.
- A **MineSet** a Silicon Graphics Inc. (SGI) terméke. Az eddig felsorolt eszközökhöz hasonlóan szintén több adatbányászati algoritmust támogat, így a társításelemzést, osztályozást, és emellett fejlett statisztikai és vizualizációs programot tartalmaz. A **MineSetet** robusztus grafikai eszközei (melyek az SGI-számítógépek hagyományosan erős grafikai támogatására építenek) különböztetik meg versenytársaitól. Az eszköztárban találunk szabálymegjelenítést, faábrázolást, térkép-megjelenítést, valamint szóródást több dimenzióban ábrázoló programokat, amelyek alkalmasak mind az adatok, mind pedig az adatbányászat eredményeinek szemléletes megjelenítésére.
- Az Integral Solutions Ltd. (ISL) által fejlesztett **Clementine** egy integrált adatbányászati fejlesztőkörnyezet mind végfelhasználók, mind pedig fejlesztők számára. Számos beépített algoritmusával indukciós, neurális hálózati, osztályozási feladatokat oldhatunk meg, és ezen túlmenően az adatok vizuális ábrázolására is használható. A **Clementine** legfőbb előnye az objektumorientált, könnyen kiterjeszthető, moduláris interfész, amelyen keresztül új, a felhasználók által kidolgozott algoritmusokkal egészíthetjük ki a Clementine rendszer vizuális programozási környezetét. A Clementine-t nemrég felvásárolta az SPSS Inc.
- A **DBMiner** Technology Inc. által kifejlesztett adatbányászati rendszer szintén többféle algoritmust kínál a felfedezésvezérelt OLAP-elemzés, a társításelemzés, az osztályozás és a klaszterezés területén. A **DBMiner** egyedisége annak adatkocka alapú online analitikus adatbányászati moduljában rejlik, mely hatékony mintakinyerő és vizuális osztályozó algoritmusokat is tartalmaz. A B) függelékben a rendszert részletesebben is bemutatjuk.

Természetesen számtalan további adatbányászati termék, kutatási prototípus létezik még. A témában érdekelt olvasók számára javasoljuk az adattárházakról és adatbányászati termékekről folyamatosan megjelenő tanulmányok időről időre történő áttekintését.

10.3. | További adatbányászati témák

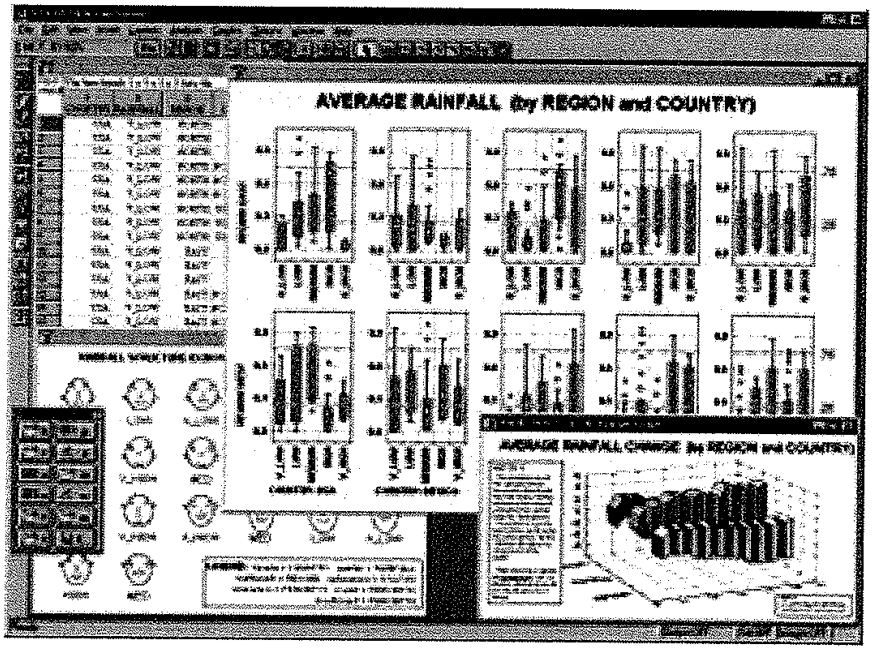
Az adatbányászat széles körű alkalmazhatósága és a létező módszerek sokfélesége miatt jelen könyvben nincs lehetőségünk minden érintett témakör tárgyalására. Ebben a fejezetben röviden megvitatunk néhány olyan érdekes témát, amelyekkel eddig nem foglalkoztunk mélyrehatóbban.

10.3.1. | Látásra és hallásra támaszkodó adatbányászat

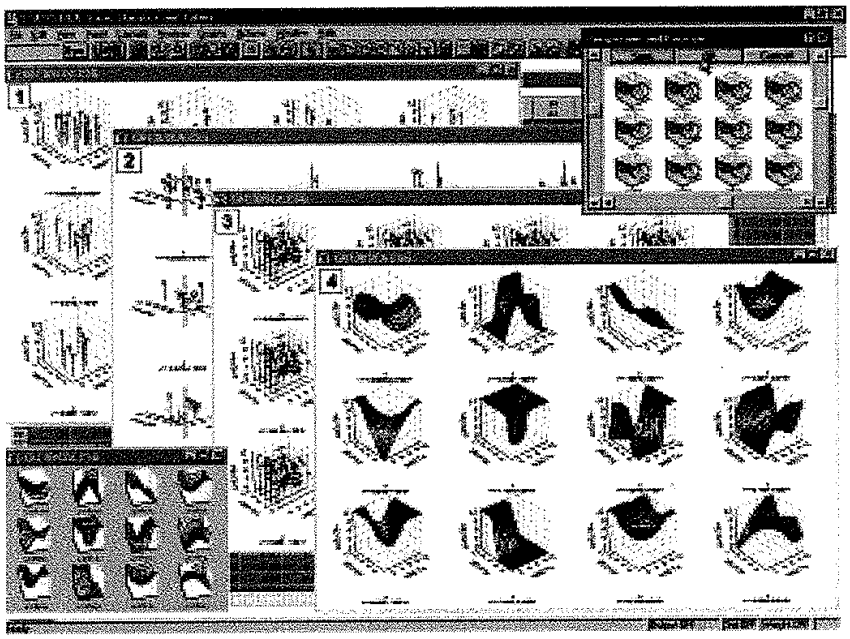
A **vizuális adatbányászat** tudásábrázoló technológiái segítségével implicit és hasznos tudásanyagra lelhetünk nagyméretű adatbázisokban. Az emberi látórendszert az agy és a szemek vezérlik, ahol az agy felfogható egy igen hatékony, masszív párhuzamos feldolgozásra képes, óriási tudásbázissal rendelkező „érvelőgépként". A vizuális adatbányászat egyszerűen ezen összetevők erejét ötvözve kínál vonzó és hatékony eszközöket az adateloszlások, minták, klaszterek, kivételek értelmezéséhez.

A vizuális adatbányászat úgy értelmezhető, mint két tudományág: az adatábrázolás és az adatbányászat integrációja. Szorosan kapcsolódik a számítógépes grafikához, a multimédia rendszerekhez, az ember-számítógép interfészekhez, a mintafelismeréshez és a nagysebességű számítógépek technológiájához. Az adatábrázolás és az adatbányászat általában a következő módokon integrálható:

- **Adatmegjelenítés** – Az adatbázisban vagy adattárházakban tárolt adatok megjelenítésénél állítható az értékek felbontása, a kívánt absztrakciós szint. Az adathalmaz különböző attribútumok függvényében, több dimenzióban is felrajzolható. Többféle megje-



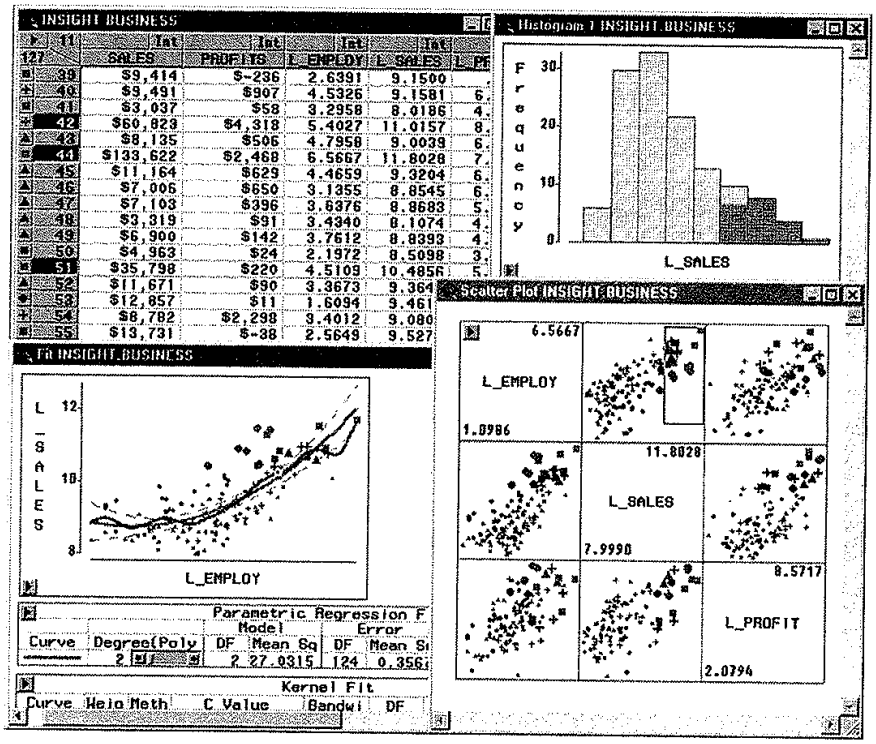
10.1. ábra | Több változó kombinációját mutató dobozdiagramok a StatSoft rendszerben



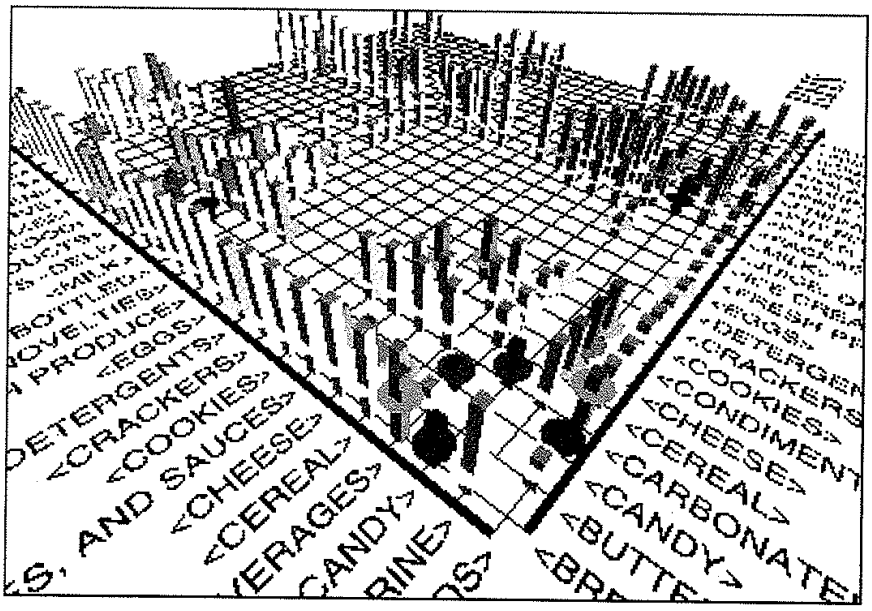
10.2. ábra | Többdimenziós adateloszlás-elemzés a StatSoft rendszerben

lenítési forma választható: dobozdiagramok, háromdimenziós kockák, eloszlásdiagramok, görbék, felületek, kapcsolatgráfok stb. A 10.1. és 10.2. ábrákon a StatSoft nevű szoftverrel készült többdimenziós eloszlásokat mutatunk be. A grafikus megjelenítés segítségével jobban átláthatóvá válnak a tárolt adatok főbb jellemzői.

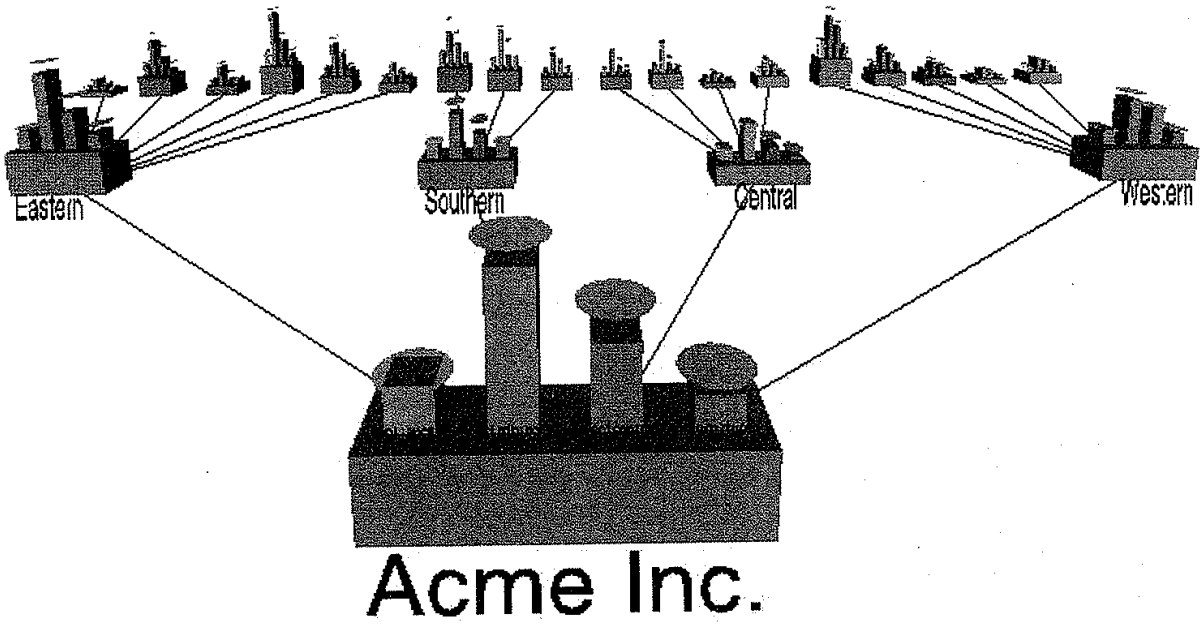
- **Adatbányászati eredmények megjelenítése** – Az adatokon lefolytatott adatbányászati vizsgálódások végeredményei több formában tekinthetőek meg: a leíró adatbányászatból nyert pont- és dobozdiagramok, valamint döntési fák, társítási szabályok, klaszterek, kivételek, általános szabályok stb. formájában. A 10.3. ábrán a SAS Enterprise Mineréből származó pontdiagramokat tekinthetünk meg. A MineSet 3.0 által generált 10.4. ábra társítási szabályokat jelenít meg. A döntési fákat ábrázoló 10.5. ábra szintén MineSet 3.0-ban készült. A 10.6. ábra IBM Intelligent Minerrel készített kép klasztereket, és azokhoz tartozó tulajdonságokat mutat.
- **Adatbányászati folyamat megjelenítése** – Az adatbányászati folyamat egyes lépései képi formában kerülnek bemutatásra, így szemléletessé válnak az adatok adatbázisokból és adattárházakból történő lekérdezései, majd az adatok tisztítása, integrálása, előfeldolgozása és az adatbányászat során feltárt eredmények. Megmutatja, hogy az adott feladathoz melyik módszer került kiválasztásra, hol tárolódnak az eredmények, és azok hogyan tekinthetők meg. A 10.7. ábrán a Clementine által felvázolt adatbányászati folyamatot láthatjuk.
- **Interaktív, grafikus adatbányászat** – Az interaktív, grafikus adatbányászat során használt eszközök támogathatják a felhasználót az adatbányászati folyamat egyes lépéseinél szükséges döntések meghozatalában. Így például a bányászott adatok különböző attribútumok szerinti eloszlását színes körcikkekként vagy oszlopokként megjelenítve (attól függően, hogy a teljes adathalmazt egy kör vagy oszlop reprezentálja) a felhasz-



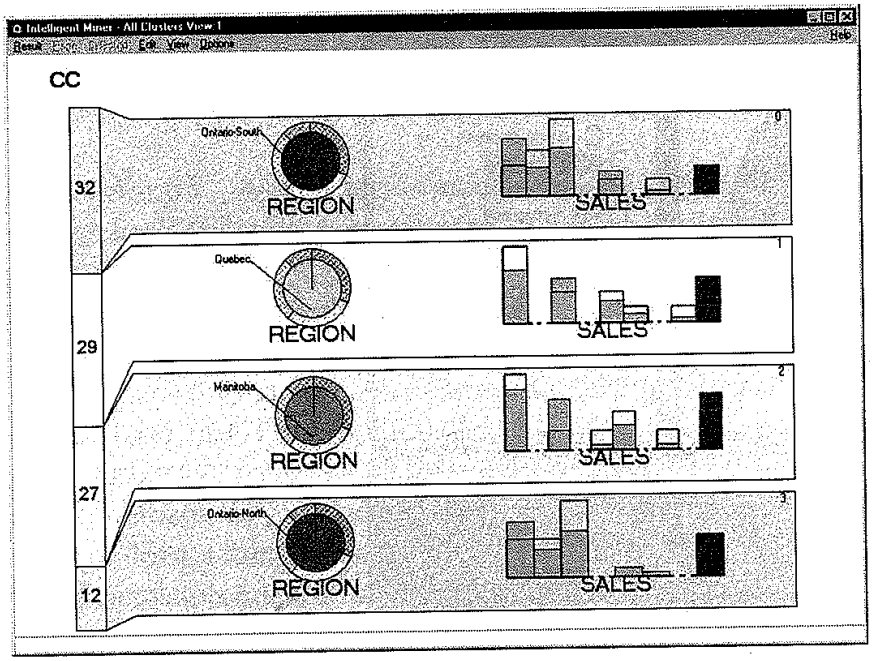
10.3. ábra | Az adatbányászat során kapott eredmények megjelenítése a SAS Enterprise Minerben



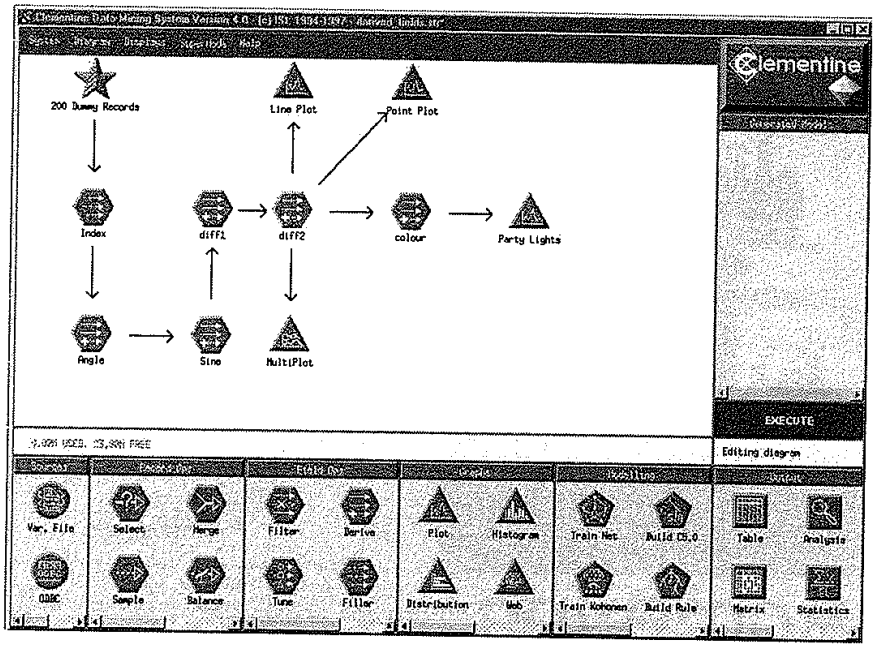
10.4. ábra | Társítási szabályok megjelenítése a MineSet 3.0-ban



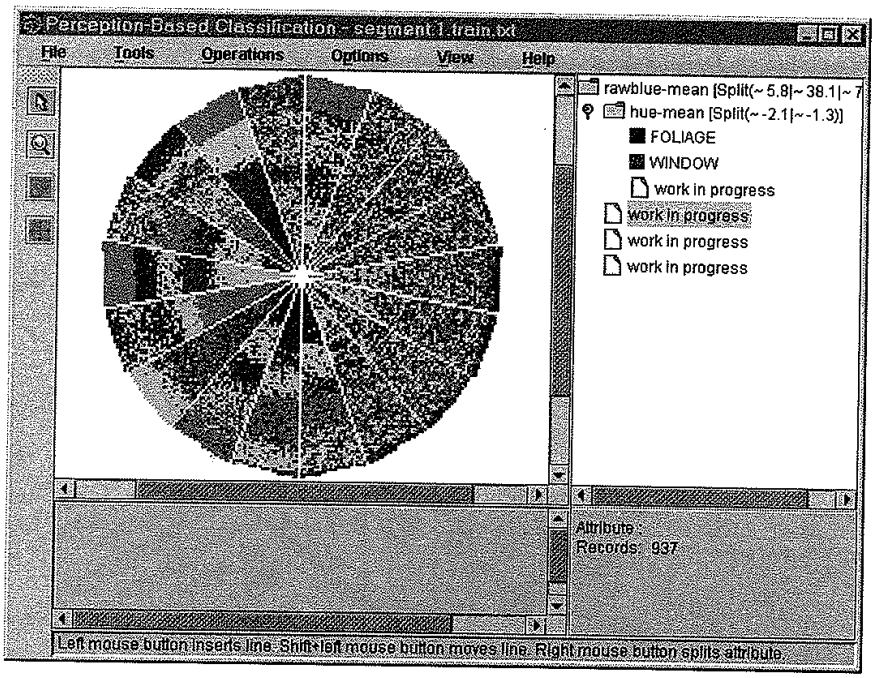
10.5. ábra | Döntési fák ábrázolása a MineSet 3.0-ban



10.6. ábra | Klasztercsoportok megjelenítése az IBM Intelligent Minerben



10.7. ábra | Az adatbányászati folyamat szemléltetése Clementine-ban



10.8. ábra | Érzékelés alapú osztályozás (PBC, Perception Based Classification): egy interaktív képi megközelítés

náló könnyebben eldöntheti, hogy például egy adathalmaz mely részét érdemes elsőként osztályokra bontani. A 10.8. ábrán egy ilyen intuitív, a müncheni egyetemen kifejlesztett, „észlelés alapú" osztályozási rendszer kimenetét tekinthetjük meg.

A **hang alapú adatbányászat** során az adatokban rejlő mintákat, illetve az elemzés eredményeit hangjelekké alakítják. Bár a képi adatbányászat számos érdekes mintára deríthet fényt, azonban a felhasználó részéről folyamatos figyelmet, koncentrációt igényel. A kép-

ernyőn megjelenő sokféle ábra közül a valóban releváns minták kiválasztása így igen fárasztóvá válhat. Amennyiben az elemzendő adatokba rejtett mintákat hangok és zene formájában közvetítjük a felhasználó felé, akkor a képernyőn megjelenő ábraözön helyett a hangmagasságra, ritmusra, dallamra kell csak fülelnünk és ebből következtetnünk az érdekes vagy szokatlan motívumokra. Ez enyhítheti a képi koncentrációs képességeinkre eső terhet, és sok esetben pihentetőbb lehet. Így a hang alapú adatbányászat a grafikus alapú, képi adatbányászat egyik érdekes alternatívájává válhat.

10.3.2. | Tudományos és statisztikai adatbányászat

A könyvben bemutatott adatbányászati technikák java adatbázis-orientált, azaz nagy mennyiségű, általában többdimenziós és több, összetett típusú adat hatékony kezelésére lett kihegyezve. Ezeken kívül azonban számos jól kipróbált, elsősorban numerikus adatok elemzésére szolgáló statisztikai módszer létezik, amelyeket elterjedten használnak tudományos (például a pszichológia, az orvostudomány, a villamosmérnöki és a gyártástudomány területéről származó kísérleti eredmények), valamint közgazdasági és társadalomtudományi adatok vizsgálatára.

Ezek közül a technikák közül a főkomponens-analízisről, a regresszióról és a klaszterezésről már esett szó. Bár nem célunk a nagyobb statisztikai adatelemzési módszerek kimerítő tárgyalása, a teljesség kedvéért az alábbiakban megemlítünk néhányat, és az irodalomjegyzékben felsoroljuk a témát részletesen taglaló kiadványokat.

- **Regresszió** – A regressziós módszerek célja általában egy *válasz (függő)* változó értékének előrejelzése a *magyarázó (független)* változó értékei alapján, ahol a változók numerikusak. A regressziónak számos változata létezik, így beszélhetünk lineáris, többváltozós, súlyozott, polinomiális, nem paraméteres és robusztus regresszióról (a robusztus módszerek normális eloszlástól különböző hibatagok, vagy jelentős mennyiségű szélsőséges adat jelenléte esetén lehetnek hasznosak).
- **Általánosított lineáris modellek (GLM)** – Ezek a modellek *(és az általánosított additív modellek)* megengedik, hogy egy *kategória* típusú válaszváltozó értékét a magyarázó változók egy csoportja (vagy annak valamilyen transzformáltja) határozza meg, hasonlóan a numerikus válaszváltozók lineáris regresszióban történő kezeléséhez. Az általánosított lineáris modellek csoportjába sorolható például a logisztikus és a Poisson-regresszió.
- **Regressziós fák** – Osztályozásra és előrejelzésre használt bináris fák. A döntési fákhoz hasonlóan a fa elágazási pontjaiban teszteket végzünk el. A különbség a kétféle fa leveleinél található, hiszen amíg a döntési fáknál többségi szavazás alapján dől el az adott levélhez történő címke hozzárendelés, addig a regressziós fák esetében az adott levélhez tartozó válaszváltozók értékeinek átlagát használjuk előrejelzésként.
- **Varianciaanalízis (ANOVA)** – Ezzel a módszerrel egy numerikus válaszváltozó egy vagy több kategóriaváltozó *(faktor)* által meghatározott populációkban külön-külön számított értékeit elemezhetjük. Az ANOVA (egyfaktoros varianciaanalízis) általában k darab populáció (vagy például gyógyszervizsgálatok esetén kezelési módszer) között-

ti különbségek meglétének igazolására vagy megcáfolására szolgál. Természetesen léteznek összetettebb ANOVA-vizsgálatok is.

- **Vegyes faktoriális modellek** – Ez a modell egy vagy több csoportváltozó alapján csoportosításra kerülő adatok elemzésére szolgál. Általában egy válaszváltozó és néhány kovariáns közötti összefüggéseket írják le az egy vagy több faktor alapján kialakított csoportokon belül. Leggyakrabban többszintű adatok, ismételt mérések, blokk tervek, és hosszanti adatok elemzésére használják.
- **Faktoranalízis** – A módszer használatával megállapítható, hogy mely változók kombinálásával állítható elő egy faktor. Például a pszichiátriában számos érdeklődésre számot adó faktor nem mérhető közvetlenül (például az intelligencia); mindazonáltal léteznek más, jobban mérhető mennyiségek (például egy diák osztályzatai), amelyekből következtetni lehet a kérdéses faktorra. E módszerben a vizsgált változók egyike sem emelhető ki mint függő változó.
- **Diszkriminancia-analízis** – A módszer kategória típusú válaszváltozók előrejelzésére szolgál. Az általánosított lineáris modellekkel ellentétben feltételezi, hogy a független változók eloszlása normális. Az eljárás több diszkriminancia-függvényt (független változók lineáris kombinációja) próbál meghatározni, amelyek elkülönítik az előrejelezni kívánt válaszváltozó szerint képzett csoportokat. A diszkriminancia-analízist gyakran használják a társadalomtudományokban.
- **Idősorok** – Az idősorok elemzésére számos statisztikai módszer kínálkozik, mint például az autoregresszív eljárások, az egyváltozós ARIMA- (Autoregressive Integrated Moving Average, autoregresszív integrált mozgóátlag) modell és a hosszú távú memóriás idősormodellezés.
- **Túléléselemzés** – A túléléselemzés eredeti célja az volt, hogy becslést adjon arra a valószínűségre, hogy egy adott orvosi kezelésen átesett páciens legalább t időt túlél. E feladatra számos elfogadott statisztikai módszer létezik. A túléléselemzést elterjedten használják ipari berendezések élettartamának becslésére is. A legnépszerűbb módszerek közé tartozik a Kaplan–Meier-túlélésbecslés, a Cox-féle arányos kockázati regressziós modell, valamint ezek továbbfejlesztései.
- **Minőségellenőrzés** – Többféle statisztika is létezik minőségellenőrzést szolgáló diagramok készítésére, mint például a Shewart-féle és a kumulált összegdiagramok (mindkettő csoportösszesítő statisztikák szemléltetésére alkalmas). A fontosabb, minőségellenőrzésnél használt statisztikák között említhetjük az átlagot, a szórást, a tartományt, a darabszámot, a mozgó átlagot, a mozgó szórást és a mozgó tartományt.

10.3.3. | Az adatbányászat elméleti alapjai

Az adatbányászat elméleti alapjai ma még nem mondhatók kiforrottnak. A szilárd és szisztematikus elméleti megalapozás azonban fontos, hiszen összefüggő keretrendszerként szolgál az adatbányászati technológiák fejlesztéséhez, értékeléséhez és gyakorlatához. A következőkben felsoroljuk az adatbányászat elméleti megalapozására hivatott főbb elméleteket.

- **Adatcsökkentés** – Az elmélet szerint az adatbányászat alapját az adatreprezentáció csökkentése képezi. Az adatcsökkentés nagy adatbázisokban elvégzendő lekérdezésekre közelítő, de gyors eredmények előállításával foglalkozik, megpróbálva egyensúlyozni a pontosság és a sebesség között. Az adatcsökkentési technikák között említhető az egyedi értékelemzés (a főkomponens-analízis fő eleme), a waveletek, a regresszió, a log-lineáris modellek, a hisztogram, a klaszterezés, a mintavételezés, valamint az indexfák építése.
- **Adattömörítés** – Ezen elmélet szerint az adatbányászat alapja az adatok bit alapú, társítási szabályok, döntési fák, klaszterek és egyéb más szempontok szerinti kódolás alapján történő tömörítése. A kódolás alapja a *minimális hosszúságú leírás elve*, mely szerint egy adathalmazból kikövetkeztethető „legjobb elmélet" (szabály) az, amelyik minimalizálja magának a szabálynak és az adathalmaznak a szabály alapján tömörített teljes hosszát. Ez a tömörítés általában bit alapú.
- **Mintakinyerés** – Az elmélet szerint az adatbányászat alapját az adatbázisban fellelhető minták – úgy, mint a társítási szabályok, osztályozási modellek, szekvenciális minták stb. – felfedezése képzi. Az elmélet számos területről merít, mint például a gépi tanulás, a neurális hálózatok, a társításelemzés, a szekvenciális mintakinyerés, a klaszterezés stb.
- **Valószínűség-elmélet** – Statisztikai alapokon nyugvó elmélet, mely szerint az adatbányászat lényegét a valószínűségi változók együttes eloszlásának meghatározása képezi. Módszerei között szerepelnek például a Bayes-féle hihetőségi hálók vagy a hierarchikus Bayes-i modellek.
- **Mikroökonómiai szemlélet** – A mikorökonómiai nézet szerint az adatbányászat feladata egy (bármilyen) vállalat döntéshozatali folyamata (például marketingstratégiák kialakítása, termelési tervek stb.) szempontjából fontos minták, összefüggések fellelésére koncentrálódik. Az elmélet nézőpontjából valami akkor hasznos, ha valamilyen formában reagálni lehet rá. A vállalatokat egy olyan optimalizációs feladat megoldásával szembesülő egységnek képzeli, ahol a célfüggvény a döntések értékének vagy hasznosságának maximalizálása. Ebben a szemléletben az adatbányászat egy nem lineáris optimalizálási feladattá válik.
- **Induktív adatbázisok** – Az elmélet szerint az adatbázisséma az adatbázisban tárolt adatokra és mintákra, összefüggésekre tagolódik. Ennek megfelelően az adatbányászat az adatbázisokra épülő következtetésből, általánosításból áll, ahol a feladat az adatok és elméletek, összefüggések (azaz minták) adatbázisból történő kikövetkeztetése, „kipuhatolása". Ez a szemlélet igen népszerű az adatbázisrendszerekkel foglalkozó kutatók körében.

A fenti elméletek nem zárják ki egymást. Így például a mintakinyerés az adatcsökkentés vagy az adattömörítés egy formájának is tekinthető. Ideális esetben az elméleti keretrendszer képes a tipikus adatbányászati feladatok (társításelemzés, osztályozás és klaszterezés) modellezésére, valószínűségi jellegű, különböző formátumú adatok kezelésére is alkalmas, és tudomásul veszi az adatbányászat lényegének iteratív és interaktív mivoltát. A felvázolt követelményeket kielégítő, jól definiált keretrendszer létrehozása azonban még további erőfeszítéseket kíván.

10.3.4. | Adatbányászat és intelligens válaszadás

Az adatbányászati folyamat kiindulópontja egy adatbányászati lekérdezés (kérdés), amely magában foglalja a feladatspecifikus adathalmazt, a feltárandó „tudás" típusát, a szükséges megszorításokat, az érdeklődési küszöböket stb. Mindazonáltal sok esetben a felhasználó nem is tudja pontosan, hogy mit keressen, sőt, esetleg az adatbázis tartalmával sincs tisztában; így előfordulhat, hogy teljesen *precíz* kérdéseket sem tud megfogalmazni. Az „intelligens válaszadás" ilyen esetekben elemezheti a felhasználó szándékait és ezek figyelembevételével adhat választ.

Az intelligens válaszadás és az adatbányászat egy általános keretrendszerben történő integrálása a következőképp valósítható meg. Egy adatbázisban alapvetően kétféle lekérdezés fogalmazható meg: *adat-* és *tudáslekérdezés*. Az **adatlekérdezés** az adatbázisban *konkrétan tárolásra került adatokat* keres és az adatbázisrendszer egy alap lekérdezési utasításának felel meg. A **tudáslekérdezés** ellenben *szabályokat, összefüggéseket, mintákat és egyéb tudást* keres, és egy következtetési szabályokat, teljességi feltételeket, általános szabályokat, gyakori mintákat és más szabályosságokat tartalmazó lekérdezésnek felel meg. Konkrét példákkal megvilágítva a különbséget: a *„Kérem az összes olyan vásárló azonosítóját, akik 2000 májusában pelenkát vettek"* egy adatlekérdezés, míg az *„Írja le ezen vásárlók általános tulajdonságait és találja meg, hogy milyen más termékek vásárlására hajlamosak még"* pedig egy tipikus tudáslekérdezés.

A válaszadó mechanizmusokat két típusba sorolhatjuk: *közvetlen válaszadás és intelligens (vagy együttműködő) kérdések megválaszolása*. A **közvetlen válaszadás** esetében *pontosan azt kapjuk feleletként, amit kérdeztünk*, míg az **intelligens esetben** a *rendszer elemzi a kérdező szándékát és általánosított, vagy a lekérdezéshez köthető, rokon jellegű információt kapunk válaszként*. Tekintsük egy vásárló egy bizonyos könyv címére, szerzőjére, árára és kiadójára vonatkozó kérdését. Ezen attribútumok egyszerű kinyomtatása a közvetlen válaszadásra lehet példa. Ellenben a kérdéshez köthető, de a kérdésben egyértelműen nem szereplő információ (például könyvkritikák, eladási statisztikák, vagy az olyan könyvek címének listája, amiket a kérdéses könyvet vásárlók valószínűleg szintén betesznek a kosarukba) megadása már az intelligens válaszadásra ad példát.

10.1. példa | Tegyük fel, hogy egy web alapú on-line bevásárlóközpont több, az üzlettel kapcsolatos adatbázist tart nyilván. Ezek között találhatunk egy *on-line katalógust*, egy szintén *on-line elérhető üzleti tranzakciós adatbázist és egy nyilvántartást a befutó weblekérdezésekről*.

Az adatlekérdezések a legtöbb on-line elérhető szolgáltatás esetében már megszokottnak tekinthetők, olyan lekérdezésekkel mint *„Listázd ki az összes biciklit"* vagy *„Keresd meg az összes árucikket, amit Jack Waterman vásárolt 2000 áprilisában"*. A közvetlen válaszadás során a rendszer egy, a kérdésnek megfelelő listát ad vissza. Intelligens válaszadás esetén azonban a felhasználó olyan kiegészítő információk birtokába jut, amik további támogatást nyújthatnak döntéseihez. Az alábbiakban néhány olyan példát említünk, ahol az adatbányászati háttérrel támogatott on-line bevásárlószolgáltatás színvonalát fejleszthetné az intelligens válaszadás.

- **Összefoglaló adatokkal bővített, tájékoztató jellegű válaszadás** – Ha egy vásárló a kerékpárok felől érdeklődik, akkor egy egyszerű listán kívül tájékoztathatjuk az éppen aktuális akciós jelleggel árusított kerékpárokról, az egyes kerékpárok eladási statisztikái-

ról, vagy az új modellek vonzó tulajdonságairól. Az ilyen összefoglaló jellegű adatok adattárházak és leíró adatbányászati technikák alkalmazásával érhetők el.

- **Kiegészítő árucikkek ajánlata társítási szabályok keresése alapján** – Amennyiben az ügyfél egy bizonyos típusú kerékpárt óhajt vásárolni, további társítási jellegű információt nyújthatunk számára, mint például: *„Az ilyen típusú biciklit vásárlók általában a következő sportcikkeket is megveszik"* vagy *„Nincs szüksége egy kerékpárról szóló karbantartási útmutatóra?"* Az ilyen ajánlatok növelhetik a cég on-line eladásait.
- **Termékreklámozás szekvenciális mintakinyerés alapján** – Amikor a vevő vásárol egy PC-t, a rendszer további kiegészítő termékek vásárlására adhat javaslatot a korábban már felfedezett szekvenciális minták alapján: *„A PC-t vásárlók zöme általában ilyen és ilyen típusú nyomtatót vagy CD-írót is vesz a vásárlást követő 3 hónapban"*, esetleg rövid időn belül ajándékutalványokat postázhat az ügyfél számára. ■

A fenti példákból kiderül, hogy az intelligens válaszadással számos új és érdekes megoldással bővíthetők a meglévő e-kereskedelmi szolgáltatások. Az adatbányászati alkalmazások körében ez a terület várhatóan egyre nagyobb hangsúlyt kaphat és további kutatásokat igényelhet.

10.4. | Az adatbányászat társadalmi hatásai

A társadalom számítógépes ellátottságának egyre gyorsabb bővülését figyelembe véve nem szabad alábecsülnünk az adatbányászat várható társadalmi hatásait sem. Vajon az adatbányászat is csak egy felfújt léggömbnek bizonyul majd, vagy valóban elérkezett az alkalmazásának ideje? Milyen akadályokat kell még az adatbányászatnak leküzdenie, hogy a technológiai fejlődés főáramába kerüljön előbb az üzleti életben, idővel pedig mindenki személyes használatában? Mit tehetünk az adatbiztonság és titokvédelem megőrzése céljából? Ebben a fejezetben ezekre a kérdésekre keressük a választ.

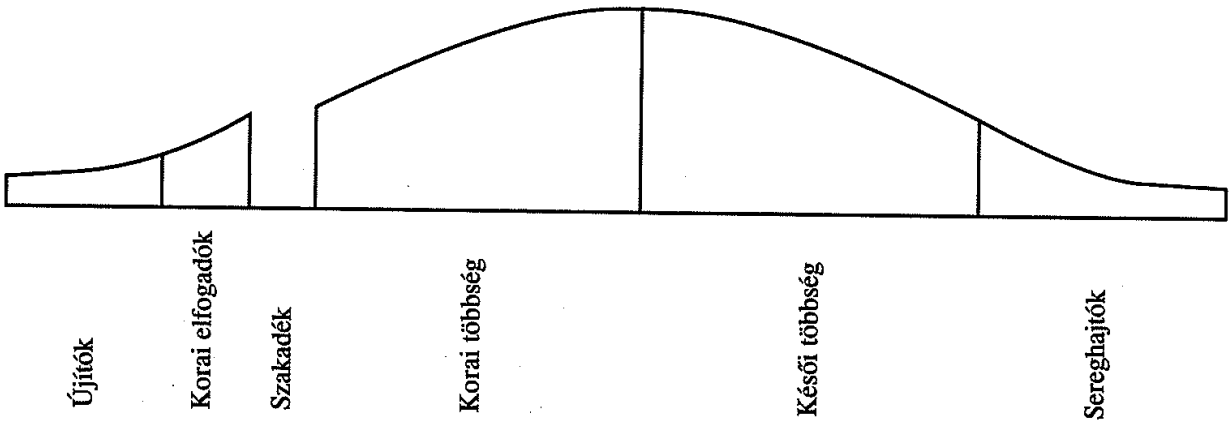
10.4.1. | Felfújt léggömb vagy folyamatosan és biztosan növekedő üzletág?

Az adatbányászat az utóbbi években rendkívül népszerűvé vált, sokan fogtak adatbányászattal kapcsolatos kutatásokba, fejlesztésekbe, vállalkozásokba, és mind többen forgalmaznak adatbányászatinak hirdetett szoftvertermékeket. Ezt látva sokan kérdezhetik: *„Vajon csak múló divatról van szó? Mennyire elfogadott mint technológia?"*

Természetesen az adatbányászat 1980-as évekre tehető feltűnése óta számos, a témával kapcsolatos híresztelés végezte kidurrant léggömbként, főképp azért, mert rengeteg ember hiszi, hogy az adatbányászat válik az adatokból történő tudáskinyerés legalapvetőbb eszközévé, hozzájárulva az üzleti élet vezetőinek stratégiai döntéseihez, a verseny kiélezéséhez és sok más csodálatos dologhoz.

Tudomásul kell venni azonban, hogy az adatbányászat is „csak" egy technológia. Mint minden más technológia esetében, itt is időre és erőfeszítésekre van szükség a kutatáshoz, a fejlesztéshez és a technológia beéréséhez; így valószínűsíthetően az adatbányászat is az életciklus következő lépésein fog keresztülmenni (10.9. ábra):

- **Újítók** – A kutatók felismerik az olyan új módszerek iránti igényt, amelyekkel bizonyos (adott esetben teljesen új) problémák megoldhatóvá válnak. Az új technológia kialakulása ezzel kezdetét veszi.
- **Korai elfogadók** – Az érdeklődés az újabb és újabb módszerek megjelenésével folyamatosan nő.
- **Szakadék** – A technológia főáramba történő befogadása előtt leküzdendő akadályokat, kihívásokat jelképezi.
- **Korai többség** – A technológia éretté, általánosan elfogadottá és széles körben használttá válik.
- **Késői többség** – A technológia már teljesen elfogadott, de az érdeklődés csökken a kezdeti (korábban a technológia megszületését is motiváló) problémák jelentéktelenné válása vagy a korábbitól eltérő új igények felmerülése miatt.
- **Sereghajtók** – A technológia idejétmúlttá válik és kihal.



10.9. ábra | Technológia elfogadási életciklus

Vajon melyik fázisnál tart jelenleg az adatbányászat? Sokan azonosítják a jelenlegi időszakot a szakadékkal. Ahhoz, hogy az adatbányászat teljesen elfogadottá váljék, további kutatásokra és fejlesztésekre van szükség azokon a területeken, amelyeket a könyvben a *„kihívás"* kulcsszóval jellemeztünk – hatékonyság és skálázhatóság, a felhasználói interaktivitás növelése, a háttértudás és grafikus megjelenítési technikák beépítése, szabványosított adatbányászati lekérdező nyelv kifejlesztése, hatékony mintakereső módszerek, összetett adattípusok kezelésének tökéletesítése, webbányászat stb.

Ahhoz, hogy az adatbányászat kimászhasson a gödörből, az adatbányászat létező üzleti megoldásokba történő *integrálására* is összpontosítani kell. Manapság jelentősnek mondható a választék általános jellegű adatbányászati rendszerekből. Ezek azonban sok esetben csak speciálisan képzett szakértők számára bizonyulnak használhatónak, akik járatosak az adatbányászati szakzsargonban és technikákban: a társítási szabályok, az osztályozás és a klaszterezés. Így ezek csak nehezen használhatók az üzleti élet vezetői és a nagyközönség számára. Ráadásul ezeket a rendszereket *horizontális megoldásként* általában az üzleti élet szinte bármilyen területén bevethetőre tervezik, és nem törekednek a *feladatspecifikus* adatbányászati megoldásokra. Mivel a hatékony adatbányászathoz szükség van az üzleti logika zökkenőmentes beépítésére, ezért nem várható, hogy egy általános célú adatbányászati rendszer olyan sikereket érjen el az üzleti életben, mint ahogy azt az

általános célú relációs adatbáziskezelő-rendszerek tették az üzleti tranzakciókezelés és lekérdezés-kiszolgálás terén.

Számos adatbányászattal foglalkozó kutató és fejlesztő úgy tartja, hogy az adatbányászat számára a *vertikális megoldások* kialakítása, azaz az üzletág-specifikus, mélyreható üzleti logika adatbányászati rendszerekbe történő integrálása jelentheti a kiutat. Mivel egyre több vállalat halmoz fel adatokat a weben vagy *e-kereskedelem* formájában folytatott üzleti folyamatai során, így ezek a vállalkozások kézenfekvő lehetőségeket nyújtanak az adatbányászat számára. Éppen ezért megvizsgáljuk, hogy milyen e-kereskedelem-specifikus adatbányászati megoldások kialakítására van lehetőség.

Jelenleg testreszabottabb rendszerek kialakítására van szükség a marketingkampányok irányítása *(e-marketing)* terén. Ideális esetben ezek a zárthurkú rendszerek egyesítik az ügyféladatok elemzését (OLAP- és adatbányászati technológiák beágyazott formában egy felhasználóbarát felület alatt), az ügyfélprofilok kezelését (vagy *ügyfélcsoportok kialakítását*), a marketingkampányok levezetésének támogatását és azok elemzését.

Ezek a rendszerek egyre nagyobb arányban használják az adatbányászati megoldásokat az **ügyfelekkel való kapcsolattartás (CRM, Customer Relationship Management)** színvonalának fejlesztésére, hiszen így a cégek sokkal személyreszabottabb szolgáltatásokat nyújthatnak ügyfeleik részére, szemben a tömegmarketinggel. A webáruházakban megfigyelhető szörfözési és vásárlási minták tanulmányozásával (például a vásárló klikkelési szokásaiból – *klikkfolyamából* – nyerhető információk elemzésével) a cégek többet tudhatnak meg egyes vásárlókról vagy vásárlói csoportokról. Ez az információ pedig egyszerre szolgálja mind a cég, mind pedig a vele kapcsolatban álló ügyfelek javát. Így például a vevőkről kialakított pontosabb modellek alapján jobban megérthetik a vásárlók igényeit. Ezen igények kiszolgálása további sikerekhez vezethet a termékek keresztértékesítésében, a rokon jellegű termékek forgalmazásában, a kereskedelmi akciókban, a személyre szabott reklámozásban, nagyobb választói kosarakban és a vevőkör megtartásában. A különböző vásárlói profilokhoz illeszkedő hirdetések kevésbé zavarják a fogyasztókat, nem annyira érzik felesleges zaklatásnak a postaládájukban megjelenő reklámanyagokat. Ezek az intézkedések alapvető megtakarításokat eredményezhetnek a cégek számára. Ezzel együtt a vevő is jól jár, hiszen nagyobb eséllyel értesül az őt érdeklő akciókról, termékekről, ami így kevesebb időveszteséggel és nagyobb megelégedettséggel társul. A vásárlóra szabott hirdetések nem merülnek ki levelekben vagy a webáruházak címlapján elhelyezett hirdetésekben. A jövőben a digitális televízió, az on-line könyvek és újságok szintén speciálisan az adott demográfiai és egyéb tulajdonságokkal jellemezhető néző vagy nézőcsoport számára tervezett és válogatott hirdetéseket közvetíthetnek.

Fontos megjegyeznünk, hogy az adatbányászat csak egy része az integrált megoldásoknak. A többi összetevőnek, mint például az adattisztítás és az adatintegrálás, OLAP, felhasználói biztonság, készlet- és megrendelés-nyilvántartás, termékmenedzsment stb. szintén a helyükre kell kerülniük.

10.4.2. | Adatbányászat: menedzsereknek vagy mindenkinek?

Az adatbányászat minden bizonnyal sokban hozzájárul ahhoz, hogy a cégvezetők pontosabb képet alkothassanak a piacról és saját üzletáguk működéséről. Mindazonáltal megkérdezhetjük, hogy *az adatbányászat vajon csak a vezetőkre, vagy mindenkire tartozik?*

Mivel egyre több és több adat érhető el a weben keresztül, vagy akár csak a saját adattárolóinkon is, nagy valószínűséggel mindenkinek szüksége lesz az adatbányászat eszközeire a hozzáférhető adatok alaposabb megértéséhez, és ezen keresztül a saját munkájának és mindennapi életkörülményeinek javításához. A jövőben várhatóan egyre több és nagyobb teljesítményű, felhasználóbarát, változatos és anyagilag elérhető adatbányászati rendszer vagy komponens jelenik meg. Így valószínűleg egyre többeknek lesz igénye és lehetősége az adatbányászatra. Más szavakkal, nem valószínű, hogy az adatbányászat felhasználói köre a tradicionálisan tudás alapú réteget alkotó menedzserekre és üzleti elemzőkre szűkülne. Ehelyett egyre inkább mindenki számára hozzáférhetővé válik.

Vajon mit kezdhetek otthon az adatbányászattal? Az adatbányászat rengeteg személyes felhasználási lehetőséget kínál. Például lehetőséget ad a család kórtörténetének tanulmányozására, így az esetlegesen genetikailag kódolt orvosi, egészségügyi tényezők, mint a rákra való hajlam vagy kromoszómaproblémák nagyobb valószínűséggel derülhetnek ki. Ezek az információk segíthetnek egy egészségesebb életstílus kialakításában. A jövőben megoldhatóvá válhat az egyén számára a vele kapcsolatban álló cégek adatainak tanulmányozása, amin keresztül értékelheti szolgáltatásaik színvonalát és így megalapozottabban választhat a szolgáltató cégek palettájáról. Lehetőségünk lesz az elektronikus leveleink szöveg alapú tartalmi bányászatára, vagy egy automatikus osztályozási rendszer kialakítására, ami rendezi a postaládába érkező leveleinket. Pénzügyi beruházásainkat megelőzheti a tőzsdei adatok és a vállalatok teljesítményének alaposabb elemzése. A webáruházak kínálatában megtalálhatjuk a számunkra legkedvezőbb termékeket vagy állásajánlatokat. Így az adatbányászat átléphet a korábban említett szakadékon, és a személyi számítógépek mind fokozottabb elterjedésével és az adatok weben keresztüli elérhetőségével az adatbányászat várhatóan egy mindenki számára hozzáférhető „kéziszerszámmá" válik.

Ezek szerint értenem kell majd az adatbányászati rendszerek belső működéséhez és az abban használt algoritmusok trükkjeihez ahhoz, hogy fel tudjam használni saját céljaimra? Hasonlóan a televíziókészülékekhez, számítógépekhez vagy az irodai programcsomagokhoz, várhatóan *felhasználóbarát*, minimális tanulással működtethető eszközökkel lesz dolgunk. Ezen túlmenően számos „okos" szoftverrel találkozhatunk majd, amelyek implicit formában használnak adatbányászati módszereket. Gondoljunk itt például kifinomult webes keresőgépekre, vásárlókhoz alkalmazkodó webszolgáltatásokra, „intelligens" adatbázisrendszerekre, együttműködő lekérdezés-megválaszoló rendszerekre, elektronikus levél- és naptár-menedzserekre, „jegyüzérekre", amelyekbe eleve egy adatbányászati modul kerül beépítésre, bár a felhasználó erről közvetlenül nem értesül. Ezt nevezzük **„láthatatlan" adatbányászatnak**. A láthatatlan adatbányászat várhatóan igen elterjedtté válik majd a jövőben.

10.4.3. | Fenyegeti-e az adatbányászat az adatbiztonságot és személyes titkainkat?

A weben keresztül elérhető egyre nagyobb mennyiségű elektronikus formában rögzített információ és az egyre hatékonyabb adatbányászati rendszerek megjelenése kapcsán felmerülhet a kérdés, hogy *vajon fenyegetést jelent-e az adatbányászat a személyes titkainkra és adataink biztonságára?* Mint minden más technológia, az adatbányászat is felhasz-

nálható jó és rossz célokra. Mivel az adatbányászat segítségével olyan információkra is fény derülhet, amiket egyébként nehezen lehetne megtalálni, nem megfelelő felhasználás estén esetleg fenyegetheti személyes adatbiztonságunkat és titkainkat.

A legtöbb fogyasztó nem bánja, ha a cégek hozzáférnek bizonyos adataikhoz, amennyiben úgy gondolják, hogy ezzel a cégek a nekik nyújtott szolgáltatások színvonalát emelhetik. Például a bevásárlók általában örömmel regisztrálják magukat a helyi szupermarketben a hűségkártya programra, ha cserében árengedményekre számíthatnak.

Elgondolkozott-e valaha is azon, hogy milyen mennyiségű információ kerül rögzítésre önről, és mire használható mindez? Minden alkalommal, amikor hitelkártyáját vagy bankkártyáját, törzsutaskártyáját használja, vagy ilyenekre feliratkozik, remek lehetőséget kínál a személyes szokásokat leíró adatok begyűjtésére. Személyes adatok gyűjthetők miközben a hálózaton szörfözünk, internetes fórumokban nyilatkozunk, magazinokra fizetünk elő, videokazettát kölcsönzünk, belépünk egy klubba, kitöltjük egy verseny jelentkezési lapját, újszülött gyermekünkről szolgáltatunk információkat (hogy ingyenes termékmintákhoz, ajándékokhoz, utalványokhoz jussunk), vényre kapható gyógyszert vásárolunk, vagy bemutatjuk a TB-számunkat az orvosi rendelésen. Világos, hogy a rólunk könnyedén összegyűjthető információk nem korlátozódnak vásárlási szokásainkra, hanem híven tükrözik hobbinkat, pénzügyi, egészségügyi és biztosítási adatainkat. Amennyiben a következő alkalommal, amikor a fent említett műveletek bármelyikét végzi, és elgondolkodik egy pillanatra, talán olyan érzése támad, mintha a „Nagy Testvér" vagy a „Nagy Bankár", esetleg a „Nagy Üzlet" folyamatosan figyelne.

Miközben a személyes adatok gyűjtése a 10.4.1. alfejezetben leírtak szerint jól jöhet mind az ügyfél, mind pedig a cégek számára, ezzel együtt lehetőséget ad visszaélésekre is. Mi történik, ha az adatokat más célra is felhasználják, például a biztosítási cégek meghatározzák a zsírfogyasztásunkat az általunk vásárolt élelmiszerek alapján? Az egyik áruház például megpróbálta az egyik ügyfél hűségkártyájának adatait annak bizonyítására felhasználni, hogy az üzletükben megcsúszó és eleső vásárlójuk valójában súlyos alkoholista (a vásárolt alkoholmennyiség alapján). Bár az ügyet a bíróság ejtette, mégiscsak szemlélteti, hogy a rólunk láthatatlanul gyűjtött adatok hogyan fordíthatók ellenünk.

Miközben elmerengünk a fentieken, többek között az alábbi kérdések merülhetnek fel bennünk:

- *Amikor adatokat szolgáltatok egy cég számára, vajon fogják-e azokat a későbbiekben olyan célokra használni, amire ma még nem is gondolok?*
- *Eladják-e az adatokat más cégek számára?*
- *Megtudom-e valaha is, hogy mit rögzítettek rólam?*
- *Van-e jogom arra, hogy megtiltsam a cégek számára a tőlem származó személyes információk használatát?*
- *Van-e lehetőségem arra, hogy a tárolt adatokban előforduló hibákat kijavítsam? Mi történik, ha szeretném kitörölni, kiegészíteni, módosítani vagy naprakésszé tenni a rólam tárolt adatokat?*
- *Anonimizálják-e az adataimat, vagy beazonosíthatóak maradnak?*
- *Mennyire biztonságosak az adatok?*
- *Mennyire vonhatóak felelősségre az adatokat gyűjtő és tároló cégek, ha az adataimat ellopják vagy visszaélnek velük?*

Ezeket a kérdéseket persze nem könnyű megválaszolni. **„Tisztességes információkezelési gyakorlat"** címmel nemzetközi iránymutatásokat fektettek le a titokvédelemmel és az adatok gyűjtésével, felhasználásával, minőségével, nyitottságával, az egyéni részvétellel és az elszámoltathatósággal kapcsolatban. Ezek a következő alapelveket tartalmazzák:

- **Szándéknyilatkozat és felhasználási korlátozás** – Az adatok felhasználásának célját az adatgyűjtés időpontjában az érintettek tudomására kell hozni, és a begyűjtött adatok felhasználása nem lépheti túl a meghirdetett célokat. Az adatbányászat jellemzően az adatgyűjtések másodlagos célja. Megegyezés született a tekintetben, hogy egy mellékletként csatolt, az adatok estleges adatbányászati célú felhasználást is lehetővé tevő nyilatkozat az esetek többségében nem fogadható el az adattulajdonos szándékainak elégséges közreadásaként. Az adatbányászat feltáró jellegéből következően nem tudható előre, hogy milyen mintákat fedeznek majd fel; így afelől sem lehetünk bizonyosak, hogy azokat hogyan használják majd.
- **Nyitottság** – Az egyéneknek joguk van tudni arról, hogy milyen információkat gyűjtenek róluk, ki férhet hozzá az adatokhoz és azokat hogyan használják fel.

Milyen megoldások léteznek tehát, amik figyelembe veszik ezeket az alapelveket? A cégeknek lehetőséget kell biztosítaniuk arra, hogy az adatgyűjtés különböző részeiből az ügyfél kimaradhasson, és korlátozásokat definiálhasson az adatai felhasználásával kapcsolatosan. Így például: (1) A vásárló adatai semmilyen formában nem használhatók adatbányászatra. (2) A vásárló adatait felhasználhatják adatbányászati célokra, de az egyes adattulajdonosok személyazonossága, vagy bármilyen információ, ami alapján a személyazonosság kideríthető, megsemmisítendő. (3) Az adatok csak belső adatbányászati célra használhatók. És végül (4) Az adatok mind belső, mind külső cégek számára felhasználhatók.

Alternatív megoldásként a cégek biztosíthatják vásárlóik számára a pozitív hozzájárulás jogát, azaz az ügyfelek külön hagyhatják jóvá (be kell jelölniük) adataik adatbányászati célú felhasználását. Ideális esetben az ügyfelek részére egy ingyen hívható telefonszámon vagy a cég honlapján keresztül biztosítani kell a jóváhagyások megadásának és törlésének lehetőségét, valamint a saját személyes adataihoz való hozzáférést.

Mi a helyzet az adatbiztonsággal? Az adatbázisrendszerek kezdetben némi ellenállásba ütköztek, mivel sokan tartottak a nagyméretű, on-line hozzáférhető adatbázisokkal kapcsolatos biztonsági problémáktól. Azóta számos *adatbiztonsági fejlesztés* látott napvilágot, és bár néha előfordulnak „hacker betörések", az emberek általában biztonságban érzik magukat és elfogadják az adatbázis-kezelők által nyújtott előnyöket. Az adatbiztonság fejlesztésére szolgáló technikák között említhetjük az ún. *„vak aláírások"* használatát (ami nyíltkulcsos titkosítási rendszereken nyugszik), a *biometrikus titkosítást* (amikor például az egyén személyes információi a személy írisze vagy ujjlenyomata alapján kerülnek titkosításra), valamint az *anonim adatbázisokat* (amelyek lehetővé teszik több adatbázis összevonását, de korlátozzák a személyes adatokhoz való hozzáférést, míg a személyes információt titkosítva egy másik helyen tárolják).

Az adatbányászat tehát veszélyeztetheti titkainkat és adatbiztonságunkat. De amint azt láthattuk, folyamatosan újabb és újabb megoldásokat fejlesztenek ki a begyűjtött adatokkal való visszaélés megakadályozására. Ezen túlmenően az adatbáziskezelő-rendszerek

számos adatbiztonságot növelő technikát vonultatnak fel az adatbányászatra begyűjtött és az adatbányászatból származó adatok biztonságának védelmére. Bár elképzelhető, hogy a ma létező adatbányászati technikák némelyikének nem sikerül „átkelnie a szakadékon", az adatbányászat sikerre van ítélve a technika iránti óriási érdeklődésnek és igénynek köszönhetően. Ahogy a cégek és vásárlóik továbbra is felelősséget éreznek a titkos adatok védelmére és biztonságára szolgáló megoldások kidolgozásában, folytathatjuk az adatbányászat hasznának learatását a megtakarított idő, pénz és a felfedezett új tudás formájában.

10.5. | Trendek az adatbányászatban

Az adatok, adatbányászati feladatok és megközelítések sokszínűsége számos kutatási kihívás elé állítja az adatbányászatot. Az adatbányászati nyelvek tervezése, hatékony és hatásos adatbányászati módszerek és rendszerek kifejlesztése, interaktív és integrált adatbányászati környezetek építése, és az adatbányászati technikák nagy problémák megoldására történő alkalmazása mind-mind lényeges feladatok az adatbányászattal foglalkozó kutatók, rendszer- és alkalmazásfejlesztők számára. Ebben a fejezetben leírunk néhány adatbányászati trendet, melyek e kihívásokra próbálnak reagálni.

- **Alkalmazáskutatás** – A korai adatbányászati rendszerek főleg a cégek számára szerezhető versenyelőnyökre koncentráltak. Ahogy azonban az adatbányászat egyre népszerűbbé válik, növekvő mértékben használják más területeken is, például az orvostudományban, pénzügyi elemzésekben vagy a távközlésben. Ezen kívül az adatbányászat üzleti célú felhasználásainak kutatása tovább terjeszkedik, ahogy az e-kereskedelem és az e-marketing a kiskereskedelemben egyre elfogadottabbá válik. Mivel az általános jellegű adatbányászati rendszerek csak korlátozottan használhatók feladatspecifikus problémák megoldására, ezért a jövőben mindinkább alkalmazásorientált adatbányászati rendszerek megjelenésének lehetünk tanúi.
- **Skálázható adatbányászati módszerek** – Szemben a hagyományos adatelemzési módszerekkel, az adatbányászatnak nagy mennyiségű adat hatékony és – amennyiben lehetséges – interaktív kezelését kell megoldania. Mivel az összegyűjtött adatok mennyisége folyamatosan növekszik, az egyedi és integrált adatbányászati funkciók megvalósítására szolgáló skálázható algoritmusok mindinkább előtérbe kerülnek. Az adatbányászati folyamat hatékonyságának és ezzel együtt interaktivitásának növelésére szolgáló legfontosabb irányzat a megszorítás alapú bányászat. Ez a felhasználók által megadható megszorításokkal lehetővé teszi az adatbányászati rendszerek irányítását és az érdekes minták utáni kutatást.
- **Az adatbányászat integrálása adatbáziskezelő-, adattárház- és webes adatbázisrendszerekkel** – Az adatbázisok, adattárházak és a web az információ feldolgozó rendszerek főáramba tartozó képviselőivé léptek elő. Fontos annak biztosítása, hogy az adatbányászat alapvető adatelemzési építőelemként zökkenőmentesen integrálódjék az ilyen információfeldolgozási környezetekbe. Amint azt a 4.4. alfejezetben említettük, az adatbányászati rendszerek az ideális architektúrában szorosan csatolódnak adatbáziskezelő-, illetve adattárházrendszerekhez. A tranzakciókezelést, a lekérdezés-

feldolgozást, az OLAP- és on-line adatbányászatot mind-mind egy egységes keretrendszerben kell integrálni. Ez biztosítja az adatok elérését, az adatbányászati rendszer hordozhatóságát, skálázhatóságát, nagy teljesítményét, és egy többdimenziós adatelemzésre és adatfeltárásra is alkalmas környezet kialakítását.

- **Az adatbányászati nyelv szabványosítása** – A szabványos adatbányászati nyelv és egyéb szabványosítási erőfeszítések megkönnyíthetik az adatbányászati megoldások módszeres fejlesztését, javíthatják a különböző adatbányászati rendszerek és funkciók közötti hordozhatóságot, előremozdíthatják az oktatást és az adatbányászati rendszerek használatát az iparban és a társadalomban. A legfrissebb ilyen irányú erőfeszítés a Microsoft OLE DB for Data Mining. [Lásd A) függelék] A további ilyen irányú törekvésekről a 4.2.7. alfejezetben már szóltunk.
- **Vizuális adatbányászat** – A vizuális adatbányászat hatékony módja a nagy mennyiségű adatból történő tudáskinyerésnek. A vizuális adatbányászati eszközök módszeres vizsgálata és fejlesztése előmozdíthatja az adatbányászat adatelemzési eszközként történő használatát.
- **Összetett adattípusok bányászatára szolgáló új módszerek** – Amint arra a 9. fejezetben rámutattunk, az összetett adattípusok bányászata mérföldkő lehet az adatbányászat fejlődésében. Bár történtek előrelépések a helymeghatározási, multimédia-adatok, idősorok, sorozatok és szöveges adatok bányászatában, még mindig nagy a szakadék az ilyen alkalmazások iránti igény és a ma elérhető technológia között. További kutatásokra van szükség elsősorban az adatbányászati módszerek már meglévő adatelemzési eljárásokkal történő integrálására a fenti adattípusok esetében.
- **Webbányászat** – A 9.6. alfejezetben már megvitatásra kerültek a web bányászatával kapcsolatos főbb kérdések. Figyelembe véve a weben keresztül elérhető óriási mennyiségű információt és annak a mai társadalomban betöltött szerepét, a webes tartalmak, webszervernaplók bányászata és az interneten keresztüli bányászat az adatbányászat egyik legvirágzóbb területévé avanzsálhatnak.
- **Titokvédelem és információbiztonság az adatbányászatban** – Az adatbányászati eszközök, számítógépek és hálózatok használatának egyre nagyobb népszerűségével párhuzamosan szembesülnünk kell a titokvédelem és információbiztonság fontosságával. Újabb és újabb védelmi és biztonsági módszerek kifejlesztésére van szükség, miközben elő kell segíteni a megfelelő információkhoz való hozzáférést és azok bányászatát.

10.6. | Összefoglalás

- Számos testreszabott adatbányászati megoldás került kifejlesztésre olyan **tudományág-centrikus alkalmazásokra**, mint a gyógyszerbiológia és DNS-kutatások, a pénzügy, a kiskereskedelem és a távközlés. Ez a gyakorlat integrálja a tudományág-specifikus tudást az adatelemzési módszerekkel, és feladatspecifikus adatbányászati megoldásokat hoz létre.
- Jó néhány adatbányászati rendszer került kifejlesztésre az elmúlt 10 évben. Amikor egy adott feladatra keresünk megfelelő adatbányászati megoldást, a szóba jövő **rendszerek számos tulajdonságát** *többdimenziós nézőpontból kell értékelnünk*. Ezek kö-

zött szerepelnek az adattípusok, rendszertámogatási kérdések, adatforrások, adatbányászati funkciók és módszerek, az adatbányászati rendszer adatbáziskezelő- vagy adattárházrendszerrel történő szoros csatolása, a skálázhatóság, a grafikus megjelenítési eszközök és felhasználói felületek.

- A **vizuális adatbányászat** integrálja az adatbányászatot és az adatmegjelenítést a nagyméretű adathalmazokban lévő implicit, de hasznos összefüggések felfedezésére. A vizuális adatbányászat formái az *adatmegjelenítés, az eredmények megjelenítése, a folyamat szemléltetése és az interaktív vizuális bányászat*. A **hang alapú adatbányászat** hangjeleket használ a különböző adatminták és tulajdonságok felismerésére.
- Számos jól elterjedt **statisztikai módszert** javasoltak adatelemzésre, úgy mint a regressziót, az általánosított lineáris modelleket, a regressziós fákat, a varianciaanalízist, a vegyes faktoriális modelleket, a faktoranalízist, a diszkriminációelemzést, az idősoranalízist, a túléléselemzést és a minőségellenőrzést. A statisztikai adatelemző módszerek kimerítő tárgyalása túlmutat e könyv lehetőségein. Az érdeklődő olvasók az irodalomjegyzékben találhatnak utalásokat a különböző statisztikai elemző módszerekről szóló szakirodalomra.
- A kutatók törekedtek az adatbányászat **elméleti alapjainak** lefektetésére. Számos érdekes javaslat merült fel, ideértve az adatcsökkentést, a mintakinyerést, a valószínűség-elméletet, az adattömörítést, a mikroökonómiai megközelítést és az induktív adatbázisokat.
- **Az intelligens válaszadás** adatbányászati technikák segítségével elemzi a felhasználói lekérdezések szándékát, és általánosított, a kérdéshez kapcsolódó információkkal szolgál válaszként. Kiterjeszti a lekérdezésfeldolgozó rendszerek erejét és használhatóságát.
- Az adatbányászathoz hasonló új technológiák befogadása egy olyan életcikluson megy keresztül, melynek során a felmerülő akadályok és kihívások miatt létezik egy **„szakadék"** is, mielőtt az adott technológia bekerülne a főáramba.
- A titokvédelem és az adatbiztonság egyike az adatbányászat társadalmi vonatkozásainak. A titokvédelem egy lehetséges megvalósítása a **megtagadási politika** alkalmazása, amely lehetővé teszi a személyes adatok felhasználásának korlátozását, míg az **adatbiztonságot növelő technikák** biztosíthatják az információ biztonsági és titkosítási szempontú anonimizálását.
- **Az adatbányászati trendekben** további törekvés figyelhető meg az új alkalmazási területek és az összetett adattípusok kezelésének, az algoritmusok skálázhatóságának, a megszorítás alapú bányászat és a vizuális módszerek fejlesztésében, az adatbányászati és adatbáziskezelő-, illetve adattárházrendszerek integrációjában, valamint az adatbányászati nyelvek szabványosítása terén.

10.7. | Feladatok

10.1. Keressünk és ismertessük az *adatbányászat egy olyan alkalmazási területét*, amiről ebben a fejezetben nem esett szó. Részletezzük, hogy a választott területen az adatbányászat mely formáit tudnánk hasznosítani.

10.2. Tegyük fel, hogy egy adatbányászati rendszer beszerzésén gondolkodunk.
(a) Milyen különbségeket tudunk felsorolni az adatbányászati rendszer és a hozzá tartozó adatbáziskezelő- vagy adattárházrendszer csatoltsága (*laza, közepes* és *szoros*) tekintetében?
(b) Mi a különbség az *oszlop- és sorskálázhatóság* között?
(c) A fenti tulajdonságok közül melyek fontosak a rendszer kiválasztásánál?

10.3. Tanulmányozzunk egy létező, *kereskedelmi forgalomban lévő adatbányászati rendszert*! A többdimenziós szemléletet figyelembe véve vázoljuk fel a rendszer főbb tulajdonságait, áttekintve a kezelt adattípusokat, a rendszerarchitektúrát, az adatforrásokat, az adatbányászati funkciókat, módszereket, az adatbáziskezelő-, illetve az adattárházrendszer közötti csatolást, a skálázhatóságot, a vizuális szemléltető eszközöket és a grafikus felhasználói felületet. Tudnánk-e valamilyen módosítást javasolni, ami valamilyen szempontból javítaná a rendszert? Fel tudjuk vázolni, hogy a javasolt módosítás miként valósítható meg?

10.4. Tegyünk javaslatot néhány *hang alapú adatbányászati módszer* implementálására! Lehetséges-e a hang és *vizuális alapú* módszerek integrálása oly módon, hogy az kellemesebbé és hatékonyabbá tegye az adatbányászatot? Van-e lehetőség videós adatbányászati módszerek kifejlesztésére? Javasoljunk néhány megoldást, és elemezzük, hogy miként lehet hatékonyabbá tenni az ilyen integrált audiovizuális bányászati módszereket!

10.5. Az általános célú számítógépek és az alkalmazásfüggetlen relációs adatbáziskezelőrendszerek piaca óriási növekedést mutatott az elmúlt évtizedekben. Ennek ellenére sokan gondolják úgy, hogy az *általános célú* adatbányászati rendszereknek nincs jövője. Mi mit gondolunk erről? Az *alkalmazási területtől független*, vagy inkább annak szempontjait figyelembe vevő, adott célra kihegyezett adatbányászati eszközök fejlesztését látnánk szükségesnek? Mik az érveink?

10.6. Miért fontos az adatbányászat *elméleti alapjainak* lefektetése? Nevezzük meg és részletezzük az adatbányászat megalapozására eddig javasolt főbb alapvetéseket! Kommentáljuk, hogy miként elégítik ki (vagy nem) ezek az elvek egy elméleti, ideális adatbányászati keretrendszerrel szemben támasztott követelményeket.

10.7. Mi a különbség a *kérdés közvetlen és intelligens megválaszolása* között? Tegyük fel, hogy egy felhasználó egy bizonyos üdülőhelyen található szállodák ára, címe és színvonala felől érdeklődik. Mutassunk példákat arra, hogy egy ilyen lekérdezés miképp válaszolható meg közvetlen és intelligens válaszadással.

10.8. Tegyük fel, hogy a helyi bank adatbányászati rendszert üzemeltet. A bank tanulmányozta bankkártya-használati szokásainkat. Észrevéve, hogy számos tranzakciót bonyolítunk lakásfelszerelési áruházakban, a bank elhatározza, hogy felveszi velünk a kapcsolatot, és javasol néhány kedvező lakásfelújítási hitelkonstrukciót.
(a) Vitassuk meg, hogy ez miképp befolyásolja *titoktartással kapcsolatos jogainkat*!
(b) Vázoljunk fel egy másik esetet, ahol szerintünk sérülhetnek a magánélet titkai!
(c) Milyen példákat tudnánk adni, ahol az adatbányászat előnyös lehet a társadalom számára? Tudunk-e olyan esetet említeni, ami a társadalom számára esetleg káros lehet?

10.9. Az adatbányászati rendszerekkel és alkalmazási területeikkel kapcsolatos jelenlegi ismereteink alapján várhatjuk-e, hogy az adatbányászati piac megélénkül? Mik az

adatbányászati kutatások és fejlesztések szűk keresztmetszetei? Vajon a jelenlegi adatbányászati szemléletmóddal nagymértékben növelhető az adatbányászati rendszerek részesedése az alkalmazások piacán? Ha nem, milyen kiutakat javasolnánk?

10.10. Tanulmányaink alapján jelöljünk meg néhány olyan lehetséges *új mérföldkövet* az adatbányászati kutatások területén, amelyek még nem kerültek említésre a könyvben.

10.8. | Irodalom

Számos, az adatbányászat alkalmazási területeit taglaló könyvvel találkozhatunk a piacon. Az adatbányászatnak a bioinformatika (ideértve az orvosbiológiai és DNS-adatok elemzését) területén használt módszereibe és algoritmusaiba nyújt betekintést Gusfield [Gus97], Waterman [Wat95], Baldi és Brunak [BB98], valamint Baxevanis és Ouelette [BO98]. Pénzügyi adatok bányászata és pénzügyi folyamatok modellezése tekintetében érdekes olvasmány lehet Benninga és Czaczkes [BC97] és Higgins [Hig97]. Az adatbányászat kiskereskedelemi és az ügyfélkapcsolatok (CRM) terén történő alkalmazásáról szól Berry és Linoff [BL99], valamint Berson, Smith és Thearling [BST99] könyve. Az adatbányászat távközlési vetületei kapcsán Mattison [Mat97] könyve lehet érdekes. Chen, Hsu és Dayal [CHD00] egy skálázható, OLAP- és adattárházrendszerrel együttműködő távközlési hálózati forgalom elemzésére alkalmas eszközt mutat be. Valdes-Perez [VP99] a tudásfeltárást célzó ember-számítógép együttműködés alapelveit tárgyalja könyvében.

Az adatbányászattal foglalkozó könyvek jelentős része tartalmaz bevezető jellegű adatbányászati rendszer- és termékleírásokat. Goebel és Gruenwald [GG99] könyve áttekinti az adatbányászatra és tudáskinyerésre alkalmas szoftvereket. Az egyes adatbányászati rendszerekről részletes információt kaphatunk az adott terméket fejlesztő cég honlapján, a használati utasításokban, illetve az adatbányászattal és adattárházakkal foglalkozó szaklapokban. A jelen fejezetben bemutatott adatbányászati rendszerek honlapjainak címe: IBM IntelligentMiner: *http://www-4.ibm.com/software/data/iminer;* SAS Enterprise Miner: *http://www.sas.com/software/components/miner.html*; SGI MineSet: *http://www.sgi.com/software/mineset*; ISL Clementine: *http://www.isl.co.uk/clem.html*; DBMiner: *http://www.dbminer.com*. Mivel az adatbányászati rendszerek és funkcióik rendkívül gyorsan fejlődnek, ezért nem célunk részletesen foglalkozni a könyv keretén belül ezekkel a rendszerekkel. Ezért amennyiben az Önök adatbányászati rendszerét kihagytuk volna, úgy elnézésüket kérjük.

A vizuális adatbányászattal foglalkozó népszerű könyvek között találhatjuk Tufte műveit [Tuf83, Tuf90, Tuf97]. Az adatok képi megjelenítésére szolgáló technikákat foglalja össze Cleveland [Cle93] könyve. Keim és Kriegel [KK94] kifejlesztett egy VisDB névre hallgató, adatbázis-feltárásra alkalmas többdimenziós képi megjelenítési módszerekre épülő rendszert. Keim konferenciaanyagaiban fogalja össze az adatmegjelenítéssel és a vizuális adatbányászati módszerekkel kapcsolatos módszereket *(http://www.infromatik.uni-halle.de/~keim/tutorials.html)*. Ankerst, Elsen, Ester és Kriegel [AEEK99] egy interaktív, érzékelés alapú, vizuális osztályozásra alkalmas megoldást írnak le.

Az adatelemzésre használt statisztikai módszerekkel számtalan könyv foglalkozik, így például: a Berthold és Hand [BH99] által írt *Intelligent Data Analysis*, Devore [Dev95]

Probability and Statistics in Engineering and the Sciences. 4. kiadás; Neter, Kutner, Nachtsheim és Wasserman [NKNW96] *Applied Linear Statistical Models*, Dobson [Dob90] *An Introduction to Generalized Linear Models*, Breiman, Friedman, Olshen és Stone [BFOS84] *Classification and Regression Trees*, Bates és Pinheiro [BP99] *Mixed Effects Models in S.*, Johnson és Wichern [JW92] *Applied Multivariate Statistical Analysis*. 3. kiadás, Huberty [Hub94] *Applied Discriminant Analysis*., Shumway [Shu88] *Applied Statistical Time Series Analysis;* és Miller [Mil81] *Survival Analysis.)*

Jelentős mennyiségű tudományos cikk foglalkozik az adatbányászat elméleti alapjaival. Mannila [Man97] könyvében az adatbányászat elméleti alapvetéseivel foglalkozó tanulmányokat gyűjti össze. Az adatcsökkentéses szemléletet foglalja össze a Barbara és társai által szerkesztett *The New Jersey Data Reduction Report* [BDF+97]. Az adattömörítés alapú megközelítésről a minimális hosszúságú leírás elvét (MDL, minimum description length principle) taglaló tanulmányokban olvashatunk, mint például Quinlan és Rivest [QR89], valamint Chakrabarti, Sarawagi és Dom [CSD98] könyvében. A mintakinyerésre alapozó elméletekről számos, a gépi tanulással és adatbányászattal foglalkozó tanulmányban találkozhatunk, mivel a mintakinyerésre használt módszerek között találhatjuk a döntési fa indukciót, a neurális hálózattal végzett osztályozást, társítási szabálykeresést, a szekvenciális mintakinyerést, a klaszterezést stb. A valószínűségelméleti szemléletmód a statisztikai irodalomban érhető tetten, így a Bayes-hálózatokról, és a hierarchikus Bayes-modellekről szóló tanulmányokban, amint az a 7. fejezetből is kiderült. Kleinberg, Papadimitriou és Raghavan [KPR98] mutatták be először a mikroökonómiai megközelítést, amelyben az adatbányászatot mint optimalizációs feladatot tekintik. Imielinski és Mannila [IM96] javasolták az adatbányászat induktív adatbázis-lekérdezésekre történő visszavezetését.

Az intelligens válaszadásra használható módszereket számos kutató tanulmányozta, így például Imielinski [Imi87], Cuppens és Demolombe [CD88], Motro és Yuan [MY90], valamint Chu és Chen [CC92]. Han, Huang, Cercone és Fu [HHCF96] az intelligens válaszadást tudásfeltárási technikák segítségével vizsgálták.

Gates *Business @ the Speed of Thought* [Gat99] című könyve az e-kereskedelemmel és az ügyfélkapcsolatok kezelésével foglalkozik, és az adatbányászat egy érdekes jövőbeli felhasználási lehetőségét is megemlíti. A technológia elfogadásának életciklusmodelljét és a szakadékon történő áthaladást taglalja Moore [Moo99]. Agrawal a Fifth International Conference on Knowledge Discovery and Data Mining (KDD'99) konferencián meghívott előadóként tartott beszédében szólt a technológia elfogadási ciklus és az adatbányászat kapcsolatáról. Az adatbányászat magántitkok védelmével, valamint az adatbiztonsággal kapcsolatos vetületeit számos cikk tárgyalja, így például Laudon [Lau96] (melyben a személyes információk eladására és vételére alkalmas szabályozott nemzeti információs piacra tesz javaslatot), Cavoukian [Cav98] (amely a tisztességes információkezelési gyakorlatról és a megtagadás lehetőségeiről szól), valamint Berry [Ber99] és Lohr [Loh99]. Agrawal és Srikant [AS00] a magántitok megőrzését garantáló adatbányászati módszereket tárgyalja.

A) FÜGGELÉK

A Microsoft OLE DB for Data Mining szabványa

Ez a függelék egy nem hivatalos, tömör bevezetést nyújt a Microsoft OLE DB for Data Mining adatbányászati szabványba.[1] Az **OLE DB for DM** szabványleírás az adatbányászati nyelv primitívjeinek szabványosítására tett jelentős lépés, ami várhatóan iparági szabvánnyá alakul. Úgy tervezték, hogy az adatbányászati (DM) kliensalkalmazások (vagy más szóval DM-fogyasztók) a legkülönbözőbb adatbányászati szoftvercsomagok által nyújtott DM-szolgáltatásokat (adatbányászati szolgáltatók) vehessenek igénybe rajta keresztül.

Az OLE DB for DM az adatbányászati folyamat absztrakt leírása. Az OLE DB-hez csatlakozó kiterjesztésként egy új **Data Mining Model** (DMM, adatbányászati modell) nevű virtuális objektumot és annak irányítására szolgáló utasításokat definiál. A DMM kezelése hasonló egy SQL-tábláéhoz. A DM-fogyasztók által a DMM-en végezhető három legfőbb művelet a következő:

(1) **DMM-objektum létrehozása** – DMM-objektumot a **CREATE** utasítás segítségével hozhatunk létre, ami a relációs adatbáziskezelő-rendszerek **CREATE TABLE** utasításhoz hasonló. A **CREATE** utasításban megadhatjuk a DMM oszlopait (azaz az adatbányászat során elemezni kívánt attribútumokat) és az adatbányászati szolgáltató által a megfelelő adatbányászati modell betanításához használni kívánt algoritmust. A választható DM-algoritmusok között említhetjük a döntési fákat, a klaszterezést (itt „segmentation"-nek hívják), és a regressziót (előrejelzésre). A **CREATE** utasítás nem határozza meg a DMM tartalmát (azaz az elsajátított grafikai struktúrát). A DMM mindaddig üres marad, amíg a következő művelet végrehajtásra nem kerül.

(2) **Betanításra szolgáló adatok modellbe történő feltöltése és a betanítási fázis indítása** – Az **INSERT** utasítást használhatjuk a modell betanítási adatokkal való feltöltésére, hasonlóan a relációs tábláknál megszokott **INSERT INTO** utasításhoz. Az **INSERT** utasítás hatására a DM-szolgáltató a DMM létrehozásakor megadott algorit-

1 | Az itt közölt információ a Microsoft Corporation OLE DB for Data Mining, Draft Specification, version 0.9 kiadványának 2000 februári verzióján alapul. Amennyiben további, itt nem szereplő részletekre is kíváncsi, kérem olvassa el a megadott dokumentumot vagy annak újabb kiadását.

A.1. táblázat | A „vevő" entitás jellemzésére beágyazott táblát is tartalmazó „Vevők" tábla

Vevőazonosító	Nem	Életkor	Árucikk_neve	Árucikk_mennyisége	Árucikk_típusa
1	N	40	CD-lejátszó	1	Szórakoztató elektronika
			TV	2	Szórakoztató elektronika
			Autóriasztó	1	Biztonságtechnika

mus segítségével feldogozza az újonnan betöltött adatot. A DMM a betanítási adatok helyett az így kapott modellt tárolja. Ezt hívjuk azután a DMM tartalmának.

(3) **Az adatbányászati modell használata** – A **SELECT** utasítással a DMM tartalma alapján előrejelzéseket végezhetünk, és a modell alapján kapott statisztikákat is böngészhetjük.

A bányászni kívánt adatokat egy relációs adatbázisban tárolt táblacsoport reprezentálja. Az olyan adatokat, amelyek egyetlen entitásra vonatkoznak, **esetnek** hívjuk. Az összes fontos esetet **esethalmaznak** nevezzük. Az OLE DB for DM a Microsoft Data Access Components (MDAC)-ban megtalálható Data Shaping Service-ben definiált **egymásba ágyazott táblák (vagy táblaoszlopok)** kezelésére is képes. Az ilyen egymásba ágyazott táblák segítségével az egy entitáshoz kapcsolódó adatrekordok halmazát tudjuk eltárolni. Például egy vásárlót egy *Vevőazonosító, Nem, Életkor* hármas ír le, míg a vásárló által megvett árucikkeket beágyazott táblában tároljuk (például *Árucikk_neve*, *Árucikk_mennyisége*, *Árucikk_típusa*), amint azt az A.1. táblázatban is megfigyelhetjük. Egy esethez több beágyazott tábla is tartozhat, és minden tábla változó számú sorból állhat. Egy eset fő sorát az **eset sorának** hívjuk. A beágyazott táblában lévő sorokat **beágyazott soroknak** nevezzük.

Lássunk néhány példát a fenti műveletek mindegyikére.

A.1. | DMM-objektum létrehozása

A.1. példa | **Osztályozást megvalósító adatbányászati modell alkotása.** A következő utasítás egy *Életkor* előrejelzésére alkalmas DMM-objektum oszlopait (illetve attribútumait), és a betanítási fázis során használandó algoritmust határozza meg.

```
CREATE MINING MODEL [Életkor-előrejelzés]
(
   [Vevőazonosító]                          LONG      KEY,
   [Nem]                                    TEXT      DISCRETE,
   [Életkor]                                DOUBLE    DISCRETIZED() PREDICT,
   [Vásárolt_áruk]                          TABLE,
   (
                [Árucikk_neve]              TEXT      KEY,
                [Árucikk_mennyisége]        DOUBLE    NORMAL CONTINUOUS,
                [Árucikk_típusa]            TEXT      RELATED TO [Árucikk_neve]
   )
)
USING [Decision Trees]
```

Az utasítás a következő információt tartalmazza: A *Vevőazonosítót* **KEY**-ként definiáltuk, tehát a mező egyedileg azonosítja a vevő eset sorát. Az *Életkor* nevű folytonos, de diszkretizált attribútumot a modell fogja meghatározni. A **DISCRETIZED()** kulcsszó utal arra, hogy az alapértelmezett diszkretizáló módszert használja a rendszer. Ehelyett a **DISCRETIZED** *(method, n)* kulcsszót is használhattuk volna, ahol a *method* az alternatív diszkretizáló módszert adja meg, míg a *Életkor* attribútum teljes terjedelmét *n* számú intervallumra osztjuk fel. A *Vásárolt_áruk* egy beágyazott tábla, mely az *Árucikk_neve* (ami a *Vásárolt_áruk* tábla kulcsa), az *Árucikk_mennyisége* és az *Árucikk_típusa* oszlopokból áll. A folytonos attribútumok eloszlását a DM-szolgáltatók felhasználhatják. Előre tudjuk, hogy az *Árucikk_mennyisége* normális eloszlású. További eloszlásmodellek is léteznek, mint például a **UNIFORM** (egyenletes), **LOGNORMAL, BINOMIAL, MULTINOMIAL, POISSON** stb. Az *Árucikk_típusa* **RELATED TO** *Árucikk_neve* kifejezés azt jelenti, hogy az *Árucikk_típusa* egy osztályba sorolja az *Árucikk_nevet*, mint ahogy például az *„szórakoztató elektronika"* osztályba sorolja a „TV"-t. További, a példában nem szereplő típusok az **ORDERED, CYCLICAL, SEQUENCE_TIME, PROBABILITY, VARIANCE, STDEV** és a **SUPPORT**. A **USING** kifejezés megmondja, hogy a szolgáltató a döntési fa modellt alkalmazza modell **INSERT** utasítással történő létrehozatalakor. Ezt az összetett utasítást az algoritmus futtatásához használt szolgáltatóspecifikus paraméterpárok követhetik. ■

A.2. példa | Társításos (asszociációs) adatbányászati modell létrehozása. A következő utasítás egy társítási szabályok keresésére alkalmas DMM-objektumot hoz létre:

```
CREATE MINING MODEL [Társításos modell]
(
   [Tranzakció-azonosító]                    LONG     KEY,
   [Vásárolt_áruk]                           TABLE    PREDICT,
   (
                        [Árucikk_neve]       TEXT     KEY
   )
)
USING [Saját társítási algoritmus] (Minimum_size = 3)
```

A **(Minimum_size = 3)** kulcsszó azt jelenti, hogy a modellben kizárólag olyan, gyakran együtt megvásárolt árucikkekre vonatkozó társítási szabályokra vagyunk kíváncsiak, ahol egy szabály legalább 3 árucikket tartalmaz. ■

A.2. | Betanítási adatok beszúrása a modellbe és a modell betanítása

A következő parancs megadja a betanításra szánt adatokat az *Életkor-előrejelzés* modell részére. A DM-algoritmus (melyet a DMM létrehozatalakor határoztunk meg) kerül meghívásra, ami aztán az adatok elemzése után generálja a modellt.

```
INSERT INTO [Életkor-előrejelzés]
(
   [Vevőazonosító], [Nem], [Életkor],
   [Vásárolt_áruk] (SKIP, [Árucikk_neve], [Árucikk_mennyisége], [Árucikk_típusa])
)
SHAPE
{
   SELECT [Vevőazonosító], [Nem], [Életkor] FROM Vevők
      ORDER BY [Vevőazonosító]
}
APPEND
(
   {SELECT [Vevő_ID], [Árucikk_neve], [Árucikk_mennyisége], [Árucikk_típusa]
      FROM Vásárlások ORDER BY [Vevő_ID]}
   RELATE [Vevőazonosító] TO [Vevő_ID]
)
AS [Vásárolt_áruk]
```

A DMM feltöltésének módja hasonlatos egy átlagos adatbázistábla feltöltéséhez. Az **INTO**-ban szereplő **SKIP** kulcsszó arra utal, hogy az adatforrásban van olyan oszlop, amit a DMM nem használ. Figyeljük meg, hogy a **SHAPE** utasítás hozza létre a *Vásárolt_áruk* nevű beágyazott táblát.

A.3. | A modell használata

A betanított modell egyfajta igazságtáblaként működik, azaz a DMM-ben minden egyes oszlop (attribútum) összes értékkombinációjához tartozik egy sor. Diszkrét attribútumok esetében az attribútum összes különböző értékét veszi figyelembe a modell. Diszkrét attribútumok esetében pedig az attribútum összes diszkrét értéke kerül a modellbe (az intervallumok középső értéke). Folytonos attribútumoknál a minimum, maximum és átlag értékeket veszi figyelembe a rendszer. Minden attribútumtípusra a „hiányzó" címke is eltárolásra kerül. Az előrejelzések elkészítése, illetve a különböző statisztikák lekérdezése során ezt a táblát lehet böngészni. Ebben az alfejezetben számos példán keresztül mutatjuk be, hogyan végezhetünk előrejelzéseket a DMM-mel, és hogyan kérdezhetjük le, illetve böngészhetjük az általános tartalmat.

A **SELECT** utasítást használjuk az előrejelzések elkészítésére, amint azt a következő OLE DB for DM utasítás is szemlélteti.

```
SELECT t.[Vevőazonosító], [Életkor-előrejelzés].[Életkor]
FROM [Életkor-előrejelzés]
PREDICTION JOIN
```

```
(
   SHAPE
   {
      SELECT [Vevőazonosító], [Nem] FROM Vevők ORDER BY [Vevőazonosító]
   }
   APPEND
   (
      {SELECT [Vevő_ID], [Árucikk_neve], [Árucikk_mennyisége],
         FROM Vásárlások ORDER BY [Vevő_ID]}
      RELATE [Vevőazonosító] TO [Vevő_ID]
   )
   AS [Vásárolt_áruk]
) AS t
ON  [Életkor-előrejelzés].Nem = t.Nem AND
      [Életkor-előrejelzés].[Vásárolt_árucikk].[Árucikk_neve] =
         t.[Vásárolt_árucikk] .[Árucikk_neve] AND
      [Életkor-előrejelzés].[Vásárolt_áruk].[Árucikk_mennyisége] =
         t.[Vásárolt_árucikk] .[Árucikk_mennyisége]
```

Az utasítás egy **PREDICTION JOIN**-t alkalmaz a DMM összes lehetséges esetének egy olyan meghatározott esethalmazra történő illesztéséhez, ahol az *Életkor* paraméter nem ismert. A **SELECT** utasítás az így előálló illesztésen végez műveleteket, eredményül az *Életkor* attribútum jósolt értékét adva minden egyes *Vevőazonosító* mellé. Figyeljük meg, hogy egy adott tesztsorozatra a **PREDICTION JOIN** az **ON** klózban található feltételeket kielégítő esetek egész halmazát találhatja. Ha ez megtörténik, akkor ezek egyetlen aggregált esetben kerülnek összevonásra, amely az *Életkor* attribútumra a lehető legjobb előrejelzést tartalmazza (vagy a legjobb előrejelzéseket a modell összes előrejelzésre kijelölt oszlopában). A **SELECT** utasítást használhatjuk a tesztsorozatokból származó ismert *Életkor* értékek kikeresésére a modell pontosságának ellenőrzésére.

A DMM összes lehetséges attribútumkombinációt tartalmazó igazságtábláját számos érték és statisztika lekérdezésére is használhatjuk. Például, ha egy *Hajszín* attribútummal egészítenénk ki az *Életkor-előrejelzési* modellünket, akkor a különböző hajszíneket a következő utasítással kérhetnénk le:

```
SELECT DISTINCT Hajszín FROM [Életkor-előrejelzés]
```

Hasonlóan, minden egyéb megvásárolható árucikk listája megkapható a következő utasítással:

```
SELECT DISTINCT [Vásárolt_árucikk].[Árucikk_neve] FROM [Életkor-előrejelzés]
```

Figyeljük meg, hogy a pont (.) operátort használhatjuk a beágyazott táblák egyes oszlopaira történő hivatkozásnál.

Az OLE DB for DM számos függvényt kínál az előrejelzések statisztikai jellemzésére. Például egy előrejelzett érték valószínűségét a **PredictProbability()** függvény használa-

távál tudhatjuk meg, amint azt a következő, az összes vásárló jósolt korát és annak valószínűségét egy táblázatban szolgáltató utasításban is megfigyelhetjük:

SELECT DISTINCT [Vevőazonosító], **Predict**(Életkor), **PredictProbability**([Életkor]), ...

Hasonlóképpen a **Cluster()** és a **ClusterProbability()** függvényeket használhatjuk az egyes klaszterekbe való tartozás valószínűségének megtekintésére, amint azt a következő utasításban is megfigyelhetjük:

SELECT DISTINCT [Vevőazonosító], [Nem], **Cluster() as** C, **ClusterProbability() as** CP, ...

Vevőazonosító	Nem	*C*	*CP*
1	N	3	0,37
2	F	5	0,26
⋮	⋮	⋮	⋮

ahol *C* annak a klaszternak az azonosítóját tartalmazza, amelyikhez az adott eset a legnagyobb valószínűséggel tartozik, és *CP* az ehhez tartozó valószínűség. A további, az oszlopnevet argumentumként bekérő függvények között említhetjük a **PredictSupport()**-ot, ami azon esetek számát adja meg, amelyek a jósolt oszlopértékeket adják; a **PredictVariance()**-t, a **PredictStdev()**-et, a **PredictProbabilityVariance()**-t és a **PredictProbabilityStdev()**-et. A **RangeMid()**, **RangeMin()** és **RangeMax()** függvények egy **DISCRETIZED** oszlop jósolt intervallum értékeinek a minimumát, maximumát és a középső értékét adják vissza.

A **PredictHistogram()** függvény használható egy jósolt vagy klaszterezett oszlop lehetséges értékeiből és a hozzájuk tartozó statisztikákból képzett hisztogram megjelenítésére. A hisztogram egy beágyazott tábla formájában jelenik meg. Például egy, a *Nem*-hez hasonló, diszkrét értékű attribútum esetében minden egyes esetre külön hisztogramot jeleníthetünk meg, amely a lehetséges *Nem* értékek előfordulását és valószínűségét mutatja. Ezt az információt a következő utasítás szolgáltatja:

SELECT [Vevőazonosító], **PredictHistogram**([Nem]) **as** GH, ...

	GH		
Vevőazonosító	Nem	$Support	$Probability ...
1	N	786	0,786
2	F	214	0,214
	N	672	0,672
	F	328	0,328
⋮	⋮	⋮	⋮

További lekérhető statisztikák: **$Variance**, **$Stdev**, **$ProbabilityVariance** és **$ProbabilityStdev**. Amennyiben a **PredictHistogram** argumentuma a **Cluster()**, akkor minden egyes esetre kapunk egy megalapozottságot, valószínűséget stb. mutató hisztogramot. Ezen túlmenően az OLE DB for DM-ben léteznek függvények a beágyazott táblák első és utolsó *N* sorának esetenkénti megtekintésére. Az ilyen függvények jól jönnek, ha az egyes esetekre jutó beágyazott táblák sorainak száma nagy.

A DMM-tábla az összes lehetséges attribútumkombináció és a hozzá tartozó jósolt érték mellett további csomópontokat (például döntési fát), szabályokat, formulákat vagy eloszlásokat is tartalmazhat. Ez az alkalmazott DM-algoritmustól függ. A tartalom egy XML-leírást tartalmazó karakterfüzér formájában tekinthető meg. Egy ilyen karakterfüzér értelmezése persze a kliens alkalmazás részéről megfelelő szakértelmet feltételez. Navigációs műveletek állnak rendelkezésre az irányított gráfként (például döntési fa) reprezentált modell tartalmának megjelenítésére. A felfedezett szabályokat PMML-ben (Predictive Model Markup Language) fejezik ki, és a modell tartalmának lekérdezésével jeleníthetők meg.

B) FÜGGELÉK

Bevezetés a DBMinerbe

A **DBMiner** egy, a kanadai Brit Kolumbiai Simon Fraser Egyetem Intelligens Adatbázisrendszerek Laboratóriumából származó adatbányászati rendszer, amit később a szintén kanadai Brit Kolumbiában alapított **DBMiner Technology Inc.** fejlesztett tovább. A termék az adatbányászati és tudáskinyerési területeken hosszú éveken keresztül végzett kutatómunka gyümölcse. Ez a függelék rövid betekintést nyújt a **DBMiner** rendszerbe.[1]

A **DBMiner** egy nagyméretű relációs adatbázisokból és adattárházakból történő többszintű tudás kinyerésére kifejlesztett interaktív, on-line analitikus adatbányászati rendszer. A **DBMiner** legfontosabb megkülönböztető jegye, hogy az OLAP-rendszereket ötvözi széles körű adatbányászati funkciókkal (karakterizáció, társításelemzés, osztályozás, előrejelzés és klaszterezés) [Han98, HCC98, HF96]. Ez a szoros integráció egy ígéretes, új adatbányászati módszertan, az **OLAM** (on-line analytical mining, on-line analitikus bányászat) kialakulásához vezetett, amelyben a rendszer többdimenziós nézeteken keresztül jeleníti meg az adatokat, és az adatbányászathoz olyan interaktív környezetet nyújt, amelyben a felhasználók dinamikusan válogathatnak az adatbányászati és az OLAP-funkciók között, és OLAP-műveleteket is végrehajthatnak (például lefúrás, kockázás, szeletelés, pivotálás) az adatbányászati eredményeken, illetve az OLAP-végeredmények elemzésére is használhatunk adatbányászati funkciókat.

A rendszer megkönnyíti a többszintű adatbázisok lekérdezéseken alapuló interaktív adatbányászatát, felhasználva a legmodernebb adatbányászati technikák sorát, mint például az OLAP alapú többdimenziós statisztikai elemzéseket, a hatékony mintakereső algoritmusokat, a fokozatos mélyítést kifinomultabb összefüggések bányászatára, a vizuális adatbányászatot, a metaszabály-vezérelt bányászatot, valamint az adat- és tudásszemléltetést. A **DBMiner** zökkenőmentesen integrálható a relációs adatbáziskezelő- és adattárházrendszerekkel, és így felhasználóbarát, nagyteljesítményű interaktív adatbányászati környezetet kínál.

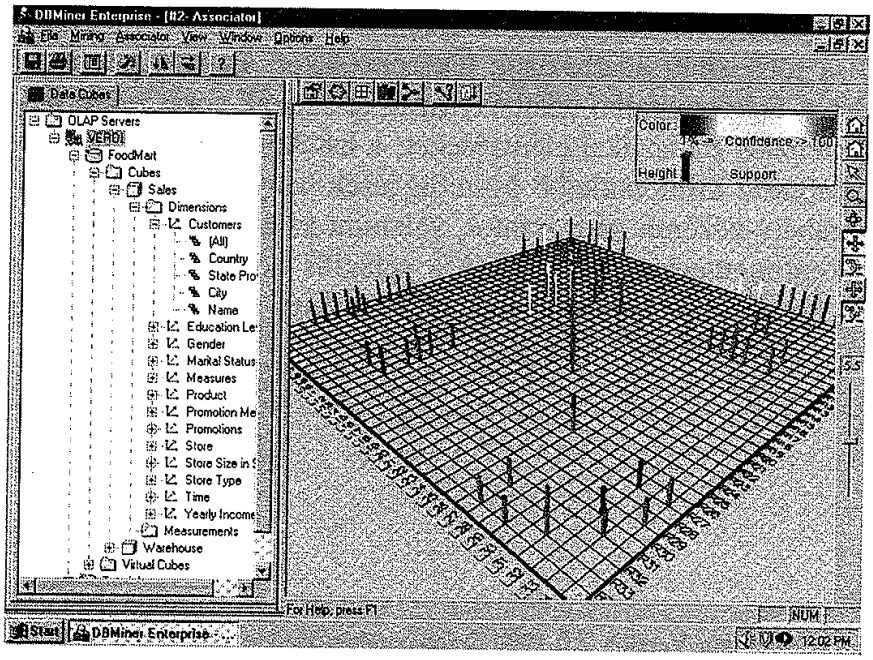
1 | A függelék a DBMiner Enterprise Edition Version 2.0 1999 decemberében kiadott változata alapján készült.

B.1. | A rendszer felépítése

A **DBMiner** felépítése a 2.24. ábrán javasolt on-line analitikus adatbányászati architektúrát követi, amelyeknél az adatokat egy relációs adatbázisból és/vagy egy adattárházból veszik, amelyeket majd egy többdimenziós adatbázisban egyesítenek (amely részben vagy egészében összevonható egy adatkockában), és azokon OLAM vagy OLAP alapú elemzést végez a végfelhasználó igényei alapján.

Az architektúra magjában egy OLAM-motor található, amely többdimenziós adatbázisokban képes on-line analitikus bányászatra, hasonlóan az OLAP-motorok által végrehajtott on-line analitikus feldolgozáshoz. A **DBMinerben** megtalálható OLAM-motor számos adatbányászati feladat elvégzésére képes, ideértve a fogalmak magyarázatát, társításelemzést, osztályozást, előrejelzést, klaszterezést és az idősorok elemzését.

Ennél is fontosabb, hogy a rendszer integrálja az OLAM- és OLAP-motorokat, melyek mindegyike képes az utasítások és lekérdezések on-line kiszolgálására egy grafikus programozói interfész felületen keresztül, és többdimenziós adatbázisok feldolgozására egy MDDB_API felület segítségével. Vegyük észre, hogy ebben az architektúrában az OLAM- és OLAP-motorok kölcsönhatásban állnak, hiszen az OLAM-motorral bányászhatjuk az OLAP által előállított eredményeket, míg az OLAP-motor feldolgozhatja az adatbányászatból származó eredményeket. Egy metaadatkönyvtár áll rendelkezésre az adatbázisséma, az adattárházséma, valamint a fogalmi hierarchiára vonatkozó információk tárolására. Ezekkel vezérlik aztán a többdimenziós adatbázishoz történő hozzáférést és az olyan dimenziófüggő OLAP-műveletek végrehajtását, mint a lefúrás és a szeletelés. A többdimenziós adatbázisok adattárházak szűrésével, és/vagy több ilyen forrás egy adatbázis API-n keresztül történő egyesítésével jöhetnek létre (amelyet jelenleg a Microsoft SQL Server 7.0 OLAP Manager támogat). A DBMiner grafikus felhasználói interfészét a B.1. ábrán láthatjuk.



B.1. ábra | A DBMiner grafikus felhasználói felülete

B.2. | Input-output

A **DBMiner** az adatokat egy SQLServer OLAP-adatkockából veszi, amit egy vagy több adatbázistáblából, adattárházból és/vagy egyéb adatformátumokból (mint például táblázatkezelőkből) építenek fel.

A kimenetet a rendszer többféle formában tárhatja elénk, a felhasználói igényektől és az adatbányászati feladat típusától függően. Az adatösszesítés és jellemzés eredményeként létrejövő kereszttáblázatok, általánosított szabályok, oszlopdiagramok, kördiagramok, görbék vagy egyéb grafikus kimeneti formák Microsoft Excel 2000-ben kerülnek előállításra. A társításelemzés eredményeként társítási szabálytáblázatok, asszociációs síkok és asszociációs gráf jön létre. Az osztályozás vizuálisan megjeleníti a döntési fákat és táblákat. A klaszterezés során (kétdimenziós elemzés esetén) „térképek" jönnek létre, amelyekben a klasztereket körvonaluk és színük jellemezi.

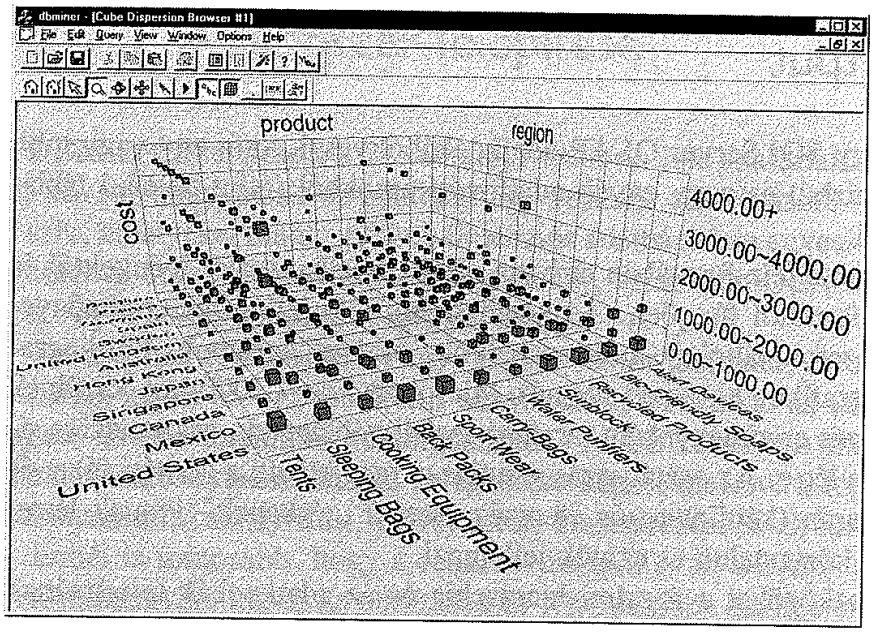
A rendszer tartalmaz eszközöket a fogalmi hierarchia és az adatkockák tartalmának megtekintésére. A fogalmi hierarchia a könyvtár/alkönyvtár rendszer felépítéséhez hasonlóan faként jelenik meg. Az adatkockák tartalmát háromdimenziós kockákként mutatják be, ahol minden részkocka a háromdimenziós kockán belül egy adott háromdimenziós intervallumon belüli mérőszámértékek összesítéséből származik. Ezen túlmenően a kétdimenziós táblázatokat kétdimenziós dobozdiagramok formájában láthatjuk, ahol minden dobozdiagram a megfelelő intervallumok adatainak szóródását jeleníti meg.

A rendszer egyik fő jellemzője a kimeneti tudás rugalmas manipulálhatósága fúrással, kockázással, és egyéb átalakítások segítségével. Így például egy különböző dimenziók és szintek kombinációja mentén végzett társításiszabály-keresés után bármely dimenzió mentén lefúrhatunk, hogy az új adathalmazban további társítási szabályokat találjunk.

B.3. | A rendszer által támogatott adatbányászati feladatok

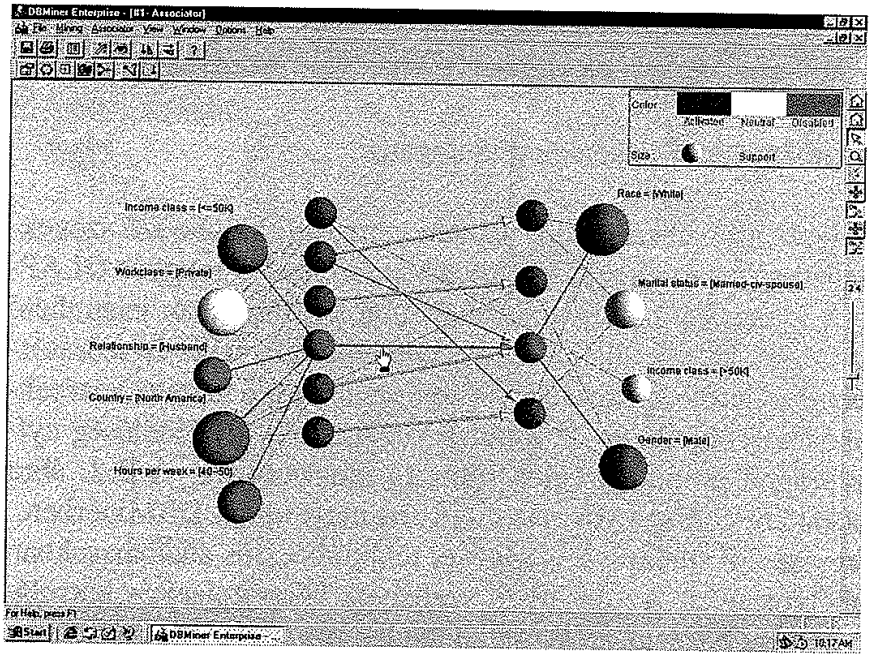
A **DBMiner** a következő adatbányászati feladatokat támogatja:

- **OLAP-elemző** – Ez a funkció több absztrakciós szinten keresztül, különböző szögekből képes bemutatni egy adatkocka tartalmát fúrás, kockázás, szeletelés és egyéb OLAP-műveletek segítségével. A kimenet többféle vizuális és grafikai formában jeleníthető meg. Ezen túlmenően az OLAP-adatok elláthatók a maximális, minimális és szórásértékekkel, valamint egyéb eloszlások szórásanalízisének eredményét mutató jelzésekkel. Általánosított adatokon további OLAP-műveletek végezhetők, így lehetőség nyílik további érdeklődésre számot tartó adatrégiók fúrására, kockázására. A B.2. ábra egy háromdimenziós kockaként ábrázolt összesített adathalmazt mutat DBMinerben.
- **Társítási szabályok keresése** – E funkció segítségével többdimenziós adatbázisokban kutathatunk társítási szabályok után. Az így nyert szabályok keresztpiaci és korrelációelemzésekre és egyéb célokra használhatók. Az adott elemzés szempontjából fontos szabályok keresését elősegíthetjük olyan szabályokra korlátozott kereséssel, amelyek

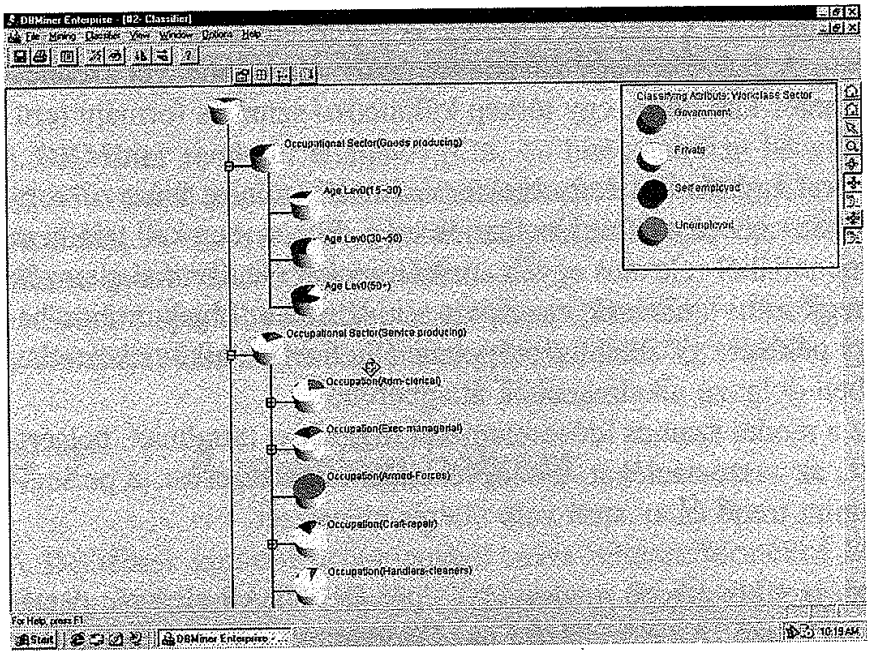


B.2. ábra | Összesített adatokat megjelenítő 3D kocka

kielégítik a felhasználó által megadott *metamintákat*, mint például *„szak(S:tanuló, X) ∧ P(S, Y) ⇒ érdemjegy(S, Y, Z)"*, ahol a *szak* és az *érdemjegy* predikátumok egy *tanuló* relációban értelmezett attribútumokra vonatkoznak; ahol az *S* változó jelképezi a tanulót, *P* egy predikátum, amit a *tanuló* egyik attribútumára állíthatunk be, és végül *X*, *Y* és *Z* a prediktumok lehetséges argumentumértékeit vehetik fel.. Amennyiben szükséges, akár fúrásokat is végezhetünk bármely dimenzió mentén, hogy különböző absztrakciós szinteknek megfelelő szabályokat találjunk. A B.3. ábrán társítási szabályokat ábrázoltunk labdagráfként, ahol a labdák adatokat, míg a nyilak szabályokat jelképeznek.

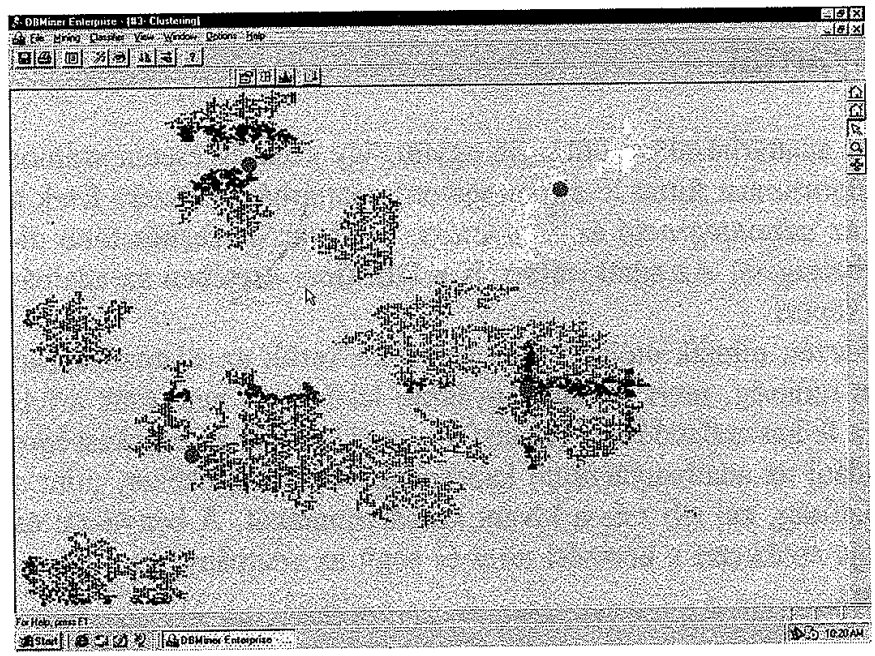


B.3. ábra | Társítási szabályok labdák és nyilak formájában



B.4. ábra | Döntési fa segítségével ábrázolt osztályozás

- **Osztályozás** – E funkció elemzi a betanítási folyamathoz tartozó adatokat (azaz olyan adatokét, amelyek osztálycímkéje már ismert), majd az adatok jellemzői alapján minden egyes osztályra modellt épít, amit aztán a tesztadatoknak megfelelően korrigál. Az így előállított modell döntési fák vagy osztályozási szabályok formájában mutatható be, és később kapott adatok osztályba sorolására, valamint az adatbázisban tárolt adatok jobb megértésére használható. A B.4. ábrán egy döntési faként ábrázolt adatosztályozási példát láthatunk.
- **Klaszterezés** – Ez a módszer a kiválasztott adatok halmazát klaszterekbe rendezi úgy, hogy biztosítsa a klaszterek közötti csekély hasonlóságot és maximalizálja a klaszteren



B.5. ábra | Klaszterezési eredményeket bemutató különböző alakú jelekből formált csoportok

belüli hasonlóságot. Sokdimenziós klaszterek esetében is végezhető klaszterezés többdimenziós adatbázisokban. A B.5. ábra egy adatklaszterezés végeredményét szemlélteti, amelyben a kétdimenziós klaszterek egy térkép különböző színű és formájú területeként tűnnek fel.

- **Előrejelzés** – E módszerrel egy kiválasztott objektumhalmaz bizonyos hiányzó vagy ismeretlen adatainak értékét vagy eloszlását jósolhatjuk meg. Ehhez meg kell találni az érdekelt attribútum szempontjából releváns egyéb attribútumokat (statisztikai elemzési módszerekkel), majd megjósolni értékének eloszlását a kiválasztott objektumhoz hasonló adatok alapján. Így például egy alkalmazott potenciális fizetése kikövetkeztethető a hasonló munkakörben dolgozó alkalmazottak átlagbéréből.
- **Idősorok elemzése** – Ez a modul számos idősorelemzési függvényt tartalmaz, beleértve a hasonlóságelemzést, az asszociációkat, a szekvenciális mintakeresést, a periodikusság vizsgálatát, valamint trend és az attól való eltérés elemzését.

A DBMiner 2.0-ás verziója a következő modulokból épül fel: OLAP-elemző, osztályozó, társításelemző és klaszterező. Az előrejelzést és idősorelemzést végző modulokat a jövőben megjelenő verziókban fogják megvalósítani.

B.4. | Feladat- és módszerválasztás

A **DBMiner** rendszer egy varázsló segítségével könnyíti meg a grafikus, ablakozó felhasználói felületen keresztül történő feladatok és a hozzájuk tartozó módszerek kiválasztását, vagy az adatbányászati eredményekkel történő összevetését úgy, hogy más dimenziókban és szinten folytassuk a bányászatot. A felhasználói inputok alapján az adatbányászati lekérdezés egy SQL-hez hasonló lekérdező nyelven (DMQL) kerül összeállításra, melyet a felhasználó még módosíthat a futtatás előtt.

B.5. | KDD-folyamat

Mivel a **DBMiner** rendszer adattárházakkal dolgozik együtt, így szükség esetén számos előszűrő és tudáskinyerő algoritmust futtat le a (támogatott) adattárházrendszer. Ezek a feladatok az *adattisztítás*, az *adatintegráció* és az *adatösszesítés* (az összesítés több dimenzió és szint mentén lehetséges). Az adatkiválasztás a DBMiner esetében az adatbányászati lekérdezés részét képezi.

A **DBMinerben** a felfedezett minták utókezelése beépítésre került az adatbányászati folyamatba. Erre azért van szükség, mert az adatbányászati lekérdezés nem csak a feladatspecifikus adatokat és a felhasználandó algoritmust árulja el, hanem az érdekességre jellemző mutatókat is (például az adatbányászattal kapcsolatos küszöbértékeket, mint a megalapozottság, megbízhatóság, zaj stb.) és a kívánt szabálymintákat. A bányászat és mintakiértékelés integrációja csökkenti a keresési teret és segít az adatbányászati folyamatra történő koncentrálásra.

B.6. | Fontosabb alkalmazások

A **DBMiner** rendszer általános célú, on-line analitikai bányászati rendszer, amely mind OLAP-, mind pedig relációs adatbázisokban és adattárházakban történő használatra is alkalmas. A rendszert közepestől nagyméretűig terjedő relációs adatbázisokon használták, és gyors válaszidőket mutatott.

Számos speciális adatbányászati prototípus, mint a GeoMiner, a MultiMediaMiner és a Weblog-miner kiegészítő csomag formájában illeszthető a DBMinerhez.

B.7. | A jelenlegi helyzet

A **DBMiner** kutatási prototípusból mára rendszertermékké lépett elő. Mindazonáltal a termékhez kapcsolódó kutatási és innovációs folyamatokban még mindig nagy szerep jut az egyetemi kutatólaboratóriumnak.

A **DBMiner** minimális hardverigénye egy 550 MHz-es Pentium processzor, legalább 64 Mbyte memóriával. A rendszer Windows NT alatt működik. A **DBMiner** közvetlenül köthető a Microsoft SQL Server 7.0-hoz, de a Microsoft SQL Server OLAP Manageren keresztül jónéhány relációs adatbáziskezelő-rendszerrel tud kommunikálni.

A **DBMiner 2.0** ingyenesen letölthető a *http://db.cs.sfu.ca/DBMiner* vagy a *http://www.dbminer.com* címről 90 napos próbahasználatra. Egyedi, csoportos és oktatási licencek a *http://www.dbminer.com* oldalon keresztül érhetők el.

Irodalomjegyzék

[**AAD+96**] Agrawal, S.–Agrawal, S.–Deshpande, P. M.–Gupta, A.–Naughton, J. F.–Ramakrishnan, R.–Sarawagi, S. On the computation of multidimensional aggregates. In *Proc. 1996 Int. Conf. Very Large Data Bases (VLDB'96)*, pages 506–521, Bombay, India, Sept. 1996.

[**AAP00**] Agarwal, R.–Aggarwal, C.–Prasad, V. V. V. A tree projection algorithm for generation of frequent item sets. In *Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining)*, 2000.

[**AAR96**] Arning, A.–Agrawal, R.–Raghavan, P. A linear method for deviation detection in large databases. In *Proc. 1996 Int. Conf. Data Mining and Knowledge Discovery (KDD'96)*, pages 164–169, Portland, OR, Aug. 1996.

[**ABKS99**] Ankerst, M.–Breunig, M.–Kriegel, H.-P.–Sander, J. OPTICS: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 49–60, Philadelphia, PA, June 1999.

[**AD91**] Almuallim, H.–Dietterich, T. G. Learning with many irrelevant features. In *Proc. 9th National Conf. Artificial Intelligence (AAAI'91)*, pages 547–552, Anaheim, CA, July 1991.

[**AEEK99**] Ankerst, M.–Elsen, C.–Ester, M.–Kriegel, H.-P. Visual classification: An interactive approach to decision tree construction. In *Proc. 1999 Int. Conf. Knowledge Discovery and Data Mining (KDD'99)*, pages 392–396, San Diego, CA, Aug. 1999.

[**AEMT00**] Ahmed, K. M.–El-Makky, N. M.–Taha, V. A note on „Beyond market basket: Generalizing association rules to correlation." *SIGKDD Explorations*, 1:46–48, 2000.

[**AFS93**] Agrawal, R.–Faloutsos, C.–Swami, A. Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. Foundations of Data Organization and Algorithms*, Chicago, IL, Oct. 1993.

[**AGGR98**] Agrawal, R.–Gehrke, J.–Gunopulos, D.–Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, Seattle, WA, June 1998.

[**AGI+92**] Agrawal, R.–Ghosh, S.–Imielinski, T.–Iyer, B.–Swami, A. An interval classifier for database mining applications. In *Proc. 1992 Int. Conf. Very Large Data Bases (VLDB'92)*, pages 560–573, Vancouver, Canada, Aug. 1992.

[**Agr96**] Agresti, A. *An Introduction to Categorical Data Analysis*. New York: John Wiley & Sons, 1996.

[**AGS97**] Agrawal, R.–Gupta, A.–Sarawagi, S. Modeling multidimensional databases. In *Proc. 1997 Int. Conf. Data Engineering (ICDE'97)*, pages 232–243, Birmingham, England, Apr. 1997.

[**AIS93a**] Agrawal, R.–Imielinski, T.–Swami, A. Database mining: A performance perspective. *IEEE Trans. Knowledge and Data Engineering*, 5:915–925, 1993.

[**Zia91**] Ziarko, W. The discovery, analysis, and representation of data dependencies in databases. In Piatetsky-Shapiro, G.–Frawley, W. J. editors, *Knowledge Discovery in Databases*, pages 195–209. Menlo Park: AAAI Press, 1991.

[**Zia94**] Ziarko, W. *Rough Sets, Fuzzy Sets and Knowledge Discovery*. New York: Springer-Verlag, 1994.

[**ZLO98**] Zaki, M. J.–Lesh, N.–Ogihara, M. PLANMINE: Sequence mining for plan failures. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 369–373, New York, Aug. 1998.

[**ZPOL97**] Zaki, M. J.–Parthasarathy, S.–Ogihara, M.–Li, W. Parallel algorithm for discovery of association rules. *Data Mining and Knowledge Discovery*, 1:343–374, 1997.

[**ZRL96**] Zhang, T.–Ramakrishnan, R.–Livny, M. BIRCH: An efficient data clustering method for very large databases. Ion *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, Montreal, Canada, June 1996.

[**ZTH99**] Zhou, X.–Truffet, D.–Han, J. Efficient polygon amalgamation methods for spatial OLAP and spatial data mining. In *Proc. 6th Int. Symp. Large Spatial Databases (SSD'99)*, pages 167–187, Hong Kong, July 1999.

[**ZXH98**] Zaïane, O. R.–Xin, M.–Han, J. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs. In *Proc. Advances in Digital Libraries Conf. (ADL'98)*, pages 19–29, Santa Barbara, CA, Apr. 1998.

Tárgymutató

Jiawei Han
Micheline Kamber

ADATBÁNYÁSZAT

Koncepciók és technikák

Az utóbbi néhány évtizedben mind az adatok előállításának, mind gyűjtésének lehetőségei igen gyorsan növekedtek. Elárasztanak bennünket az adatok – tudományos adatok, orvosi adatok, demográfiai adatok, pénzügyi adatok, üzleti adatok. Az ebben szerepet játszó tényezők között említendő a vonalkódok elterjedése, az üzleti, kutatási, közigazgatási tranzakciók számítógépesítése, az adatgyűjtő eszközök fejlődése, ami a szövegek és képek számítógépes beolvasó berendezéseitől a műholdas távérzékelőkig terjed. A World Wide Web globális információs rendszerként való használatának népszerűsége további rendkívüli adat- és információáradatot zúdított ránk. Az adatok robbanásszerű növekedése sürgős igényt váltott ki olyan technológiák és automatizált eszközök iránt, amelyek intelligens segítséget nyújtanak számunkra abban, hogy ezt a rendkívüli adattömeget hasznos információvá és tudássá alakítsuk.

Az adatbányászat az 1980-as évek végén jelent meg, jelentős felfutást ért el az 1990-es évek során, és várhatóan tovább virágzik az új évezredben is. Ez a könyv ennek a területnek átfogó képét mutatja be az adatbázis-kutató szemszögéből, amelynek során bevezet érdekes adatbányászati technikákba és rendszerekbe, tárgyalja az alkalmazásokat és kutatási irányokat. A könyv megírásához fontos motivációt jelentett az igény egy rendszerezett keret felépítésére az adatbányászat tanulmányozásához. Ez ugyanakkor nagy kihívást jelentő feladat ennek az igen gyorsan fejlődő területnek erősen multidiszciplináris jellege miatt.

Azt reméljük, hogy ez a könyv bátorítást ad a különböző háttérrel és tapasztalatokkal rendelkezők számára az adatbányászatra vonatkozó nézeteik kicserélésére úgy, hogy ezzel hozzájáruljanak ennek az izgalmas és dinamikus területnek a további fejlődéséhez és formálódásához.

PANEM
5900 Ft
www.panem.hu
ISBN 963545394-9
9789635 453948