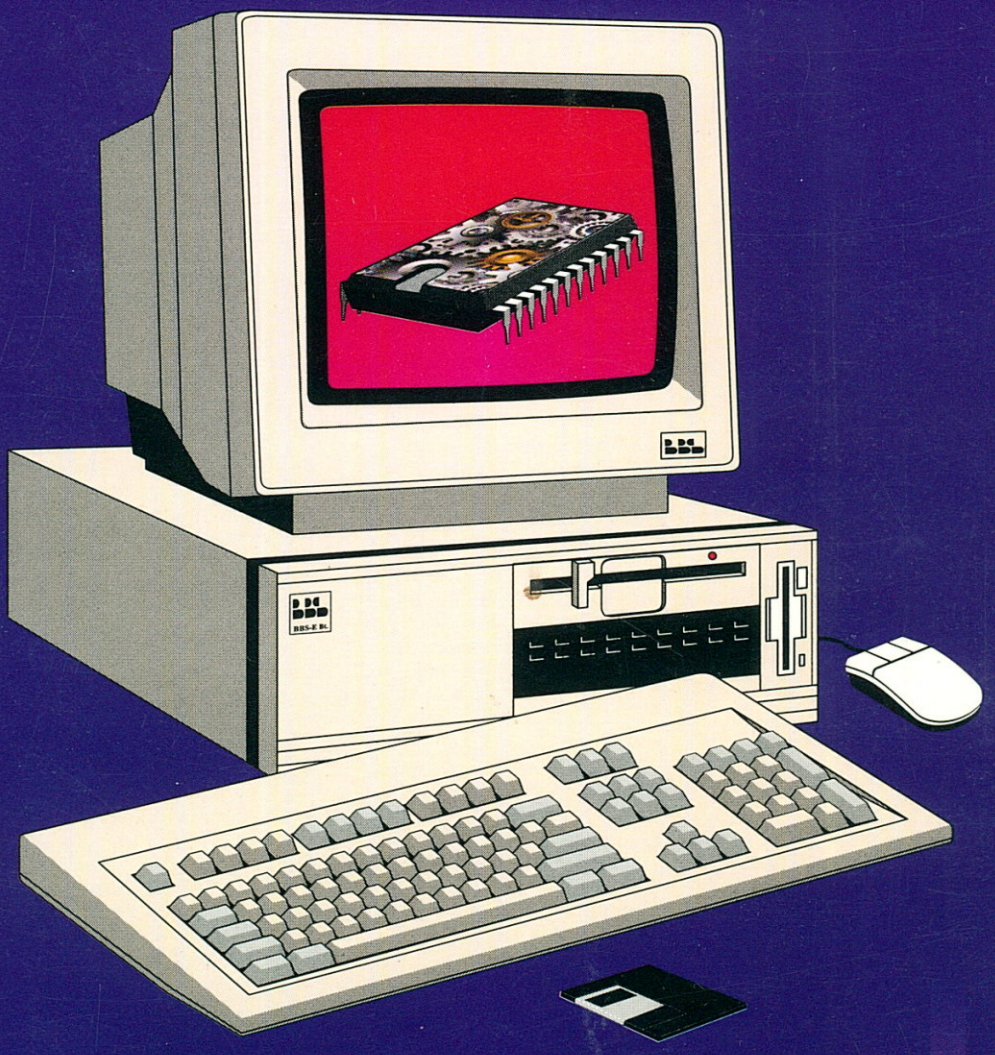
BBS-E Számítástechnikai és Könyvkiadó Betéti Társaság

Informatikai füzetek



dBase III+, Excel, Access

6. ADATBÁZIS-KEZELÉS

AJÁNLJUK:
ÖNÁLLÓ TANULÁSHOZ, OKJ-S KÉPZÉSEKHEZ (Számítógép-kezelő és Szoftverüzemeltető), SZAKKÖZÉPISKOLÁK, GIMNÁZIUMOK INFORMATIKA ÓRÁIRA, EGYÉB TANFOLYAMOKHOZ, STB.

VI.
Adatbázis-kezelés

Szerzők: Bártfai Barnabás (dBase III+)
Budavári Oszkár (Excel, Access)

ISSN 1418-8791
ISBN 963 03 5285 0

Kiadja a BBS-E Betéti Társaság
1630 Budapest, Pf. 21.
Felelős kiadó: a BBS-E Betéti Társaság ügyvezetője

Nyomtatta és kötötte a Kaposvári Nyomda Kft. – 180623
Felelős vezető: Mike Ferenc

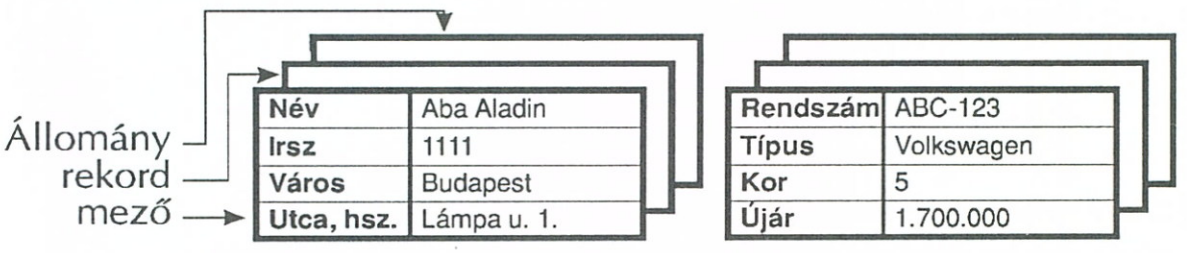
Tartalomjegyzék

1. Alapfogalmak

Mint azt már korábban is láttuk a számítógép elsősorban adatok feldolgozására és tárolására szolgáló eszköz. Az adatok tárolásával korábban már foglalkoztunk, de a feldolgozásról, annak igen sokrétű volta miatt részletesen még nem esett szó. Az adatok feldolgozásának egyik legelterjedtebb módja az adatbázis-kezelés.

Mivel az adatokat nem tárolhatjuk egyenként, külön-külön (hiszen akkor azok kezelése szinte lehetetlen volna), ezért szükséges őket valamilyen módon egy jól szervezett állományban elhelyezni. Az **adatállomány** tehát egy olyan összefüggő adathalmaz, melyben minden olyan adat megtalálható, amire egy bizonyos cél megvalósítása érdekében szükséges lehet. Mivel azonban több okból is szükségünk lehet különböző adatokra, így a számítógépünkön többféle egymástól független adatállományt is tárolhatunk. Néha azonban előfordulhat, hogy ezen adatok között kapcsolatot kell találnunk, s ilyenkor nélkülözhetetlen azok ismételt tárolása. Az egymással kapcsolatba hozható adatállományok összességét nevezzük **adatbázis**nak.

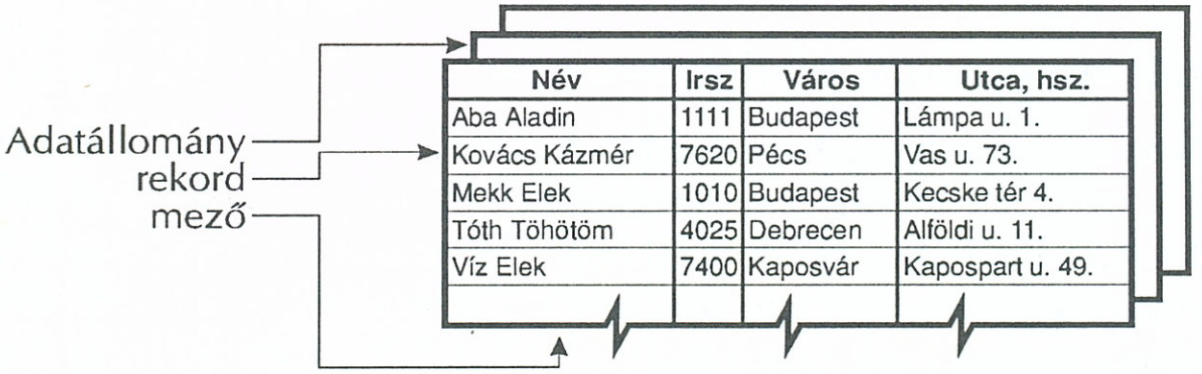
Mivel az önálló adatállományok igen sokféle adat tárolását egyszerre szolgálják, szükséges valamilyen szerkezet felépítése az adatállomá-



nyon belül. E szerkezet kezelését az adatbázis-kezelő program végzi, így a felhasználónak illetve a különböző alkalmazásoknak csupán a szerkezetét kell ismerniük, az adatbázis fizikai tárolásának módját nem. A felépített illetve felépítendő adatbázis szerkezetét még az adatok felvitele előtt meg kell határozni, így ilyenkor kell eldönteni azt, hogy milyen adatokat szeretnénk tárolni az adatállományokban.

Az adatállományok szerkezetére vonatkozóan azonban néhány kifejezés megismerése nélkülözhetetlen.

Az adatállományban **rekord**nak nevezzük az egy egységet leíró különböző jellemzőket. (Pl. egy rekord az adatállományban egy ember neve, a hozzátartozó irányítószámmal, városnévvel, illetve utca, házszámmal.)



Név	Irsz	Város	Utca, hsz.
Aba Aladin	1111	Budapest	Lámpa u. 1.
Kovács Kázmér	7620	Pécs	Vas u. 73.
Mekk Elek	1010	Budapest	Kecske tér 4.
Tóth Töhötöm	4025	Debrecen	Alföldi u. 11.
Víz Elek	7400	Kaposvár	Kapospart u. 49.

Mező alatt az adatbázis összes elemének egyazon jellemző adatát értjük. (Mező például a név.)

A kétdimenziós adatállományokat ezekből kifolyólag lehetséges táblázatos formában is ábrázolni, ahol a sorok jelképezik a rekordokat, az oszlopok pedig a mezőket.

Amennyiben egy adatállománnyal folyamatosan dolgozunk, a rekordok számát szaporítjuk, esetleg csökkentjük. Ezzel az adatállomány nagyobbá illetve kisebbé válik. A mezők számának megváltoztatása nem jellemző feladat, a szerkezetet még az adatok bevitele előtt célszerű kialakítani, hiszen egy utólagos módosítás az adatbázist használó programok megváltoztatásának szükségességét vonná maga után. Mindemellett egy adatállományban túl sok mező elhelyezése sem célszerű, ez ugyanis bonyolultabbá teheti az adatállományt.

Amikor létrehozunk egy adatállományt, tulajdonképpen azt határozzuk meg, hogy abban az adatok mely jellemzői kerüljenek tárolásra. Fontos tehát, hogy az összes szükséges jellemző tárolása megtörténjen, de felesleges és többszörös adatokat ne tároljunk. Az adatok többszörös tárolása ugyanis **redundanciá**hoz vezet, amely azon kívül, hogy megnöveli az adatállomány méretét, ellentmondásokat is eredményezhet. Ennek megfelelően tehát egy adatállományban azon adatokat, amelyek a tárolt jellemzők felhasználásával egyértelműen számítható soha se tároljuk el. (Például ne tároljunk egy ismerősről személyi számot, születési időt és kort, hiszen a személyi szám elegendő mind a születési idő, mind pedig a kor meghatározásához.)

1.1. Adatbázis szerkezetek

Egy-egy önálló adatállomány sok esetben nem tartalmaz elegendő adatot bizonyos információk meghatározásához, az összes adat egy állományban történő tárolása pedig bonyolult vagy megvalósíthatatlan. Ilyenkor válhat szükségessé az adatállományok együttes kezelése, amely valamilyen adatbázis szerkezetben realizálódik.

1.1.1. Hierarchikus adatbázis-szerkezet

Ebben a szerkezetben a DOS fa struktúrájához hasonló módon valósul meg az adatok tárolása. Az adatok között ún. szülő-gyermek kapcsolatot hoznak létre oly módon, hogy az adatoknak tetszőleges számú leszármazottja, de csak egy őse lehet. Ezt a szerkezetet személyi számítógépek esetén csak igen ritkán alkalmazzák, főként nagygépes környezetben fordul elő.

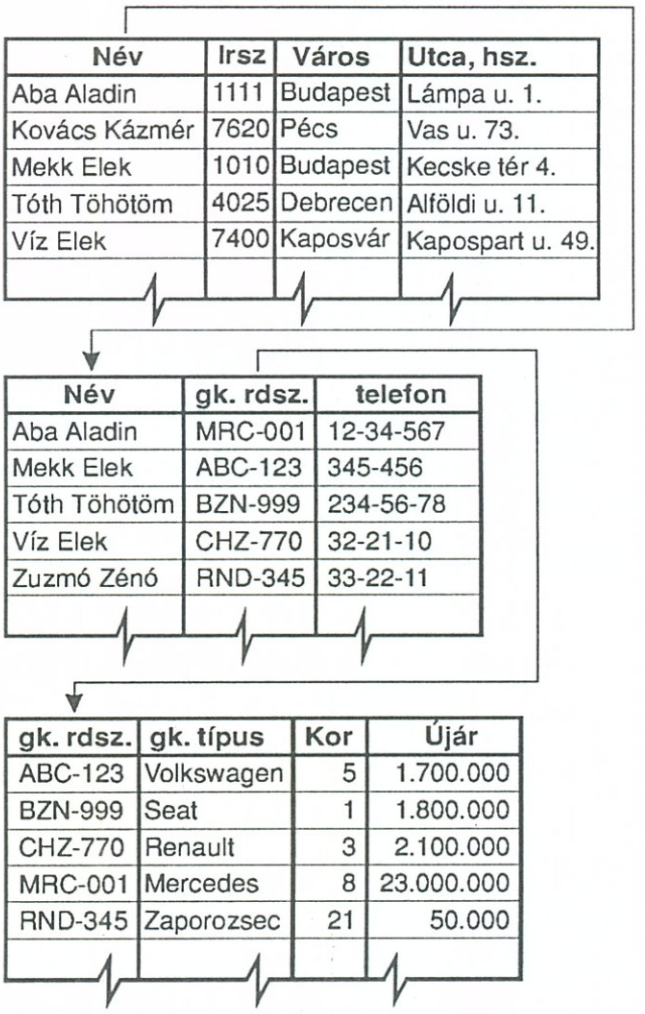
1.1.2. Hálós adatbázis-szerkezet

A hálós adatbázis nagyban hasonlít a hierarchikus szerkezethez, de itt nem csak több leszármazottja, hanem több őse is lehet az adatnak. A többnyire szintén nagygépes környezetben használt szerkezet hátránya, hogy a használathoz bonyolult láncolólisták szükségesek, így nagy a tárigényük.

1.1.3. Relációs adatbázis-szerkezet

Manapság legelterjedtebben a relációs adatbázis-szerkezetet alkalmazzák, hiszen ezen szerkezet könnyen illeszthető a korábban illusztrált táblázatos leírásra. A relációs adatbázisban az önálló állományok többnyire csak azon adatokat tartalmazzák, melyeknek közük van egymáshoz, s használatuk is többnyire egyidőben történik. A különböző jellegű, de mégis kapcsolatba hozható adathalmazokat önálló állományokban tároljuk, amely állományok között egy azonos adatot tartalmazó mező tartja a kapcsolatot. Ezt nevezzük **kapcsolómező**nek. Mivel ezen kapcsolómezőnek a kapcsolat irányában egyértelmű azonosítást kell megvalósítani, célszerű, ha a kapcsolt adatállománynak ez az azonosítómezője is. Az azonosítómező a rekordok egyértelmű azonosítását szolgálja, így ennek adattartalma ilyen esetben nem ismétlődhet. (A gyakorlatban azonban nem minden adatbázisban teljesül az a feltétel, hogy az adatbázis bármely mezője is egyértelműen azonosít, így ilyenkor egyértelmű kapcsolat nem hozható létre.)

Ha például az ábrán látható adatállományok felhasználásával a budapesti autótulajdonosok gépkocsi típusait szeretnénk megtudni, akkor ez a következőképpen valósítható meg:



Név	Irsz	Város	Utca, hsz.
Aba Aladin	1111	Budapest	Lámpa u. 1.
Kovács Kázmér	7620	Pécs	Vas u. 73.
Mekk Elek	1010	Budapest	Kecske tér 4.
Tóth Töhötöm	4025	Debrecen	Alföldi u. 11.
Víz Elek	7400	Kaposvár	Kapospart u. 49.

Név	gk. rdsz.	telefon
Aba Aladin	MRC-001	12-34-567
Mekk Elek	ABC-123	345-456
Tóth Töhötöm	BZN-999	234-56-78
Víz Elek	CHZ-770	32-21-10
Zuzmó Zénó	RND-345	33-22-11

gk. rdsz.	gk. típus	Kor	Újár
ABC-123	Volkswagen	5	1.700.000
BZN-999	Seat	1	1.800.000
CHZ-770	Renault	3	2.100.000
MRC-001	Mercedes	8	23.000.000
RND-345	Zaporozsec	21	50.000

Megnézzük, hogy az első adatállományban mely rekordok esetében szerepel a város mezőben Budapest. Az így megkapott nevekhez a második adatállományból rendszámokat rendelhetünk, amely rendszámokhoz tartozó gépkocsi típust a harmadik adatállomány fogja megmutatni. Az első és második állomány közti kapcsolat megteremtésekor a név mezőt, a második és harmadik állomány kapcsolásához pedig a gk.rdsz. mezőt használtuk kapcsolómezőként. Ilyen adatbázis létrehozásakor tehát célszerű mindig szem előtt tartani azt, hogy a kapcsolómező mindig legyen egyértelmű és lehetőleg rövid.

A relációs adatbázis szerkezetnek tehát az áttekinthető kezelésen túl az is előnye, hogy az önálló állományok mindig csak az adott feladatkörhöz tartozó adatokat tárolják. (Ez a hozzáférés korlátozása esetén különösen előnyös.)

További előny, hogy feleslegessé válik a kapcsolómezőn túli adatok ismételt tárolása, továbbá a néha együtt használt, de kevés összetartozást mutató adatállományok esetén az üres mezők felesleges tárolásának elkerülése. (Pl. ha az előző példában csak kevés embernek lenne autója, de egy állományban tárolnánk mindent, úgy rengeteg üres adatot kellene feleslegesen tárolnunk.)

1.2. Az adatbázis-kezelők feladatai

Akár közvetlenül adatbázis-kezelővel, akár saját felhasználói programmal is dolgozunk, bizonyos feladatokat az adatbázis kezelés során el kell látni. Ezen feladatok közé a következők tartoznak:

- Adatállomány létrehozása
- Új adatok bevitele
- Adatmódosítás
- Adatok törlése
- Adatok megjelenítése, nyomtatása
  - folyamatosan
  - adott szempontok szerint kiválasztva (keresés)
- Adatok rendezése
- Egyéb tevékenységek
  - operációs rendszer funkciók,
  - keresés, válogatás, karbantartás, számolások, stb.

2. dBase III+

Az adatbázis kezelő programok közt legelterjedtebben a dBase III + adatbázis kezelőt alkalmazzák. A dBase-t használata során többféle üzemmódban tudjuk működtetni. Egyrészt lehetőségünk van a DOS-hoz hasonlóan parancsok begépelésével utasítani a gépet különböző tevékenységek végzésére, másrészt használhatunk menüket, harmadrészt pedig futtathatunk programot.

2.1. A dBase kezelése

2.1.1. A program indítása

A program indításához először lépjünk be abba a könyvtárba, ahol a dBase állományai találhatók (pl. **CD DBASE**), majd gépeljük be a program nevét:

```
DBASE
```

A program indulásának módját egy ún. konfigurációs állomány határozza meg, amely a benne leírtaknak megfelelően indítja a dBase-t (ez olyan mint a DOS-ban az AUTOEXEC.BAT). A konfigurációs állomány CONFIG.DB névvel tárolódik, és benne szövegszerkesztővel módosítható dBase parancsok helyezkednek el. Ebbe az állományba célszerű tehát beírni azon beállítás-értékeket is, amelyeket amúgy rendszeresen begépelnénk a program használatának megkezdésekor.

2.1.2. Képernyő-területek

Amennyiben nem menüvezérelt üzemmódban dolgozunk, úgy a képernyő nagy részét információs illetve párbeszédes területként használhatjuk. Ide íródnak ki a parancsok eredményei, itt tudunk megadni további adatokat.

A képernyő legalsó sora az üzenetsor, amelyben a dBase üzenetei illetve rövid magyarázatai jelennek meg.

A felette található navigációs sor a mozgáshoz, menü kiválasztáshoz, illetve a munkához szükséges üzeneteket tartalmazza.

A navigációs sor felett inverzben látható a státuszsor, amely a végrehajtás alatt álló parancsot, az aktuális meghajtót, az aktív adatbázis nevét, az aktuális rekordsorszámot illetve össz rekordszámot (/ jellel elválasztva), a beszúrás üzemmódot (Ins), a CapsLock és NumLock gomb bekapcsolását jelzi.

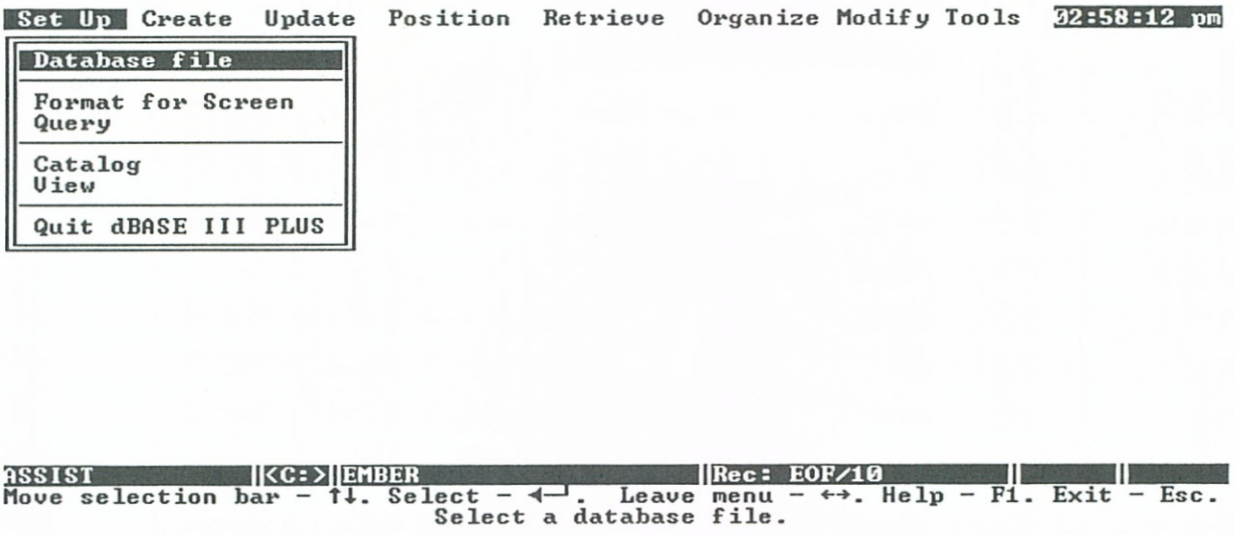
A státuszsor felett helyezkedik el az akciósor, amelybe a végrehajtandó parancsot gépelhetjük. Ennek a sornak az elején találjuk a dBase esetében pont formájában megjelenő promptot.

2.1.3. Menüvezérelt kezelés

A dBase menüvezérelt kezelését úgy érhetjük el, hogy begépeljük az

```
ASSIST
```

parancsot. A gép ezt követően a képernyő felső sorában megjelenít egy menüsort, melyből a kurzormozgató nyilak segítségével kiválaszthatjuk a megfelelő menüpontot. Az Enter billentyű hatására a gép a kiválasztott menüpontot aktivizálja, s ha szükséges, további információk bevitelét kéri.



2.1.4. Interaktív kezelés

A dBase elterjedt kezelési módja az a párbeszédes formájú kezelés, amit már a DOS-nál is megszoktunk. Ilyenkor a gépnek a különböző parancsokat billentyűzetről begépeljük, majd Entert nyomunk. Helyes begépelés eredményeként a parancs végrehajtódik, hibás parancs esetén pedig hibaüzenetet ad. Hibás parancsbevitel esetén – amennyiben nem tiltottuk le – a gép segítséget ajánl fel. Ha ezt igényeljük, nyomjunk Y-t, ha nem N-t.

A dBase prompt-ja, azaz visszajelző karaktersorozata egy pont. Ennek megjelenése jelenti azt, hogy a gép a következő parancsunkra vár.

A parancsok bevitelének fontos szabályai vannak, melynek be nem tartása szintaktikai hibákat eredményezhet.

A begépelt sornak paranccsal kell kezdődnie. A parancs után szóközzel elválasztva állhatnak paraméterek, de csak azokat használhatjuk, amelyek az adott parancshoz tartoznak.

A begépelt sor maximum 254 karakterből állhat. (Mivel a képernyő 80 karakter széles ezért annak szélénél automatikusan balra gördülnek a begépelt karakterek.)

A parancs érvényesítése az Enter billentyű megnyomásának hatására történik meg, amíg az Entert nem nyomtuk meg, addig bármit javíthatunk a sorban.

A dBase lehetőséget ad a korábban begépelt parancsok visszahozatalára a fel és lefelé mutató kurzormozgató nyilakkal. A visszahozott sorban a DOS-hoz hasonló módon a vízszintes nyilakkal tudunk mozogni, illetve ha kell az adott pozíción javítani. A Home és End billentyű a dBase-ben azonban csak egy szóval mozdítja el a kurzort.

A dBase parancsokat szükség esetén rövidíthetjük is, hiszen elegendő a parancs első 4 karakterének begépelése.

A parancsok illetve függvények begépelésénél egyaránt használhatunk kis- illetve nagybetűket is. (Parancsban ennek nincs jelentősége, de az adatokban már különbséget tesz a gép köztük.)

A funkcióbillentyűk megnyomása egy-egy parancs végrehajtását eredményezi. Az F1 kivételével e billentyűk mindegyike átprogramozható tetszőleges parancsra.

A programokban alkalmazott, párban használható parancsok esetében mindkét parancs megadása kötelező, bármelyik elhagyása hibát eredményez. (pl. IF, ENDIF)

A parancsok egy részénél kötelező paramétert megadni, ezeket semmiképpen sem hagyhatjuk el. A paraméterek egy része azonban nélkülözhető, ezért ezen paramétereket szögletes zárójelbe téve szokás jelezni. Amennyiben tehát a későbbiekben a parancsoknál vagy függvényeknél ilyen szögletes zárójelbe tett paramétert is találunk, úgy az azon kívül eső karaktereket minden esetben be kell gépelnünk, de a köztük lévőket elhagyhatjuk. Például az

```
STR (Nkif[,hossz[,tizedes]])
```

függvény esetén gépelhetünk

```
STR (Nkif)
STR (Nkif,hossz)
```

vagy

```
STR (Nkif,hossz,tizedes)
```

sorokat, mivel a szögletes zárójelek egymásba ágyazva kerültek jelölésre. (a szögletes zárójeleket azonban soha ne gépeljük be.)

Szintén előforduló jelölés még a függőleges vonal is, amely vagylagosságot jelöl. Az ilyen esetekben vagy a függőleges vonal előtt álló vagy az utána lévő paramétert kell begépelni, de mindenképp csak az egyiket.

Amennyiben listákat készítünk, úgy azok elemeit vesszővel kell egymástól elválasztani. Az utolsó elem után nem kell vesszőt tenni, s a vessző másutt nem szerepelhet. Például:

```
LIST NEV, IRSZ, VAROS
```

A későbbiekben sok esetben használni fogjuk az Nkif, Kkif, Lkif, hossz, tizedes, fájlnév, stb. jelöléseket. Begépelni természetesen nem ezen szavakat, hanem az ilyen jellegű adatot kell.

2.1.5. Segélykérés

Amennyiben hibás parancsot gépelünk be, a dBase automatikusan felajánlja a segítségét. Ezt a segítséget azonban bármikor más esetben is kérhetjük, ha begépeljük a

```
HELP
```

parancsot. Amennyiben konkrét parancsról szeretnénk tájékoztatást kapni, úgy a

```
HELP parancsszó
```

begépelése szükséges, ahol a parancsszó helyére értelemszerűen az ismertetendő parancsot kell gépelni.

2.1.6. DOS parancsok kiadása

A dBase lehetőséget nyújt arra, hogy egy DOS parancs kiadásához ne kelljen elhagynunk az adatbázis kezelőt, hanem annak keretein belül végezzünk operációs rendszer funkciókat. Ezen lehetőséghez a dBase promptja után egy felkiáltójellel kezdődően kell a DOS parancsot gépelnünk. Például a katalógus kéréshez a

```
!DIR
```

szócska begépelése szükséges. A felkiáltójel helyett használhatjuk a

```
RUN parancs
```

formulát is, de ez több gépelést igényel.

2.1.7. Hibaüzenetek

A munkánk során gyakran előfordul, hogy nem pontosan azt, vagy úgy gépeltük be, ami szükséges lett volna. Sok esetben a gép végrehajtja a rosszul kiadott parancsot is, ha azt szintaktikailag helyesnek találja (persze ilyenkor a helytelen művelet vagy adat miatt nem a helyes, vagy általunk várt eredményt szolgáltatja). Az ilyen esetek zömében azonban a parancs nem lesz végrehajtható sem, így a gép angol nyelvű hibaüzenetet küld, amely megjelenése után célszerű vagy segítséget kérni, vagy pedig a begépelt parancsot részletesen áttanulmányozni, hogy megtaláljuk benne a hibát. Sok esetben a hibát csupán egy felesleges vagy éppen hiányzó vessző, zárójel vagy szóköz okozza.

2.1.8. Környezeti paraméterek

A dBase használata során sok bosszúságot okozhatnak azok az apróságok, amelyeket a nem megfelelő mennyiségű tizedesjegyek, margóértékek, képernyőbeállítások, dátumformátumok stb. eredményeznek. Ezen gondok megoldására, illetve egyéb speciális jellemzők megváltoztatására szolgálnak a környezeti paraméterek. A környezeti paramétereket a SET parancs segítségével tudjuk megváltoztatni olymódon, hogy a SET után gépelt kulcsszónak megfelelő jellemző értékét be, illetve kikapcsoljuk az ON illetve OFF szócskák segítségével.

Például ha azt szeretnénk, hogy a gép ne ajánlja fel segítségét szintaktikai hiba esetén, akkor a

```
SET HELP OFF
```

parancssor begépelése szükséges. A segítség felajánlásának ismételt visszakapcsolására a

```
SET HELP ON
```

paranccsal tudjuk utasítani a gépet.

Nézzük meg tehát, hogy melyek ezek a kulcsszavak, melyekkel a környezeti paramétereket állíthatjuk:

ALTERNATE	kimeneti fájlba írás engedélyezése
BELL	adatbevitelkori figyelmeztető hang ki-be kapcsolása
CARRY	az előző rekord újba másolása APPEND parancsnál
CATALOG	új állományok katalógusba vételének engedélyezése, tiltása
CENTURY	dátumkiírásnál az évszázad megjelenítése

COLOR	színes üzemmód ki- és bekapcsolása
CONFIRM	megerősítés szükségessége teljesképernyős adatbevitelnél
CONSOLE	üzenetek képernyőre írásának engedélyezése, tiltása
DEBUG	az ECHO parancs kimenetének nyomtatón történő megjelenítésének ki- és bekapcsolása
DELETED	törlésre jelölt rekordok elrejtése, ill. mutatása
DELIMITER	mezőhatárok jelölése
DOHISTOR	végrehajtott utasítások eltárolása parancsállománynál
ECHO	parancsok megjelenítése a végrehajtás folyamán
ENCRYPTION	az állományról esetlegesen készült másolat hálózatos alkalmazásban rejtve lesz más felhasználók elől.
ESCAPE	az ESC billentyű parancsmegszakító hatásának engedélyezése, tiltása
EXACT	pontos egyezés vizsgálata karakterláncok összehasonlításakor
EXCLUSIVE	hálózati alkalmazásban a megnyitott állományok csak az adott munkaállomáson használhatók.
FIELDS	előzőleg definiált mezőlista figyelembe vétele
FIXED	kötött számú tizedesjegy megjelenítése
HEADING	DISPLAY illetve LIST parancs esetén a fejléc megjelenítésének engedélyezése, tiltása
HELP	segítség felajánlásának engedélyezése, tiltása
HISTORY	végrehajtott parancsok eltárolásának engedélyezése, tiltása
INTENSITY	inverz fényű mezőmegjelenítés teljesképernyős adatbevitel esetén
MENUS	billentyűzetmenü megjelenítése teljesképernyős szerkesztésnél
PRINT	folyamatos nyomtatás engedélyezése, tiltása
SAFETY	állományok felülírása esetén történő megerősítés engedélyezése, tiltása
SCOREBOARD	dBase üzenetek engedélyezése, tiltása
STATUS	állapotsor megjelenítése, tiltása
STEP	lépésenkénti programvégrehajtás engedélyezése, tiltása
TALK	a parancs végrehajtásának eredményéről megjelenő üzenet kiírásának engedélyezése, tiltása

TITLE emlékeztető cím bekérése illetve tiltása új állomány katalógusba vételekor

UNIQUE indexállományba írásnál több azonos kulcsú rekord beírásának engedélyezése, tiltása

A környezeti paraméterek másik típusa, amikor valamilyen értéket rendelünk a megadott jellemzőkhöz:

ALERNATE TO fájlnév megadott nevű kimeneti állomány létrehozása (amit begépelünk, az egy szöveges állományba is beíródik)

CATALOG TO fájlnév katalógusállomány megnyitása, ill. létrehozása

COLOR TO színkód képernyőre írás színének beállítása (színkódok: N→fekete, B→kék, G→zöld, R→piros, W→fehér, RB→lila, GR→barna, BG→türkiz, X→rejtett, +→magasfényű, monochrom monitornál: I→inverz, U→aláhúzott.)

DATE TO dátumforma a dátum megjelenési formájának beállítása (a dátumforma lehet: AMERICAN, ANSI, BRITISH, ITALIAN, FRENCH, GERMAN) (a Magyarországon megszokott dátumforma: ANSI)

DECIMALS TO n a minimálisan megjelenítendő tizedesjegyek száma

DEFAULT TO meghajtó az alapértelmezés szerinti meghajtó megváltoztatása

DELIMITERS TO k új mező-határolójelek meghatározása

DEVICE TO egység a @SAY parancs kimenetének helye (SCREEN | PRINT →képernyő | nyomatató)

FIELDS TO mezőlista mezőlista meghatározása

FILTER TO szűrőfeltétel szűrőfeltétel beállítás

FORMAT TO fájlnév képernyőformátum állomány megnyitása

FUNCTION TO kif programozható billentyűk átdefiniálása

HISTORY TO n parancsmentési terület méretének beállítása

INDEX TO fájlnév indexállomány megnyitása

MARGIN TO n a nyomtató bal margójának beállítása (n karakternyi margót hagy)

MEMOWIDTH TO n a memomezők megjelenítésének szélessége

MESSAGE TO üzenet felhasználói üzenet megjelenítése
ORDER TO indexnév főindex kijelölése a nyitott indexek közül
PATH TO elérési út elérési út meghatározása (a DOS könyvtárrendszerének megfelelően)
PRINTER TO egységazonosító nyomtatóeszköz egységazonosítójának megváltoztatása (pl. második nyomtató esetén: LPT2)
PROCEDURE TO fájlnév procedúraállomány megnyitása
RELATION TO adatállomány logikai kapcsolat létesítése két nyitott adatbázis között
TYPEAHEAD TO n billentyűzetpuffer méretének beállítása
VIEW TO fájlnév áttekintő állomány megnyitása

Például ha az aktuális szint (amivel a gép írni fog) sárgára akarjuk állítani akkor a

```
SET COLOR TO GR+
```

parancssort kell begépelnünk.

2.1.9. A dBase által használt állományok

A dBase program működése során az adatállományokon kívül sok egyéb állománytípussal is dolgozik. Az állományokat – mint ahogy azt a DOS-ban is tapasztaltuk – egy nyolcbetűs névvel és egy hárombetűs kiterjesztéssel azonosítjuk. Mivel ez a kiterjesztés határozza meg az állomány szerkezetét és funkcióját, ezért ez alapján tudunk különbséget tenni a különböző célra szolgáló állományok között.

Nézzük most meg, hogy melyek is ezek az állománytípusok:

Megnevezés	Kiterjesztés	Feladat
Katalógus-állomány	.CAT	Az egy alkalmazáscsoporthoz tartozó állományok nyilvántartására szolgál.
Adatbázis állomány	.DBF	Maga az adatbázis állomány, amely az adatokat tartalmazza mező és rekordszerkezetnek megfelelően.
Adatbázis memoállomány	.DBT	Az adatbázis memomezőinek tárolására szolgáló állomány.
Formátum-állomány	.FMT	A felhasználó által definiált képernyőformátum létrehozására szolgáló parancsokat tartalmazó szöveges állomány.
Nyomtatási listaformátum állomány	.FRM	A nyomtatott listákhoz szükséges információkat tartalmazó jelentésformátum állomány.

Megnevezés	Kiterjesztés	Feladat
Cimkeformátum állomány	.LBL	Az adatbázis rekordjainak cimkeformátumban történő nyomtatásához szükséges adatokat tartalmazó állomány.
Memória-állomány	.MEM	Memóriaváltozók (max 256) nevének és értékének megőrzése céljából létrehozható állomány.
Indexállomány	.NDX	A logikailag rendezett állomány rekordjainak sorrendjét tartalmazó állomány.
Program-állomány	.PRG	Szabványos ASCII formában tárolt program, parancs és procedúraállományok.
Szűrőállomány	.QRY	Egy adatbázishoz tartozó szűrőfeltételt tartalmazó állomány.
Képernyő-állomány	.SCR	A formaállományok létrehozásakor a képernyőforma előállítására használt állomány.
Szövegállomány	.TXT	Az adatbázisba esetlegesen beolvasható szabványos ASCII szöveget tartalmazó állomány.
Áttekintő állomány	.VUE	Egy előzőleg megtervezett munkakörnyezetet tartalmazó állomány.

2.1.10. Dátumformátumok

A dBase a dátumokat többféle megjelenési formában tudja kezelni nemzetiségi sajátosságoktól függően. A Magyarországon megszokott dátumformát, ahol az évszám áll legelöl, majd ezt követi a hónap és a nap, a

```
SET DATE ANSI
```

paranccsal aktivizálhatjuk. Célszerű ezt a parancsot rögtön a dBase használatba vételekor kiadni. Ez esetben azonban az évszám még mindig csak kétszámjegyű. Ha az évszázad megjelenítését is szeretnénk bekapcsolni, úgy a

```
SET CENTURY ON
```

parancs kiadása is szükséges.

2.1.11. Kilépés a programból

A dBase-ből történő kilépést soha ne egy egyszerű gépkikapcsolással végezzük, mert ez esetben elveszhetnek vagy megsérülhetnek adatállományaink. A kilépést tehát mindig a

```
QUIT
```

paranccsal tegyük meg, így a gép lezárja és elmenti az éppen nyitva lévő adatállományokat is.

MESSAGE TO üzenet felhasználói üzenet megjelenítése
ORDER TO indexnév főindex kijelölése a nyitott indexek közül
PATH TO elérési út elérési út meghatározása (a DOS könyvtárrendszerének megfelelően)
PRINTER TO egységazonosító nyomtatóeszköz egységazonosítójának megváltoztatása (pl. második nyomtató esetén: LPT2)
PROCEDURE TO fájlnév procedúraállomány megnyitása
RELATION TO adatállomány logikai kapcsolat létesítése két nyitott adatbázis között
TYPEAHEAD TO n billentyűzetpuffer méretének beállítása
VIEW TO fájlnév áttekintő állomány megnyitása

Például ha az aktuális szint (amivel a gép írni fog) sárgára akarjuk állítani akkor a

```
SET COLOR TO GR+
```

parancssort kell begépelnünk.

2.1.9. A dBase által használt állományok

A dBase program működése során az adatállományokon kívül sok egyéb állománytípussal is dolgozik. Az állományokat – mint ahogy azt a DOS-ban is tapasztaltuk – egy nyolcbetűs névvel és egy hárombetűs kiterjesztéssel azonosítjuk. Mivel ez a kiterjesztés határozza meg az állomány szerkezetét és funkcióját, ezért ez alapján tudunk különbséget tenni a különböző célra szolgáló állományok között.

Nézzük most meg, hogy melyek is ezek az állománytípusok:

Megnevezés	Kiterjesztés	Feladat
Katalógus-állomány	.CAT	Az egy alkalmazáscsoporthoz tartozó állományok nyilvántartására szolgál.
Adatbázis állomány	.DBF	Maga az adatbázis állomány, amely az adatokat tartalmazza mező és rekordszerkezetnek megfelelően.
Adatbázis memoállomány	.DBT	Az adatbázis memomezőinek tárolására szolgáló állomány.
Formátum-állomány	.FMT	A felhasználó által definiált képernyőformátum létrehozására szolgáló parancsokat tartalmazó szöveges állomány.
Nyomtatási listaformátum állomány	.FRM	A nyomtatott listákhoz szükséges információkat tartalmazó jelentésformátum állomány.

Megnevezés	Kiterjesztés	Feladat
Cimkeformátum állomány	.LBL	Az adatbázis rekordjainak cimkeformátumban történő nyomtatásához szükséges adatokat tartalmazó állomány.
Memória-állomány	.MEM	Memóriaváltozók (max 256) nevének és értékének megőrzése céljából létrehozható állomány.
Indexállomány	.NDX	A logikailag rendezett állomány rekordjainak sorrendjét tartalmazó állomány.
Program-állomány	.PRG	Szabványos ASCII formában tárolt program, parancs és procedúraállományok.
Szűrőállomány	.QRY	Egy adatbázishoz tartozó szűrőfeltételt tartalmazó állomány.
Képernyő-állomány	.SCR	A formaállományok létrehozásakor a képernyőforma előállítására használt állomány.
Szövegállomány	.TXT	Az adatbázisba esetlegesen beolvasható szabványos ASCII szöveget tartalmazó állomány.
Áttekintő állomány	.VUE	Egy előzőleg megtervezett munkakörnyezetet tartalmazó állomány.

2.1.10. Dátumformátumok

A dBase a dátumokat többféle megjelenési formában tudja kezelni nemzetiségi sajátosságoktól függően. A Magyarországon megszokott dátumformát, ahol az évszám áll legelöl, majd ezt követi a hónap és a nap, a

```
SET DATE ANSI
```

paranccsal aktivizálhatjuk. Célszerű ezt a parancsot rögtön a dBase használatba vételekor kiadni. Ez esetben azonban az évszám még mindig csak kétszámjegyű. Ha az évszázad megjelenítését is szeretnénk bekapcsolni, úgy a

```
SET CENTURY ON
```

parancs kiadása is szükséges.

2.1.11. Kilépés a programból

A dBase-ből történő kilépést soha ne egy egyszerű gépkikapcsolással végezzük, mert ez esetben elveszhetnek vagy megsérülhetnek adatállományaink. A kilépést tehát mindig a

```
QUIT
```

paranccsal tegyük meg, így a gép lezárja és elmenti az éppen nyitva lévő adatállományokat is.

2.2. Adattípusok

Mivel az adatbázis-kezelő rendszerek legfontosabb építőkövei az adatok, szükséges ezeket típusok szerint csoportosítani. Nem mindegy ugyanis, hogy ezen adatokat milyen célra használjuk fel, és mit szeretnénk kezdeni ezen adatokkal. Más-más típusú adat tárolására van szükség olyankor, amikor számolni szeretnénk az adattal, és másra olyankor, amikor valamilyen szövegösszefüggést kell vizsgálnunk.

Az adattípusok tehát a következők lehetnek:

2.2.1. Numerikus adatok

A numerikus adatok valójában számok. Olyan esetekben használjuk őket, amikor az adott adattal a későbbiekben valamilyen matematikai műveletet szeretnénk végezni, vagy értékét összehasonlítani. A numerikus adatok lehetnek pozitívak és negatívak is, de a dBase maximum csak 13 tizedesjegyig tud pontosan összehasonlítani és 15 jegyig pontosan számolni. A numerikus adatok használata esetén többnyire előre meg kell adni a tizedesjegyek számát is.

2.2.2. Karakteres adatok

A karakteres adatok előre meghatározott hosszú, de maximum 254 karakternyi szövegek, amelyek az ASCII kódtáblában megtalálható karaktereket tartalmazhatnak. A szövegekkel nem végezhetünk matematikai műveleteket, de hozzáírhatunk, megváltoztathatjuk és összehasonlíthatjuk más szövegekkel.

2.2.3. Dátum típusú adatok

A dátum típusú adatok időpontok tárolására szolgáló, előre meghatározott formájú és méretű adatok. (Pl. születési idő tárolására.)

A dátumok használata során azonban vigyázni kell arra, hogy a dátumok megjelenítése többféleképen is lehetséges, így adott esetben állítsuk be a megfelelő dátumformát (`SET DATE ANSI`).

2.2.4. Logikai adatok

A legrövidebb adat, értéke igaz vagy hamis lehet. Olyan helyeken alkalmazzák, ahol csak két lehetőség közül lehet választani. (Pl. férfinő, igen-nem, van-nincs, lehet-nem lehet, stb.) Értékadáskor a megfelelő angol szavak (True-Fals, Yes-No) kezdőbetűi használhatók a logi-

kai állításokat és függvényeket jelző kezdő és záró ponttal együtt. Igaz érték esetén használható a .T. és a .Y., hamis érték esetén pedig .F. és .N.

Logikai adatokkal ugyan csak logikai műveletek hajthatók végre, de kiválóan alkalmasak összehasonlítások esetén.

2.2.5. Memo típusú adatok

Memo típusú adatokat csak adatbázisban használhatunk hosszabb, előre nem meghatározott (max. 5000 karakter) hosszúságú szöveg (pl. megjegyzések) tárolására. A memo adattípussal semmilyen művelet nem végezhető, csupán szükség esetén lehet kérni megjelenítésüket.

2.3. Változók használata

A munkánk során szükségünk lehet az adatbázisban tároltaktól függetlenül felhasználható adatokra is. Az ily módon használt adatok egy része állandó értékkel bír, így ezeket **állandók**nak nevezzük, míg más részük tartalmát a felhasználó módosíthatja, így ezek lesznek a **változók**. Mivel ezen adatok az adatbázistól függetlenül használhatók fel és tárolhatók, ezért ezek csak numerikus, karakteres, dátum és logikai típusúak lehetnek.

A dBase-ben a módosítható adatok kezelése kétféle módon történhet. Az egyik lehetőség az, amikor a nyitott adatbázis aktuális rekordjának* valamelyik mezőjére hivatkozunk a mező elnevezésével.

A másik lehetőség az adatbázistól független ún. *memóriaváltozó* használata. A memóriaváltozókra egy általunk meghatározott névvel tudunk hivatkozni, s többnyire csak ideiglenes céllal használjuk őket. A memóriaváltozó nevének meghatározásakor azonban figyeljünk oda arra, hogy ne használjunk olyan nevet, amely már valamely adatbázismezőt azonosítja. A memóriaváltozó létrehozása nagyon egyszerű feladat, hiszen elegendő egy értékadó utasítást használnunk, ahol a memóriaváltozó valamilyen értéket vesz fel. A memóriaváltozó mindig olyan típusúnak deklarálódik, amilyen típusú volt az első rá vonatkozó értékadó művelet, így a későbbiekben is csak ugyanazon típusú adattal tudjuk módosítani. Ha egy létező memóriaváltozónak adunk új értéket, akkor a régi érték helyett az új értékkel íródik felül.

* Az aktuális rekord jelentéséről később lesz szó.

2.4. Műveletek adatokkal, változókkal

Memóriaváltozónak úgy tudunk a legegyszerűbben értéket adni, hogy a parancssorban a változó neve után egy egyenlőségjelet téve leírjuk a leendő értékét.
Például a

```
valami=3
```

parancs nem csinál mást, mint a `valami` nevű memóriaváltozóba 3-t ír. Természetesen a változók értékadása történhet úgy is, hogy a változó valamilyen kifejezés eredményét veszi fel. Pl.

```
valami=1+2
```

Ez esetben azonban vigyázni kell az adattípusok egyezésére is, hiszen ezen esetekben a `valami` változó numerikus típusú lesz. (Ha véletlenül ezt a változót már pl. karakteres adat tárolására használtuk volna, úgy a műveletet a gép nem lenne képes végrehajtani.)

A változók értékeit meg is tudjuk kérdezni a géptől, ha a változó neve elé kérdőjelet gépelünk:

```
?valami
```

a parancs hatására egyértelműen a hármas szám fog a képernyőn megjelenni.

A fenti parancsok persze sok mindennel helyettesíthetők, bár ezek használata nem mindig a legegyszerűbb. Az értékadásra ugyanis használható lehet a

```
STORE kifejezés TO memóriaváltozó
```

parancs is, s a megjelenítésre is léteznek más módszerek. (A kis illetve nagybetűknek itt nincs jelentősége.)

Az értékadások esetében azonban nagyon fontos az adatok típusa, így az adatbevitelkor is meg kell őket különböztetni. Amennyiben numerikus adat bevitele szükséges, úgy számokat gépelünk. Szöveges adatbevitel esetén azonban mindenképp jeleznünk kell a gépnek az adat határait, mivel ellenkező esetben adott nevű változót keresne. A szöveges adatok határoló jele lehet az idézőjel, az aposztróf, vagy a szögletes zárójel. Ezeket azonban csak párban szabad használni, hiszen rengeteg probléma forrása a határolójelek párjának elhagyásából fakad. Ily módon a

```
masik="valami"
```

parancs esetében a `masik` nevű memóriaváltozó tartalma a valami szó lesz, karakteres típusú változót definiálva. Ezzel ellentétben a

```
masik=valami
```

parancs kiadása numerikus típusú adatot hoz létre, hiszen a `masik` nevű változó itt a `valami` nevű változó értékét (azaz a 3-t) veszi fel.

Dátum típusú változót közvetlenül bevinni nem tudunk, csak valamilyen függvény alkalmazásának segítségével (például átkonvertálhatjuk a dátumot tartalmazó szöveget dátum típusúra).

Logikai változó használata esetén a változó értékadása úgy történik, hogy a definíciós részben egy logikai kifejezés áll. Például:

```
igaze=.T.
```

vagy

```
STORE masik=valami TO igaze
```

2.4.1. Alapműveletek

Műveletek használata során nagy hangsúlyt kap a változó típusa, ugyanis a műveletek többségét csak azonos típusú változókkal lehet végrehajtani, hiszen numerikus változóba csak számot, karakteres változóba pedig csak szöveget írhatunk.

A numerikus változókon végezhető műveletek tehát az összeadás (+), a kivonás (–), az előjelváltás (–), a szorzás (*), az osztás (/) és a hatványozás (^ vagy **). A

```
valami=3*2
```

parancs esetén tehát a valami változó értéke 6 lesz.

Karakteres változók használata esetén a konkatenáció művelete a + jellel kérhető. Például a

```
szoveg="irgum"+"burgum"
```

parancs hatására a `szoveg` változó értéke az „irgumburgum" szó lesz így egybeírva. (ha szóközt is akarnánk a kettő közé, akkor vagy az irgum és az idézőjel, vagy az idézőjel és burgum közé kellene tenni.) Vigyázzunk azonban a konkatenációval, mert a

```
namilesz="1"+"2"
```

parancs eredményeképp a `namilesz` változó értéke 12 lesz, méghozzá karakteres típusú adatként.

Dátum-műveletek használata során a kifejezésnek maximum két dátum tagja lehet. Így például a

```
ma=DATE()
```

paranccsal a `ma` változóba a mai nap dátumát írhatjuk. Ha a

```
tegnap=ma-1
```

parancsot használjuk, úgy a `tegnap` nevű változóba valóban a tegnapi dátum kerül. Ellenőrizzük le:

```
?tegnap
```

Logikai változók használata során az operandusba vagy összehasonlító, vagy logikai műveletet írhatunk két azonos típusú változó közé. Az összehasonlító művelet lehet a nagyobb (>), a kisebb (<) a nagyobb vagy egyenlő (>=), a kisebb vagy egyenlő (<=), az egyenlő (=) és a nem egyenlő (<>). Logikai műveletet csak logikai változókkal vagy kifejezéssel végezhetünk. A logikai műveletek közé tartozik a negálás (`.NOT.`), ahol az eredmény az operandus ellentettje (igaz esetén hamis, hamis esetén igaz), a logikai és kapcsolat (`.AND.`) ahol az eredmény csak akkor lesz igaz, ha az operandusok mindegyike igaz, és a logikai vagy kapcsolat (`.OR.`), ahol az eredmény akkor igaz, ha vagy az egyik vagy a másik operandus igaz (tehát csak akkor hamis, ha mindkét operandus hamis).

Fontos: A dBase-ben lehetőség van szövegek egyezőségének a vizsgálatára is, ilyenkor alapesetben a két szöveg akkor egyenlő, ha a másodikként megadott szöveg pontosan (nagy és kisbetűk különbözőnek számítva) ráillik az elsőként megadott szöveg elejére. (A teljes egyezőség kéréséhez a **SET EXACT ON** környezeti paraméter beállítása szükséges.) A műveletek végrehajtásakor a zárójelek használhatók, nélkülük azonban a matematikában megszokott prioritás szerint hajtódnak végre a műveletek.

2.4.2. Függvények

Az adatokkal végzett műveletek egy része a fent részletezett alapműveletek csoportjába sorolható, de egy ennél sokkal elterjedtebben alkalmazott csoportról, a függvényekről még nem esett szó. A függvények olyan értékadó műveletek, amelyek meghatározott eljárás alapján egyetlen értéket állítanak elő a függvény bemenetként megadott egy vagy több adatból. Ily módon az alkalmazható függvények száma és annak leírása az adott programnyelvtől függ. A dBase-ben a függvények leírása úgy történik, hogy az eredményt felvevő változó és az egyenlőségjel után kell gépelnünk a függvény megnevezését, majd zárójelben annak operandusát, vagy operandusait (vesszővel elválasztva egymástól). Például ha a `valami` változóban tárolt szám négyzetgyökét szeretnénk kiszámolni az `eredmeny` nevű változóba, és a négyzetgyök számolásra az **SQRT** függvény szolgál, akkor az

```
eredmeny=SQRT(valami)
```

parancsot kell alkalmazni.

A függvények alkalmazása során azonban fontos figyelni arra, hogy a függvény bemenetként milyen típusú operandust kíván és milyen típusú eredményt ad, hiszen az alkalmazott változókat ennek megfelelően kell kiválasztani. Mivel a fenti függvény numerikus adatot adott eredményül úgy, hogy a bemenete is numerikus adat volt, ezért mind az `eredmeny`, mind a `valami` nevű változónak numerikusnak kell lennie. Ezen szempont figyelembe vételével szükség esetén csoportosíthatjuk is a függvényeket aszerint, hogy milyen típusú adatokkal dolgoznak (bár némely függvénynek nem kell bemeneti operandust adni). Ily módon léteznek matematikai, karakterkezelő, dátum- és időlekérdező, konverziós, és speciális lekérdező függvények. Mivel sok esetben problémát okoznak a nem megfelelő adattípusok, alkalmazhatjuk a konverziós függvényeket, melyek az adattípusok átalakításában segédkezhetnek. Jó példa erre a dátum típusú adat értékadása, hiszen mint korábban láttuk csak numerikus és karakteres adatot tudunk közvetlenül gépelni. Ennek megfelelően, ha a `szuldatum` változóba szeretnénk dátum típusú adatot vinni, úgy a **CTOD** konverziós függvényt alkalmazva karakteres adat átalakításával a

```
szuldatum=CTOD("69.02.24")
```

paranccsal már megvalósíthatjuk az értékadást. (A fenti példa ANSI típusú dátumot feltételez.)

Lehetőség van arra is, hogy több függvényt (vagy műveletet) egymásba ágyazzunk, s így egy lépéssel oldjunk meg összetettebb műveleteket. Ha például negyedik gyököt szeretnénk vonni, akkor az

```
eredmeny=SQRT(SQRT(valami))
```

függvényt használhatjuk. Jó és gyakran használatos példa lehet még az, amikor azt szeretnénk meghatározni, hogy hány évesek vagyunk:

```
eredmeny=ROUND((DATE()-CTOD("69.02.24"))/365,0)
```

ahol az `eredmeny` úgy adja meg az életkort, hogy a **DATE()** függvénnyel megkapott mai napból kivontuk a karakterből konvertált születési dátumot, s ezt az eredményt osztottuk 365-el a **ROUND** függvénnyel nulla tizedesre kerekítve. Mindezt azért tehettük meg, mert két dátum típusú adat kivonása numerikus adatként adja meg a két dátum között eltelt napok számát.

A fent ismertetett néhány példa után lássuk most konkrétan melyek is azok a függvények, amelyeket a dBase-ben használhatunk:

Függvény	eredm. típusa	értelmezés
ABS(Nkif)	N	abszolutérték
EXP(Nkif)	N	exponenciális
INT(Nkif)	N	egészrész
LOG(Nkif)	N	e alapú logaritmus
MAX(Nkif1,Nkif2)	N	két szám közül a nagyobb
MIN(Nkif1,Nkif2)	N	két szám közül a kisebb
MOD(Nkif1,Nkif2)	N	két szám osztási maradéka
ROUND(Nkif1,n)	N	n darab tizedesjegyre kerekítés
SQRT(Nkif)	N	négyzetgyök
CDOW(Dkif)	K	a hét napjának neve
CMONTH(Dkif)	K	a hónap neve
DATE()	D	a mai dátum (a gép belső dátuma)
DAY(Dkif)	N	a hónap napja (ahányadika van)
DOW(Dkif)	N	a hét napja számmal
MONTH(Dkif)	N	az év hónapja
TIME(Dkif)	K	a pontos idő (a gép belső órája)
YEAR(Dkif)	N	évszám (évszázaddal)
ASC(Kkif)	N	az első karakter ASCII kódja
CHR(n)	K	n kódú karakter
CTOD(Kkif)	D	adott karakterláncból dátum előállítása
DTOC(Dkif)	K	adott dátumból karakterlánc előállítása
STR(Nkif[,n1[,n2]])	K	Nkif numerikus adatból, n1 hosszon, n2 tizedest tartalmazó karakterlánc előállítása
VAL(Kkif)	N	karakterláncból numerikus adat előállítása
AT(Kkif1,Kkif2)	N	a Kkif2-ben hányadik pozíciótól található meg Kkif1
LEFT(Kkif,Nkif)	K	a Kkif karakterlánc bal oldalán lévő Nkif darab karaktere
LEN(Kkif)	N	a karakterlánc hossza
LOWER(Kkif)	K	a karakterláncot kisbetűssé alakítja
LTRIM(Kkif)	K	a karakterlánc kezdő szóközeinek levágása
REPLACE(Kkif,Nkif)	K	a Kkif karakterlánc Nkif-szer történő ismétlése
RIGHT(Kkif,Nkif)	K	a Kkif karakterlánc jobb oldalán lévő Nkif darab karaktere
RTRIM(Kkif)	K	a karakterlánc végén elhelyezkedő szóközök levágása
SPACE(Nkif)	K	Nkif darab szóközből álló karakterlánc létrehozása
STUFF(Kkif1,n1,n2,Kkif2)	K	a Kkif1 karakterlánc n1 kezdőpozíciójától n2 hosszon elhelyezkedő részének kicserélése Kkif2 karakterláncra

SUBSTR(Kkif,n1,n2)	K	a Kkif karakterlánc n2 hosszú, n1 pozíción kezdődő részének kiválasztása
TRIM(Kkif)	K	a karakterlánc végén elhelyezkedő szóközök levágása
UPPER(Kkif)	K	a karakterláncot nagybetűssé alakítja
BOF()	L	igaz, ha a rekordmutató a fájl elején áll
COL()	N	az aktuális oszloppozíció a képernyőn
DBF()	K	az aktuális adatbázis neve
DELETED()	L	igaz, ha a rekord törlésre jelölt
DISKSPACE()	N	az aktuális meghajtón lévő szabad hely byte-ban
EOF()	L	igaz, ha a rekordmutató a fájl végén áll
ERROR()	N	a bekövetkezett hiba kódja
FIELD(Nkif)	K	az adott sorszámú mező neve
FILE(Kkif)	L	igaz, ha az adott állomány létezik
FKLABEL(Nkif)	K	az adott sorszámú funkcióbillentyű neve
FKMAX()	N	a funkcióbillentyűk száma
FOUND()	L	igaz, ha a keresőparancs megtalálta a keresett rekordot
GETENV(Kkif)	K	a Kkif DOS környezeti paraméter
IIF(Lkif,Kkif1,Kkif2)	K	az eredmény Kkif1 ha Lkif igaz, Kkif2 ha Lkif nem igaz
INKEY()	N	az éppen lenyomott billentyű kódja
ISALPHA(Kkif)	L	igaz, ha Kkif betűvel kezdődik
ISCOLOR()	L	igaz, ha a SET COLOR be van kapcsolva
ISLOWER(Kkif)	L	igaz, ha Kkif kisbetűvel kezdődik
ISUPPER(Kkif)	L	igaz, ha Kkif nagybetűvel kezdődik
LUPDATE()	D	az adatbázis utolsó módosításának dátuma
MESSAGE()	K	az utolsó hibához tartozó üzenet
NDX(Nkif)	K	az adott sorszámú indexállomány neve
OS()	K	az operációs rendszer neve
PCOL()	N	az aktuális oszloppozíció a nyomtatón
PROW()	N	az aktuális sorpozíció a nyomtatón
READKEY()	N	a teljesképernyő szerkesztő parancsból való kilépést megvalósító billentyű kódja
RECCOUNT()	N	az aktuális adatbázis rekordjainak száma
RECNO()	N	a rekordmutató értéke
RECSIZE()	N	az aktuális adatbázis egy rekordjának mérete
ROW()	N	a képernyő aktuális sorpozíciója
TYPE(kif)	K	a kif típusa
VERSION()	N	a dBase verzió száma

2.4.3. Értéknövelés

Gyakran előfordul, hogy egy változó értékét, annak korábbi tartalmától függően valamilyen értékkel (például eggyel) növelni kell. Mivel nem minden esetben tehetjük meg azt, hogy több lépés alkalmazásával megkérdezzük a változó értékét, hozzáadunk egyet, majd egy megint külön lépésben visszaírjuk a növelt értéket, ezért a programnyelvekben egy praktikus módszer van e cél megoldására. Így például a `valami` nevű numerikus típusú változó értékének eggyel való növelésére a

```
valami=valami+1
```

parancsot lehet alkalmazni.

2.4.4. Számológép funkció

A dBase-ben történő közvetlen számolásokhoz nem minden esetben kell változókat alkalmaznunk. Az egy lépésben kiszámolandó műveletek vagy függvények esetén közvetlenül is megkérdezhetjük a gépet az eredményről, ha arra a későbbiekben nem lesz szükségünk. Például:

```
? 1+2*3
```

2.5. Munkaterületek

A korábban tárgyalt változókat a dBase az adatállományoktól függetlenül kezeli. Az adatok nagy részét azonban nem változókban, hanem adatállományokban tároljuk, mégpedig a korábban ismertetett mező- és rekordszerkezet szerint. A dBase-ben egyszerre maximum tíz adatállomány kezelhető az esetlegesen hozzájuk tartozó egyéb állományokkal együtt. Mindezt úgy valósíthatjuk meg, hogy tíz úgynevezett munkaterületet képzelünk el, amelyek mindegyikén lehet egy-egy adatállomány. Természetesen ezen munkaterületek közül mindig csak egy aktív, így közvetlenül csak arra az egy adatállományra tudunk hivatkozni.

Az aktív munkaterület váltására a

```
SELECT munkaterület
```

parancs szolgál olymódon, hogy a munkaterületeket azonosíthatjuk egy 1 és tíz közötti egész számmal, az A és J közötti betűk valamelyikével, vagy ha az adott munkaterületen már található nyitott állomány, akkor annak a hivatkozási nevével. (Ily módon tehát az A és az 1 munkaterület azonosnak tekintendő.)

2.6. Adatállományok használata

2.6.1. Adatállomány létrehozása

Ahhoz, hogy egy adatbázissal dolgozni tudjunk, legelső feladat az adatállomány(ok) létrehozása. Ez két jól elkülöníthető feladatból áll: megtervezni az adatállomány szerkezetét, majd feltölteni megfelelő adatokkal.

Az adatbázis szerkezetének megtervezése az adatbázis kezelés egyik legfontosabb és legkritikusabb lépése. Egy rosszul megtervezett adatbázis ugyanis rengeteg pluszmunkát okozhat a későbbi feldolgozás során. Az adatbázisok megtervezésénél tehát mindig fordítsunk kellő figyelmet az alábbiakra:

- egy adatot egy adatállományba csak egyszer tároljunk,
- a más adatokból egyértelműen számítható adatokat szintén felesleges tárolni, hiszen ez redundanciát okoz,
- minden szükséges adatnak legyen hely, legfeljebb nem minden rekord esetén lesz minden mező kitöltve,
- a mező-elnevezések legyenek tömörek, rövidek, s utaljanak a mezők tartalmára,
- az adatbázis rekordirányban tetszőlegesen növelhető, mezőirányban azonban nem, így lehetőleg olyan adatállományt tervezzünk, amelyhez új mezőket már ne kelljen szúrni, viszont a rekordok bővítése és törlése megoldott lehessen,
- mivel az azonos mezőkben tárolt adatok típusát előre meg kell határozni, így az adott jellemzőhöz annak jellegétől és a későbbi felhasználási igénytől függően válasszunk adattípust,
- a mezők maximális hossza szintén előre definiálandó, így mindig nézzük meg, hogy maximum milyen hosszú adatot írunk az adott mezőbe,
- a képernyő többnyire 80 karakter széles, így az egyszerűbb megjelenítéseknél csak akkor kapunk könnyen áttekinthető listát, ha az adatszélességek összege az egyéb közökkel és rekordsorszámmal együtt sem éri el ezt az értéket (amennyiben ez nem oldható meg, más megjelenítési formákra lesz szükség).

Amennyiben tehát papíron vagy fejben megterveztük az adatállomány szerkezetét, azaz meghatároztuk, hogy milyen nevű, típusú és hosszú mezőket alkalmazunk, megkezdhetjük az adatállomány létrehozását. Erre a célra dBase-ben a

```
CREATE állománynév
```

parancs szolgál, ahol az állománynév a DOS szabályai szerint képzett elnevezés. Kiterjesztés megadása nem szükséges, hiszen azt a gép automatikusan DBF-ként generálja. Ha tehát az `ismeros` nevű adatállományt szeretnénk létrehozni, akkor a

```
CREATE ismeros
```

parancsot gépeljük. (A parancsok esetében a kis és nagybetűk közt a gép nem tesz különbséget, így itt mindegy hogy mit írunk kis vagy nagybetűvel. Az idézőjelek közé tett szöveges adatnál azonban mindenképp figyelnünk kell, mert ott már eltér az értelmezésük.)

A parancs hatására tehát az adatállomány eltervezett szerkezetét kell gépre vinni olymódon, hogy egy teljesképernyős szerkesztési formában kell az adatbázis mezőinek jellemzőit megadni. A definiálás folyamán soronként egy mezőt írunk le úgy, hogy megadjuk a mező nevét, a típusát, a szélességét és numerikus mező esetén a tizedesjegyek számát (ez esetben a hossz a tizedesjegyekkel és a tizedesponttal együtt értelmezendő). A mező típusának megadása úgy történik, hogy vagy a típusnak megfelelő kezdőbetűt ütjük le (C, D, L, M, N), vagy a szóköz billentyűvel beforgatjuk a megfelelő mezőtípust. (Amennyiben a kívánt mezőtípusnak megfelelő karaktert látunk, úgy üthetünk Entert a következő adat megadásához.) A szélesség megadása csak karakteres és numerikus mező esetén szükséges, egyéb esetekben automatikusan generálódik. A megfelelő adat beírása után az Enter billentyűvel tudunk továbblépni. A meződefiniálás befejezése a Ctrl-End billentyűk egyidejű megnyomásával, vagy egyszerűen a következő üres mezőnéven leütött Enterrel történhet. Természetesen ezt egy ismételt Enterrel is meg kell erősítenünk. (A rekordszerkezet felvitele során is van lehetőségünk az utólag észrevett tévedések javítására a kurzormozgató nyilak segítségével.)

	Field Name	Type	Width	Dec
1	NEV	Character	20	
2	IRSZ	Numeric	4	0
3	CIM	Character	30	
4	NO	Logical	1	
5	SZULIDO	Date	8	

Az adatállomány létrehozását követően a gép megkérdezi, hogy megkezdjük-e az állomány feltöltését új adatokkal (*Input data records now?*). Az N billentyű hatására visszakapjuk a promptot, az Y hatására elkezdhetünk adatokat felvinni.

2.6.2. Új rekord hozzáfűzése

Az aktív munkaterületen lévő adatállomány végéhez új rekordot az

```
APPEND
```

paranccsal tudunk írni. (A parancs természetesen csak akkor adható ki, ha az aktív munkaterületen van nyitott adatbázis.)

Akár új rekordot fűzünk az adatbázishoz, akár a létrehozást követően viszünk fel rekordokat, az adatbevitel minden esetben úgy történik, hogy egy teljesképernyős szerkesztési módban baloldalt megjelennek a mezők nevei, mellettük pedig a kitöltésre váró terület. Írni mindig a kurzor pozíciójára tudunk, s a kurzort a kurzormozgató nyilakkal lehetőségünk van mozgatni is. Az Enter billentyűvel az adott sort érvényesítve ugorhatunk a következő mezőre, de a mezőben rendelkezésre álló hely teleírása után a gép automatikusan is a következő mezőre visz. Az adott rekord mezőinek kitöltése után a következő rekord üres mezői automatikusan megjelennek, így lehetőség van folyamatos adatfelvitelre. Adatbevitelt befejezni az üres rekordon nyomott Enterrel, vagy az utolsó rekordon nyomott Ctrl-End billentyűkkel tudunk.

Memo típusú mezők kitöltéséhez a memo feliraton állva kell megnyomni a Ctrl-Home vagy Ctrl-PgDn billentyűket. Ennek hatására megjelenik egy szövegszerkesztőhöz hasonló adatbeviteli felület, ahova gépelhetjük megjegyzéseinket. Visszatérni a Ctrl-End vagy Ctrl-PgUp billentyűkkel tudunk.

Figyeljünk arra is, hogy csak akkor kerülnek tárolásra módosításaink ha kilépni is szabályosan - a **QUIT** paranccsal - lépünk ki az adatbázis kezelőből.

2.6.3. Meglévő adatállomány megnyitása

Lemezen tárolt meglévő adatállományt az aktív munkaterületre megnyitni a

```
USE adatállománynév
```

paranccsal tudunk. Egy munkaterületre csak egy állományt nyithatunk, s több munkaterületre sem tudunk azonos állományt megnyitni.

2.6.3.1. Adatállomány szerkezetének módosítása

Meglévő adatállományok szerkezetének módosítása a

```
MODIFY STRUCTURE
```

parancs kiadásával lehetséges. A parancs az aktív munkaterületen lévő adatállomány szerkezetének módosítására ad lehetőséget olymódon, hogy teljesképernyős szerkesztéssel módosíthatjuk a mezők azonosító adatait. (Mező törlését a Ctrl-U, beszúrását a Ctrl-N billentyűkkel valósíthatjuk meg.)

Fontos: Az adatbázis szerkezetének módosítása adatvesztéshez vezethet akkor, ha mezőket szüntetünk meg, mezőket nevezünk át, vagy mezőhosszokat veszünk rövidebbre. Az adatbázis szerkezetét csak végső esetben módosítsuk, s figyeljünk mindig arra, hogy a mezők hosszának rövidítése azt eredményezi, hogy a gép levágja az adott mezőbe így már el nem férő adatok végét.

2.6.4. Adatállományok lezárása

Az adatállományok automatikusan lezáródnak a dBase-ből QUIT paranccsal történő kilépéskor, de egyes adatállományokat külön is lezárhatunk a

```
CLOSE adatállománynév
```

parancs segítségével, vagy az adott munkaterületre megnyitott másik adatbázissal.

2.6.4.1. Adatok listázása

Az aktív adatállomány teljes tartalmát egyszerűen megjeleníteni a

```
LIST
```

parancs kiadásával tudjuk. A **LIST** parancs azonban rengeteg paraméterrel is ellátható, amelyekről majd később az adatok megjelenítése c. pontban foglalkozunk.

2.6.5. Rekordmutató mozgatása

Ha már van használatban lévő adatbázisunk, akkor az adatbevitelen túl szükség lehet listázásra, keresésre, törlésre, stb. Mindezen műveletekhez azonban tisztában kell lenni néhány elengedhetetlen fogalommal. A dBase úgy kezeli az adatállományokat, hogy nem csak aktív adatállomány létezik, hanem ezen belül egy **aktív rekord** is. Ez azt jelenti, hogy létezik egy ún. **rekordmutató**, amely az adatállomány valamelyik rekordjára mutat. Ha az adatállományból adatot akarunk kiolvasni, akkor ennek a rekordnak a mezőit tudjuk olvasni, ha műveletet akarunk végezni, akkor ezzel a rekorddal tudunk. Az aktív rekord sorszámát, azaz a rekordmutató pozícióját az alsó inverz sorban a *Rec:* szó után látjuk. (Itt jelzi a gép, az aktuális és az összes rekordot.) Ha a rekordmutató a szimbólumaként nem egy szám, hanem az EOF szócska látható, akkor az azt jelzi, hogy a rekordmutató túlmutat a létező rekordokon, az adatállomány végén áll.

Ha tehát nem egy változóval, hanem az adatbázis megfelelő elemével szeretnénk számolni, akkor a mezőnévre kell hivatkoznunk, s így a rekordmutató által mutatott rekord megadott mezője kezelhető ugyanúgy, mintha az egy változó lenne.
Például a

```
? ROUND((DATE()-SZULIDO)/365,0)
```

parancs nem a begépelt, hanem a rekordmutató által mutatott rekordban található személy életkorát számolja ki, ha az aktív adatbázisban található `szulido` nevű mező valóban a születési időt tárolja.

Mindezek tehát azt követelik meg tőlünk, hogy tudjuk mozgatni a rekordmutatót.

A rekordmutató mozgatása többféleképen is történhet. Egyik esetben automatikusan megy végig az adatbázison a listázó, átlag és összegszámoló illetve teljes helyettesítő parancsok használatakor.

A manuális mozgatás során abszolút és relatív mozgatásokat is végezhetünk. Abszolút értelembe vett mozgatás során a megadott számú rekordra tudunk ugrani, relatív értelemben pedig a rekordmutató jelenlegi értékéhez viszonyítva történik a mozgás. Az abszolút értelemben vett mozgatásra a

```
GOTO n
GO n
n
```

parancsok valamelyike szolgálhat, ahol n az a rekordsorszám ahova a rekordmutatót állítani akarjuk. A **GO**, vagy **GOTO** után írhatunk **TOP** és **BOTTOM** szavakat is, ilyenkor a

```
GO TOP
```

vagy

```
GO 1
```

parancs az adatállomány első, a

```
GO BOTTOM
```

parancs az adatállomány utolsó rekordjára irányítja a rekordmutatót. Relatív mozgatást a

```
SKIP n
```

paranccsal tudunk megvalósítani, ahol n az aktuális rekordsorszámtól való eltérést jelzi. Így például a

```
SKIP -1
```

paranccsal eggyel visszaléptethetjük a rekordmutatót. Amennyiben a **SKIP** parancs paraméterébe nem gépelünk adatot, úgy az eggyel történő előreléptetésnek fog megfelelni. (További rekordmutató mozgatást a

rendezett adatállományokban történő kereső parancsokkal is megvalósíthatunk.)

2.6.6. Érvényességi kör

A parancsok egy része nem csak egy hanem több rekordra is vonatkozhat. Az ilyen parancsoknál nem rekordsorszámot, hanem úgynevezett érvényességi kört kell megadni, amely nem más, mint a rekordok valamilyen szempont szerint összeválogatott csoportja, amelyen az adott parancsot végre kell hajtani. Az érvényességi kör lehet:

- **ALL** az adatállomány összes rekordja
- **REST** az adatállomány aktuális rekordjától kezdve az adatállomány végéig elhelyezkedő rekordok beleértve az aktuális rekordot is.
- **RECORD n** az adatállomány n-edik rekordja
- **NEXT n** az adatállomány aktuális rekordjától kezdve n darab rekord, beleértve az aktuális rekordot is.

Ennek megfelelően ha például csak az aktuális rekordtól szeretnénk listázni állományunkat, akkor az alábbi parancs alkalmazása szükséges:

```
LIST REST
```

2.6.7. WHILE és FOR paraméter

Bizonyos parancsoknál lehetőség van az érvényességi kört – a fenti módon történő megadása helyett vagy mellett – feltétel megadásával is meghatározni. Feltétel megadása esetén a feltételben leírt igényeknek eleget tevő rekordokra fog vonatkozni a parancs.

WHILE paraméter használata esetén a parancs a rekordmutató aktuális pozíciójától kezdve hajtódik végre mindaddig, amíg a megadott feltétel teljesül. Amennyiben valamely rekordnál a feltétel nem teljesül, úgy a végrehajtás megáll, s a további rekordok egyikén sem történik végrehajtás. A **WHILE** paraméter használata esetén tehát az **ALL** (ha a rekordmutató az állomány elejére mutat) és **REST** érvényességi körök megadása elhagyható, a **NEXT n**-el szükség esetén tovább korlátozható. A

```
LIST WHILE NO
```

parancs tehát addig listázza a rekordokat, amíg a rekordok **NO** nevű logikai típusú mezője igaz. (Nem listázza ki az összes nőt!)

A **FOR** paraméter a **WHILE**-al ellentétben nem állítja le a folyamatot akkor, ha nem teljesül a feltétel, hiszen a **FOR** paraméter mindazon rekordokra végrehajtatja a parancsot, ahol a feltétel teljesül. A

```
LIST FOR IRSZ=8000
```

parancs tehát kilistázza mindazon rekordokat, ahol az IRSZ mező tartalma megegyezik 8000-rel. Az érvényességi kört a **FOR** paraméter mellett is tovább korlátozhatjuk a **NEXT** és **REST** érvényességi körök megadásával.

2.6.8. Új rekord beszúrása

Az adatállomány végéhez rekordot hozzáfűzni az **APPEND** paranccsal már tudunk. Az adatállomány belsejébe a rekordmutató által mutatott rekord után az

```
INSERT
```

paranccsal lehet rekordot szúrni. Amennyiben a rekordmutató által mutatott rekord elé szeretnénk rekordot szúrni, úgy az

```
INSERT BEFORE
```

parancs kiadása szükséges.

2.6.9. Rekordok tartalmának módosítása

A meglévő rekordok tartalmának teljesképernyős módosítására használhatjuk az

```
EDIT
```

parancsot, amely a rekordmutató által mutatott rekord módosítását teszi lehetővé. Másik javítási lehetőség az ún. pásztázás, amikor a képernyőn egyszerre több (legfeljebb 17) rekord jelenik meg, melyekben az egészet áttekintve tudunk javítgatni. A tevékenység kiválasztásához a

```
BROWSE
```

parancs kiadása szükséges. A szerkesztés befejezéséhez itt is a Ctrl-End billentyűket kell megnyomnunk.

2.6.10. Rekord törlése

A dBase adatbázis kezelőben az adatok törlése biztonsági okok miatt nem egylépcsős feladat. Bár a

```
ZAP
```

paranccsal az aktív adatállományt teljes egészében törölhetjük, a rekordok törlése mégis csak egy ún. törlésre jelöléssel (logikai törlés) és

egy a törlésre jelölt rekordok fizikai törlésével valósítható meg. (Az egyenkénti fizikai törlés ugyanis nagy adatállományok esetén jelentősen lelassítja a munkát.) Ily módon tehát a törlésre jelölést a

```
DELETE [érvényességi kör] [felt. kif.]
```

paranccsal tehetjük meg, ahol a feltételben (**FOR** és **WHILE** paraméter) megadott rekordok kerülnek törlésre jelölésre. Természetesen nem kötelező sem az érvényességi kör sem a feltétel megadása, ilyenkor az aktuális rekord kerül törlésre jelölésre. Ha például az összes budapesti ismerősünket szeretnénk törlésre jelölni, úgy a

```
DELETE FOR IRSZ<2000
```

parancsot is tudjuk használni, hiszen a budapestiek irányítószáma biztosan kisebb 2000-nél. A törlésre jelölt rekordokat listázás esetén a gép egy *-gal jelzi, s a

```
SET DELETED ON
```

parancs kiadása után meg sem jeleníti, s más célra sem tudja használni.

Adott esetben lehetőség van a törlésre jelölt rekordok visszaállítására is a

```
RECALL [érvényességi kör] [felt. kif.]
```

paranccsal, ahol a paraméterek a **DELETE** parancshoz hasonlóan értelmeződnek. A **RECALL** azonban csak a

```
SET DELETED OFF
```

esetén használható. Ily módon az összes törlésre jelölt rekordot visszaállítani tehát a

```
RECALL ALL
```

paranccsal tudnánk. A törlésre jelölt rekordokat fizikailag is törölni a

```
PACK
```

paranccsal lehet.

2.6.11. Adatok megjelenítése

Az adatok megjelenítésének egyik módja a rekordmutató által mutatatott rekord egy mezőjének lekérdezése. Például a

```
? nev
```

parancs hatására az aktuális rekord `nev` mezője jelenik meg, ha a rekordmutató nem az állomány végén áll. Ez a módszer azonban elég ritkán használatos, sokkal elterjedtebb az adatállomány folyamatos listázására szolgáló

```
LIST [érvényességi kör] [kifejezéslista] [OFF]
```

vagy

```
DISPLAY [érvényességi kör] [kifejezéslista] [OFF]
```

parancsok. A két parancs majdnem ugyanarra szolgál, a különbség csupán az, hogy a **DISPLAY** a listázásban megáll képernyőoldalanként és az alapértelmezés szerinti érvényességi köre az aktuális rekord, addig a **LIST** folyamatosan listáz és az érvényességi kör megadása nélkül az összes rekordot megjeleníti. Az érvényességi kör megadása helyett a kifejezéslista után mindkét parancsnál gépelhetünk **WHILE** vagy **FOR** feltételt is, amennyiben csak egy megadott részét szeretnénk megjeleníteni a rekordjainknak. A kifejezéslista elhagyása esetén az összes mező megjelenik, míg annak megadásával lehetőség van kijelölni a megjelenítendő mezőket. (Ilyenkor fel kell őket sorolni egymástól vesszővel elválasztva.) Fel kell hívni a figyelmet arra, hogy az adatállományok memo mezői csak akkor jelennek meg, ha azokat a kifejezéslistában is felsoroljuk. Az **OFF** paraméter használata esetén a rekordsorszám megjelenítésének elhagyására van lehetőség.

Nézzünk most néhány példát az adatok megjelenítésére: Először is listázzuk ki az összes ismerősünk nevét és születési idejét:

```
LIST nev, szulido
```

ha azt szeretnénk megjeleníteni, hogy melyik ismerősünk hány éves, akkor a

```
LIST NEV, ROUND((DATE()-SZULIDO)/365,0)
```

parancs lesz használható, hiszen az életkor a mai dátumból és a születési időből számítható.

Most jelenítsük meg az összes adatot úgy, hogy a rekordsorszám ne jelenjen meg a listában:

```
LIST OFF
```

A következő példával a rekordmutatót a második rekordra állítjuk, majd az aktuális rekordtól kezdve három rekordot kilistázunk (feltéve, hogy van annyi az adatállományban):

```
GO 2
LIST NEXT 3
```

2.6.12. Adatok megjelenítése másik munkaterületről

Lehetőségünk van más munkaterületről is megjeleníteni adatokat oly módon, hogy a mezőnév előtt a másik megnyitott adatállománynévre hivatkozunk. Az adatállománynév és a mezőnév közé **->** jeleket (kötőjel és nagyobbjel) kell tenni. Például ha az `adat` nevű adatállomány

rekordmutatója által mutatott rekordjának gkrdsz mezőjét szeretnénk megtudni, akkor a

```
? adat->gkrdsz
```

paranccsal ezt megtehetjük. Mindemellett szem előtt kell tartani azt, hogy független adatállományok esetén csak az aktív munkaterületen tudjuk mozgatni a rekordmutatót, a hivatkozott adatállományban nem. Így ha az aktív munkaterületen lévő adatállomány név és születési idő mezője mellett egy másik munkaterületen lévő adat nevű adatállomány gkrdsz mezőjére is kíváncsiak vagyunk, kiadhatnánk a

```
LIST nev, szulido, adat->gkrdsz
```

parancsot, de ekkor az aktív munkaterületen lévő adatállomány minden rekordjához, a másik munkaterület ugyanazon rekordja kerül megjelenítésre. (Ezen problémára az adatállományok összekapcsolása fog megoldást nyújtani.)

2.6.13. Adatok feltételes megjelenítése, válogatás

Az adatok feltételes megjelenítésére legjobb módszer a **LIST** vagy **DISPLAY** parancs **FOR** (esetleg **WHILE**) paraméterrel történő kiegészítése. Ez esetben ugyanis a **LIST FOR** után egy olyan logikai kifejezést kell írnunk, amely csak azoknál a rekordoknál teljesül, amelyeket meg szeretnénk jeleníteni. Ez a kifejezés bármilyen logikai mező, összehasonlítás vagy egyéb formula lehet, lényeg az, hogy logikai eredményt adjon. Ezen kifejezések megismerése igen fontos, hiszen más esetekben (adatmódosítás, rekordtörlés, stb.) is jól hasznosítható lehet.

Nézzünk egy egyszerűbb feltételt. Jelenítsük meg azon rekordokat, amelyeknél az irányítószám mező nagyobb vagy egyenlő kétezernél.(azaz a nem budapestieket):

```
LIST FOR IRSZ>=2000
```

A parancs természetesen más módon is felírható, például így:

```
LIST FOR 1999<IRSZ
```

Az összehasonlítás után nézzünk egy példát a logikai típusú változók használatára. Jelenítsük meg azon rekordokat, amelyeknél a nő mező igaz értéket szolgáltat:

```
LIST FOR NO
```

Ennek megfelelően a férfiak megjelenítésére a

```
LIST FOR .NOT. NO
```

parancs lesz használható. Lehetőségünk van szövegegyezőség vizsgálatára is olymódon, hogy figyelembe vesszük azt a tényt, hogy a gép

egyenlőnek veszi azon szövegeket, ahol a másodikként megadott szöveg pontosan (nagy és kisbetűk különbözőnek számítva) ráillik az elsőként megadott szöveg elejére. (A teljes egyezőség kéréséhez **a SET EXACT ON** környezeti paraméter beállítása szükséges.) Ha tehát azt szeretnénk megtudni, hogy mely rekordok esetén kezdődik a címet tartalmazó mező Budapesttel, akkor a

```
LIST FOR CIM="Budapest"
```

paranccsal ezt megtehetjük, de ne feledjük azt, hogy az adatállományoknál nem tudhatjuk, hogy hol írták kis és hol nagybetűvel a címet. A fenti parancs ugyanis nem találja meg a nagybetűs változatokat, így kénytelenek vagyunk egy olyan parancsot kiadni, ahol a `cim` mezőt nagybetűssé konvertáljuk, és ezt hasonlítjuk össze a szintén végig nagybetűvel megadott szóval:

```
LIST FOR UPPER(CIM)="BUDAPEST"
```

Természetesen kisbetűs változatban is használhatjuk a parancsot:

```
LIST FOR LOWER(CIM)="budapest"
```

A fentiekből következőleg jogosan merülhet fel a kérdés, mi van akkor, ha a címben Budapest és Bp rövidítés is szerepel? Ilyenkor összetett feltételvizsgálatot kell tennünk, azaz egy logikai művelet segítségével mind a két esetet vizsgálnunk kell. A

```
LIST FOR UPPER(CIM)="BUDAPEST" .OR. UPPER(CIM)="BP"
```

parancs tehát azokat a rekordokat írja ki, ahol a `cim` mező vagy a Budapest, vagy a Bp rövidítéssel kezdődik. Amennyiben azokra a rekordokra vagyunk kíváncsiak, amelyek `cim` mezője tartalmazza a Budapest szót vagy a Bp rövidítést, akkor más módszert kell választanunk:

```
LIST FOR "BUDAPEST"$UPPER(CIM) .OR. "BP"$UPPER(CIM)
```

A parancsban a **$** jel szolgál arra, hogy ne csak a szöveg elejére történő ráillesztést, hanem a „bennelévőséget" is vizsgálja. Ez esetben azonban az operandusokat fordított sorrendben kell írni, a művelet ugyanis azt vizsgálja, hogy az elsőként megadott karakterfüzér megtalálható-e a másodikként megadott karaktersorozatban. (Vigyázzunk arra, hogy a fenti példa kiírná például a Zabpuszta településnevet is, hiszen a bp betűpáros abban is megtalálható.)

Összetett feltételvizsgálatot más esetekben is végezhetünk, például ha meg szeretnénk jeleníteni a 30 év alatti férfiak adatait:

```
LIST FOR(DATE()-SZULIDO)/365<30.AND..NOT. NO
```

A fenti parancs tehát csak azon rekordokat jeleníti meg, ahol a mai dátum és a születési idő közt eltelt napok számának 365-öd része ki-

sebb mint harminc (azaz fiatalabb harminc évesnél) és a no mező értéke hamis (azaz férfi). A két feltétel közti *és* kapcsolatot a **.AND.** szócska teremti meg.

Utolsó példaként nézzünk meg egy olyan parancsot, ahol a 20 évnél idősebb budapesti emberek nevét és születési dátumát szeretnénk megjeleníteni úgy, hogy a gép a rekordsorszámot ne írja ki:

```
LIST NEV, SZULIDO FOR (DATE()-SZULIDO)/365<20 .AND.
(UPPER(CIM)="BUDAPEST" .OR. UPPER(CIM)="BP")  OFF
```

vagy

```
LIST NEV, SZULIDO FOR IRSZ<2000 .AND.
(DATE()-SZULIDO)/365<20  OFF
```

A fenti parancsokat természetesen egy sorba kell gépelni.

2.6.14. Mezők helyettesítése

A memóriaváltozókkal ellentétben az adatbázist alkotó adatoknak nem lehet úgy értéket adni, hogy a mezőnév után egyenlőségjellel írjuk a felveendő adatot. Az adatbázisban adatot módosítani csak a

```
REPLACE [érv. kör] mezőnév WITH újérték [felt. kif]
```

paranccsal tudunk. Amennyiben külön érvényességi kört nem adunk meg, úgy a parancs csak a rekordmutató által jelzett rekordra fog vonatkozni. WHILE vagy FOR feltétel megadása esetén az érvényességi kör az összes rekordra kiterjed. Ennek megfelelően ha az aktuális rekord nev mezőjét Gipsz Jakabra szeretnénk módosítani, úgy a

```
REPLACE nev WITH "Gipsz Jakab"
```

parancsot kell használnunk, vagy ha például egy adatállomány jutalom mezőit szeretnénk kitölteni a fizetes mező 20%-ával, akkor a

```
REPLACE ALL jutalom WITH fizetes*0.2
```

parancs használatos. Feltétel megadása esetén a parancs után WHILE vagy FOR feltételt írhatunk. Ily módon ha a fenti példában csak a nők kapnának a fizetésük 30%-ának megfelelő jutalmat, akkor a

```
REPLACE jutalom WITH fizetes*0.3 FOR NO
```

parancsot használhatjuk, ha létezik egy no nevű logikai mező.

2.6.15. Összeg, átlag, darabszám számítása

Lehetőségünk van közvetlenül egy paranccsal is numerikus mezők összegét vagy átlagát kiszámolni. Az átlagszámításra az

```
AVERAGE mezőnév [TO változó] [felt. kif.]
```

szolgál, ahol a mezőnév annak a numerikus típusú mezőnek a neve,

amelyben lévő adatok átlagát szeretnénk kiszámolni. Ily módon ha egy `fizetes` mezőt tartalmazó adatállományból átlagfizetést szeretnénk számolni, úgy az

```
AVERAGE fizetes
```

parancs kiadása szükséges. Ha a kapott eredményt nem csak megjeleníteni szeretnénk, hanem egy későbbi felhasználás érdekében egy `atlfiz` változóban tárolni is akarjuk, akkor az

```
AVERAGE fizetes TO atlfiz
```

parancsot kell alkalmaznunk. Ha a korábbi életkor-számítási módszernek megfelelően az `af` változóba a 30 év feletti emberek átlagfizetését szeretnénk írni, akkor a használatos parancs formája:

```
AVERAGE fizetes TO af FOR (DATE()-SZULIDO)/365>30
```

Az összeg számítása az átlagszámítás parancsához hasonlóan a

```
SUM mezőnév [TO változó] [felt. kif.]
```

paranccsal történik. A változó és feltételes kifejezések használatának módja szintén megegyezik az átlagszámításra szolgáló parancséval. Az adott feltételnek eleget tevő mezők darabszámának megszámlálására a

```
COUNT [felt.kif] [TO változó]
```

parancsot kell alkalmazni. Ha például a `hanyno` memóriaváltozóba az adatállományban szereplő nők számát szeretnénk írni, akkor a

```
COUNT FOR no TO hanyno]
```

parancsot lehet kiadni, ha a `no` logikai mező létezik.

2.7. Nyomtatás, párhuzamos nyomtatás

Az adatok nyomtatón történő megjelentetésének a dBase-ben két fő módja létezik. Az egyik, amikor a

```
SET PRINT ON
```

paranccsal bekapcsoljuk a párhuzamos nyomtatást. Ilyenkor minden ami a képernyőre kerül, megjelenik nyomtatón is. A párhuzamos nyomtatás kikapcsolására a

```
SET PRINT OFF
```

parancs szolgál. A másik módszer az, amikor előre megadjuk, hogy a kért parancs eredménye ne képernyőre, hanem nyomtatóra kerüljön. Ezt úgy tehetjük meg, hogy a parancs legvégére odaírjuk a **TO PRINT** szócskákat is. Például:

```
LIST TO PRINT
```

2.8. Egyéb parancsok

2.8.1. Képernyőtörlés

A dBase képernyőjének letörlésére szolgáló parancs a

```
CLEAR
```

2.8.1.1. Színek állítása

Bár a képernyőre kerülő színek meghatározására szolgáló parancsot már a környezeti paraméterek témakörénél ismertettük, de a parancs valójában jóval több állítási lehetőséget kínál, ugyanis a

```
SET COLOR TO n1/n2, n3/n4, n5
```

formával a következőket állíthatjuk:

n1 →normál szöveg előtérszíne, n2 →normál szöveg háttérszíne
n3 →kiemelt szöveg előtérszíne, n4 →kiemelt szöveg háttérszíne
n5 →keret színe

Nem kötelező minden adat megadása, ugyanis ha például csak a keret színét szeretnénk kékre állítani, akkor azt megtehetjük a

```
SET COLOR TO ,,b
```

parancs segítségével. A használatos színkódok pedig a monitor adott színű nyalábjának bekapcsoltságától függően a következők:
N→fekete, B→kék, G→zöld, R→piros, W→fehér, RB→lila, GR→barna, BG→türkiz, X→rejtett, +→magasfényű (pl. GR+ sárga).
Monochrom monitornál: I→inverz, U→aláhúzott.

2.8.2. Állapotkijelzés

Az adatbázis kezelő állapotáról tájékoztatást kaphatunk a

```
DISPLAY STATUS
```

paranccsal, ahol a nyitott adatbázisok nevei, a funkcióbillentyűk szerepe és a környezeti paraméterek beállításai mellett egyéb adatokat is megtekinthetünk. A memóriában tárolt változókról a

```
DISPLAY MEMORY
```

parancs segítségével kaphatunk tájékoztatást, míg az aktív adatállomány szerkezetét a

```
DISPLAY STRUCTURE
```

parancs jeleníti meg.

2.8.3. Adatállományok részeinek másolása

A dBase-ben lehetőségünk van az adatállományok adott feltételnek eleget tevő részéről, vagy bizonyos mezőiről másolatot készíteni. Ehhez a

```
COPY TO  újállománynév [érv.kör] [feltétel]
         [FIELDS mezőlista]
```

parancsot kell kiadnunk, ahol az új állománynév a létrehozandó másolati adatállomány neve, a feltétel a FOR vagy WHILE paraméterekkel megadott feltétel, amelynek a másolati adatállományba bekerülő rekordoknak eleget kell tenniük, a mezőlista pedig azon mezők megnevezései, amelyek a létrehozandó adatbázisban megtalálhatók lesznek. Természetesen akár az érvényességi kört, akár a feltételt, akár a mezőlistát (egymástól függetlenül) elhagyhatjuk, így például a

```
COPY TO masik
```

paranccsal egy az aktív adatállománnyal teljesen megegyező adatállományt tudunk létrehozni `masik` néven.

Ha mondjuk egy olyan adatállományt szeretnénk létrehozni, amelyben csak a harminc év alatti emberek neve és születési ideje szerepeljen, akkor a

```
COPY TO  fiatal FOR (DATE()-szulido)/365<30 FIELDS
         nev, szulido
```

parancsot kell kiadnunk, természetesen egy sorba gépelve.

2.8.4. Adatállományok összefűzése

A dBase-ben lehetőségünk van több adatállomány felhasználásával egy megadott adatokat tartalmazó állományt készíteni. Ez olyan esetben lehet célszerű, amikor a szükséges adatok külön adatállományokban találhatók meg, de a későbbi felhasználás során ezekre egy állományban lesz szükség. Mindezt a

```
JOIN  WITH hivnév TO állománynév [FOR feltétel]
      [FIELDS mezőlista]
```

paranccsal tehetjük meg. Így ha például van egy ember nevű adatállományunk, amelyben megtalálható egy `nev` és egy `varos` nevű mező és egy adat nevű másik állományunk, amelyben pedig egy `nev` és egy `gkrdsz` mező, úgy készíthetünk egy harmadik, auto nevű adatállományt, amely az adott városban található gépkocsirendszámokat tartalmazza a

```
JOIN  WITH adat TO auto FOR nev=adat->nev
      FIELDS varos, adat->gkrdsz
```

parancs segítségével (amit természetesen egy sorba kell gépelnünk), ha az ember nevű adatállomány az aktív.

2.9. Adatbázis rendezése

Az adatbázisok rendezése alatt a rekordok adott sorrendbe (pl. ábécébe, vagy növekvő rendbe) történő áthelyezését értjük. Rendezés után a rendezett adatbázis rekordjaival könnyebben dolgozhatunk, hiszen azok egy általunk megkívánt egymásutánban követik egymást.

A dBase program segítségével kétféle módszer szerint rendezhetjük adatállományainkat: fizikailag és logikailag.

2.9.1. Fizikai rendezés

A legkézenfekvőbb megoldás az, amikor az adatállomány egy megadott mezőjében található adatok valamilyen sorrendje (Pl. a nevek ábécé sorrendje) szerint hozunk létre egy új adatállományt. Ily módon a kiindulási állomány rekordjai az új állományban már az általunk kívánt rend szerint fognak elhelyezkedni. Ezt a módszert nevezzük *fizikai rendezésnek*. Természetesen ha szükséges meghatározhatunk csökkenő sorrendet, vagy azonos rekordok esetén fennálló további rendezési szempontokat is. A fizikai rendezésnek azonban a nagyméretű állományok lassú rendezése mellett az is hátránya, hogy új rekord felvétele esetén az állomány ismét rendezetlenné válik, hiszen az új rekord az állomány végéhez íródik. A fizikai rendezés megvalósításához a

```
SORT ON mezőnév[lista] TO állománynév [érv.kör] [feltétel]
```

parancsot használhatjuk, ahol a mezőnév (amely nem lehet logikai vagy memo típusú) annak a mezőnek (vagy mezőknek) a neve, amely a sorrendet meghatározza, az állománynév pedig annak a létrejövő új állománynak a neve, amelyben már rendezve találjuk az adatokat. Fontos tehát, hogy a fizikai rendezés során az aktuális adatállomány változatlan marad, csak a létrejövő új állomány lesz rendezett. Emiatt tehát vagy le kell zárni az aktuális adatbázist, majd azt kitörölve az újat kell a régi nevére változtatni (pl. DOS parancsokkal), vagy pedig egy másik munkaterületre átlépve, oda betölteni az új rendezett adatállományt. (Pl. **SELECT B** majd **USE újállománynév** parancsokkal.) Amennyiben a rendezést nem növekvő sorrendben szeretnénk végezni, úgy a mezőnév után írt opciókkal ezt megváltoztathatjuk. Amennyiben a **/D** opciót írjuk, úgy a rendezés csökkenő sorrendben történik, amennyiben pedig **/C**-t úgy a dBase nem tesz különbséget kis és nagy-

betűk között. (Együttes használat esetén az opció **/CD**) Megtehetjük azt is, hogy az új rendezett állományba nem írjuk be az összes rekordot, ilyenkor az érvényességi kör, illetve a WHILE vagy FOR paraméteres feltétel megadása szükséges.

Ha tehát az aktív adatállományt szeretnénk a név mező ábécé sorrendje szerint rendezni úgy a

```
SORT ON nev TO ujfile
```

parancs kiadása után az `ujfile` nevű adatállományban már név szerint rendezve lesznek a rekordjaink. Ezt megnézni a

```
SELECT B
USE ujfile
DISP ALL
```

parancsokkal tudjuk (ez esetben célszerű, ha nem foglalt a B munkaterület). Ily módon például ha a rendezést elsősorban az irányítószám szerint szeretnénk végezni, azonos irányítószámok esetén pedig a nevet tartalmazó mező csökkenő sorrendje számítson, s a létrehozandó `masik` nevű rendezett adatállományba csak a nők szerepeljenek, akkor a

```
SORT ON irsz, nev/D TO masik FOR no
```

parancsot kell használnunk (ha létezik `no` nevű logikai mező).

2.9.2. Logikai rendezés

A dBase másik rendezési módszere a *logikai rendezés*, amely esetén az adatbázis rekordjaiból nem építünk fel új rendezett sorrendű adatbázist, csupán azt jegyezzük le, hogy a rekordoknak milyen sorrendben kell egymást követniük. Ez esetben tehát a gép egy másik, úgynevezett *indexállomány*t hoz létre, amely állományban a gép a valós adatállományban lévő rekordok sorrendjét tartja nyilván. Ebből eredően a logikai rendezést szokás indexelésnek nevezni. A logikailag rendezett állományt mindig az indexállományban megadott sorrendben látjuk, így kényelmesen bővíthetjük vagy módosíthatjuk, hiszen az indexállomány mindezeket képes automatikusan követni. Logikai rendezés esetén azonban a rekordmutató mozgatásánál óvatosan kell eljárnunk, hiszen az adott sorszámú rekordra állás (**GO n** parancs) a fizikai (azaz rendezetlen) sorrend szerint, a relatív mozgatás (**SKIP n** parancs) pedig a logikailag rendezett sorrend szerint történik. A logikailag rendezett állományt a későbbi felhasználás esetén történő újbóli megnyitáskor mindig az indexállománnyal együtt kell megnyitnunk, ellenkező esetben az állományt ismét rendezetlenül látjuk és ezért újrainde-

xelés válhat szükségessé. A logikai rendezés nagy előnye, hogy nagy adatállományok esetén is gyors, követi a változásokat, valamint az, hogy egyedi szempontok szerint is rendezhetünk. (Bizonyos dBase funkciók használatának feltétele a logikailag rendezett adatállomány.) A logikai rendezés megvalósításához az

```
INDEX ON kulcskifejezés TO indexállománynév
```

parancs kiadása szükséges, ahol a kulcskifejezés egy olyan mezőnév vagy összetett kifejezés, amely meghatározza a sorrendet, az indexállománynév pedig annak az NDX kiterjesztésű állománynak a neve, amit a dBase a rekordok sorrendjének tárolása céljából hoz létre (Ez többnyire lehet az adatállomány neve is.). Ha tehát az aktuális adatállományt név szerint szeretnénk logikailag rendezni, úgy az

```
INDEX ON nev TO nevindex
```

parancs hatására a rekordok sorrendjét a gép a `nevindex.ndx` állományba rögzíti, s emiatt a későbbiekben az adatállományt már rendezve tudjuk használni anélkül, hogy azt másik munkaterületen megnyitni kényszerülnénk. Amennyiben azonban kilépünk az adatbáziskezelőből és a későbbiekben ismét használni szeretnénk az adatállományt (amely neve legyen mondjuk EMBER), úgy a

```
USE ember INDEX nevindex
```

parancsot kell használnunk annak megnyitására. Amennyiben az **INDEX indexállománynév** részt kihagynánk az állományt megnyitó parancsból, úgy az adatállomány ismét a fizikai (nem rendezett) sorrend szerint válna láthatóvá.

Amennyiben a sorrendet meghatározó feltétel összetett, úgy nekünk kell meghatározni azt a kulcskifejezést, amely ezt az összetett feltételt leírja. Ha például most elsősorban a név első öt karaktere, ezek azonossága esetén pedig az irányítószám számít, úgy a parancs logikai rendezés esetén:

```
INDEX ON LEFT(nev,5)+STR(irsz) TO masikind
```

(Az indexelés a `masikind` nevű indexállományba történt.) A kifejezés megadásakor a **LEFT** függvénnyel levágtuk a név első öt karakterét, majd ennek a végéhez fűztük hozzá (konkatenációval: +) az irányítószám **STR** függvénnyel karakteresre alakított változatát.

Csökkenő sorrendű rendezést negatív előjellel érhetünk el. Például az irányítószám csökkenő sorrendje szerinti rendezés a `harmind` nevű indexállományba az

```
INDEX ON -irsz TO harmind
```

paranccsal valósítható meg.

Az adatállományokat természetesen mindig csak egy kulcs szerint (amely akár összetett is lehet) láthatjuk, de ezen indexelési (rendezési) módszerek között bármikor válthatunk. Ilyen esetben az adatállomány megnyitásával egyidőben több indexállományt nyitunk meg, melyek közül az elsőként megadott lesz a főindex, amely szerint látjuk rendezve az adatállományt. Például:

```
USE ember INDEX nevindex, masikind, harmind
```

A parancs esetén tehát az ember adatállomány a nevindex indexállomány szerint látható, de az adatállomány módosításakor a többi indexállomány is megváltozik. Amennyiben pedig a munkánk folyamán egy másik indexállományban meghatározottak szerint szeretnénk sorrendet látni, úgy nem kell mást tennünk, mint megváltoztatni a főindexet a

```
SET ORDER TO 2
```

paranccsal, s így máris a másodikként megadott (masikind nevű) indexállomány fogja a megjelenési sorrendet meghatározni.

Vigyáznunk kell azonban arra, hogy ha egy adatállományt úgy használunk, hogy a hozzá tartozó indexállomány nincs nyitva, úgy az adatállomány változásakor az indexállomány nem változik vele együtt. Ez azt eredményezi, hogy egy későbbi együttes használatkor az indexállomány már nem fogja az adatállomány sorrendjét helyesen meghatározni. Ez esetben újraindexelésre van szükség, amely vagy a korábbi **INDEX ON...** paranccsal, vagy pedig a

```
REINDEX
```

parancs kiadásával történik.

Természetesen az indexállomány nélkül megnyitott adatállományokhoz is nyithatunk utólag indexállományt (persze csak olyat, amelyet már korábban létrehoztunk), a

```
SET INDEX TO indexállomány[lista]
```

paranccsal, de fontos, hogy az indexállományt csak ahhoz az adatállományhoz nyithatjuk meg, amelyhez készítettük.

2.10. Keresés az adatbázisban

Ahhoz, hogy egy rekordon valamilyen parancsot végrehajthassunk a rekordmutatót az adott rekordra kell mozgatni. Amennyiben azonban nem a rekord sorszáma, hanem a tartalma ismert, ez elég nehéz feladat lenne csupán a korábbi ismeretek felhasználásával. A keresés funkcióval tehát lehetőségünk van a rekord tartalmára utalva mozgatni a re-

kordmutatót. A rendezéshez hasonlóan a dBase a keresésre is kétféle módszert kínál: a soros keresést és a gyorskeresést.

2.10.1. Soros keresés

Soros keresés esetén tehát az adatbázisban található, megadott tulajdonsággal rendelkező rekordot tudunk olymódon keresni, hogy a dBase az adatbázis elejéről indulva megvizsgálja a rekordokat, s annál a rekordnál áll meg, amelyre a feltétel igaz. A parancs formája tehát:

```
LOCATE FOR feltétel
```

Ha tehát az első olyan rekordra szeretnénk keresni, amelynél az irányítószám (`irsz` mező) 1165, akkor a

```
LOCATE FOR irsz=1165
```

parancsot kell használnunk. A feltétel persze bármely dBase által értelmezhető feltétel lehet, ily módon összetett is. Az első B betűvel kezdődő nevű 30 évnél idősebb emberre való keresés tehát a

```
LOCATE FOR nev='B'.AND. (DATE()-szulido)/365<30
```

paranccsal történik, ha a nevet a `nev`, a születési időt pedig a `szulido` mező tartalmazza. (A keresett rekordot megjeleníteni a **DISPLAY** paranccsal tudjuk.) Természetesen gyakran előfordul, hogy az adatállományban több ilyen rekord is található, a parancs pedig csak az elsőt keresi meg. Ilyenkor a keresés folytatása a

```
CONTINUE
```

paranccsal valósítható meg, amely a feltételnek eleget tevő következő rekordra állítja a rekordmutatót, feltéve, hogy van még ilyen rekord. (Amennyiben nem talál a feltételnek eleget tevő rekordot, úgy a rekordmutató az állomány végére áll.)

(A kereső parancsot ugyan érvényességi körrel is kiegészíthetjük, de ez igen ritkán fordul elő, hiszen többnyire mindig a teljes adatállományban keresünk.)

2.10.2. Gyorskeresés

Nagyméretű adatbázisban a soros keresés néha igen lassú is lehet, s ha ráadásul az adatbázisunk rendezve is van felesleges egyenként végigvizsgálni az összes adatot. A gyorskeresés parancsa olyan esetekben nyújt tehát segítséget, amikor az adatállományunk logikailag rendezett, és a keresést a főindex tartalma szerint szeretnénk megvalósítani. A keresésre ugyan a

```
FIND karaktersorozat | n
```

és a

```
SEEK kifejezés
```

parancs egyaránt használható, de mivel a **FIND** parancs után csak karaktersorozatot vagy számjegyet tartalmazó konstanst írhatunk, ritkábban használjuk, mint a **SEEK** parancsot, ahol a kifejezés bármilyen változó vagy konstans lehet. Ha tehát az adatállományunk név szerint rendezett, úgy gyorskereséssel csak adott névre tudunk keresni, más adatra nem. Például:

```
SEEK "Gipsz Jakab"
```

A keresés folytatását szolgáló parancsra gyorskeresésnél nincs szükség, mivel az adatállomány úgyis e szerint a mező szerint rendezett, így a ha van még ilyen rekord, akkor az biztosan a következő. (Ráállni a **SKIP** paranccsal tudunk.)

2.11. Kapcsolatteremtés adatállományok között

Sok esetben előfordul, hogy adataink nem minden esetben vannak egy adatállományban. Ilyen esetben párhuzamosan kell használnunk több adatállományt olymódon, hogy azok valamilyen kapcsolat révén összetartozó adatbázist alkossanak. Amikor az adatbázisok szerkezetéről beszéltünk, láthattuk, hogy a dBase egy relációs adatbázis kezelő rendszer, így emiatt az önálló adatállományainkat mi is ennek megfelelően kezelhetjük.

Természetesen dolgozhatunk külön-külön is az adatállományainkkal, de ez igen bonyolult, hiszen minden egyes művelet, amely több adatállományt érint több váltást és több keresést von maga után, hiszen hiába nyitunk meg több olyan adatállományt, amelyekben megtalálható ugyanazon mező, mégis ha az egyikben mozgást végzünk, az a másikra nem lesz hatással. Így persze olyan listákat sem tudunk készíteni, amelyekben több adatállományból származnak az adatok.

Ezen problémára nyújt megoldást az adatállományok összekapcsolása, hiszen ily módon egyszerre tudjuk mozgatni a különböző adatállományok rekordmutatóit. Természetesen az összekapcsolásnak szigorú feltételei vannak, így mindkét adatállománynak a kapcsolómező szerint indexelve (azaz logikailag rendezve) kell lenni. Ha tehát két olyan adatállományt szeretnénk összekapcsolni, ahol mindkettőben szerepel egy név mező, úgy e mező szerint kell az adatállományoknak indexelve lenniük.

Az összekapcsolást abból az adatállományból kiindulva kell kezdeményezni, amelyhez szeretnénk a másikat rendelni. Ez azért fontos, mert ily módon az aktív adatállomány minden elemét látjuk (azt is, amelyhez nem tartozik másik adatállománybeli rekord), viszont azokat a másik adatállománybeli rekordokat nem, amelyeknek az aktív adatállományban nincs megfelelője.

Az összekapcsolást így a

```
SET RELATION TO kulcs INTO kapcs.adatáll.név
```

paranccsal végezhetjük el, ahol a kulcs az a mindkét adatállományban megtalálható mező, amely szerint az állományok indexelve vannak, és amely szerint az összekapcsolás történik, a kapcsolandó adatállomány név padig annak az adatállománynak a neve, amelyet az aktív adatállományhoz kapcsolunk.

Az összekapcsolást követően tehát a rekordmutatók párhuzamosan mozognak, így ha valamely mezőre ráállunk, úgy a kapcsolt adatbázisban is arra a mezőre mozdul a rekordmutató (összekapcsolás nélkül egy helyben maradna). Ezt nagyon jól kihasználhatjuk vegyes listák készítésénél, ahol mindkét adatállományból vehetünk adatokat.

A következő példában nézzük meg, hogyan tudjuk az alábbi ember és adat nevű adatállományokat összekapcsolni, s mit kell ehhez tenni.

Az ember nevű adatállomány:

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	IRSZ	Numeric	4	0
3	CIM	Character	30	
4	NO	Logical	1	
5	SZULIDO	Date	8	

Az adat nevű adatállomány:

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	GKRDSZ	Character	7	
3	TELEFON	Character	15	
4	FIZETES	Numeric	6	0
5	JUTALOM	Numeric	5	0

Elsőként be kell tölteni az adatállományokat egy-egy munkaterületre, s név szerint indexelni kell őket:

```
SELECT A
USE ember
INDEX ON nev TO embnev
SELECT B
USE adat
INDEX ON nev TO adatnev
```

ha ezt megtettük, az ember adatállományt tartalmazó munkaterületre visszatérve megtehetjük az összekapcsolást:

```
SELECT A
SET RELATION TO nev INTO adat
```

Most már az adatállományok összekapcsolódtak, így kezelésük során az ezzel járó előnyöket ki is használhatjuk:

Készíthetünk listát, melyben a név, az irányítószám, a cím, a telefonszám és a rendszám szerepel:

```
LIST nev, irsz, cim, adat->telefon, adat->gkrdsz
```

A parancs kiadása ily módon az összetartozó adatokat fogja kiírni, hiszen a rekordmutatók párhuzamosan mozognak. (Összekapcsolás nélkül minden rekordhoz ugyanazt a telefonszámot és rendszámot írta volna ki a gép.)

Most írassuk ki azon negyven év alatti emberek nevét, címét és telefonszámát, amelyek fizetése nagyobb, mint ötvenezer forint:

```
LIST  nev, cim, adat->telefon FOR adat->fizetes>50000 .AND.
(DATE()-szulido)/365<40
```

Természetesen egy sorba gépelve.

Mindezt annak ellenére megtehettük, hogy a feltétel egyik részét képező életkort az ember, a másik részét képező fizetést pedig az adat nevű adatállományból tudjuk megkapni.

2.12. Rekordok elrejtése

Gyakran szükség van arra, hogy több parancson át csak ugyanazon feltételnek eleget tevő rekordokkal foglalkozzunk. Ilyen esetben felesleges minden parancsban megadni a feltételt, ugyanis a dBase lehetőséget nyújt egy globális szűrőfeltétel definiálására, amely megadása után már csak az annak eleget tevő rekordokkal tudunk dolgozni. Ilyen szűrőfeltételt megadni a

```
SET FILTER TO feltétel
```

paranccsal tudunk, ahol a feltételben kell megadni, hogy a későbbiekben mely rekordokkal dolgozhassunk. Ennek megfelelően, ha például a későbbiekben csak a B betűvel kezdődő nevű emberekkel akarunk foglalkozni úgy a

```
SET FILTER TO nev='B'
```

parancsot kell kiadnunk. A szűrőfeltételt törölni, azaz a használatot minden rekordra engedélyezni a

```
SET FILTER TO
```

paranccsal lehet.

2.13. Mezők elrejtése

A rekordokhoz hasonlóan mezőket is elrejthetünk a használat elől ha a

```
SET FIELDS TO mezőlista
```

paranccsal meghatározzuk a használható mezőket. Így tehát ha a későbbiekben a név, a cím és a születési időt meghatározó mezőkön kívül mással nem akarunk foglakozni, úgy a

```
SET FIELDS nev, cim, szulido
```

parancsot kell kiadnunk. Visszaállítani, azaz az összes mező megjelenítését engedélyezni a

```
SET FIELDS ALL
```

paranccsal lehet.

2.14. A dBase III+ programozása

Munkánk során gyakran van szükségünk arra, hogy bizonyos adatbázisok használata során felmerülő azonos jellegű feladatokat egyszerűen kezelhetővé tegyünk. A programozás segítségével megtehetjük, hogy az előre meghatározott tevékenységeket megtanítva a számítógépnek azokat egy gombnyomásra előhívjuk. Természetesen programokat csak akkor célszerű készíteni, ha gyakran kell ugyanolyan jellegű feladatokat ellátni, s előre, teljes mélységében ismerjük a későbbiekben elvégzendő feladatokat. A programozás, azaz a program készítése nem más, mint az elvégzendő feladatok felsorolása, azok elvégzésének feltételeinek és a végrehajtás körülményeinek definiálásával együtt. A programozás tehát nagyban hasonlít a kötegelt feldolgozáshoz, annyi kiegészítéssel, hogy most a dBase parancsnyelvén kell előállítanunk a programot tartalmazó szöveges állományt, s így lehetőségeink sokkal nagyobbak.

Mielőtt azonban elkezdjük megírni programunkat, gondoljunk át néhány dolgot. Ezért ha nem vagyunk gyakorlottak a dBase használatában, úgy a program kényesebb részeit próbáljuk ki először Interaktív módban.

Gondoljuk át, hogy milyen kimeneti adatokra lesz szükségünk és ezekhez milyen adatokat kell adatbázisban tárolnunk, és milyeneket kell a felhasználótól bekérnünk.

Tervezzük meg gondosan adatállományainkat, s ha a későbbiekben úgy látjuk jónak, ne sajnáljunk változtatni rajta.

Tervezzük meg, hogy a kívánt kimeneti adatokat hogyan, milyen algoritmus segítségével tudjuk előállítani, s ha lehet, próbáljuk is ezt ki.

Tervezzük meg a felhasználói felületet, határozzuk meg a programban végezhető lehetőségeket, találjuk ki a menüpontokat és azok szolgáltatásait.

Miután mindent alaposan átgondoltunk, jöhet a kódolás, azaz a program megfogalmazása dBase utasításokkal. Ezen utasítások sorozatát valamilyen szövegszerkesztő segítségével tudjuk gépre vinni olymódon, hogy az utasításokat az interaktív módhoz hasonlóan begépeljük. (Programozáskor az Enter megnyomását követően az utasítások nem kerülnek azonnal végrehajtásra, csupán a következő utasítás gépelését kezdhetjük meg egy új sorban.) Használhatjuk a saját kedvenc szövegszerkesztőnket is, ha képes ASCII formátum kezelésére, de a legkézenfekvőbb a dBase saját szerkesztőjét használni. Ehhez a

```
MODIFY COMMAND programnév
```

parancsot kell begépelni. (A dBase lehetőséget ad arra is, hogy ezen paranccsal a saját szövegszerkesztőnket indítsuk, ha a konfigurációs állományba ezt bejegyeztük.) A program neve a DOS szabályainak megfelelő név lehet, melyhez a dBase egy PRG kiterjesztést illeszt.

Természetesen a program megírásával még korántsem értünk feladatunk végéhez, a munka neheze még csak most jön. Az elkészült programot ugyanis hibátlanul működővé kell tenni. Ezt szinte soha sem ússzuk meg, hiszen még a legprofibb programozóknak sem sikerül elsőre tökéletes programot kreálni. Ne keseredjünk el tehát, ha elsőre nem megy, keressük meg a hibákat, teszteljük programunkat a leglehetetlenebb adatokkal (a leendő felhasználó ugyanis előre nem látható dolgokat is végezhet).

A kész programokat futtatni, azaz az elkészített programunkat indítani a

```
DO programnév
```

paranccsal tudjuk. (Ezt a parancsot a pont prompt után kell gépelni, nem pedig a programba!) Kiadása után a gép úgynevezett interpreterként viselkedik, ami azt jelenti, hogy az indított programot sorról sorra haladva utasításonként hajtja végre.

Amennyiben a végrehajtás során hiba lép fel, a program végrehajtása hibajelzéssel megszakad. A nem szintaktikai hibák esetén azonban egyáltalán nem biztos, hogy abban a sorban keletkezik a hiba, ahol a gép azt észleli, hiszen lehet, hogy egy három sorral korábban elvégzett rosszul beprogramozott, de végrehajtható számítás miatt nem tudott a kijelzett sorban elvégezni egy utasítást a gép.

Hiba bekövetkezésekor a program végrehajtása ideiglenesen felfüggesztésre kerül. A folytatás miként jéről a *Cancel, Suspend or Ignore?* kérdés lehetőségei közül választva tudunk továbblépni a megfelelő kezdőbetűk beütésével.

Cancel választása esetén a programvégrehajtás megszakad, a memóriaváltozók törlődnek, a programállomány lezáródik s visszakapjuk a promptot.(Ilyenkor célszerű nekilátni a program kijavításának.)

Suspend választása esetén a programvégrehajtás úgy szakad meg, hogy a programállomány nem záródik le (tehát nem javíthatjuk), viszont a prompt visszaadása miatt lehetőségünk van parancsok begépelésére, memóriaváltozók létrehozására vagy tartalmuk megtekintésére. A programvégrehajtás folytatását a

```
RESUME
```

paranccsal kérhetjük. Amennyiben végképp meg szeretnénk szakítani a programvégrehajtást úgy a

```
CANCEL
```

parancsot kell gépelnünk.

Az Ignore választása a hiba bekövetkezésétől függetlenül a következő utasításnál folytatja a program futtatását.

2.14.1. Programok készítése

Amikor programot készítünk, jó néhány olyan utasítás használata is szükségessé válik, amelyet párbeszédes módban értelmetlen volna használni. Ilyen utasítások lehetnek például a felhasználótól történő adatbekérés, a megjegyzések elhelyezése, vagy a feltételvizsgálatok nagy része. Ezen utasításokról tehát a programozáskor nem szabad megfeledkeznünk, hiszen egy program futásakor nem gondolhatjuk meg magunkat, hogy most akkor nem így, hanem úgy végzünk valamit.

A program írásakor a szintaktikai helyesség mellett tehát jó néhány egyéb szempontra is oda kell figyelnünk. Ezek közül megemlítendő, hogy a program végét célszerű egy

```
RETURN
```

utasítással jelölnünk. Ez tehát a program utolsó sora. Fontos az is, hogy ha úgynevezett páros utasításokat (pl. **IF ... ENDIF**) alkalmazunk, akkor soha nem szabad megfeledkezni annak mindkét tagjának kiírásáról. Amennyiben egy hosszabb programot írunk, előbb-utóbb elérkezünk ahhoz a ponthoz, amikor már nem tudjuk, hogy egy korábban elkészített programrészt miért is írtnuk oda. Ilyen esetekben segíthet az, ha a programban megjegyzéseket helyezünk el az új sorba gépelt

```
NOTE megjegyzés
```

vagy a

```
* megjegyzés
```

parancsok valamelyikével. Amennyiben utasítás mögött, vele azonos sorban szeretnénk megjegyzést írni úgy a

```
&& megjegyzés
```

karaktersorozatot kell a parancs után gépelni. Természetesen a programban üres sorokat is hagyhatunk, ezeknek funkciójuk nincs, viszont átláthatóbbá teszi a program tagolását.

A programok többsége úgy épül fel, hogy egy úgynevezett főmenüből választhatunk tevékenységeket, vagy almenüket, amelyekből újabb tevékenységek választhatók. Például a főmenüből választhatunk az új adat bevitele, adatmódosítás, törlés, megtekintés és kilépés menüpontok közül, ahol mondjuk a megtekintés menüpont kiválasztása egy olyan almenüt nyit meg, amelyben a különböző szempontok szerinti megtekintéseket támogató menüpontok közül választhatunk. Ezt a felépítést többnyire a következőkben megemlítendő ciklusokkal és esetszétválasztásos döntési szerkezetekkel tudjuk megvalósítani.

2.14.2. Változók használata

Míg interaktív módban elég ritka, addig a programozás során szinte nélkülözhetetlen a változók használata. Ez annak köszönhető, hogy az időközben keletkező, számunkra jelentéktelen, de a működéshez szükséges adatokat a programutasítások egymásnak csak változók segítségével tudják átadni. Ennek egyik legfontosabb példája az, amikor az adatot a program kezelőjétől kérjük be, s ennek függvényében kell a programnak különböző tevékenységeket végezni. Hasonlóan fontos terület lehet az is, amikor az adatokat a más módon szeretnénk megjeleníteni, mint ahogy rendelkezésre állnak, így ilyenkor az átmeneti tárolásukat változók segítségével végezzük.

Amennyiben azonban bármilyen okból is változót használunk a programban, sose felejtsük el azt, hogy azokat először definiálni kell (akár egy fiktív értékadással).

2.14.3. Adatmegjelenítés

Az adatok megjelenítése az adatbázis kezelés egyik legfontosabb feladata. Ez az adatmegjelenítés történhet egyszerű tájékoztatás miatt is, de gyakoribb az adatbázis adatainak, vagy azok megfelelő feldolgozás, átalakítás után történő közlése miatt is. Akár változóról, akár adatbázis adatáról legyen is szó, minden esetben szükséges az adatokat megfelelő módon áttekinthetően kivinni képernyőre vagy nyomtatóra. Ez többnyire nem az interaktív kezelésben használt parancsokkal a

legesztétikusabb, bár a hagyományos kérdőjeles módszer is alkalmazható. A képernyő megadott koordinátájú pontjára üzenetet írni a

```
@ sor,oszlop SAY üzenet
```

paranccsal tudunk, ahol az üzenet a megjelenítendő változó vagy karaktersorozat, a sor és oszlop pedig az elhelyezendő üzenet kezdetének megfelelő képernyő-koordináta. (Ennek megfelelően a sor 0 és 24, az oszlop – azaz a soron belüli pozíció – 0 és 79 közötti egész számot képviselő szám vagy változó.)

Ennek megfelelően tehát ha a képernyő tizedik sorának huszadik pozíciójától kezdődően szeretnénk kiíratni az „Ismerős neve:" szöveget, majd ezt követően folyamatosan a `nev` változó tartalmát, akkor a

```
@ 10,20 SAY 'Az ismerős neve:'+nev
```

parancsot kell kiadnunk, hiszen a megjelenítendő üzenetet konkatenációval (+) elő tudjuk állítani.

Lehetőség van a képernyő adott részének törésére, vagy körülvonalazására is a `@` paranccsal. Ha például a képernyő x,y és v,w koordinátái közötti területet szeretnénk törölni, akkor a

```
@ x,y CLEAR TO v,w
```

parancsot kell kiadnunk. A képernyő adott területének (jelen esetben x,y és v,w koordinátái közötti) körülvonalazását szimpla vonallal a

```
@ x,y TO v,w
```

dupla vonallal a

```
@ x,y TO v,w DOUBLE
```

paranccsal valósíthatjuk meg. (A parancsok valójában a terület szélére az ASCII kódtábla vonalas karaktereit helyezi.)

2.14.4. Adatbeviteli módszerek

A felhasználótól adatot bekérni igen sokféle okból szükséges lehet. Ez az adatbázisba való adatbeírás, a tevékenység kiválasztása, eseti műveletvégzés részadatának megkérdezése, viszonyítási adatok megkérdezése, illetve sok egyéb célból történhet.

A hagyományos adatbázisba közvetlenül történő adatbeírás a programokban igen ritkán használatos, szinte tilos, hiszen így a felhasználó szabadon garázdálkodhatna a teljes adatbázisban. Ilyen esetben legygyakrabban azt a módszert alkalmazzuk, amikor az adatbázisba írandó adatot először egy változóba töltjük, majd ezt a változót írjuk megfelelő ellenőrzések után az adatbázisba.

A legegyszerűbb adatbeviteli módszer az

```
INPUT [üzenet] TO memóriaváltozó
```

utasítás használata, ahol mint látható, az üzenet akár el is hagyható. Az utasítás végrehajtásakor a gép egy megadott memóriaváltozóba írja a begépelt adatot úgy, hogy a program végrehajtása akkor folytatódik, ha a felhasználó a kért adatot begépelte és megnyomta az Enter billentyűt. Egyetlen karakter bevitelére, vagy a program billentyűnyomásig történő felfüggesztésére alkalmas a

```
WAIT [üzenet] [TO memóriaváltozó]
```

parancs, amely végrehajtáskor csak akkor engedi a programot továbbfutni, ha megnyomtunk egy billentyűt. Ha az utasításnál begépeljük a TO memóriaváltozó részt is, akkor a megnyomott billentyűnek megfelelő karakter bekerül egy egy karakter hosszú memóriaváltozóba. Természetesen a megjelenítendő üzenet megadása sem kötelező. (Megjegyzendő, hogy a parancs végrehajtására hatástalanok a Shift, a Ctrl, az Alt, a Caps-Lock, a Scroll-Lock és a Num-Lock billentyűk.)
Adatbázismezőben is felhasználható adatok bevitelére leggyakrabban a

```
@ x,y GET memóriaváltozó
```

utasítást használhatjuk. Az utasítás azonban csak korábban definiált változóba, vagy mezőbe képes adatot vinni, olymódon, hogy az adatot a képernyő x-edik sorának y-odik pozíciójára gépelhetjük. Az utasítás végrehajtásának folyamán a memóriaváltozó korábbi értéke megjelenik a képernyőn, így azt kényelmesen átjavíthatjuk. További előny, hogy több `GET` utasítás kiadásával egyszerre több sornyi adatot tudunk kényelmesen bevinni. Ennek persze van egy feltétele is, mégpedig az, hogy a GET utasítást, vagy utasításokat mindig követnie kell egy

```
READ
```

utasításnak is, különben az adatbeolvasás nem történik meg. Például ha a 8. sor 15. pozícióján a `nev` változóba, a 9. sor 15. pozícióján pedig a `cim` változóba szeretnénk adatot bekérni, akkor a

```
@ 8,15 GET nev
@ 9,15 GET cim
READ
```

utasításokat kell begépelni. A GET utasítás további előnye az is, hogy a beolvasandó numerikus vagy dátum típusú adatnak határértékeket, vagy a karakteres típusú adatnak maszkot határozhatunk meg. Ezt úgy tudjuk megvalósítani, hogy a

```
@ x,y GET memóriaváltozó RANGE ah,fh
```

utasítással a RANGE szócska után beírjuk az alsó határt, majd ezt követően vesszővel elválasztva a felső határt. Az alsó és felső határ ter-

mészetesen nem csak konstans, hanem változó, vagy adatbázis mező is lehet. Ennek megfelelően a végrehajtáskor a határokon kívül eső adat begépelésekor a gép figyelmeztető hangjelzést ad, majd a szóköz billentyű megnyomását követően újbóli adatmegadásra készteti a felhasználót.

A maszk megadásával a beolvasandó karaktersorozat formáját határozhatjuk meg olymódon, hogy megadjuk, mely pozíción, milyen típusú karaktereket fogadhatunk el. Ennek megfelelően a

```
@ x,y GET memóriaváltozó PICTURE maszk
```

utasítás hatására csak olyan adatot fogad el a gép, amelyet előre meghatároztunk. A maszkban használható karakterek, amelyek az adott pozíción elfogadható karaktereket meghatározzák, a következők:

- **A** csak angol betűk
- **N** csak angol betűk és számjegyek
- **L** csak logikai adat
- **X** tetszőleges karakter
- **Y** logikai adatként csak Y és N
- **9** csak számjegyek (illetve előjel)
- **#** csak számjegyek (illetve előjel) és szóköz
- **$** a számok elejére **$** jel kerül a pénznem bevitelekor
- ***** a számok elejére ***** jel kerül
- **.** a tizedespont helyét határozza meg
- **,** a számok tagolására alkalmas szeparátor (csak szám bevitelénél jelenik meg például az ezresek tagolására, pl.: 12,345,678)
- **!** csak nagybetűket visz be (átalakítja a kisbetűket is nagybetűssé)
- **@** funkció karakter, amely után begépelt speciális karakter a teljes megjelenítést illetve bevitelt meghatározza.

Például a

```
@ 2,3 GET rendszam PICTURE 'AAA-999'
READ
```

utasítások a `rendszam` memóriaváltozóba egy olyan adatot olvasnak be, amelynek első három jegye betű, az utolsó három jegye szám, s a középső kötőjelet nem kell begépelni.

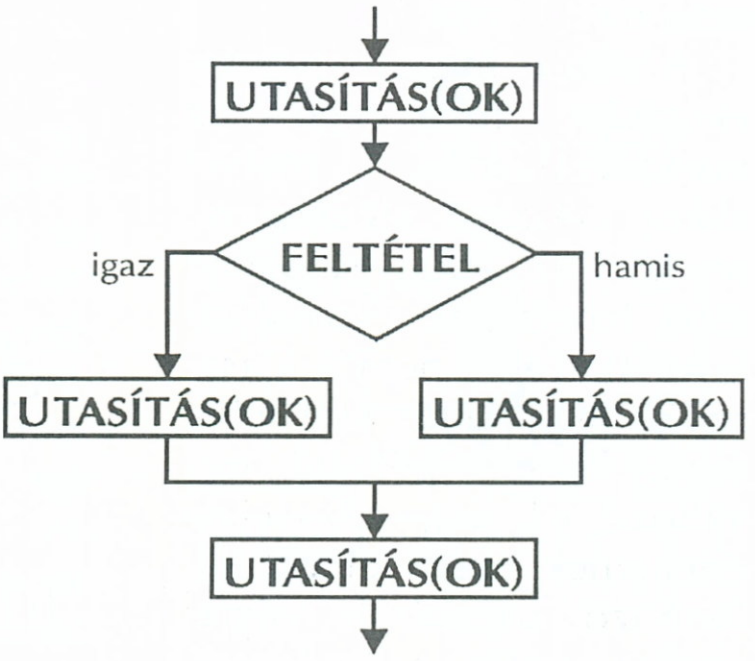
Amennyiben a programozás során az adatot memóriaváltozóba olvassuk be, úgy a beolvasás után az adatbázisba a memóriaváltozó értékét a már korábban megismert

```
REPLACE mezőnév WITH memóriaváltozó
```

utasítással tudjuk bevinni.

2.14.5. Feltételes elágazások

A programok készítésének egyik legfontosabb eszköze a folyamatok feltételtől függő szétválasztása. Mivel nem interaktív módban dolgozunk, a szituációkat nem tudjuk mi magunk megvizsgálni, majd annak eredményétől függően a következő tennivalónkat meghatározni. Meg kell tudni tehát előre mondani a gépnek, hogy egy olyan szituációban, ahol többféle módon is folytatódhat a program végrehajtása, mikor milyen utasításokat kell végrehajtani. A dBase az ilyen feltételes szerkezetek készítésére két módszert ismer: az egyszerű feltételes szerkezetet és esetszétválasztást.



Az egyszerű feltételes szerkezet esetében a feltétel teljesülése és nem teljesülésétől függően vagy így, vagy úgy folytatódhat a program végrehajtása. Külön utasításokat hajt végre a program, ha a feltétel teljesül (igaz), és külön más utasításokat, ha nem teljesül (hamis). A feltételes szerkezet megvalósítására az **IF** és **ENDIF** utasításpár használható:

```
IF feltétel
   a feltétel teljesülése esetén végrehajtandó utasítások
ENDIF
```

vagy

```
IF feltétel
   a feltétel teljesülése esetén végrehajtandó utasítások
ELSE
   a feltétel nem teljesülése esetén végrehajtandó utasítások
ENDIF
```

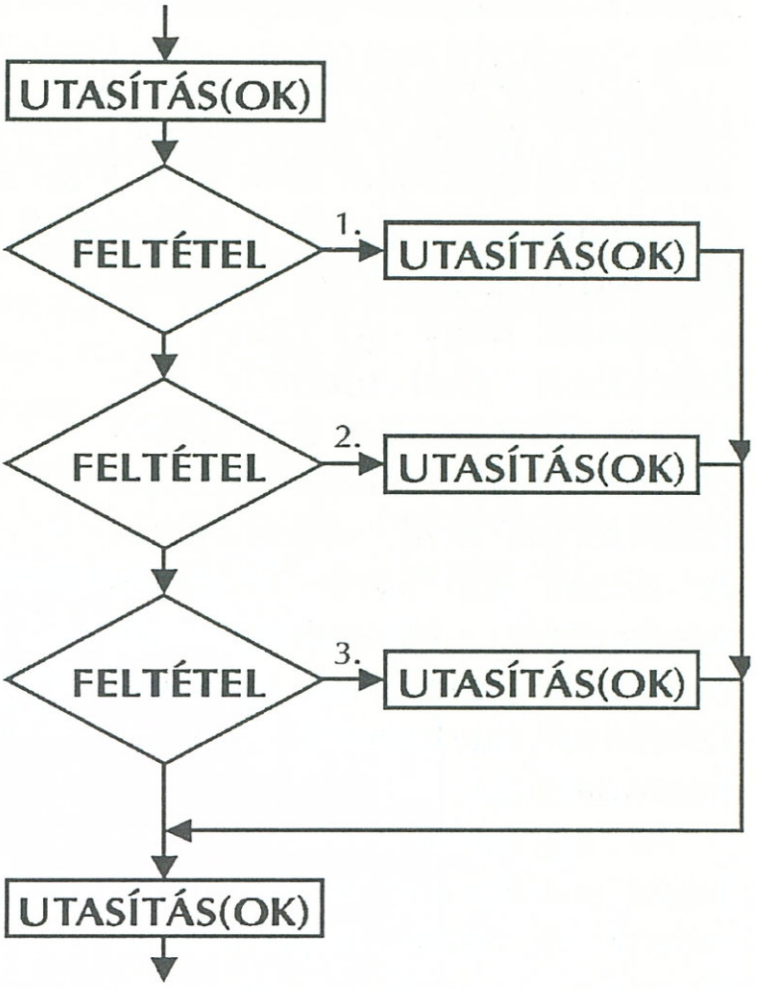
Ha tehát például azt szeretnénk megvalósítani, hogy ha a `fizetes` mező értéke kisebb mint 50000, akkor a fizetést emeljük meg 1000 forinttal akkor használhatjuk az

```
IF fizetes<50000
   fizetes=fizetes+1000
ENDIF
```

utasítássorozatot is, vagy a korábban megismert

```
fizetes=IIF(fizetes<50000,fizetes+1000,fizetes)
```

utasítást. Az **IF ENDIF** utasítások használatakor azonban soha ne feledkezzünk meg arról, hogy ezeket csak párban szabad alkalmazni, tehát az **ENDIF**-et soha ne felejtsük le.



A dBase másik feltételes elágazási lehetősége az esetszétválasztás, melyet akkor célszerű alkalmazni, amikor az adott szituáció több lehetséges alternatívát is kínál. Tehát nem a klasszszikus igaz-hamis eset lehet csak, hanem például egy változó n1, n2, n3, ... nx értékeinek megfelelően különböző utasítások végrehajtása válik szükségessé. Az utasítássorozat formája a következő:

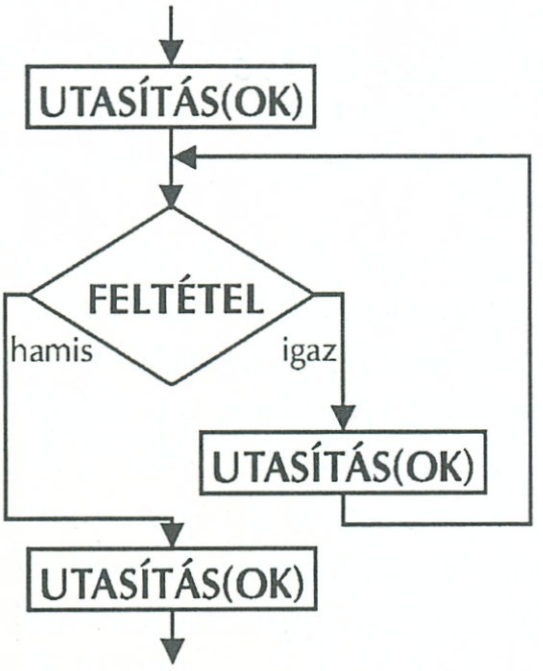
```
DO CASE
   CASE feltétel1
        feltétel1 teljesülése esetén végrehajtandó parancsok
   CASE feltétel2
        feltétel1 teljesülése esetén végrehajtandó parancsok
   CASE feltételn
        feltételn teljesülése esetén végrehajtandó parancsok
   OTHERWISE
        egyéb esetekben végrehajtandó parancsok
ENDCASE
```

Az utasítássorozat kiadása során természetesen tetszőleges számú ágat megadhatunk, s az **OTHERWISE** ág is adott esetben elhagyható. Például:

```
DO CASE
   CASE kod=1
        targy='autó'
   CASE kod=2
        targy='ház'
   CASE kod=3
        targy='mackó'
ENDCASE
```

Az utasítássorozatot követően a `targy` változó autó, ház és mackó értékeket vesz fel attól függően, hogy a `kod` változó értéke 1, 2, vagy pedig 3.

2.14.6. Ciklusok



A ciklusok használata elengedhetetlen feltétele a folyamatos programfutásnak. Amennyiben ugyanis nincs lehetőség arra, hogy a program egy adott tevékenység végeztével ismét valahonnan az elejéről folytassa a futását, úgy az egyszeri lefutást követően állandóan kilépne. A ciklusok segítségével megvalósíthatók olyan szerkezetek, amelyekben az utasítások mindaddig végrehajtódnak, ameddig egy feltétel teljesül. Természetesen a feltétel nem teljesülése esetén a program kilép a ciklusból és a ciklust követő utasításon folytatja végrehajtást. Bár a programozási rendszerek többféle ciklusszervezést is támogatnak, a dBase csak az elöltesztelős ciklust képes közvetlen utasítással megvalósítani. Ez azt jelenti, hogy a feltételt még a ciklus végrehajtása előtt kiértékeli, így annak nem teljesülése esetén a ciklus egyszer sem hajtódik végre. (A hátultesztelős ciklusok egyszer mindenképp lefuttatják a ciklusmagot alkotó utasításokat.) A ciklust tehát a következő utasításokkal tudjuk megvalósítani:

```
DO WHILE feltétel
   a ciklusmagot alkotó utasítások
ENDDO
```

Az utasítássorozat tehát a mindaddig végrehajtja a ciklusmagot alkotó utasításokat, amíg a feltétel teljesül. Fontos tehát, hogy a ciklus belsejébe mindenképp helyezzünk olyan utasítást, amely kihatással van a feltételre, ellenkező esetben ugyanis végtelen ciklust kaphatunk.
Például:

```
szam=0
DO WHILE szam<100
   INPUT "Kérek egy számot" TO szam
   valamennyi=valamennyi+szam
ENDDO
```

A fenti ciklus tehát mindaddig kér egy számot, amivel folyamatosan növeli a `valamennyi` nevű memóriaváltozó értékét, ameddig a megadott szám kisebb mint száz.

Másik eset, amikor egy valamilyen utasítássorozatot adott számúszor szeretnénk végrehajtani. Ennek megfelelően a következő ciklus segítségével például 100-szor irattatjuk ki a Hello szót:

```
n=0
DO WHILE n<100
   ? 'Hello'
   n=n+1
ENDDO
```

A ciklusok és a feltételes szerkezetek közös használatával kényelmes programkezelést realizálhatunk, ha menüszerkezeteket készítünk. A következő példa egy m darab főmenüpontot tartalmazó olyan programváz, melyben az n-edik főmenüpont p darab almenüpontot tartalmaz. A főmenü utolsó x-edik menüpontja (x értelemszerűen egyel nagyobb m-nél) szolgál a programból való kilépésre, míg az almenü y-odik menüpontja (y értelemszerűen egyel nagyobb p-nél) szolgál az almenüből a főmenübe való visszatérésre.

```
fm=0
DO WHILE fm<>x && ahol x a kilépésre szolgáló menüpont
menüpontok megjelenítésére szolgáló parancsok
WAIT " " TO fm
DO CASE
CASE fm=1
   első menüpontot realizáló parancsok
   CASE fm=2
   második menüpontot realizáló parancsok
   CASE fm=n
   n.-edik menüpontot realizáló parancsok
   am=0
   DO WHILE am<>y && ahol y a kilépésre szolgáló menüpont
   almenüpontok megjelenítésére szolgáló parancsok
   WAIT '' TO am
   DO CASE
      CASE am=1
      első almenüpontot realizáló parancsok
      CASE fm=2
      második almenüpontot realizáló parancsok
      CASE fm=p
      p.-edik almenüpontot realizáló parancsok
   ENDCASE
   ENDDO
   CASE fm=m
   m.-edik menüpontot realizáló parancsok
ENDCASE
ENDDO
```

2.14.7. Eljáráshívás

Hosszabb programok, vagy több programban is ismétlődő részek esetén lehetőségünk van bizonyos programrészleteket a főprogramból kiemelve külön állományban tárolni, így e módszereket eljáráshívásoknak nevezzük. A dBase két fajta eljáráshívást támogat. Az egyszerűbb, amikor egy adott programból egy másik programot hívunk meg a korábban már megismert **DO** utasítás segítségével. Ilyenkor az utasítást követően a megadott nevű önálló állományban található utasítások végrehajtódnak, majd ezt követően a **DO** utasítást követő sorban folytatódik a végrehajtás.

A másik módszer az úgynevezett procedúrahívás. Ekkor a főprogram elején a

```
SET PROCEDURE TO procedúraállománynév
```

utasítással meg kell nyitni a szintén prg kiterjesztésű procedúraállományt, majd a főprogram megfelelő részén a

```
DO procedúranév
```

utasítással hívhatjuk a procedúraállományban megtalálható adott nevű eljárást. A főprogram végén a procedúraállományt a

```
CLOSE PROCEDURE
```

paranccsal zárhatjuk le. A procedúraállomány önmagában nem futtatható programrészletek sorozatát tartalmazza olymódon, hogy a megadott nevű procedúra utasításainak kezdetét a

```
PROCEDURE procedúranév
```

utasítás, míg a végét a

```
RETURN
```

utasítás jelzi. A végrehajtás tehát a következőképpen zajlik: A főprogram mindaddig saját utasításait hajtja végre, amíg nem talál egy **DO procedúranév** parancsot. Ilyenkor fogja magát és a megnyitott procedúraállományban található azonos nevű **PROCEDURE procedúranév** sort követő utasításon folytatja a végrehajtást. Mindaddig a procedúraállományban található utasításokat hajtja végre, amíg nem ér el egy **RETURN** utasításhoz. Ez esetben ugyanis a végrehajtás a főprogram **DO procedúranév** sorát követő utasításánál folytatja a programfuttatást.

2.14.8. Programozási tanácsok

A korábbiakon túlmenően természetesen még jó néhány egyéb lehetőség is kínálkozik a programok készítése során, melyeknek csak a

programozó fantáziája szab határt. Ha azonban programot készítünk, a korábban már ismertetett szabályokon túl, nem árt ha néhány egyéb dolgot is szem előtt tartunk. Ahhoz, hogy a programunk futása során minél kevesebb hibalehetőség kínálkozzon, soha ne engedjük a felhasználókat közvetlenül az adatbázisba írni. Mindezt megtehetjük nagyon esztétikus adatbeviteli lehetőségek segítségével.

Kerüljük továbbá a teljesképernyős parancsokat, valamint ellenőrizzünk minden bevitt adatot.

Lehetőség szerint menürendszerű programokat készítsünk, ahol esztétikus, áttekinthető (és nem túldíszített) képernyőterveket alkalmazzunk. Gondoljunk arra, hogy az adott képernyő hogy mutat színes és monochrom monitoron.

A gyakrabban használt bevált utasítássorozatokat tárolhatjuk külön procedúraállományban, így azokat máshol is felhasználhatjuk.

Használjunk logikailag rendezett adatbázisokat.

Használjunk bátran olyan speciális utasításokat és függvényeket, amelyek csak program módban előnyösek. Ilyen utasítás például az **APPEND BLANK** is, amely az adott adatállomány végére üres rekordot szúr. Különösen ciklusokban előnyösek a következő függvények alkalmazása is:

RECCOUNT() megadja, hogy az adatbázis hány rekordból áll,

RECNO() megadja a rekordmutató jelenlegi pozícióját.

Célszerű továbbá a program elején bizonyos környezeti paramétereket beállítani:

```
SET DATE ANSI
SET TALK OFF
SET STATUS OFF
```

Természetesen ezeket állítsuk vissza a program végén.

Mivel a program megírását követően az valószínűleg nem fog elsőre tökéletesen működni, ezért szükség van olyan programtesztelési lehetőségek megismerésére is, amivel az esetleges hibákat könnyebben felderíthetjük. A szintaktikai hibákat könnyen javíthatjuk, hiszen a program azon soron áll meg, ahol a hibát észleli, így nincs más dolgunk, mint a sort tüzetesen ellenőrizni. A logikai hibák azonban sokkal alattomosabbak, így a tesztelés egyik igen fontos eszköze a változók ellenőrzése. Lehetőség van a lépésenkénti programvégrehajtásra a

```
SET STEP ON
```

paranccsal, vagy a végrehajtott utasítások képernyőre írására a

```
SET ECHO ON
```

paranccsal.

2.14.9. Mintaprogram

Az alábbi program, melynek segítségével nyilvántarthatjuk ismerőseink adatait, a korábban létrehozott EMBER és ADAT nevű adatállományok meglétét, vagy az alábbi adatállományok létrehozását igénylik:

Az EMBER nevű adatállomány

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	IRSZ	Numeric	4	0
3	CIM	Character	30	
4	NO	Logical	1	
5	SZULIDO	Date	8	

Az ADAT nevű adatállomány

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	TELEFON	Character	15	

Természetesen a program működőképes, de sokkal inkább alkalmas arra, hogy áttanulmányozva ötleteket merítsünk belőle, bizonyos szituációk megvalósítását kielemezzük, megértsük a programkészítés fortélyait, vagy egyedi igényeinknek megfelelően kiegészítve, módosítva saját céljainkat kielégítő programot készítsünk. Mindemellett hasznos ötletként szolgálhat a menüvezérelt programfelépítés megvalósításának módjában, illetve a különböző adatbeviteli és megjelenítési módszerek elkészítését illetően.

Célszerű a program begépelését követően azt úgy kipróbálni, hogy közben azt vizsgáljuk, mely utasítás milyen hatást vált ki.

```
set talk off
set help off
set console off
set confirm on
set status off
set date ansi
set century on
set safety off
close all
clear
@ 5,14 say 'Kis türelmet kérek, Adatok aktualizálása folyik.'
select a
use ember
select b
use adat
index on nev to adat
select a
index on nev to ember
set relation to nev into adat
fomenu=space(1)
almenu=space(1)
gomb=space(1)
jo=.f.
rendben=.f.
hiba=space(32)
vkp=space(1)
do while fomenu<>'0'
set color to b+,,n/n
clear
set color to b/b
@ 4,10 clear to 20,70
set color to rg+/rb+
@ 3,8 clear to 19,68
@ 3,9 to 19,67 double
@ 7,20  say '   FOMENU'
@ 10,20  say '1  Uj ember adatainak felvétele'
@ 11,20  say '2  Meglévő adatok módosítása'
```

```
@ 12,20  say '3  Törlés'
@ 13,20  say '4  Adatok megtekintése'
@ 14,20 say '0  Kilépés'
wait '' to fomenu
do case
case fomenu='1'
     hiba=space(32)
     vnev=space(20)
     virsz=1000
     vcim=space(30)
     vnem=space(1)
     vtel=space(1)
     vszulido=date()
     jo=.f.
     DO while .not. jo
        set color to b+,,n/n
        clear
        set color to b/b
        @ 4,10 clear to 20,70
        set color to rg+/rb+
        @ 3,8 clear to 19,68
        @ 3,9 to 19,67 double
        @ 7,12  say ' ADATOK FELVÉTELE'
        @ 10,12  say '              Név:'
        @ 11,12  say '     Irányítószám:'
        @ 12,12  say '              Cím:'
        @ 13,12  say 'Születési időpont:'
        @ 14,12 say '  Neme (n vagy f):'
        @ 15,12 say 'Van telefonja? in:'
        set color to r*/rb+
        @ 18,12 say hiba
        set color to rg+/rb+,rb+/n
        @ 10,31  get vnev
        @ 11,31  get virsz picture '9999'
        @ 12,31  get vcim
        @ 13,31  get vszulido
        @ 14,31 get vnem
        @ 15,31  get vtel
        read
        jo=.t.
        if upper(vnem)<>'F' .and. upper(vnem)<>'N'
           hiba='A nem csak F vagy N lehet!'
           jo=.f.
        endif
        if vnev=' '
           hiba='A név megadása kötelező!'
           jo=.f.
        endif
        seek vnev
        if found()
           hiba='Az adott név már létezik!'
           jo=.f.
        endif
     enddo
     append blank
     replace nev with vnev
     replace irsz with virsz
     replace cim with vcim
     replace szulido with vszulido
     vno=iif(upper(vnem)='N',.t.,.f.)
     replace no with vno
     if upper(vtel)='I' or upper(vtel)='Y'
        vtelszam=space(15)
        @ 16,12 say '      Telefonszam:'
        set color to r*/rb+
        @ 16,31  get vtelszam
        read
        select b
        append blank
        replace nev with vnev
        replace telefon with vtelszam
        select a
     endif
case fomenu='2'
     hiba=space(32)
     vnev=space(20)
     jo=.f.
     DO while .not. jo
        set color to b+,,n/n
        clear
        set color to b/b
        @ 4,10 clear to 20,70
        set color to rg+/rb+
        @ 3,8 clear to 19,68
        @ 3,9 to 19,67 double
```

```
   @ 7,15  say 'ADATOK MODOSITÁSA'
   @ 10,12 say '   Kérem a nevet:'
   set color to r*/rb+
   @ 18,12 say hiba
   set color to rg+/rb+,rb+/n
   @ 10,31  get vnev
   read
   jo=.t.
   if vnev=' '
      hiba='Nem adott meg nevet - javítás nem történik'
      @ 18,12 say hiba
      wait
   else
      seek vnev
      if .not. found()
         hiba='Nincs ilyen nevű ember!'
         jo=.f.
      else
         hiba=space(32)
         vtelszam=space(15)
         rendben=.f.
         do while .not. rendben
            @ 11,12 say '     Irányítószám:'
            @ 12,12 say '             Cím:'
            @ 13,12 say 'Születési időpont:'
            @ 14,12 say '  Neme (n vagy f):'
            @ 15,12 say '     Telefonszáma:'
            set color to r*/rb+
            @ 18,12 say hiba
            seek vnev
            virsz=irsz
            vcim=cim
            vszulido=szulido
            vno=no
            vnem=iif(no,'N','F')
            vtelszam=space(15)
            vtel='n'
            select b
            seek vnev
            if found()
               vtel='i'
               vtelszam=telefon
            endif
            select a
            set color to rg+/rb+,rb+/n
            @ 10,31 get vnev
            @ 11,31 get virsz picture '9999'
            @ 12,31 get vcim
            @ 13,31 get vszulido
            @ 14,31 get vnem
            @ 15,31 get vtelszam
            read
            rendben=.t.
            if upper(vnem)<>'F' .and. upper(vnem)<>'N'
               hiba='A nem csak F vagy N lehet!'
               rendben=.f.
            endif
            if vnev=' '
               hiba='A név megadása kötelező!'
               rendben=.f.
            endif
         enddo
         replace nev with vnev
         replace irsz with virsz
         replace cim with vcim
         replace szulido with vszulido
         vno=iif(upper(vnem)='N',.t.,.f.)
         replace no with vno
         if lower(vtel)='i'
            select b
            seek vnev
            replace telefon with vtelszam
            select a
         else
            if .not. vtelszam=' '
               select b
               append blank
               replace nev with vnev
               replace telefon with vtelszam
               select a
            endif
         endif
      endif
   endif
enddo
```

```
case fomenu='3'
     hiba=space(32)
     vnev=space(20)
     jo=.f.
     DO while .not. jo
        set color to b+,,n/n
        clear
        set color to b/b
        @ 4,10 clear to 20,70
        set color to rg+/rb+
        @ 3,8 clear to 19,68
        @ 3,9 to 19,67 double
        @ 7,15  say '   ADATOK TÖRLÉSE'
        @ 10,12 say '   Kérem a nevet:'
        set color to r*/rb+
        @ 18,12 say hiba
        set color to rg+/rb+,rb+/n
        @ 10,31  get vnev
        read
        jo=.t.
        seek vnev
        if .not. found()
           hiba='Nincs ilyen nevű ember!'
           jo=iif(vnev=' ',.t.,.f.)
        else
           delete
           pack
        endif
     enddo
case fomenu='4'
     almenu=' '
     do while almenu<>'0'
     set color to b+,,n/n
     clear
     set color to b/b
     @ 4,10 clear to 20,70
     set color to rg+/rb+
     @ 3,8 clear to 19,68
     @ 3,9 to 19,67 double
     @ 7,15  say '   MEGTEKINTÉS'
     @ 10,20 say '1  Teljes listázás'
     @ 11,20 say '2  Névrészlet szerinti keresés'
     @ 12,20 say '3  Címrészlet szerinti keresés'
     @ 13,20 say '4  Telefonszám szerinti keresés'
     @ 14,20 say '0  Kilépés'
     wait '' to almenu
     do case
        case almenu='1'
             go top
             clear
             sor=3
             @ 1,1  say 'Nev                    Irsz Cim'
             @ 1,58 say 'Szül. idő  Neme'
             @ 2,1  say '-------------------- ---- ------------------------------'
             @ 2,58 say '---------- -----'
             do while .not. eof()
                @ sor,1  say NEV
                @ sor,22 say IRSZ
                @ sor,27 say CIM
                @ sor,58 say SZULIDO
                @ sor,69 say IIF(no,'Nő','Férfi')
                if sor=22
                   sor=2
                   wait ''
                   @ 4,1 clear to 23,79
                endif
                skip
                sor=sor+1
             enddo
             wait ''
        case almenu='2'.or. almenu='3' .or. almenu='4'
             vnev=space(20)
             vcim=space(30)
             vtelszam=space(15)
             set color to b+,,n/n
             clear
             set color to b/b
             @ 4,10 clear to 20,70
             set color to rg+/rb+
             @ 3,8 clear to 19,68
             @ 3,9 to 19,67 double
             do case
                case almenu='2'
                     @ 7,15  say '   Névrészlet szerinti megtekintés'
                     @ 10,12 say '   Kérem a névrészletet:'
```

```
                    set color to rg+/rb+,rb+/n
                    @ 10,37 get vnev
                    read
                    locate for trim(vnev)$nev
                    if eof()
                       go 1
                    endif
                 case almenu='3'
                    @ 7,15  say '   Címrészlet szerinti megtekintés'
                    @ 10,12 say '   Kérem a címrészletet:'
                    set color to r*/rb+
                    set color to rg+/rb+,rb+/n
                    @ 10,37 get vcim
                    read
                    locate for trim(vcim)$cim
                    if eof()
                       go 1
                    endif
                 case almenu='4'
                    @ 7,15  say '   Telefonszám szerinti megtekintés'
                    @ 10,12 say '   Kérem a telefonszámot:'
                    set color to r*/rb+
                    set color to rg+/rb+,rb+/n
                    @ 10,37 get vtelszam
                    read
                    locate for trim(vtelszam)$adat->telefon
                    if eof()
                       go 1
                    endif
              endcase
              set color to rg+/rb+
              @ 3,8 clear to 19,68
              @ 3,9 to 19,67 double
              gomb='x'
              do while .not. gomb=' '
                 @ 10,12 say '              Név: '+NEV
                 @ 11,12 say '     Irányítószám: '+ltrim(str(IRSZ))
                 @ 12,12 say '              Cím: '+CIM
                 @ 13,12 say 'Születési időpont: '+dtoc(SZULIDO)
                 @ 14,12 say '  Neme (n vagy f): '+IIF(no,'Nő','Férfi')
                 @ 15,12 say '     Telefonszáma: '+adat->telefon
                 @ 18,12 say '- vissza   + előre  * következő   spc kilépés'
                 wait '' to gomb
                 if .not. eof() .and. gomb='+'
                    skip
                    if eof()
                       skip -1
                    endif
                 endif
                 if .not. bof()  .and. gomb='-'
                    skip -1
                 endif
                 if gomb='*'
                    poz=iif(eof(),1,recno())
                    continue
                    if .not. found()
                       go poz
                    endif
                 endif
              enddo
     endcase
     enddo
endcase
enddo
set talk on
set console on
set status on
close all
```

2.15. A dBase parancsok összefoglalója

Mivel az eddigiekből néhány ritkábban alkalmazott parancs illetve paraméter használata nem minden esetben került teljes részletességgel ismertetésre, ezért az alábbiakban összefoglaljuk a dBase parancsait, azok funkcióit. (Emlékeztetőül megjegyezzük, hogy a szögletes záró-

jelben lévő kifejezések elhagyhatók, a szögletes zárójelek nem begépelendők, a | jelek pedig a vagylagos kapcsolatot jelölik. Pl: **GO TOP | BOTTOM | Nkif** esetén gépelhető a **GO TOP** a **GO BOTTOM** vagy a **GO 3** parancs is. Értelemszerűen a **WAIT ['üzenet'][TO változó]** helyett a **WAIT** a **WAIT 'várj'** a **WAIT 'várj' TO bvalt** vagy a **WAIT TO bvalt** parancsok használhatók.)

? kif[lista] megjeleníti a következő sor elején az adott kifejezés[listá]t.

?? kif[lista] megjeleníti a kurzor pozíciójától az adott kifejezés[listá]t.

! DOS-parancs Végrehajtja az adott DOS parancsot

@ sor,oszlop[SAY kif [PICTURE maszk]][GET vált [PICTURE maszk]] [RANGE alsóhat, felsőhat]] megadott formátumú adatmegjelenítés SAY, adatbekérés GET esetén.

@ sor,oszlop CLEAR adott méretű terület törlése a képernyő jobb alsó sarkától kezdődőleg.

@ sor1,oszlop1 [CLEAR] TO sor2,oszlop2 [DOUBLE] terület törlése vagy keret rajzolása.

&& megjegyzés elhelyezése parancssor végén

* megjegyzés elhelyezése program vagy procedúraállományban külön sorban

ACCEPT [adat] TO vált határolójelek nélküli adat bevitele memóriaváltozóba.

APPEND [BLANK] állomány végéhez adatrekord hozzáfűzése. BLANK esetén üres rekord hozzáfűzése.

APPEND FROM fájlnév [FOR felt] [TYPE] [DELIMITED [WITH határolójel | BLANK | fájltípus] másik adatállományból való rekordhozzáadás.

ASSIST menüvezérelt használat bekapcsolása.

AVERAGE [mezőkif-lista][érvényességi kör][WHILE feltétel][FOR feltétel][TO vált[lista]] mezők átlagának számítása [és változóba helyezése]

BROWSE [FIELDS mezőlista] [LOCK Nkif] [FREEZE mezőnév] [NOFOLLOW] [NOMENU] [WIDTH Nkif] [NOAPPEND] teljesképernyős szerkesztés bekapcsolása [LOCK: bal szélen rögzíthető mezők, FREEZE: egyetlen mező javítása, WITH: a mezőkijelzés maximális hossza, NOFLOW: rekordmutató helye index szerinti átrendezésnél, NOAPPEND: hozzáfűzési lehetőség nélkül, NOMENU: opciók menüjének használata nélkül].

CALL modulnév [WITH Kkif | vált] végrehajt egy gépi kódú modult, melyet a LOAD paranccsal töltünk be.

CANCEL programvégrehajtás felfüggesztése.

CASE feltétel elágazáság meghatározása DO CASE parancs esetén.

CHANGE [FIELDS mezőlista][érvényességi kör][WHILE felt][FOR felt] adott rekordok teljesképernyős szerkesztése.

CLEAR a képernyő letörlése.

CLEAR ALL a memóriaváltozók törlése és az állományok zárása.

CLEAR FIELDS minden SET FIELDS paranccsal kijelölt mező törlése.

CLEAR GETS a @GET paranccsal beolvasott adatok törlése.

CLEAR MEMORY a memóriaváltozók törlése.

CLEAR TYPEAHEAD a billentyűzetpuffer törlése.

CLOSE ALL | ALTERNATE | DATABASES | FORMAT | INDEX | PROCEDURE a megadott típusú állományok zárása.

CONTINUE a LOCATE paranccsal megkezdett keresés folytatása.

COPY FILE forrásállománynév TO célállománynév állomány másolása.

COPY STRUCTURE fájlnév [mezőlista] az aktív adatbázis szerkezetének másolása.

COPY TO fájlnév [érvényességi kör] [FIELDS mezőlista] [WHILE feltétel] [FOR feltétel][TYPE] [DELIMITED [WITH határolójel]] | fájltípus az aktív adatállományról való másolat készítése.

COUNT [érvényességi kör] [WHILE feltétel] [FOR feltétel] [TO vált] feltételnek eleget tevő rekordok darabszámának meghatározása, esetleg memóriaváltozóba helyezése.

CREATE fájlnév új adatállomány létrehozása.

CREATE fájlnév FROM szerkezetállománynév szerkezetállomány segítségével új adatállomány létrehozása.

CREATE LABEL fájlnév | ? cimkeformátum-állomány létrehozása.

CREATE QUERY fájlnév | ? szűrőfeltétel-állomány létrehozása.

CREATE REPORT fájlnév | ? listaformátum-állomány létrehozása.

CREATE SCREEN fájlnév | ? képernyőformátum állomány létrehozása.

CREATE VIEW fájlnév FROM ENVIRONMENT munkakörnyezet áttekintő állományba való másolása.

DELETE[érvényességi kör] [WHILE feltétel] [FOR feltétel] rekord(ok) törlésre jelölése.

DELETE FILE fájlnév állomány törlése.

DIR [path] adatállomány jellemzők megjelenítése.

DISPLAY [érvényességi kör] [kifejezéslista] [OFF] [WHILE feltétel] [FOR feltétel] [TO PRINT] képernyőoldalankénti rekordmegjelenítés (OFF esetén rekordsorszám nélkül.)

DISPLAY FILE [LIKE fájlnévcsoport] [TO PRINT] katalógus megjelenítése.

DISPLAY HISTORY [LAST Nkif] [TO PRINT] a korábban használt parancsok megjelenítése.

DISPLAY MEMORY [TO PRINT] memóriaváltozók megjelenítése.

DISPLAY STATUS [TO PRINT] az adatállomány és környezeti paramétereinek megjelenítése.

DISPLAY STRUCTURE [TO PRINT] adatállomány szerkezetének megtekintése.

DISPLAY USERS megjeleníti a hálózat aktív munkaállomásait.

DO fájlnév | procedúranév [WITH paraméterlista] program indítása, procedúra hívása [WITH paraméterben átadott változókkal].

DO CASE feltételes elágazás

DO WHILE feltétel a feltétel teljesüléséig futó ciklus létrehozása. A ciklust ENDDO-val zárjuk.

EDIT teljesképernyős rekordtartalom módosítás.

EJECT lapdobás nyomtatón.

ELSE feltételes elágazás az IF feltétel nem teljesülése esetén.

ENDCASE DO CASE elágazásrendszer záró utasítása.

ENDDO DO WHILE ciklus záró utasítása.

ENDIF IF paranccsal kezdett feltételrendszer záró utasítása.

ENDTEXT a TEXT és ENDTEXT parancsok közötti sorokat szövegként kezeli.

ERASE fájlnév állománytörlés

EXIT feltétel nélküli kilépés a DO WHILE ciklusból.

EXPORT TO fájlnév TYPE PFS FPS típusú fájl készítése.

FIND karaktersorozat | Nkif adott rekord keresése indexelt állományban.

GO TOP | BOTTOM | Nkif rekordmutató állítása adott rekordra.

GOTO TOP | BOTTOM | Nkif rekordmutató állítása adott rekordra

HELP [kulcsszó] segítségkérés.

IF feltétel feltételes elágazásrendszer lérehozása.

IMPORT FROM fájlnév TYPE FPS dBase állomány létrehozása FPS típusú állományból.

INDEX ON kulcs TO fájlnév [UNIQUE] adatállomány indexelése (UNIQUE esetén az azonos kulcsú rekordok közül csak az első kerül az indexállományba).

INPUT [üzenet] TO vált adatbevitel memóriaváltozóba

INSERT [BEFORE] [BLANK] rekord beszúrása a rekordmutató után (BEFORE esetén elé, BLANK esetén üres rekord).

JOIN WITH alias TO új-fájlnév FOR feltétel [FIELDS mezőlista] adatállományok összefésülése.

LABEL FORM fájlnév | ? [SAMPLE] [érvényességi kör] [WHILE feltétel] [FOR feltétel] [TO PRINT] [TO [FILE] fájlnév] címke nyomtatása vagy szöveges állomány készítése a címkeformátum-állományban meghatározottak szerint.

LIST [kifejezéslista] [érvényességi kör] [WHILE feltétel] [FOR feltétel] [OFF] [TO PRINT] adatállomány rekordjainak listázása.

LIST HISTORY [LAST Nkif] [TO PRINT] korábban végrehajtott parancsok megjelenítése.

LIST MEMORY [TO PRINT] változók megjelenítése.

LIST STATUS [TO PRINT] a dBase állapotának és a nyitott adatállományok jellemzőinek megjelenítése.

LIST STRUCTURE [TO PRINT] adatállomány-szerkezet megjelenítése.

LOAD bináris-fájlnév gépi kódú programrutin betöltése.

LOCATE [érvényességi kör] [FOR feltétel] [WHILE feltétel] feltételnek eleget tevő rekord keresése.

LOOP ciklus elejére ugrás a DO WHILE feltételének újbóli kiértékelése mellett.

LOGOUT hálózati kilépés.

MODIFY COMMAND fájlnév adott nevű program szerkesztése beépített vagy megadott szövegszerkesztő segítségével.

MODIFY LABEL fájlnév | ? adott nevű címkeformátum-állomány szerkesztése.

MODIFY QUERY fájlnév | ? adott nevű szűrőfeltétel-állomány szerkesztése.

MODIFY REPORT fájlnév | ? adott nevű listaformátum-állomány szerkesztése.

MODIFY SCREEN fájlnév | ? adott nevű képernyőformátum-állomány szerkesztése.

MODIFY STRUCTURE adatállomány szerkezetének megváltoztatása.

MODIFY WIEV fájlnév | ? megtekintő-állomány szerkesztése.

NOTE megjegyzés megjegyzés elhelyezésére szolgáló parancs.

ON ERROR | SCAPE | KEY parancs hiba esetén végzendő tevékenység meghatározása.

OTHERWISE a DO CASE ciklus meg nem határozott feltétele esetén végrehajtandó parancsainak kezdő sora.

PACK a törlésre jelölt rekordok végleges fizikai törlése.

PARAMETERS paraméterlista átvett változók definiálása procedúrában.

PRIVATE [ALL [LIKE | nem egyértelmű változónév]] | [változólista] lokális (csak az adott procedúrában használható) változók definiálása.

PROCEDURE procedúranév adott procedúra kezdő utasítása procedúraállományban.

PUBLIC változó-lista globális (minden procedúrában és programszinten használható) változók definiálása.

QUIT kilépés a dBase-ből. (lezárja az adatállományokat is.)

READ [SAVE] adatbevitel a program korábban írt GET utasításaiban meghatározott változóiba. [SAVE esetén a következő GET-ig történő kép-elmentés].

RECALL [érvényességi kör] [WHILE feltétel] [FOR feltétel] törlésre jelölés megszüntetése.

REINDEX adatállomány újraindexelése (csak korábban indexelt állomány esetén).

RELASE [változó-lista] | [ALL [LIKE | EXCEPT nem egyértelmű változónév]] | [MODULE modulnév] adott változók illetve gépi kódú rutinok memóriából való törlése.

RENAME régi-állománynév TO új-állománynév állomány átnevezése.

REPLACE [érvényességi kör] mezőnév WITH kif [,mező2név WITH kif2 [,...]] [FOR feltétel] [WHILE feltétel] mezőtartalom helyettesítése.

REPORT FORM fájlnév | ? [érvényességi kör] [FOR feltétel] [WHILE feltétel] [[PLAIN] | [HEADING fejléc]] [NOEJECT] [TO PRINT] [TO [FILE] fájlnév] [SUMMARY] adatformátum-lista készítése. PLAIN: fejléc tiltása, HEADING: fejléc definiálása, NOEJECT: nyomtatás előtti lapdobás nélkül, TO PRINT: nyomtatóra, TO FILE: állományba, SUMMARY: csak összegek megjelenítése.

RESTORE FROM fájlnév [ADDITIVE] korábban mentett változók betöltése háttértárolóról. ADDITIVE: a memóriában lévő változók megmaradnak (az azonos nevűek természetesen felülíródnak).

RESUME programfutás folyatatása (SUSPEND-del történt megszakítás esetén).

RETRY procedúra vagy program végrehajtásának befejezése.

RETURN [TO MASTER] procedúra befejezése visszatéréssel.

RUN DOS-parancs DOS parancs végrehajtása.

SAVE TO fájlnév [ALL LIKE | EXCEPT nem egyértelmű változónév] változók háttértárolóra mentése.

SEEK kif a kifejezést tartalmazó indexkulcsú rekord keresése indexelt adatállományban.

SELECT munkaterület | alias | Nkif munkaterület váltás.

SET környezeti paraméterek állítása (részletesen a könyv környezeti paraméterekkel foglalkozó részében ismertetve.)

SKIP Nkif rekordmutató elmozdítása Nkif értékével.

SORT TO új-állománynév ON mező [/A][/C][/D][,mező2 [/A][/C][/D] [,...]] [érvényességi kör] [FOR feltétel] [WHILE feltétel] rendezett sorrendű adatállomány-másolat készítése. /A: emelkedő sorrend, /C: kis és nagybetűk között nem tesz különbséget, /D: csökkenő sorrend.

STORE kifejezés TO memóriaváltozó[lista] értékadás változóknak.

SUM [érvényességi kör] [kifejezéslista] [FOR feltétel] [WHILE feltétel] [TO memóriaváltozó[lista]] numerikus mezők összegének megjelenítése vagy változóba helyezése.

SUSPEND programvégrehajtás felfüggesztése.

TEXT a TEXT és ENDTEXT parancsok közötti sorokat szövegként kezeli.

TOTAL TO új-fájlnév ON kulcs [érvényességi kör] [FOR feltétel] [WHILE feltétel] [FIELDS mezőlista] összegzetállomány létrehozása rendezett, numerikus mezőket tartalmazó adatállományból.

TYPE fájlnév [TO PRINT] szöveges állomány képernyőre [vagy nyomtatóra] írása.

UNLOCK [ALL] hálózatban zárolt rekord [ALL: összes] hozzáférését engedélyezi más felhasználók részére.

UPDATE ON kulcsmező FROM alias [RANDOM] REPLACE mező WITH kifejezés [,mező2 WITH kifejezés2 [,...]] másik nyitott adatállományból való adatállomány aktualizálás. [RANDOM: indexelt állomány esetén.]

USE [fájlnév | ? [INDEX fájlnév-lista][ALIAS alias]] [EXCLUSIVE] adatállomány megnyitása az aktív munkaterületre. [INDEX indexállománnyal együtt történő megnyitás, ALIAS: hivatkozási név megadása, EXCLUSIVE: hálózatban kizárólagos használatra.]

WAIT ['üzenet'][TO változó] várakozás billentyűlenyomásra [TO változó: a megnyomott billentyűnek megfelelő karakter változóba tárolása.]

ZAP törli az aktív adatállomány összes rekordját.

2.16. Ellenőrző kérdések

1. Milyen jellegű feladatokat tud megoldani a dBase programmal?
2. Ismertesse az adatbázis, adatállomány, mező és rekord fogalmakat!
3. Milyen adatbázis-szerkezetek léteznek?
4. Ismertesse a dBase kezelési módjait!
5. Ismertesse a fontosabb környezeti paramétereket!
6. Sorolja fel a dBase által használt állománytípusokat!
7. Sorolja fel az adattípusokat és azok jellemzőit!
8. Ismertesse a dBase adatállományainak felépítését!
9. Definiálja az állandó és változó fogalmát!
10. Ismertesse a változókkal végezhető műveleteket!
11. Ismertesse a gyakoribb függvényeket!
12. Mire szolgálnak a konverziós függvények?
13. Mutassa meg az értéknövelés módját!

14. Ismertesse az adatállomány létrehozásának módját!
15. Hogyan tud megnyitni létező adatállományt?
16. Hogyan tudja az adatállomány szerkezetét megváltoztatni?
17. Határozza meg a rekordmutató fogalmát!
18. Ismertesse a rekordmutató mozgatásának lehetőségeit!
19. Mutassa be a While és For paraméterek használatát!
20. Hogyan tud új rekordot hozzáfűzni adatállományhoz?
21. Hogyan tud új rekordot beszúrni adatállományba?
22. Ismertesse a rekord törlésének módját!
23. Hogyan tudja a rekordok tartalmát megváltoztatni?
24. Ismertesse az adatok megjelenítésének módszereit!
25. Ismertesse az adatok feltételes megjelenítésének módszereit!
26. Ismertesse a mezők helyettesítésének módszereit!
27. Hogyan tud összeget és átlagot számolni?
28. Ismertesse a nyomtatási lehetőségeket!
29. Mi a különbség a logikai és fizikai rendezés között?
30. Ismertesse a logikai és fizikai rendezés módját!
31. Ismertesse az indexelt adatbázisok használatával kapcsolatos tudnivalókat!
32. Ismertesse a soros- és gyorskeresés módszereit!
33. Hogyan tud kapcsolatot teremteni adatállományok közt?
34. Ismertesse a szűrők használatát!
35. Ismertesse a programkészítés módját dBase-ben!
36. Ismertesse az adatbeviteli és adatmegjelenítési lehetőségeket!
37. Ismertesse a feltételes elágazások használatának módját!
38. Ismertesse a ciklusok használatának módját!
39. Ismertesse az eljáráshívások használatának módját!
40. Ismertesse a programtesztelési lehetőségeket!
41. Mire kell figyelni programok készítése során!
42. Hogyan tud meglévő programot indítani?

2.17. Gyakorló feladatok

Ebben a pontban a korábbiakban tanultak gyakorlására találunk mintafeladatokat. A feladat jelzésére a ☞ jelet, a megoldások jelzésére a ✍ jelet használtuk.

☞ **1.** Állítsa be a dátumot a magyar írásmódnak megfelelően!

✍ Gépeljük be az alábbi parancsot:

```
SET DATE ANSI
```

☞ **2.** Hozza létre az alábbi szerkezetű adatbázist EMBER névvel:

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	IRSZ	Numeric	4	0
3	CIM	Character	30	
4	NO	Logical	1	
5	SZULIDO	Date	8	

✍ Gépeljük be az alábbi parancsot:

CREATE EMBER

majd teljesképernyős szerkesztéssel vigyük fel az adatbázis szerkezetét. Az utolsó megjelenő üres sorba nyomjunk Entert.

☞ **3.** Töltsük fel az adatbázist az alábbi rekordokkal:

Record#	NEV	IRSZ	CIM	NO	SZULIDO
1	Aba Aladin	1111	Budapest Lámpa u. 1.	.F.	12.12.12
2	Mekk Elek	1010	Budapest Kecske tér 4.	.F.	56.10.23
3	Kovács Kázmér	7620	Pécs Vas u. 73.	.F.	67.01.02
4	Tóth Töhötöm	4025	Debrecen Alföldi u. 11.	.F.	81.12.31
5	Víz Elek	7400	Kapospart u. 49.	.F.	49.03.15
6	Zuzmó Zénó	8000	Székesfehérvár Kék u. 9	.F.	49.09.09
7	Cica Cecília	7013	Cece Macska u. 1.	.T.	70.01.01
8	Gipsz Jakab	9873	Alsókutykurutty Cement u. 4.	.F.	61.01.01
9	Kerti Virág	3996	Füzér Rózsa u. 42.	.T.	69.06.11
10	Kék Ibolya	7621	Pécs Mecseki u. 93.	.T.	68.02.29

✍ Vagy a létrehozás után feltett adatfelvitelre vonatkozó kérdésre válaszolunk Y-nal, vagy begépeljük az

APPEND

parancsot. Ezután hozzákezdhetünk az adatok teljesképernyős felvitelének. Az utolsó üres rekordnál nyomjunk végig Entert, így befejezhetjük az adatfelvitelt.

☞ **4.** Hozza létre az alábbi szerkezetű adatbázist ADAT névvel a második munkaterületen:

Field	Name	Type	Width	Dec
1	NEV	Character	20	
2	GKRDSZ	Character	7	
3	TELEFON	Character	15	
4	FIZETES	Numeric	6	0
5	JUTALOM	Numeric	5	0

✍ Gépeljük be az alábbi parancsokat:

SELECT 2
CREATE ADAT

majd teljesképernyős szerkesztéssel vigyük fel az adatbázis szerkezetét. Az utolsó megjelenő üres sorba nyomjunk Entert.

☞ **5.** Töltsük fel az adatbázist az alábbi rekordokkal:

Record#	NEV	GKRDSZ	TELEFON	FIZETES	JUTALOM
1	Aba Aladin	MRC-001	12-34-657	121000	0
2	Mekk Elek	ABC-123	345-45-67	43700	0
3	Kovács Kázmér			38000	0
4	Tóth Töhötöm	BZN-999	(06-52)34-56-78	72320	0
5	Víz Elek	VIZ-123		38000	0
6	Zuzmó Zénó	RND-345	333-22-11	0	0
7	Cica Cecília	CIC-191		42000	0
8	Gipsz Jakab	GIP-521		35240	0
9	Kerti Virág			27000	0
10	Kék Ibolya	CLS-999	(06-72) 456-789	48000	0

✍ Vagy a létrehozás után feltett adatfelvitelre vonatkozó kérdésre válaszolunk Y-nal, vagy begépeljük az

APPEND

parancsot. Ezután hozzákezdhetünk az adatok teljesképernyős felvitelének. Az utolsó üres rekordnál nyomjunk végig Entert, így befejezhetjük az adatfelvitelt.

☞ **6.** Listázzuk ki a teljes adat nevű adatállományt!

✍ Gépeljük be az alábbi parancsot:

```
LIST
```

☞ **7.** Listázzuk ki a teljes ember nevű adatállományt!

✍ Gépeljük be az alábbi parancsokat:

```
SELECT A
LIST
```

☞ **8.** Listázzuk ki az ember nevű adatállományt úgy, hogy csak a nevek és a címek jelenjenek meg (még a rekordsorszám sem)!

✍ Gépeljük be az alábbi parancsot:

```
LIST nev, cim OFF
```

☞ **9.** Listázzuk ki a budapesti férfiak adatait!

✍ Gépeljük be az alábbi parancsot:

```
LIST FOR UPPER(cim)='BUDAPEST' .AND. .NOT. NO
```

☞ **10.** Listázzuk ki a 25 év alatti K betűvel kezdődő emberek adatait.

✍ Gépeljük be az alábbi parancsot:

```
LIST FOR (DATE()-SZULIDO)/365<25 .AND. nev='K'
```

☞ **11.** Hozzunk létre egy HAPSI nevű adatállományt az ember adatállomány férfi rekordjainak név és cím mezőiből.

✍ Gépeljük be az alábbi parancsot:

```
COPY TO hapsi FOR .NOT. no FIELDS nev, cim
```

☞ **12.** Listázzuk ki az adat adatbázisból az e betűt tartalmazó nevű emberek nevét és fizetését!

✍ Gépeljük be az alábbi parancsokat:

```
SELECT B
LIST nev, fizetes FOR 'e'$nev
```

☞ **13.** Listázzuk ki az átlag alatti fizetésű emberek adatait!

✍ Gépeljük be az alábbi parancsokat:

```
AVERAGE fizetes TO atlagfiz
LIST FOR fizetes<atlagfiz
```

☞ **14.** Nyomtassuk ki azon emberek adatait, melyek nevének harmadik betűje k.

✍ Gépeljük be az alábbi parancsot:

```
LIST FOR SUBSTR(nev,3,1)='k' TO PRINT
```

☞ **15.** Listázzuk ki azon emberek nevét, melyeknek gépkocsi-rendszámuk számjegye 300-nál kisebb!

✍ Gépeljük be az alábbi parancsot:

```
LIST nev FOR VAL(SUBSTR(gkrdsz,5,3))<300
```

☞ **16.** Készítsünk egy állományt `sorbavan` névvel, amely az emberek adatait azok nevének ábécé sorrendjében tartalmazza, úgy hogy a kis és nagybetűk között ne legyen különbség!

✍ Gépeljük be az alábbi parancsot:

```
SORT ON nev/C TO sorbavan
```

☞ **17.** Rendezzük logikailag mindkét megnyitott állományunkat név szerint! (A *B* munkaterületen tartózkodunk.)

✍ Gépeljük be az alábbi parancsokat:

```
INDEX ON nev TO adatind
SELECT A
INDEX ON nev TO emberind
```

☞ **18.** Váltsunk vissza B munkaterületre, majd listázzuk ki név szerinti sorrendben az adatokat!

✍ Gépeljük be az alábbi parancsot:

```
SELECT B
LIST
```

(Miután az előbb pont név szerint rendeztünk elegendő csak a LIST)

☞ **19.** Kapcsoljuk össze a két adatbázist!

✍ Gépeljük be az alábbi parancsot:

```
SET RELATION TO nev INTO adat
```

☞ **20.** Listázzuk ki az átlagtól 20%-nál jobban eltérő fizetésű emberek nevét fizetését és címét.

✍ Gépeljük be az alábbi parancsokat:

```
AVERAGE adat->fizetes TO atlagfiz
LIST  nev, adat->fizetes, cim FOR
      adat->fizetes>atlagfiz*1.2 .OR.
      adat->fizetes<atlagfiz*0.8
```

vagy

```
AVERAGE adat->fizetes TO atlagfiz
LIST  nev, adat->fizetes, cim FOR
      abs(adat->fizetes-atlagfiz)>atlagfiz*0.2
```

A beljebb kezdett sorokat folyamatosan kell gépelni.

☞ **21.** Adjon az embereknek a fizetésük 30%-ának megfelelő jutalmat, de legalább 10000 forintot!

✍ Gépeljük be az alábbi parancsot:

```
REPLACE adat->jutalom WITH
        MAX(10000,adat->fizetes*0.3)
```

☞ **22.** Adjon az ötven év feletti nőknek 10% fizetésemelést, de maximum 4000 forintot!

✍ Gépeljük be az alábbi parancsot:

```
REPLACE adat->fizetes WITH
        MIN(adat->fizetes*1.1,adat->fizetes+4000)
        FOR (DATE()-SZULIDO)/365>50 .AND. no
```

vagy

```
REPLACE adat->fizetes WITH
        adat->fizetes+MIN(adat->fizetes*0.1,4000)
        FOR (DATE()-SZULIDO)/365<25 .AND. no
```

☞ **23.** Menjen a B munkaterületre, majd ott jelölje törlésre azon rekordokat, amelyeknél a rendszám és a név azonos betűvel kezdődik!

✍ Gépeljük be az alábbi parancsokat:

```
SELECT B
DELETE FOR LEFT(nev,1)=LEFT(gkrdsz,1)
```

☞ **24.** Keressen rá az első olyan rekordra, amelynél a keresztnév (!) Z betűvel kezdődik!

✍ Gépeljük be az alábbi parancsot:

```
LOCATE FOR SUBSTR(nev,AT(" ",nev)+1,1)='Z'
```

(A névben lévő szóközt követő karakternek kell Z-nek lennie)

☞ **25.** Zárja be az összes adatbázist és lépjen ki a programból!

✍ Gépeljük be az alábbi parancsot:

```
QUIT
```

3. Adatbázis-kezelés az Excel táblázatkezelővel

3.1. Fogalmak

3.1.1. Adat

Értelmezhető (észlelhető), érzékelhető, felfogható, megérthető ismeret.

3.1.2. Információ

Új ismeretté értelmezett adat. Adatok, amiket felismertünk, értelmeztünk, megértettünk és ismereteket nyújtottak számunkra.

3.1.3. Adatbázisok

Szövegszerűen leírt tényszerű adatbank.

3.1.4. Egyedek

Egyednek nevezünk minden olyan dolgot (objektumot), ami minden más dologtól megkülönböztethető, és amit ismeretekkel akarunk leírni. Idegen szóval ezt entitásnak nevezzük. Az ismeretekkel leírandó jelenségek absztrakt (elvont) osztályait egyedtípusokkal tükrözzük. Pl. kocsi, cipő, stb.

3.1.5. Tulajdonság

Az egyedeket tulajdonságaikkal írjuk le. A jelenségeket leíró ismeretek absztrakt osztályait tulajdonságtípusokkal tükrözzük. Tulajdonság idegen szóval Attribútum. Az egyedek közötti viszonyok fogalmi tükörképét kapcsolatnak nevezzük. A jelenségek közötti viszonyok absztrakt osztályait kapcsolattípusokkal tükrözzük.

3.1.6. Adatmodell

Tükrözik az általuk leképezett rendszer legfontosabb információelemeit (egyedeit) és ezen információelemek kapcsolatait. Az adatmodellek fogalmi kategóriák, azt írják le, hogy az egyedek hogyan épülnek fel. Tehát, egyedek tulajdonságainak és kapcsolataiknak halmaza, melyek absztrakt módon tükrözik mindezek elvont kategóriáit. Véges számú egyedtípusnak, azok egyenként is véges számú tulajdonságtípusainak és kapcsolattípusainak szervezett együttese.

3.1.7. Adatbázis

Véges számú egyed előfordulásnak azok egyenként is véges számú tulajdonságértékeinek és kapcsolat előfordulásának az adatmodell szerint szervezett együttese. Logikailag összefüggő információ vagy adatgyűjtemény. Szerkezetét az adatmodellek írják le. Az egyedek konkrét előfordulásait tartalmazzák. Adatbázist alkotnak, pl. a bélyegalbum bélyegei, a telefonregiszter bejegyzései vagy a főkönyv forgalmi adatai, vagy például egy kartotékrendszer, amely egy cég dolgozóinak adatait tartalmazza. A kartotékok mindegyike egy-egy dolgozóra vonatkozik (belső összefüggés), a dolgozók mindegyike ugyanannál a cégnél dolgozik (külső összefüggés). Gondoljuk most a kartotékrendszert olyan táblázatnak, amelynek sorai az egyes kartotékokon szereplő adatokat tartalmazzák nevek, címek, irányítószámok, telefonszámok, ezek mindegyike külön oszlopba kerül. Az ilyen táblázatot, vagy táblázatok összefüggő rendszerét adatbázisnak nevezzük. A táblázatban az egyes kartotékok egy-egy ember adatait tartalmazzák, ezek egy sorba kerülnek, és rekordoknak nevezzük. Egy-egy rekordba az egyes egyedek tulajdonságai vannak leírva, ezeket mezőknek nevezzük. Az egyedek azonos tulajdonságai egy táblázatban oszlopokba van rendezve.

3.1.8. Táblázat

Táblának nevezzük a logikailag összetartozó (tehát azonos objektumot leíró) adatok sorokból és oszlopokból álló elrendezését. A táblák sorait adatrekordoknak, az oszlopait pedig rekordmezőknek mondjuk.

3.1.9. Adatbázis-kezelés

Valahányszor egy adatbázishoz fordulunk, hogy új adatokat adjunk hozzá, meglévő adatokat módosítsuk, jelentéseket, lekérdezéseket készítsünk, vagy az adatokat átrendezzük, adatbázis-kezelést végzünk.

Ilyen tevékenységek elvégzésére hozták létre az adatbázis-kezelő rendszereket.

3.1.10. Relációk, relációs adatbázisok

Relációs adatbázis az olyan adatbázis, amelyik több összekapcsolt táblázatból áll. Olyan programot nevezünk relációs adatbázis-kezelő rendszernek, amelyik az adatokat számítógépen táblázatokban tárolja, rendezi, illetve onnan keresi vissza. A reláció az adatelemek megnevezett, összetartozó csoportjából kialakított olyan kétdimenziós táblázat, amelyik sorokból és oszlopokból áll, és ahol az oszlopok egy-egy tulajdonságot írnak le. Ahhoz, hogy egy táblázatot relációnak lehessen tekinteni, a következő feltételeket kell kielégítenie:

- Nem lehet két egyforma sora.
- Minden oszlopnak egyedi neve van.
- A sorok sorrendje tetszőleges.
- Az oszlopok sorrendje tetszőleges.

Sor-szám	Megne-vezés	Dátum	Érték	Pénz-nem	Árfo-lyam	Érték FT	Költség-vetés
0001	Benzin	98.05.01	11,2	USD	200	2240	Utazás
0002	Gázolaj	98.05.02	12	USD	200	2400	Utazás
0003	Fénymásoló	98.05.03	1300	DEM	100	130000	Fejlesztés
0004	Számítógép	98.05.03	2500	DEM	100	250000	Fejlesztés
0005	Számítógép	98.05.03	1700	DEM	100	170000	Fejlesztés
0006	Szoftver	98.05.03	1300	USD	205	266500	Fejlesztés
0007	Benzin	98.05.07	15	USD	205	3075	Utazás

Figyeljük meg, hogy a fenti táblázat kielégíti a feltételeket, tehát reláció, mégpedig olyan, amelyik a vásárlást nyolc tulajdonságával:

- a rekord sorszámával,
- az áru megnevezésével,
- a vásárlás időpontjával,
- az elköltött valuta mennyiségével,
- a valuta típusával,
- a valuta árfolyam értékével,
- az áru értéke forintban,
- a vásárlás céljával

írja le. Látható, hogy a reláció sorai konkrét egyed-előfordulásokat, oszlopai pedig konkrét tulajdonságértékeket adnak meg. A relációs adatbázisok általában nem egy, hanem több logikailag összekapcsolt relációból (táblázatból) állnak.

3.1.11. Az adatbázis elkészítése

Kisebb adatbázisok elkészítésével nincs különösebb gond, de a nagyobb méretű adatbázisokat célszerű előre megtervezni. Ehhez a feladatot meg kell fogalmazni, részfeladatokra bontani, majd az adatok összegyűjtése, elemzése következik. Az elemzés után megtervezzük az egyes részfeladatokhoz szükséges táblákat.
A táblákat úgy kell meghatározni, hogy

- minden mező álljon közvetlen kapcsolatban a tábla témakörével,
- ne tartalmazzanak származtatott, kiszámított és feleslegesen ismétlődő adatot. A kiszámított adatok a lekérdezésekben jelenjenek meg.
- az egyes mezők elemi információt tartalmazzanak.

Egy adatbázis-kezelő programhoz készülő táblázatokat ezek után egy többlépcsős normalizálási eljárással 1, 2, 3, 4, 5NF-jú (normál forma) alakra kell hozni. Általában 3NF formátummal már megelégedhetünk. Az Excelben erre nincs szükség a kis méretek miatt, ezért a normalizálási eljárásra nem térünk ki.

Ne készítsünk nagyméretű táblázatokat, mert mi sem és mások sem látják át az egyes mezők közötti összefüggéseket.

3.2. Adatbázisok létrehozása, rendezése, szűrése, lekérdezések

Az Excel-ben rendkívül egyszerű létrehozni egy adatbázist. Elég az adatbázis első sorában megadnunk az oszlopok neveit, és máris vihetjük be az adatokat. Nem szükséges az adatbázis blokknak még nevet sem adni, de ha a formázó szalagon a stílusablakban megadjuk az **adatbázis** nevet, akkor a rendszer a blokk első sorát automatikusan, mint mezőneveket tartalmazó sort fogja felhasználni. Továbbiakban a megadott **adatbázis** névvel tudunk hivatkozni a táblázatunkra. Az alábbi táblázat példa egy kisebb adatbázisra. Az első sor tartalmazza az oszlopok neveit. Az első oszlop (sorszám) az egyedek azonosítója, melynek egyediségéről az Excelben sajnos nekünk kell gondoskodni.

Az adatbevitelre használhatjuk az Adatok menü Rekordonként parancsát. Ilyenkor egy párbeszédpanel segítségével, (melyet a rendszer generál), vihetjük be az adatokat rekordonként, de sokkal egyszerűbb közvetlenül kitölteni a táblázat mezőit

Sor-szám	Megnevezés	Dátum	Érték	Pénznem	Árfolyam	Érték FT	Költségvetés
0001	Benzin	98.05.01	11,2	USD	200	2240	Utazás
0002	Gázolaj	98.05.02	12	USD	200	2400	Utazás
0003	Fénymásoló	98.05.03	1300	DEM	100	130000	Fejlesztés
0004	Számítógép	98.05.03	2500	DEM	100	250000	Fejlesztés
0005	Számítógép	98.05.03	1700	DEM	100	170000	Fejlesztés
0006	Szoftver	98.05.03	1300	USD	205	266500	Fejlesztés
0007	Benzin	98.05.07	15	USD	205	3075	Utazás
0008	Gázolaj	98.05.08	13,5	USD	205	2767,5	Utazás
0009	Szálloda	98.05.19	890	USD	205	182450	Ellátás
0010	Irodabérlet	98.05.19	600	USD	205	123000	Bérlet
0011	Igazgató	98.06.01	520	USD	205	106600	Fizetés
0012	Titkárnő	98.06.01	220	USD	205	45100	Fizetés
0013	Tolmács	98.06.01	270	USD	205	55350	Fizetés
0014	Testőr 1	98.06.01	350	USD	205	71750	Fizetés
0015	Testőr 2	98.06.01	350	USD	205	71750	Fizetés
0016	Irodabútor	98.06.01	96500	YEN	0,8	77200	Fejlesztés
0017	Telefon	98.06.02	25000	YEN	0,8	20000	Működés
0018	Felszerelés	98.06.03	14000	YEN	0,8	11200	Működés
0019	Irodaszerek	98.06.04	800	DEM	105	84000	Működés
0020	Frissítő	98.06.05	2500	YEN	0,8	2000	Működés
0021	Irodaszerek	98.06.06	1800	YEN	0,6	1080	Működés
0022	Gázolaj	98.06.07	11	USD	210	2310	Utazás

3.2.1. Keresés az adatbázisban

Adatok keresésére a táblázatban a **VKERES()** és az **FKERES()** függvények kiválóan alkalmasak. Ezeket a függvényeket legegyszerűbben, a függvényvarázslóval helyezhetjük el a megfelelő cellákba.

FKERES(keresendő_érték;adatbázis;oszlopszám;keresési_tartomány)

A függvény az adatbázis első oszlopában megkeresi a keresendő értéket, vagy a közelítő értékét a keresési_tartomány paraméter állapotától függően, és a megadott oszlopszámban található mező értékét adja vissza. Ha a keresési tartomány paraméter igaz értéket tartalmaz, akkor a keresés stílusa kisebb vagy egyenlő és ennek a relációnak megfelelő legnagyobb értéket adja vissza a megadott számú oszlopból.

Nagyon fontos, hogy az adatbázis első oszlopának rendezettnek kell ilyenkor lenni, mégpedig növekvő sorrendben, különben az Fkeres függvény hibás értéket adhat vissza. Amennyiben az utolsó paraméter „hamis", akkor csak abban az esetben ad vissza értéket, ha talál az első oszlopban a keresendő értékkel pontosan megegyezőt, ellenkező esetben a „#HIÁNYZIK!" értéket adja vissza. Logikai HAMIS keresés esetén az adatbázis első oszlopának nem kell rendezettnek lenni. Ha az oszlopszám paraméter egynél kisebb, akkor az FKERES az „#ÉRTÉK!" hibaértéket adja vissza, ha nagyobb oszlopszámot adunk meg, mit amennyi az adatbázisban van, akkor „#HIV!" hibaértéket kapunk visszza.

A VKERES() függvény hasonlóképpen működik, csak vízszintesen kell a keresendő értéket megadni. Akkor célszerű ezt a függvényt használni, ha az összehasonlítandó érték az adatbázis, vagy tömb első sorában van, és a visszaadandó érték alatta lévő valamelyik sorban van.

	A	B	C	D	E	F	G	H	I
1									
2									
3		Darabszám	Árengedmény						
4	0-9	0	0%			Darabszám		41	
5	10-19	10	5%			Árengedmény		20%	
6	20-29	20	10%						
7	30-39	30	15%						
8	40-49	40	20%						
9	>50	50	25%						

A bemutatott példánkban az áruk vásárolt darabszámától függően kapnak a vásárlók árkedvezményt. A H4 cellába beírjuk a vásárolt áru darabszámát, és a H5 cellába írt fkeres() függvény kikeresi a beírt darabszámhoz tartozó árkedvezményt. Ehhez a H5 cellába a következőt kell beírni.

=FKERES(H4;B4:C9;2;IGAZ)

Ahol a **H4** cella a keresendő szám, a **B4:C9** tartomány az adatbázis, amiben a számot keresni kell, a **2** annak az oszlopnak a száma, amelyből az értéket visszaadja a függvény, az **IGAZ** paraméter pedig a közelítő érték keresését engedélyezi.

Amennyiben az adatbázis rendelkezik egyedi kulccsal, egy olyan oszloppal melyben nincs két egyforma tartalommal rendelkező mező

(példánkban a sorszám oszlop), akkor ez a kulcs kitűnően felhasználható az egyedek keresésére. Felhasználva kis adatbázisunkat, adatokat gyűjthetünk ki a sorszám megadásával.

A C2 mezőbe beírt sorszám alapján a rendszer azonnal kikeresi a megadott sorszámhoz tartozó adatokat a C3 – C9 cellákba írt fkeres() függvény segítségével.

C3: **=fkeres(C2;adatbázis;2)**
C4: **=fkeres(C2;adatbázis;3)**
C5: **=fkeres(C2;adatbázis;4)**
C6: **=fkeres(C2;adatbázis;5)**
C7: **=fkeres(C2;adatbázis;6)**
C8: **=fkeres(C2;adatbázis;7)**
C9: **=fkeres(C2;adatbázis;8)**

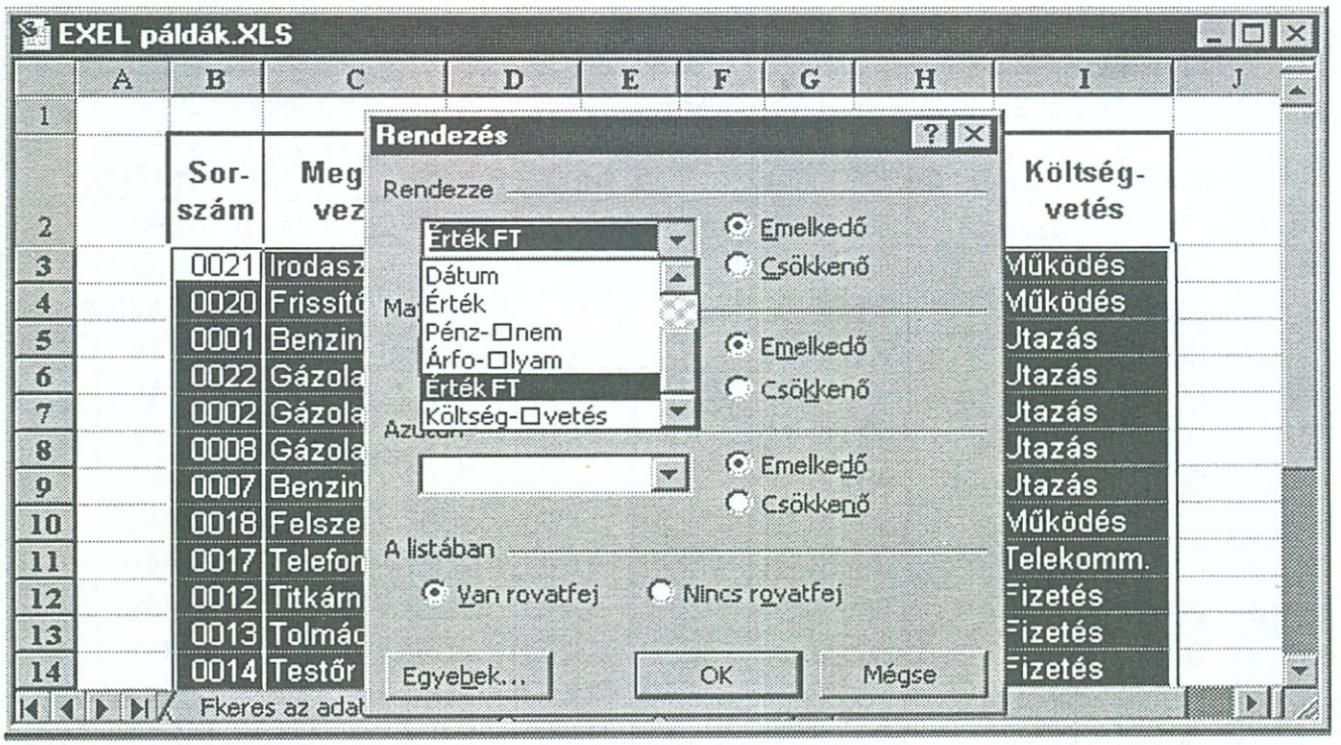
Vegyük észre, hogy a negyedik paraméter hiányzik a függvény argumentum listájából. Ha elhagyjuk az igaz, vagy hamis logikai paramétert, a rendszer a hiányzó paramétert "IGAZ"-nak tekinti.

	A	B	C	D
1			Keresendő rekord száma	
2		Sorszám	10	
3		Megnevezés	Irodabérlet	
4		Dátum	35934	
5		Érték	600	
6		Pénznem	USD	
7		Árfolyam	205	
8		Érték FT-ban	123000	
9		Költségvetés	Bérlet	
10				

3.2.2. Az adatbázis rendezése

Rendezésnek nevezzük azt a műveletet, amikor az adatbázisban lévő adatokat betűrendes, numerikus vagy időrendi sorrendbe rendezzük. Ez történhet növekvő vagy csökkenő sorrendben. **Rendezéskor a rekordok fizikai sorrendje megváltozik.** Az adatbázisban kijelölünk egy cellát, vagy az egész adatbázist, majd az **Adatok** menü **Sorba rendezés** parancsot kiválasztva a megjelenő párbeszédablakban beállíthatjuk a rendezési elvet maximum három oszlopot kiválasztva, mely-

ben külön-külön megadhatjuk, hogy emelkedő, vagy csökkenő sorrendbe rendezze a táblázatot. Amennyiben csak egy mező szerint akarunk rendezni, akkor a szokásos eszközsoron találunk e célra két ikont. Ügyeljünk arra, ha csak a táblázat egy részét jelöljük ki, akkor tönkre tehetjük az adatbázisunkat, mert a rendszer csak a kijelölt részt rendezi.



3.2.3. Leválogatás

Az automatikus szűréssel egyszerűen válogathatjuk ki a nekünk szükséges rekordokat. Ilyenkor a rendszer csak a beállított feltételeknek megfelelő rekordokat jeleníti meg magában a táblázatban. Mindezt az **Adatok** menü **Szűrő, Autószűrő** parancsával tehetjük meg. A rendszer a táblázat fejlécének minden egyes cellájában egy-egy legördítő nyilat helyez el. A nyilakkal legördített ablakban megjelennek az oszlopban található adatok, melyek közül egyet kijelölhetünk. A legördülő ablakban választhatjuk az **egyéni** beállítást, ekkor két reláció között dönthetünk, az és / vagy kapcsolat között. Ezek után csak azok a rekordok jelennek meg az adatbázisban, amely rekordokban szerepelnek a kiválasztott adatok és megfelelnek a beállított relációknak. Egyszerre sajnos csak egy autószűrést alkalmazhatunk egy időben a táblázatunkban.

A képen mutatott példánkban a pénznem oszlopa van szűrve, mégpedig a legördülő ablakban kiválasztott USD szerint. Ha egy oszlopban szűrés van beállítva, akkor a legördítő nyíl színe megváltozik, így a

táblázatra nézve rögtön megállapítható, hogy mely oszlopokban van valamilyen szűrés beállítva. Természetesen a táblázat nyomtatásakor a legördítő gombok nem nyomtatódnak ki. Az autószűrés kikapcsolása ugyanazzal a menüvel kapcsolható ki, amellyel bekapcsoltuk. (Adatok, szűrő, autószűrő).

EXEL példák.XLS

	A	B	C	D	E	F	G	H
1	Sor-szá	Megne-vezés	Dátum	Érték	Pénz-ner	Árfo-lya	Érték FT	Költség-vetés
2	0001	Benzin	98.05.01	(mind)		200	2240	Utazás
3	0002	Gázolaj	98.05.02	(Helyezés...)		200	2400	Utazás
7	0006	Szoftver	98.05.03	(Egyéni...)		205	266500	Fejlesztés
8	0007	Benzin	98.05.07	DEM		205	3075	Utazás
9	0008	Gázolaj	98.05.08	ENC		205	2767,5	Utazás
10	0009	Szálloda	98.05.19	USD		205	182450	Ellátás
11	0010	Irodabérlet	98.05.19	890	USD	205	182450	Bérlet
12	0011	Igazgató	98.06.01	600	USD	205	123000	Fizetés
13	0012	Titkárnő	98.06.01	520	USD	205	106600	Fizetés
14	0013	Tolmács	98.06.01	220	USD	205	45100	Fizetés
15	0014	Testőr 1	98.06.01	270	USD	205	55350	Fizetés
16	0015	Testőr 2	98.06.01	350	USD	205	71750	Fizetés
23	0022	Gázolaj	98.06.07	350	USD	205	71750	Fizetés
				11	USD	210	2310	Utazás

Fkeres az adatbázisban / Rendezés

3.2.3.1. Összetett leválogatás

Összetett leválogatási feltételeket is tudunk alkalmazni, így egyszerre több feltételt adhatunk meg, akár ugyanarra az oszlopra is. Létrehozunk egy úgynevezett feltétel blokkot, és abban helyezzük el a leválogatáshoz szükséges feltételeket. Az **Adatok** menü **Szűrő, Irányított szűrő** parancs kiválasztásakor a megjelenő párbeszéd ablakban megadjuk

- annak a táblázatnak, adatbázisnak a koordinátáit, amelyből a leválogatást el akarjuk végezni,
- a feltétel blokk koordinátáit, ahova a különböző feltételeket írtuk a munkalapra,
- annak a területnek a koordinátáit, ahova a leválogatást el akarjuk végeztetni a rendszerrel,

valamint bejelölhetjük, hogy a beállított feltételeknek megfelelő azonos rekordokból csak az első rekordokat jelenítse meg.

Ügyeljünk arra, hogy a leválogatási terület első sora ne essen egy sorba az adatbázis első sorával, mert a rendszer valószínűleg nem tudja kezelni az egy sorban lévő azonos mezőneveket, azonkívül a feltétel terület mindenképpen a leválogatási terület fölött legyen, mert a rendszer a leválogatáskor esetleg felülírja, és ez hibához vezet, mivel a megadott feltételek helyett más lesz a területen.

Pénznem	Érték FT
USD	>10000

Feltétel terület, <ÉS> kapcsolat.

Sorszám	Megnevezés	Dátum	Érték	Pénznem	Árfolyam	Érték FT	Költségvetés
0006	Szoftver	98.05.03	1300	USD	205	266500	Fejlesztés
0009	Szálloda	98.05.19	890	USD	205	182450	Ellátás
0010	Irodabérlet	98.05.19	600	USD	205	123000	Bérlet
0011	Igazgató	98.06.01	520	USD	205	106600	Fizetés
0012	Titkárnő	98.06.01	220	USD	205	45100	Fizetés
0013	Tolmács	98.06.01	270	USD	205	55350	Fizetés
0014	Testőr 1	98.06.01	350	USD	205	71750	Fizetés

3.2.3.2. A feltételblokk megadásának szabályai

A feltételblokknak legalább egy oszlopból és két sorból kell állni. Az első sor a mező neveket, számított feltételek esetén a feltétel neveket, az alatta lévő további sorok a kiválasztás feltételeit tartalmazzák.

Az összehasonlító **ÉS** feltételeket egymás melletti cellákban, ugyanabban a sorban adjuk meg a megfelelő mezőnevek alatt. A **VAGY** feltételeket lejjebb, további sorokban kell megadni. Tehát az egy sorban szereplő relációk ÉS kapcsolatot, a különböző sorokban szereplő relációk VAGY kapcsolatot létesítenek a rekordok megfelelő oszlopában található mezők adataival. Ha ugyanazon mezőre több feltételt akarunk beállítani, akkor a feltétel blokkban többször is használhatjuk az ugyanazon mezők neveit.

3.2.3.2.1. Számított feltételek alkalmazása

Számított feltételek esetén (ami nem más, mint valamilyen függvény, vagy függvényekkel végzett művelet) a feltétel névnek különbözni kell az adatbázisban található mezők neveitől, mert egyébként a rendszer, mint összehasonlító ÉS, VAGY feltételt értelmezi. A megadott képlet-

nek logikai értéket kell visszaadnia eredményül, ami IGAZ vagy HAMIS lehet. A feltételnek tartalmaznia kell legalább egy olyan hivatkozást, amelyik az adatbázis valamelyik oszlopára (mezőjére) hivatkozik. Ezt a hivatkozást relatív hivatkozás formájában kell megadni. A relatív hivatkozás helyett használhatjuk a megfelelő mező nevét is.

Azokat a feltételeket, amelyek több részfeltételből állnak (összehasonlító és-vagy számított), **összetett feltételeknek** nevezzük.

3.2.4. A leválogatás megszüntetése

Az **Adatok** menü **Szűrő / Minden látszik** parancsával tudjuk a leválogatást megszüntetni.

3.3. Műveletek leválogatott adatokkal

3.3.1. A leválogatott adatok nyomtatása

Amikor a **Fájl** menü **Nyomtatás** parancsát használjuk a rendszer csak a leválogatott rekordokat nyomtatja ki. Ha az autószűrő parancs legördítő nyilai is a táblázatban vannak, azok nem lesznek nyomtatva.

3.3.2. Szerkesztés és formázás

1. **Formátum** menü **Cellák** parancs, csak a látható cellákra vonatkozik, ennek következtében, ha megszüntetjük a szűrést, akkor elég furcsán néz majd ki a táblázatunk.
2. Szerkesztés menü parancsai csak a látható cellákra vonatkozik. A **Kivág, Másol, Beilleszt**.
3. A sorok törlése szintén csak a látható sorokat törli.

3.3.3. Rendezés

Táblázatunk rendezésekor, csak az éppen látható sorokat rendezi a rendszer, csak a leválogatott rekordok sorrendje változik meg.

3.3.4. Grafikon

Csak a leválogatott adatok jelennek meg a grafikonunkban.

3.4. Összegek és részösszegek képzése az adatbázisban

Az összegképzés három módját használhatjuk az adatbázisban.

- ♣ Automatikus összeg és részösszeg képzés.
- ♣ Összetett összegzések.
- ♣ Keresztreferencia tábla, amely egy interaktív összefoglaló táblázata az adatbázisunknak.

3.4.1. Automatikus összegzések

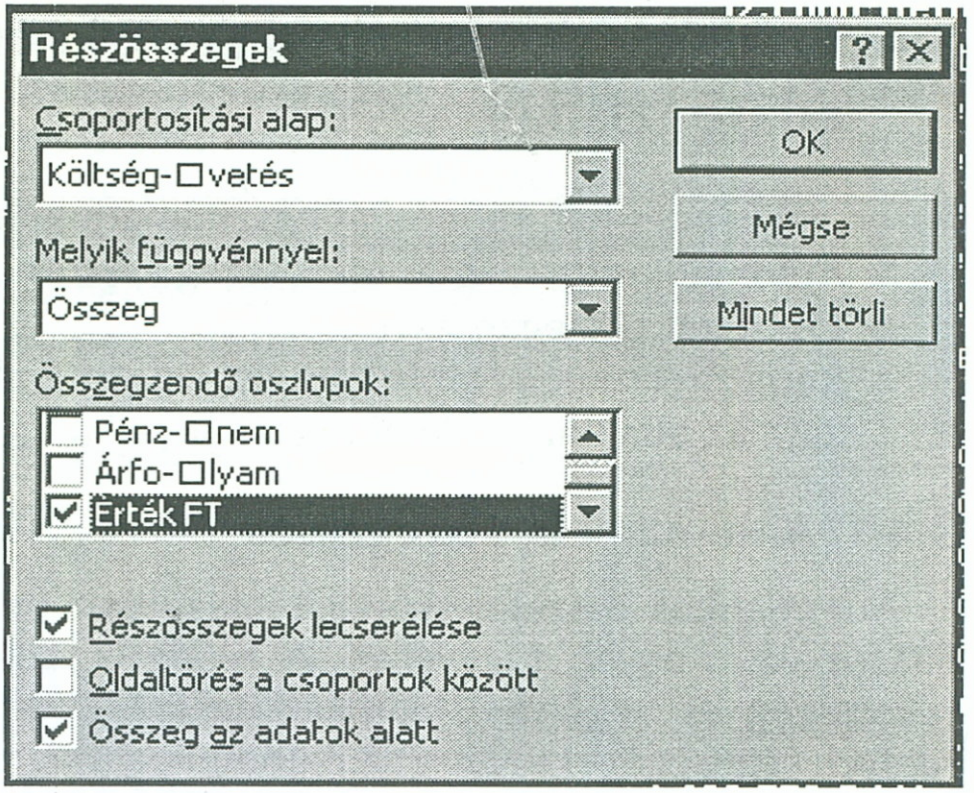
Automatikus összegzéskor a táblázatban nem kell elhelyeznünk a SZUM vagy egyéb függvényt. A rendszer automatikusan beszúrja a részösszeg és végösszeg sorokat, valamint elhelyezi a vázlatjeleket, melyek segítségével tovább szűkíthetjük a megjelenített adatok körét az összegekre. 11 függvényt használhatunk az összegzésekre. A végösszeget a rendszer az összes látható rekord, és nem a részösszegek felhasználásával végzi. Amennyiben módosítjuk a táblázat valamely rekordját, a rendszer automatikusan újraszámolja a végösszegeket.

	A	B	C	D	E	F	G	H
1	Sor-szám	Megne-vezés	Dátum	Érték	Pénz-nem	Árfo-lyam	Érték FT	Költség-vetés
3							**123 000**	**Bérlet Össz.**
5							**182 450**	**Ellátás Össz.**
6	0003	Fénymásoló	98.05.03	1300	DEM	100	130000	Fejlesztés
7	0004	Számítógép	98.05.03	2500	DEM	100	250000	Fejlesztés
8	0005	Számítógép	98.05.03	1700	DEM	100	170000	Fejlesztés
9	0006	Szoftver	98.05.03	1300	USD	205	266500	Fejlesztés
10	0016	Irodabútor	98.06.01	96500	ENC	0,8	77200	Fejlesztés
11							**893 700**	**Fejlesztés Össz.**
17							**350 550**	**Fizetés Össz.**
18	0017	Telefon	98.06.02	25000	ENC	0,8	20000	Működés
19	0018	Felszerelés	98.06.03	14000	ENC	0,8	11200	Működés
20	0019	Irodaszerek	98.06.04	800	DEM	105	84000	Működés
21	0020	Frissítő	98.06.05	2500	ENC	0,8	2000	Működés
22	0021	Irodaszerek	98.06.06	1800	ENC	0,6	1080	Működés
23							**118 280**	**Működés Össz.**
29							**12 792,5**	**Utazás Össz.**
30							**1 680 772,5**	**Végösszeg**

Példánkban először rendezni kellett az adatbázist a költségvetés oszlop szerint, majd az Adatok menü részösszegek parancsára bejövő párbeszéd panelon a következőket kell beállítani.

Összegzés előtt rendeznünk kell az adatbázist az összegzendő mezőknek megfelelően, majd az **Adatok** menü **Részösszegek** parancs ki-

választásakor megjelenő párbeszédablakban kell beállítani az összegzendő mezőket. Készíthetünk egymásba ágyazott összegzéseket is. A párbeszédpanelon több összegző függvény is kiválaszthatunk. Ilyenkor és az egymásba ágyazott összegek képzésekor ne felejtsük el kikapcsolni a **Részösszegek lecserélése** kapcsolót a panelon. Ebben az esetben az Excel megtartja a már meglévő összegzéseket is.



3.4.2. Az összegzések megszüntetése

- Az előző művelettel létrehozott részösszeget a Szerkesztés menü Visszavonás parancsával szüntethetjük meg.
- A részösszeg párbeszédpanelon bekapcsolhatjuk a Részösszegek lecserélése kapcsolót.
- A párbeszédpanelon a mindent töröl gombra kattintva is megszüntethető.

3.4.3. A vázlat használata

Részösszeg képzésekor a munkalapon a táblázat előtt megjelennek + és – jellel ellátott gombok, amellyel szűkíthető, vagy a szűkített táblázatban bővíthető, a megjelenített adatok köre. Ez különösen előnyös grafikonok készítésénél. Ekkor a grafikonon csak a látható cellák adatai jelennek meg. Persze csak akkor, ha ki is volt jelölve a diagramvarázsló indítása előtt.

3.4.4. Összetett összegek

Komplex összegzéseknél a képletet saját magunknak kell megalkotni az adatbázis függvényeket, vagy más, egyéb függvényt (pl. a SZUMHA)

felhasználva. Előnyt jelent, hogy ilyenkor a munkalap bármely részére elhelyezhetjük a képleteinket.

SZUMHA(blokk,feltétel,összegző_blokk)

- A „blokk" az a tartomány, amelyben a feltételt vizsgálni kell.
- A „feltétel" a vizsgálandó feltétel.
- Az „összegző_blokk" azok a cellák, amelyeket összegezni kell.

Használhatók még a darab és az összes ÁB függvény az összetett összegek képzésére.

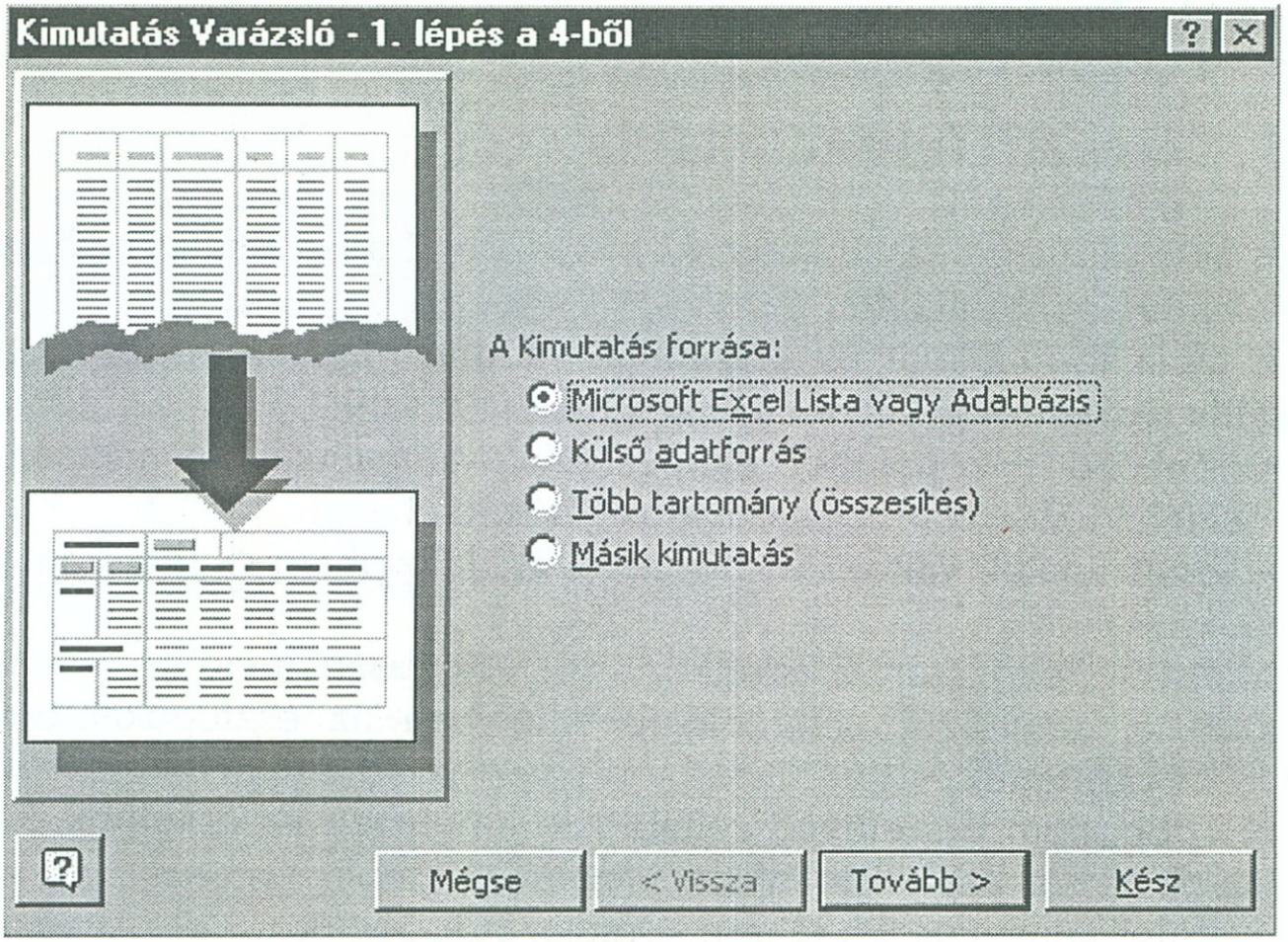
3.5. Keresztreferencia tábla

1. A keresztreferencia táblát nagy mennyiségű adat áttekinthető megjelenítésére használjuk.
2. A tábla formátumát és a számítás módját magunk adjuk meg. A tábla részletességét is nekünk kell meghatározni.
3. Különböző nézőpontból elemezhetjük adatainkat. A változás a megfelelő mezők mozgatásával végezhető el.
4. A forrás adatok változásával a tábla aktualizálható és újraszámítható.
5. A tábla létrehozására a következő források használhatók.
   a) Excel adatbázis, cella blokk, amelyben az oszlopokat névvel azonosítottuk.
   b) Több, nem összefüggő cellablokk, amelyben az oszlopok és sorok névvel azonosítottak.
   c) Külső adatbázis. Pl. Dbase, Oracle, SQL Server.
   d) Egy már létező keresztreferencia tábla.
6. A tábla létrehozásakor a rendszer, különböző összegző függvényeket használ. Ilyen, pl. a SZUM vagy az ÁTLAG.
7. E tábla segítségével könnyen hozhatunk létre különböző részletezésű grafikonokat, adataink változásakor a grafikonok azonnal jelzik azokat.

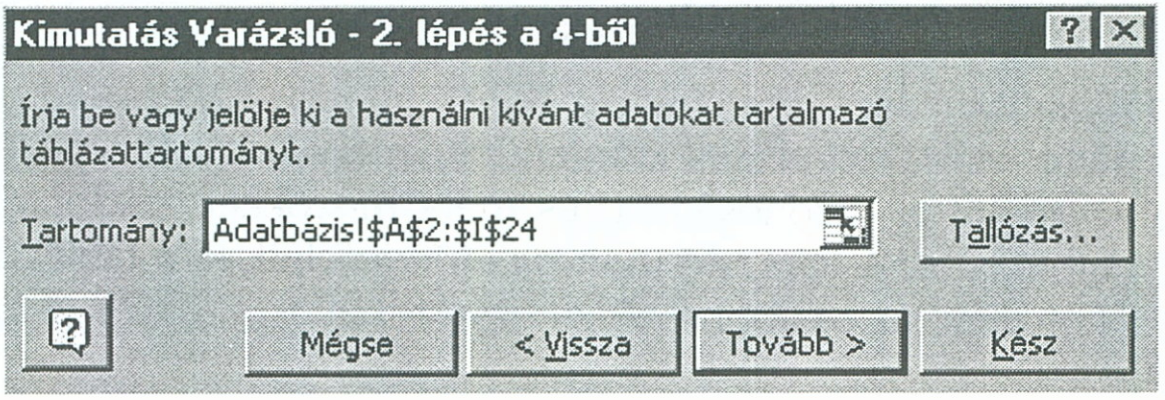
A tábla létrehozásakor el kell döntenünk, mely adatbázismezőket akarjuk felhasználni az azonosításra, és mely mezőkkel, oszlopokkal töltjük fel a táblázatot. A varázsló indítása előtt jelöljük ki a táblázatot, mégpedig a fejléccel, az oszlopnevekkel együtt

3.6. A keresztreferencia tábla létrehozása

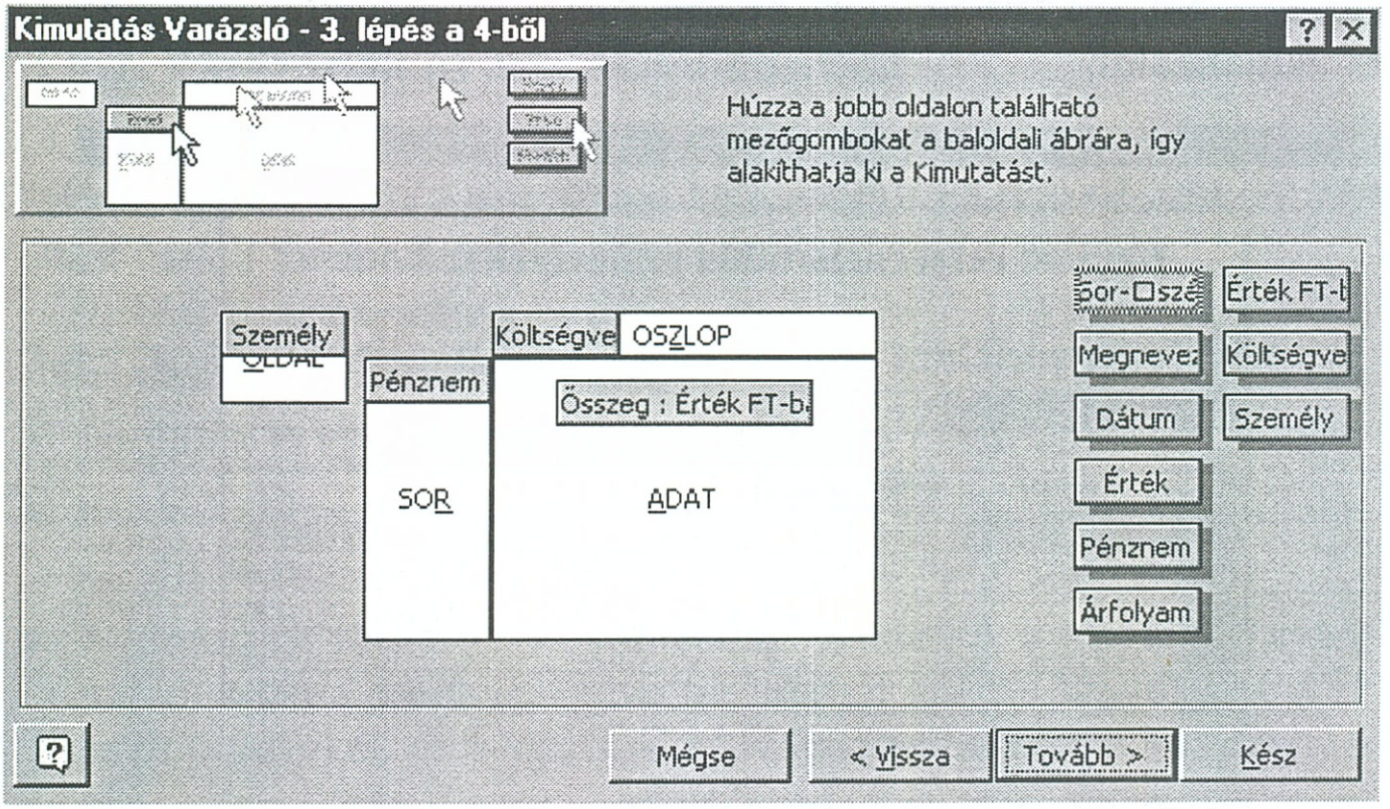
A táblát az **Adatok** menü **Kimutatás** paranccsal indítható, négylépéses varázslóval készíthetjük el. Az első lépésben a forrás adatok típusát adhatjuk meg. Ez lehet bármilyen Excel lista, vagy táblázat, tartomány, vagy tartományok, egy bármilyen külső adatbázis, vagy akár egy másik ugyanilyen kimutatás. A megfelelő forrást kiválasztva léphetünk a varázsló 2. Lépésére.



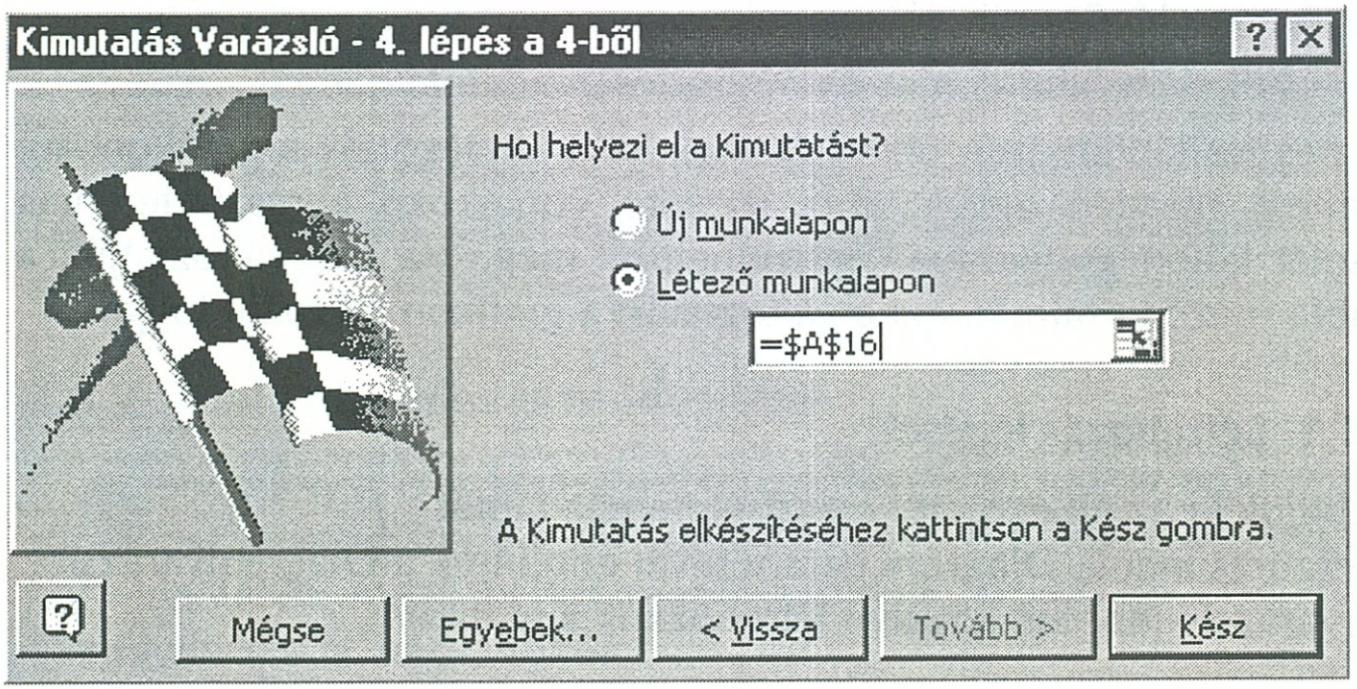
A második lépésben az adatok pontos elhelyezkedését, cellablokk hivatkozást adjuk meg. Ebben benne kell lenni az oszlop neveknek is. Tehát az adatbázis fejlécének is.



A harmadik lépésben meghatározzuk, mely mezőket használunk sor és mely mezőket oszlop kategóriaként. Mivel a táblázatot mező nevekkel együtt jelöltük ki, ezért a mezőnevek automatikusan megjelennek. A panel jobb oldalán található nevek közül a kiválasztott mezőnevet egyszerű vontatással a megfelelő kategória fölé visszük, és ott elengedjük.



A negyedik lépésben a keresztreferencia tábla helyét adjuk meg, ami azt jelenti, hogy egy másik munkalapon készítse el, vagy arra, amelyiken éppen állunk.



3.7. Grafikon létrehozása keresztreferencia táblával

A rendszer a tábla létrehozásakor automatikusan összeget képez, ha az adatok számok. Darabszámot határoz meg, ha az adatok karakteresek. A tábla elkészültekor egy új ikonsor jelenik meg, amelyekkel különböző parancsokat adhatunk ki. Amennyiben az adatbázisban változás történt, akkor a táblánkat az **Adatok** menü **Adatfrissítés** parancsával lehet aktualizálni.

Kereszt referencia tábla személyekre bontva

Személy	Kun			
Összeg : Érték FT-ban	Költségvetés			
Pénznem	Berendezés	Fizetés	Működés	Végösszeg
DEM	175000	0	0	175000
ENC	0	0	11200	11200
USD	0	55120	0	55120
Végösszeg	175000	55120	11200	241320

Grafikon létrehozásakor a következőket kell betartani:

a) A forrás tábla ne tartalmazzon kettőnél több sor, vagy oszlop kategóriát. Nem sort és oszlopot, hanem a Kimutatás Varázsló harmadik lépésében lévő sor és oszlop kategóriáról van szó

b) Töröljük a táblából a rész és végösszesen adatokat.

c) Az egyes táblamezők nem lehetnek külön kategóriák a grafikonban, hanem minden egyes tábla mezőre vonatkozó keresztreferencia táblát külön grafikonban jeleníthetünk meg. Ha megváltoztatjuk a tábla mező tartalmát, ezt a változtatást a grafikon tükrözni fogja.

3.7.1.1. Létrehozás lépései

1. Jelöljük ki az elkészített keresztreferencia táblát.
2. Beszúrás menü, **Diagram** parancsával elindítjuk a Diagramvarázslót.
3. A varázsló lépéseit követve létrehozzuk a grafikont.

3.8. Az alap keresztreferencia tábla módosítása

Az alap keresztreferencia tábla létrehozása csak a kezdet a használat terén. A következő lépésekben leírt módosításokkal a legkülönfélébb kimutatást tudjuk elkészíteni, kihasználva az Excel adta lehetőségeket.

3.8.1. Kategóriák és adatok módosítása

Mivel a keresztreferencia tábla aktív kapcsolatban van a forrás adatokkal, ezért a tábla adatterülete közvetlenül nem módosítható! Módosítást csak a forrásadatokon végezhetünk.

3.8.1.1. Sor, oszlop, adat és tábla mezők táblához rendelése

- Jelöljünk ki egy cellát a keresztreferencia táblában.
- Klikkeljünk a Lekérdezés és kimutatás eszközsoron a Kimutatás varázsló ikonon, vagy az Adatok menü Kimutatás parancson.
- A kimutatás varázslóban vontatással rendeljük a megfelelő kategória mezőt a tábla megfelelő részéhez. Tehát egérrel fogjuk meg a kívánt oszlop neveket, és vontassuk a kiválasztott területre, majd engedjük el.

3.8.1.2. Kategóriák és tábla mezők törlése a táblából

- Jelöljük ki a megfelelő kategóriát, vagy tábla mezőt.
- Vontassuk a kijelölt mezőt a **keresztreferencia táblán kívülre.**
- Amikor az egér kurzora megváltozik „**X**"-re, engedjük el.

Tehát a nem kívánt nevet egyszerűen megfogjuk és kidobjuk a táblából.

3.8.2. Tábla elrendezés módosítása

Az elrendezést is a keresztreferencia táblában tudjuk módosítani, a sor és oszlopkategória mezők vontatással történő áthelyezésével. Amikor valamelyik mező helyét megváltoztatjuk, az összes hozzátartozó elem helyzete is megváltozik.

3.8.3. A tábla mező használata

A táblamező egyfajta lekérdezést tesz lehetővé. Segítségével egyszerre egyetlen mezőre vonatkozó adatokat jeleníthetünk meg a keresztreferencia táblában. A kimutatás varázsló segítségével tudunk táblamezőt használni. Egyszerűen **a kiválasztott oszlop nevét vontatással elhelyezzük az OLDAL pozícióba**. Ezáltal az elhelyezett oszlop laponkénti bontásban jelenik meg, miután bezártuk a kimutatás varázslót. A keresztreferencia táblában egyszerre egynél több táblamező is elhelyezhető.

3.8.4. Összegzések a táblában

A Kimutatás Varázsló negyedik lépésének párbeszédpanelján találunk egy „egyebek" nevű gombot, amelyre kattintva különböző összegzésre és formázásra vonatkozó parancsot kapcsolhatunk be.

3.8.4.1. Végösszeg

Ha az ADTOK mezőbe számokat tartalmazó oszlopnevet helyezünk, a tábla létrehozásakor a rendszer automatikusan elhelyezi a keresztreferencia táblában a végösszeg sorokat és oszlopokat. Az összegeket a rendszer mindig a forrás adatok (adatbázis) felhasználásával határozza meg. Amennyiben a táblában egynél több adat mező van, a rendszer mindegyik adat mezőre létrehozza a végösszeg sorokat és oszlopokat. Tudni kell, hogy nem lehet a végösszeget létrehozni egy kiválasztott oszlopra, csak az összesre. Szövegmezők elhelyezésekor az összegzéskor a szövegsorok számát kapjuk meg.

3.8.4.2. Összes összeg

Amennyiben a tábla egynél több sor, vagy oszlopkategóriát tartalmaz, a rendszer automatikusan létrehozza a részösszesen sorokat, persze csak akkor, ha az „egyebek" panelon ezt engedélyeztük.

3.8.5. A tábla formázása a számítás módjának megváltoztatása

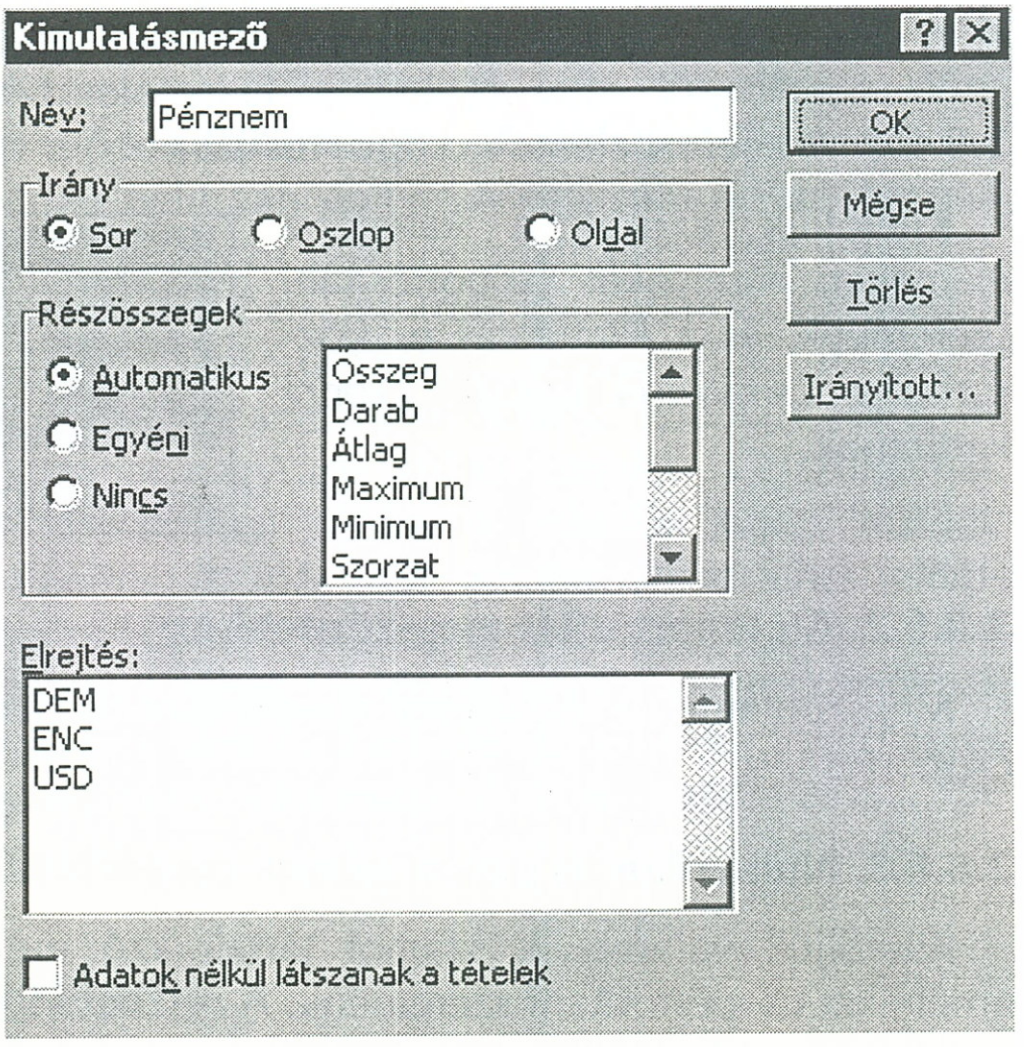
3.8.5.1. Autóformázás

Ha a Kimutatás varázsló 4. lépésében az „egyebek" panelon „**a tábla autóformázása**" kapcsolót bekapcsoljuk, akkor a rendszer automatikusan megformázza táblánkat. Amennyiben ezt nem tettük meg,

akkor a **Formátum** menü **Autóformázás** parancsával formázhatjuk meg automatikusan táblázatunkat.

3.8.5.2. Adat mezőkre vonatkozó formázás

Miután a táblában kijelöltünk egy mezőt, az Adatok menü **Összesítés** parancsával módosíthatjuk táblánkat. A megjelenő párbeszédablakban különböző összegzéseket rendelhetünk hozzá (darab, átlag, minimum, maximum, stb.). Alapértelmezésben a rendszer a számokat tartalmazó mezőket összegzi, a szöveges mezőket megszámolja hány darab. Tudunk a panelon egyéb beállításokat is végezni. Ha a kategória neveken duplán kattintunk, akkor a megjelenő Kimutatásmező panelon különféle beállításokat végezhetünk. Megváltoztathatjuk a kategória helyét, a hozzárendelt összegzéseket, az összegképzés módját, mely oszlopok jelenjenek meg, melyek legyenek rejtve a táblázatban. Az irányított gombra kattintva, pedig még különböző rendezéseket is beállíthatunk.



3.8.5.3. Részletek megjelenítése, eltüntetése

A Kimutatás eszközsoron a megfelelő ikonnal lehet az adatok részletezését bekapcsolni, vagy kikapcsolni. Ugyanezt a hatást érjük el, ha az adat mezők nevén kettőt kattintunk.

3.8.5.4. Adat cella forrás adatainak megjelenítése

Ha kettőt klikkelünk a keresztreferencia tábla adatokat tartalmazó egy celláján, a rendszer a cellához tartozó forrásadatokat egy új táblába átmásolja.

3.8.6. Csoportok képzése

Három féle képen képezhetünk csoportokat:

a) A kijelölt kategóriákat rendezzük csoportokba.
b) A numerikus adatokat a határok megadásával rendezzük csoportokba automatikusan.
c) A dátum és idő adatainkat rendezzük automatikusan csoportokba.

3.8.6.1. A kijelölt elemek csoportosítása

Kijelöljük azokat a kategória elemeket, amelyeket csoportosítani szeretnénk, és a Kimutatás eszközsoron a **Csoportba foglalás** ikonra kattintunk.

3.8.6.2. A csoportosítás megszüntetése

Megszüntetni az eszközsor **Csoportbontás** ikonjával lehetséges a csoportosítást.

3.8.6.3. Numerikus és dátum adatok csoportosítása

Kijelöljük valamelyik számot tartalmazó mezőjét a táblázatnak, amelyiket csoportosítani szeretnénk. A részletcsoport ikonra kattintva a megjelenő párbeszédpanelon beállítjuk az alsó és felső határát a csoportosításnak, valamint a lépésközt.

A dátum értékeket hasonló képen rendezhetjük csoportokba. Ha a kijelölt cella dátumot tartalmaz, akkor a rendszer egy másik párbeszédpanelt jelenít meg, ami a dátumok csoportosítását teszi lehetővé.

4. Adatkezelés MS Access adatbázis kezelővel

A Microsoft Access relációs adatbáziskezelő rendszer, az egyik legnagyobb szoftverje a Microsoftnak. Korszerű objektumorientált, és a legkülönbözőbb rendszerelemeket integrálták bele. Kezelése a Windows-ban megszokott módon történik. Az alkalmazásfejlesztéstől, az SQL lekérdezésig minden bele van építve. Mégis mire jó az Access? Egyszerűen fogalmazva, egy olyan relációs adatbáziskezelő rendszer, melynek segítségével a különböző információk, adatok könnyedén tárolhatók, visszakereshetők, lekérdezhetők, rendezhetők. A kisebb cégek remekül tudják használni adataik tárolására, feldolgozására, de SQL szerverre kötve közepes nagyságú vállalkozásokat is kiszolgálhat. Az Access különböző forrásokból származó adatokat képes kezelni, pl.: dBase, Paradox, FoxPro, és más SQL adatbázisokat lehet feldolgozni általa. Képes az előzőkben leírt adatbázis-kezelők adatait importálni, és képes saját adatbázisát Ms Word, Ms Excel programoknak átadni. Az OLE (Objektum Linking Embedded) objektumcsatolás és beágyazás lehetőségével más alkalmazásokat is beépíthetünk, pl. Access űrlapba, lekérdezésbe.

4.1. A normalizálásról

Könyvünk előző részében leírt fogalmakat ki kell bővíteni az úgynevezett normalizálással. Az 1-5-ig terjedő számozással ellátott normál formájú táblázatok egyre „rendesebben viselkednek". Az 5NF alakja az adatbázisnak a legrendesebb alakja, de általában megelégszünk már a 3NF-jú alakkal.

4.1.1. 1NF meghatározása

Egy táblázatot akkor mondunk első normál formájúnak, ha mezőiben (oszlopaiban) csak egyszerű tulajdonságok szerepelnek. Ebben a

formában nincs két megegyező sor és van legalább egy vagy több olyan tulajdonág, vagy tulajdonság csoport, amelyekkel a sorok egyértelműen megkülönböztethetők egymástól. Amennyiben ezeket a kulcsjelölteket a táblázat sorainak tényleges megkülönböztetésére használjuk, akkor ezeket a táblázat elsődleges kulcsának hívjuk. Ha a kulcs egy tulajdonságból áll, egyszerű kulcsnak hívjuk.

4.1.2. 2NF meghatározása

Egy táblázat akkor második normál formájú, ha 1NF-ben van, és minden mezője, amelyik nem tartozik az elsődleges kulcshoz, funkcionálisan teljesen függ az elsődleges kulcstól. Ha van olyan mező, amelyik az elsődleges kulcsnak csak egy részétől függ, akkor azt a tulajdonságot másik táblázatba kell helyezni.

4.1.3. 3NF meghatározása

Egy táblázat akkor van harmadik normál formájú alakban, ha 2NF-ben van, és minden függőség kulcsjelölttől származik, nem függ idegen kulcstól is. Idegen kulcsnak nevezzük a táblázatban az olyan mezőket, amelyek egy másik táblázatban kulcsjelölt, vagy éppen elsődleges kulcs. Ezekkel tarthatjuk a kapcsolatot más táblázatokkal.

4.2. Kapcsolatok

Több táblás adatbázisban a kulcsokkal az egyes táblák össze vannak kapcsolva egymással. A kapcsolatoknak három változata van.

- Egy az egyhez 1:1.
- Egy a többhöz 1:M.
- Több a többhöz N:M

1:1 a kapcsolat, ha A tábla egy rekordja B tábla egy rekordjával kapcsolódik és fordítva.

1:M a kapcsolat, ha A tábla egy rekordja B tábla több rekordjához kapcsolódik, de B tábla egy rekordja A táblának csak egy rekordjával kapcsolódhat.

N:M kapcsolat akkor, ha A tábla egy rekordja B tábla több rekordjával kapcsolódhat, és B tábla egy rekordja A tábla több rekordjához kapcsolódhat. Ezeket a táblákat nem lehet egyszerűen összekapcsolni, csak egy olyan közbeiktatott táblával, amelyik egyikkel és másikkal is egy a több kapcsolatban áll.

Az Access programot indítani az ikonjára duplán kattintva tudjuk, vagy a start menü programok parancsából kiválasztjuk a Microsoft Access elemet.

Kilépni a Fájl menü Kilépés parancsával, vagy az ablak vezérlőmenüjére, a gyufára duplán kattintunk.

4.3. Adatbázis ablak

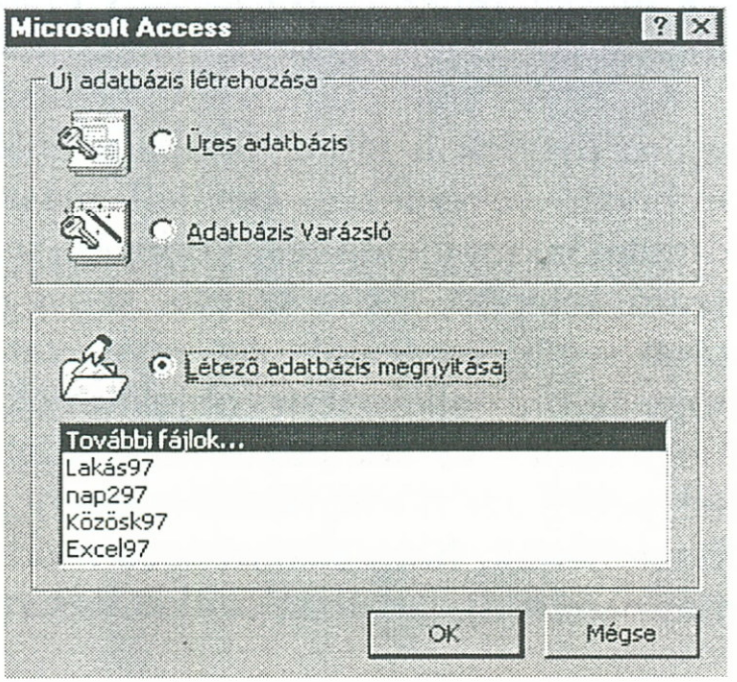
Az Access adatbázisa fájlban

- táblát,
- lekérdezést,
- űrlapot,
- jelentést,
- makrót,
- modult

tartalmazhat. Ezek az adatbázis főbb objektumai.

4.4. Adatbázis létrehozása, megnyitása

Kétféle módon hozhatunk létre új adatbázist, amit az indítás után megjelenő párbeszédpanel határoz meg. Egy üres adatbázist hozunk létre a megfelelő rádiógombot kiválasztva, vagy ugyanitt elindítjuk az Adatbázis varázslót. A létrejövő adatbázis fájl kiterjesztése mdb, de egy ldb kiterjesztésű állományt is létrehoz a rendszer. Egy meglévő adatbázist a Fájl menü Adatbázis parancsával nyithatunk meg, de a párbeszédpanelon is kiválaszthatjuk ezt a funkciót.



Amikor egy már létező adatbázist nyitunk meg, az Access adatbázis ablaka jelenik meg monitorunkon. Ennek címsora alatt találhatók a különbözőobjektumok fülei. Valamelyik fülecskére kattintva a kiválasztott objektumtípusok között válogathatunk, melyet tervező vagy adatnézetben nyithatunk meg az adatbázis ablak jobb oldalán található gombokkal.

4.5. Az Access moduljai

4.5.1. Táblák

Az adatainkat tartalmazzák valamilyen rendezett formában. Sorokból (rekordokból) és oszlopokból (mezőkből) állnak.

4.5.2. Lekérdezések

A táblák adatainak összegyűjtésére, megtekintésére, módosítására, törlésére hivatottak. Jelentések, űrlapok forrása is lehet egyben.

4.5.3. Űrlapok

Adataink bevitelére szolgál, de a már létező adatokat is módosíthatjuk velük.

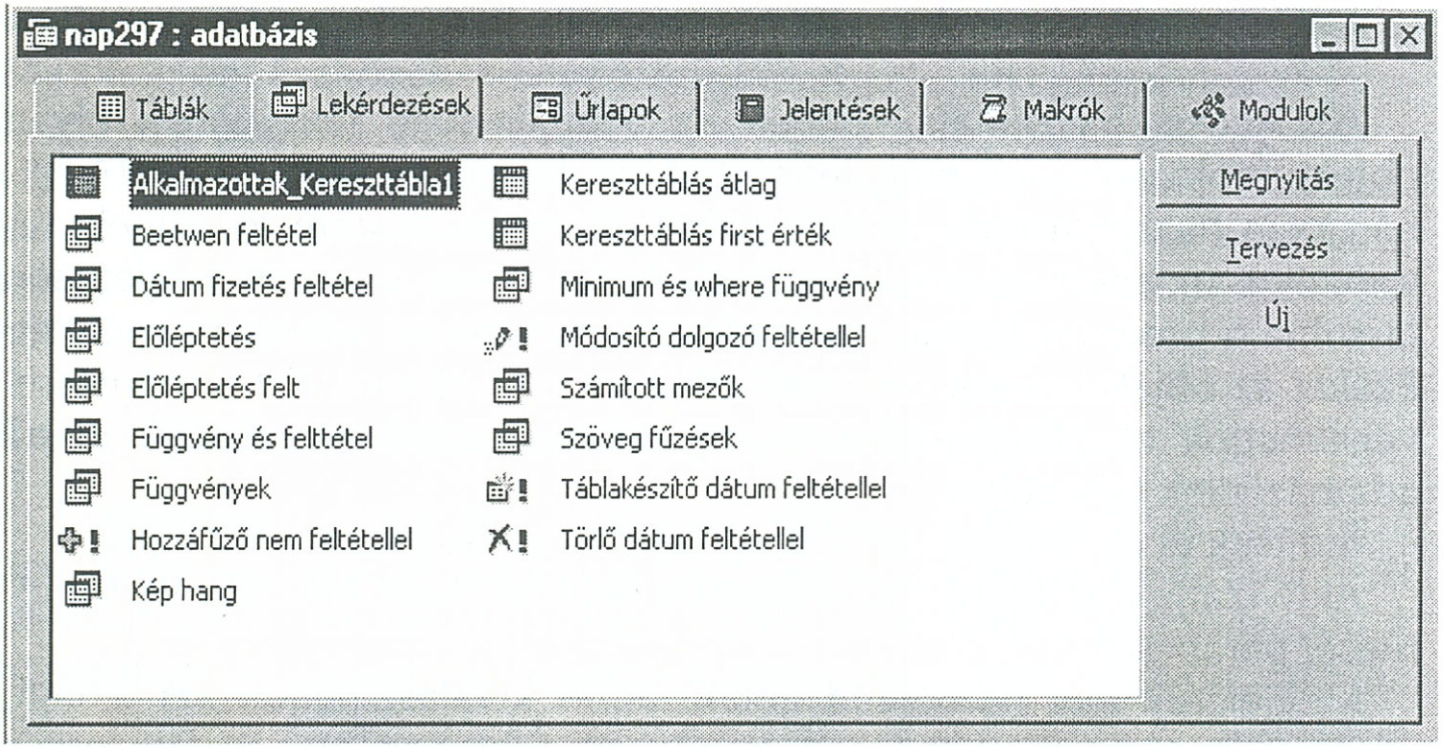
4.5.4. Jelentések

A különböző lekérdezések által kigyűjtött adatainkat tudjuk kinyomtatni.

4.5.5. Makrók

A gyakori ismétlődő feladatokból makrókat készíthetünk. A makrók és az eseményvezérelt eljárások segítik az objektumaink összekapcsolását, automatizálhatunk bizonyos feladatokat vele. Az egyes események, vagy saját függvények Visual Basic kódjait tartalmazza. A makrók futását valamilyen esemény indítja el, pl.: az egér valamilyen kattintása, gomb felengedése, lenyomása, stb.

Az egyes objektumok létrehozását a rendszer rengeteg varázslóval segíti. Ilyenek a tábla, űrlap, jelentés, adatbázis, adatbázis daraboló, archiváló lekérdezés, beviteli maszk, adatvédelmi, címke, diagram, kimutatás, megjelenítés, körlevél, segédűrlap, parancsgomb, táblaanalizáló, telepítő, testre szabás, szövegimportáló és vezérlőelem létrehozását segítő varázslók. Az egyes varázslók lehetővé teszik számunkra, hogy előre elkészített objektumokból válogathassunk igényünknek megfelelőt. Az adatbázist használathoz előbb tehát meg kell nyitni, mégpedig azt is meg kell határozni, hogy milyen módon nyitjuk meg. Megtehetjük, hogy kizárólagos módon nyitjuk meg, ebben az esetben más nem tudja adatbázisunkat használni, míg be nem zárjuk azt.



Létezik az Access-ben egy speciális makró, melynek neve AutoExec. Ez a makró automatikusan elindul, mint a DOS operációs rendszerben az autoexec.bat. Az adatbázist elindítva a rendszer képes elindítani elkészült alkalmazásunkat ennek a makrónak a segítségével. Ha tovább akarjuk fejleszteni alkalmazásunkat, a shift billentyű lenyomásával megakadályozhatjuk az automatikus indulást, amikor az OK gombra kattintunk.

Parancssorból is megnyithatjuk adatbázisunkat megfelelően paraméterezve az access parancsot, ilyenkor az első paraméter az adatbázis neve, majd a megnyitást módosító kapcsolók következnek a parancssorban.

- /ro írásvédettséghez
- /excl kizárólagos használatra megnyitás
- /x <a futtatandó makró neve> a kapcsoló segítségével bármelyik makrónkat lefuttathatjuk.

4.6. Adatbázisaink karbantartása

4.6.1. Konvertálás

A régebbi verzióban készült adatbázisunkat célszerű az újabb Access-hez konvertálni a jelenleg használatban lévőhöz. Tekintettel arra, hogy az újabb verzió számú szoftverek mindig többet tudnak, így használhatjuk az újabb funkcióit is rendszerünkkel a régi adatbázisban.

4.6.2. Tömörítés

Az Access használata közben az adatbázisban létrejönnek olyan objektumok, amelyek csak létrejöttükkor használatosak, a továbbiakban nem használatosak. Ezek a halott objektumok nem törlődnek automatikusan, mert ez sok lemezművelettel járna, s emiatt lényegesen lassulna a program futása, így ezek csak növelik az adatbázis méretét. Ezeket az objektumokat célszerű időnként eltávolítani. Ezt tömörítéssel végezhetjük el. Tehát időnként az alábbiak szerint végezzük el az adatbázisunk takarítását.

Indítsuk el az Access-t úgy, hogy nem nyitunk hozzá adatbázist.

Aktivizáljuk az Eszközök/Adatbázis segédeszközök/Adatbázis tömörítése/ parancsot.

Megjelenik egy párbeszéd ablak, amin kiválaszthatjuk azt a fájlt, amelyet tömöríteni akarunk. A tömörített adatbázis egy új fájl lesz a lemezen, ezért a nevét majd meg kell adni az új fájlnak. A tömörítést parancssorból is elvégezhetjük, pl. C:\Access\msaccess.exe c:\acces\adatok\lakas.mdb /compact. Ebben a példában az Access elindul és tömöríti a lakas.mdb nevű adatbázist, majd elmenti ugyanazon a néven az eredeti helyére. A tömörítés befejezésekor visszatér a parancssorhoz.

4.6.3. Az adatbázis helyreállítása

Bármennyire is gondosan bánunk lemezeinkkel, mégis előfordulhat, hogy az adatállományunk megsérül. Ezt okozhatja bármi, pl. a hálózati áram kimaradása, lemezen lévő szektor hiba, stb. Elindítva az Access-t lehetőségünk van a hibás fájl esetleges kijavítására. Természetesen a javítás nem jár mindig teljes sikerrel, de azért többé-kevésbé sikerülhet. Mindezt az Eszközök/Adatbázis segádeszközök/Adatbázis helyreállítása menü segítségével végezhetjük el. Ez a menüparancs részlegesen, vagy szerencsés esetben teljesen helyreállítja a sérült adatbázisunkat, de teljes biztonságot csak a rendszeres másolat készítésével érhetjük el.

4.6.4. Titkosítás

Egy egyszerű DBF adatállományt, mint például a dBase adatbázis állománya egy szövegszerkesztővel simán olvasható. Ez nem biztos, hogy nekünk jó. Lehetőségünk van az Access adatállományok titkosítására úgy, hogy lemezre rejtjelezve írjuk fel. Természetesen tökéletes titkosítás nem létezik, de ha sikerül megnehezíteni a visszafejtést annyira, hogy a visszafejtéshez befektetett energia több mint a kapott

eredmény, akkor elmondhatjuk, hogy sikeresen rejtettük el adatainkat. Mindezt az Eszközök/Adatvédelem/Adatbázis titkosítása/visszafejtése menü segítségével végezhetjük el, amennyiben jogosultak vagyunk a táblák terveinek módosítására. Sajnos a titkosítással némileg lelassítjuk az adatállományunk használatát. Hozzá kell tenni, hogy ez a módszer csak kiegészíti az egyéb titkosítási módszereket, ezzel csak az esetleges szövegszerkesztővel történő olvasást nehezítjük meg.

4.6.5. Adatbázisok többszörözése, szinkronizálása

Elképzelhető, hogy egy adatbázist többen is használnak egyszerre úgy, hogy ugyanaz az adatbázis több gépen van elhelyezve. Ilyen eset például akkor fordul elő, mikor valaki hordozható számítógépet is használ munkájához. Ez egyszerű másolással nem nagyon oldható meg. Ilyenkor az adatbázist többszörözhetővé kell tenni. A munkavégzés során az egyes adatbázisokban történt változtatásokat időnként szinkronizálni kell. Az Eszközök/Kópia/Kópia készítése menüvel elvégezhetjük az adatbázis másolását. A mennyiben az adatbázis még nem volt többszörözhető, akkor az eredeti példányt automatikusan főkópiává alakítja / ebben vannak eltárolva a legfontosabb rendszerinformációk /. Csak a főkópiában módosítható az adatbázis eredeti terve. Többszörözéskor kérhetünk tartalék másolatot is az eredetiről Az időnkénti szinkronizálást az Eszközök/Kópia/Szinkronizálás menü parancsával végezhetjük el, ekkor a kiválasztott adatbázist főkópiává is lehet alakítani.

4.6.6. Objektumok

Az adatbázis egyes kategóriáiban / lekérdezések, űrlapok, jelentések, stb. / létrehozhatunk új objektumokat. Kiválasztjuk a kívánt kategóriát, majd az új gombra kattintva, vagy az Eszköztár új lekérdezés, új űrlap, stb. ikonjára kattintva létrejön az új objektum. A már meg lévő objektumainkat átnevezhetjük az egér jobb gombjával kattintva, a megjelenő helyi menüben található Átnevezés parancs használatával. Felesleges objektumainkat törölni a Del billentyűvel, vagy a Szerkesztés/Törlés paranccsal tudjuk. Létező objektumainkat megnyitni az adatbázis ablakban található Megnyitás, Tervezés gombokkal tudjuk. Ezek más-más nézetben jelenítik meg objektumainkat.

4.6.7. Nyomtatás

Az Access alapértelmezése szerint az adatokat papíron megjeleníteni, nyomtatni elsősorban a jelentésekkel lehet. Azért van lehetőség bármely objektum nyomtatására, a Fájl/Nyomtatás menüvel, vagy az eszközsoron található nyomtatóikon segítségével. Nyomtatás előtt természetesen a nyomtatandó objektumot megtekinthetjük a nyomtatási képen és ellenőrizhetjük adatainkat nyomtatás előtt. Mindezt a Fájl/Nyomtatási kép menüparanccsal tehetjük meg. Lehetőségünk van az objektumok terveinek kinyomtatására, megkönnyítve alkalmazásunk dokumentációjának elkészítését.

4.6.8. Az adatbázis ablakának elrejtése

Az Ablak/Elrejtés, Ablak/Felfedés menükkel lehetőséget kaptunk arra, hogy az adatbázis ablakunkat elrejtsük, így teljesen egyéni megjelenítésű alkalmazásokat is készíthetünk. Ettől még az egyes objektumok az alkalmazásunk számára elérhetők maradnak

4.6.9. Az Access testre szabása

A többi windows programokhoz hasonlóan az Access is beállítható igényeinknek megfelelően. Ezt a Nézet/Eszköztárak menüben található eszközök segítségével tehetjük meg. A kiválasztott eszközt bekapcsolhatjuk, kikapcsolhatjuk, és a Testreszabás gomb segítségével a kívánt változtatásokat megejthetjük. Az Eszközök/Beállítások menü kiválasztása után megjelenő párbeszédpanelon az egész Access viselkedését be tudjuk állítani. Ez egy több fülecskével / több oldallal / rendelkező panel, szinte minden különleges igény beállítható itt.

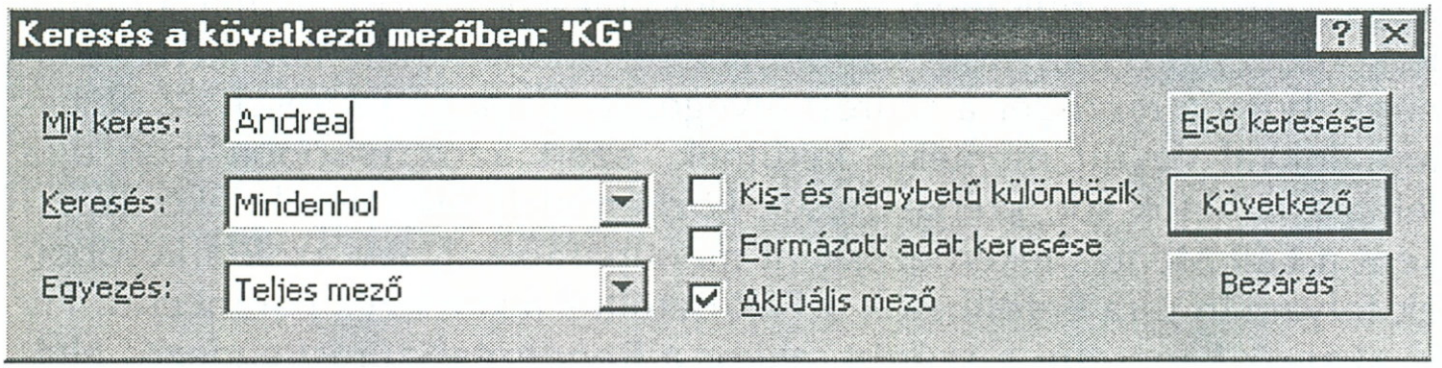
4.7. Adatkezelés

Adatbázis kezelésekor előfordul, hogy egy feltételnek megfelelő rekordot kell megkeresni, módosítani, vagy egyéb műveletet elvégezni rajta. Az Access megkönnyíti nekünk ezeket az egyszerűbb feladatokat.

4.7.1. Adatok keresése

Két lehetőségünk van az egyszerű keresésre. Ha tudjuk melyik mezőben van a keresendő adat, álljunk a megfelelő kategóriában / tábla, lekérdezés, űrlap / erre a mezőre és kattintsunk a Szerkesztés/Keresés menüparancson. A megjelenő párbeszédpanelon válasszuk az Aktuális

mező lehetőséget, írjuk be a keresendő értéket, majd kattintsunk az Első keresése gombra. Ha az adat további előfordulására is szükségünk van, kattintsunk Következő gombra. Amennyiben nem tudjuk melyik mezőben kell keresni az adatot, választhatjuk a mindenhol beállítást a Keresés ablakban ugyanezen a panelon. Beállíthatjuk itt azt is, hogy a kereséskor a mezőben található adatnak milyen mértékben kell egyeznie a keresendő adattal. Speciális keresésre is van lehetőségünk. A speciális választási lehetőséget megadhatjuk szögletes zárójelek között. Pl. a Cházár és a Császár szavak keresésére megadhatjuk a következő keresendő szót: **C[hs]á*zár.** Ebben az esetben a C betű után a h és az s betűt próbálja megkeresni, a csillag helyén pedig mindegy milyen betűt talál az Access.



4.7.2. Adatok cseréje

Az adatok cseréje hasonlít a kereséshez. Amennyiben a keresés paranccsal meg tudjuk találni amit keresünk, akkor már nem gond a csere, hiszen a Szerkesztés/csere menüparancs párbeszédpanelje nagyon hasonlít a kereséséhez. Kiegészült egy csere sorral, ahova be kell írni, hogy mire cserélje le a megtalált adatot. Ez rendkívül hasonló az eddig megismert szövegszerkesztőkben használt keresés, csere parancsokhoz.

4.7.3. Adatok rendezése, szűrése

Erre több lehetőséget is kínál a rendszer. Amennyiben nem kívánjuk lekérdezésre alapozni a rendezést, akkor tábla adatlap, űrlap adatlap vagy űrlap nézetben rendelkezésünkre áll a gyorsrendezés. Álljunk arra a mezőre, amelyik szerint el akarjuk végezni a rendezést, és válasszuk a Rekord/Rendezés Növekvő, vagy Csökkenő parancsát. A gyorsrendezést a Rekord/Szűrés/Rendezés eltávolítása menüvel kapcsolhatjuk ki. Rendezésre használhatók az eszköztáron található Rendezés csökkenő, vagy a Rendezés növekvő ikonok is. Végezhetünk különböző szűréseket is adatbázisunkon, ilyenkor az adataink szűrve jelennek meg. Erre több lehetőséget is kínál a rendszer. Legegyszerűbb a kijelölés alapján

elvégzett szűrés. A kiválasztott mezőre lépve, ekkor nyilván kiválasztottunk egy adatot, a Rekordok/Szűrő/Szűrés kijelöléssel menüt választva a rendszer csak azokat a rekordokat jeleníti meg, amelyik a feltételnek megfelel. Nevezetese azokat a rekordokat, amelyik rekordban az adott mezőben az előzőleg kiválasztott adat van. Végezhetjük a szűrést űrlappal is, ekkor az adatlap nézetű űrlapunkon az egyes mezőknél megjelenő legördülő ablakban állíthatjuk be a kívánt feltételt. Ehhez a Rekordok/Szűrő/Szűrés űrlappal menüt kell kiválasztani, majd a kívánt feltételek kiválasztása után a Rekordok/Szűrés/Rendezés menüparancscsal rendezzük adatbázisunkat. A legbonyolultabb szűrési lehetőségeket az Irányított szűrés kínálja. Ilyenkor a szűrést lekérdezés segítségével végezzük el. A lekérdezések összeállítását későbbiekben a lekérdezések között ismerjük meg. A megfelelő lekérdezés összeállítása után a Rekordok/Szűrő/Irányított szűrő/rendezés menüpontot kell kiválasztanunk, hogy a rendszer a szűrést elvégezze. Ezek az szűrések az objektumainkkal együtt mentésre kerülnek, ezért azok későbbiekben újra felhasználhatók sőt, szerkesztésük közben el is menthetők lekérdezésként a Fájl/Mentés lekérdezésként menüparaccsal, így azok a későbbiek folyamán újra betölthetők a Fájl/Betöltés lekérdezés paranccsal. Ha ki akarjuk kapcsolni a szűrést, akkor a Rendezés/Szűrés/rendezés eltávolítása parancsot kell választanunk. Ezek a szűrők alkalmasak a lekérdezésekhez hasonlóan arra, hogy adatbázisunkból megjeleníthessük például a budapesti lakosokat, vagy a budapesti V. kerületi lakosokat, stb.

4.7.4. Megjelenítés, formázás

Mivel az adatlap nézetet gyakran használjuk megjelenítésre, ezért az Access lehetőséget ad a megjelenítés formázására. Formázott táblázatunk nyomtatáskor sokkal esztétikusabban jelenhet meg ezáltal. Formátum/Cellák paranccsal az Excel-ben megszokott módon formázhatjuk celláinkat. A Formátum/Betütípus paranccsal a Word-höz hasonlóan formázhatunk. Beállíthatjuk a Formátum menüben az oszlopok szélességét, a sorok magasságát, elrejthetjük, felfedhetjük az egyes oszlopokat, valamint lehetőségünk van az egyes oszlopok rögzítésére, fagyasztására, így azok mindig a képernyőn maradhatnak. Az oszlopokat átrendezhetjük úgy, hogy a mozgatni kívánt oszlopot kijelölése után, a fejléc fölött az egérrel megfogjuk és vonszolással az új helyre dobjuk. Ily módon átrendezhetjük adatbázisunk oszlopait a kívánt sorrendbe.

4.8. Access-ben található vezérlőelemek

Bár a Windows más irodai alkalmazásaiban is megtalálhatók a vezérlőelemek, itt kiemelt jelentőségük van. Az egyes alkalmazásokban rendkívül sokat használjuk egyes műveletek elvégzésére, a megjelenítés szabályozására. Azok a vezérlőelemek amelyekkel az adatok megjelenítését szabályozzuk, használhatók mind a négy objektumfajtában, atöbbit az űrlapokon és jelentéseken használjuk. Az egyes vezérlőelemek számos tulajdonsággal rendelkeznek, ezeket négy csoportba sorolhatjuk: formátum-, adat-, esemény-, egyéb tulajdonságok.

4.8.1. Beviteli mező

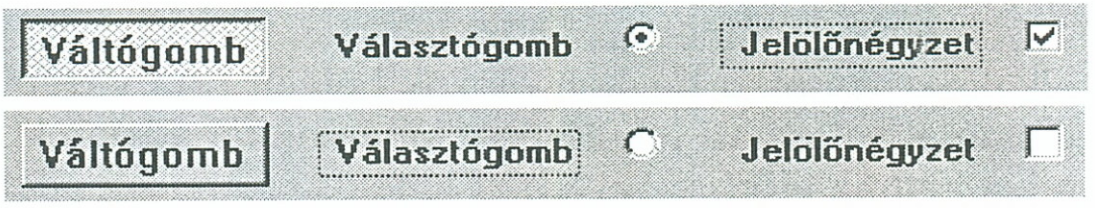
Elsősorban adatok beírásához, vagy szerkesztéséhez, táblából, lekérdezésből kapott információk és számított értékek megjelenítéséhez használjuk. Az egyes űrlapok leggyakrabban használt vezérlőeleme, melyhez gyakran kapcsolódik egy másik vezérlőelem a címke.

4.8.2. Címke

E vezérlőelem segítségével különféle magyarázó szöveget helyezhetünk el, illetve egyes vezérlőelemhez kapcsolódhat címet adva a vezérlőelemnek. Ha többsoros címet szeretnénk adni egy vezérlőelemnek, akkror a címkében új sort a Ctrl-Enter billentyűkombinációval tehetjük ezt meg.

4.8.3. Váltógomb, választógomb, jelölőnégyzet

Ha egy adatnak két értéke van, akkor esetleg választani kell köztük. Erre szolgálnak a fenti ábrán látható vezérlőelemek. A három elemnek két-két állása van, melyet kattintással választhatunk ki. A Választógombot elsősorban csoportban, a Jelölőnégyzetet önállóan használjuk. Egy mezőhöz kötött vezérlőelem automatikusan kiválasztott állapotban jelenik meg az űrlapon, ha a mezőben lévő adat megegyezik a gomb választási tulajdonságával. Ezek a vezérlőelemek teljesen egyenértékűek és az űrlap arculata dönti el hogy melyiket alkalmazzuk. Ez



a saját ízlésünknek megfelelően alakul. Váltógombunkat egy beleírt szöveggel, vagy egy belehelyezett képpel szemléletesebbé tehetjük, ha az űrlap tervező nézetében a jobb gombbal a vezérlőelemen kattintva a tulajdonságokat választjuk, vagy dupla kattintással a váltógombon annak tulajdonságlapján található cím mezőbe beleírjuk a gomb címét. Ezen a lapon a kép tulajdonságra, majd a három pontra kattintva hívhatjuk meg a képszerkesztőt és választhatunk a lehetőségek közül, vagy tallózhatunk a vincseszterünkön, vagy akár magunk is összeeszkábálhatunk ízlésünknek megfelelő képet a gombra.

4.8.4. Vezérlőelem csoport

Vezérlőelem csoportot akkor alkalmazunk, mikor több egymást kizáró lehetőség közül kell választanunk. Ennek megfelelően ezt az elemet választógomb, váltógomb, vagy jelölőnégyzet vezérlőelemekkel kombinálva használjuk. A fenti vezérlőelemekből képzett csoportok egyes elemei közül mindig csak az egyik elem lehet kiválasztva. Ha vezérlőelem csoportot kívánunk létrehozni, akkor célszerű a  Vezérlőelem varázslót az eszközsoron előtte bekapcsolni, majd szintén az eszközkészlet eszközsoron található Vezérlőelem csoport gombra kattintva elindul a Vezérlőelem varázsló, így lényegesen  leegyszerűsödik az elem létrehozása. A varázsló első lépésében meg kell adnunk az elemek címkeneveit, következő lépésben meghatározzuk az alapértelmezett elemet, majd az elemek választási tulajdonságait kell beállítanunk. Továbblépve a varázsló következő oldalára meg kell határozni a visszakapott értékek sorsát, amit felhasználhatunk az űrlapon választásra. Az utolsó előtti lépés a vezérlőelemek megjelenési formáját határozza meg, hogy mely elemekből álljon a csoport, majd az utolsó lépés a vezérlőelem csoport nevének meghatározása lesz.

4.8.5. Lista és kombinált lista

Azért, hogy minél kevesebb gépelési munkánk legyen az űrlapokon, alkalmazhatunk a beviteli mezők helyett lista vezérlőelemet. Ezzel az elemmel választási lehetőséget kínálunk fel adatainkból, így korlátozva a bevitelt az előre megszabott lehetőségekre. Az adatbevitel választása a Windows-ban megszokott módon történhet egérkattintással, vagy az enter billentyű leütésével, de az ablakban a gyors ugráshoz beírhatunk kezdőbetűt is. Az esetek nagyobb részében az elem tartalma kiválaszt-

ható egérrel is a készletből, de szükség lehet egyedi adat beírására, ilyenkor a kombinált listát alkalmazzuk, ami egy listát és egy beviteli mezőt tartalmaz egyben. Első ránézésre ez egy beviteli mezőnek tűnik, de az ablak jobb szélén található legördítő nyíl segítségével a legördülő listából választhatunk is a megjelenő adatok közül. Persze a beviteli mezőbe közvetlenül is beírhatjuk a kívánt értéket. Ezeket az elemeket szintén a varázslóval a legcélszerűbb létrehozni. A varázslógomb bekapcsolása után a lista és a kombinált lista vezérlőelemeket húzzuk az űrlapon a megfelelő helyre, majd a varázsló kérdéseire válaszolva beállítjuk a vezérlőelem tulajdonságait.

4.8.6. Parancsgomb

Lehetőségünk van az űrlapokon elhelyezni parancsgombokat, melyekkel eseményeket, különböző műveleteket indíthatunk el, vagy esetleg nyugtázhatunk bizonyos eseményeket, például elhelyezünk egy OK gombot, vagy egy Mégsem gombot és elég a gombra kattintani az esemény nyugtázására. A parancsgombokat tetszőleges funkcióval ruházhatjuk fel. Elindíthatunk egy makrót, vagy futtathatunk egy Visual Basic programot, vagy akár ki is léphetünk az Acces-ből. A gombbal tulajdonképpen egy ógynevezett eseményvezérelt eljárást indítunk el. Mint a többi vezérlőelemet ezt is a varázsló segítségével hozzuk létre. A varázsló első lépésében az eseményvezérelt eljárást állítjuk be. Erre számos lehetőséget kínál fel. Meg kell írni előzőleg a végrehajtandó makrót, és hozzárendelni a varázslóval a gombhoz. A hozzárendelés után a varázsló javasol egy képet a gombra, majd megadhatjuk a gomb címét is. Ha megnyitjuk a már az előző elemeknél említett módon a tulajdonságpanelt, akkor egy sor új tulajdonságát állíthatjuk be parancsgombunknak. Az automatikus ismétlés tulajdonsággal például beállíthatjuk, hogy a makró csak egyszer fusson le, vagy folyamatosan újrainduljon, mikor a gombot folyamatosan nyomva tartjuk. Parancsgombot a varázslógomb bekapcsolása után az eszközsorról behúzva helyezzük el a kiszemelt helyre, majd a varázsló indulása után beállítjuk a tulajdonságait.

4.8.7. Egyéb vezérlőelemek

Elhelyezhetünk űrlapunkon még számos vezérlőelemet, a varázslók ebben segítséget nyújtanak. Ilyen elemek a kép, amit díszítésre használhatunk, kötetlen objektumkeret, amellyel egy külső alkalmazásból

származó adatot tudunk elhelyezni űrlapunkon, vagy jelentésünkön. Kötött objektumkeretben például egy külső objektumot tartalmazó mezőt helyezhetünk el, például a vállalatunk dolgozóinak fényképét. Az új oldal vezérlőelemmel új oldalt nyithatunk. A vonal vezérlőelemmel vonalat, a téglalap vezérlőelemmel téglalapot helyezhetünk el. Külső fejlesztők által készített vezérlőelemet is elhelyezhetünk, de ezeket előbb regisztráltatni kell az Access-el az Eszközök/Egyéni vezérlőelemek menüparancs segítségével, majd tervező nézetben a Beszúrás/Egyéni vezérlőelemek menüparanccsal helyezhetjük el a jelentésen, vagy az űrlapon.

4.9. Tulajdonságok

4.9.1. Formátumok

Az adatok megjelenítéséhez, hogy igény szerint lehessen például kinyomtatni egy jelentést, be kell állítanunk azok megjelenítési formáját, formátumát. Ehhez a rendszer rengeteg lehetőséget biztosít, számos szabványos formátum között válogathatunk. Amennyiben egyik sem megfelelő a felkínáltak közül, saját magunk is készíthetünk megfelelő formátumot. Pénznem és dátum formátum esetén a rendszer a Windows nemzetközi beállításait kínálja fel, ezt az alapértelmezett lehetőséget a Windows-ban kell módosítani. Saját formátumot egy karakterlánccal tudunk készíteni, ahol minden egys betűnek meghatározott jelentése van. Természetesen a karakterlánc betűi kötöttek, nem lehet akármelyik betűkből összeállítani.

4.9.1.1. Dátum-, idő:

➢	c	általános dátumformátum
➢	/	elválasztójel
➢	.	idő elválasztó jele
❖	d	a napok számmal, egy vagy két digiten
❖	dd	napok számmal 2 digiten
❖	dddd	napok betűkkel
❖	ddddd	rövid dátum
❖	dddddd	hosszú dátum
❖	m	hónapok számmal egy, vagy két digiten
❖	mm	hónapok számmal két digiten
❖	mmm	hónap betűvel rövidített formában
❖	mmmm	hónap betűvel teljes hosszúságban
❖	yy	évszám évszázad nélkül

- yyyy év évszázaddal
- w a hét napjai számmal
- ww az év hetei számmal
- q negyedév számmal
- h óra egy, vagy két digiten
- hh óra két digiten
- n percek egy, vagy két digiten
- nn percek két digiten
- s másodpercek egy, vagy két digiten
- ss másodpercek két digiten
- tttt teljes hosszúságú idő
- am/PM 12 órás idő, az am kis, vagy a PM nagybetűvel
- a/P 12 órás megjelenítés, az a kis, vagy P nagybetűvel
- AMPM 12 órás megjelenítés az AMPM a Windows Vezérlőpult beállítás szerint jelenik meg a délelőtt, délután

4.9.1.2. Szám és pénznem

- , tizedesjel
- . ezres csoportok
- 0 kötelező számjegy, akkor is ha a jegy 0 értékű
- # számjegy, ha nincs számjegy, akkor nem jelenik meg
- % százalék, a számot szorozza százzal és a % jel
- E normál forma

Az összeállított formátum karakterlánc négy részből állhat, ezek egymástól pontosvesszővel vannak elválasztva. Az egyes részek másmás számtípusra vonatkoznak.

1. rész A pozitív számokra
2. rész A negatív számokra
3. rész Nulla értékek
4. rész Null és üres értékek

4.9.1.3. Logikai

Három részből állhat a karakterlánc.

1. rész Logikai értéknél nem használjuk, de a pontosvesszőt ki kell rakni
2. rész Igaz érték
3. rész Hamis érték

A részekben bármilyen karakterlánc használható, pl.: (;"Nehéz"; "Könnyű"). Az igaz –1, a hamis 0 értékű. Egy feltétel vizsgálat eredmé-

nye igaz, vagy hamis lehet. A visszakapott logikai értéknek megfelelő számot használja fel a rendszer a további műveletek elvégzéséhez. A fenti példában, ha a logikai vizsgálat eredménye –1, akkor a Nehéz érték kerül visszaadásra, ha 0, akkor a könnyű értéket adja vissza a rendszer további felhasználás végett. Ez a forma pontosan megfelel a számformátumban használt karakterlánc típusnak. Az első rész, amit nem használunk a pozitív számok, a második rész a negatív számok (-1), a harmadik rész a nulla értékek (0).

4.9.1.4. Szöveg

A formátumozó karakterlánc három részből állhat. Az egyes részeket egymástól pontosvesszővel kell elválasztani.

1. rész Szöveges mezők
2. rész Nulla hosszúságú karakterláncok
3. rész Null értékek

A karakterláncban az alábbi jelölések lehetnek:

<	Az összes karakter nagybetűs lesz
>	Az összes karakter kisbetűs lesz
&	Nem kötelező karaktert írni
@	Kötelező karaktert írni, ez lehet szóköz is

4.9.2. Tulajdonságok

4.9.2.1. Kép

Ennek segítségével állíthatjuk be egy vezérlőelemben (parancsgomb, váltógomb, képkeret) megjelenítendő kép nevét. Ez tulajdonképpen a megjelenítendő képet tartalmazó fájl neve, és elérési útja.

4.9.2.2. A kép típusa

Azt határozza meg, hogy a képfájl csatolt, vagy beágyazott típusú. Amennyiben csatolt, akkor az elkészült alkalmazással a képfájlt is vinni kell, ha másik gépre telepítjük.

4.9.2.3. Átlátszó

Ez igen érdekes tulajdonság, ha igenre állítjuk egy parancsgomb esetén, akkor a parancsgomb nem látszik. Több ilyen parancsgomb elhelyezésével azt a látszatot kelthetjük, hogy az űrlapon bárhol kattintunk, mindig történik valami, pedig nem is látszik semmi az űrlapon azon a helyen.

4.9.2.4. Méretfajta

Az egyes vezérlőelemekbe kerülő kép (kép-, kötött- kötetlen objektumkeret) méretezését állítja be a beillesztéshez.

- Méretezés érték esetén az objektum a rendelkezésre álló helyhez igazodik, arányait megtartva.
- Kitöltés érték esetén kitölti a rendelkezésre álló helyet, arányai így torzulhatnak.
- Kivágás érték esetén eredeti méretben kerül beillesztésre, a kieső objektum részletei nem látszanak, a rendszer levágja.

4.9.2.5. Képigazítás

A vezérlőelemen belül igazíthatjuk a megjelenő képet a megfelelő helyre.

4.9.2.6. Mozaikszerű

Amennyiben igenre állítjuk ezt a tulajdonságot, akkor a vezérlőelemben a kép mozaikszerűen ismétlődni fog. Mint a csempék a falon.

4.9.2.7. Látható

E tulajdonság segítségével állíthatjuk be, hogy az űrlapon látszódjéke, vagy nem a vezérlőelem (bármely típusú). Amennyiben ezt a tulajdonságot egy eseményvezérelt makróba helyezzük, akkor a vezérlőelem egy esemény bekövetkezésétől függően fog megjelenni.

4.9.2.8. Kijelzés

Azt állíthatjuk be, hogy a vezérlőelem hol jelenhet meg (bármelyik).

- Csak nyomtatón érték esetén, csak a nyomtatot papíron fog látszani.
- Csak képernyőn érték esetén, csak a monitoron fog látszani.
- Mindkét érték esetén mind a két periférián megjelenik a vezérlőelem.

4.9.2.9. Gördítősáv

Az adott objektumban, persze a megjelenítendő hossztól függően jelenjen, vagy ne jelenjen meg gördítősáv.

4.9.2.10. Növelhető

A beviteli mező, vagy a segédűrlap mérete függőlegesen növelhető, vagy nem. A beállított mérettől eltérhetünk-e, ha az adat, vagy adatok nem férnek el bennük. Igenre állítva a vezérlőelemet tartalmazó hasonló tulajdonságokat igenre állítja, ezzel elronthatjuk az űrlap megjelenési formáját. Tehát fontoljuk meg a beállítást.

4.9.2.11. Összenyomható

A beviteli mező magasságát csökkenthetjük, az üres sorok így kiiktathatók.

4.9.2.12. Balra

A szakasz bal szegélyéhez viszonyítva a vezérlőelem hol helyezkedik el vízszintesen.

4.9.2.13. Fel

Az adott szakaszban a vezérlőelem függőlegesen hol helyezkedik el a szakasz felső szegélyéhez viszonyítva.

4.9.2.14. Szélesség

A vezérlőelem szélességét állíthatjuk ezzel.

4.9.2.15. Magasság

A vezérlőelem magassága állítható ezzel.

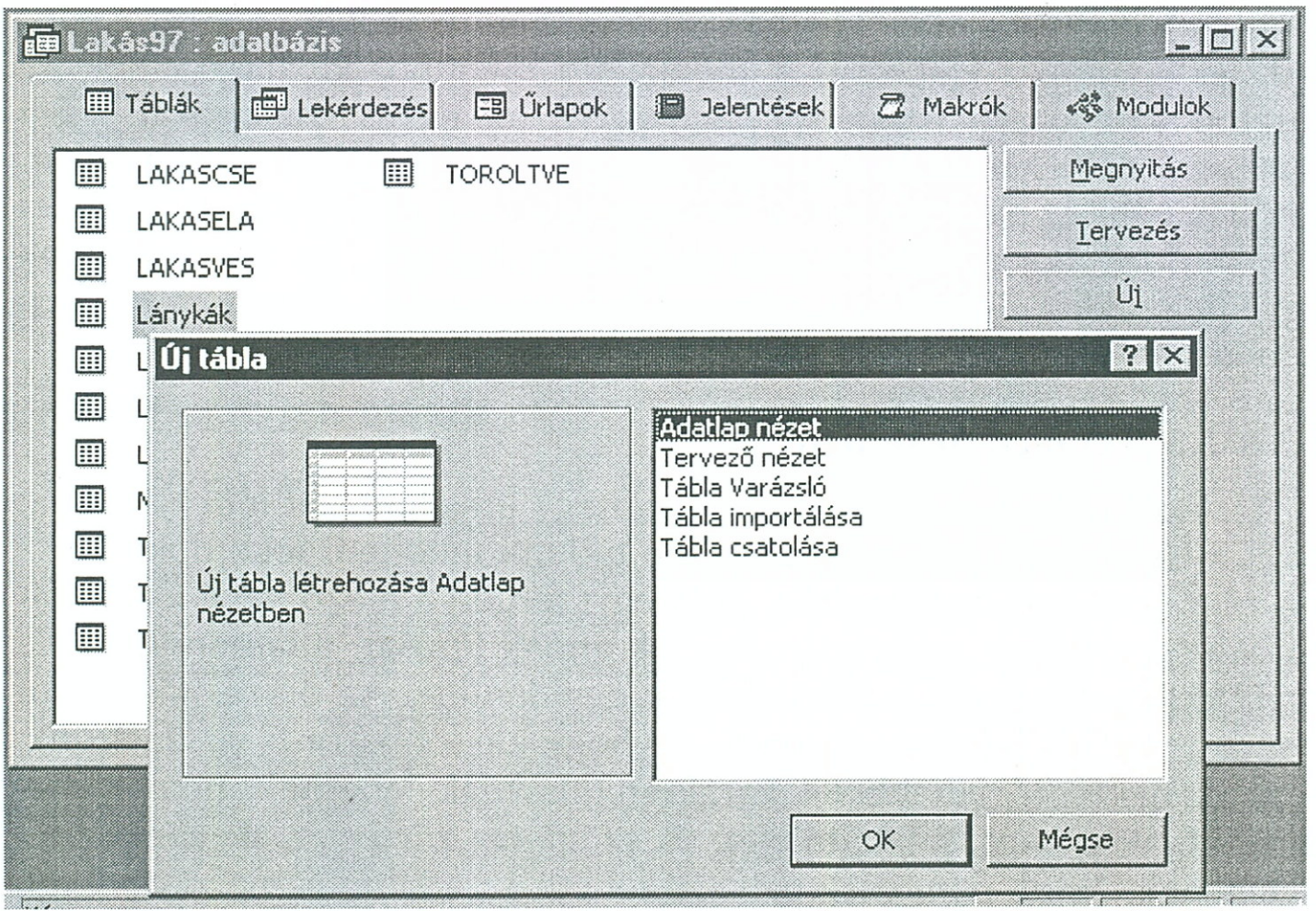
És még számtalan tulajdonsága állítható rendkívül egyszerűen az egyes vezérlőelemeknek. Ezek a tulajdonságok párbeszédpaneleken állíthatók,

némelyik tulajdonság állítását még varázsló is segíti. A párbeszédpanel (tulajdonságlap) előcsalogatható, ha a vezérlőelemen a jobb gombbal klikkelünk, és az előugró menüben a tulajdonság parancsot választjuk. További tulajdonságai például a háttérstílus, háttérszín, speciális hatás, szegély stílusa, színe, szélessége, előtér színe, betűtípus, betűméret, betűvastagság, dőlés, aláhúzás, szövegigazítás, automatikus címke, kettőspont hozzáadása, címke X, címke Y (a címke helyzetét határozza meg a vezérlőelem bal felső sarkához viszonyítva), címke igazítása (a vezérlőelemben megjelenő szöveget igazíthatjuk).

4.10. A táblák

4.10.1. Létrehozás

A táblák oszlopokból és sorokból álló adatok együttese, valamilyen rendezett formában. Az Access alkalmazások egy vagy több táblában tárolhatják adatainkat, ezt az adatbázis tervezésekor kell eldönteni. Ez nem jelenti azt, hogy a későbbiekben ez nem módosítható. Az információk összegyűjtése után megtervezzük adatbázisunkat, hogy hány táblából álljon, és milyen mezőkből, majd létrehozzuk az egyes táblá-



kat. Többféle módon hozhatunk létre táblát. Adatbázis Varázslóval, ekkor nem csak a táblák jönnek létre, hanem az összes objektuma az alkalmazásnak. Tábla Varázsló segítségével. Tervező nézetben üres táblából kiindulva. Üres adatlapba írással. Létrehozhatunk táblát még táblakészítő lekérdezéssel is, importálással, csatolással.

Az adatbázis ablakban található új gomb lenyomásakor az Új tábla párbeszédpanel nyílik meg, mely a fenti lehetőségeket kínálja fel nekünk.

4.10.1.1. Tábla Varázslóval

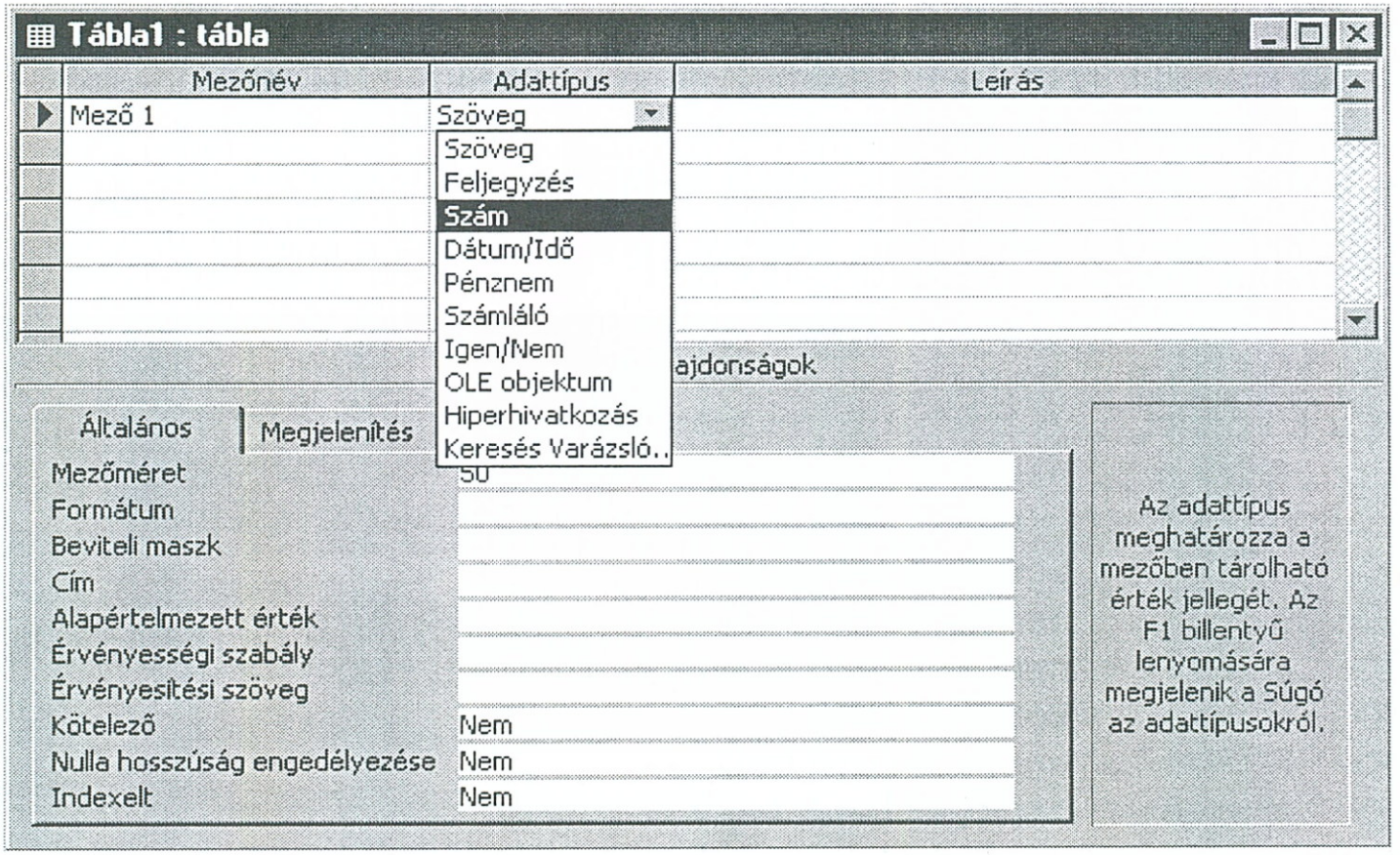
Előre elkészült mintákat kínál fel, ezekből választhatjuk ki a hasonló témájú táblát, a varázsló utasításainak megfelelően eljárva rendkívül egyszerűen létrehozhatunk táblákat, amely a későbbiekben természetesen módosítható. A mezők adattípusát és formátumát a varázsló a minták alapján határozza meg.

4.10.1.2. Adatlapba írással

Az Új tábla panelon (lásd fent) válasszuk ki az Adatlap nézetet. Ekkor kapunk egy 30 sorból, és 20 oszlopból álló adatlapot. A tábla neve Tábla1 lesz, a mezők neve Mező1, Mező2, Mező3, ..., Mező20 lesz. Duplán kattintva a mezők nevén, átnevezhetjük azokat. A jobb gombbal, az adatlapra kattintva, a megjelenő menüből kiválaszthatjuk az Oszlop átnevezése parancsot, átnevezve az egyes oszlopokat, vagy törölve a nem kívánt oszlopokat, elkezdhetjük kitölteni adatlapunkat. Kitöltés után az eszközsoron a lemezre kattintva elmentjük. A rendszer megkérdezi, hogy létrehozzon elsődleges kulcsot, ha a rekordjaink egyediek, akkor általában a nemmel kell válaszolni, ellenkező esetben a rendszer létrehoz egy elsődleges kulcsot is mentés előtt.

4.10.1.3. Tervező nézetben

Tervező nézetben lehetőségünk van egy üres táblázat létrehozására, amit a létrehozás után, adatnézetben tudunk feltölteni rekordokkal. Az egyes mezők létrehozása után, amit egyszerűen a mező nevének beírásával végezhetünk, meghatározhatjuk annak típusát, tulajdonságait, és az elsődleges kulcsot is kijelölhetjük egyidejűleg. Az összes táblát ebben a nézetben lehet módosítani, úgyhogy ide többször is visszatérünk. **A mezőnevek legfeljebb 64 karakter hosszúak lehetnek, nem tartalmazhatnak pontot, felkiáltójelet, és szögletes zárójelet, de szóköz lehet benne.** Tehát nem csak egy szó lehet! Az adattípus mező dönti el,



hogy milyen adatok kerülhetnek a mezőkbe, az egyes mezők beállítható tulajdonságait a mezők adattípusa határozza meg.

⌦ Szöveg	alfanumerikus	255 karakter
⌦ Feljegyzés	szavak, mondatok, bekezdések	64000 karakter
⌦ Szám	tört és egész számok	1, 2, 4, 8 bájt
⌦ Pénznem	4 tizedes jegy pontossággal	8 bájt
⌦ Számláló	eggyel növekszik automatikusan	4 bájt
⌦ Dátum, idő	dátumok, időpontok	8 bájt
⌦ Logikai	igen, nem, ki, be, true, false	1 bit
⌦ Objektum	kép, rajz, hang, stb.	1 Gbájt

4.10.2. Bemeneti maszk

Ha azonos adatokat viszünk be kötött módon, célszerű beviteli maszkot létrehozni. A bemeneti maszk elválasztó karaktereivel csoportosíthatjuk a mezőbe bevitt adatot, és biztosítja, hogy csak a helyes adatot vigyük be. A maszkot létrehozhatjuk a Beviteli maszk Varázslóval, de létrehozhatunk egyedi maszkokat is úgy, hogy az űrlapon, vagy a jelentésen lévő vezérlőelem Bemeneti maszk tulajdonságát beállítjuk, beírjuk a helyes maszkot. Ez a tulajdonságmaszk legfeljebb három részből állhat, melyek pontosvesszővel vannak egymástól elválasztva. Az első rész a maszk formátumát, a második a formátumkarakterek tárolását vezérlő számkaraktert tartalmazza, a harmadik pedig a

maszkban kitöltendő helyeket kijelölő karaktert tartalmazza. Amennyiben a második rész 1–et tartalmaz, vagy nem tartalmaz semmit, akkor csak a mezőbe vitt karakterek tárolódnak, ellenkező esetben, ha 0-t tartalmaz, akkor a maszk formázó karakterei is tárolódnak. Ha a harmadik részt nem adjuk meg, akkor a rendszer az „anderscore" „_" jelet használja Például: (999) 000-00001;0;" "bemeneti maszkot jobbról balra kell kitölteni, az első három szám nem kötelező. A második részben lévő 0 miatt nem csak a számot kell tárolni, hanem a zárójeleket és a kötőjelet is. A harmadik részben szereplő szóköz karakter azt jelenti, hogy a be nem vitt karakterek helyére szóközt jelenítsen meg a rendszer. A használható maszk karakterek a következők lehetnek:

- 0 kötelező számjegy
- 9 nem kötelező számjegy, vagy szóköz
- # nem kötelező számjegy előjellel, vagy szóköz
- L kötelező betű
- ? nem kötelező betű
- A kötelező betű, vagy számjegy
- A nem kötelező betű, vagy számjegy
- & bármilyen karakter, vagy szóköz, kötelező
- C bármelyik karakter, vagy szóköz, nem kötelező
- . tizedes hely
- : ezres elválasztó
- - dátum elválasztó
- / idő elválasztó

A felsorolásban szereplő utolsó négy maszk karakter a Windows beállításától függ.

- < minden karaktert, ami ez után szerepel kisbetűre vált
- > minden karaktert, ami ez után következik nagybetűre vált
- ! bárhol szerepel a maszkban, akkor a kitöltését a maszknak jobbról balra kell végezni
- \ ezt a jelet követő karakter a maszkban változatlanul fog megjelenni. Például: \X használatakor X jelenik meg

4.10.3. Index

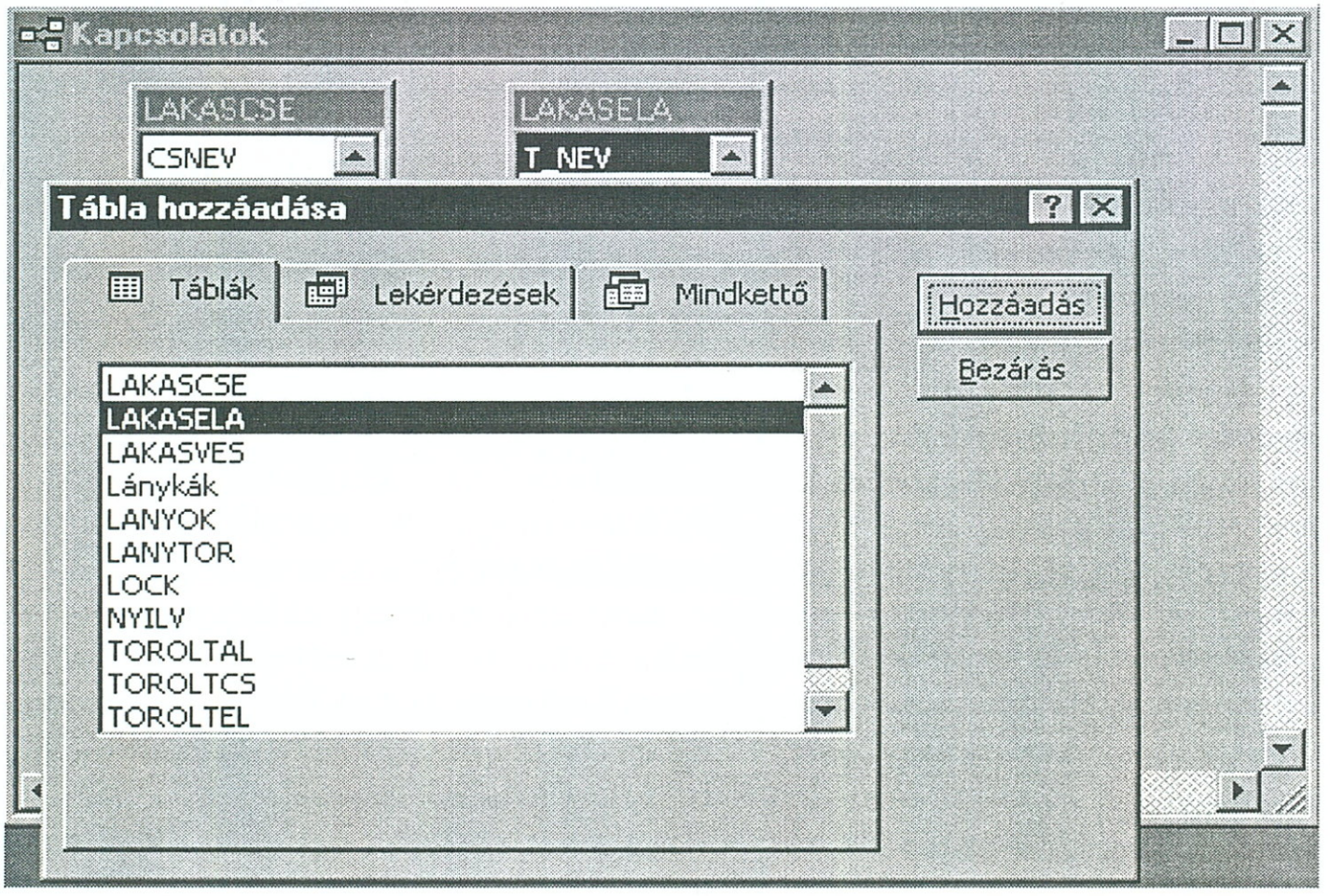
Egy belső táblázat, amely tartalmazza az indexelt mezők értékeit, és a sor számát. Ez lényegesen felgyorsítja az adat visszakeresését. Az indexek lassítják az adatok bevitelét, mert mindig újra kell indexelni. Fontoljuk meg mikor nyerünk az indexeléssel.

4.10.4. Elsődleges kulcs

Az Elsődleges kulcs az a mező, mezők együttese, mellyel egyértelműen lehet egy rekordot azonosítani. Táblák összekapcsolásához mindig létre kell hozni. A rendszer az elsődleges kulcsot figyeli adatbevitelkor, és kizárja a lehetőségét az azonos adatok bevitelének. Ha nem állítjuk át, akkor a jelentéseken, és az űrlapokon az elsődleges kulcs szerinti sorrendben jeleníti meg a rekordokat. Elsődleges kulcsnak egy számláló típusú mező is megfelel. A kulcs beállításához először ki kell jelölni a tábla tervezőnézetében a kulcsnak szánt mezőt, vagy mezőket. Ha több mezőt kívánunk kijelölni egyszerre, akkor a ctrl gomb lenyomása mellet kattintunk a mezők előtti kijelölő négyzetre. A kulcsnak szánt mező, mezők kiválasztása után az eszközsoron az Elsődleges kulcs ikonra kattintunk.

4.10.5. A táblák kapcsolása

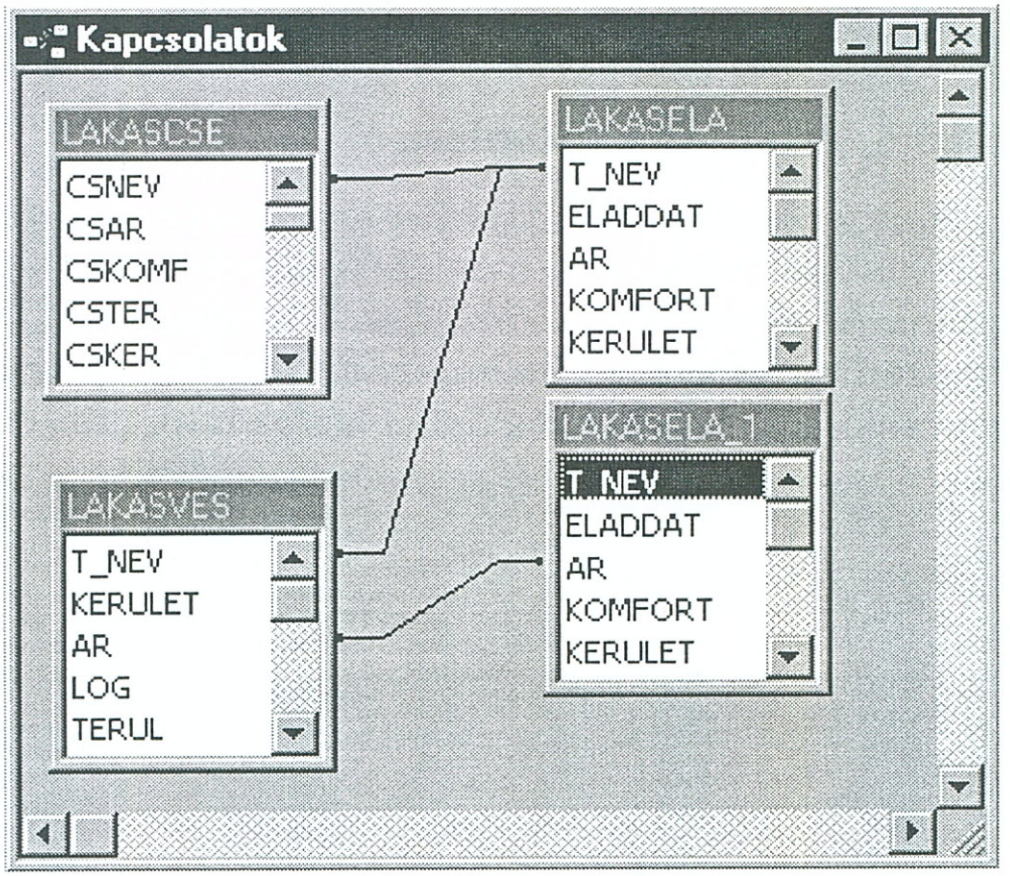
Ahhoz, hogy a több táblába lévő adatainkat egyszerre kezelhessük, a különböző táblákat a megfelelő kulcsok segítségével össze kell kapcsolni. Az egyes táblák között az 1:1 kapcsolatot és az 1:M kapcsolatot egyszerűen létre lehet hozni, de az N:M kapcsolatot csak egy közbülső kapcsolótábla segítségével lehet kialakítani. Lásd a Normalizálásnál.



A kapcsolatok létrehozását az eszközsoron lévő Kapcsolatok ikonra kattintva, vagy az Eszközök menü Kapcsolatok parancsával kezdeményezhetjük.

A kapcsolatok ablakban megjelenő Tábla hozzáadása panelon a Hozzáadás gombbal vihetjük be a kapcsolni kívánt táblákat, ahol azután létrehozhatjuk a megfelelő kapcsolatokat. Ha valamelyik tábla nevét az Adatbázis ablakban megfogjuk, azt egyszerűen át lehet húzni a Kapcsolat ablakba, így is felkerülhet a tábla a kapcsolatok közé. A konkrét kapcsolatokat is rendkívül egyszerű létrehozni. Minden kapcsolatban két tábla vesz részt, az elsődleges, és a kapcsolt.

Az elsődleges táblában kijelöljük a kapcsolni kívánt mezőt, vagy mezőket és az egérrel megfogva átvonszoljuk a kapcsolt tábla kapcsolni kívánt mezője fölé, majd elengedjük. A megjelenő Kapcsolatok párbeszédpanelon ezután beállíthatjuk az új kapcsolat tulajdonságait.



Kapcsolatokat törölni a Del billentyűvel lehetséges, miután a Kapcsolatok ablakban az összekötő vonalra kattintottunk, ezzel kijelölve a megfelelő kapcsolatot. Az egyes táblák is a Del billentyűvel törölhetők, miután a törlendő táblára kattintottunk.

4.10.6. Táblákon végezhető módosítások

4.10.6.1. Táblák törlése

Jelöljük ki az Adatbázis ablakban a feleslegessé vált táblánkat, majd a Del billentyű leütésével, vagy a Szerkesztés menü Törlés parancsával az törlődni fog.

4.10.6.2. Táblák átnevezése

Bizonyos esetekben szükség lehet valamely táblának az átnevezésére, megtehetjük ezt, ha az Adatbázis ablakban kijelöljük az átnevezni kívánt táblát, majd a Szerkesztés menü Átnevez parancsot választva, végrehajtjuk.

4.10.6.3. Táblák másolása

Az Adatbázis ablakban kijelöljük a másolni kívánt táblát. A következő lépésben kivisszük a vágólapra. Szerkesztés menü Másolás parancsára kattintunk. Ezt követően már csak be kell illeszteni az adatbázisunkba, de természetesen két egyforma nevű táblánk nem lehet egy adatbázisban, ezért másik nevet kell adni a beillesztendő táblának. A Szerkesztés menü Beillesztés parancsra kattintva egy párbeszédpanel jelenik meg, ahol meg kell adni a beillesztendő tábla új nevét. Itt be lehet állítani még, hogy a tábla mely része illesztődjön be, csak a tábla struktúrája, vagy az adatok is.

4.10.6.4. Mezőkön végezhető módosítások

A tábla szerkezetén történő módosításokat a tábla tervező nézetében kell végrehajtani. Megnyitjuk a táblát tervező nézetben, és itt a mezőket átnevezhetjük törölhetjük, áthelyezhetjük vonszolással új helyre, a Beszúrás menü Sorok parancsával új mezőt helyezhetünk el, és a mező adattípusát is megváltoztathatjuk. Természetesen előre meg kell fontolni ezeket a módosításokat, hogy milyen hatással lesznek ezek a műveletek az adatbázis egészére, gondoljunk csak a kapcsolatokra.

4.10.7. Adatok importálása, exportálása

Az Access lehetőséget biztosít más adatbázisok adatainak feldolgozására. Ezt importálással tehetjük meg. Megtehetjük azt is, hogy az Access adatait más adatbázisba, illetve szövegfájlba küldjük, ezt exportálásnak nevezzük. Az importálás is és az exportálás is elvégzi helyettünk az Access.. A Fájl menü Külső lekérdezés parancsra kattintva ki kell választanunk a megfelelő könyvtárból az importálandó fájl nevét és az Importálás gombra kattintva rendszer automatikusan átalakítja Access formátumúra, beillesztve adatbázisunkba egy új táblaként. Az exportálást a Fájl menü Mentés/Exportálás parancsával végezhetjük el. Erre kattintva a beérkező párbeszédpanelon kiválaszthatjuk, hogy külső, vagy a jelenlegi adatbázisba mentse más néven a táblánkat. Amennyiben külső adatbázist választjuk, és az Exportálás gombra kattintunk, még át kell állítani a fájl típusát a kívánt kiterjesztésre, ebből

állapítja meg az Access, hogy milyen adatbázisba exportáljuk táblánkat. Ezek lehetnek Access, dBASE, Excel, FoxPro, és ezek különböző verziói, szövegfájlok, Rich text formátum, Word körlevél, ODBC adatbázisok, stb.

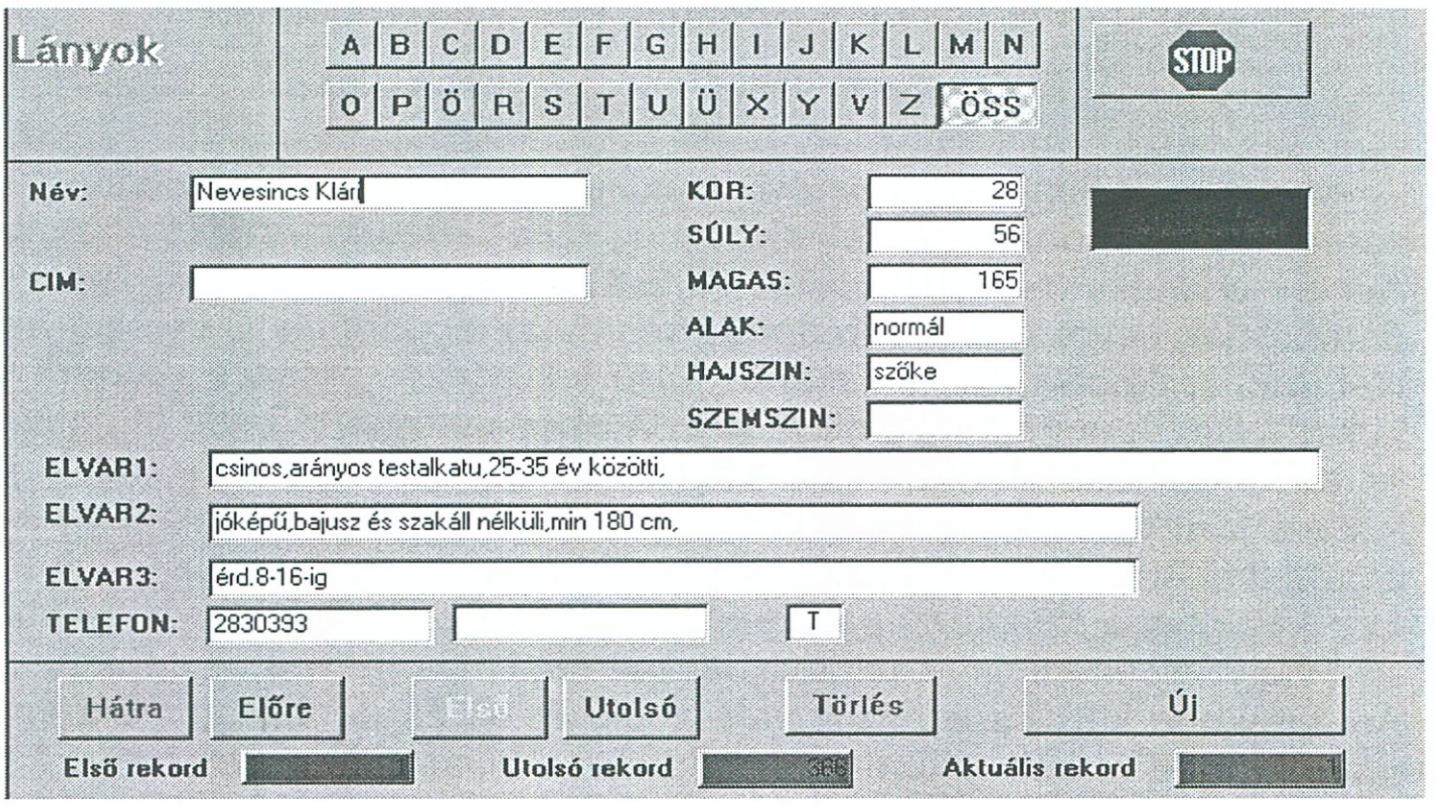
4.11. Űrlapok

Elsősorban adatok beviteléhez használjuk, de a már meglévő adatainkat megtekinthetjük, sőt módosíthatjuk segítségükkel. Ezekhez a műveletekhez a Windows-ban megszokott elemeket használhatjuk (jelölőnégyzetek, beviteli mezők, legördülő listák, és más vezérlőelemeket). Adataink módosításához úgynevezett kötött űrlapot használunk, mert nyilván valamilyen adatforráshoz kell kötni, és ez lehet lekérdezés, vagy tábla. Adatbevitelhez, pedig nem kötött űrlapot használunk. Ezeken az űrlapokon vezérlőelemeket helyezünk el, amelyek az űrlap formáját határozzák meg, és az adatbevitelt szabályozzák.

4.11.1. Űrlaptípusok

4.11.1.1. Egyoszlopos

Lehet bonyolultabb, és egyszerűbb felépítésű, a lényege, hogy egy rekord adatait tudjuk vele bevinni, módosítani. Az összes vezérlőelemtípust elhelyezhetjük rajta, amelyekkel aztán keresgélhetünk adataink között, módosíthatjuk azokat, és új adatokat vehetünk fel adatbázisunkba. Mindezt ízlésünknek megfelelő kivitelű képernyőn.



4.11.1.2. Táblázatos

A táblázatos formájú űrlap az oszlopokat táblázatos formában jeleníti meg, így egyszerre több rekordot is láthatunk. Természetesen ezt is ízlésűnk szerint alakíthatjuk, formázhatjuk. Adhatunk neki hátteret, beállíthatjuk az egyes oszlopok szélességét, az egyes adatmezők magasságát, és még számtalan formázást végezhetünk az űrlapon, hogy megfeleljen ízlésünknek. Az esetlegesen megjelenő gördítősávokkal tovább gördíthetjük ablakunkban az űrlapot, amennyiben nagyobb méretű, mint az ablakunk, és bármely cellába belekattinthatunk azért, hogy a benne lévő adatot módosíthassuk, vagy esetleg új rekordot vegyünk fel az adatbázisunkba.

Lánykák

KOR	Súly	MAGAS	ALAK	HAJSZIN	SZEMSZIN	ELVÁRÁS
28	56	165	normál	szőke		csinos,arányos testa
32	50	164	vékony	fekete		35-40 év között,min.1
38	70	168	molett	fekete	barna	szexpartnert keres,ni
19	55	170		fekete	barna	4-3
38	70	165	molett	vörös	kék	tartós szexpartnert k
43	65	174	normál	barna	barna	magas,jó megjelenési
23	53	172	arányos	fekete	fekete	23-55 év között,rend

Rekord: 1 összesen 368

4.11.1.3. Diagram

Ez az űrlap elsősorban adatok megjelenítésére használatos, bár új adatok felvételét is végezhetjük vele, de jobb ha az előzőkben említett űrlapokat használjuk e célra.

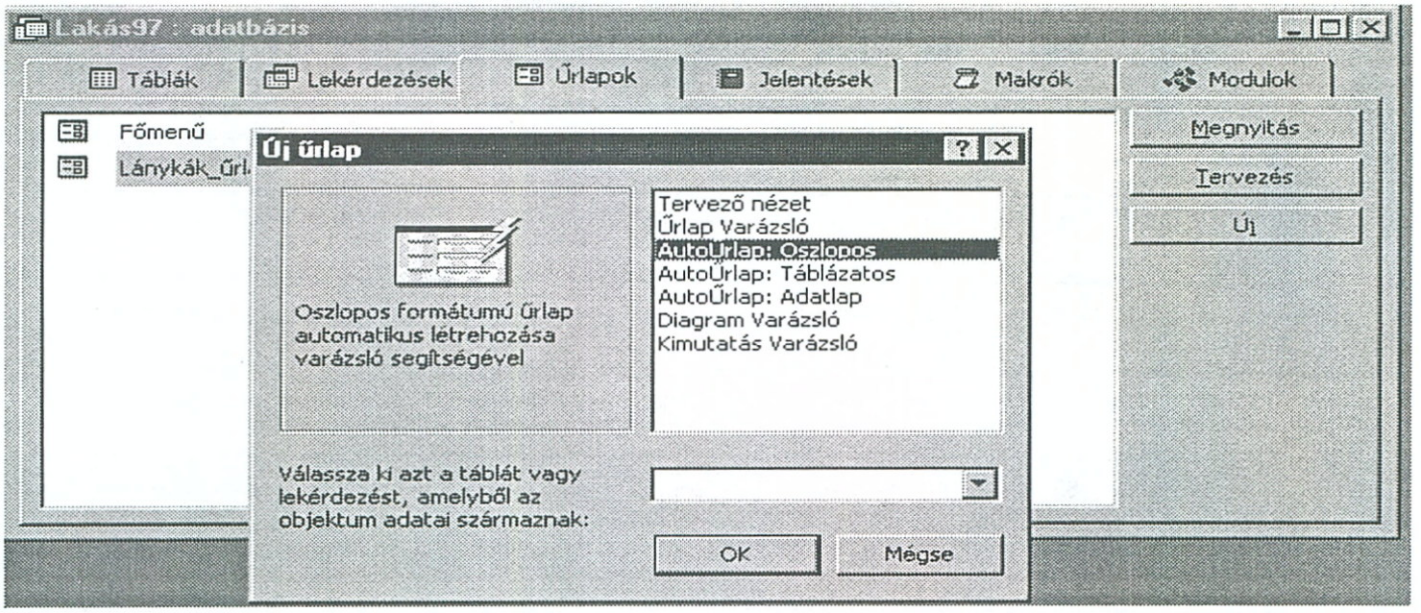
4.11.1.4. Segédűrlap

Ebben az esetben a főűrlapon elhelyezünk egy úgynevezett segédűrlapot. Akkor lehet ilyenre szükség, ha a kapcsolat típusa miatt két táblázatot kel egyidejűleg megjeleníteni. Az 1:N kapcsolat esetén lehet ilyenre szükségünk. Ilyenkor a főtáblához kapcsolt adatokat a főűrla-

pon, a kapcsolódó táblában lévő kapcsolt rekordokat a segédűrlapon jelenítjük meg.

4.11.2. Űrlapok létrehozása

Az új űrlapok létrehozását legjobb ha a varázslókra bízzuk, melyek automatikusan létrehozzák a kívánt űrlapot. Ez persze nem jelenti azt, hogy az elkészült űrlap nem módosítható. Az új űrlap minden eleme változtatható, sőt törölhetünk belőle és hozzá is adhatunk újabb vezérlőelemeket. Az egyes vezérlőelemek tulajdonságait is állíthatjuk, módosíthatjuk a nekünk megfelelő működés érdekében. Az Adatbázis ablakban az Űrlapok regiszterfülre kattintunk, majd az Új gombbal a megjelenő párbeszédpanelon kiválasztjuk a kívánt űrlap típusának megfelelő űrlapvarázslót, és az OK gombra kattintva elindítjuk. Választhatjuk itt többek közt az egyszerű Űrlap Varázslót, az Oszlopos, Táblázatos, vagy az Adatlap AutoŰrlap varázsló stb.



A megfelelő varázsló elindítása után el fogja nekünk készíteni az űrlapunkat az alapértelmezett formában, amit aztán a kívánt formájúra alakíthatunk.

Ha elkészítünk varázslóval egy űrlapot és tervezőnézetbe kapcsolunk, rögtön szembetűnik, hogy több részből tevődik össze. Több szakaszból áll. Ezek lehetnek:

- Űrlapfejléc
- Oldalfejléc
- Törzs
- Oldallábléc
- Űrlaplábléc

Űrlapfejléc az űrlap első oldalán jelenik meg, a többi oldalon, amennyiben több oldalas az űrlap, nem. Például az űrlap címét lehet itt megadni. Az oldalfejléc csak nyomtatásban jelenik meg, minden oldalon látható lesz. Ebbe szoktuk az egyes mezők neveit elhelyezni. A törzs tartalmazza a rekordokat, a tényleges adatainkat. Az oldallábléccben helyezzük el az olyan adatokat, amelyeket minden oldalon, az oldal alján meg akarunk jeleníteni. Például az oldalszámot. Az űrlaplábléccben, mivel ez csak az utolsó oldal alján jelenik meg, szoktuk elhelyezni az OK gombot, vagy az olyan vezérlőgombot, amit az űrlap kitöltése után fogunk csak használni.

4.11.3. Űrlap nézetek

Az egyes űrlapot háromféle nézetben jeleníthetjük meg, attól függően, milyen műveletet akarunk végrehajtani rajta.

4.11.3.1. Űrlapnézet

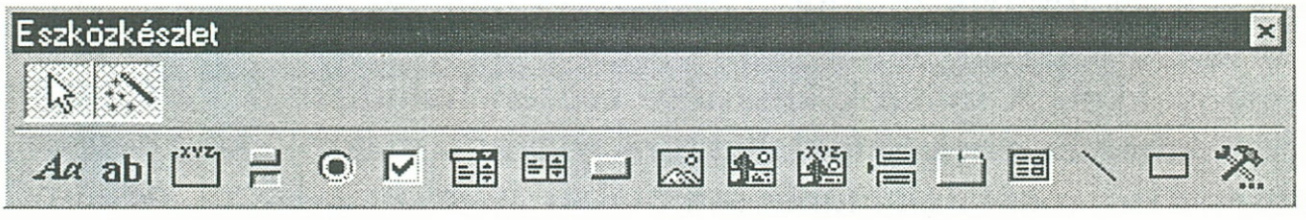
Ha egyetlen adatrekordot kívánunk egyszerre látni, módosítani, akkro választjuk ezt a nézetet. Persze ebben a nézetben is elérjük az összes rekordot, de csak egy van jelen az űrlapon. Az űrlap alján van vezérlőelem amivel továbbléphetünk a következő rekordokra.

4.11.3.2. Adatlap nézet

Ha egyszerre szeretnénk több adatrekordot látni, akkor kell ezt a nézetet választani. Itt egymás alatt jelennek meg a rekordjaink.

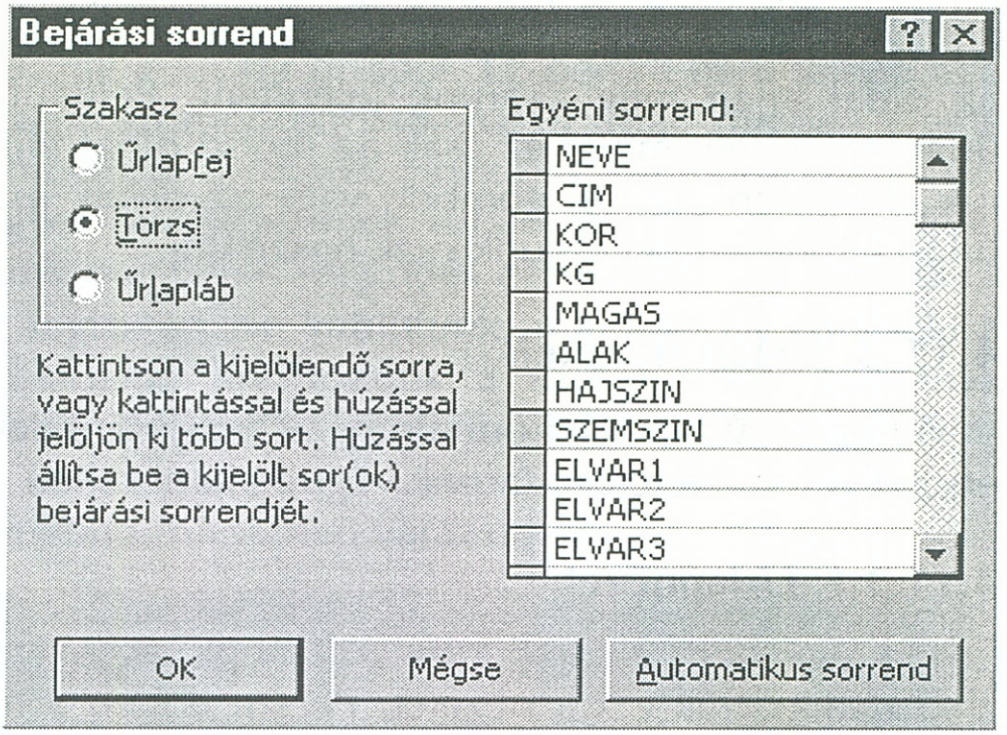
4.11.3.3. Tervező nézet

Az űrlap módosítására használjuk. Ebben a nézetben helyezhetjük el a különféle vezérlőelemeket, állíthatjuk be azok pozícióját, méretét, egyéb tulajdonságát. Itt formázhatjuk meg az egész űrlapunkat. Tervező nézetben az eszközkészletek közül bekapcsolható a tervezéshez szükséges vezérlőelem eszközkészlet.



Balról jobbra haladva az egyes elemek a következők: Objektum kijelölése, Varázsló indítása, Felirat, Beviteli mező, Vezérlőelem csoport,

Váltógomb, Választógomb, vagy más néven rádiógomb, Jelölőnégyzet, Kombi panel, Listapanel, Parancsgomb, Kép, Kötetlen objektumkeret, Kötött objektumkeret, Oldaltörés, Karton vezérlőelem, Segédűrlap segédjelentés, Vonal, Téglalap, További vezérlők.

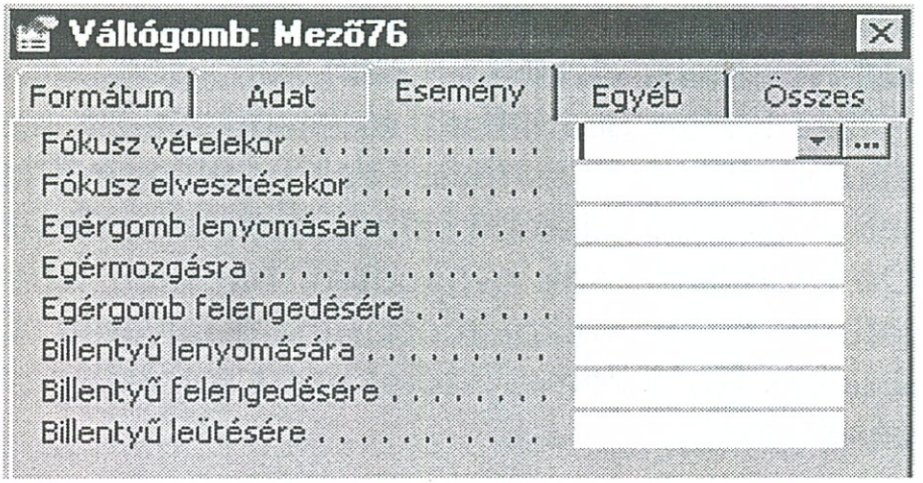


Az űrlap készítésekor az Access egy úgynevezett bejárási sorrendet határoz meg az űrlapon, ami azt jelenti, hogy az egyes vezérlőelemeket milyen sorrendben érhetjük el, például akkor, ha a Tab billentyűvel, vagy a kurzormozgató nyilakkal lépkedünk az űrlapon. Ezt a bejárási sorrendet változtathatjuk meg a Nézet menü Bejárási sorrend parancsával. A megjelenő párbeszédpanelon választhatjuk ki a beállítandó szakaszt, Űrlapfej, Törzs, Űrlapláb, majd választhatjuk az automatikus sorrend állító gombot is , de az egérrel megfogva az egyes vezérlőelemek nevét, vontatással átrendezhetjük a sorrendet. a megfelelőre.

Az űrlap egyéb tulajdonságait, ami az egész megjelenést befolyásolja az űrlap tervező nézetében jobb gombbal kattintva, az előugró menün található Tulajdonságok parancs hatására megjelenő Tulajdonság lapon állíthatjuk be.

4.11.4. Eseményvezérlő tulajdonságok

Az eseménytulajdonságok segítségével szabhatjuk meg egy bekövetkezett eseményre adandó választ. Ezt tehetjük az előre megszabott, az űrlap lehetőségei közül kiválasztva. De makrókat is indíthatunk az eseményekkel. A makrók készítése erősen haladóknak való, ezért ebben a könyvben nem foglalkozunk egyedi makrók készítésével. Az alábbiakban megtalálhatunk számtalan lehetőség közül néhány eseményt, amikre a lehetséges válaszokat csak be kell állítani az űrlapon a vezérlőelemek tulajdonságlapján.



Az ábra egy Váltógomb vezérlőelem tulajdonságlapja. Nézzük a beállítható lehetőségeket.

Fókusz vételekor, fókusz elvesztésekor: amikor az egyik vezérlőelemről átcsúszunk a másikra, akkor az egyik elem elveszti, a másik elem megszerzi a vezérlést, ezt hívjuk fókusz vételének, elvesztésének. Azt állíthatjuk be, hogy mi történjen a fenti estben.

Egérgomb lenyomás, felengedés, egérmozgatás. Megszabhatjuk külön-külön, hogy mi történjen ezekben az esetekben. Például az egérgomb lenyomásakor ugorjon a vezérlés az első rekordra, stb.

Billentyű lenyomására, felengedésére, leütésére milyen esemény következzen be.

A fenti kép egy Beviteli mező tulajdonságlapjának esemény regiszterfüle. Az itt található eseményre kattintva, a beállító mezőben megjelenik egy legördítő gomb, ami egy lista. Ebben a listában lévő események közül lehet választani, beállítani. A képen az a pillanat látható,

amikor az egér bal gombjának dupla kattintására bekövetkező eseményt választjuk ki.

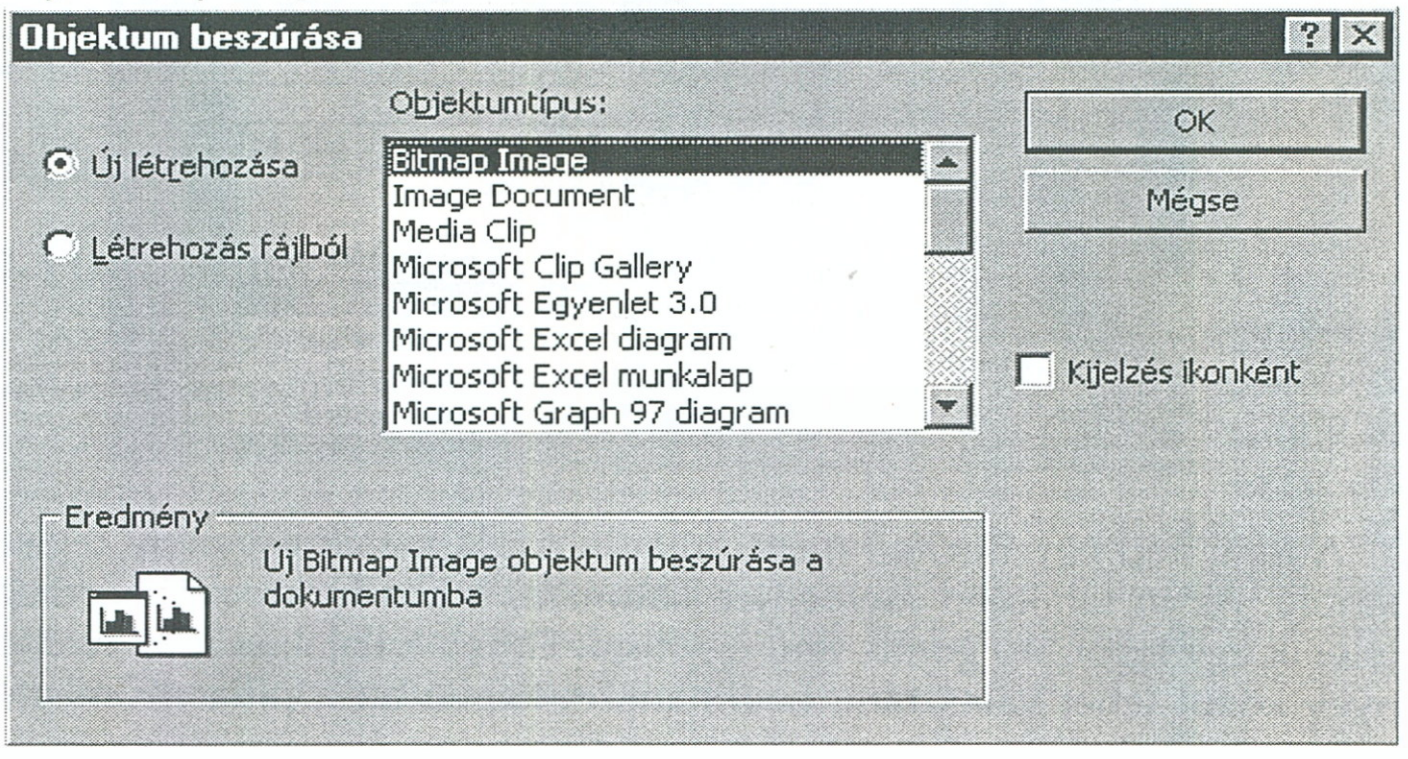
Számtalan beállítható esemény létezik az Access-ben, minden vezérlőelemre más és más. Ezek beállításáról külön könyvet lehetne írni. Jelenlegi terjedelembe sajnos nem fér bele az eseményeknek a leírása. De próbálkozzunk nyugodtan, baj nem lehet belőle, ha nem jót állítottunk be, bármikor módosítható ugyanitt. Ez egy jó játék, azonkívül érdekes is. Kellemes szórakozást, próbálkozást az események felfedezéséhez.

4.11.5. Objektumok keretei

Mint a többi Windows alatt futó programnál már megszoktuk, az Access is támogatja az OLE objektumok beágyazását, csatolását. Beilleszthetünk egy Paint képet, vagy egy Excel grafikont, stb. Ilyenkor egy objektumkeret vezérlőelem jön létre, ebben jelenik meg a csatolt, vagy beágyazott objektum. Az objektumkeretet elhelyezhetjük például egy űrlapon. A keret lehet kötött, vagy nem kötött. Kötött objektumkeretet akkor használunk, ha valamelyik táblánkból illesztünk be egy részletet, ekkor a megjelenő objektum függ a tábla tartalmától, ezért nevezzük kötöttnek. A nem kötött objektumkeretbe táblától független objektumot helyezhetünk el. Például a munkahelyünk logóját.

4.11.5.1. Objektumkeret létrehozása

Objektumkeret létrehozására van varázsló, ezt a Beszúrás menü Objektum parancsával aktivizálhatjuk.



A megjelenő párbeszédpanelon választhatjuk ki a beszúrandó objektum típusát. Választhatunk, hogy újat hozunk létre, vagy fájlból készítjük el. Lehetőségünk van még megszabni itt, hogy a beszúrt objektum ne látszódjon teljes nagyságában, csak ikonként jelenjen meg az űrlapon. Majd ha látni akarjuk, akkor rákattintva megnyílik.

4.11.6. Diagramok

Űrlap tervező nézetében elhelyezhetünk diagramot is az elegánsabb külső eléréséhez. A Beszúrás menü Diagram parancsával indíthatjuk a varázslót. Válaszolva a varázsló kérdéseire megadjuk, hogy melyik táblánkból, annak melyik mezőiből készítse, milyen típusú grafikon legyen, legyen e jelmagyarázat, és mi legyen a diagramunk címe, ezek után a varázsló elkészíti nekünk a diagramot. Megfelelően elrendezve az űrlapon lényegesen feldobhatjuk küllemét.

4.12. Lekérdezések

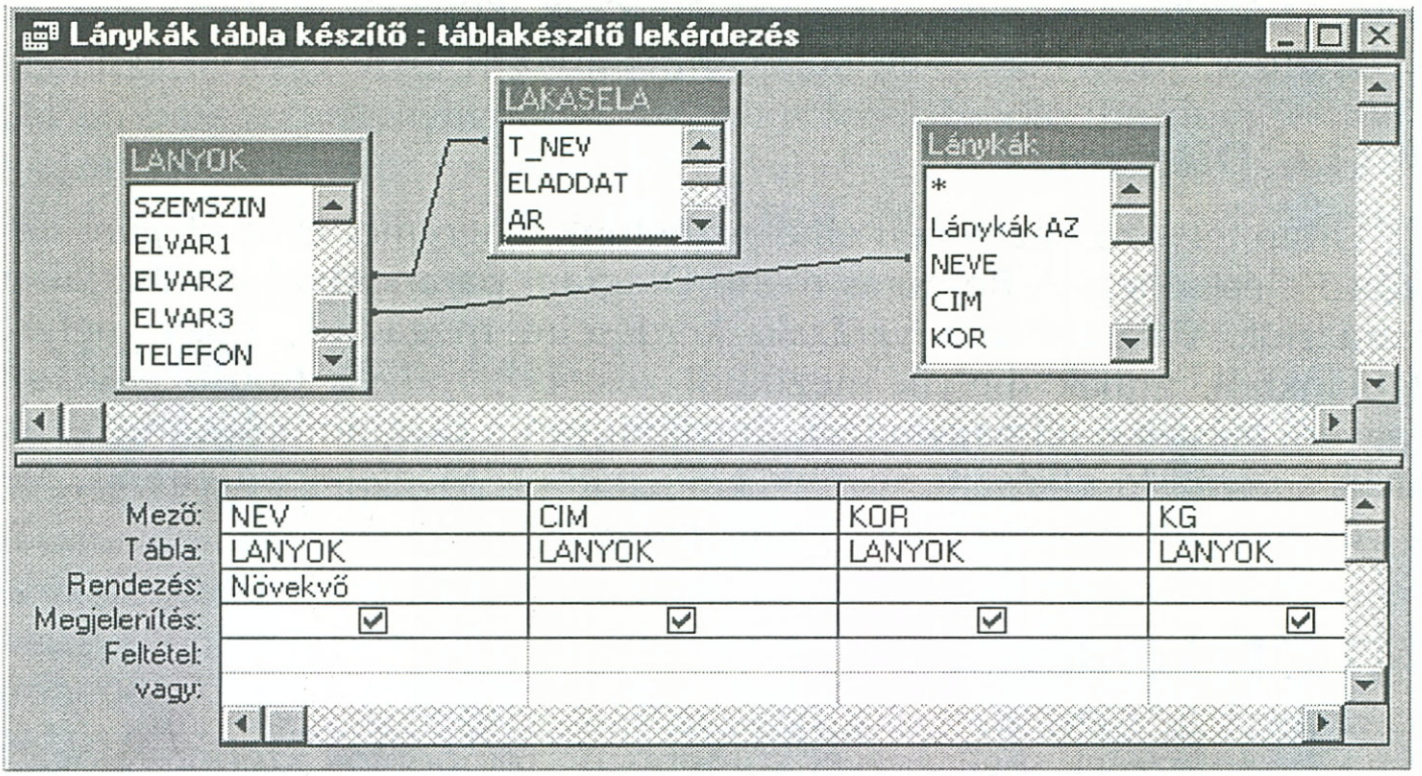
A lekérdezések segítségével különböző szempontok szerint válogathatjuk le adatainkat. A lekérdezések jól átgondolt megfogalmazott kérdések, melyeket grafikus, vagy az **SQL** (Structured Query Language) (Strukturált Lekérdező Nyelv) segítségével készítünk el, és tesszük fel. A lekérdezéskor összekapcsolhatunk több táblát, kikereshetünk rekordokat, amelyek csak bizonyos feltételeknek felelnek meg, sorba rendezhetjük adatainkat. A lekérdezéseket táblázatszerűen kapjuk meg. Ennek megfelelően akár új tábla is létrehozható vele. A lekérdezések általában jelentések, űrlapok alapjául szolgál. Azért használjuk ezt a közbülső lépcsőt, és nem közvetlenül a táblákat, mert így több táblát összekapcsolva összetettebb feladatokat is el tudunk végezni. Adataink megfelelő sorrendben való megtekintéséhez szintén a lekérdezéseket célszerű alkalmazni. A kereszttáblás lekérdezésekkel adatainkat elemezni is tudjuk. Az egyes lekérdezésekben megjelenő adatok módosíthatók, ezek a módosítások megjelennek a táblákban is, így adatokat törölhetünk, hozzáfűzhetünk a táblákhoz rekordokat, létrehozhatunk új táblákat. Lekérdezéseinknek három féle nézete van.

4.12.1. Lekérdezés nézetek

4.12.1.1. Tervező

Tervező nézetben készítjük el a tervet. Ez az ablak két részből áll. Felső részen látjuk a használt táblákat, azok mezőit, és a táblák közti

kapcsolatokat. Alsó részében helyezhetjük el a lekérdezéshez szükséges mezőket és azok jellemzőit, valamint beállíthatjuk az egyes oszlopokhoz tartozó feltételeket.

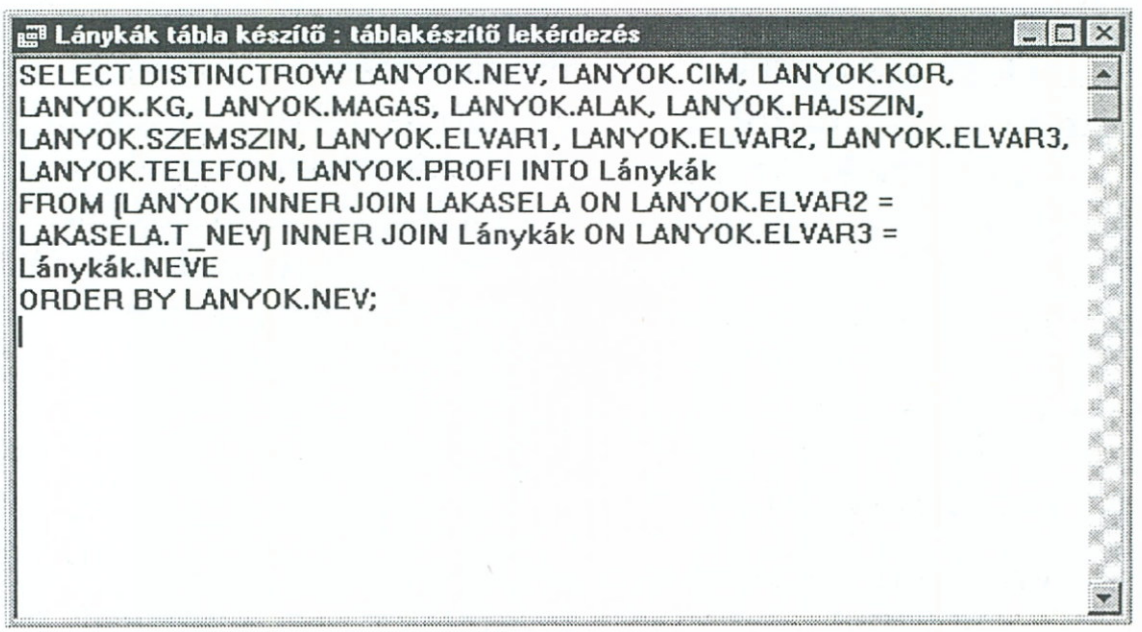


4.12.1.2. Adatlap

Az adatlap nézet tulajdonképpen megfelel a többi adatlap nézetnek, Oszlopokból és sorokból áll. A lekérdezés típusától függően módosíthatjuk adatainkat.

4.12.1.3. SQL

A létrehozott lekérdezések tulajdonképpen a rendszer által generált SQL kifejezések, melyeket mi készítünk el grafikus úton. Ezeket a rendszer által készített SQL kifejezéseket jeleníthetjük meg, módosít-



hatjuk ebben a nézetben. Profi alkalmazások készítésekor bizonyos feladatok csak ilyen SQL lekérdezésekkel valósíthatók meg.

4.12.2. Létrehozás

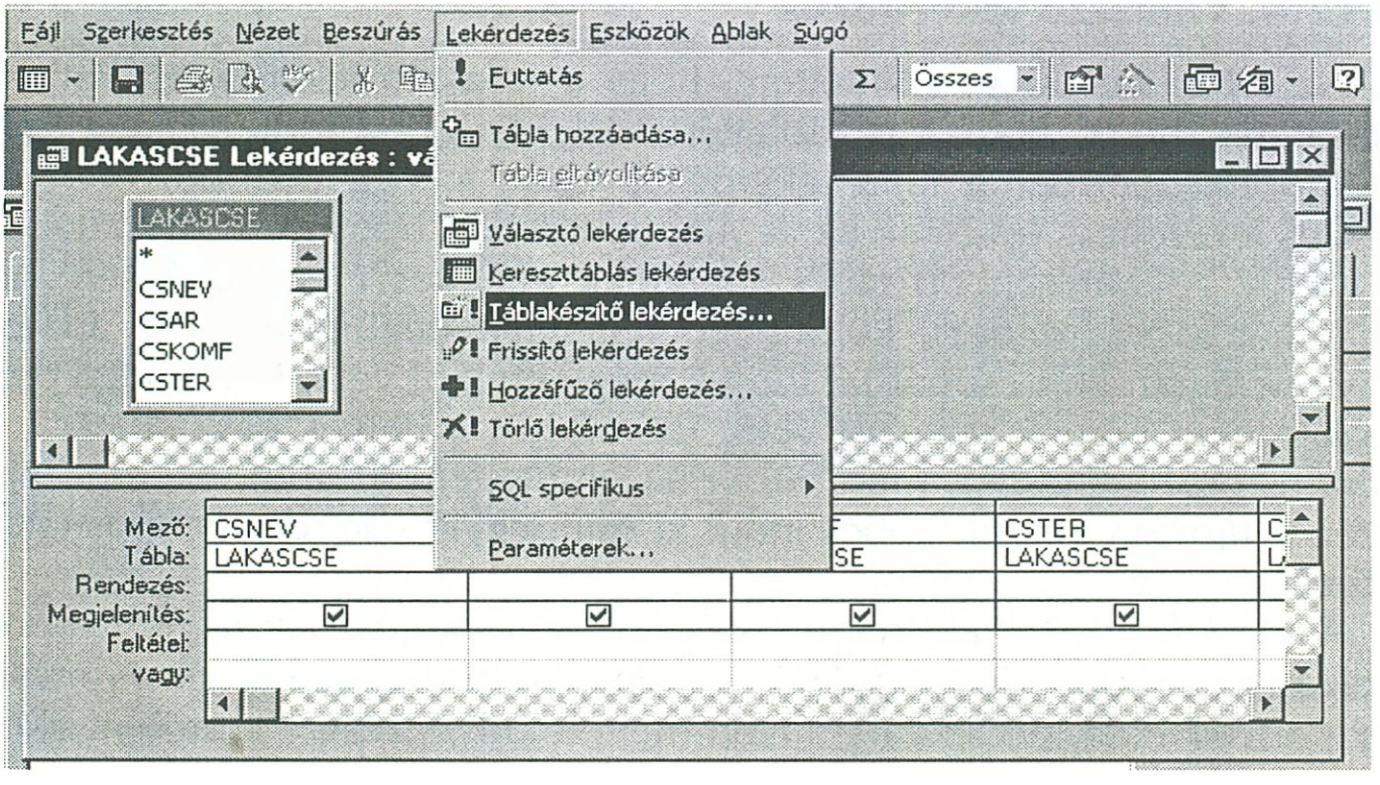
Új lekérdezés készítéséhez az Adatbázis ablakban kattintsunk a Lekérdezések regiszterfülre, majd az Új gombot lenyomva a megjelenő párbeszédablakban kiválaszthatjuk a megfelelő típusú Lekérdezés Varázslót. Itt választható Tervező nézet, Egyszerű lekérdezés Varázsló, Kereszttáblás lekérdezés Varázsló, Azonosakat kereső Varázsló és Nem egyezőket kereső Varázsló. A megfelelő Varázsló kiválasztása után az OK gombra kattinthatunk.

4.12.3. Táblakészítő lekérdezés

Egy olyan lekérdezés típus, amely nem jeleníti meg a lekérdezés eredményét, csak végrehajt egy műveletet, aminek az eredmény egy új tábla lesz. Az új tábla adattartalma a lekérdezésben megadott feltételeknek megfelelő rekordok lesznek. A tábla szerkezete, pedig megfelel a lekérdezés szerkezetének.

4.12.3.1. Létrehozása

Létrehozunk egy lekérdezést, majd átalakítjuk táblakészítő lekérdezésre. Tehát először az Adatbázis ablakban a Lekérdezés fülre kattintunk, majd az Új gombra. A kívánt Varázslóval létrehozzuk a lekérdezést, ez választó lekérdezés lesz. Az elkészült lekérdezést tervező né-



zetbe kapcsoljuk, erre a menüsoron megjelenik egy új menü, a Lekérdezések menü.

A Lekérdezés ablak alsó részén beállítjuk a kívánt feltételeket, bekapcsoljuk azokat a mezőket, (a Megjelenítés sorban) amelyeket át akarunk vinni az új táblába. Csak azok az oszlopok kerülnek át az új táblába, amelyeknek a Megjelenítés sorában, a jelölőnégyzetében betettük a pipajelet. Ezt egy kattintással tudjuk bekapcsolni. Ha mindezekkel megvagyunk, akkor legördítjük a Lekérdezés menüt, és a Táblakészítő lekérdezés parancsra kattintunk egyet. A megjelenő párbeszédpanelon megadjuk az új tábla nevét. Ha azt akarjuk, hogy egy másik adatbázisba készüljön el az új tábla, akkor ezt is megtehetjük, csak át kell kapcsolni a Rádiógombot a Másik adatbázisra, majd beírni a meglévő másik adatbázis nevét, és elérési útját. Ekkor a rendszer megkérdezi, hogy kívánjuk e tárolni a lekérdezést, természetesen kívánjuk, majd az új lekérdezésre duplán kattintva lefuttatjuk. Természetesen, ha tervező nézetben vagyunk, akkor a **Lekérdezés menü Futtatás** parancsát is választhatjuk, ebben az esetben is lefut a lekérdezés és elkészíti táblánkat. Futtatás után a tábla elkészül, persze mindegy, hogy ebben, vagy a másik adatbázisban. Amennyiben később újra akarjuk futtatni a lekérdezést, akkor ezt bármikor megtehetjük, csak tervezésre kell megnyitni, majd a fenti menüparancsot kell választanunk. Ha az előző képet megnézzük, akkor láthatjuk, hogy többféle lekérdezés típusra kattinthatunk, nézzük meg a legördülő menüt, a továbbiakban ezekről lesz szó.

4.12.4. Választó lekérdezés

Az előző példában említettem, hogy a választó lekérdezést kell elkészíteni ahhoz, hogy egy táblakészítő lekérdezést tudjunk készíteni. Tulajdonképpen ez a fajta lekérdezés típus a leggyakoribb, mert az összes lekérdezés típusnak ez az alapja. Ebből a típusból származtatjuk a többit. Alkalmas több tábla adatainak együttes megjelenítésére, az adatokat sorba rendezi, különböző feltételek beállításával az adatok egy meghatározott csoportját tudjuk megtekinteni vele, különböző összesítéseket tudunk végezni vele. Ügyeljünk a helyes használatára, a gondos összeállítására, mert mint már említettem a többi lekérdezés is ezen a típuson alapul. Ebből indul ki mindegyik lekérdezés típusnak a készítése.

A lekérdezéseket két nagy csoportba sorolhatjuk. Az egyik csoport dolga, az információk kigyűjtése, megjelenítése a táblákból. A másik csoport feladata, pedig valamilyen művelet elvégzése. Ez utóbbit akci-

ós lekérdezésnek is nevezhetjük. Ez a típus, a választó lekérdezés, valamint a kereszttáblás lekérdezés az első csoportba sorolhatók. Feladatuk az információk kigyűjtése. Mivel ezt a lekérdezést a varázslók hozzák létre, külön nem ismertetem a létrehozását. Egyébként az előző esetben le van írva.

4.12.5. Kereszttáblás lekérdezés

Ez a lekérdezés fajta is információ kigyűjtésre való. Tulajdonképpen egy táblában a kiszemelt oszlopban lévő adatokból, egy-egy oszlopot készít. Tehát az egyik táblában kiválasztunk egy oszlopot. Az oszlopban minden egyes egymástól különböző adatból a lekérdezésben egy önálló oszlopot készít. Ezekből lesznek az oszlopok fejlécei, vagyis az egyes oszlopok nevei. Egy kiszemelt oszlopból, pedig sorokat, rekordokat készít. Ezek lesznek az egyes sorok fejlécei, vagyis az első oszlopban ezek szerepelnek, mint a sorok neve. Az oszlopok, és sorok találkozásában lévő cellákban, pedig az értékek lesznek, ami természetesen egy harmadik kiszemelt oszlopból kerül beléjük. Hozzunk létre egy választó lekérdezést, nyissuk meg az elkészült lekérdezést tervező nézetben, majd a Lekérdezés menü Kereszttáblás lekérdezés parancsot válasszuk ki. A kereszttáblát tervező ablak Kereszttábla nevű sorába kattintva jelöljük ki az oszlopfejlécnek, sorfejlécnek, értéknek kiszemelt oszlopokat. Az értéknek kijelölt oszlopban valamilyen összegző függvény kell választanunk. A Rendezés sorban megszabhatjuk a rendezési sorrendet, és a Feltételek sorban bizonyos feltételekhez köthetjük az adatok megjelenítését. Egyébként a rendszer által ez a legkevésbé támogatott lekérdezés, emiatt elég nehézkes használni.

4.12.6. Frissítő lekérdezés

Akciós lekérdezés, mellyel egy tábla rekordjait lehet módosítani. Akkor használatos, mikor a rekordok meghatározott csoportjában lévő adatokat akarjuk megváltoztatni. Ezt a lekérdezést is választó lekérdezésből alakítjuk át. A tervező nézetben lévő lekérdezés alsó felében lévő úgynevezett QBE rácsban megjelenik a módosítás sor, ahova a módosító adatokat lehet beírni. A rekordok csoportját különböző feltételek megadásával tudjuk behatárolni.

4.12.7. Hozzáfűző lekérdezés

Egy meglévő táblához fűzhetünk új rekordokat ezzel a lekérdezéssel. Archiválásra kiválóan alkalmas. Azonban ennél a lekérdezésnél

figyelni kell arra, hogy a hozzáfűzendő rekordok egyedi azonosítója még véletlenül se szerepeljen a már meglévő táblában (amihez hozzáfűzzük). Hiszen egy táblában két rekordnak nem lehet ugyanaz az egyedi azonosítója (pl. sorszáma). Ezt a figyelést elkerülhetjük, ha az Access-re bízzuk az egyedi azonosító képzést azzal, hogy egy számláló mezőt veszünk fel a lekérdezésbe azonosítónak, és az eredeti azonosítót lehagyjuk a lekérdezésből. Mint tudjuk a számláló mezőnek az a jó tulajdonsága, hogy a rendszer növeli automatikusan az értékét. Ezért nem fordulhat elő, hogy két rekordnak azonos lesz a sorszáma. Az elkészítés menete a következő. Készítsük el a választó lekérdezést, felvéve azokat az oszlopokat, amelyeknek tartalmát hozzá akarjuk adni a táblához. Adjuk meg a feltételeket ahhoz, hogy mely rekordokat fűzzük a táblához. Majd a Lekérdezés menü Hozzáfűző lekérdezés parancsát választva a megjelenő párbeszédablakban megjelölhetjük azt a táblát, amelyikhez hozzáfűzünk.

4.12.8. Törlő lekérdezés

Ezzel a lekérdezéssel a beállított feltételeknek megfelelő rekordokat törölhetjük ki egy táblából. A lekérdezést az előzőkben leírtak szerint készíthetjük el, a Lekérdezés menü Törlő lekérdezés parancsát választva. Ha a lekérdezést átkapcsoljuk adatlap nézetbe, láthatjuk azokat a rekordokat, amelyek a beállított feltételeknek megfelelnek. Ezeket a rekordokat fogja törölni az Access.

4.12.9. SQL lekérdezés

Az egyes lekérdezések tulajdonképpen SQL kifejezések, csak grafikus felületen készítjük el azokat. Azonban vannak olyan feladatok, amelyeket csak SQL kifejezésekkel tudunk megírni, végrehajtani. Amennyiben magunk akarjuk megírni SQL nyelven a teljes lekérdezést, vagy módosítani szeretnénk a már elkészült lekérdezésünket, a Lekérdezés menü SQL specifikus parancsát kell választanunk. Ennek a parancsnak három lehetősége van. Az egyesítő, az átadó, és az adatdefiniáló. Ezt a három fajta lekérdezést írhatjuk meg SQL nyelven. Természetesen ehhez ismerni kell az SQL nyelv utasításait, szabályait is.

4.12.9.1. Egyesítő lekérdezés

Ez a fajta, több táblából kérdezhet le adatokat, amelyeket egyetlen lekérdezésben egyesítve jeleníthet meg. A lekérdezésnek legalább két SELECT utasítással kell rendelkeznie, ezekkel határozzuk meg a két

táblát, és azt, hogy melyik mezőket használjuk fel az egyes táblákból. A két SELECT utasításban a mezők számának azonosnak kell lenni, és az oszlopok fejlécében a neveket az első utasításban szereplő mezőnevek adják. Ezeket persze átírhatjuk az SQL nyelv megfelelő utasításával.

4.12.9.2. Átadó lekérdezés

Az átadó lekérdezésekkel ODBC adatbázisoknak küldhetünk utasításokat. Ezeket az adatbázisokat így SQL nyelven közvetlenül elérhetjük, csatolás nélkül.

4.12.9.3. Adatdefiniáló lekérdezés

Ennek a lekérdezés fajtának segítségével táblákat hozhatunk létre, módosíthatjuk, vagy akár törölhetjük azokat. Csak speciális esetekben lehet szükség ennek használatára, hiszen az Access-ben ezeket más módszerekkel könnyebb elvégezni.

A fenti lekérdezések komolyabb adatbázis kezelési ismereteket igényelnek, ezért nem részletezem alkalmazásukat.

Miután a kívánt lekérdezéseket elkészítettük, azokat le kell futtatni, hogy az akció megtörténjen, a lekérdezés végrehajtódjon. Mindezt a Lekérdezés menü Futtatás parancsával indíthatjuk el.

4.12.10. SQL parancsok

Ezeket a parancsokat három csoportba soroljuk.

- A táblák létrehozása, és törlése. Tehát az adatbázis létrehozása.
- A táblák adatainak bevitele, feltöltése, módosítása.
- A táblákból az adatok visszanyerése, kigyűjtése.

Az adatbázis ablak Lekérdezés regiszterfülre kattintunk, majd az Új gombbal előcsalogatjuk a Lekérdezés Varázslót. A varázsló ablakban a Tervező nézetet választjuk ki és az OK gombra kattintunk. A megjelenő Tábla hozzáadása párbeszédpanelt a bezár gombbal bezárjuk anélkül, hogy egy táblát is hozzáadtunk volna. Ezzel létrehoztunk egy üres választó lekérdezést, és megjelenik a menüsoron a Lekérdezés menüpont. Erre kattintva az SQL specifikus parancsot választjuk ki, majd a megjelenő almenüből az Adatdefiniáló lekérdezést választjuk. Ekkor a rendszer a választó lekérdezésünket átalakítja üres adatdefiniáló lekérdezésre, amelybe azután beírhatjuk a szükséges SQL kifejezéseket. Akkor most lássunk néhány egyszerűbb utasítást.

4.12.10.1. Magyarázat a jelölésekhez

NAGBETŰVEL írt szavakat úgy kell leírni ahogy a parancsban szerepe. Kötelező beírni a szót.
A *dőlt kisbetű,* a felhasználói változók.
[] Opcionális, elhagyható.
... adatismétlődés.
<> közé írtakat muszáj megadni, kötelező.
{} az ezek között szereplő elemek közül legalább egyet meg kell adni.
| választási lehetőségünk van az adatok közül.

4.12.11. CREATE TABLE

CREATE TABLE tábla név (mező1 típus [(méret)] [index1] [, mező2 típus [(méret)] [index2] [, ...]] [, többmezősindex [,...]])

Ezzel az utasítással új táblát hozunk létre. A táblanév a létrehozandó tábla neve. A mező1, mező2, a létrehozandó mezők neve. Legalább egy mezőt létre kell hozni. A típus a létrehozott mező adattípusa. A méret, Szöveges típusú mezőknél a karakterek maximális száma. Index a CONSTRAINT utasítással defininiált index (egymezős). Többmezősindex szintén a CONSTRAINT utasítással definiált index, de több mezős.

4.12.12. CONSTRAINT

Kapcsolatot létesíthetünk más táblával, a közös mezőjük segítségével.

4.12.12.1. Egymezős

COSTRAINT név {PRIMARY KEY | UNIQUE | REFERENCES külső tábla [(külső mező1, külső mező2)]}

4.12.12.2. Többmezős

CONSTRAINT név { PRIMARY KEY (elsődleges1 [,elsődleges2 [,...]]) |
UNIQUE (egyedikulcs1 [,gyedikulcs2 [,...]]) |
FOREIGN KEY (hivatkozás1 [, hivatkozás2 [,...]])
REFERENCES külsőtábla [(külsőmező1 [, külsőmező2 [,...]])]}

A *név* a létrehozott index neve, az *elsődlegesek* az elsődleges kulcsmezők neve, *egyedikulcsok* az egyedi kulcsok mezőnevei, *külsőtábla* a *külsőmezőket* tartalmazó tábla neve. Primary key, egy, vagy

több mezőt elsődleges kulcsnak jelölhetünk. Foreign key, egy, vagy több mezőt külső kulcstulajdonsággá alakíthatunk.

4.12.13. CREATE INDEX

CREATE [UNIQUE] INDEX indexnév ON tábla (mezők ASC | DESC) [WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

Az Access-ben ezzel az utasítással hozhatunk létre új indexeket.

4.12.14. DROP

Adatbázisból táblát, vagy létező indexeket törölhetünk.
DROP {TABLE tábla név | INDEX index név ON tábla név.

4.12.15. ALTER TABLE

ALTER TABLE tábla név {ADD {[COLUMN] mező típus [(méret)] [CONSTRAINT index név] | CONSTRAINT többmezősindex } | DROP {COLUMN mező név | CONSTRAINT index név } }

Ez az utasítás már létező táblák megváltoztatására való. ADD paraméterrel hozzáadhatunk, DROP paraméterrel törölhetünk mezőt a táblából.

4.12.16. DELETE

Ezzel az utasítással a WHERE feltételben megadott rekordokat, ha WHERE kulcsszó nincs, akkor az egész táblát törölhetjük. Egy törlő lekérdezést hozunk vele létre.
DELETE [tábla.*] FROM tábla kifejezés WHERE feltétel.

Törlés előtt ellenőrizzük a törlendő sorokat a SELECT-tel, mert nem állíthatók vissza a rekordok.

4.12.17. INSERT INTO

Hozzáfűző lekérdezést készít, sorokat illeszt be a táblába.
INSERT INTO tábla név [(oszlopnevek)] VALUES (oszlopértékek)

4.12.18. UPDATE

UPDATE tábla név SET {oszlop név = {kifejezés | NULL}} WHERE feltétel

Egy tábla meglévő rekordjait módosíthatjuk vele. Ez egy módosító lekérdezés, és három részre oszthatjuk. Egyszerre több rekord is módosítható, amelyik kielégiti a WHERE feltételt. A WHERE elhagyható.

4.12.19. SELECT

A legfontosabb SQL utasítás. Az többi utasítás csak ezt készíti elő. A meglévő táblákban az adatok részleges, teljes, rendezett, rendezetlen lekérdezésére, kiválasztására használjuk.
SELECT [szűkítő feltétel] { * | tábla.* [tábla.]mező1 [AS másnév1] [,[tábla.]mező2 [AS másnév2] [,...]]}
FROM táblakifejezés [,...] [IN külsőadatbázis]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
WITH OWNERACCESS OPTION]

A szűkítő feltétel szolgálja azt a célt, hogy az ismétlődő adatokat, sorokat hagyhatjuk el. A tábla, a Mezőt specifikáló táblanév. Azonos nevű mezőket úgy különböztetünk meg, hogy az egyes mezők neve elé a tábla nevét írjuk, közéjük pontot téve. Mező1, mező2, azok a mezőnevek, amelyeknek az adatait látni kívánjuk. Másnév1, másnév2, az egyes mezék neveinek megváltoztatására való. Táblakifejezés, az adatokat tartalmazó táblák neve. A külsőadatbázis, a táblakifejezés tábláit tartalmazó adatbázis neve.

Amennyiben a mezőnév valamilyen elválasztó jelet tartalmaz, akkor a mezőnevet szögletes zárójelbe kell rakni. Ne használjunk a nevekben elválasztó jelet. Ez csak szokás kérdése. Ha nem a mező nevét akarjuk feliratként alkalmazni, akkor az AS után kell másnévként megadni.

4.12.20. SELECT FROM

SELECT <mezők> FROM <táblanév> [IN<külsőadatbázis>]
A From, a Select utasításban résztvevő táblákat határozza meg.

4.12.21. SELECT WHERE

A Where szóval adjuk meg azon feltételeket, mely alapján a Select rekordokat válogat le. Amennyiben elhagyjuk, akkor az összes sor figyelembe lesz véve. A Whewre maximum 40 tagú logikai operátorokkal összekapcsolt kifejezés. A Group By ... HAVING helyettesítheti. A feltétel csak amerikai formátumú lehet. Ha van From, akkor a Whernek utána kell következnie. A Where után írt feltételek lehetnek:

⌦ Relációs jelekkel képzett egyszerű feltételek.

⌦ BETWEEN / WHERE kif1 [NOT] BETWEEN kif2 AND kif3 / Azokat a rekordokat válogatja ki, amelyekben a kif1-ben foglaltak beleesnek a kif2 és kif3 közé.

⌦ AND, OR / WHERE felt1 [AND | OR felt2 [...]] / Normál ÉS, VAGY feltétel vizsgálat.

⌦ LIKE / WHERE [NOT] mező [NOT] LIKE / hasonlítandó Összehasonlítja a mezőben lévő karaktereket a hasonlítandóval, ezekben lehetnek joker karakterek is. ? egy karakter, # számkarakter, * karakterfűzér részlet.

4.12.22. COUNT

E függvénnyel megszámolhatjuk egy lekérdezés, jelentés, vagy űrlap rekordjainak számát. COUNT (kifejezés).

4.12.23. MIN, MAX

A minimum érték, vagy maximum érték kiszámítására használhatjuk a kiválasztott mezőben.

4.12.24. SUM

A lekérdezések, jelentések, vagy űrlapok kiválasztott mezőinek összegzésére használjuk.

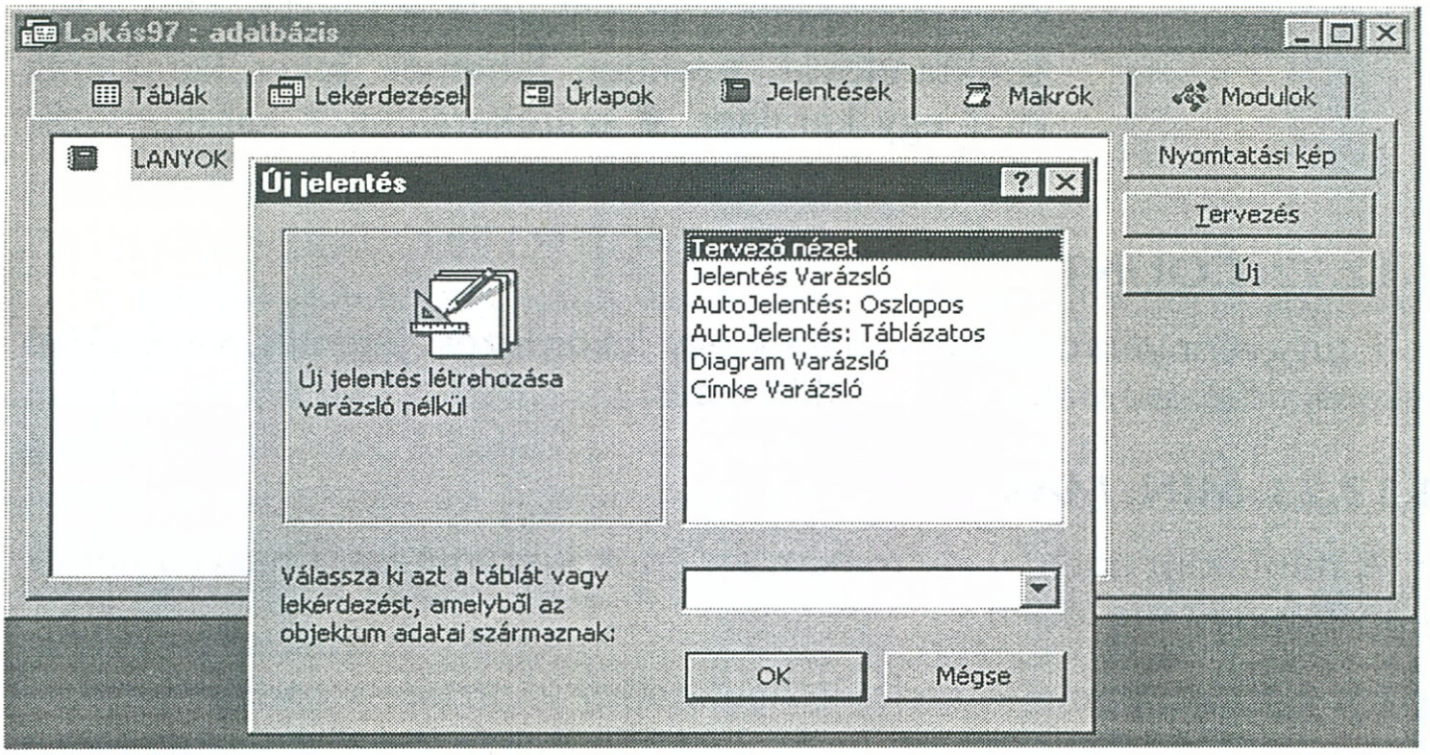
Fenti SQL utasítások csak minimális kivonata a teljes SQL utasításoknak. E könyvnek nem célja az SQL nyelv használatának bemutatása, tanítása, azt egy ebből a témakörből íródott szakkönyvbál célszerű elsajátítani, de azért akit érdekel, a súgó segítségével megpróbálkozhat néhány utasítással.

4.13. Jelentések

Adataink megjelenítését szolgálja, akárcsak a lekérdezések, vagy űrlapok, a felhasználás célja ami eldönti, hogy melyiket használjuk. A jelentéseket elsősorban adatainkat papíron történő megjelenítésre használjuk. A nyomtatást rendkívül jól segítő szolgáltatásokkal rendelkeznek. Esztétikus, könnyedén számolhatunk rész és végösszegeket, csoportosíthatjuk adatainkat. Származhatnak táblából, vagy lekérdezésekből. Egyszerre több táblából álló összetett jelentés készítésénél előbb elkészítünk egy lekérdezést, majd ebből a jelentést.

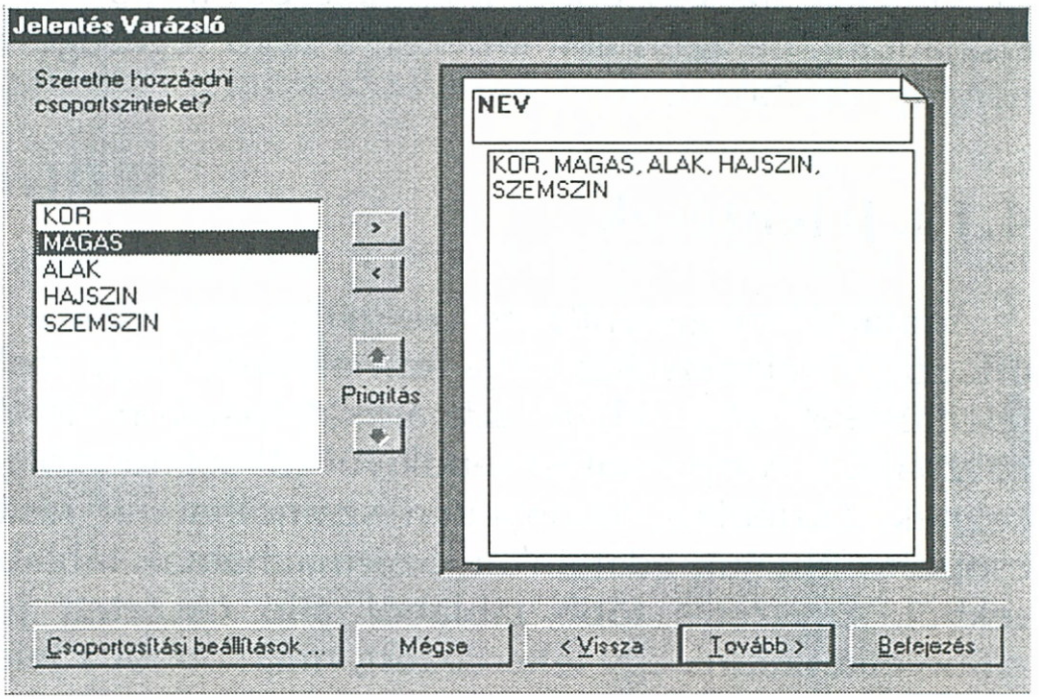
4.13.1. Létrehozás

A jelentéseket létrehozhatjuk Jelentés Varázslóval, és manuálisan is. A varázsló használata a javasolt, hiszen a munka dandárját elvégzi helyettünk. Akkor is a varázsló használata javasolt, ha egyéni jelentésre van szükségünk, a jelentés elkészülte után sokkal könnyebb módosítani, mint létrehozni.



Az Adatbázis ablakban a Jelentés regiszterfülre kattintunk, majd az Új gombra. A megjelenő ablakban választhatunk megfelelő varázslót.

Választhatunk tervező nézetet, ekkor magunknak kell összeállítani a jelentést. De választhatunk a fenti varázslók közül is tetszés szerint, például egyszerű Jelentés Varázslót, Oszlopos, Táblázatos, auto jelentést, Diagram Varázslót, Címke Varázslót. Kiválasztjuk a megfelelő varázslót, majd alul a legördíthető listaablakban kiválasztjuk a kívánt táblát, és az

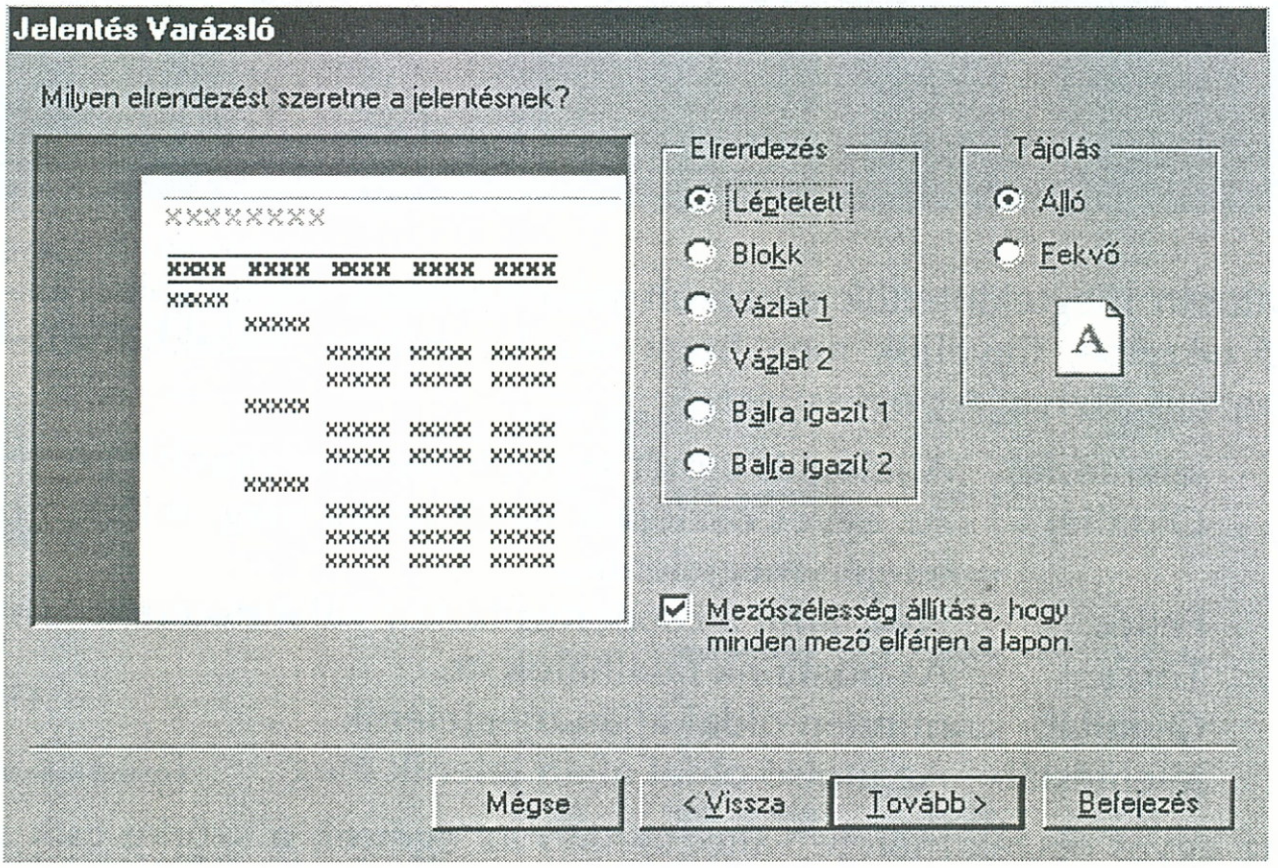


OK gombra kattintva elindul a varázsló. Az első lépésben meghatározzuk, hogy melyik mezőkre van szükségünk a különböző táblákból. Több táblából is összeszedhetjük mezőinket, de a táblák között kapcsolatnak kell lenni. A következő lépésben különböző csoportosításokat állíthatunk be a kiválasztott mezőnevekkel.

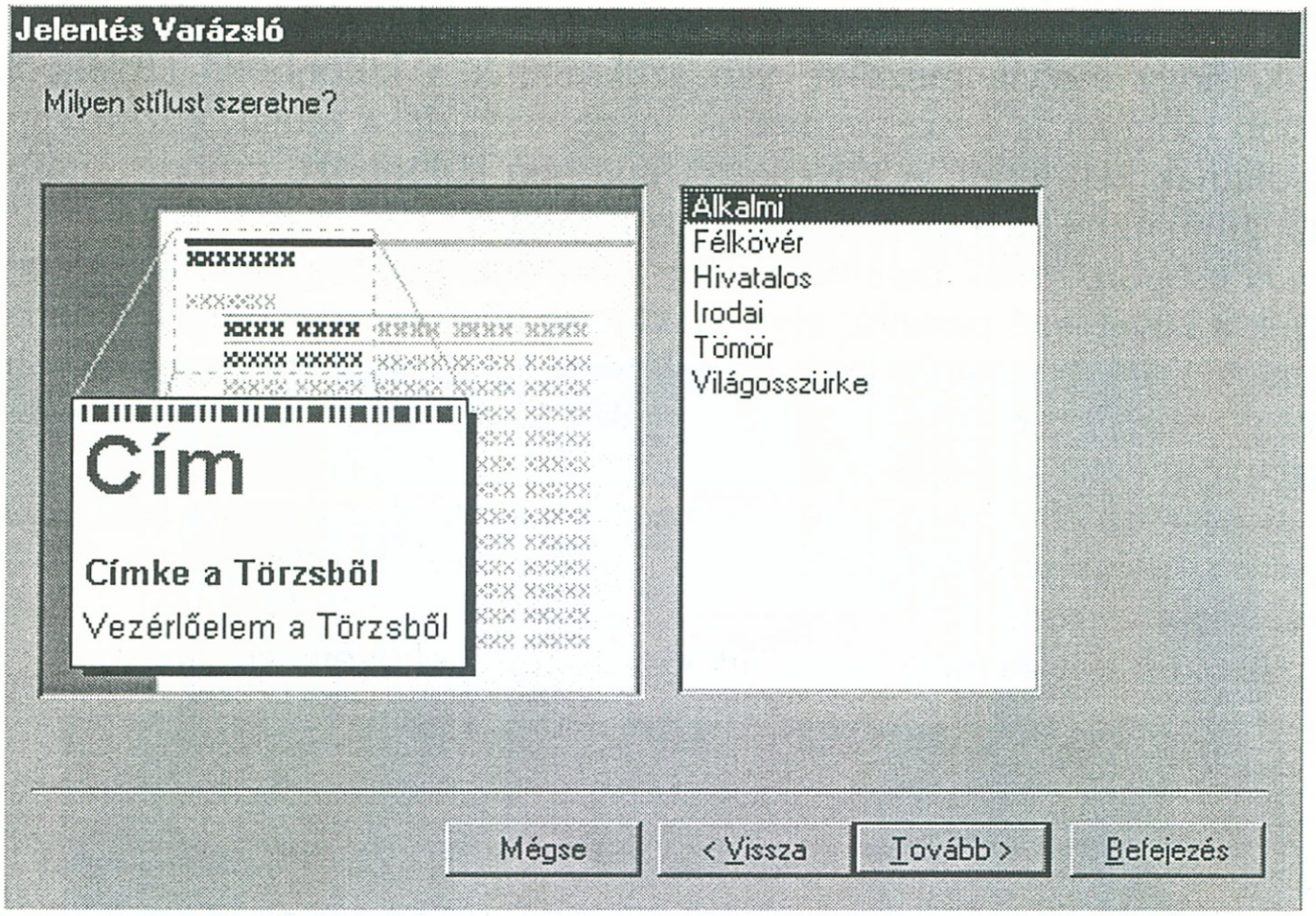
A csoportszintek beállítása után a Tovább gombra kattintva léphetünk a következő panelra, ahol maximum négy mezőre meghatározhatunk növekvő, vagy csökkenő rendezést. Találunk itt még egy összesítés gombot, ezen összegző függvényeket alkalmazhatunk mezőink összegzésére, ha kívánjuk.

Következő lépésben az egyes mezők helyét, elrendezését, igazítását határozhatjuk meg a jelentésen, valamint a papír tájolását, hogy fekvő, vagy álló legyen.

Ezt követi a jelentés stílusának kiválasztása a következő lépésben.



Ez lehet alkalmi, mint a képünkön, félkövér, hivatalos, irodai, tömör, vagy világosszürke. Ezt ízlésünk szabja meg. A tovább gombra kattintva eljutunk a befejező panelra, ahol a jelentésünk címét adhatjuk meg. A Jelentés nyomtatási képe, vagy a Jelentésterv módosítása rádiógomb kiválasztása után a Befejezés gombra kattinthatunk. Megtekinthetjük munkánkat, vagy előbb módosítjuk az elrendezést, stb. A befejezés gombra a többi panelon is kattinthatunk, ekkor is elkészül jelentésünk, csak az alapértelmezés szerint.

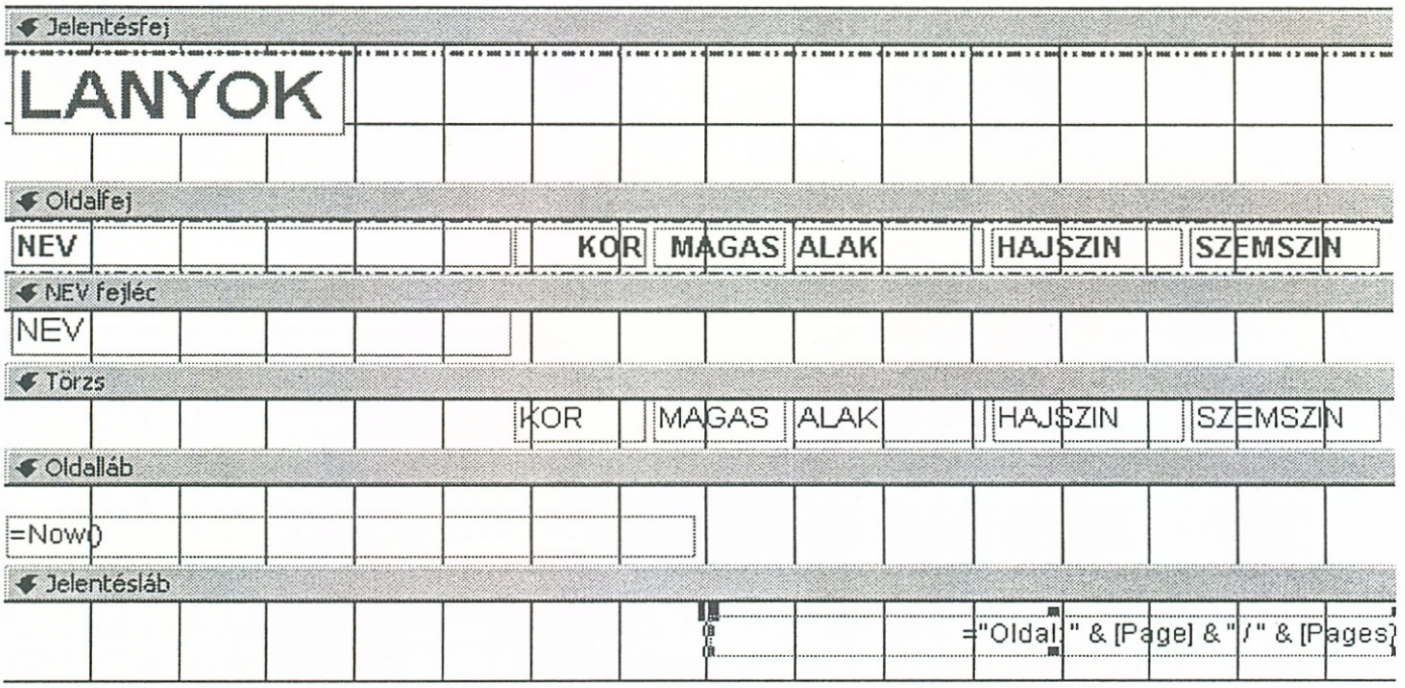


Miután megtekintettük elkészült jelentésünket, átkapcsolhatjuk tervező nézetbe, figyeljük meg hogy néz ki, milyen részekből áll. Hat részre van osztva. Az egyes részek:

- Jelentésfej — A jelentés címét szoktuk ide elhelyezni.
- Oldalfej — Az egyes oszlopok nevét adjuk meg, minden oldal tetején megjelenik
- Fejléc — Név, amit a csoportosításkor választottunk.
- Törzs — Az adataink találhatok itt.
- Oldalláb — minden oldal alján megjelenik.
- Jelentésláb — Az utolsó oldal alján jelenik meg.

Az egyes részek mérete változtatható az egérrel, a szürke csík alsó, vagy felső szélét megfogva függőlegesen állíthatjuk az adott rész méretét. A Beillesztett mezőket, címkéket formázhatjuk, vonszolással átméretezhetjük, törölhetjük, és újakat illeszthetünk be. Tehát teljesen átszabhatjuk a megrendelő kívánságának megfelelően.

Az egyes szakaszok tulajdonságait is állíthatjuk. A tervező nézetben a kívánt szakasz szürke csíkjára duplán kattintva bejön a tulajdonságlapja. Ezeken több regiszterfül is található számtalan tulajdonsággal.



Ezeket nem sorolom fel a nagy számuk miatt, de bátran állítgathatjuk a különböző elemeket. A súgóban részletes leírás található ezek állításához. A már meglévő jelentésünket bármikor megváltoztathatjuk, ha tervező nézetbe kapcsolunk. A formázáson, a stíluson pedig a legegyszerűbb változtatni, ha a Formátum menü Automatikus formázás parancsot választjuk.

LANYOK

NEV	KOR	MAGAS	ALAK	HAJSZIN	SZEMSZIN
Adácsy Júlia	28	165	normál	szőke	
Ágnes	32	164	vékony	fekete	
Alhusznisreh Marcella	38	168	molett	fekete	barna
Aliz	19	170		fekete	barna
Almási Elekné	38	165	molett	vörös	kék
Ambrusz Gyuláné	43	174	normál	barna	barna
Andrea	23	172	arányos	fekete	fekete
Anikó	22	174	arányos	fekete	fekete

Ez a leírás az Access lehetőségeit nem tárja fel teljes egészében, hiszen azt csak több száz oldal terjedelemben lehetne leírni. Vannak olyan részei, amelyekkel egyáltalán nem foglalkoztunk. A leírtak csak kivonatnak tekinthetők. Próbálkozzunk bátran, fedezzük fel működését, remélem elegendő alapot nyújt leírásunk ehhez.

Jegyzet

Informatikai füzetek

A sorozat kötetei:

1. Alapismeretek
2. Operációs rendszerek
3. Kiegészítő ismeretek
4. Szövegszerkesztés
5. Táblázatkezelés
6. Adatbázis-kezelés
7. Programozás

Sorozatszerkesztő:
Bártfai Barnabás

ISSN 1418-8791

ADATBÁZIS-KEZELÉS

ISBN 963 03 5285 0

BBS-E Számítástechnikai és Könyvkiadó Betéti Társaság

Könyvsorozatunk segítségével alapszintről elindulva, a számítógép megvásárlásától és első bekapcsolásától kezdve a DOS parancsain és üzenetein át, a különböző felhasználói programok alkalmazásáig mindent könnyedén megtanulhat.

A leírtak tanfolyamok tapasztalataira épülve, gyakorlati példákat bemutatva segítenek elsajátítani a számítógép kezelését, megismerni részegységeinek használatát olymódon, hogy azt Ön a mindennapi munkájában is kamatoztathassa.

Kiadványaink a nagy sikerű Hogyan használjam? Című könyv alapján készültek, teljesen kezdő felhasználókhoz szólnak, s önálló tanulásnál is jól használhatóak. Továbbra is fontosnak tartottuk, hogy ne azt mutassuk meg, hogy egy adott programfunkció mire való, hanem azt, hogy egy adott feladatot milyen módon tudunk megoldani.