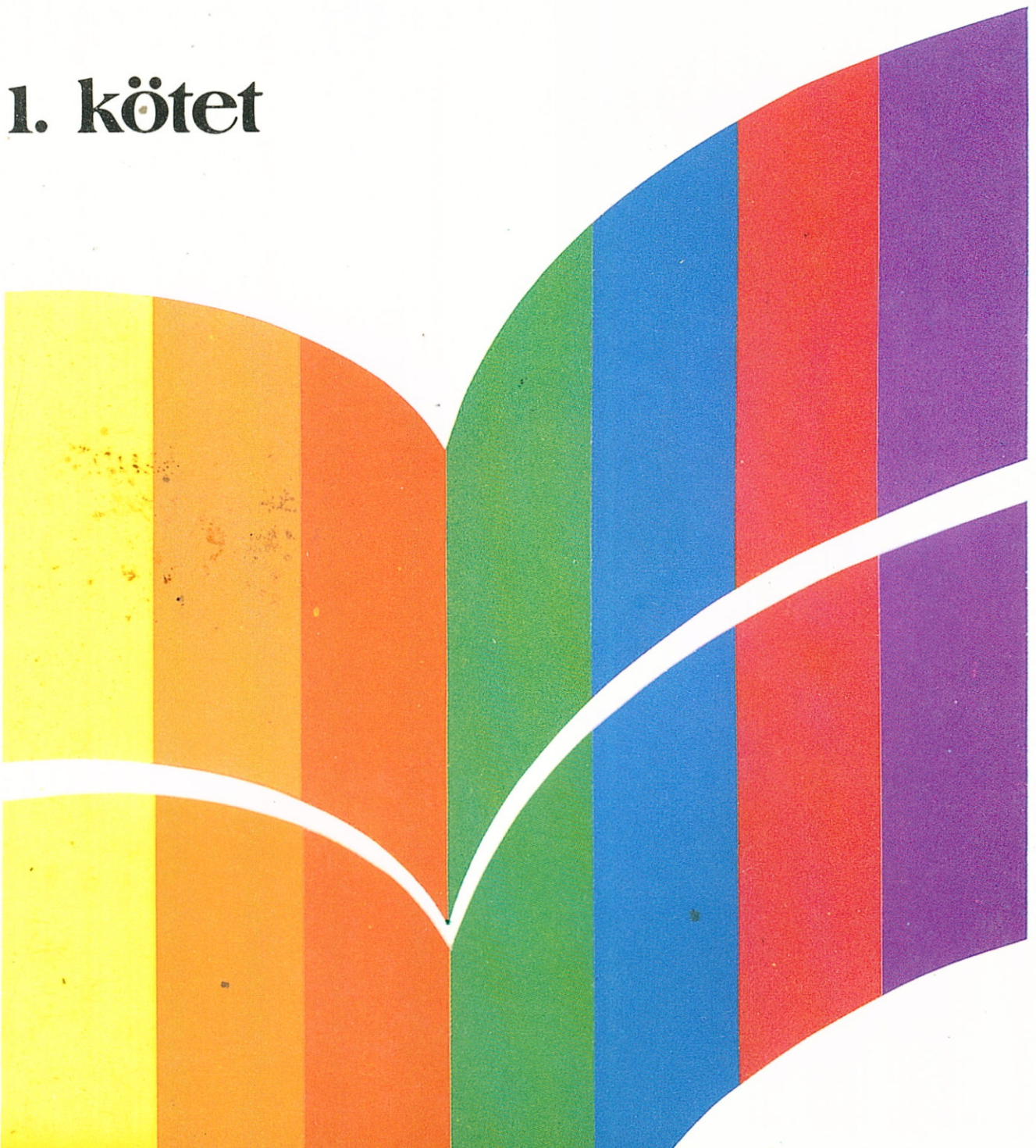
SZLOVÁK GÁBOR • TÓTH TIBOR • KŐRI GÁBOR

Adatbázis-kezelés, programozás dBASE IV-ben

1. kötet



A dBASE IV adatbáziskezelő rendszer programozása

I. kötet

LSI ATSZ – aPLUS Kft, Budapest, 1989

Szerzők:

Első kézirat:	Kőri Gábor	2,6,E,F,G,I,J,K,M fejezetek
	Molnár M. Attila	Előszó, 1,2,A,B fejezetek
	Szlovák Gábor	4,5,8,C,L fejezetek
	Tóth Tibor	3,7,D,H fejezetek
Végső változat:	Molnár M. Attila	
	Tóth Tibor	
Szerkesztő:	Molnár M. Attila	
Lektorálta:	Apor György	

ISBN: 963-592-973-0 I. kötet
963-592-974-9 II. kötet
963-592-975-7 összkötet

Megjelent az **LSI ATSZ** és az **Intersoft Kft.** közös kiadásában.
Felelős kiadó: Dr. Kovács Magda és Apor György
Témafelelős: Salgóné Sziklai Klára és Szűcs Ernő
A kézirat és szedés az **Intersoft Kft.** műhelyében készült.
Műszaki szerkesztő: Molnár M. Attila
A kiadványt az **InterSoft Kft.** menedzselte és finanszírozta.

Készült a Nógrád megyei Nyomdaipari Vállalat balassagyarmati üzemében.
Felelős vezető: Kelemen Gábor igazgató

Tartalomjegyzék

A függelék tartalomjegyzéke

Előszó

Ezzel a könyvvel, amely meglehetősen gyorsan készült, igyekeztünk egy hiányt kielégíteni. A világszerte, így hazánkban is elterjedt dBASE programcsalád megismeréséhez, használatához eddig kevés igazán jól használható hazai kiadványt ismerünk, a legújabb változathoz, a dBASE IV-hez pedig egyet sem.

Úgy gondoltuk, ezt az igényt gyorsan ki kell elégíteni, és lehetőséget kell teremteni azoknak, akik kezdőként szeretnének megismerkedni a dBASE IV adatbáziskezelővel, de a teljeskörü használathoz is álljanak rendelkezésre a fontos információk.

Könyvünk szerkezete ezért az alábbiak szerint épül fel:

Az első rész a megismerkedéshez, áttekintéshez szükséges, bevezető jellegű információkat taglalja:

- az adatbáziskezelőkről álatalában,
- az adatok szerkezetéről, azok meghatározásáról,
- programozási alapismeretekről,

majd részletesebben a dBASE IV főbb elemeiről:

- a Control Center-ről
- az Alkalmazás-generátorról
- az SQL-ről
- a hálózati alkalmazásról és adatvédelemről, valamint a
- dBASE III PLUS és dBASE IV kapcsolatáról, eltéréseiről.

A második rész, a függelék tömör formában

- néhány alapfogalom magyarázatát
- a jelölésrendszert és a használt szimbólumokat,
- utasítások és függvények leírását,
- hibaüzeneteket és
- egyéb kiegészítő információkat (pl: fájlok adatszerkezete, technikai jellemzők, a dBASE IV konfigurálása, stb)

tartalmazza.

Törekedtünk arra, hogy a könyv segítségével kezdők is megpróbálkozhassanak a dBASE IV-gyel, ezért minden lényeges részt rövid bevezetővel kezdtünk. Ezekben tisztáztunk néhány alapfogalmat is, amelyeket a tárgymutatóba kigyűjtöttünk. Néhány olyan terminust a függelék tartalmaz, amelyeknek magyarázata nem nagyon illett a fejezetekbe. Ebben sem törekedtünk a teljességre, mert úgy gondol-

tuk, hogy ezeket a fogalmakat sok szakkönyvben bővebben és pontosabban megtalálhatja az Olvasó. Az alapfogalmak elsősorban emlékeztetőül állnak azért, mert az a tapasztalatunk, hogy olykor komolyabb gyakorlattal rendelkező programozók sincsenek tisztában néhányukkal, ami azt okozhatja, hogy munkájuk során megmagyarázhatatlanul ismétlődő problémákkal találkoznak szembe, amelyeknek megoldása az ismeretek híján inkább próbálkozással történik és ez rendszerint nem vezet jóra. Rossz hasonlattal élve:
Aki nincs tisztában a ház alapozásának és statikájának ismereteivel, az lehet hogy tetszetős házat is tud építeni, de sohasem tudjuk biztosan, hogy mikor dől össze.

Javasoljuk tehát a gyakorlottabaknak is, legalább fussák át ezeket a részeket. Nem lennénk meglepődve, ha néhány új dologra is bukkannának.

A könyv megértéséhez tehát nincs szükség komolyabb számítástechnikai felkészültségre, de feltételeztük, hogy az Olvasó az operációs rendszer, a DOS alapvető fogalmaival tisztában van. Szükség van viszont egy működő dBASE IV rendszerre, de nem nélkülözhető egy kis angol szótár sem azoknak, akik az angol számítástechnikai terminológiát egyáltalán nem ismerik. Annak ellenére így van ez, hogy minden fontos kulcsszót megmagyaráztunk a terjedelem-engedte korlátok között, illetve arról fordítást adtunk.

Akik mélyebben ismerni akarják az utasítások működését, természetesen nem nélkülözhetik az eredeti kézikönyvet. Mindenesetre búzunk abban, hogy könyvünk elolvasásával és az abban írtak gyakorlati kipróbálásával kezdők is sikerrel használatba vehetik a dBASE IV-et, de az meglehetősen bonyolult alkalmazások eredményes elkészítéséhez is elégséges és jól használható információkat nyújt.

Előre kell bocsátanunk, hogy nem volt időnk minden utasítást alaposan tesztelni, de mentségül szolgáljon, hogy ezt még az ASHTON - TATE sem tudta teljeskörüen elvégezni. Ezért fel kell hívnunk a figyelmet arra, hogy az első dBASE IV változatban nem működik minden tökéletesen, úgyhogy akiknek módja van az eredeti kézikönyveket tanulmányozni, azokat is érhetik komoly meglepetések. Természetesen ezzel még véletlenül sem azt akarjuk mondani, hogy a dBASE IV nem használható jól, hanem inkább azokra a lassan megdönthetetlennek látszó igazságokra kívánjuk a figyelmet felhívni, miszerint:

- hibátlan szoftver még nem született a világon,
- minél bonyolultabb, több szolgáltatással rendelkezik egy szoftver, annál több az előforduló hibák száma,
- nem készült még olyan programtermék, amelynek dokumentációja annak minden tulajdonságát pontosan és hűen leírta volna.

Hangsúlyoznunk kell azt is, hogy könyvünk nem az eredeti kézikönyvek fordítása — nem is lehet az, hiszen azok terjedelmének csak töredéke —, hanem elsősorban bevezető jellegű, a megismerkedést, a tanulást szolgálja, de minden, a teljes használathoz fontos információt is összegyűjtöttünk benne, sőt gyakorlati

tapasztalatunkból megismert néhány olyan adalékot is szerepeltetünk, amely az eredetikben nem található.

Reméljük sikerült célunkat elérni, és hasznos segítséget adunk a dBASE IV alkalmazásához.

Kívánunk ehhez sok sikert minden Olvasónknak.

A Kiadók és a Szerzők

1. Bevezető

"dBASE IV". Nem sokat tévedünk, ha arra tippelünk, hogy ez a név a "database" angol szóból származik. Hiába lapozunk fel egy régebbi (1971) angol szótárat, nyomát sem leljük, és ez csak részben a szótár hibája. Az "adatbázis" fogalom – amelyet napjainkban már az általános iskolások is ismernek – valóban a legújabb kor és ezen belül a számítástechnika és informatika robbanásszerű fejlődésének terméke. Magát a szót a köznapi életben is egyre gyakrabban hallhatjuk, azonban – mint az sok új fogalommal előfordul – nem tudjuk pontosan, hogy mit takar. Ez nem is csoda, hiszen sokféle értelmezésben használják, és a tartalma is állandóan változik, fejlődik.

Anélkül, hogy hosszasan fejtegetnénk alapfogalmakat, vagy részletes történelmi áttekintést adnánk e szakterületről, túl kell esnünk néhány olyan fogalom és összefüggés rövid bemutatásán, amely itt és most a dBASE IV megismeréséhez, használatához elengedhetetlen. Mindazok akik járatosak az adatbáziskezelés területén nyugodtan átlapozhatják a bevezető oldalait. Azok, akik tudományosabb igényű, részletesebb magyarázatot kívánnak, több munka közül is választhatnak. Fellapozhatják akár Halassy Béla régi könyvét is, amely az "Adatbáziskezelés alapvető kérdései" címmel 1978-ban jelent meg a SZÁMOK kiadásában.

1.1. Információ és adat

Ha valamiről tudunk valamit, akkor egy **adat**ot ismerünk, amely a valóság egy része, **egyed**e valamilyen **tulajdonság**ának **érték**ét határozza meg. (Itt az egyedet a köznapinál tágabban kell értelmezni. Egyed: minden aminek önálló léte van, lehet esemény, tárgy, gondolat, stb).

Például: "A Balaton hőmérséklete 12 Celsius fok", "Gipsz Jakab személyi száma 1-530221-2300", "A berendezés érintése életveszélyes". Példáinkban az egyedek: Balaton, Gipsz Jakab, berendezés; a tulajdonságok: hőmérséklet, személyi szám, érintés; az értékek: 12, 1-530221-2300, életveszélyes.

Ha a tudásunkat fel is tudjuk használni valamire, azaz dönteni tudunk annak alapján, akkor már nemcsak egy adatot ismerünk, hanem egy **információ** birtokosai vagyunk. Így az adattal ellentétben az információ relatív (viszonylagos) és ezért szűkebb fogalom, mert az információ csak az az adat, amely a felhasználójának számára jelent is valamit. Könnyen belátható, hogy ugyanaz az adat egyvalaki számára információ (azaz neki jelent is valamit), másvalaki számára pedig használhatatlan. Példánknál maradva egy ausztrál bennszülöttnek semmit sem

jelent a 12 fokos Balaton, de nekünk fontos információ, ha éppen másnapra kellemes strandolást terveztünk.

Ebben a könyvben az adatok információtartalmáról nem lesz szó, csak magukról az adatokról, azok egymáshoz való kapcsolódásáról. E kis kitérőt mégis azért tartottuk szükségesnek, mert az adatok feldolgozásánál lépten-nyomon beleütközünk abba a kérdésbe, hogy miért van szükség bizonyos adatok gyűjtésére, feldolgozására, ha azokat végülis senki nem használja.

Ha adatról beszélünk, az **egyed, tulajdonság** és **érték** szorosan egymáshoz kapcsolódó fogalmak.

Példa helyett ezúttal szemléltesse ezt a jól ismert "Mennyi? Harminc. Mi harminc? Miért, mi mennyi?" anekdota.

A gyakorlati életben különféle egyedekről kell adatokkal rendelkeznünk. Általában egyféle egyedből nemcsak egynek, hanem többnek adataira van szükségünk. Így a példánkat továbbfolytatva: a Meteorológiai Intézetnél nemcsak a Balaton, hanem az összes hazai vizeink adatait is gyűjtik, nyilvántartják. Ebben az esetben mindegyik egyednek (víznek) ugyanazokra a tulajdonságaira vagyunk kíváncsiak, és így gyakorlatilag ezek az egyedek azonos kategóriába, másnéven egy **egyedtípus**ba tartoznak. Meteorológiai esetünkben ez az egyedtípus a "hazai természetes vizek" lehet.

Rendszerint egyfajta egyedtípus minden egyedénél ugyanazokra a tulajdonságokra vagyunk kíváncsiak, ezért egy típushoz egy felhasználónál – kevés kivételtől eltekintve – egy tulajdonságsorozat(-halmaz) kötődik.

Az egyedtípus már nem egy adattal, hanem adathalmazzal kapcsolatos.

1.2. Logikai adatszerkezetek

Az adatokról szólva említettük, hogy az adat a valóság valamilyen egyedének tulajdonságát, e tulajdonság értékét határozza meg. Így az adat maga is modell, hiszen nem más, mint a valóság egy részének valamilyen *leképezése*.

Ezek az egyedek sokféle kapcsolatban, viszonyban állnak egymással, így az is érthető, hogy a róluk szerzett, gyűjtött adatok között is sokszámú és -fajú kapcsolat létezik.

Anélkül, hogy részleteznénk a különböző logikai adatszerkezeteket, két alapszerkezetről mégis említést kell tennünk.

A valóságban sok helyütt előforduló szerkezettípus a **hierarchia**. (Több helyütt is, mint kellene.) Gondoljunk csak egy vállalati szervezetre, ahol főosztály, osz-

tály, csoport és végül a dolgozók között az alá-, fölérendeltség ábrázolása egy piramis felépítésű rajzot eredményez. Ugyanez az összefüggés áll fenn egy számla és annak tételsorai között.

Elemei közti viszonyokat szülő-gyermek, *tulajdonos-tag* viszonynak is nevezik. Ebben a könyvben az utóbbi kifejezést fogjuk használni. Jellemzője, hogy egy tulajdonoshoz több tag tartozhat, egy tag viszont csak egy tulajdonoshoz kötődhet.

A hierarchikus szerkezet természetesen az adatkapcsolatokban is nagyon gyakran megjelenik. Egy megrendelésen feltüntetett vevő máris egy hierarchia alulról (a megrendeléstől) felfelé (a vevőre) mutató kapcsoló-adata, hiszen egy vevőnek több megrendelése is lehet.

Még általánosabb a **hálós** szerkezet, annak ellenére, hogy a fogalom a hétköznapi életben kevésbé ismert. Ebben a szerkezetben – a tulajdonos-tag terminológiát használva – egy tulajdonoshoz több tag tartozik, de egy tag is kapcsolódhat több tulajdonoshoz. Ha lerajzolnánk ezt a szerkezetet, valóban egy hálóhoz hasonló ábrához jutnánk, amelynek elemei a háló "csomói", az összekötő vonalak pedig a háló szálai. Jellemző példa lehet erre egy szoftverház, amelynek programozói egyszerre több programrendszeren dolgoznak, így egy programrendszerhez kapcsolódik több programozó, de egy programozó kapcsolódhat több programrendszerhez is. Az előző példánkat is kiegészíthetjük úgy, hogy a hálós szerkezetet szemléltesse. Ha arra gondolunk, hogy egy megrendelésen általában több termék is szerepelhet, máris egy hálós szerkezettel állunk szemben, hiszen egy termék több rendelésen is előfordulhat.

Mi köze mindennek a dBASE IV-hez?

Aki már ismeri az elődjeit, az tudja, aki még nem, annak magyarázatul csak annyi szolgáljon, hogy a dBASE használatakor mindig szembekerülünk az adatokkal, szerkezetekkel és ezeket fel kell ismernünk annak érdekében, hogy megtervezhessük, mit is akarunk. Tervezés nélkül ugyanis még egy ilyen "összkomfortos" rendszer alkalmazása is kudarccal végződhet.

1.3. Adatok tárolása, előkeresése

Ha adatainkat a megszületésük pillanatában használnánk fel, valószínűleg ez a könyv sem készült volna el. Sokkal inkább az a jellemző, hogy az adatokat a keletkezésüktől a felhasználásukig valahol, valamilyen módon **tárolni** kell. Azonnal felmerül a gond: Hol, és hogyan tároljuk az adatokat? Sem a hely, sem a mód nem közömbös, hiszen amikor szükségünk van egyes adatokra, többnyire kevés idő áll rendelkezésünkre, hogy előkeressük. Ezért a tárolás módja és helye a kőtáblák előtt is "divatos" kérdés volt és az maradt a számítógépek korában is.

Bizonyára mindenki találkozott már valamiféle nyilvántartással, és nemegyszer rendetlenséggel a nyilvántartásokon belül. Erre, – úgy gondoljuk – kár példákkal a szót szaporítani.

Ennek a rendetlenségnek a köznapi felismerése azonban azt is jelenti, hogy a nyilvántartásokban valamilyen **szervezettségnek, rend**nek kellene uralkodni és ennek hiánya gyakran még kívülálló számára is észrevehető. A rend természetesen nem öncélú, hanem általában az adatok előre megfontolt szempontok szerinti előkeresését szolgálja.

Térjünk vissza a hierarchikus szerkezetet szemléltető vevő-rendelés példánkra.

Ha a megrendeléseket vevőnként külön dossziéba rakjuk, máris létrehoztunk fizikai szinten egy olyan tárolási módszert, amely ezt a hierarchiát mutatja, és lehetővé teszi, hogy gyorsan előkereshessük a megrendelést, ha a vevő türelmetlenül reklamál.

Más a helyzet egy hálós szerkezetű adatkapcsolattal. Nézzük a megrendelés-termék példát:

Ugyancsak bajban lennénk, ha ezt a szerkezetet segédeszközök nélkül akarnánk az adattárolás szintjén megvalósítani. Legfeljebb ollóval vághatnánk szét a megrendeléseket tételenként, hogy azokat termékenként egy halomba gyűjtsük. Ez is elvetélt ötlet, mert lehet, hogy könnyen megtudjuk, miből mennyit kell gyártani, de ha kész a termék törhetnénk a fejünket: Kinek küldjük el?

Nem marad más hátra, mint termékenként egy papírlapot elővenni és rámásolni a megrendelések adott termékre vonatkozó fontos adatait (mennyiség, vevő neve, megrendelés kelte), de ne felejtsük el a termék valamilyen azonosító adatát is felírni.

1.4. Mező, rekord, file, adatbázis

Az előzőekben eljutottunk az adat az információ fogalmának hétköznapi meghatározásáig. Szó esett a logikai adatszerkezetekről, adattárolásról, a számítógépekről azonban csak annyit említettünk, hogy létezik és fejlődik.

Válasszunk más szemszöget, most már egy kicsit közelebb a számítástechnikához.

Ha adatokat tárolunk, akkor általában egy egyedtípus több egyedének többféle tulajdonságáról értékeket tárolunk valamilyen adathordozón, valamilyen rendezettségben és formában.

Különösebb magyarázat nélkül nevezzük az egyetlen *értéket* tartalmazó adatelemet **mezőnek**. Hívjuk **rekordnak** az egyetlen *egyed* tulajdonságait leíró rendezett érték-, azaz mezősorozatot, és végül nevezzük **fájlnak** (az angol "file" szó megfelelője) az egy *egyedtípus*hoz tartozó rekordhalmazt (Így tettek a számítástechnikusok is.).

Itt kell megemlítenünk, hogy adatok szempontjából azonos egyedtípusba azok az egyedek tartoznak, amelyekről azonos tulajdonságok értékeit tartjuk nyilván. Ebből következik, hogy az egyedtípus relatív fogalom, akárcsak az egyed. Így fordulhat elő, hogy ugyanaz az egyed, nevezetesen egy titkárnő a munkahelyén az osztályvezetővel és az ügyintézővel együtt a "dolgozók" egyedtípusba tartozik, és többek között a fizetése, a neme, stb tulajdonságok értékeit tartjuk nyilván róla, a szépségkirálynő-választáson pedig a "jelöltek" egyedtípusba tartozik, akikről a magasságot, hajszínt, kebelméretet, stb. tartanak nyilván.

Említettük, hogy a valóságban a dolgok (egyedek) között is számtalan kapcsolat létezik. Ugyanezek a kapcsolatok az adatok terén is megmutatkoznak. Így kapcsolódhatnak a termékek a rendeléstételekhez a termékszámon keresztül, rendeléstételek a rendelésekhez a rendelés számán keresztül, a rendelések e vevőkhöz a vevő valamilyen azonosítóadatán keresztül.

A fentiek analógiájára nevezzük **adatbázisnak** azt rendezett adathalmazt, amely az egymáshoz valamilyen adaton keresztül *egymáshoz kapcsolt egyedtípusok* adathalmazai alkotnak. Példánkhoz visszatérve adatbázisunkat a vevő-fájl, a termékfájl, a rendelés-fájl és a rendeléstétel-fájl alkotja.

Hangsúlyoznunk kell, hogy egy **adathalmaz** csak akkor adatbázis, ha az **strukturált**, azaz abból bármely kívánt adat előre meghatározható lépésekkel előkeresheto. Egyébként inkább jelhalmaznak nevezhető.

Úgy gondoljuk, hogy a fentiek elegendőek a későbbi tudnivalók megértéséhez annak ellenére, hogy meghatározásaink nem igazán szabatosak és főként nem "nemzeközileg elfogadottak". A továbbiakban tehát mezőkről, rekordokról, fájlokról és adatbázisokról fogunk beszélni. Tehetjük ezt akkor is, ha nem számítástechnikai megoldásról beszélünk, ugyanis egy kartonos nyilvántartásban is könnyen azonosíthatjuk ezeket a fogalmakat. Így nevezhetjük a terméknyilvántartó karton egy rovatát mezőnek, magát a kartont rekordnak, és az összes kartont fájlnak. (Bizonyára furcsán fognak ránk nézni.)

1.5. Adatbáziskezelő rendszer

A tárolt adataink időről-időre változnak. Gyakran merülnek fel olyan kérdések, amelyekre az adataink előkeresése, feldolgozása révén tudunk csak választ adni. Hovatovább az is előfordul, hogy egy dologról az eddigieken kívül más adatokat (tulajdonságot) is tárolnunk kell. Szükség van ezért olyan eszközökre, amely az adatok

- létrehozását,
- megváltoztatását,
- különböző szempontú előkeresését,
- szerkezetének kialakítását, megváltoztatását (beleértve az adatkapcsolatokat is)

lehetővé teszik. Ezeknek a fizikai és logikai eszközöknek az együttesét nevezzük **adatbáziskezelő rendszer**nek. Az adatbáziskezelő már elválaszthatatlan fogalom a számítástechnikától, mert ilyen lehetőségeket biztosító más technikai eszköz nincs.

A számítógépek megjelenéséig az adatok tárolásához, előkereséséhez rendkívül sok elmés, esetenként bizarr szerkezetet találtak ki, amelyeket még ma is (néha a számítógépnél is eredményesebben) használnak. Gondoljunk csak az egyes bűnügyi filmekből jólismert forgó kartotéktárolóra, amely kicsit emlékeztet a mozgólépcső működésére.

A számítógépek megjelenésével és fejlődésével új lehetőségek nyíltak az adatok tárolására, és az előkeresésük érdekében való rendezésük módjára.

Egyre-másra jelentek meg a különböző technikai és szoftvereszközök, amelyek e területen felmerült igények egyre magasabb színvonalú kielégítésére törekedtek. Mind többen és többen foglalkoztak az adatok kezelésével, tárolásával, belső összefüggéseinek kutatásával. Megjelent az **adatmodell** fogalma, és kialakult az adatmodellezés tudománya is.

A kezdetben a fizikai eszközök igen szerény lehetőségeket nyújtottak erre. A mágnesszalagos tárolóra csak sorban (szekvenciálisan) lehetett adatokat felírni, és leolvasni. A mágneslemezek megjelenésével forradalmi változás kezdődött, hiszen fizikailag lehetővé vált az adatok közvetlen elérése.

Megjelentek az első adatkezelők, fájlkezelők. Ezek azonban csak egy fájlon belül tették lehetővé az adatok kezelését, a fájlok közötti kapcsolatokkal már nem tudtak mit kezdeni. Különösen hátrányos volt az, hogy az adatok szerkezetének megváltozásánál egész programrendszereket kellett újraírni.

Az adatmodellezés fejlődésével jelentek meg az első igazi adatbáziskezelők, amelyek már a különböző típusú rekordok közötti kapcsolatokat is kezelni tudták.

Már megjelenésükkor megfogalmazódtak az igények egy jó adatbáziskező rendszerrel szemben:

Adatfüggetlenség: Tegye lehetővé, hogy az adatkapcsolatok, adatszerkezet változásánál a korábbi feldolgozóprogramok módosítás nélkül működhessenek.

Emberközeliség: Gyors válaszidőkkel, laikus számára is érthető párbeszédes lehetőségekkel tegye lehetővé az adatok kezelését.

Teljesség: Tartalmazzon minden olyan eszközt és lehetőséget, amely az adatok kezeléséhez szükséges

Rugalmasság: Minden logikai adatszerkezet (hierarchikus, hálós) kezelésére alkalmas legyen

Adatbiztonság: Rendelkezzen hatékony eszközökkel a technikai zavarok -miatti adatvesztések ellen, valamint az adatokhoz való jogosulatlan hozzáférés ellen.

Minden adatbáziskezelőnek volt egy lényeges eleme, amely megkülönböztette őket a korábbi programnyelvektől. Rendelkezett egy külön **adatleíró nyelvvel**, amely speciálisan és főként a feldolgozóprogramoktól függetlenül az adatok szerkezetének leírására szolgált. E leíró nyelven megfogalmazott meghatározások képeztek hidat a programok és az adattárolás, -kezelés között.

Voltak olyan rendszerek, amelyek teljes programnyelvet alkottak, azaz utasításkészletükkel az adatkezelésen kívüli feladatok megoldhatóak. Ezeket nevezték **önálló nyelvű adatbáziskezelőknek**. (A legközelebbi példa erre maga a dBASE IV)

Más rendszerek csak az adatleírással, adatkezeléssel kapcsolatos utasításokat tartalmazták, és megengedték, hogy a többi, szükséges funkciót más programnyelvek segítségével oldja meg a felhasználó, azok utasításaival biztosították az együttműködést. Ezeket nevezzük **befogadó nyelvű adatbáziskezelő** rendszereknek. (Példáért itt sem kell messzire mennünk: a dBASE IV-SQL befogadó nyelvű adatbáziskezelő, igaz, hogy csak a dBASE IV nyelvét fogadja be.)

Más csoportosítás szerint megkülönböztethetünk **célra orientált** rendszereket, amelyek egy körülhatárolt feladatkör hatékony megoldására születtek; jobbára azonban az **általános**, minden feladatra alkalmazható adatbáziskezelők hódítottak tért tekintettel a nagyobb példányszámuk miatt olcsóbb árukra.

A kezdeti adatbáziskezelők általában nagyszámítógépeken működtek, ezért nem volt igazán fontos a laikus felhasználóval a barátságos, segítőkész kommunikáció, mert ezt az alkalmazási programok teljesítették.

Az első adatbáziskezelőket a hierarchikus adatszerkezet kezelésére fejlesztették ki. Tekintetbe véve a gépek akkori teljesítőképességét talán a legjobb kompromisszum ez volt, mert rendezettség és az adatkapcsolatok kezeléséhez magukban

a rekordokban rejtették el a mutató adatot (pointer), és ez a megoldás nagy sebességet eredményezett.

Hasonló megoldásokkal dolgoztak a hálós adatszerkezetekre "kihegyezett" adatbáziskezelők is.

Mindegyiknek azonban jelentős hátránya volt az adatszerkezetek változásánál a nagyfokú rugalmatlanság, az adatfüggetlenség alacsony foka.

1.6. Relációs adatbáziskezelő

A hierarchikus és hálós adatbáziskezelők fejlesztésével szinte párhuzamosan jelentős kutatások folytak más szemléletű megoldások iránt. Ez az irányzat arra törekedett, hogy a felhasználó számára emberközeli megfogalmazást, és ábrázolást találjon.

A legtermészetesebb, legérthetőbb adatszerkezetnek a táblázatot találták. Ez az évszázados hagyományok után számunkra kézenfekvőnek tűnik, de annál több kételkedést váltott ki a megvalósítás lehetőségeit illetően az akkori szakemberekből. Nem nekik lett igazuk, mert ma már az újonnan született adatbáziskezelők között a relációs adatbáziskezelők szinte egyeduralkodók.

Mi is tehát a relációs adatmodell?

Képzeljünk el egy táblázatot, amelyben a kiválasztott egyedtípus adatait akarjuk elhelyezni. Szinte magától értetődőnek tűnik, hogy az oszlopok fejlécébe rendre a tulajdonságok (azok megnevezései) kerülnek, a sorok pedig egy egyedet jellemeznek. Így egy adott oszlop és egy adott sor által meghatározott táblázatelemben az egyik egyed egyik (az oszlop által meghatározott) tulajdonságának értékét találjuk.

NÉV	*BEOSZTÁS*	FIZETÉS	BELÉPÉS KELTE
1. Kis János	főprogramozó	4000	1979.01.01.
2. Közepes János	programozó	3000	1980.02.02.
3. Nagy János	*alprogramozó*	2000	1981.03.03.
4. Legnagyobb János	alprogramozósegéd	1000	1982.04.04.

Ez a táblázatos szerkezet csak akkor működik helyesen, ha az alábbi követelményeknek eleget teszünk:

1. A táblázat nem tartalmazhat két egyforma sort, azaz két egyedelőfordulásnak legalább egy tulajdonság egy értékében el kell térnie.

2. Az egyes oszlopokban mindig azonos típusu adatnak, azaz azonos tulajdonság egy értékének kell szerepelni.

(A számítástechnikai megvalósításnál még néhány kiegészítő megkötés is létezik).

Az a táblázat, amelyben ezek a feltételek teljesülnek, megfelel a **matematikai reláció** fogalmának. Innen ered a relációs adatmodell elnevezés. Anélkül, hogy halmazelméleti meghatározásokba bonyolódnánk, magyarázatot kell adnunk a halmazokon értelmezhető relációkról, valamint arról, hogy ennek mi a jelentősége adatbáziskezelés szempontjából.

Értelmezzük a tulajdonságokat értékek halmazának. Ha mindegyik tulajdonságból egy értéket (akár találomra) kiválasztunk, akkor a kiválasztott értéksorozat nem más, mint ezeknek a halmazoknak **egy relációja** *(egyféle egymáshoz viszonyulása)*. Észrevehetjük, hogy minél több tulajdonságunk (halmazunk) és azokon belül minél több értékünk (halmazelemünk) van, annál több reláció (többféle egymáshoz viszonyulás) lehetséges, sőt a lehetséges relációk száma a halmazok, és halmazelemek számának ismeretében kiszámítható.

Mindez nekünk, a Felhasználóknak pusztán érdekességnek tűnhet, de korszakalkotó jelentősége mégis abban rejlik, hogy ez az első olyan adatmodell, amely egyértelmű matematikai alapokra került, és lehetővé tette matematikai (halmaz)műveletek alkalmazását az adatkezelés területén.

Mint látni fogjuk, a relációs adatbázishoz hozzátartozik az oszlopokon végezhető műveletek széles skálája. Szükség esetén az adatbázisban már meglévő oszlopok megfelelő összevonásával a létező táblázat része képezhető és ugyancsak létrehozható új táblázat több különböző táblázatban levő oszlop összekapcsolásával. Tekintettel arra, hogy az így keletkezett táblázaton ugyanúgy elvégezhetők a különböző műveletek ezért ez a módszer széles körű variálhatóságával szinte minden igényt kielégít.

Melyek az előnyei a relációs adatbáziskezelőknek?

Emberközeli adatábrázolás

Adatfüggetlenség, nem tartalmaz az adatleírás tárolási és hozzáférési meghatározásokat. A szervezettség, rendezettség külső adatok segítségével (nem az adatbázisban) valósul meg

Egyszerűsödés az adatkezelésben, áttekinthetőség

Minden (hierarchikus, hálós) adatszerkezet könnyen kezelhetővé válik.

Egyértelműség és pontosság a matematikai alapok eredményeként.

A tárolt adatok (táblázat) információtartalmát nem befolyásolja a táblázat (adatok) szerkezete.

1.7. A dBASE IV szerkezete

A dBASE IV egy önálló nyelvű, általános célú, több felhasználó együttes kiszolgálására alkalmas relációs adatbáziskezelő rendszer. Fejlesztője, az ASHTON - TATE különös figyelmet fordított arra, hogy a (rendszer elődeihez hasonlóan) alkalmas legyen a laikus felhasználásra is, azaz emberközeli módszereivel tegye lehetővé, hogy különösebb számítástechnikai előképzettség nélkül is a lényeges adatkezelési funkciókat elvégezhessük, sőt kisebb alkalmazásokat magunk is elkészíthessünk. Kicsit túlozva azt is mondhatjuk, hogy ez volt az elsődleges cél. Ennek megfelelően a dBASE IV az alábbi lényeges elemekből épül fel:

Dot Prompt

("pont" prompt) – Az az üzemmód, amikor a begépelt utasításokat rögtön (prompt). értelmezi (interpretálja) és végrehajtja. Ebben az üzemmódban néhány utasítás kivételével az egész utasításkészlet rendelkezésre áll. A továbbiakban **interpreter**nek fogjuk nevezni.

Control center

(vezérlő központ) – A dBASE IV felhasználói csatolója. Menürendszerrel, párbeszédes módon a legfontosabb adatbáziskezelő funkciók az utasítások ismerete nélkül elérhetők. Beléphetünk innen az alkalmazásgenerátorba is, amely egy másik önállóan is működő alkotóelem.

Application Generator

(Alkalmazás-generátor) – A Control Centerhez hasonló menürendszeres, párbeszédes módon alkalmazási programok készíthetők a konkrét utasítások egy jelentős részének ismerete nélkül. A tervezés befejeztével az alkalmazásgenerátor forráskódu programot készít (generál).

SQL

Structured Query Language (Struktúrált Lekérdező Nyelv) – A relációs adatbáziskezelés területén az egyik és legjelentősebb szabványosítási törekvés az SQL. Azzal együtt, hogy ez a mai napig szakmai viták tárgya, az ASHTON-TATE figyelmet fordított arra, hogy adatbáziskezelőjét felszerelje egy modullal, amely a szabványos SQL utasításkészlettel rendelkezik, és az SQL műveletek a dBASE adatállományokon végrehajthatók.

Könyvünk is követi a fenti tagolást, de külön fejezet szól a programozás alapjairól, a többfelhasználós üzemmódról, a dBASE IV és a dBASE III PLUS különbségeiről, valamint függelékekbe kerültek a fontos tudnivalók tárgyszerű és teljeskörű magyarázatai.

2. Az adatok szerkezete

Könyvünk bevezetőjében kétféle megközelítésben is beszéltünk az adatszerkezetekről:

- szó esett a dolgoknak (az adatokkal jellemzett dolgoknak) a valóságos kapcsolataiból származó logikai adatszerkezetekről (hierarchikus, hálós),
- említést tettünk a mező, rekord, fájl, adatbázis fogalmakról, amelyek úgyszintén az adatok egyfajta szerkezeti csoportosítását jelentik mégpedig abból a szempotból, hogy mire vonatkoznak.

Itt és most az utóbbiról lesz szó, nevezetesen arról, hogyan van ez a dBASE IV-ben.

Elöljáróban meg kell jegyeznünk, hogy az adat kifejezés használatánál visszatérünk bevezetőbeli definícióról a hétköznapi értelmezésre, azaz adat alatt valamiféle konkrét értéket értünk, függetlenül attól, hogy az az érték mit jelent, és mire vonatkozik. Ez ugyan ellentmondani látszik korábbi meghatározunknak, de – mint látni fogjuk – ez egyáltalán nem lesz zavaró és egy-egy esetet jól végiggondolva semmiféle tartalmi ellentmondást nem fogunk találni.

2.1. Adattípusok

A dBASE IV hasonlóan más programnyelvekhez és adatbáziskezelőkhöz, az adatokat bizonyos jellemzőik szerint csoportosítja:

Megnevezés	Kód	Leírás
Character	C	karakteres vagy alfanumerikus (legfeljebb 254 karakter)
Numeric	N	numerikus (BCD – binárisan kódolt decimális szám)
Float	F	numerikus (lebegőpontos szám)
Logical	L	logikai típus
Date	D	dátum típus
Memo	M	Memo: (megjegyzés) különleges alfanumerikus

2.1. táblázat

Az adattípus meghatározza, hogy milyen értékeket (jeleket) tartalmazhat, az adattal milyen műveleteket lehet végezni. Ezért a típus az adat igen fontos jellemzője. A táblázatban az adattípus kódjait is feltüntettük. A későbbiekben ezzel

fogunk hivatkozni rájuk. A TYPE függvény is ezekkel a betűkódokkal tér vissza. Lásd később. Lássuk sorra mit kell tudni az egyes adattípusokról.

2.1.1. Karakteres (Character)

Bármilyen, 0-255 ASCII kódu jelet tartalmazhat. Legfeljebb 254 jel (karakter) hosszú lehet. Így tartalmazhat nagy betűket (A,B,C,...,Z), kis betűket (a,b,c,...,z), számjegyeket (0,1,...,9), írásjeleket (pl.: !,:;., stb.), grafikus karaktereket, stb. Típuskódja a C.

2.1.2. Numerikus, BCD (Numeric)

A BCD (Binary Coded Decimal – Binárisan Kódolt Decimális) szám típusu adat. Csak a számjegyek, előjel (+-) és a tizedespont (.) alkothatják. Hossza legfeljebb 20 számjegy lehet az előjellel és a tizedesponttal együtt. (Természetesen a pozitív előjelet nem kell feltüntetni. Ellentétben a lebegőpontos számábrázolással a BCD módszer minden egyes számjegyet ténylegesen ábrázol, ezért *kerekítési hibák nem fordulhatnak elő*. Az üzleti és gazdasági életben (főként a könyvelésnél) a számszaki egyezőség követelménye miatt nagy jelentősége van a kerekítési hibák elkerülésének, ezért ezen a területen kifejezetten a BCD típusú numerikus adatok használata javasolt. Típuskódja az N.

2.1.3. Numerikus, lebegőpontos (Float)

Csak a számjegyek, előjel (+-) és a tizedespont (.) alkothatják. Hossza legfeljebb 20 számjegy lehet az előjellel és a tizedesponttal együtt. (Természetesen a pozitív előjelet itt sem kell feltüntetni). A lebegőpontos számok nem számjegyenkét kerülnek ábrázolásra, ebből adódóan az N (BCD) típusunál lényegesen nagyobb és lényegesen kisebb számok is előfordulhatnak. A legnagyobb ábrázolható szám a 0.9E+308, a legkisebb pedig a 0.1E-307. A nagyobb ábrázolási tartomány miatt miatt elsősorban a matematikai, tudományos számításokra alkalmas, az előforduló kerekítési hibák miatt viszont nem igazán ideális a számszaki egyezőséget igénylő könyvelési, pénzügyi feladatokra. Típuskódja az F.

Az F és N típusu adatok csak belső ábrázolási formában térnek el egymástól. Megjelenítéskor, számításokban való használatkor ugyanúgy viselkednek, leszámítva természetesen az F típusu mezők esetleges pontatlanságát.

2.1.4. Logikai

Sok olyan tulajdonság, jellemző létezik, amelynek csak **két értéke** lehet. Például: a lámpa ég – nem ég, a tétel nincs feldolgozva – fel van dolgozva, férfi – nő, beteg – egészséges, dohányzó – nemdohányzó stb. Ilyen kétértékű jellemzők értékeinek megadásához használhatjuk a logikai adattípust. A matematikai logikából ered a két érték elnevezése: Igaz – Hamis. A dBASE IV az angol True – False (Igaz – Hamis) mellé a Yes – No (Igen – Nem) szavak kezdőbetűit is felajánlja, de ezek természetesen csak két értéket képviselnek. Ezek közül az **Igaz** értéket jelenti a **T, t,** illetve **Y, y** betűk bármelyike, a **Hamis** értékhez pedig az **F, f , N, n** betűkből választhatunk.
Meg kell jegyeznünk, hogy az adatbeviteltől eltérően programíráskor a betűk elé és mögé mindig kell tennünk egy pontot is.(Így: .T./.t./.Y./.y./.F./.f./.N./.n.). Típuskódja az **L.**

2.1.5. Dátum

Csak naptári időpont-értékek alkothatják alapértelmezésben hó/nap/év formában. Hossza 8 bájt. A formátum a SET DATE, SET CENTURY utasításokkal megváltoztatható, így számunkra a megszokottabb év.hó.nap alak is beállítható. A dBASE IV ellenőrzi a dátum helyességét és érvénytelen adatot nem fogad el. Például 11/32/1989 érvénytelen dátum. A legkisebb elfogadható dátum 01/01/0100, a legnagyobb 12/31/9999. Általában a rövidebb alak használatos, ahol az évnek csak az utolsó két számjegyét kell megadnunk.
A dBASE IV sokrétűen támogatja a dátumokkal kapcsolatos műveleteket. Lehetséges a dátumok összehasonlítása, különbségük képzése, hozzáadhatunk napokat egy dátumhoz, stb. Típuskódja a **D.**

2.1.6. Memo

Különleges lehetőséget nyújt a dBASE IV az olyan adatok számára, amelyeket nehezen vagy nem tudunk formalizálni, amelyeknek hosszát, szervezettségét előre nem tudjuk.

Gondoljunk csak arra, hogy egy vendéglőben a "repertoárt" akarjuk számítógépen nyilvántartani, hogy abból később az aktuális étlapot kinyomtathassuk. Az étel számához, típusához, nevéhez, árához könnyen megtaláljuk a megfelelő adattípust, és kis töprengés után az adatok hosszát is. Bajban vagyunk viszont, ha az igényes vendégnek az étlapon fel akarjuk tüntetni az étel összetételét és egyéb jellemzőit. A "Rántott gomba rizzsel, tartárral" nem kíván magyarázatot, a "Többszemélyes adatbáziskezelő tál Ashton-Tate módra" nevű ételről talán ez a könyv is kevés. (Reméljük nem.)

A memo típusu adat alapvetően e célból született. Ugyanazokat a jeleket tartalmazhatja, mint a karakteres adat. Hossza legfeljebb 512 kilobájt(!) lehet. Típuskódja az M.

2.2. Az adatbázis szerkezete

A dBASE az adatokat (a konkrét értékeket) részben a számítógép memóriájában részben valamilyen háttértárolón (merev-, hajlékonylemez) tárolja (hasonlóan más programnyelvekhez). A memória csak az ideiglenes adatok tárolására szolgál. Azok az adatok, amelyeket egy feldolgozáson túl is meg kell őrizni csak a háttértárolóra kerülhetnek, és a memória szűkössége (nemkevésbé az áramkimaradás okozta adatvesztés) miatt ott vannak a feldolgozás ideje alatt is.

Az hogy fizikailag hol és hogyan rögződnek az adatok, már hosszú ideje az operációs rendszerek dolga, így itt sem foglalkozunk vele.

Lényegében mondhatjuk azt, hogy adatkezelés alatt a háttértárolón lévő adatok kezelését kell érteni, ezért a továbbiakban ha adatállományokról beszélünk az itt tárolt adatokat értjük alatta.

A dBASE IV sokféle adatot tárol. Megkülönböztethetünk ezek között **elsődleges vagy forrás** és **másodlagos vagy származtatott** valamint **technikai** jellegű adatokat.

Kiemelt jelentőségűek az elsődleges adatok, ugyanis ez alatt az általunk (a Felhasználó által) tárolás és feldolgozás céljából bevitt adatokat értjük. Ez az adatbázis-fájl. A következőkben jobbára ennek szerkezetéről, összefüggéseiről lesz szó.

A másodlagos adatok az elsődleges adatokból származnak és elsősorban az általunk kívánt (kimenő) adatok gyors előkeresését, megjelenítését szolgálják. Ilyenek az index-állományok, a lekérdező-állományok.

Se szeri se száma azoknak a technikai jellegű adatfajtáknak, amelyek az adatok különböző helyen való megjelenítésének formátumával és feldolgozásuk módjával kapcsolatosak, egyesek pedig nem is az adatokhoz, hanem a feldolgozó eljárásokhoz kapcsolódnak, tulajdonképpen azok részei. Ilyenek a formátumleíró állományok, a forrásprogramlisták, tárgykódra fordított programok, stb.

Mindezek részletes felsorolása a különféle állományokat azonosító kiterjesztésekkel együtt a függelékben megtalálható.

2.2.1. Adatbázis-fájl

A bevezetőben az adatszerkezetnél alulról fölfelé, az elemitől az összetettig, azaz a mezőtől a fájlon keresztül az adatbázisig haladtunk. Ezúttal fordítva tesszük.

Adatbázisunkban többféle egyedtípusról egy-egy adatbázis-fájlban őrzünk adatokat. Ezek között a kapcsolat az adatok tartalmán keresztül valósul meg, így semmiféle technikai kapcsolóeszközre, állományra nincs szükség, pusztán arra, hogy a kapcsolóadatot az összekapcsolni kívánt fájlok mindegyike tartalmazza. (Emlékezzünk a vevőkódra a vevő-fájlban és a vevőkódra a megrendelés-fájlban.) Így egy fájlhalmazról nem is tudjuk ránézésre eldönteni, vajon egy adatbázisba tartoznak e vagy sem. Még akkor sem lehetünk biztosak, ha belenézve az adatok közé felfedezünk valami olyant, amely kapcsolóadat is lehet, mert nem biztos hogy a lehetséges kapcsolat értelmes-e és főként *létezik-e*. A kapcsolatot ugyanis csak mi magunk, vagy a számunkra készített program fogja megteremteni. Ezért az adatbázis a dBASE-ben nem jelenik meg külön technikai elemként, a kezelőrendszer szempontjából lényegtelen, mely fájlok tartoznak össze, melyek nem. (Természetesen nem így van ez a Felhasználó szemszögéből.) Ezért a dBASE-ben az adatok legnagyobb csoportosító fogalma az **adatbázis-fájl**, amelyet – jobb híján – a továbbiakban az **adatfájl** névvel fogjuk illetni (vállalva a bírálatokat).Külön felhívjuk a figyelmet, hogy a **dBASE terminológiában** az **adatbázis** (Database) leggyakrabban **az adatfájl szinonimája** (éppen a fentiekből következően).

A fájl fogalom alatt ezentúl a típusától függetlenül minden együvé tartozó adathalmazt fogunk érteni, kibővítve egy kicsit korábbi értelmezésünket, azaz fájlról beszélünk egy szöveges adatállomány, egy futtatható program lemezen tárolt állománya, stb esetén. Ha tehát a "fájl" kifejezést látjuk, ez bármilyen típusu adatállományt jelent, így akár adatfájlt is. Ha az szükséges, mindig megjelöljük a típust is. (Pl: makrofájl, formátumfájl, programfájl, stb).

Az adatfájl tehát nem más, mint az egy egyedtípushoz tartozó adathalmaz. A relációs kifejezéshasználattal élve az adatfájl maga a táblázat, és ahogyan már sejtjük is *elsődleges jellegű* adatok halmaza.

A dBASE az adatfájlt a más típusu fájloktól a **.dbf** kiterjesztéssel különbözteti meg.

2.2.2. Adatbázis-rekord

A táblázat egy sora, amely az egyedtípus egy egyede jellemző (általunk kiválasztott) tulajdonságainak értékeit tartalmazza, az adatbázis-rekord.

A rekord szerkezete kötött, azaz azonos adatfájlhoz tartozó rekordokban a **tulajdonságok azonos sorrendben** kell hogy kövessék egymást, sőt az egyes értékek is csak **azonos hosszon** és **azonos adattípussal** ábrázolhatók.

2.2.3. Adatbázis-mezők

A mezők az adatszerkezet legkisebb elemei, amellyel mint Felhasználók foglakoznunk kell.

A mező kifejezésnek kétféle értelmezése fordul elő:
Általában a mező kifejezést használjuk, amikor egy egyedtípus egy tulajdonsága egy konkrét értékéről, azaz a táblázat egy sorának és egy oszlopának metszéspontjában lévő táblaelemről beszélünk.
Gyakran magukat a tulajdonságokat, azaz a táblázat egy oszlopát hívjuk mezőnek. Így amikor azt mondjuk, hogy kiválasztjuk egy adatfájl egy mezőjét, akkor tulajdonképpen egy oszlop kiválasztására kell gondolni.

Ha most talán nem is, de később ki fog tűnni, hogy a kétféle értelmezés világosan elkülönül és mindíg tudni fogjuk miről van szó.

A mező csak adatfájl – illetve azon belüli rekord – része lehet, tehát olyan adat, amelyet a háttértárolón tárolunk, és nem a memóriában.

2.3. Az adatbázis-fájl meghatározása

Amikor elhatározzuk, hogy adatokat gyűjtünk, annak érdekében, hogy ezeket az adatokat feldolgozzuk a döntéseinkhez szükséges információelőállítás céljából, voltaképpen egy adatbázist, azaz táblázatokat határozunk meg. Ilyenkor – ha megfelelő a hozzáértésünk – már szinte teljes mélységében tudnunk kell, hogy mire (milyen információkra) vagyunk (leszünk) kíváncsiak, hiszen ez fogja meghatározni, mely dolgok milyen tulajdonságairól gyűjtsünk az értékeket (adatokat). Mindez szorosan hozzátartozik egy adatbázis és így az adatfájlok meghatározásához, azonban egészen más tudomány, amellyel könyvünk nem foglalkozhat.

Az adatfájl meghatározásánál tehát abból indulunk ki, hogy pontosan ismerjük, hogy miről (egyedtípus), és milyen adatokat (tulajdonságokat) kell gyűjtenünk, sőt nagyvonalakban tisztában vagyunk azzal is, hogy az adatokat az információnyeréshez milyen szempontok alapján kell előkeresni.

Így a következőkben arról lesz szó, mit is kell eldöntenünk és megadnunk ahhoz, hogy a dBASE IV-ben egy adatfájlt meghatározzunk. A hogyant, azaz a meghatározás utasításait, azok szintaktikáját a későbbiekben tárgyaljuk.

Először meg kell határoznunk az **adatfájl nevét**, hogy ezzel a többitől (táblázattól) megkülönböztessük.(Ez legfeljebb 8 karakterből állhat, csak angol betűket, számokat és aláhúzásjeleket tartalmazhat.)

2.3.1. A mezők jellemzői

Meg kell adnunk sorra a tulajdonságok (oszlopok), azaz **mezők jellemzőit**, nevezetesen:

- a nevét (legfeljebb 10 karakter, angol betűk, számok és aláhúzásjel, csak betűvel kezdődhet).
- az adattípusát,
- a hosszát,
- az adattípushoz tartozó egyéb jellemzőket,
- azt, hogy az adott mezőt használjuk-e előkeresési szempontként (INDEX) Lásd később.

Összesen legfeljebb 255 mezőt határozhatunk meg egy adatfájl esetén.

Amint azt láthatjuk, az adatfájl szerkezetének meghatározása gyakorlatilag egyet jelent a rekord szerkezetének meghatározásával, ez pedig nem más, mint a rekordokat alkotó mezők sorrendjének és jellemzőinek meghatározása. Ebből egyben az is következik, hogy egy fájl csak **azonos szerkezetű** (azonos típusú) és így **azonos hosszúságú rekordok**ból állhat. A rekordok hossza korlátozott: nem haladhatja meg a 4000 bájtot.

2.3.2. Egyes mezőtípusok sajátosságai

A mezők meghatározásánál néhány mezőtípus különleges jellemzőit figyelembe kell vennünk, és néhol az adattípusnál leírtakat ki kell egészítenünk.

Karakteres mező

0 hosszúságú mező nincs, ezért a karakteres mező hossza — az adattípusnál leírtaktól eltérően — 1-től 254-ig változhat.

Numerikus mezők (BCD,FLOAT)

Mindkét numerikus típusnál a hossz mellett a tizedesjegyek számát is meg kell adni. (Ez a hosszba beleszámít a tizedesponttal együtt!). Például, ha szám hosszának tizenötöt adtunk meg, a tizedesjegyek száma pedig kettő, akkor az

egész számjegyek száma 15-2-1-1 = 11 . Egész számok esetén a szám hosszába csak az előjel számít bele. Így egy 15 hosszúra kijelölt numerikus mező 14 számjegyet és a szám előjelét tartalmazhatja.

Logikai

Nem használható keresési szempontként (INDEX). Lásd később.

Memo

A memo különleges mező. A dBASE fejlesztői ezzel a lehetőséggel enyhíteni akartak azokon a gondokon, amelyek a fix (kötött) rekordhosszúságból adódnak. A memo mező segítségével változó hosszúságú adatokat fűzhetünk minden, vagy csak némelyik rekordhoz. Az adatbáziskezelő gondoskodik arról, hogy a változó hosszú adataink a lehetőség szerinti legkisebb tárolóterületet foglalják el, és arról is, hogy az adott rekord előkeresésével együtt ezek az adatok is rendelkezésre álljanak. A memo mező az adatfájlban mindig csak 10 bájtot foglal el, de maga az adat 512 Kbájt hosszú is lehet. Ez úgy válik lehetővé, hogy a dBASE a memo adatokat nem magában az adatfájlban, hanem az adatfájllal azonos nevű, de **.dbt** kiterjesztésű fájlban helyezi el. Az adatfájlban a memo mező lényegében csak egy mutatót tartalmaz, amely a .dbt kiterjesztésű fájlban tárolt adat helyét (és hosszát) határozza meg. A .dbt fájl létrehozása, kezelése számunkra észrevehetetlen.
Természetesen a fentiekből adódóan a memo mezők felhasználása csak korlátozott lehet, mert minden olyan művelet, amely a fix hosszúságú adatokkal elvégezhető (oszlopműveletek, keresés, stb) a memo mezővel lehetetlen.

A *.dbt* kiterjesztésű fájl az adatfájlhoz hasonlóan elsődleges jellegű adathalmaz, mert tulajdonképpen nem más mint az adatfájl egy "nyúlványa".

2.4. Szempontok szerinti szervezés (index)

Az adatok tárolását úgy kell megszerveznünk, hogy abból a kívánt szempontok szerinti adatokat gyorsan megtaláljuk. Egyszerű példaként a telefonkönyvet vehetjük, ahol általában név szerint keressük ki a telefonszámot, ezért a neveket ABC sorrendbe rendezik. Nem igaz ugyanez a Posta cím szerinti tudakozójánál, ahol ugyanezeket az adatokat a gyors előkeresés érdekében cím szerint kell rendezni.

A dBASE – hasonlóan sok más adatbáziskezelőhöz nem több különböző rendezettségü telefonkönyvet (fájlt), hanem minden szempontra egy ún. **index**táblázatot készít. Ezek csak az adott szempont szerinti adatot és egy mutatót tartalmaznak, ami az eredeti adatfájl rekordjának sorszámára utal. Az index ezért –

többnyire lényegesen – kisebb az eredeti adathalmaz méreténél, ezért a keresés is lényegesen gyorsabb a segítségükkel. Az eredeti (adatfájl) adatok módosulásakor az adatbáziskezelő gondoskodik arról, hogy az indextáblák is megfelelően módosuljanak. Igaz ugyan, hogy egyszerre csak 10 indextábla karbantartása lehetséges (egy adatfájlhoz csak ennyit lehet egyidőben megnyitni), így a többiről nekünk kell intézkedni. (REINDEX)

2.4.1. Összetett indexfájl

A dBASE IV az indextáblázatokat általában összetett indexfájlban tárolja, és ezzel a használatot néhány konkurensénél is kényelmesebbé teszi. Nem kell ugyanis egyenként a különböző szempontokat (indexeket) a karbantartáshoz kijelölni, hanem elég csak erre az összetett fájlra hivatkoznunk. A dBASE IV az **összetett indexfájlt** **.mdx** kiterjesztéssel különbözteti meg a többi fájltípustól. Egy összetett indexfájl legfeljebb **47** egyedi szempontot, azaz indexet tartalmazhat. Így ha egy adatfájlhoz megnyitunk egy összetett indexfájlt akkor legfeljebb(!) 47 szempont valamelyike szerint kereshetünk adataink között.

2.4.2. Egyszerű indexfájl

Ez a korlát sem áthághatatlan, mert alkalmazhatunk **egyszerű indexfájlt** is, amely csak egy *keresési kulcs*ot (szempontot) tartalmazhat. Az egyszerű indexfájlt **.ndx** kiterjesztéssel találjuk a fájlok között.

2.4.3. Keresési kulcsok

A lehetőségeket szinte korlátlanra bővíti az, hogy a keresési kulcs nemcsak egyszerűen egy mező lehet, hanem a mező értékének szinte minden művelet szerinti transzformációja, sőt megengedett több mező bonyolult kombinációja is.

Itt kell megismételnünk azt, hogy a logikai és memo adattípusok indexként (keresési szempontként) nem használhatóak.

2.4.4. Adatkeresés index nélkül

Meg kell említemünk, hogy a dBASE az adatok előkereséséhez más lehetőségeket is nyújt. Ilyen például a megadott mező (szempont) szerint (SORT), a szekvenciális keresés feltételek szerint (LOCATE), stb. Természetesen ezek lényegesen időigényesebb eljárások.

2.5. Memóriaváltozók

2.5.1. Változókról általában

Egy adott mező mindig egy adott fájlhoz tartozik. Ha egy adatfájl egy mezőjére hivatkozunk, akkor az aktuális rekordban levő értéket kapjuk vissza, illetve az aktuális rekordban levő érték módosul. Egy-egy alkalmazás (program) futása során sok ideiglenes adatot kell tárolni. Nem megoldás, ha ezeket az adatokat lassú elérésű lemezen tartjuk egy adatfájlban.

"Kezdetben vala" a memóriaváltozó. A számítástechnika "kőkorszakában" nem is léteztek igazi háttértárolók, így az adatokat minden feldolgozás elején be kellett vinni a memóriába, és csak annyi adatot kezelhettünk, amennyi ott elfért.

A memóriaváltozók abban különböznek a mezőktől, hogy **függetlenek minden adatfájltól**, adatkezelő művelettől és így folyamatosan rendelkezésünkre állnak az egész feldolgozás során. Használatuk lényegesen gyorsabb a háttértárolón tárolt adatokénál (a mezőknél), mert hozzáférésükhöz semmiféle mechanikus mozgásra nincs szükség.

Lényeges kiemelnünk, hogy **memo típusú memóriaváltozó nincs**. (Nincs is értelme.)

A tömörség kedvéért a továbbiakban a memóriaváltozókat egyszerűen **változók**nak fogjuk hívni tekintettel arra, hogy másfajta változó a dBASE-ben nincs.

2.5.2. Tömbök

A tömb különleges változó. Tulajdonképpen nem is egy, hanem több változót helyettesít. Az egydimenziós tömböt felfoghatjuk egy oszlopként (akár sorként is), amelyre az a jellemző, hogy minden eleme azonos típusú. Ugyanez vonatkozik a kétdimenziós tömbre is, azzal a különbséggel, hogy azt egy táblázatként képzelhetjük el. A tömb elemeire számokkal hivatkozhatunk, és jelentősége is ebből adódik, mert az egyes elemek tartalmát (értékét) matematikai műveletek, kifejezések segítségével kereshetjük elő. Minden egyes tömbelem önállónak számít és így rendre más és más értékeket tartalmazhat.

Világítsuk meg egy példával mire is jó a tömb:

Tételezzük fel, hogy igényes Felhasználó egy kinyomtatott listán a dátumot nem a hónap számával jelölve, hanem szövegesen szeretné látni, azaz "04" helyett "április"-t "08" helyett "augusztus"-t. (Emlékeznünk kell arra, hogy a dBASE dátum adattípusa a hónapokat számmal jelöli. Lásd: adattípusok) Természetesen nem tudjuk előre, hogy a kiíráskor milyen dátum jön. Ugyancsak megizzadnánk, ha a

dBASE utasításaival, de tömb alkalmazása nélkül akarnánk a feladatot megoldani. (Tessék megpróbálni!) Ha viszont a hónapok nevét rendre elhelyezzük egy (12 elemü) egydimenziós tömbbe (oszlopba), akkor annyiadik tömbelemre kell csak hivatkoznunk, amennyi a hónap száma és máris "kezünkben" a hónap neve is.

2.6. A változók alkalmazása

2.6.1. Általános tudnivalók

Név

A változókat is névvel kell megkülönböztetnünk egymástól. A nevekre ugyanazok a szabályok vonatkoznak mint a mezőknél. A változó neve megegyezhet bármelyik mező nevével. Ha ilyen esetben hivatkozás történik erre a névre, akkor mindig a mezőben levő értéket kapjuk eredményül, illetve értékadó utasításnál a mező értéke módosul. Tudnunk kell tehát, hogy azonos neveknél a dBASE a mezőt részesíti előnyben. Ha ilyenkor (névegyezőségnél) mégis a változóval akarunk műveletet végezni, akkor a változónév elé az "**M->**" **előtagot** > előtagkell illesztenünk. Ez nem számít bele a változó nevének (10) hosszába.
Például: M- > ALMA, M- > EZ10HOSSZU.

Korlátok

A változók a számítógép operatív memóriájában találhatók, ezért használatuk bizonyos korlátokba ütközhet. Programunkban alaphelyzetben nem alkalmazhatunk egyszerre 500 változónál többet, de szükség esetén ezt a korlátot 15000-re növelhetjük. (Lásd: *dBASE IV konfigurálása*)

Létrehozás és adattípuskijelölés

Változót értékadó utasítással hozhatunk létre (STORE vagy =) és a RELEASE utasítással szüntethetünk meg. Például: STORE 5 TO BARACK. Ha eddig nem volt BARACK nevű változó, akkor ettől kezdve létezni fog és a tartalma (értéke) 5 lesz.

Az értékadással eldől a változó adattípusa is. A változó típusát az értékéül adott kifejezés típusa határozza meg. Például: STORE "Géza" TO NEV, ekkor a NEV alfanumerikus típusu lesz, STORE CTOD("06/21/89") TO SZULNAP esetén pedig a SZULNAP dátum típusu lesz.

Hatáskör

Egy tipikus dBASE IV alkalmazás egymásba skatulyázott (ágyazott) eljárások sokaságából áll. A főprogram meghívja a saját eljárásait, és azok is meghívják a hozzájuk tartozó eljárásokat. A főprogramban létrehozott változó az összes al-

programban "látható", elérhető. Egy eljárás által létrehozott változó már csak az eljárásból meghívott (alárendelt) eljárásokban használható, és persze magában a hívó eljárásban. Azt, hogy egy változó mely eljárásokban érvényes, a változó hatáskörének nevezzük.

A változó hatáskörét a dBASE IV automatikusan jelöli ki, de ezt az alapértelmezést szükség esetén megváltoztathatjuk, azaz elérhetjük, hogy egy változó csak az adott eljárásban legyen érvényes, de azt is, hogy az összes eljárásban elérhető legyen. (Lásd: *PRIVATE, PUBLIC utasítások*)

2.6.2. Tömbök alkalmazása

A dBASE IV egy és két dimenziós tömbök használatát engedi meg. Bár egy tömb esetenként sok elemből áll, mégis az elemek számától függetlenül csak 1 változónak számít. Így ha a változók száma 500-ra korlátozott és létezik egy tömb 500 elemmel, akkor is még 499 egyéb változót használhatunk vele párhuzamosan.

Tömbök *létrehozás*ához az egyszerű változóktól eltérően más utasításokat kell használnunk. (lásd: *PUBLIC, DECLARE utasítások*). Létrehozáskor csak a tömb nevét, a dimenzióját, és a méretét, azaz elemeinek a legnagyobb számát tudjuk meghatározni. A tömb létrehozását, méretének meghatározását **deklaráció**nak nevezzük. Egy tömb legfeljebb 1170 elemből állhat.

A deklarációs utasítással nem lehet a tömb *adattípus*át meghatározni. A létrehozott tömb minden eleme kezdetben logikai típusu lesz .F. (False = Hamis) tartalommal. Itt is az első értékadó utasítás dönti el (változtatja meg) az adattípust. Ehhez elég a tömb (bármelyik) egyetlen elemének értéket adni.

A dBASE IV sok utasításában kihasználja a tömbök különleges tulajdonságait. (Lásd : *APPEND, AVERAGE, CALCULATE, COPY, COUNT, SUM, CLEAR ALL, CLEAR MEMORY, LIST/DISPLAY MEMORY, PUBLIC, RELEASE, SAVE, RESTORE*)

2.6.3. Sajátosságok az adattípusoknál

A változókra vonatkozó szabályok csak néhány esetben térnek el az adattípusoknál fölsoroltaktól. Itt most ezekre térünk ki megismételve azt is, hogy memo típusú változó nem létezik.

Karakteres

A karakteres változó hossza — a mezőétől eltérően — lehet 0 is. Így a hossz 0 és 254 közé eshet. Különbözik a mezőtől abban is, hogy a — mező fix

hosszával ellentétben – a változó hossza létezése alatt változhat. Minden változó a memóriában karakterenként 1 bájtot foglal el. Változónként további 2 bájtot foglal a dBASE egyéb információk tárolására. Így egy 3 karakteres változó 5 bájt hosszú.

Numerikus BCD

Ha egy numerikus változót hozunk létre egyszerű értékadó utasítással (STORE 0 TO OSSZEG vagy ATLAG = ((N1 + N2 + N3)/3), akkor a keletkező változó mindig N típusu lesz.

Numerikus lebegőpontos (FLOAT)

Ha F típusu változót akarunk létrehozni, akkor nem elég az egyszerű értékadás, hanem erre külön függvényt – FLOAT() – kell használni. Például LEBEGO = FLOAT(10). Ha egy matematikai kifejezésben egyaránt szerepel N és F típusu adat, akkor az eredmény F típusú lesz (feltéve, ha az eredményt nem éppen N típusú mezőben tároljuk, aminek – tudvalevő – a típusa értékadással nem változik meg).

2.6.4. Rendszerváltozók

Néhány speciális célú, kiemelt jelentőséggel rendelkező változót is találhatunk a dBASE IV-ben. Ezeket rendszerváltozóknak nevezzük. A dBASE IV szerves részét képezik, ebből következően létszámuk és nevük megváltoztathatatlan. Hogy ne lehessen összekeverni őket a hagyományos változókkal, neveik "_" (aláhúzás) karakterrel kezdődnek.

A legtöbb rendszer változó a – leginkább a nyomtatóra – kimenő adatfolyamra (stream output) hat, a megjelenítés formájával kapcsolatos paramétereket tartalmazza. Például: a bal margó nagysága (_lmargin), a nyomtatott karakterek sűrűsége (_ppitch), stb.

A rendszerváltozókra a közönséges változókkal megegyező szabályok vonatkoznak, de

- nem lehet őket a memóriából törölni (CLEAR, RELEASE, stb).
- Típusuk adott és nem változtatható.
- A rendszer indításakor felveszik alapértelmezés szerinti értékeiket, amelyek természetesen bármikor megváltoztathatóak.

Részletesen a dBASE IV programozás alapjai c. fejezetben, és a Függelékben olvashatunk róluk.

3. A dBASE IV programozás alapjai

3.1. Bevezetés

A dBASE IV Control Center-e és azonbelül az Alkalmazás-generátor segítségével egy nem túl bonyolult rendszer különösebb programozói előképzettség nélkül, kényelmesen elkészíthető, de egy ily módon generált programtól ne várjuk a dBASE IV nyelvén egy adott célra fejlesztett program rugalmasságát és hatékonyságát.

A dBASE IV procedurális nyelve szintén alkalmas teljes programrendszerek megalkotására, de kimondottan ezt érdemes használni olyan gyakran ismétlődő, rövid eljárások megírásakor, amelyeket az Alkalmazás-generátorral meglehetősen nehézkes lenne előállítani.

Röviden összefoglalva azt javasoljuk, hogy tervezett rendszerünk vázát, s bizonyos globális funkcióit tervezzük meg az Alkalmazás-generátorral (ezáltal sok időt takaríthatunk meg), majd az így kapott forráskódot tovább írva alakítsuk ki a részletes programokat. **Mindenekelőtt azonban gondoljuk végig, tervezzük meg, hogy mit is akarunk.**

Egy alkalmazás tervezésekor először mindig az adatbázist alakítsuk ki. Lehetőség szerint kerüljük a redundáns adattárolást. Inkább több fájlt hozzunk létre, s ezeket kapcsoljuk össze a SET RELATION utasítással, mintsem néhány fájlba zsúfoljunk bele minden információt. Azonban a másik végletbe sem szabad esni, ugyanis gondoljunk arra, hogy egyidőben legfeljebb 10 adatfájlt tarthatunk nyitva.

3.2. A programok szerkezete

A programrendszer tervezésekor a legfontosabb szempont a **modularitás**. Ez azt jelenti, hogy a rendszert olyan egymást kizáró részekre kell felosztanunk, amelyek egyenként egy-egy jól körülhatárolt feladatot hajtanak végre. Ezáltal növelhetjük ugyanis a programok olvashatóságát, s (különösen többszemélyes feladatoknál) a fejlesztés hatékonyságát (a programok tesztelésénél jelentkező előnyökre az erről szóló alfejezetben külön is felhívjuk a figyelmet). Az így megtervezett modulokat általában egy ún. **főprogram** vezérli. Ebben többnyire csak a felhasználót támogató menürendszer és a megfelelő programhívások kapnak helyet.

Nézzük meg ezt egy kicsit részletesebben. Egy tipikus főprogram az alábbi három részre osztható:

● Előkészítő rész, amelyben kialakítjuk rendszerünk környezetét, azaz megnyitjuk a szükséges fájlokat, inicializáljuk a változókat stb. Ez rendszerint öt egységre bomlik.

- Program fejrész, amely csupa megjegyzés sorokból áll, és a következő információkat tartalmazhatja:
  - a programozó neve
  - a modul neve
  - a modul feladatának rövid leírása
  - az utolsó módosítás dátuma stb.
- A későbbi működési zavarok megelőzése érdekében célszerű törölni a változókat és lezárni a nyitott fájlokat (ugyanis rendszerünk indítása előtt a felhasználó esetleg interaktív módon használta a dBASE IV-et).
- A program futási környezetének beállítása. Ez többnyire SET utasítások sorozata. Előtte a SET függvénnyel lekérdezhetjük, illetve el is menthetjük az aktuális állapotot. Így a program befejezésekor módunkban áll majd visszaállítani azt.
- A szükséges fájlok (beleértve az index-, képernyőformátum-, szűrő- és egyéb fájlokat) megnyitása. Ezt legegyszerűbben a SET VIEW TO utasítással tehetjük meg, ha van egy korábbi elmentett környezetünk.
- A változók és tömbök kijelölése. Ez lehet egyszerű értékadás, de lehet memóriafájlból (.MEM) történő visszatöltés is.

A felsorolt lépések mindössze egyszer, programunk elején kerülnek végrehajtásra. Az alábbi vázlat egy tipikus előkészítő részt illusztrál:

```
Procedure Inic
Public set_var1,...,set_varN && A SET értékek elmentése
set_var1 = Set("STATUS")
.
.
.
set_varN = Set("BELL")
Save To SETFIL All Like set_var*
Clear All
Set ScoreBoard Off      && Új értékek beállítása
Set Help Off
Set Clock Off
Set Deleted On
Set Exclusive Off
Set Status Off
Set Safety Off
Set Talk Off
Set Exact Off
```

```
Set Bell Off
Set Near Off
Clear          && Képernyőtörlés
Return
```

Fontos tudnunk, hogy a **mezőnevek precedenciája nagyobb, mint a változóké**. Ez azt jelenti, hogy ha egy változó neve megegyezik egy megnyitott és kiválasztott adatfájl mezőnevével, akkor minden hivatkozásnál a mező tartalmával dolgozik a dBASE IV, ami nagyon csúnya és nehezen felderíthető programhibákat eredményezhet. Ezért javasoljuk, hogy minden változó (leszámítva természetesen a rendszerváltozókat) neve "v_"-vel kezdődjék. Például a NEV mezőhöz rendelt változó neve legyen v_nev. Általános eljárások (szubrutinok) írásakor azonban nem készülhetünk fel minden eshetőségre (ugyanis lehet, hogy a rutinunkat használó programozó éppen egy v_nev nevű adatmezővel dolgozik), ezért ilyen esetekben használjuk a dBASE IV által is kezelt "M->" prefixet.

- A mag, amely tulajdonképpen a menürendszert tartalmazza, és elindítja a megfelelő alprogramot, ha a felhasználó kiválasztott egy tevékenységet.

- A lezáró rész, amelyben visszaállítjuk a programunk elindítása előtti dBASE IV környezetet.
  - Mentsünk el minden olyan változót, amelyre a későbbi munkánk során szükség lehet.
  - Töltsük vissza a program elején elmentett SET értékeket.
  - Zárjunk le minden fájlt, töröljünk minden változót (CLEAR ALL).
  - Töröljük a képernyőt.
  - Kilépés a programból (a dBASE IV-ből nem feltétlenül).

3.2.1. A programírás eszköze: az editor

Az adatbázis, adatkapcsolatok valamint a rendszer moduljainak részletes megtervezése után kerülhet sor a programok kódolására. Aki már hozzászokott a hazánkban is elterjedt "profi" szövegszerkesztők valamelyikének (Norton Editor, SideKick, Personal Editor, Brief stb...) használatához, valószínűleg továbbra is kitart majd mellettük. Kezdő programozóknak azonban bízvást javasolhatjuk a dBASE IV beépített szövegszerkesztőjét, annál is inkább, mivel a korábbi verziókhoz képest (dBASE II, dBASE III) jelentős fejlesztéseket hajtottak végre rajta.

A MODIFY COMMAND <programnév> utasítás hatására lép működésbe az editor.

Munkaterülete az éppen aktív ablak (lásd ablakkezelés), ha nincs ilyen, akkor az egész képernyő. Munkánkat az alábbi hat menüben összegyűjtött speciális utasítások segítik:

- Layout
- Words
- Go To
- Print
- Exit

A menüket részletesen a Control Center-ről szóló fejezetben ismertetjük, itt csak az editor néhány alapvető tulajdonságát soroljuk fel:

- A megírt program legfeljebb 32000 sor terjedelmű lehet.
- Egy sorban maximum 1024 karaktert helyezhetünk el.
- Megjelölt szövegrészekkel (ún. blokkokkal) különböző műveleteket végezhetünk (másolás, törlés ...). Ez egy fontos előrelépés a korábbi verziókhoz viszonyítva.

A beépített szövegszerkesztőben a navigálás a más szövegszerkesztőkhöz hasonlóan a kurzormozgató, PgUp, PgDn, Home, End, stb billentyűkkel történik. Sok vezérlő billentyűkombináció a WORDSTAR-ral egyezik. Mindezt részletesen megnézhetjük a függelékben a Szerkesztőbillentyűk fejezetben. Most itt csak a legfontosabbakat említjük:

- A munka elmentésére és kilépésre a szinte mindenütt érvényes **Ctrl-End** billentyűket kell használni.
- Elmentés kilépés nélkül a **Ctrl-Return** billentyűkkel lehetséges.
- A szövegblokk kijelölés, áthelyezés, másolás az **F6, F7, F8** billentyűk segítségével történik.

Megjegyzések:
Ha a Control Center Application (alkalmazás) paneljében kiválasztunk egy programnevet, szintén az editort aktivizáljuk.
Ha a beépített editor helyett valamilyen külső szövegszerkesztőt akarunk használni, akkor a CONFIG.DB fájlba be kell illesztenünk a

TEDIT = <editornév>

utasítást.
Ezzel kapcsolatban azonban két dolgot nem szabad szem elől tévesztenünk. Egyrészt meg kell bizonyosodnunk afelől, hogy van-e elegendő memória a dBASE IV és az adott editor számára. Másrészt ha a beépített szövegszerkesztőt használjuk, akkor (SET DEVELOPMENT ON beállítása után) a dBASE IV automatikusan törli a régi .DBO fájlokat és újrafordít minden módosított forrásprogramot(.PRG), míg külső editor alkalmazása esetén ez nem teljesül.

3.2.2. A környezet beállítása, a dBASE IV konfigurálása

Egy alkalmazás során a környezetet bizonyos jellemzőit viszonylag könnyen változtathatjuk (SET utasítások), vannak olyanok azonban amelyeket csak előre állíthatunk be. Ezeket a beállításokat célszerű megtenni, mert így a munkát sok szempontból megkönnyíthetjük és elkerülhetünk komolyabb hibaforrásokat is.

Mielőtt használatba vennénk a dBASE IV rendszerét, valamilyen szövegszerkesztővel érdemes létrehoznunk a kívánalmainknak és lehetőségeinknek megfelelő konfigurációt. Ezt ugyan az üzembehelyező (installáló) program megteszi, azonban csak a legalapvetőbb paraméterek tekintetében. A konfigurációt a **config.db** fájl módosításával állíthatjuk be, amely egy szabványos ASCII szövegfájl, így bármilyen szövegszerkesztővel kezelhető. Megtehetjük ugyanezt valamivel kényelmesebben a programcsomaghoz tartozó **dBSETUP** segédprogram segítségével is, de néhány parancsot mindenképpen szövegszerkesztővel bevinnünk. Ebben a konfigurációs állományban gondoskodhatunk az operatív memória megfelelő felhasználásáról , a környezetjellemzők igényünknek megfelelő beállításáról, a helyes üzemmód megválasztásáról. Ügyeljünk arra, hogy ne lépjük túl az operációs rendszer (config.sys) konfigurációs fájljában beállított értékeket. A később futtatandó program vagy programrendszer "jóságát" nagymértékben befolyásolja a feladatnak megfelelő léptékű, előre átgondolt és helyesen megtervezett konfiguráció. A "túlméretezés" nagyobb helyigényt , az alkalmazási rendszereink lelassulását eredményezi. Ellenkező esetben pedig hibaüzenetekkel találkozhatunk. Ezért fontos az optimális konfiguráció kialakítása.

Vázlatszerűen a következő funkciókra van hatással a konfigurációs fájl:

- Induló utasítás, amellyel automatikusan egy programrendszert is elindíthatunk.
- Az interpreterben a prompt pont karakterét kicserélhetjük más, akár több karakterre is.
- Az induló üzemmód kijelölés (SQL).
- A funkcióbillentyűk értéke.
- Környezeti jellemzők.
- Az operatív memória.
- Pufferterület mérete.
- Memomezők és a programok írásához használt szövegszerkesztő.

3.3. Műveletek fájlokkal

Előrebocsátjuk, hogy ebben a fejezetben a dBASE IV fájl kezelő eszköztárát a programozás szempontjai szerint tárgyaljuk, így az utasítások interaktív formájával illetve kimondottan interaktív jellegű utasításokkal (általában a CREATE / MODIFY kezdetű utasítások) nem foglalkozunk.

3.3.1. Létrehozás, megnyitás, lezárás

● Adatfájlok **létrehozása** programból a következő módokon lehetséges:

Legegyszerűbben **másolással** pl: a COPY FILE utasítással fájl duplikálása, a COPY TO paranccsal egész fájl, vagy a fájl egy részének másolása az aktív adatfájlból, a COPY STRUCTURE TO utasítással az aktív fájl struktúrájának másolása végezhető el; az .MDX, .MEM, .NDX kiterjesztésű fájlokat csak COPY-val lehet létrehozni.

Másképpen néhány utasítás is, így például a CREATE FROM, IMPORT FROM, JOIN WITH vagy EXPORT TO utasítások is új adatfájlt eredményez. A változókat tartalmazó .MEM fájlok a SAVE TO utasítás kiadásakor keletkeznek.

A COPY utasítás az aktív adatfájl meghatározott rekordjait (FOR/WHILE cikkelyek) illetve meghatározott mezőit (FIELDS cikkely vagy a SET FIELDS utasítás) másolja át a megadott nevű fájlba. Az új fájlt (amely lehet nem dBASE IV típusú is) a dBASE IV létrehozza.

Az EXPORT utasítás dBASE IV adatok más rendszerek felé történő közvetítésére szolgál.

A TOTAL utasítással az aktív adatfájl numerikus mezőinek összegzését készíthetjük el valamely indexkulcs szerint. Az eredmény egy újonnan létrejött fájlban keletkezik.

Az UPDATE utasítás lehetővé teszi, hogy az aktív fájl meghatározott mezőit egy másik adatfájl mezőiből alkotott kifejezéssel helyettesítsük, megint csak egy indexkulcs alapján.

A JOIN utasítással pedig két adatfájlból egy harmadikat készíthetünk az eredeti kettő rekordjainak megadott feltételek szerinti összefésülésével.

● Az alábbiakban a különböző típusú fájlok **megnyitási** módjait foglaljuk össze:

- adat- és indexfájl (.DBF,.NDX,.MDX): USE ...[INDEX],SET INDEX TO
- nézetfájl (.VUE, QBE): SET VIEW TO, SET FILTER TO FILE
- képernyőformátum-fájl (.FMT): SET FORMAT TO
- katalógus (.CAT): SET CATALOG TO
- ablak fájl (.WIN): RESTORE WINDOW

● A változtatások biztonságos tárolása érdekében a fájlokat **le kell zárni**, ha egy munkafolyamatban már nincs szükségünk rájuk. A dBASE IV-ből történő kilépéskor (QUIT) vagy a CLEAR ALL utasítás kiadásakor ugyan minden fájl lezárása megtörténik, mégis egyrészt az adatbiztonság másrészt az egyidőben nyitvatartható fájlok korlátozott száma miatt javasoljuk, hogy minden fájlt a lehető legkevesebb ideig tartsunk nyitva, és lezárásukról a programozó gondoskodjék. A fájlok egy részének lezárása automatikusan megtörténik új fájl megnyitásakor. Általában igaz, hogy egy fájl lezárása vagy a megnyitó utasítás paraméter nélküli kiadásával (pl.: USE, SET CATALOG TO stb.) vagy a megfelelő CLOSE típusú utasítás segítségével (pl.: CLOSE ALTERNATE) történik.

3.3.2. Adatfájlok tartalmának változtatási módjai

Alapvetően két csoportba sorolhatjuk a dBASE IV adatmódosító eszközeit aszerint, hogy egy rekordon (esetleg rekord csoporton) vagy egy egész fájlon manipulálnak. Utóbbiakat gyakran kötegelt (batch) utasításoknak is szokták hívni a szakirodalomban. Először a rekordmódosító utasításokat, függvényeket tekintjük át.

● **Új rekordot** az APPEND BLANK és a REPLACE utasítások kombinációjával fűzhetünk a kiválasztott fájlhoz. A rekordmutató a fájl végéhez fűzött új rekordon fog állni. Minden megnyitott indexfájl (beleértve az .MDX fájlokat is) tartalma az adatfájl új rendezettsége szerint módosul. Ha a SET AUTOSAVE környezeti szolgáltatás OFF állásban van, akkor a hozzáfűzött új rekordok csak a memóriapuffer betelésekor íródnak ki a lemezre, míg ON esetén ez azonnal megtörténik. Nyilvánvaló, hogy az utóbbi esetben nagyobb az adatbiztonság, de cserébe (a nagy számú lemezhez fordulás miatt) lassúbb lesz a működés. Mindig az aktuális feladat szabja meg, hogy melyik módszert érdemes választani.

● **Rekordok** vagy rekord csoportok **törlését** a DELETE utasítással valósíthatjuk meg. Valójában ez csak megjelöli a megadott rekordokat, fizikailag nem távolítja el őket a fájlból, így azok a RECALL paranccsal bármikor visszahívhatók. A DELETED() függvény segítségével állapítható meg egy rekordról, hogy törlésre kijelölt vagy nem. A SET DELETED utasítással beállított környezeti paraméter aktuális értéke határozza meg azt, hogy a törlésre kijelölt rekordokat

bizonyos dBASE IV utasítások (pl.: LOCATE,SEEK stb.) figyelembe veszik-e vagy sem. SET DELETED ON esetén a legtöbb utasítás a törlésre kijelölt rekordokat úgy tekinti, mintha azok nem is léteznének. Ez alól a legfontosabb kivétel az INDEX és a REINDEX utasítás. A rekordok fizikai törlése a PACK utasítással valósítható meg. Óvatosan használjuk, mert utána már nincs mód a rekordok visszaállítására. Ha a fájl teljes tartalmát egyszerre kell törölni, akkor a DELETE ALL/PACK utasításpár helyett célszerű inkább a ZAP-ot használni, ugyanis ez sokkal gyorsabb.

● Ha nem a tartalom, hanem maga a fájl válik feleslegessé, akkor az ERASE utasítással töröljük a lemezről. **Adatfájl törlése** nem vonja maga után az esetleg hozzátartozó .DBT és .MDX fájlok törlését, ezeket külön utasítással el kell végezni. Eltérően a DOS DEL utasításától, az ERASE nem ismeri a *,? ún. maszk karaktereket.

● A **batch utasítások** közül elsőként az APPEND FROM-ot említjük meg, amely egy létező adatfájl meghatározott rekordjait az aktuálisan kiválasztott fájl végéhez illeszti. Az input fájl nem csak .DBF lehet, így az APPEND FROM egy lehetséges eszköz más rendszerek (pl.: Lotus, VisiCalc, FrameWork) által létrehozott fájlok tartalmának dBASE IV formátumra hozására. Fontos tudni, hogy SET DELETED ON esetén csak a törlésre meg nem jelölt rekordok kerülnek át az új fájlba, másrészt csak azok a mezők, amelyek mindkét fájlban ugyanazon a néven szerepelnek. Egy másik lehetőség nem dBASE IV típusú fájlok "importjára" az IMPORT utasítás, amelyről az előző alfejezetben már szó esett.

3.3.3. Rendezés, indexelés, keresés

A rendezés az egyik leggyakoribb művelet a számítástechnikában. Ez nem véletlen, hiszen bármely adatművelet vagy valamilyen kulcs szerinti keresésre vagy a módosított adatok valamilyen kulcs szerinti beillesztésére alapszik. Keresni persze lehet szekvenciálisan is (a dBASE IV-ben ezt a LOCATE FOR/CONTINUE utasításokkal tehetjük meg), de ez nyilvánvalóan lényegesen lassúbb működést eredményez, ezért, ha csak lehet, e módszer kerülendő.

A dBASE IV az adatfájlok **fizikai rendezését** a SORT utasítással valósítja meg. Ennek hatására a megadott szempontok szerint rendezett rekordok egy új fájlban keletkeznek. Ekkor tehát teljes rekordok fizikai mozgatásáról van szó, amely különösen nagyméretű fájlok esetén meglehetősen lassú folyamat, nem beszélve arról, hogy a rendezettség fenntartása a továbbiakban a programozóra hárul (új rekord hozzáfűzésekor, kulcs módosításakor vagy rekord törlésekor a fájl rendezettsége általában felborul).

Indexeléskor, azaz **logikai rendezéskor**, az aktív adatfájlhoz tartozó (.NDX vagy .MDX) indexfájlok keletkeznek (Lásd az INDEX ON/REINDEX/SET INDEX TO/SET ORDER TO utasításokat). Az indexfájlok (.NDX vagy .MDX indextagok)

minden rekordra csak a megadott kulcsokat és egy mutatót tartalmaznak. Hogy a két módszer közül melyiket alkalmazzuk, azt a két utasítás lehetőségeinek részletes tanulmányozása után döntse el a programozó. Itt néhány alapvető szempontot emelünk ki.

A **rendezés** (SORT) *hátrányai* közé tartozik, hogy rendezett fájljainkra a SEEK, FIND utasításokat nem használhatjuk. *Előnye* viszont az, hogy (a FOR és WHILE cikkelyekkel) az eredeti fájlnak egy tetszőleges részhalmazát rendezetten hozhatjuk létre az új fájlban, valamint az, hogy összetett kulcs szerinti rendezés esetén lehetőséget nyújt a részkulcsok szerint különböző irányú rendezésre (pl.: név szerint növekvő, de azon belül életkor szerint csökkenő rendezettség létrehozása). Megjegyezzük, hogy mindkét előnyös tulajdonság egy kis trükkel megvalósítható indexelt fájlok esetén is. Ugyanis a rendezett részhalmaz létrehozható az INDEX ON és a COPY utasítások egymás utáni alkalmazásával is, másrészt a különböző irányú rendezettség megvalósítható felhasználói függvények segítségével, ha elég ügyesek vagyunk. Az alábbi példa numerikus típusú mezőkre ad egyszerű ötletet:

Példa:

Tegyük fel, hogy adatfájlunkban szerepel a nevek (NEV) és a fizetések (BER) mező az alábbi definíció szerint:

```
NEV C 30
BER N 5
```

Szeretnénk egy olyan listát, amely név szerint növekvő, de azon belül fizetés szerint csökkenő sorrendben tartalmazná a nyilvántartott személyeket. Nyilvánvaló, hogy erre az INDEX ON utasítás önmagában nem alkalmas. De ha az alábbi

```
NEV + Str(100000-BER,5)
```

kulcsot, indexkifejezést alkalmazzuk, látható, hogy a kívánt eredményt érjük el.

Karakteres és egyéb típusú mezők esetén ez természetesen nem ilyen magától értetődő, de az ötlet alapján elindulva nem nagy fáradsággal készíthetők olyan felhasználói függvények, amelyek megoldják ezeket a problémákat.

Az **indexelés** legnagyobb *előnye* abban rejlik, hogy az adatállománnyal együtt megnyitott indexfájlok tartalma bármely adatváltoztató művelet során úgy módosul, hogy az általa biztosított logikai rendezettség továbbra is fennáll.

A SORT paranccsal ellentétben indexelés során tetszőleges bonyolultságú (pl.: rész karaktersorozatok) kulcsokat is alkalmazhatunk. A SEEK/FIND utasításokkal történő keresés még nagy fájlok esetén is viszonylag gyors. Mivel a SEEK bonyolult kifejezések értékének keresésére is alkalmas, programokban inkább ezt célszerű használni. Végül megemlítjük, hogy a dBASE IV egy újabb hatékony eszközt biztosít adatok keresésére, ez pedig a LOOKUP() függvény.

3.3.4. Több adatfájl párhuzamos használata

Az esetek többségében egyidejűleg több adatfájl-lal kell dolgozunk. Ezeket egymástól elkülönített ún. munkaterületeken nyitjuk meg. Mivel a dBASE IV munkaterületei 1-től 10-ig számozottak, nyilvánvaló, hogy egyidőben legfeljebb 10 adatfájlt használhatunk. Ez csak látszólag mond ellent annak, hogy dBASE IV-ben az egyszerre nyitva tartható fájlok maximális száma 99. Ugyanis ez a korlát nem csak az adatfájlokra vonatkozik, és egy aktív adatfájlhoz még jó néhány egyéb típusú fájlt is megnyithatunk (pl.: index-, formátum- vagy nézetfájlok).

A 10-es munkaterület a katalógusok számára foglalt. Ez azt jelenti, hogy bármely katalógus mindig ezen a munkaterületen nyílik meg, s az ott esetleg nyitott fájl automatikusan lezárul, másrészt, amíg van nyitott katalógus, addig a 10-es munkaterületen adatfájlt nem tudunk megnyitni.

A dBASE IV indulásakor az aktív munkaterület az 1-es. Ezt a SELECT utasítással változtathatjuk meg. A SELECT() függvénnyel pedig az első szabad munkaterület számát kérdezhetjük le. A munkaterület váltása az egyes fájlok rekordmutatóinak állását nem befolyásolja.

3.3.5. Adatfájlok összekapcsolása

Adatfájljaink logikai összekapcsolásával adatfájlt hozhatunk létre. Ezt a SET RELATION TO utasítás segítségével tehetjük meg. Ekkor az aktív adatfájlhoz egy vagy több, más munkaterületeken már megnyitott adatfájlt rendelhetünk hozzá egy közös mezőn keresztül. E mező szerint a hozzárendelt fájloknak indexelve vagy rendezve kell lenniük. A hozzárendelt fájlokhoz hasonlóképpen újabbakat rendelve relációs láncokat alakíthatunk ki. Ez tulajdonképpen egy ún. fa szerkezet, amelynek gyökerében az eredetileg aktív fájl áll. Ettől kezdve a gyökérfájlban elmozduló rekordmutató az összes kapcsolódó fájl rekordmutatójának elmozdulását is kiváltja, mégpedig a kapcsolómezők aktuális értékének megfelelően. Erre a folyamatra hatást gyakorol a SET SKIP, SET DELETED, SET FIELDS környezeti paraméterek aktuális értéke is.

3.3.6. A környezet elmentése

Nagyon sok fájl egyidejű használatát igénylő feldolgozások során előfordulhat, hogy nem tudunk tovább lépni addig, amíg néhány fájlt le nem zárunk (pl.: ha a "Too many files are open" hibaüzenet jelenik meg). A folyamat egy későbbi szakaszában viszont esetleg éppen így vagy nagyon hasonló felépítés szerint lehet szükségünk az adatfájlok, indexfájlok és egyéb elemek alkotta környezetre, ezért jó lenne, ha ennek elmentésére és későbbi visszaállítására lenne mód. Nos, ezt valósítja meg a CREATE VIEW FROM ENVIRONMENT utasítás. A dBASE IV egy .VUE kiterjesztésű fájlban tárolja az aktuális fájl-környezetről az alábbiakat:

- minden megnyitott adatfájl, indexfájl s a hozzájuk tartozó munkaterület számok;
- minden létező reláció;
- az aktív munkaterület száma;
- az aktuális mezőlista (SET FIELDS);
- minden aktív filter kifejezés;
- minden aktív formátumfájl.

A felsorolt információk egy tetszőleges későbbi időpontban a SET VIEW TO utasítással visszaállíthatóak.

3.3.7. Tranzakciók

Az eddigi dBASE verziók egyik legnagyobb hiányosságaként tüntették fel azt, hogy hosszabb adatfeldolgozási folyamatok megszakadásakor nem adtak lehetőséget a hibás adatok visszaállítására. A dBASE IV tartalmaz egy eszköztárat a BEGIN TRANSACTION, END TRANSACTION, ROLLBACK, RESET utasítások és a ROLLBACK(), COMPLETED(), ISMARKED() függvények formájában, amelyeket az ún. tranzakciós feldolgozás során használunk. A tranzakció tulajdonképpen nem más, mint a BEGIN TRANSACTION ... END TRANSACTION utasításpár közt felsorolt (elsősorban fájlmanipulációs) dBASE IV utasítások összessége. Megjegyezzük, hogy egy megkezdett tranzakciónak még ugyanabban a programban illetve eljárásban be kell fejeződnie. A BEGIN TRANSACTION hatására a dBASE IV egy ún. **naplófájl**ban elkezdi jegyezni a tranzakció során végbement elemi adatműveletek eredményeit.

Elemi adatműveletnek nevezünk minden olyan utasítást, amely:

- új rekordokat hoz létre;
- rekordok tartalmát módosítja;
- rekordokat törlésre kijelöl;
- törlésre kijelölt rekordokat visszaállít.

A naplófájl neve egyfelhasználós környezetben "TRANSLOG.LOG", míg hálózati környezetben <munkahelynév>.LOG, ahol a <munkahelynév> annak a munkaállomásnak a hálózati neve, amely elindította a tranzakciót. Ha nem adunk meg útleírást (path), akkor a naplófájl az aktuális alkönyvtárban keletkezik.

A naplófájl tehát tartalmazza az adatfájlokban és az indexfájlokban végbemenő összes változást. A dBASE IV ezenkívül beállít egy jelet minden résztvevő adatfájl fejrészében, amely azt jelzi, hogy az adott fájl tranzakció alatt áll. Ha a tranzakció sikeresen befejeződik (ennek eldöntése természetesen a programozó dolga), adjuk ki az END TRANSACTION utasítást. Ennek hatására a naplófájl törlődik, s a résztvevő adatfájlok fejrészében beállított jel is alaphelyzetbe áll. A naplófájl törlése után az egyetlen mód az eredeti állapot helyreállítására a mentéslemezek visszatöltése.

Ha a **tranzakció megszakad** valamilyen ok (pl.: áramkimaradás, hiba a lokális hálózatban, megtelik a lemez stb.) miatt, adjuk ki a **ROLLBACK** utasítást. Ennek hatására a dBASE IV a naplófájl alapján **visszaállítja** a tranzakciót megelőző állapotot. Új tranzakciót nem indíthatunk addig, amíg az előző sikeresen le nem zárult és a naplófájl nem törlődött, vagyis a tranzakciók egymásbaágyazása nem megengedett.

A USE utasítás egy fájlra csak akkor lehet sikeres, ha a fájl fejrészében tárolt tranzakciós jel megfelelő állapotban van (azaz a fájl nem egy félbeszakadt tranzakcióban vett részt, vagy éppen tranzakciós feldolgozás alatt áll). Ellenkező esetben figyelmeztető üzenet jelenik meg, s ekkor csak egy ROLLBACK vagy egy RESET utasítás kiadása után haladhatunk tovább.

Ha egy tranzakcióban már létező fájlokat törlünk vagy felülírunk, illetve megnyitott fájlokat lezárunk, akkor a ROLLBACK nem lehet sikeres, ezért néhány alapvető szabályt meg kell említenünk a különböző utasítások tranzakción belüli használhatóságát illetően.

● Az elemi műveleteket végző alábbi utasítások korlátozás nélkül használhatók:

APPEND	BROWSE	CHANGE
DELETE	EDIT	RECALL
REPLACE	UPDATE.	

● Az alábbi utasítások olyanok, amelyek új fájlokat hoznak létre. Tranzakcióban ezek csak akkor megengedettek, ha nem írnak felül létező fájlokat.

COPY STRUCTURE EXTENDED	CREATE ... FROM
EXPORT TO ...	IMPORT FROM ...
INDEX ... TO ...	JOIN ... TO ...
SET CATALOG TO ...	SORT ... TO ...
TOTAL ... TO ...	

● Az alábbi utasítások tranzakcióban csak akkor vehetnek részt, ha nem zárnak le korábban megnyitott fájlokat:

CREATE FROM ...	IMPORT FROM ...
INDEX ON ...	SET CATALOG TO ...
SET INDEX TO ...	USE

● Az itt következő utasítások tranzakcióban egyáltalán nem használhatóak

CLEAR ALL	CLOSE ALL/DATABASE/INDEX
DELETE FILE	ERASE
INSERT	MODIFY STRUCTURE
PACK	RENAME
ZAP	

Ha a legutóbb felsoroltak valamelyikét mégis megpróbáljuk használni egy tranzakcióban, a dBASE IV hibát fog jelezni.

A lokális hálózatokat ismertető fejezetben látható egy rövid programrészlet, amely a tranzakciók kezelését demonstrálja.

3.4. A nyomtató vezérlése

3.4.1. Rendszerváltozók

Néhány speciális célú, kiemelt jelentőségű változót is találhatunk a dBASE IV-ben. Ezeket rendszerváltozóknak nevezzük. A rendszer indításakor felveszik alapértelmezés szerinti értékeiket, amelyek természetesen bármikor megváltoztathatóak. A dBASE IV szerves részét képezik, ebből következően darabszámuk és neveik adottak. Hogy ne lehessen összekeverni őket a hagyományos változókkal, neveik "_" (aláhúzás) karakterrel kezdődnek. Más változó nem kezdődik ilyen jellel, és nem is lehet ilyent létrehozni. Rendszerindításkor a dBASE IV önműködően inicializálja a rendszerváltozókat.

A rendszerváltozók vezérlik a nyomtatón vagy a képernyőn történő megjelenítés formáját. Konkrétabban:

- Egy nyomtatási köteg (PRINTJOB) formátumát (például soremelések vagy a kinyomtatott szöveg példányszáma).
- Egy bekezdés megjelenését, (például a sorigazítás és a margók).
- A kinyomtatott oldal megjelenési formáját, (például a betű típus, a nyomtatás minősége vagy az oldal eltolásának mértéke).

A változók értéke módosítható a listatervezőn és a címketervezőn keresztül (MODIFY REPORT, MODIFY LABEL), de az interpreter alatt közvetlenül is. A CLEAR MEMORY és a RELEASE utasítások nem törlik ki őket a számítógép memóriájából.

A rendszerváltozókra a közönséges változókkal megegyező szabályok vonatkoznak. Ha például egy programban a rendszerváltozókat lokálisnak (PRIVATE) deklarálta, akkor a program végeztével ezek a változók visszanyerik eredeti értéküket.

A legtöbb rendszerváltozó az adatfolyam jellegű nyomtatásra hat (streaming output). Minden kimenő adatot előállító utasítás adatfolyamot készít, kivéve a @, @...TO, EJECT utasításokat. Az adatfolyam célja lehet a képernyő, a nyomtató esetleg egy kijelölt fájl. Ezt a SET CONSOLE, SET PRINTER, SET ALTERNATE utasításokkal lehet kiválasztani.

Az adatfolyam a képernyő aktuális pozíciójánál kezdődik, nyomtató esetén a nyomtatófej helyénél, illetve fájl esetén a fájlmutató által kijelölt helyen. Habár a @, @...TO, EJECT utasítások nem gyártanak adatfolyamot mégis hatással lehetnek az aktuális pozícióra. Például ha egy @ utasítás megváltoztatja a kurzor helyét a képernyőn, akkor az hatással van a későbbi ?/?? utasításokra.

A következő részben csak címszavakban tárgyaljuk a rendszerváltozók működését. Ezeknek végén egy példaprogramot is találhatunk, amely kilistázza a *személy.dbf* fájlt. A program induláskor több kérdést tesz fel. Először érdemes az ajánlott értékeket kipróbálni, majd újabb és újabb paraméterekkel lefuttatni a programot. Ez a módszer segíthet a rendszerváltozók hatásmechanizmusának jobb megértésében. A rendszerváltozók bővebb leírása a függelékben található.

3.4.1.1. Az adatfolyamot befolyásoló rendszerváltozók

_box	A DEFINE BOX által kijelölt keret megjelenését vezérli. (A @ TO utasítással létrehozott keret a közvetlen pozícionálás miatt elrontaná az adatfolyam megjelenését)
_pageno	Az éppen nyomtatás alatt álló oldal számát tartalmazza, amely értékadással módosítható. Lapdobás esetén automatikusan növekszik az értéke.
_pcolno	Az írófej aktuális oszloppozícióját tartalmazza, amely módosítható. Ellentétben a PCOL() függvénnyel SET PRINTER OFF esetén is helyesen működik.

_pform — A címke- és a listatervezésnél beállított és (ebbe a változóba) elmentett nyomtatási jellemzőket tartalmazza. Segítségével az összes nyomtatási paramétert helyre lehet állítani a mentés előtti állapotnak megfelelően.

_plength — Az oldal sorainak számát, azaz az oldal hosszát tartalmazza, amely módosítható. Ha nem gondoskodunk az ON PAGE utasítás segítségével az oldalváltás kezeléséről, akkor a dBASE IV önműködően lapot emel.

_plineno — Az írófej aktuális sorpozícióját tartalmazza, amely módosítható. Ellentétben a PROW() függvénnyel SET PRINTER OFF esetén is helyesen működik.

_pspacing — A kinyomtatott sorok közötti soremelés mértékét tartalmazza.

_tabs — A tabulátorpozíciókat tartalmazza. Elsősorban táblázatokkal kapcsolatban célszerű használni.

3.4.1.2. A nyomtató paramétereivel kapcsolatos rendszerváltozók

_padvance — Meghatározza a lapemelés módját, azt, hogy a lapdobás az ASCII 12-es kódú karakter kiküldésével történjen, vagy soremelések (ASCII 10) sorozatával.

_pdriver — Az éppen aktuális nyomtatómeghajtó (PRINTER DRIVER) nevét tartalmazza, amely (más nyomtató esetére) értékadással megváltoztatható. A dBASE IV számos nyomtatómeghajtót kínál installáláskor, így lehetőség van szinte az összes ismertebb nyomtató használatára, beleértve a lézer nyomtatókat is.

_ploffset — A bal alapmargó nagyságát tartalmazza. A SET MARGIN TO utasítással egyenértékű. Csak a nyomtatásnál hatásos.

_ppitch — A nyomtatott karakterek szélességét, azaz a karaktersűrűséget tartalmazza, állítja be. Segítségével például 132 karakter széles lista is kinyomtatható 80 pozíciós nyomtatón.

_pquality — A minőségi nyomtatást kapcsolja ki/be.

_pwait — Használatával elérhető, hogy a nyomtatást a dBASE IV minden lapemelés után felfüggeszti, lehetővé téve az esetleges kézi lapadagolást.

3.4.1.3. A nyomtatási köteggel (PRINTJOB) kapcsolatos rendszerváltozók

_pbpage, _pepage — Segítségükkel meghatározható, hogy a PRINTJOB futása közben melyik legyen az első és utolsó kinyomtatott oldal. Hasznos lehet, ha a listának csak egy részét szeretnénk kinyomtatni.

_pcopies — A nyomtatási kötegre érvényes példányszámot határozza meg.

_pscode, _pecode — Nyomtatóvezérlő karaktereket (szekvenciákat) tartalmaz, amely a nyomtatási köteg megkezdése előtt (_pscode) illetve után (_pecode) kerülnek a nyomtatóra. Használhatóvá válnak a nyomtató által ismert, de a nyomtatómeghajtó által nem támogatott lehetőségek. Használatához az alkalmazott nyomtató kézikönyve szükséges.

_peject — Beállítja, hogy a PRINTJOB előtt és/vagy után történjen-e lapdobás.

3.4.1.4. Bekezedés-orientált rendszerváltozók

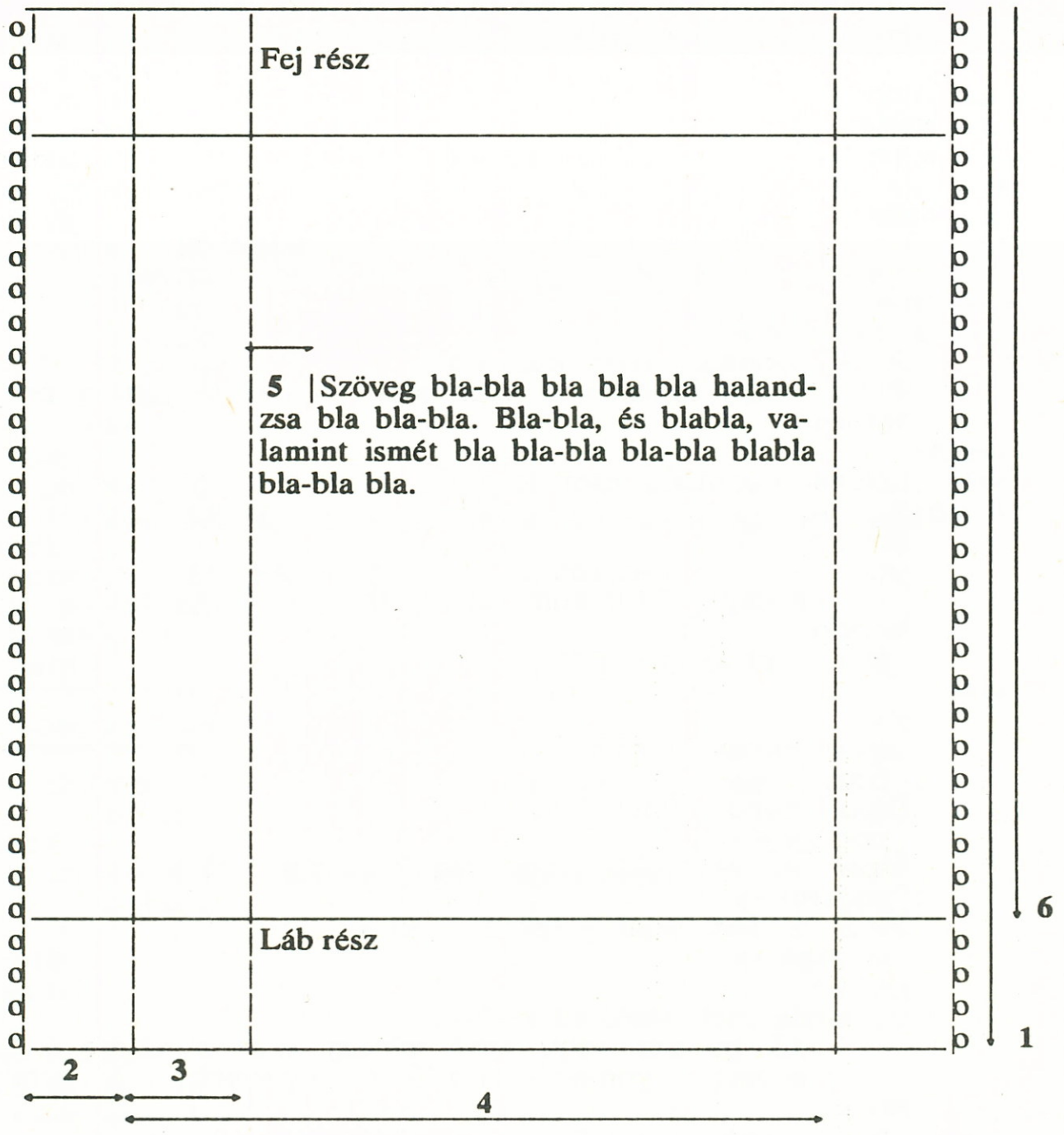
A következő rendszerváltozók az adatfolyamot (stream output) bekezedések szerint formálják. Ezek segítségével egyszerű szövegkezelési feladatok végezhetők el.

_wrap — Ha értéke .T., akkor a megjelenített szövegek szavait sor végén nem töri ketté, hanem az egész szót új sorban kezdi. A következő négy változó hatása is csak ekkor érvényesül.

_alignment — Beállítja, hogy a bekezdés egyes sorai balra, jobbra vagy középre igazítva jelenjenek meg az aktuálisan érvényes bal és jobb margó között.

_indent — A bekezdés első sorának a bal (alap)margóhoz viszonyított "behúzásának" mértékét határozza meg.

_lmargin, _rmargin — A bal illetve a jobb margó beállítására szolgálnak. Lásd a következő ábrát.



3.1. ábra

1 – _plength

2 – _ploffset

3 – _lmargin

4 – _rmargin

5 – _indent

6 – ON PAGE és _plength-tel együtt kezelhető.
Például ON PAGE AT LINE _plength-4, ahol a láb rész 4 sorból áll.

3.4.1.5. Példaprogram a rendszerváltozók alkalmazására

```
set echo off
set status off
set score off
set talk off
set safety off
private _plength,_pform,_pcopies,_pbpage,_pageno,;
_pepage,_plength,_pdriver,_wrap,_peject,_pwait,_pquality,;
_plineno,_ppitch
if .not. printstatus()
        ? "A nyomtató nem működik!"
        ? "Helyezze üzembe a nyomtatót, és inditsa újra a programot!"
        return
endif
Wait "Létezik nyomtató formátum fájl ?" to YN
if Yn $ "Yy"
        prnform = ""
        do while .not. file(prnform + ".prf")
                accept "Formátum fájl ?" to prnform
        enddo
        _pform = prnform + ".prf"
else
        x=0
        input "Példány szám ? (pl:2) " to x
        _pcopies = x
        input "Kezdő oldal ? (pl:1) " to x
        _pbpage = x
        input "Az elso oldal sorszáma ? (pl:10) " to x
        _pageno = x
        input "Utolsó oldal ? (pl:13) " to x
        _pepage = x
        prndrv = ""
        do while .not. file(prndrv + ".pr2")
                && generic vagy ascii mindig lehet a válasz
                accept "Nyomtató vezérlő ? (pl:generic) " to prndrv
        enddo
        _pdriver = prndrv + ".pr2"
        input "A lap hossza ? (pl:8) " to x
        _plength = x
endif
_wrap = .F.          && A bekezdés formázó funkciót ki kell kapcsolni
_peject = "both"  && Lapdobás legyen a nyomtatás elött és utána is
_pwait = .F.          && Az egyes lapok között nem kell szünetet tartani
_pquality = .T.     && Minőségi nyomtatás legyen
if .not. file("szemely.dbf")
        ? "Hianyzik a listazando szemely.dbf allomany!"
        return
endif
use szemely
&&set console on
set console off
set printer on
```

```
_plineno = 0
set spac on
set mark to "."
set century on
printjob                    && PRINTJOB kezdete
on page at line _plength-3 do lab_fej
do fej
do while .not. eof()
        ? nev ,
        ?? szsz ,
        ?? szuldat ,
        ?? irany ,
        ?? cim
        skip
enddo
if _plineno  0
eject page && láb rész kiiratását aktivizálja az on page-el
endif
endprint    && PRINTJOB vege
set printer off
set statu on
return
*******************
procedure lab_fej
*******************
if _plineno  0     && Ha nem volt még láb rész
        do lab
endif
if .not. eof ()    && Ha van még rekord
        do fej
        ?
endif
return

***************
procedure fej
***************
        ? "Nev",
        ?? "szem. szam" at 22,
        ?? "szuletes" at 35,
        ?? "cim" at 46
        ? replicate ("_",84)
return
**************
procedure lab
**************
? _pageno at 10
?? ". oldal"
eject page
return
```

```
. disp stru

Structure for database: A:\KG\SZEMELY.DBF
Number of data records:     14
Date of last update    : 1989.05.30
Field  Field Name  Type        Width    Dec    Index
1      NEV         Character      20          N
2      SZSZ        Numeric        11          Y
3      SZULDAT     Date            8          N
4      IRANY       Numeric         4          N
5      CIM         Character      35          N
6      FERJ_NOS    Logical         1          N
** Total **                      80

. list

Record#  NEV                  SZSZ          SZULDAT   IRANY CIM
FERJ_NOS
      1  kovacs geza          16210251234   1962.10.25 2500 Pilisszentlaszo      Voros
marty u 34 .F.
      2  nagy agnes           26702116324   1967.02.11 1987 Budapest  XIII ker.  Ja
cint u 3   .F.
      3  szalai aranka        23505096463   1935.05.09 5921 Pecs      voroshadsereg ut
ja 142     .T.
      4  pap laszlo           17212310001   1972.12.31 3713 Hatvan    Setany koz 13
           .F.
      5  lachazi peter        39804129393   1898.04.12 6292 nagykoros csalogany u
t 76       .F.
      6  kiss peter           15711128584   1957.11.12 5251 gyor      kovacs laszlo ut
ca 45.T.
      7  kissne nagy eva      25901158442   1959.01.15 5251 gyor      pogany ut 82
           .T.
      8  gergely eva          27111254547   1971.11.25 6122 szentes   kikirics ut 8
2          .T.
      9  szilagy attila       13410262345   1934.10.26 2343 Koszeg    buzogany ut 22
           .T.
     10  pongracz elemer      14411112115   1944.11.11 1626 budapest  lenin korut
232        .T.
     11  zsuga zsolt          12301134534   1923.01.13 2346 szentendre dunakanyar
korut 31   .F.
     12  hevesi antalne       22808142342   1928.08.14 3424 debrecen  harpia utca
17         .T.
     13  kelemen lajos        13212284081   1932.12.28 8903 szeged    arpa utca 7
           .F.
     14  nemet ibolya         23702084492   1937.01.08 8903 szeged    csenderi ut 95
           .F.
```

3.4.2. A dBASE IV nyomtatómeghajtói (printer driver)

A dBASE IV rugalmasságát mutatja az is, hogy (megfelelő rendszer konfigurálás esetén) különféle típusú nyomtatókkal képes működni. Az installáló lemezcsomagban számos ún. nyomtatómeghajtó (printer driver) fájl található. Mint tudjuk, a nyomtatás jellemzőit (karakter sűrűség, sortávolság, a különböző betűtípusok: fél-

kövér (bold), dőlt (italic), aláhúzott (underlined), felső index (superscript), alsó index (subscript).stb.) ún. **ESC szekvenciák** vezérlik. Ezek speciális, nyomtatónként meghatározott kódsorozatok. Nos, a nyomtatómeghajtók a nekik megfelelő nyomtatóval kapcsolatos vezérlő kódsorozatokat tartalmazzák.

A dBSETUP program futtatásakor megadható, hogy milyen típusú nyomtatókkal kívánunk dolgozni, s a megfelelő driver installálását elvégzi a dBSETUP. A nyomtatómeghajtók hozzárendelése azonban a rendszer működése közben is szabadon változtatható a CONFIG.DB PDRIVER paraméterének vagy a _pdriver rendszerváltozó értékének megfelelő megválasztásával.

3.4.3. A CONFIG.DB nyomtatást vezérlő paraméterei

Ha a dBSETUP futásakor nem installáltuk nyomtatónkat, a dBASE IV induláskor a "GENERIC.PR2" meghajtót rendeli hozzá. Ez a fájl csak bizonyos alapvető vezérlőkódokat tartalmaz, így nyomtatási lehetőségei korlátozottak (pl.: hasonlóan a dBASE korábbi verzióihoz, csak két speciális betűtípust ismer). Ha nyomtatónk ennél lényegesen többre képes (ez a nyomtatóhoz mellékelt kézikönyvből derülhet ki), és az installáló lemezen létezik hozzá megfelelő nyomtatómeghajtó, akkor egyrészt a már említett dBSETUP segítségével valósíthatjuk meg a kívánt hozzárendelést, másrészt a CONFIG.DB fájl PDRIVER utasításával ez később bármikor módosítható.

Megjegyzés: A dBASE IV indulásakor megvizsgálja, hogy a CONFIG.DB-ben szerepel-e a PDRIVER utasítás. Ha nem, akkor az installáláskor beállított driver-t használja, ha igen, akkor az ott megadottak szerint jár el. A rendszer működése közben azonban még a CONFIG.DB által meghatározott driver is felülbírálható a _pdriver rendszerváltozó értékének megfelelő megválasztásával. Fontos tudni, hogy a megadott driver-nek léteznie kell a dBASE IV alkönyvtárban vagy az aktuális PATH-ban ahhoz, hogy hatását kifejthesse.

Egyidejűleg négy különböző nyomtatómeghajtót és minden nyomtatóhoz legfeljebb öt különböző betűtípust határozhatunk meg a PRINTER utasítás segítségével.

Az utasítások pontos szintaxisát a konfigurálásról szóló függelékrész tartalmazza.

3.4.4. Adatok megjelenítése nyomtatón

A dBASE IV számtalan lehetőséget biztosít adataink nyomtatón történő megjelenítésére. Ezeknek egy része kimondottan interaktív jellegű (pl.: DISPLAY,LIST), másik részét külön alfejezet tárgyalja (pl.: REPORT). Itt most inkább a programozási munkát támogató utasításokat, függvényeket tekintjük át. Az utasítások

pontos szintaxisa a függelék részben olvasható, itt csak utalunk az utasításnevekre, miközben alkalmazhatósági szempontjaikat tárgyaljuk.

Alapvetően két típusú nyomtatást kell megkülönböztetnünk. Egyiket a ?/??,LIST stb... utasításokkal valósítjuk meg. Az angol szakirodalomban ez "streaming output"-ként terjedt el. Megfelelően pontos magyar kifejezés hiányában a továbbiakban **adatfolyamnak** nevezzük.
A másik típust a teljes-képernyős (full-screen) utasításokkal (@) valósítjuk meg. A legszembetűnőbb különbséget köztük az jelenti, hogy míg az első típus esetén a nyomtatott sorok szigorú szekvenciális rendben követik egymást, a második típusú nyomtatás során a lapon az írófej pozíciója direktben címezhető (sor,oszlop koordináták). Meg kell azonban jegyeznünk, hogy tényleges direkt címzésről csak megfelelő nyomtató birtokában beszélhetünk (amely képes a fejet visszafelé is mozgatni egy lapon belül), ellenkező esetben a lapon belüli PROW(),PCOL() értékeknél kisebb koordinátára történő írási kísérlet lapdobást eredményez.

A két típus megkülönböztetésének ennél lényegesebb oka is van. A dBASE IV bizonyos utasításai, rendszerváltozói ugyanis eltérően viselkednek attól függően, hogy melyik nyomtatási módot alkalmazzuk. Először a mindkét típusban ugyanúgy működő utasításokról beszélünk, majd két rövid alfejezetben áttekintjük a különbségeket is.

Mielőtt programból nyomtatni próbálnánk, célszerű meggyőződni arról, hogy a nyomtató üzemképes állapotban van-e. Ezt a PRINTSTATUS() függvény segítségével tehetjük meg, mint az alábbi példa is mutatja:

```
prnok = " "
Do While prnok = " "
      @ 23,3 Say "Üzemkész a nyomtató ? [I/N]";
            Get prnok Pict "!"
      Read
      If At(prnok,"IN") = 0 .Or. .Not. PrintStatus()
            prnok = " "
            Loop
      EndIf
EndDo
If prnok
      nyomtatási utasítások
EndIf
```

A SET MARGIN utasítás minden nyomtatott output esetén a nyomtató bal margójának ún. bázisértékét állítja be. Ezt nevezzük alapmargónak. Ugyanerre szolgál a _ploffset rendszerváltozó is. A két utasítás közül mindig az utoljára kiadottnak a hatása érvényes.

Példa:

```
.Set Margin To 10
.? _ploffset
10
._ploffset = 3
```

A harmadik utasítás után az alapmargó 3. Az alapmargó (offset) elnevezést az indokolja, hogy _wrap = .T. esetén a tényleges bal margót a _ploffset és az _lmargin értékeinek összege adja.

Az EJECT utasítás hatására a nyomtató fej a következő lap tetejére áll. Ez a _padvance rendszerváltozó aktuális értékétől függően következik be. A _padvance = FORMFEED esetén a szabványos lapdobás történik (ASCII 12-es kód), míg _padvance = LINEFEEDS esetén annyi soremelést (ASCII 10-es kód) hajt végre a dBASE IV, amennyi a következő lap tetejére álláshoz szükséges (ezt a _plength-_plineno különbség adja meg).

Ahány listázó program, annyiféle betűtípust, laptördelést, margót stb. használ nyomtatás közben. Ehhez persze át kell programozni a nyomtatót, azaz a memóriáját (RAM-ját) fel kell tölteni a megfelelő vezérlőkód-kombinációkkal. Egy "tisztességes" program lefutása után alapállapotba állítja vissza a nyomtatót. Bizonyos listák több példányban készülnek, ekkor ciklust kell szervezni, és még sorolhatnánk néhány (persze nem túl nagy) problémát, amelyet egyetlen utasításpárral megoldhatunk dBASE IV-ben. Ez pedig a PRINTJOB/ENDPRINTJOB, amely az azonos jellemzőkkel kinyomtatandó ún **nyomtatási köteg** utasításait fogja egybe.

3.4.4.1. Nyomtatás teljes-képernyős utasításokkal

A SET DEVICE TO PRINTER utasítás kiadásától kezdve minden @ ... SAY ... utasítás eredménye a nyomtatón jelenik meg. Ebben az esetben a GET utasításokat a dBASE IV figyelmen kívül hagyja.

3.4.4.2. Adatfolyam-nyomtatás (streaming output)

A SET PRINTER ON utasítás hatására minden nem teljes-képernyős utasítással létrehozott kimenő adat az aktuálisan hozzárendelt nyomtatón is megjelenik. Nyomtatás közben a képernyőre írást a SET CONSOLE OFF paranccsal megtilthatjuk. Ez az utasítás a teljes-képernyős utasításokkal történő nyomtatás esetén hatástalan.

Adatfolyam nyomtatásánál gyakori alkalmazás a láb- és fejrészek nyomtatásához az ON PAGE utasítás, amelyről az eseménycsapdáknál ejtünk szót.

3.4.5. Néhány hasznos tanács a nyomtatással kapcsolatban

● Meglehetősen gyakran előforduló jelenség, hogy a lista utolsó sora nem jelenik meg. Ez az esetek többségében azt jelenti, hogy a hiányolt szövegrész a nyomtató pufferében maradt. Ez a kellemetlenség megelőzhető, ha minden nyomtatást végző eljárást vagy az EJECT paranccsal vagy (ha nem akarunk lapdobást) a ??? "{RETURN}" utasítással zárunk le, de elegendő lehet egy "üres" ? utasítás is, mert ez is soremelést s ezzel együtt a puffer kiürítését eredményezi.

● Ha a nyomtatófejet újra helyesen szeretnénk beállítani, adjunk ki egy EJECT utasítást.

● Ha a REPORT FORM utasítással történő nyomtatás egy nem várt lapdobással kezdődik, használjuk a NOEJECT opciót.

● Ha a listázó program, amelyet a @...SAY utasításokkal készítettünk el, futtatása közben egy általunk nem specifikált lapdobással kezdődik, akkor valószínűleg az történt, hogy a program behívása előtti aktuális nyomtató (vagy képernyő) sor-koordináták (PROW()/ROW()) értéke nagyobb, mint ami az általunk kiadott első @ ...SAY utasításban szerepel. Mivel a nyomtatók többsége nem tud visszafele pozicionálni, az utasításunk valóban csak egy lapdobás után végrehajtható. Azonban ez a jelenség is megelőzhető, ha az alábbiak szerint járunk el:

● A SET DEVICE TO PRINTER utasítás előtt a listázó programban szerepeljen a @ 0,0 utasítás.

● A SET DEVICE TO SCREEN utasítás előtt pedig feltétlenül adjunk ki egy EJECT utasítást.

3.4.6. A nyomtató vezérlésével kapcsolatos utasítások és függvények felsorolása

Csupán felsorolásszerűen összegyűjtöttük a nyomtató vezérléséhez valamilyen formában kapcsolódó utasításokat, függvényeket.

Utasítások

?/??	Egy vagy több kifejezés értékét jeleníti meg nyomtatón, ha SET PRINT ON
???	Egy kifejezést direkt módon a nyomtatóra küld, megkerülve az aktuálisan hozzárendelt driver-t.
DEFINEBOX	A nyomtatott szövegsorok köré keretet rajzol.

EJECT	A nyomtatófejet a következő lap elejére állítja.
EJECT PAGE	Az ON PAGE használatától függően állítja be a nyomtatófejet.
ENDPRINTJOB	A PRINTJOB által elindított nyomtatási köteg végét jelöli ki.
ON PAGE	Lehetővé teszi, hogy lap vége esetén elindítsunk egy tevékenység sorozatot.
PRINTJOB	Egy nyomtatási köteg elejét jelöli ki.

Függvények

PCOL()	A nyomtató aktuális oszlopának számát adja vissza.
PRINTSTATUS()	Értéke logikai igaz (.T.), ha a nyomtató üzemkész állapotban van.
PROW()	A nyomtató aktuális sorának számát adja vissza.

SET utasítások

SET ALTERNATE ON/OFF	A képernyőn adatfolyamként megjelenő adatok ON esetén a SET ALTERNATE TO utasítás által meghatározott fájlba is bekerülnek.
SET ALTERNATE TO	Létrehozza a szövegfájlt, amelyet a SET ALTERNATE ON paranccsal aktivizálhatunk.
SET DEVICE	A teljes-képernyős utasítások eredményének megjelenési helyét határozza meg (képernyő/nyomtató/fájl).
SET MARGIN	A bal margó nagyságát állítja be a nyomtatón.
SET PRINTER ON/OFF	A nyomtató ki-be kapcsolását végzi.
SET PRINTER TO	A nyomtatásra használni kívánt DOS vagy hálózati eszköz kiválasztására szolgál.
SET SPACE ON	esetén a ?/?? utasításokkal megjelenített kifejezések között egy üres helyet (space) hagy.

A pontos szintaxis és a részletes leírás a függelékben megtalálható.

3.5. Adatkatalógus

A dBASE IV használata közben adatfájlok, a hozzájuk tartozó indexfájlok, jelentés (report) fájlok, formátum fájlok és egyéb más, különböző rendeltetésű fájlok keletkeznek. Ezek szervezett használatát segíti a katalógusrendszer. Egy katalógus tulajdonképpen valamilyen szempontból összetartozó fájlok jellemző információit (pl.: név, elhelyezkedés a lemezen, rövid leírás) tartalmazza. Valójában ezeket az összetartozó fájlokat nevezhetnénk adatbázisnak. Az összetartozás szempontjait természetesen a felhasználó alakítja ki, ő határozza meg, hogy mely fájlok mely katalógushoz tartozzanak. Például célszerű a SET RELATION paranccsal adatbázissá összekapcsolt fájlokat egy katalógusban nyilvántartani.

A katalógusok egyik fő előnye, hogy a Control Center-ben és egyéb fájl-listákban csak a konkrét munkához szükséges (az aktuálisan megnyitott katalógusba tartozó) fájlnevek jelennek meg, amivel az áttekinthetőség jelentősen nő.

3.5.1. Katalógusok előállítása, szerkezete

Adatkatalógust a SET CATALOG TO <fájlnév> paranccsal hozhatunk létre. A fájlnév lesz a katalógus neve, kiterjesztése mindig .CAT. Ha a megadott néven már létezik katalógus, akkor a dBASE IV megnyitja azt, ellenkező esetben újat hoz létre. Az első katalógus előállításakor keletkezik egy CATALOG.CAT nevű fájl, amely a katalógusokat tartja nyilván. Ez tulajdonképpen a katalógusok katalógusa. Ha SET TITLE ON állásban van, akkor az új katalógus rövid leírását is meg kell adnunk. Ez csak a könnyebb azonosítást szolgálja (pl.: a ? opció használatakor ezek a rövid leírások jelennek meg a fizikai név helyett).

Minden katalógus tulajdonképpen egy rögzített szerkezetű adatfájl. Felépítése a következő:

Mezőnév	Típus	Hossz
PATH	C	70
FILE_NAME	C	12
ALIAS	C	8
TYPE	C	3
TITLE	C	80
CODE	N	3
TAG	C	4

A TITLE és a TAG mezőkön kívül a dBASE IV automatikusan kitölt minden mezőt, ha egy fájl bekerül a katalógusba.

PATH : Az elérési út (szabványos DOS path), ha a katalógus nem az aktuális könyvtárban van.

FILE_NAME : Az aktuálisan bejegyzett fájl neve a kiterjesztéssel együtt.

ALIAS : Adatfájlok esetén a másodnév (ha nincs, akkor a fájlnév), egyéb fájlok esetén üres.

TYPE : Erre a mezőre akkor van szükség, ha a bejegyzés nem az alapértelmezés szerinti kiterjesztéssel történik. Ekkor a FILE_NAME az általunk megadott kiterjesztést tartalmazza, de a dBASE IV az eredetit ide írja be a későbbi azonosíthatóság végett.

TITLE : Mint említettük, ez egy opcionális mező, a SET TITLE ON esetén kerül kitöltésre. Ha tartalmát később módosítani kívánjuk, nyissuk meg a katalógust, mint egy szabványos dBASE fájlt,

```
.USE katalógusnév.CAT
```

ha már nyitva van, akkor

```
.SELECT 10
```

és mindkét esetben pl. az

```
.EDIT
```

paranccsal változtathatjuk meg a mező tartalmát.

CODE : Ezt a számot a dBASE IV rendeli hozzá a bejegyzett fájlhoz a következő szabályok szerint:

- programfájl esetén 0;
- adatfájl létrehozásakor a következő, eggyel nagyobb egész szám, amely kezdetben 1;
- egyéb fájlok (index, formátum, képernyő stb.) esetén az a szám, amelyet az az adatfájl kapott, amelyhez az említett fájlok tartoznak.

Igy tehát azonos kódszámmal rendelkeznek a logikailag egybetartozó fájlok.

TAG : Ezt a mezőt nem használja a dBASE IV.

3.5.2. Katalógusok használata

A SET CATALOG TO-val megnyitott vagy most létrehozott katalógus azonnal aktív lesz (SET CATALOG ON). Ez azt jelenti, hogy az alább felsorolt utasítások használatakor a megfelelő fájlok adatai bekerülnek a katalógusba.

COPY STRUCTURE	COPY STRUCTURE EXTENDED
CREATE	CREATE FROM
CREATE/MODIFY LABEL	CREATE/MODIFY QUERY
CREATE/MODIFY REPORT	CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW	IMPORT FROM
INDEX	JOIN
SET FILTER TO FILE	SET FORMAT
SET VIEW	SORT
TOTAL	USE

Átmenetileg, a katalógus lezárása nélkül is megszüntethetjük ezt a hatást a SET CATALOG OFF kiadásával. Ha ezután létrehozunk vagy törlünk fájlokat (amelyek eredetileg a katalógushoz tartoztak), akkor a SET CATALOG ON kiadása után figyelmeztető üzenet jelenik meg. Ez érthető, hiszen a katalógus "elromlott". A dBASE IV általában is nagy gondot fordít arra, hogy a katalógusok tartalma összhangban legyen a lemezen található tényleges állapottal. A katalógusok megnyitásakor elvégzi az ellenőrzést és végrehajtja a szükséges korrekciókat.

Egy fájl több katalógushoz is tartozhat. Ez megoldható úgy, hogy minden érintett katalógus részére csinálunk egy másolatot a fájlról, de úgy is, hogy a fájl egyetlen példányát megosztjuk a katalógusok között.

Katalógusaink tervezése közben célszerű egyensúlyt teremteni az említett két módszer között. Az alábbiakban néhány megszívlelendő szempontot adunk közre ezzel kapcsolatban:

Az adatfájlok általában nagyméretűek, és gyakran változnak. Ez már két érv amellett, hogy ezeket lehetőleg egy példányban tároljuk, s csak a nevüket soroljuk fel esetleg több katalógusban is. Ellenkező esetben minden egyes példányt párhuzamosan karban kellene tartanunk, s ez számtalan hibalehetőséget rejt magában.

A lekérdezések (queries) viszonylag kisméretűek, ráadásul hatékonyabbak, ha minden egyes alkalmazás speciális igényeire szabva külön csináljuk meg őket. Később zavartalanul módosíthatók, hiszen nem kell tekintettel lennünk a többi katalógusra. Ebből következik, hogy a lekérdezéseket, s a környezeti fájlokat (.VUE) nem célszerű megosztani a katalógusok között.

Egyidőben csak egy katalógus lehet nyitva, és mindig a 10-es munkaterületen. Ez azt jelenti, hogy más fájlok megnyitására ilyen esetben a 10-es SELECT nem használható.

Ha belépünk a Control Center-be, és még nincs egyetlen katalógusunk sem, a dBASE IV létrehoz egy UNTITLED.CAT nevű kezdeti katalógust, amely az összes abban a pillanatban létező fájlt tartalmazza. Bővebben erről a Control Center tárgyalásánál olvashatunk.

3.5.3. Katalógusok lezárása

Egy katalógust lezárhatunk:

- egy másik katalógus megnyitásával;
- a SET CATALOG TO (név nélkül kiadott) paranccsal;
- a CLEAR ALL vagy a CLOSE ALL utasításokkal;
- a QUIT paranccsal.

3.6. Vezérlő szerkezetek

Ha dBASE IV utasításokat egymás után írunk, és ezt az utasításlistát tároljuk egy fájlban, akkor tulajdonképpen már létre is hoztunk egy dBASE programot. Ám ha valóban csak ennyi lehetőséget teremtene a nyelv, nem igazán lenne érdemes programokat írni. Természetes igény, hogy az esetleg más programnyelvekből már ismert alapszerkezetek, nyelvi panelek (mint a feltételes elágazás, a ciklus) itt is elérhetők legyenek. Nos, a dBASE IV alapvetően az alábbi hat vezérlő szerkezetet biztosítja:

- Elágazás egyetlen feltétel szerint (If ... Else ... EndIf);
- Elágazás több feltétel szerint (Do Case ... EndCase);
- Ciklus (Do While ... EndDo);
- Ismétlődő adatkezelő utasítások (Scan ... EndScan);
- Program ill. eljáráshívások (Do, Call, Run)
- Eseménycsapdák (ONERROR/ESCAPE/KEY)

3.6.1. Elágazás egyetlen feltétel szerint (If ... Else ... EndIf);

A struktúrált programozás egyik alapszerkezetével állunk szemben. Feladata rendkívül egyszerű: az IF után megadott feltétel teljesülése esetén az IF-et követő utasítások kerülnek végrehajtásra, míg ellenkező esetben az ELSE után állók. Az ELSE ág opcionális. Ha hiányzik, és a feltétel nem teljesül, akkor a vezérlés az ENDIF utáni első utasításra adódik át. Több egymásba ágyazott IF ... ELSE ...

ENDIF struktúra használata is megengedett, így tetszőleges bonyolultságú elágazások megvalósíthatók. Az ENDIF utasítás után (ugyanabban a sorban) tetszőleges megjegyzés írható. Általában az IF után álló feltételt szokták ideírni, hogy tudni lehessen, az adott ENDIF melyik IF-et zárja le. Ez hosszabb programok esetén egyenesen ajánlott, ugyanis nagymértékben növeli a programok olvashatóságát.

Példa:

```
If x > 100
      ? "Száznál nagyobb."
Else
      If x > 10
            ? "Tíznél nagyobb, de száznál kisebb vagy egyenlő."
      Else
            If x > 1
                  ? "Egynél nagyobb, de tíznél kisebb vagy egyenlő."
            Else
                  ? "Legfeljebb egy."
            EndIf
      EndIf
EndIf
```

Sok esetben gyorsíthatja a programozási munkát, ha az utasítás függvény-megfelelőjét használjuk. Az IIF() (az angol "immediate IF" kifejezésből) függvény két általunk megadott érték közül visszaadja az elsőt, ha a feltétel teljesül, illetve a másodikat, ha nem.

Példa:

```
Use nevek
Do While .Not. Eof()
      If ber < 5000
            Replace ber With ber*1.2
      Else
            Replace ber With ber*1.1
      EndIf
      Skip
EndDo
```

A programrészlet szerint a NEVEK.DBF-en végighaladva a BER mezőt az eredeti értéktől függő arányban növelni kell.

Nézzük ugyanezt az IIF() segítségével:

```
Use nevek
Replace All ber With Iif(ber < 5000,1.2,1.1)*ber
```

3.6.2. Elágazás több feltétel szerint (Do Case ... EndCase);

Programozás közben gyakran kerülünk olyan helyzetbe, amikor valamely változó aktuális értéke szerint kell tovább haladnunk (erre kézenfekvő példa egy menüválasztás), s ilyenkor az okozza a problémát, hogy több feltételt kell megvizsgálnunk. Természetesen ezt is megoldhatjuk az imént tárgyalt IF...ENDIF utasítással, ám a dBASE IV ennél alkalmasabb eszközt biztosít a DO CASE ... ENDCASE utasításfoglalat formájában. A CASE kulcsszavak után álló feltételek közül egyidőben több is igaz lehet (nem kell, hogy kizárják egymást), de a dBASE IV az első teljesülő feltétel után befejezi a további kiértékelést és az ENDCASE-t követő utasítással folytatja működését (természetesen, miután végrehajtotta a CASE után álló utasításokat). Most nézzük meg az előző pontban közölt egyszerű példát a DO CASE ... ENDCASE utasítással megvalósítva:

Példa:

```
Do Case
      Case x > 100
            ? "Száznál nagyobb."
      Case x > 10
            ? "Tíznél nagyobb, de száznál kisebb vagy egyenlő."
      Case x > 1
            ? "Egynél nagyobb, de tíznél kisebb vagy egyenlő."
      OtherWise
            ? "Legfeljebb egy."
      EndCase
```

Vegyük észre, hogy a feltételek sorrendje nem véletlenül alakult így. Fordított sorrendben megadott CASE-ek esetén a programrész helytelenül működne.

3.6.3. Ciklus (Do While ... EndDo);

A számítógép egyik legnagyobb előnye, hogy bizonyos sokszor ismétlődő műveletsorozatokat nagy biztonsággal végre tud hajtani. Ezért bármilyen problémakörben működő, magára valamit adó program biztosan tartalmaz legalább egy ciklust. A ciklus alapvetően két részből áll: a ciklusfeltételből, amely egy feltétel, és a ciklusmagból, amely utasítások tetszőleges sorozata. Amíg a ciklusfeltétel teljesül, a ciklusmag utasításai végrehajtódnak. A ciklusok szervezését több módon lehet megvalósítani a ciklusfeltétel és a ciklusmag egymáshoz való viszonyától és a ciklusfeltétel típusától függően (hátultesztelő ciklus: a ciklusmag végrehajtásra kerül, s a ciklusfeltételt csak ezután értékeljük ki; elöltesztelő ciklus: az előbbinek a fordítottja, azaz csak a ciklusfeltétel teljesülése esetén hajtódik végre a ciklusmag; a FOR típusú ciklusok, ahol a ciklusmag előre megadott számú alkalommal kerül végrehajtásra). A dBASE IV csak a WHILE típusú ciklust ismeri, tehát a feltétel kiértékelésével kezd, s ha az nem igaz, akkor a ciklusmagot egyszer sem hajtja végre. A ciklus önmagában szintén a struktúrált progra-

mozás egyik alapszerkezete, ám ezt a programozási módszert (áttekinthetősége, könnyebb tesztelhetősége stb. ellenére is) sokan nehézkesnek tartják. Valószínűleg az ő érdekükben jelent meg a DO WHILE ciklus kiegészítő utasításaként a LOOP és az EXIT. A ciklusmagban bárhol elhelyezett LOOP hatására a dBASE IV az utána következő utasításokat figyelmen kívül hagyja és a ciklusfeltétel újbóli kiértékelésére ugrik. Az EXIT pedig az ENDDO után következő utasításra adja át a vezérlést. A két "kiugró" utasítás segítségével mentesülünk néhány pótlólagos logikai változó használata alól.

A DO WHILE, DO CASE, IF utasítások tetszőlegesen egymásba ágyazhatók (természetesen azzal a megszorítással, hogy minden utasítás-zárójelet korrekt módon be kell zárni.

Példa:

```
DO WHILE < feltétel 1 >
      IF < feltétel 2 >
            ENDDO
      ENDIF
```

konstrukció nyilvánvalóan hibás]).

A ciklusfeltételben makrókat is használhatunk, de ügyeljünk arra, hogy a makró függvény (&) argumentumaként használt változó tartalma a ciklusmagban ne változzék meg. Ugyanis a DO WHILE utasításban szereplő makró kifejtése csak az első alkalommal (a ciklusba való belépéskor) történik meg, a feltétel összes többi kiértékelését a dBASE IV már a kifejtett makró alapján végzi. Az alábbi példa egy korrekt makróhasználatot mutat be:

```
nem = [SubStr(szemszam,1,1) = "1"]
Use nevek Index szemszam
Do While .Not. Eof() .And. &nem
      ? nev,ber
      Skip
EndDo { .Not. Eof() }
```

3.6.4. Ismétlődő adatkezelő utasítások (Scan ... EndScan);

A SCAN...ENDSCAN utasításpár az aktuális adatfájl összes vagy az utasítás FOR, WHILE cikkelyeiben megadott feltételek által kiválasztott rekordjainak egyszerű feldolgozását segíti. Ezek a tevékenységek a DO WHILE, a SET FILTER, a CREATE QUERY stb. utasítások kombinációinak felhasználásával is elvégezhetők, de mint a példa bizonyítja, a SCAN esetén jóval kevesebbet kell írni.

Példa:

A feladat a novemberben született nők bérét kiírni a NEVEK fájlból. Először nézzük a DO WHILE típusú megoldást:

```
Use nevek
Locate For SubStr(szemszam,1,1)="2" .And. SubStr(szem-
szam,6,2)="11"
Do While .Not. Eof()
        ? nev,ber
        Continue
EndDo
```

Most pedig lássuk ugyanezt a SCAN segítségével:

```
Use nevek
Scan For SubStr(szemszam,1,1)="2" .And. SubStr(szemszam,6,2)="11"
        ? nev,ber
EndScan
```

Megjegyezzük, hogy az Eof() vizsgálata és a Skip utasítás használata a SCAN esetén felesleges, sőt kerülendő.

3.6.5. Program ill. eljáráshívások (Do, Call, Load, Run, Return)

A korszerű és hatékony programozás egyik alapkövetelménye a modularitás, amely nagy vonalakban azt jelenti, hogy a megvalósításra váró feladatot jól körülhatárolható részekre, ún. modulokra bontjuk, amelyek már csak egyszerű, elemi feladatokat hajtanak végre, s az így kapott részproblémákat külön-külön oldjuk meg. E módszer nagy előnye, hogy egyrészt a modulok egyenként tesztelhetőek, így a hibás programrészek lényegesen könnyebben izolálhatók, másrészt bizonyos gyakran használt részprogramokat (keret rajzolás, üzenet kiírása stb.) csak egyszer kell megírni, s ezeket később egy eljáráshívással más programrendszerekben is felhasználhatjuk, ezáltal nagymértékben csökkentve rendszerünk méretét. A modulok közti vezérlésátadásnak több formája létezik. A dBASE IV az alábbiakat támogatja:

- A **DO** elindít egy másik dBASE programot vagy eljárást, majd annak lefutása után a vezérlés visszakerül a DO-t követő utasításra. Ha egy forrásprogramot (.PRG vagy .PRS) első ízben futtatunk, a dBASE IV egy belső tárgykódra fordítja le (.DBO), s csak utána indítja el (Lásd még a SET DEVELOPMENT utasításnál).

- A **RUN** paranccsal az operációs rendszer által futtatható .COM vagy .EXE kiterjesztésű programokat, a .BAT kiterjesztésű utasításfájlokat vagy az operációs rendszer belső utasításait (pl.: DIR) indíthatjuk el. A DOS utasítás értelmezőjének (COMMAND.COM) egy másodpéldánya töltődik be ilyenkor a memóriá-

ba, majd ez indítja el a kívánt programot. Ezért saját alkalmazásunk mellett elegendő memóriaterület szükségeltetik egyrészt a COMMAND.COM, másrészt a futtatandó program számára, egyébként az **Insufficient memory** hibaüzenet jelenik meg. Minden esetben, de különösen hálózati alkalmazásoknál fontos, hogy a COMMAND.COM tartózkodási helye ismert legyen. Ezért a dBASE IV indítása előtt célszerű beállítani a DOS környezetét.

Példa:

```
SET COMSPEC = C:\COMMAND.COM
```

A külső program lefutása után a vezérlés visszkerül a RUN-t követő utasításra, s a lefoglalt memóriarész felszabadul.

Megjegyzés: Néhány DOS utasítás, mint például a PRINT vagy az ASSIGN a RUN-nal való futtatás után a memóriában marad, ezért ezeket nem célszerű a RUN felhasználásával elindítani. Ehelyett még a dBASE IV indítása előtt futtassuk le őket.

● A **CALL** olyan speciális problémák lekezelésére (általában assembly nyelven) írt programok futtatását végzi, amelyeket a dBASE IV nem tud megoldani (pl.: gyorsított sztringműveletek, DOS illetve BIOS hívások). A dolog igényel némi előkészítést. Ugyanis programunkat először bináris formára kell alakítani (EXE2BIN), majd a dBASE IV **LOAD** utasításával be kell töltenünk a memóriába ahhoz, hogy a CALL aktivizálni tudja. A meghívott eljárásnak legfeljebb 7 paramétert adhatunk át. A LOAD semmiféle ellenőrzést nem végez, ezért betöltés előtt győződjünk meg arról, hogy a fájl valóban bináris formában van-e. Hasonló okok miatt csak teljesen letesztelt programokat használjunk.

Bináris programokat hívhatunk a CALL() függvénnyel is, amely csak annyiban tér el a hasonló nevű utasítástól, hogy visszaad egy értéket a hívónak. A LOAD-dal betöltött programokat a RELEASE MODULE paranccsal távolíthatjuk el a memóriából.

● A **RETURN** hatására a vezérlés visszaadódik a hívó program DO utáni sorára, ha volt ilyen.

3.6.6. Eseménycsapdák (ON ERROR/ESCAPE/KEY)

Az eseménycsapdák különleges utasítások. Eltérően a szokványos programutasításoktól végrehajtásuk nem akkor történik meg, amikor a megírt forrásprogram szerint éppen soron következnének, hanem akkor, amikor egy meghatározott esemény bekövetkezik függetlenül attól, hol járunk a programban. Elnevezésük is ebből ered, ugyanis a feladatuk a megfelelő esemény "csapdába ejtése". Ilyenek az ON ERROR, ON ESCAPE, ON KEY, ON READERROR, ON PAGE illetve a menüfigyelő (ON PAD, ON SELECTION .., stb) utasítások.

A vezérlő szerkezetek közé azért sorolhatjuk őket, mert velük általában szintén egy másik eljárást vagy programot aktivizálunk, tehát ezek is egyfajta vezérlésátadások. Az a közös bennük, hogy ezek feltételes vezérlésátadások, azaz a programhívás csak a megadott feltétel teljesülése esetén következik be. Ezek a feltételek a következők:

- ON ERROR esetén valamely dBASE IV hiba előfordulása;
- ON ESCAPE esetén az ESC billentyű megnyomása;
- ON KEY esetén a kijelölt billentyű (vagy ennek hiányában bármely billentyű) megnyomása.
- ON READERROR esetén a beviteli előírásoknak (@ .. GET RANGE, VALID) nem megfelelő adat bevitele.
- ON PAGE az adatfolyam éppen megjelenített sorának száma elérte a lap hosszát.
- Menüfigyelő utasításoknál a menüpont kiválasztása.

Az összes eseményfigyelést célszerű a program előkészítő részében elhelyezni, mivel csak az adott "jelenség" bekövetkezése esetén fejtik ki hatásukat. Természetesen ez nem azt jelenti, hogy egy adott eseményhez menetközben nem rendelhetünk más végrehajtandó utasítást.

Az utasítások hatása addig van érvényben, amíg egy üres (tehát aktivizálandó utasítás nélküli) ON ERROR/ESCAPE/KEY utasítást ki nem adunk. Megjegyezzük, hogy az ON ESCAPE utasítás a SET ESCAPE OFF esetén hatástalan. Fontos tudni továbbá, hogy a meghívott programnak lefordított állapotban kell lennie, mert a dBASE IV ebben az esetben nem végzi el az automatikus fordítást.

3.6.7. Néhány tudnivaló a vezérlő szerkezetek alkalmazásához

- Egyidőben legfeljebb 32 .DBO fájl lehet aktív.

- A LOAD paranccsal betöltött bináris programok mérete egyenként legfeljebb 32000 bájt lehet, s ezekből egyidejűleg legfeljebb 16 db lehet a memóriában,

- A CALL paraméterátadása karakteres típus esetén érték szerint, míg a többi típus esetén név szerint történik.

- Egyidőben legfeljebb 99 fájl lehet nyitva dBASE IV-ben, beleértve a .DBO-kat is (maga a dBASE IV 5-öt lefoglal, így a valódi érték 94).

- Az ON ERROR utasítás az operációs rendszer szintű hibák (a lemezmeghajtó rossz, a nyomtató nincs on-line-ban stb.) esetén hatástalan.

- Az ON PAGE utasításnak tipikus alkalmazása az, amikor nyomtatás közben minden lap alján egy lábrészt szeretnénk elhelyezni. Az utasításban megha-

tározott sorszám elérésekor a ?/?? vagy EJECT PAGE utasítások végrehajtása közben a dBASE IV elindítja a megjelölt utasítást (amely természetesen egy programhívás is lehet). Azt a bizonyos sorszámot javaslatunk szerint a

laphossz - alsómargó - lábrészhossz

formula kiértékelésével határozzuk meg.

● Lehetőség szerint kerüljük a rekurzív programhívásokat. Ez azt jelenti, hogy egy eljárásból saját magát az eljárást hívjuk meg közvetlenül vagy közvetve. Ez ugyan megengedett, de ilyenkor a dBASE IV minden híváshoz újra megnyitja ugyanazt a .DBO fájlt, így nagyon hamar találkozhatunk a **Too many files are open** vagy a **DOS nested too deep** hibaüzenetekkel. Az eleganciáról lemondva az ismétlődő tevékenységekre szervezzünk inkább ciklust.

3.7. Ablak kezelés

Egy programmal szemben alapkövetelmény, hogy hibátlanul és a kitűzött célnak megfelelően működjön. Legalább ennyire fontos azonban az is, hogy kezelőjével emberközeli (ún. felhasználóbarát) módon érintkezzen, szolgáltatásai könnyen kezelhetők, képernyői áttekinthetők legyenek, sőt manapság még ergonómiai szempontokat is figyelembe kell vennünk a programok tervezésekor (gondoljunk csak arra, hogy egy egész napját a képernyő előtt töltő dolgozó számára milyen fontos pl.: a megfelelő színösszeállítás). Az említett szempontok figyelembevételével használjuk ki a dBASE IV idevonatkozó lehetőségeit, amelyekkel tulajdonképpen egy felhasználói csatolót készíthetünk programunkhoz. A felhasználói csatoló nem más, mint a felhasználóval történő érintkezés felülete.

Ebben az alfejezetben az ablakok használatáról lesz szó.

3.7.1. Ablakok létrehozása, használata, megszüntetése

Az ablak a képernyőnek egy téglalap alakú része, amelyet speciális kerettel, színekkel és egyéb attribútumokkal emelhetünk ki környezetéből. Ablakokat általában adatcsoportok megjelenítésére illetve a felhasználóval történő kommunikációra használunk. Mivel a képernyőn egyidőben több ablak is lehet, a nagyobb áttekinthetőség kedvéért célszerű az eltérő funkciókat megvalósító tevékenységeket külön ablakokban végezni.

Ablakot a **DEFINE WINDOW** ablaknév paranccsal **hozhatunk létre**. Meg kell adnunk az ablak nevét, koordinátáit, a keret karaktereit, a színek összeállítását. A keretet (lásd SET BORDER) ASCII kódsorozatként határozhatjuk meg. Lehe-

tőség szerint kerüljük az ASCII 7, 8, 10, 12, 13, 27, 127 kódokat, ugyanis ezek nyomtatás közben problémákat okozhatnak. Egyidejűleg legfeljebb 20 ablakdefiníció lehet jelen a memóriában. Ezzel még csak az ablakunk kijelölése történt meg.

Ahhoz, hogy ez **meg is jelenjen** a képernyőn, az **ACTIVATE WINDOW** ablaknév utasítást kell kiadnunk. Ennek hatására a kívánt ablak megjelenik a képernyő meghatározott részén, s ettől kezdve minden képernyőre szánt kimenő adat az ablakba kerül. Egyetlen ACTIVATE WINDOW utasítással több (korábban már kijelölt) ablak is megjeleníthető, sőt akár az összes, pillanatnyilag memóriában lévő ablak is, de mindig a listában *utoljára* felsorolt lesz az aktív. Minden megjelenő ablak olyan kerettel rendelkezik, amelyet létrehozásakor megadtunk hozzá. Ha egy ablakhoz nem határoztunk meg keretet, akkor a megjelenítést a SET BORDER aktuális állása szabályozza. Az ablakunkat tehát már meg tudjuk jeleníteni.

A **MOVE WINDOW** utasítás segítségével még el is **mozgathatjuk** a képernyő egy tetszőleges másik részére. Az utasításnak két formája van. Az egyikkel (MOVE WINDOW ablaknév TO <sor,oszlop> direkt módon határozhatjuk meg a bal felső sarok új koordinátáit, míg a másikkal (MOVE WINDOW ablaknév BY <sor,oszlop>) az aktuális helyzethez relatívan, az eltolás mértékét megadva jelölhetjük ki az ablak új helyét. Ha az utasítást koordináták nélkül adjuk ki, az ablak visszakerül a kijelölésekor hozzárendelt pozícióba.

Előfordulhat, hogy az ablak, mint munkaterület, átmenetileg szűknek bizonyul (például hibaüzenetek, segítő információk megjelenítése, s egyáltalán a felhasználóval történő bármilyen kommunikáció során) vagy valamely más ok miatt rövid időre szükségünk lesz az egész képernyő használatára. Ekkor az **ACTIVATE SCREEN** paranccsal térhetünk vissza a **teljes-képernyős üzemmód**ba. Az aktuálisan látható ablakok nem tűnnek el, a képernyő tartalma változatlan marad, de ettől kezdve minden megjelenő adat most már a teljes képernyőre kerül. Ezért persze, ha nem vigyázunk, felülírhatjuk az ablakaink tartalmát. Tevékenységünk végeztével egy újabb ACTIVATE WINDOW utasítással ismét az ablakon belül dolgozhatunk tovább.

Ha ablakaink **paramétereit meg akarjuk őrizni** más programok számára is, el kell mentenünk őket egy fájlba a **SAVE WINDOW** ablaknévlista TO < fájlnév > utasítással. Elmenteni csak az aktuálisan memóriában lévő ablakokat lehet. Az így keletkező fájl **.win** kiterjesztést kap.

Egy tetszőleges későbbi időpontban a megadott fájlban tárolt ablakdefiníciók a **RESTORE WINDOW** <ablaknévlista> FROM <fájlnév> utasítással **visszanyerhetőek**.

Most már minden eszközt megismertünk, amely ablakok létrehozásához, aktivizálásához, tárolásához szükséges. A továbbiakban eltávolításuk módjait vesszük sorra. Egy ablak **törölhető a képernyőről** úgy, hogy a memóriában még ottmarad

(hogy szükség esetén gyorsan újra aktivizálható legyen). Ezt a **DEACTIVATE WINDOW** ablaknévlistaz utasítással tehetjük meg. Ekkor az előzőleg aktivizált ablak lesz újra aktuális. Ha nincs ilyen vagy az ALL opcióval minden ablakot megszüntettünk, a dBASE IV teljes-képernyős üzemmódban működik tovább. A **RELEASE WINDOWS** <ablaknévlista> utasítás a felsorolt ablakokat **törli a képernyőről és a memóriából is**. Ez azt jelenti, hogy ha korábban nem használtuk a SAVE utasítást, akkor az eltávolított ablakdefiníciók visszanyerésére nincs többé mód. A törölt ablakok helyén *visszaáll az eredeti képernyőtartalom*.

Az eltávolítás leggyorsabb módját a **CLEAR WINDOWS** utasítás biztosítja. Ennek hatására az **összes kijelölt ablak törlődik a képernyőről és a memóriából is**, majd visszaáll a teljes-képernyős üzemmód, és a korábban "eltakart" szövegrészek ismét láthatóvá válnak.

3.7.2. Az ablakok és a memo-mezők kapcsolata

Az APPEND, BROWSE, CHANGE, EDIT, READ utasítások használata közben a memo-mezők szerkesztése alapértelmezésben a teljes képernyőn történik. Ezt megváltoztathatjuk a SET WINDOW OF MEMO TO ablaknév utasítás kiadásával. Ennek hatására az említett utasításokban a memo-mezők a megadott ablakban szerkeszthetők. Az ablaknak léteznie kell a memóriában, mielőtt hivatkoznánk rá, mert egyébként a **Window has not been defined** üzenet jelenik meg. A @...GET utasítás WINDOW cikkelye felülbírálja a SET WINDOW által megjelölt ablakot. Ezt a következő példa segítségével világítjuk meg:

Példa:

```
.Define Window Abl1 From 1,1 To 12,30
.Define Window Abl2 From 12,12 To 23,40
.Set Window Of Memo To Abl1
.@ 1,1 Get MEMO1
.@ 2,1 Get MEMO2 WINDOW Abl2
```

Az első Get az *Abl1* ablakban szerkeszti a MEMO1-et, míg a második az *Abl2*-ben a MEMO2-t.

3.8. A menürendszer felépítése

3.8.1. Bevezetés, alapfogalmak

A jó program felhasználói csatolójának másik fontos eleme a menürendszer. Programjaink vezérléséhez feltétlenül áttekinthető, könnyen kezelhető menüket kell előállítanunk. A dBASE IV az eddigi verziókhoz képest szokatlanul bőséges eszköztárat biztosít erre a célra. Alapvetően két menütípust támogat:

- bar menu;
- popup menu.

A részletes ismertetés előtt tisztáznunk kell néhány igen fontos alapfogalmat. Sajnos nem minden kifejezéshez találtunk megfelelően pontos magyar változatot, de ez a jelenség nem szokatlan a számítástechnikában.

A **bar menu** a képernyőn általában vízszintesen elhelyezkedő menüpontokból (angolul: *pad*) áll. Mi ezt a menütípust a továbbiakban **vonalmenünek** vagy még egyszerűbben **menünek** fogjuk nevezni.

A **popup menu** függőleges elhelyezkedésű. Általában akkor jelenik meg, amikor a felhasználó kiválaszt egy elemet valamilyen más (vonal)menüből. Elemei nemcsak a tevékenységet jelölő szövegek, hanem fájlnevek, mezőnevek és egyéb tetszőleges objektumok is lehetnek. Ennek a menütípusnak elnevezéseként néhány magyar nyelvű kiadványban olvasható a "beugró menü" kifejezés, mi azonban (ha szerencsésebbnek nem is de) rövidebbnek tartjuk a **rolómenü** kifejezést, amely a vizuális megjelenéséhez is közelebb áll

A pull-down menü az előbbi kettő összekapcsolásakor keletkezik. Minden vonalmenü eleméhez tartozhat egy rolómenü, de az csak akkor jelenik meg, ha a vonalmenüben a megfelelő elemen áll az sávkurzor. (sávkurzor fogalmát lásd a Control Center fejezetben). Ezt az összetett konstrukciót is csak menünek fogjuk a továbbiakban hívni, de ez nem okoz zavart.

Megjegyezzük, hogy az angol szakirodalom a vonalmenü elemeit "*pad*"-nek, míg a popup menü elemeit "*bar*"-nak nevezi. Ez az utasítások szintaxisának megértéséhez lesz fontos. Mi a könnyebb érthetőség kedvéért ahol csak lehet, egyszerűen a menüelem vagy **menüpont** megnevezést használjuk.

Menürendszerünket először fel kell építenünk, majd ha használni akarjuk, aktivizálnunk kell, akárcsak az ablakokat. Legkönnyebben az Alkalmazás-generátorral készíthetünk menürendszert, ugyanis az előállítja a programkódot, nekünk csak a speciális módosításokat kell elvégeznünk rajta. A következő két alfejezetben ismertetjük a két menütípus kezelésével kapcsolatos utasításokat, függvényeket. Ez egyrészt könnyebbé teszi az Alkalmazás-generátorral készített programkód megértését, másrészt hozzásegít ahhoz, hogy önálló programokat írhassunk.

Előrebocsájtjuk, hogy a menük körül megjelenő keretek típusa a SET BORDER utasítással beállítható.

3.8.2. Vonalmenü létrehozása, használata

Vonalmenü létrehozásakor az első teendőnk **nevet adni**. Ezt a **DEFINE MENU** < vonalmenünév > utasítással tehetjük meg. Ezzel természetesen még csak annyi történt, hogy kijelöltük a létrehozni kívánt vonalmenü nevét, és az esetleges MESSAGE cikkellyel hozzárendeltünk egy legfeljebb 79 karakteres üzenetet, amely majd a képernyő alsó részén fog megjelenni.

Ezek után a **menü elemeit** kell meghatároznunk a
DEFINE PAD < menüpont-név > OF < vonalmenünév > PROMPT < N-kifejezés > utasítások sorozatával. A < vonalmenünév > egy már kijelölt vonalmenü neve kell legyen (DEFINE MENU). A < menüpont-név > -re ugyanazok a megkötések érvényesek, mint a mezőnevekre. A PROMPT után megadott karakteres kifejezés értéke maga a menüpont. Ezeket az AT cikkely segítségével a képernyőn bárhol elhelyezhetjük (akár függőlegesen egymás alá is). Ha nem adtunk meg koordinátákat, a dBASE IV az elsőként kijelölt elemet a képernyő bal felső sarkában kezdődően, míg a többit ugyanabban a sorban, egymástól egy szóközzel elválasztva helyezi el. A menüpontokon a **balra** illetve **jobbra** mutató **nyilakkal** mozoghatunk, egyet kiválasztani természetesen az **Enter** leütésével lehet.

Ha a felhasználó kiválasztott egy elemet, a megfelelő **tevékenység elindításához** használjuk az **ON PAD** < menüpont-név > OF < vonalmenünév > illetve az **ON SELECTION PAD** < menüpont-név > OF < vonalmenünév > utasításokat. A második utasítás elindítja a kijelölt tevékenységet, amely lehet egyetlen dBASE IV utasítás, de rendszerint egy program vagy egy eljárás hívása. Az ily módon a menüponthoz rendelt program tartalmazhatja a MENU() és a PAD() függvényeket is azért, hogy eldönthető legyen, melyik menü pontosan mely eleme volt a legutoljára kiválasztott (ez olyan általános programok esetén lehet szükséges, amelyeket több menüben is meghívunk). Az utasítás vagy program befejeződése után a vonalmenü ismét rendelkezésre áll további választások céljára.

Az ON SELECTION PAD < menüpont-név > utasítás vagy programhívás nélküli kiadása azt jelzi, az aktuális menüpont nem végrehajtható.

Az utasítás másik formája, az **ON PAD** ... kimondottan a **rolómenükhöz kapcsolódást** valósítja meg. Az utasításban megnevezett rolómenü minden esetben aktivizálódik, ha a sávkurzor a megfelelő vonalmenü menüelemén áll. Ha az utasítást az ACTIVATE POPUP opció nélkül adjuk ki, a meghatározott menüpontot nem lehet kiválasztani. Az **ACTIVATE POPUP** használatával egyidejűleg aktív a vonalmenü és a menüponthoz kapcsolt aktuális rolómenü. A jobbra-balra mutató nyilakkal más és más rolómenüket aktivizálhatunk, míg a felfele-lefele mutató nyilakkal az adott rolómenün belül mozoghatunk.

3.8.3. Rolómenü létrehozása, használata

A rolómenü tulajdonképpen egy ablak a képernyőn, amely speciális elemeket tartalmaz, és tartozik hozzá egy üzenetsor meg egy keret.
Létrehozni a DEFINE POPUP < menünév > FROM <sor1,oszlop1> ... utasítással lehet. A FROM és a TO után álló koordináták az ablak bal felső és jobb alsó sarkának elhelyezkedését jelölik ki. Egyszerre csak egyetlen rolómenü lehet aktív, ezért még azt is megtehetjük, hogy a képernyő ugyanazon részére több, különböző rolómenüt jelölünk ki. Ha a TO cikkelyt nem adtuk meg, azaz az ablaknak csak a bal felső sarkát jelöltük ki, a dBASE IV úgy választja meg az ablak méretét, hogy elég széles legyen a maximális elem befogadására is, valamint elegendően magas legyen ahhoz, hogy az összes menüsort tartalmazni tudja. A rolómenü ablakának legalább egy sort tartalmaznia kell, s ennek szélessége legalább 1 kell legyen. Az ablak elhelyezésére is vannak korlátok: a maximális oszlop 79, a maximális sor 22 (amennyiben a státusz sor nincs a képernyőn, akkor 24) lehet.

A **PROMPT cikkely három** lehetséges **formájának** mindegyike valamilyen különleges lehetőséget biztosít a rolómenü menüelemeinek kijelölésére, mint ezt azonnal látni fogjuk.

- A **PROMPT FIELD** < mezőnév > cikkely lehetővé teszi, hogy a rolómenü ablakában egy megnyitott **adatfájl megadott mezőjének tartalma** jelenjen meg a fájl minden rekordjára (ez nem lehet memo-mező).
- A **PROMPT FILE** [LIKE] [maszk] cikkely hatására az ablakba az aktív katalógusban (Lásd 4.3.) nyilvántartott **fájlok nevei kerülnek** majd. A LIKE paraméterrel megszabhatjuk, hogy csak a maszkban (mintában) megjelölt fájlnevek kerüljenek be az ablakba.
- A **PROMPT STRUCTURE** cikkely hatására az **aktuális adatfájl mezőnevei** (ha SET FIELDS ON, akkor csak a SET FIELDS TO utasításban meghatározott mezőnevek) kerülnek az ablakba.

Ha a három lehetőség valamelyikét választjuk, később nincs módunk a menüelemeket másként meghatároznunk, azaz DEFINE BAR utasítást kiadni az adott rolómenüre vonatkozóan. Ennek természetesen értelme sem lenne, hiszen a PROMPT segítségével az ablakot éppen a speciális elemekkel akartuk feltölteni.

Ha a felsorolt PROMPT cikkelyek egyikével sem élünk, akkor a **rolómenü menüpontjait a DEFINE BAR** <sorszám> OF <menünév> PROMPT <C-kifejezés> utasítások sorozatával határozhatjuk meg. Ha egy menüpont sorszáma nagyobb, mint a kijelölt rolómenü ablakmérete, akkor a menüpontok függőlegesen "gördülnek" az ablakon belül. Ha egy menüpont szélesebb az ablaknál, a dBASE IV levágja a végét, tehát a vízszintes rollozás nem megengedett. Ha egy menüpontnak nem adunk értéket, a menü megfelelő sora üresen marad. A rolómenünek legalább egy kitöltött sort tartalmaznia kell, ellenkező esetben nem aktivizálha-

tó. Egy **menüpont kiválaszthatóságát a SKIP** cikkellyel szabályozhatjuk. Pontosabban, a SKIP használatával az adott sor ugyan meg fog jelenni a menüben, de nem választható. A SKIP FOR < feltétel > cikkely használatával a választhatóságot a feltétel szabja meg.

A menürendszerünk tehát teljes. Már csak azt kell elmondanunk, hogyan **aktivizáljunk** valamilyen **tevékenységet**, ha a felhasználó (Enter megnyomásával) kiválasztotta a rolómenü valamelyik sorát. Nos, ezt az **ON SELECTION POPUP** utasítással tehetjük meg. A kiváltott tevékenység itt is lehet egyetlen dBASE IV utasítás, de egy egész alprogram is. Választás után a menü természetesen inaktívvá válik egészen az elindított program (vagy utasítás) befejeződéséig. A meghívott program rendszerint egy DO CASE...ENDCASE utasítást és a POPUP(), PROMPT() valamint a BAR() függvények valamelyikét tartalmazza.

Példa:

```
Do Case
      Case Bar() = 1
            Do Prog1
      Case Bar() = 2
            Do Prog2
      .
      .
      .
EndCase
```

3.8.4. Általános menükezelő utasítások

A menürendszer kijelölése, meghatározása önmagában még nem eredményezi annak működését, ehhez, azaz az **aktiváláshoz** előbb ki kell adnunk egy **ACTIVATE MENU** <vonalmenünév> utasítást is. A PAD cikkely használatával meghatározhatjuk, hogy a sávkurzor kezdetben melyik menüponton álljon (alapértelmezés szerint ez az első menüpont). Az aktív menüt ismét **használaton kívüli helyezhetjük** egy újabb menü aktivizálásával, az **Esc** billentyű megnyomásával, illetve a **DEACTIVATE MENU** utasítással. A használaton kívül helyezett vonalmenü törlődik a képernyőről, de a memóriában továbbra is megmarad, hogy később bármikor újra aktivizálhassuk. A DEACTIVATE MENU utasítás nem igényel paramétert, ugyanis mindig az aktuális vonalmenü helyezi használaton kívül. Ilyenkor a vezérlés az adott vonalmenüt aktivizáló utasítást követő utasításra adódik át.

Az **ACTIVATE/DEACTIVATE POPUP** utasítás ugyanezeket a funkciókat valósítja meg a **rolómenük esetén.**

Ha a **vonalmenüt** (vagy -menüket) a **memóriából is el akarjuk távolítani**, akkor a **RELEASE MENUS** < vonalmenülista > utasítást kell használnunk. Ennek hatására a listában felsorolt menük eltűnnek a képernyőről és a memóriából egya-

ránt. Ugyanekkor érvényüket vesztik az adott vonalmenükkel kapcsolatos ON SELECTION illetve ON PAD típusú utasítások is. Az éppen aktív vonalmenü a RELEASE utasítással nem távolítható el. A **RELEASE POPUPS** utasítás ugyanerre a célra szolgál a rolómenüket illetően. Az **összes vonalmenü** egyidejűleg történő **megszüntetésére** szolgál a **CLEAR MENUS** utasítás. Megjegyezzük, hogy a RELEASE MENUS menülista nélkül kiadva ugyanezt a hatást éri el. A **CLEAR POPUPS** utasítás az összes rolómenüt távolítja el a képernyőről és a memóriából egyaránt, valamint **törli** a hozzájuk **kapcsolódó ON SELECTION** utasításokat is. Ez az utasítás az aktív rolómenüt is inaktívvá teszi.

A **MENU()** függvény a pillanatnyilag **aktív vonalmenü nevét** adja vissza. Ha nincs ilyen, akkor a visszaadott érték az üres sztring.

A **PROMPT()** függvény az aktuális **menüpont szövegét** adja vissza. Ha a DEFINE POPUP utasítás PROMPT FIELD cikkelyét használtuk, akkor a megfelelő **mezőtartalmat**, a PROMPT FILES esetén a **teljes fájlnevet** beleértve az elérési utat és a kiterjesztést is, a PROMPT STRUCTURE esetén pedig a **mezőnevet** adja vissza. Az utóbbi két esetben a sztringeket csupa nagybetűs formában (uppercase) kapjuk meg.

A **BAR()** függvény a rolómenüből kiválasztott **menüpont sorszámát**, a **POPUP()** pedig az aktív **rolómenü nevét** adja vissza.

Végezetül megemlítjük, hogy a dBASE IV menürendszereink teszteléséhez is segítséget nyújt a **SHOW MENU** illetve a **SHOW POPUP** utasításokkal. Ezek az utasítások csak **megjelenítik** a szóbanforgó menüket, **de nem aktivizálják** őket. Így ellenőrizhetjük és szükség esetén még a programba illesztés előtt korrigálhatjuk menüink elhelyezkedését a képernyőn.

Az elmondottak alapján összefoglaljuk egy komplett menürendszer létrehozásának lépéseit.

- Nevet adunk a vonalmenünek: DEFINE MENU.
- Kijelöljük a vonalmenü elemeit: DEFINE PAD.
- Minden egyes vonalmenüponthoz hozzárendelünk egy rolómenüt: ON PAD.
- Kijelöljük a megfelelő rolómenüket: DEFINE POPUP.
- Minden rolómenühöz meghatározzuk a benne szereplő menüpontokat: DEFINE BAR.
- Minden egyes rolómenühöz hozzárendelünk egy programot, amely a konkrét vezérlést megvalósítja: ON SELECTION POPUP.
- Elkészítjük az előző lépésben kijelölt programokat.

Az így felépített menü elindításához már csak az ACTIVATE MENU utasítást kell kiadnunk.

3.8.5. Példaprogram

Ebben a példaprogramban egy szokványos törzsadatkezelés egy lehetséges menürendszerének megvalósítását mutatjuk be. Tegyük fel, hogy két törzsfájlt – a NEVEK.DBF és a BEREK.DBF fájlt – kell karbantartanunk. Mindegyiken a szokásos négy alapműveletet kell elvégeznünk:

- új rekord felvitele;
- módosítás;
- törlés;
- listázás.

A menüben először a fájlt (vonalmenü), majd az elvégzendő műveletet (rolómenük) kell kiválasztanunk.

Lássuk tehát a programot.

```
*
* A vonalmenü kijelölése
*
Define Menu torzs
Define Pad nevek Of torzs Prompt "Névtörzs adatok" At 1,3
Define Pad berek Of torzs Prompt "Béradatok" At 1,20
*
* A rolómenük hozzárendelése
*
On Pad nevek Of torzs Activate Popup nevpop
On Pad berek Of torzs Activate Popup berpop
*
* A nevpop kijelölése
*
Define Popup nevpop From 3,1 Message "Névtörzs karbantartás"
*
* A nevpop elemeinek meghatározása
*
Define Bar 1 Of nevpop Prompt "Új felvitel"
Define Bar 2 Of nevpop Prompt "Módosítás"
Define Bar 3 Of nevpop Prompt "Törlés"
Define Bar 4 Of nevpop Prompt "Listázás"
* A nevpop kijelölése
*
Define Popup berpop From 3,20 Message "Béradatok karbantartá-
sa"
*
* A berpop elemeinek kijelölése
*
Define Bar 1 Of berpop Prompt "Új felvitel"
Define Bar 2 Of berpop Prompt "Módosítás"
Define Bar 3 Of berpop Prompt "Törlés"
Define Bar 4 Of berpop Prompt "Listázás"
```

```
*
* A kiválasztott funkció elindítása
*
On Selection Popup nevpop Do Nev_prg
On Selection Popup berpop Do Ber_prg
*
* A menü aktivizálása
*
Activate Menu torzs
*
* Példaprogram vége
```

Megjegyezzük, hogy a program akkor lesz teljes, ha a hivatkozott Nev_prg, Ber_prg eljárásokat is megírjuk.

Példaként álljon itt a Nev_prg forráskódja:

```
Procedure  Nev_prg
Private  v_bar
v_bar  =  Bar()
Do Case
      Case v_bar  =  1
            Do N_Add          && A felvitelt végző program
      Case v_bar  =  2
            Do N_Mod          && A módosítást végző program
      Case v_bar  =  3
            Do N_Del          && A törlést végző program
      Case v_bar  =  4
            Do N_Lst          && A listázást végző program
EndCase
Return
```

3.9. Programok tesztelése

3.9.1. Bevezetés

Mielőtt saját programjainkat ún. "éles" adatokkal kezdenénk üzemeltetni (különösen hálózati környezetben), ajánlatos meggyőződnünk arról, hogy a program egészében és az egyes eljárások részleteiben valóban a kívánt feltételek szerint működnek-e. Amíg nyugaton a rendszerfejlesztések igen tekintélyes hányadát éppen az elkészült termékek igen körültekintő tesztelése teszi ki, hazánkban ez sajnos még nem eléggé elterjedt módszer.

Teljesen hibátlan programrendszert készíteni csaknem lehetetlen. Az immár könyvtárnyi szakirodalommal dokumentált programhelyesség-bizonyító módszerek adhatnának megfelelően hatékony eljárást tökéletes rendszerek fejlesztéséhez, ám ezek

gyakorlati alkalmazhatósága még nem teljesen meggyőző. Így nem tehetünk mást, mint arra törekszünk, hogy a rendszerünkben előforduló rejtett hibák számát minimalizáljuk.

Kétségtelen, hogy a programok alapos tesztelése nem túl kellemes feladat, de ezzel együtt a programozási munka szerves részét kell képeznie, mert így későbbi, esetlegesen igen komoly következményektől óvhatjuk meg magunkat.

Néhány alapvető programtesztelési szempontot illetve tanácsot sorolunk fel az alábbiakban:

- Mindig legyen kéznél a rendszer egy korábbi verziója, hogy szükség esetén visszatérhessünk hozzá;
- Próbáljuk ki programjainkat szélsőséges esetekben (pl.: adatműveleteknél üres fájl vagy nem létező fájl, képernyős adatbevitelnél az általunk jónak tartott intervallumon kívül eső értékek megadása, stb.)
- Nagyon fontos lépés a tesztadatok összeállítása. Lehetőség szerint olyan komplex bemenőadat-rendszert készítsünk, amely a programok összes ágát kipróbálják.
- Próbáljuk meg bármilyen módon megzavarni a program működését (értelmetlen karaktersorozatok gyors begépelése, a klaviatúra teljes tenyérrel történő nyomkodása, stb.). Ez kicsit erőltetettnek tűnhet, de egy igazán jól megtervezett programnak még ilyen "tisztességtelen" behatásokat is ki kell védenie.
- Ha úgy érezzük, hogy rendszerünk hibátlan, adjuk oda valaki másnak. Mivel ő nem ismeri programjaink felépítését, majdnem biztos, hogy újabb hibák egész sorát fogja megtalálni.
- Végül, a tesztelés során megtalált és kijavított alapvető hibák listáját célszerű megőrizni. Ugyanis ezeket valószínűleg elkövetnénk újabb rendszerek készítésénél is, viszont ha a lista előttünk van, ennek kisebb a valószínűsége.

3.9.2. A programtesztelést segítő eszközök

Ebben az alfejezetben néhány olyan eszközt mutatunk be, amelyek a programok működésének áttekinthető követését, a programhibák viszonylag gyors felderítését könnyítik meg.

A **SET ECHO ON** utasítás hatására egy program futása közben **megjelennek a képernyőn** mindazok a dBASE IV **utasítások, amelyeket a program** abban a pillanatban **végrehajt.** Ez már önmagában is nagyon nagy segítség, de még hatékonyabbá tehető néhány további utasítás alkalmazásával:

- A **SET DEBUG ON** hatására az említett utasítások a képernyő helyett a **nyomtatón** jelennek meg. Ez akkor lehet hasznos, ha programunk éppen amúgy is a képernyőre ír, s a megjelenő adatok illetve a SET ECHO által kiírt utasítások együtt zavaróan hatnának. Megjegyezzük, hogy a SET TALK ON, LIST, DISPLAY utasítások eredményei illetve a hibaüzenetek továbbra is a képernyőn fognak megjelenni, hacsak a SET PRINT ON utasítást korábban ki nem adtuk.
- A **SET TALK ON** kiadása azt eredményezi, hogy bizonyos utasítások (APPEND FROM, COPY, INDEX stb.) valamint változóknak történő **értékadások eredményei**, továbbá a **rekordsorszámok** az utasítás végrehajtása után megjelennek a képernyőn.
- S ha még mindig nem határoltuk körül a hibát, adjuk ki a **SET STEP ON** utasítást. Ettől kezdve a dBASE IV programunkat **lépésenként** hajtja végre, azaz minden utasítás után felfüggeszti annak futását, s kiírja az utasítás eredményét is, majd az alábbi
  Press SPACE to Step, S to Suspend, or Esc to Cancel ...
  üzenet jelenik meg. A dBASE IV nem csinál semmit, amíg a három lehetőség közül valamelyiket ki nem választjuk.

A **CANCEL** utasítással bármely ponton **megszakíthatjuk a program futását.** Ekkor a dBASE IV minden utasításfájlt (.DBO) lezár és megjelenik a pont. Ugyanakkor az eljárás-fájlokat nem zárja le a dBASE, ezt nekünk kell megtennünk a CLOSE PROCEDURE utasítás kiadásával.

Hosszabb ideig futó programok tesztelésekor, ha közben eszünkbe jut valami, nem szükséges megszakítani, majd újraindítani a programot. Az **Esc** billentyű megnyomásakor **(ha SET TRAP állása OFF)** a dBASE IV az alábbi kérdést teszi fel:
Cancel, Ignore, Suspend
"S"-t (vagyis a Suspend-et) válaszolva módunk nyílik arra, hogy bármilyen utasítást hajtsunk végre. Ilyenkor például értéket adhatunk egy elfelejtett változónak, megnyithatunk egy fájlt, stb., majd a **RESUME** utasítás beírásával **folytatódik a program futása** most már a beállított új környezetben. Megjegyezzük, hogy az ily módon pótlólagosan létrehozott változók a felfüggesztett programra nézve

lokálisak, azaz PRIVATE típusúak lesznek. Célszerű a RESUME előtt egy @ ..CLEAR utasítást kiadni, hogy az interaktív módon beírt utasításaink által "elrontott" képernyő ne zavarja a program további működését. Fontos tudni, hogy egy SUSPEND-del felfüggesztett tranzakció (Lásd a BEGIN TRANSACTION utasításnál) közben kiadott ROLLBACK, majd RESUME utasítás hatására a vezérlés az END TRANSACTION utasítást közvetlenül követő utasításra adódik át.

A **RETRY** utasítás egy hibát okozó **utasítássorozatot** próbál **újra indítani**, amíg az sikeresen be nem fejeződik. Ez az utasítás az ON ERROR paranccsal és az ERROR() függvénnyel (amely az előfordult dBASE IV hiba sorszámát – ezek a függelék részben megtalálhatók – adja vissza) együtt hatékony, és a tesztelés során kiderült olyan hibák kivédésére használható, amelyek a rendeltetésszerű használat közben is előfordulhatnak. Az alábbi rövid példán megpróbáljuk illusztrálni annak az esetnek a kezelését, amikor egy fájl megnyitási kísérlet hibát idéz elő, mivel az adott fájl egy másik munkaterületen már nyitva van.

Példa:

```
*
* Főprogram
*
On Error Do Hiba With Error()
<Egyéb utasítások>
Use  nevek                    && Ez a sor okozza a hibát.
<Egyéb utasítások>
Return
*
* Főprogram vége
*


*
* HIBA.PRG
*
Parameters  hiba
Do Case
      Case hiba = 3
            Close Databases
            Retry
      <Egyéb vizsgálatok>
EndCase
Return
*
* HIBA.PRG vége.
*
```

Megjegyezzük, hogy a RETRY nem értékeli ki ismét az IF és a DO WHILE után megadott feltételeket.

3.9.3. A beépített nyomkövető program (debugger)

Az előző alfejezetben ismertetett nyomkövető eszközök biztosította lehetőségeken túl a dBASE IV még egy beépített **debuggert** is rendelkezésünkre bocsát. Kétféleképpen aktivizálhatjuk: egyrészt a SET TRAP ON paranccsal, másrészt a **DEBUG** < programnév > **WITH** < paraméterlista > utasítással.

A **SET TRAP ON** utasítás hatására a **vezérlés átkerül a debuggerre, ha** programunkban **hiba** fordul elő **vagy** ha megnyomjuk az **Esc** billentyűt. Hiba esetén a program a hibás utasításon áll meg (ha az ON ERROR utasítás érvényben van, akkor a SET TRAP hatástalan). Az *Esc* megnyomása hasznos lehet például gyanúsan hosszú ideig tartó ciklusok megszakításakor.

A **DEBUG** utasítás ugyanúgy **elindítja** a megadott **programot,** mint a DO, s egyúttal aktivizálja a debuggert is. A paraméterlistában a hívott program megszokott paramétereit adhatjuk át. A DEBUG képernyőjén négy ablak látható. Bennük futás közben nézhetjük a programunk éppen végrehajtott utasításait, szerkeszthetjük ezeket, töréspontokat helyezhetünk el, s végül a futás eredményeit is láthatjuk. Az ablakok nevei: **Debug; Edit; Help; Breakpoint; Display.**

```
C:\DBASE\TT.PRG
 1 *
 2 * Debug proba. Torolheto
 3 *
 4
 5 set trap on
 6 set talk off
 7 x=3
 8 y=4
 9 wait
10 return

DISPLAY                                  BREAKPOINTS
x                    :          3        1:
y                    :          4        2:
                     :                   3:
                     :                   4:

DEBUGGER
Work Area: 1        Database file:        Program file: tt.prg
Record:    1        Master Index:         Procedure:    TT
ACTION:    d                              Current line: 9

ESCape interruot.
```

3.1. ábra A DEBUG ablakai

A képernyő alsó részén elhelyezkedő **Debug ablakban** a program aktuális állapotára jellemző információk (aktuális munkaterület, adatfájl, program neve, eljárás neve, program sorszám, rekord sorszám, fő indexfájl) valamint a debugger *utasítás-sora* kapott helyet. A debug működése közben a képernyőn bárhol kiadott **Esc** vagy **Ctrl-End** az utasítás-sort aktivizálja.

Az **Edit ablak** a képernyő felső részén helyezkedik el. Itt látható az aktuálisan futó program forráskódjának éppen végrehajtás alatt lévő utasítása és annak környezetében lévő forrás-sorok, amelyet a dBASE IV beépített szövegszerkesztőjével azonnal módosíthatunk, ha ez az ablak aktív. A módosítások azonban csak akkor fejtik ki hatásukat, ha a megváltoztatott forráskódot újrafordítjuk.

A **Help ablak** a debugger utasításait és azok rövid leírását tartalmazza. A képernyő jobb felső sarkában jelenik meg, elfedve az Edit és a BreakPoint ablakokat. A debugger indulásakor valamint az F1 billentyű lenyomásakor aktivizálódik, bármilyen más utasítás végrehajtásakor eltűnik.

A **Breakpoint ablak** a képernyő jobb oldalán látható. Ha az ablak aktív, akkor feltételeket adhatunk meg (törési feltételek [breakpoint conditions]), amelyeket minden programutasítás után kiértékel a debugger. Ha közülük valamelyik igazzá (.T.) válik, a program futása megszakad és a debug vezérlő képernyője jelenik meg.

A képernyő bal oldalán látható **Display ablak** függőlegesen két részre oszlik. A bal oldali részben tetszőleges dBASE IV kifejezéseket adhatunk meg, amelyeknek eredményei a jobb oldali ablakrészben jelennek meg. Programunk eredményeinek, algoritmusainak helyességét ellenőrizhetjük itt.

A továbbiakban a Debug ablak utasítás-sorában kiadható utasításokat vesszük sorra:

B A Breakpoint ablak kiválasztása.

D A Display ablak kiválasztása.

E Az Edit ablak kiválasztása.

L A program futása az itt megadott soron folytatódik.

[n]N Végrehajtja a következő programutasítást, majd visszaadja a vezérlést a debug utasítás-sorának. Az [n] opcionális paraméter egy egész szám. Ha megadjuk, akkor a következő **n** programsort hajtja végre, s utána tér vissza az utasítás-sorba. Ha a következő n sorban (n = 1 is lehet) programhívás szerepel, akkor az alprogram a debugger környezetén kívül kerül végrehajtásra, bár a törési feltételeket továbbra is figyeli a dBASE IV.

P Futás közben nyomkövetési információkat (aktuális program, eljárás, sorszám) jelenít meg.

Q Kilépés a debugger-ből.

R A program addig fut tovább, amíg töréspontho nem ér vagy hiba nem fordul elő.

[n]S Ugyanaz, mint az N utasítás, azzal a különbséggel, hogy az alprogramok is a debugger környezetében futnak.

X Kilépés az interpreter környezetbe. Az ott kiadott RESUME utasítás visszaadja a vezérlést a debugger-nek.

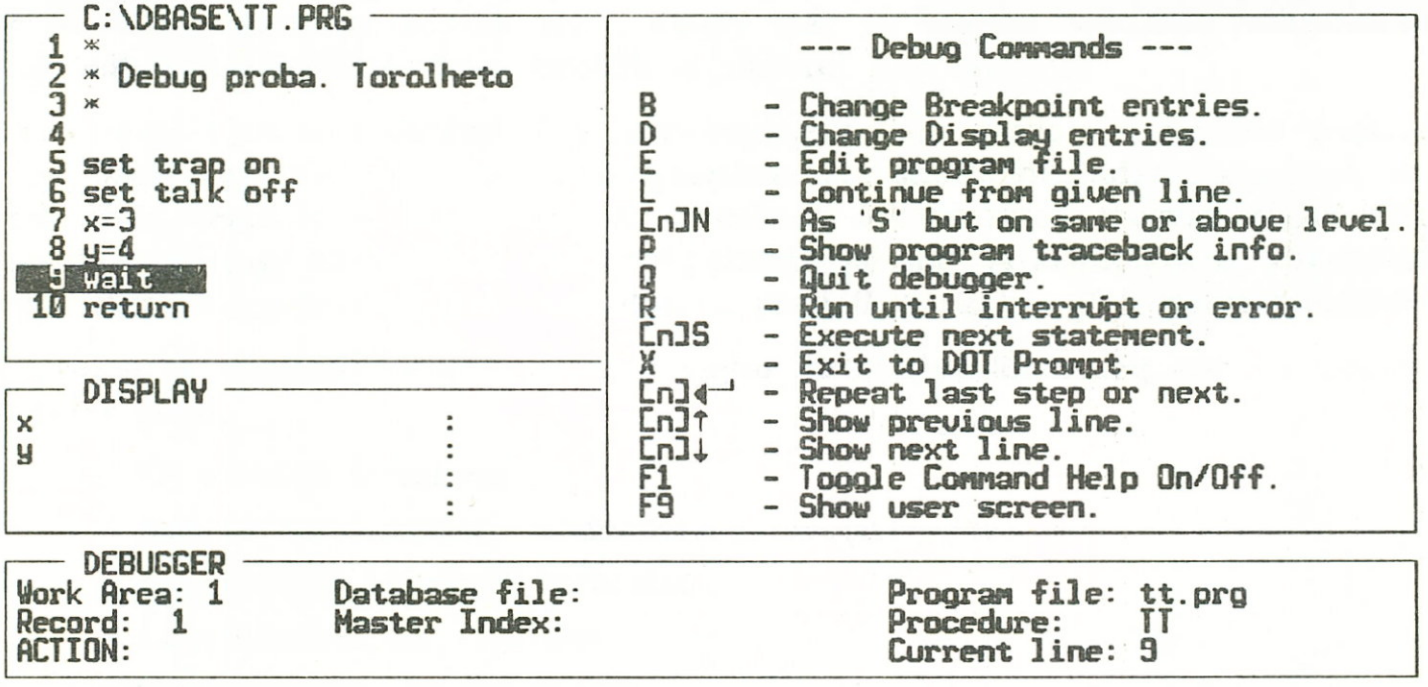
Enter Ha legutóbb egy [n]N utasítást hajtottunk végre, akkor Enter hatása olyan, mintha ezt ismételnénk. Minden más esetben a korábbi [n]S utasításnak megfelelő tevékenység következik be.

[n] ↑ Az Edit ablakban látható forrásprogramban **n** programsorral **előre** mozog.

[n] ↓ Az Edit ablakban látható forrásprogramban **n** programsorral **hátra** mozog.

F1 Aktivizálja/deaktivizálja a Help ablakot. Ez a funkció a többi ablak esetén is hatásos.

F9 Megmutatja a programunk által kialakított aktuális képernyőt, újbóli megnyomáskor pedig visszatér a debugger képernyője. A többi funkció esetén is működik.



3.2. ábra A DEBUG Help ablaka

Az **F1** leütésekor a képernyő bármely részén vagyunk is, a debugger utasításait összefoglaló Help ablak jelenik meg, amelyet egy újabb F1 lenyomásával tüntethetünk el.

Az **F9** egyszeri megnyomásakor a programunk által kialakított aktuális képernyőt láthatjuk, majd újbóli leütésére ismét megjelenik a debugger vezérlő képernyője.

4. Control center

4.1. Bevezetés, alapfogalmak

A dBASE IV lényegesen összetettebb rendszer, mint elődei (dBASE II, dBASE III, dBASE III PLUS), kezelését mégis sokkal egyszerűbben elsajátíthatjuk a jól összefogott, szinte valamennyi funkciót lefedő felhasználói csatoló, a **Control Center** segítségével. A fejlesztő ASHTON-TATE már a dBASE III PLUS esetén is tett kísérletet arra, hogy a felhasználót lépésről-lépésre vezesse az alapoktól az egyre bonyolultabb feladatok felé. Ez a keretrendszer volt az *ASSIST*, amely szemléletessége ellenére sokszor inkább akadályozta, mint segítette a munkát. Úgy tűnik, az alkotók megszívlelték a jogos kritikákat, hiszen az ASSIST továbbfejlesztéseként egy olyan (a dBASE IV ütőképességét nagymértékben elősegítő) keretprogramot hoztak létre, amely már az igazán "vájtfülű" felhasználók igényeit is kielégítheti, de a kezdők is sikerrel használhatják.

A Control Center segítségével a tevékenységek nagy része elvégezhető anélkül, hogy akár egyetlen parancsot is megadnánk. Az adatfájlok, adatkapcsolatok létrehozását, tetszőleges bonyolultságú keresések, lekérdezések megvalósítását elérhetjük úgy, hogy közben alig kell begépelnünk valamit. A szükséges mezőneveket, változókat, függvényeket kiválaszthatjuk a rendelkezésünkre álló menükészletből.

Röviden és pusztán felsorolásszerűen álljon itt néhány lehetőség, amit a Control Center kínál:

- Új adatfájl létrehozása.
- Adatok szerkesztése, módosítása és megjelenítése.
- Adatbeviteli képernyő tervezése.
- Listaformátumok készítése.
- Címketervezés és nyomtatás.
- dBASE IV program generálása.
- DOS fájl és alkönyvtárkezelés.
- Munkakörnyezet beállítása.
- Billentyű-makrók létrehozása, használata.

A Control Center sok funkciója lényegében két, világosan elkülöníthető csoportra osztható.
Az *egyik* a bennünket kiszolgáló valamilyen szerkezet, objektum megtervezését támogatja. Ezt sok esetben a dBASE terminológia maga is a "**Design**" szóval illeti, amelynek magyar fordítása a – **tervezés** – mellett mi is megmaradunk.
A *másik* a már megtervezett szolgáltató-objektumok használatára irányul. Erre

nincs külön fogalom, de nem is szükséges, hiszen a tervezési jellegű funkciókon kívül minden más ide értendő.

A továbbiakban a Control Center használatának megértéséhez néhány nélkülözhetetlen alapfogalmat kell bevezetnünk.

Munkafelület

Nem keverendő össze az adatfájlok megnyitásánál megismert "munkaterület" fogalommal. A dBASE IV használata során egy-egy munkafolyamathoz kapcsolódó jellemző képernyőt, annak is a középső, legnagyobb részét jelöli, amelyben a különböző formátumok tervezése, az adatok kezelése (pl.: EDIT,BROWSE) történik. Minden tervező és adatkezelő művelethez más-más munkafelület tartozik. E fejezet során lényegében ezek használatát ismertetjük részletesen.

sávkurzor

A képernyőn mindig valamilyen kurzorral mozoghatunk. Az esetek többségében ez egy villogó, egy karakter széles, egyszerű téglalap. Más esetekben (és a Control Center-ben ez a jellemzőbb) a kurzor a képernyőnek egy nagyobb területét jelöli. Az angol szakirodalom ezt "highlight"-nak nevezi, mi a "sávkurzor" kifejezést használjuk helyette.

menüsor

A munka során a másik leggyakrabban használt eszköz a menüsor. Minden munkafelülethez megjelenik a hozzátartozó menü a képernyő legfelső sorában. *Ez mindig az* **F10** *billentyű lenyomására aktivizálódik.*
A menüsor vagy vonalemnü elemei mindig egy rejtett almenü azonosítói. Ha egy elemet kiválasztunk, akkor megjelenik a hozzátartozó almenü, amelyben újabb lehetőségek közül választhatunk. A tevékenység befejezése után az almenü eltűnik, így ismét a munkafelület egésze rendelkezésünkre áll a további munkához. E tulajdonsága miatt az ilyen menüt "pull-down" (szó szerinti fordításban "lehúzó") menünek hívják az angol szakirodalomban. Néhány magyar nyelvű kiadványban olvasható a "beugró menü" kifejezés, mi azonban (ha szerencsésebbnek nem is, de) rövidebbnek tartjuk a "rolómenü" kifejezést, amely részben vizuális megjelenéséhez, részben az eredeti angol kifejezéshez is közelebb áll. Ha ez nem zavaró illetve kétértelmű, akkor gyakran csak az egyszerű "menü" terminológiánál maradunk.

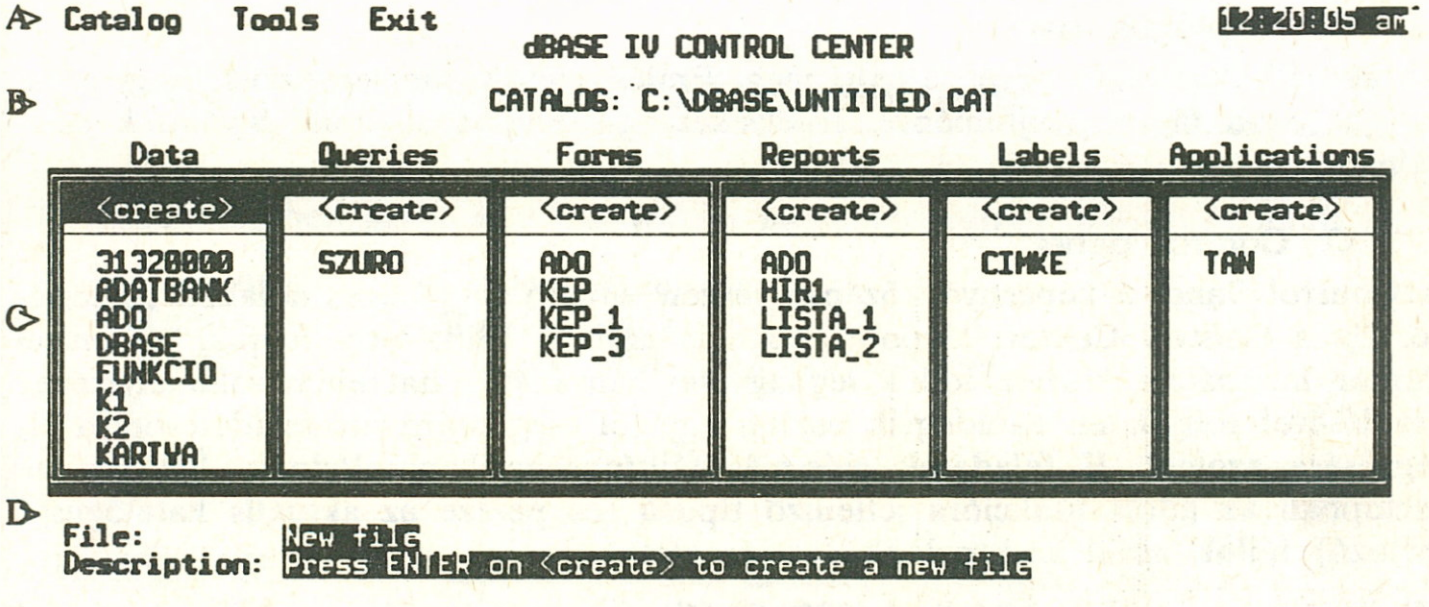
A tömörség kedvéért itt is bevezetünk egy jelölést, azért, hogy ne kelljen mindig kiírnunk: *válasszuk az XXXXXX menüpont YYYYYY opcióját*. Ezentúl ezt a következőképpen jelöljük: **<menüpont> • <opció>**, amely a menüpont rolómenüjének egyik választási lehetőségére utal.

Most nézzük meg a képernyő felosztását a Control Center esetében, ezzel röviden felsorolva a keretrendszer alkotóelemeit úgy, ahogy használat közben találkozunk velük.

4.2. A Control Center alapfelülete

A Control Centert az interpreterből az **assist** utasítás begépelésével indíthatjuk el, de használhatjuk az **F2** (assist) funkciós billentyűt is. Amennyiben a konfigurációs fájlban (config.db) elhelyezzük a COMMAND = ASSIST sort, úgy a dBASE IV indításakor a Control Center automatikusan betöltődik.

A Control Center bejelentkező képernyőjét az alábbi ábrán láthatjuk. Ezt nevezzük a Control Center **alapfelületének**. Minden egyes munkafelületről a feladatok befejezésével ide történik a visszatérés.



4.1. ábra

A: Menüsor
B: Katalógusnév
C: Control panel
D: Fájl információk
E: Navigációs sor
F: Üzenetsor

A: Menüsor

A választható menüelemeket (Catalog - Katalógus, Tools - Eszközök, Exit - Kilépés) és a rendszeridőt tartalmazza. A menü az F10 funkciós billenyű leütésével a Control Center bármely pontján aktivizálható. A menürendszerből, a sávkurzor megfelelő helyre mozgatásával és az Enter leütésével választhatunk. Egy

menüelem természetesen újabb almenüket hívhat meg, amelyekből szintén hasonló módon jelölhetjük ki a kívánt tevékenységet.

A navigáló billentyűk :

→	Jobbra lép egy menüt.
←	Balra lép egy menüt.
↓	A rolómenüben a következő opcióra lép.
↑	A rolómenüben az előző opcióra lép.
Home	Az első menüre lép.
End	Az utolsó menüre lép.
a....z	A menüt megnevezésének kezdőbetűjével hívjuk meg.

B: Katalógus név

Az aktuális katalógus nevét jeleníti meg. Emlékeztetőül megjegyezzük, hogy a katalógus azon fájlok gyűjteménye, amelyeket egy rendszeren belül kívánunk használni.

C: Control panel

A Control Panel a képernyő középső részén látható függőleges ablakok összessége. Ez a Control Center központi része is egyben. Mint látni fogjuk, a Control Center használata közben időnk legnagyobb hányadát a hat elem valamelyikének kezelésével töltjük el. Mindegyik oszlop egy jól meghatározott feladatcsoport elvégzésére szolgál. E feladatok más-más fájltípushoz kapcsolódnak. Így minden oszlopban az adott funkcióra jellemző típusú (és persze az aktuális katalógusba tartozó) fájlok nevei szerepelnek egymás alatt.

Mindegyik panel elem legfeljebb 200 db fájlnevet tartalmazhat, ezért ha egy oszlop megtelik, újabb neveket csak néhány meglévő fájl törlése árán adhatunk hozzá.

A Control Panel elemei:

Data :	Adatfájlok
Queries :	Nézetfájlok
Forms :	Képernyőformátumok
Reports :	Listaformátumok (Jelentések)
Labels :	Címkék
Applications :	Alkalmazások

D: Fájl információk

A Control Panel alatt lévő két sorban a sávkurzor által megjelölt fájl neve és a hozzátartozó megjegyzés látható.

E: Navigációs sor

Megmutatja a fontosabb billentyűkhöz rendelt funkciókat.

F: Üzenetsor

A munka aktuális fázisát segítő üzeneteket tartalmazza.

Bárhol járjunk a Control Center menürendszerében, az **F1** (HELP) leütésével mindig bővebb információt is kaphatunk az adott folyamat korrekt végrehajtását illetően.

A továbbiakban a Control Center-t funkcionálisan két nagy részre osztva tárgyaljuk. Az elsőben a menük által nyújtott szolgáltatásokat vesszük sorra, míg a másodikban az egyes panelek részletes ismertetésére kerül sor.

4.3. A menürendszer az alapfelületen

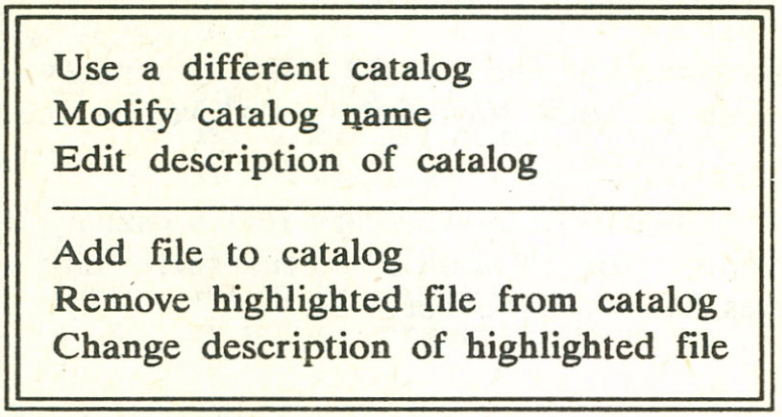
Mint a bevezetőben arról már szó volt, a menü alapvetően három elemből áll:

Catalog	(Katalóguskezelés)
Tools	(Segédeszközök)
Exit	(Kilépés, azaz a munka befejezése)

4.3.1. Katalógusok - Catalog

A katalógusfájl az egy adatrendszerhez tartozó, különböző fájltípusokat tartja nyilván. A katalógus tartalmazza az egyes fájlok neveit és a hozzátartozó megjegyzéseket. Amikor belépünk a Control Centerbe, a rendszer automatikusan betölti az aktuális alkönyvtárhoz tartozó katalógusfájlt. Ha az alkönyvtárban még nincsen dBASE felhasználói fájl, – így katalógus sincsen, – akkor az első fájl létrehozásakor automatikusan keletkezik egy katalógusfájl is (ha a SET CATALOG állása ON). Ha nem jelöltünk meg katalógus nevet az interpreter vagy a config.db segítségével, úgy a katalógusfájl neve automatikusan *UNTITLED.CAT* lesz. (A katalógusfájl nevének a kijelölése, és a katalógus használat a SET CATALOG TO és a SET CATALOG ON/OFF beállításával történik.)

A katalógusok kezelését a képernyőn megjelenő almenü segíti:



4.3.1.1. Másik katalógus megnyitása - Use a different catalog

Ez az opció az összes létező katalógus nevét megjeleníti a képernyőn, a hozzájuk tartozó megjegyzésekkel együtt. A sávkurzor segítségével jelölhetjük ki a fájlok közül azt, amelyik igazodik a pillanatnyi munkánk témájához, és így azt az Enter leütésével aktivizálhatjuk, illetve a < create > kiválasztásával hozhatunk létre új katalógust.

4.3.1.2. Katalógusnév és -leírás módosítása

Katalógus nevének módosítása a **Modify catalog name** opció, leírásának megváltoztatására pedig az **Edit description of catalog** opció szolgál.

4.3.1.3. Katalógusba vétel, törlés, fájl-leírás módosítása

Ha az aktuális katalógusba újabb fájlokat szeretnénk bevinni, használjuk az **Add file to catalog** opciót. Ilyenkor egy panelben megjelennek a kiválasztható fájlok, amelyből a sávkurzor segítségével választhatunk. Csak olyan típusú fájlok vehetők a katalógusba, amelyek az aktuálisan kijelölt Control Panel-elemben is vannak. Ha például a sávkurzor a Control Panelben a képernyőformátum (FORM) oszlopon áll, úgy csak képernyőformátum fájlt vihetünk az aktuális katalógusba.

Egy katalógus egy fájltípusból legfeljebb 200-at tartalmazhat.panelben legfeljebb 200 (legfeljebb 200 darabot).

A sávkurzorral kijelölt fájlokat eltávolíthatjuk az aktuális katalógusfájlból a **Remove highlighted file from catalog** opció kiválasztásával, és szükség esetén a háttértárolóról is törölhetjük azt. Ha adatfájlt akarunk törölni a háttértárolóról, akkor előbb le kell zárnunk.

A svákurzorral kijelölt fájl leírását is módosíthatjuk a **Change description of highlighted file** opcióval.

4.3.2. Segédeszközök - Tools

A Control Center menürendszerének **Tools** opciója tovább bővíti lehetőségeinket. Segítségével többek között meghívhatunk DOS eljárásokat a dBASE IV interpreterből, adatszinten kommunikálhatunk más adatkezelő rendszerekkel. Kijelölhetünk belépési jogosultságot (ez különösen hálózati környezetben válik fontossá), a makrók alkalmazásával pedig módosíthatjuk, átdefiniálhatjuk a billentyűzetet. Szükség esetén állíthatunk a "munkakörnyezeten" is.

A Tools menü kiválasztásakor a következő almenürendszer jelenik meg :

```
Macros
Import
Export
Dos utilities
Protect data
Settings
```

A **Macros** és a **Dos utilities** menüpontokról külön szólunk. Nézzük a többit részletesebben:

4.3.2.1. Más adatkezelő rendszerek adatainak beolvasása

Az **Import** opció megválasztásával más adatkezelő rendszerekből hozhatunk át adatokat adatfájlokba. A megjelenő almenü az adatkezelők neveit tartalmazza, amelyek adatait a dBASE IV fogadni képes.

```
Rapidfile        (.rpd)
dBASE II         (.db2)
Framework II     (.fw2)
Lotus 1-2-3      (.wk1)
PFS:FILE
```

4.3.2.2. Adatok kiírása más adatkezelő rendszerek számára

Az **Export** opció lehetőséget ad adatfájlok átalakítására más adatkezelő rendszerek által értelmezhető adatformátumba. A megjelenő almenü a lehetséges rendszereket mutatja.

Rapidfile	(.rpd)
dBASE II	(.db2)
Framework II	(.fw2)
Lotus 1-2-3	(.wks)
VisiCalc	(.dif)
PFS:FILE	
SYLK-Multiplan	
Text fixed-length fields	(.txt)
Blank delimited (.txt)	
Character delimited	(.txt)

4.3.2.3. A védelmi rendszer létrehozása

A **Tools • Protect data** opciójának kiválasztásával úgy önálló, mint hálózati környezetben működő védelmi rendszert építhetünk fel. Beállíthatjuk az egyes hozzáférési jogokat jelszókat kötve azokhoz. Az opció analóg a PROTECT paranccsal, amelynek ismertetését megtalálhatjuk a hálózati alkalmazások tárgyalásánál, és a függelékben is.

4.3.2.4. A környezeti jellemzők beállítása

A **Settings** opció megválasztásakor beállíthatunk néhány környezeti jellemzőt. A jellemzők megjelennek a képernyőn, s értékeiket a *szóköz* billentyű leütésével kapcsolhatjuk át. A paraméterek jelentése részletesen megtalálható a függelék SET utasításokat tárgyló részében.

4.3.3. Billentyűmakrók - Tools • Macros

A billentyűmakró nem más, mint a billentyűzetről történő adat- és utasításbevitel "szimulálása", helyettesítése. A billentyűmakró használatakor a számítógép, illetve az abban futó program úgy "érzékeli", mintha a makróban lévő billentyűleütés-sorozatok valóban a billentyűzetről (tehát a felhasználótól) származnának.
A dBASE IV-ben a leütéssorozatokat, azaz makrókat egy betűhöz vagy az **F1 — F9** funkcióbillentyűkhöz rendelhetjük hozzá, és adhatunk külön nevet is neki. A makrók tekintetében két alapvető funkciót kell megismernünk:
a **makró előállítását**, és a **makró használatát**.

A dolog nagyon hasonlít a magnó működéséhez, csak itt nem hangokat, hanem billentyűleütéseket "veszünk fel", majd egy tetszőleges későbbi időpontban ezeket "játsszuk vissza". Ezért nem véletlen, hogy a makró előállítását a "record", míg lejátszását (azaz alkalmazását) a "play" szóval illetik az angol szakirodalomban.

Az elkészült makró elmenthető a memóriában lévő aktuális makró-könyvtárba, sőt a makrókat tartalmazó makró-könyvtár a memóriából lemezre is kitölthető.

A leütéssorozat, a makró tartalmazhat dBASE IV utasítást, szöveget vagy vezérlőbillentyű leütést (pl: Enter). A billentyűsorozatnak, makrónak adott név segítségével játszhatóak le a billentyűleütések felhasználói programunkban is.
Az alábbiakban bemutatunk néhány alkalmazhatósági területet a könnyebb érthetőség kedvéért :

- Az adatbevitel során gyakran előforduló szöveget hozzárendelhetjük egy billentyűhöz (például F5), így ugyanazt a szöveget nem kell "ezerszer" leírni.
- Meghatározhatunk egy parancsot, amely a megfelelő billentyű leütésére végrehajtásra kerül (pl: LIST).
- Gyakran előfordulhat, hogy "mélyen benne vagyunk" egy felhasználói menürendszerben (például az ötödik szinten), ahonnét egyből a főmenübe szeretnénk ugrani. Ez megoldható makróval is programban úgy, hogy a lejátszott makró — melyre nevével hivatkozhatunk — példánknak megfelelően négy **Esc** leütést tartalmaz. Akik ismerik a *Clipper*-t, már találkozhattak hasonló lehetőséggel. Ott a KEYBOARD utasítással valósították meg ugyanezt a célt.

Az elkészített makró kétféleképpen játszható le. Egyrészt a PLAY MACRO parancs segítségével, másrész az **Alt-F10** és a makróhoz rendelt billentyű leütésével.

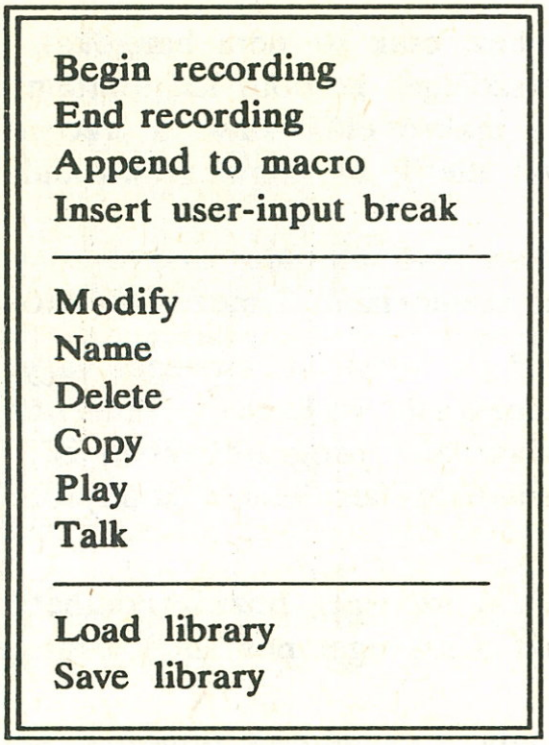
Például legyen az "A" billentyűhöz rendelt kifejezés {ESC}{ESC}, "Vissza" makrónévvel. A makró visszajátszása a következő :

– PLAY MACRO Vissza, vagy:

– **Alt-F10** és utána "A" billentyű leütése.

Minden egyes-makró könyvtár összesen 35 makró funkciót tartalmazhat. A kijelölés felgyorsítása érdekében használhatjuk a Shift-F10 leütést.

A makrók kezelését a következő almenürendszer segíti:



4.3.3.1. A makró felvétele és a lejátszás módjai

A **Begin recording** opció megnyitja, megindítja a makró előállítását az aktuális makró-könyvtár tartalmának megjelenítésével. Kiválaszthatjuk azt a betűt vagy funkcióbillenyűt, amelyhez makrót akarunk rendelni. A kiválasztás a megfelelő billentyű (pl: "A") leütésével történik.

Előírhatjuk a makró lejátszásának megszakítását is az **Insert user-input break** kiválasztásával. Ilyenkor a lejátszás (nem a felvétel!) *folytatása* majd a **Shift-F10** segítségével történik.

Az **End recording** befejezi a makró felvételét. Ezt elérhetjük a **Shift-F10** billenytű és az E billentyű lenyomásával is. Az elkezdett makrót mindig le kell zárni.

A **Play** opció a PLAY MACRO utasítással azonos módon végrehajtja, visszajátssza a makrót, úgy hogy az előírt megszakításoknál megáll, és várja a **Shift-F10** megnyomását.

A makró lejátszása közben a makró "szövege" megjelenik a képernyőn, ha a **Talk** opciónál az **ON** állást választjuk ki. Nincs kapcsolatban a SET TALK környezeti paraméterrel.

4.3.3.2. Másolás, bővítés, módosítás szövegszerkesztővel

A **Copy** opció segítségével az egyik makró tartalma átmásolható egy másikba.

Az **Append to macro** opció segítségével egy létező makró végéhez újabb kifejezéseket illeszthetünk.

A makrók tárolása szabályos ASCII fájlban történik, ezért lehetőségünk van arra, hogy egy korábban készített makró tartalmát egyszerű szövegfájlként módosítsuk. Ehhez bármilyen külső szövegszerkesztőt is használhatunk, de a **Modify** opció aktivizálja a beépített szövegszerkesztőt. A makrófájl tartalma első ránézésre kissé meglepő lehet, ugyanis a speciális karakterek (pl.: a nyilak,Enter,Esc stb ...) kapcsos zárójelek között szerepelnek. Az alábbi táblázatban összefoglaljuk a makró szerkesztése során használható speciális jelek korrekt megadási módját. A kis- és nagybetűk szabadon keverhetők.

Vezérlő billentyű	Makró kifejezés tartalma
Enter	Enter
Esc	Esc
Del	Del
PrtSc	PrtSc
Backspace	Backspace
Tab	Tab
Shift-Tab	Shift-Tab
→	Rightarrow
←	Leftarrow
↓	Uparrow
↑	Downarrow
PgUp	PgUp
PgDn	PgDn
Home	Home
End	End
Shift	Shift
Ins	Ins
Ctrl-	Ctrl-
Alt-	Alt-

Mégegyszer hangsúlyozzuk, hogy a vezérlő billentyűket { } jelek közé kell tenni.

Példa:

Tekintsük meg a következő makrót:

```
@ 12,12 SAY "Át egy másik menübe." {Enter}
  {Esc}{Esc}{Rightarrow}{Enter}
```

Az makró meghívása (lejátszása) kiírja az "Át egy másik menübe" szöveget, utána a menürendszerben két szintet felfelé halad, majd az aktuális menürendszerben egyel jobbra lép a sávkurzor, és belép a kiválasztott másik menübe.

4.3.3.3. Átnevezés, törlés, másolás, mentés lemezre és visszatöltés

A **Name** opcióval billentyűleütés-sorozathoz rendelt nevet, a makrónevet változtathatjuk meg.

Az aktuális makró könyvtárból kitörölhetjük a megfelelő billentyű leütésével kiválasztott makrót. Először a **Delete** opciót kell kiválasztanunk, majd a megfelelő billentyű leütésével a makró törlődik.

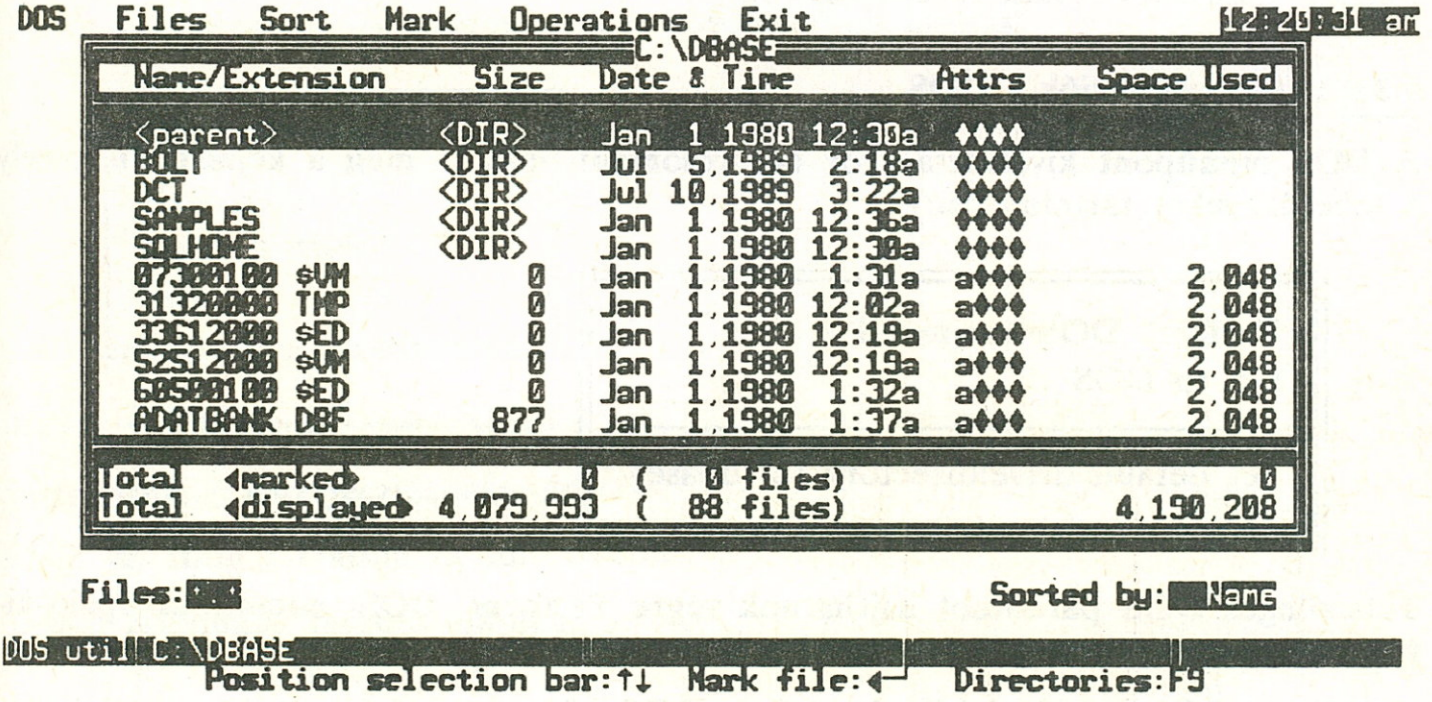
A **Save library** segítségével a memóriában lévő makrókat — az aktuális makró-könyvtárat — elmenthetjük egy megnevezett makrófájlba.

A **Load library** kiválasztásával a kijelölt makrófájl betölthető a memóriába, és ezzel aktuális makró-könyvtárrá válik.

4.3.4. DOS segédprogramok - DOS utilities

Az DOS utilities opció segítségével anélkül, hogy elhagynánk a Control Centert számos DOS parancsot hajthatunk végre anélkül, hogy azok pontos szintakszisát ismernénk.

Az opció kiválasztásakor egy felhasználói csatoló lép működésbe és ezzel ismét egy új munkafelületre kerülünk, amely a következő képernyővel jelenik meg:



4.2. ábra

A munkafelületen az aktuális alkönyvtár fájljai láthatók az utolsó módosításuk dátumával, jellemzőikkel (attribútumaikkal), méreteikkel együtt. A legalsó két sorban a megjelölt (lásd később) fájlok darabszáma és összmérete jelenik meg. Az alkönyvtárak közti közlekedés legegyszerűbb módja, ha a fájl-lista bármely pontján megnyomjuk az **F9** billentyűt. Ekkor a munkafelületen a lemez összes alkönyvtárát tartalmazó "fa"-diagram jelenik meg grafikus formában. A fa elemein a sávkurzorral mozogva a könyvtárváltás egyszerűen végrehajtható. Másik lemezmeghajtóra a diagram tetején lévő meghajtó azonosító kiválasztásával térhetünk át. Az **F9** újbóli leütése után ismét visszatér a fájl lista.

Az első sorban található menü a következő elemeket tartalmazza:

DOS	(DOS funkciók végrehajtása a dBASE IV elhagyása nélkül)
Files	(Speciális fájlcsoportok kiválasztása)
Sort	(A fájl-lista rendezettségének változtatása)
Mark	(Fájlok megjelölése)
Operations	(Fájlműveletek a megjelölt vagy az összes fájlra vonatkozóan)
Exit	(Kilépés a menüből)

Lássuk a menüpontokat részletesebben:

4.3.4.1. DOS funkciók - DOS

A **DOS** menüpont kiválasztásakor egy rolómenü jelenik meg a képernyőn, amely a lehetőségeket tartalmazza:)

```
Perform DOS command
Go to DOS
─────────────────────────────────
Set default drive:directory {c:\dbase}
```

Tetszőleges DOS parancsot hajthatunk végre **Perform DOS command** opció kiválasztása után. Ilyen lehet például a DIR, CHKDSK, stb.

A **GO to DOS** opcióval kiléphetünk a DOS környezetbe és ott (a memóriakorlátokra is figyelve) tetszőleges tevékenységet végezhetünk. Ezután visszatérni az *Exit* parancs begépelésével tudunk.

A **Set default drive:directory** az aktuális meghajtó, alkönyvtár beállítására szolgál. Ennek meghatározásától kezdve minden fájl a megadott alkönyvtárban keletkezik.

4.3.4.2. Fájlok - Files

Ismét egy almenü jelentkezik:

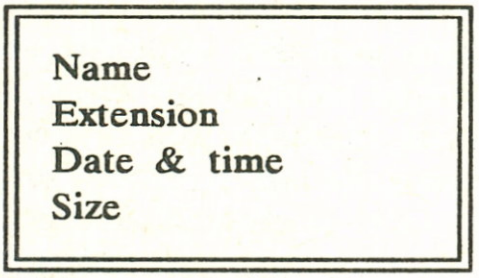
```
Change drive:directory      {c:\dbase}
Display only                {*.*}
```

Ha más alkönyvtárban lévő fájlok listáját szeretnénk használni, válasszuk a **Change drive:directory** (Válts meghajtót/könyvtárat) pontot. Figyelem! Ez nem változtatja meg az aktuális meghajtót/alkönyvtárat.

A **Display only** (Megjelenítés csak..) választásával leszűkíthetjük a fájl-listát a bennünket érdeklő csoportokra. A DOS által használt *, ? maszk karaktereket használhatóak a minta kijelölésében. Például **.dbf* kijelölés esetén, az aktuális alkönyvtár összes adatfájlja (és csak ezek) jelenik a munkafelületen. Ez a lehetőség jól kombinálható a *Mark* menüben ismertetett funkciókkal.

4.3.4.3. Rendezés - Sort

A munkafelületen megjelenő fájlnevek négy szempontnak megfelelő sorrendbe rakhatók a képernyőn. A szempontok a következők lehetnek:



Name : Állomány Neve szerint.

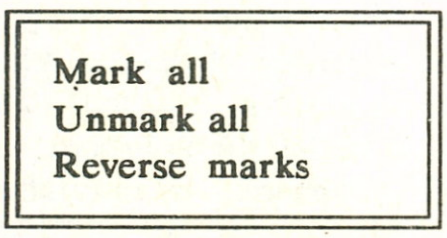
Extension : Kiterjesztés alapján.

Date & time : Dátum és idő szerint.

Size : Állomány mérete alapján.

4.3.4.4. Fájlok megjelölése - Mark

Az opció alkalmazásával megjelölhetünk fájlokat a munkaterületen, a sávkurzor segítségével. Az így kijelölt fájlcsoportokat az *Operations* menü együtt kezeli. (Például az összes megjelölt fájlt egyszerre kitörli az aktuális alkönyvtárból. Az egyenkénti kijelölés az **Enter** leütésével lehetséges, de az alábbi menü még kényelmesebbé teszi a sokszor előforduló csoportos kijelölést:



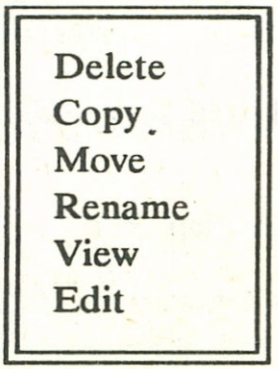
Mark all esetén képernyőn megtalálható valamennyi fájl megjelölésre kerül.

Unmark all választásával egyik fájl sem marad megjelölve.

Reverse marks a megjelöléseket ellenkezőjére fordítja, azaz ami az opció kiválasztása előtt megjelölt fájl volt, az utána nem lesz az, és fordítva.

4.3.4.5. Műveletek - Operations

A képernyőn megjelenített fájlokkal az almenüben felsorolt műveletek végezhetők el. Ha több fájl is megjelölésre került, akkor az összes megjelölt fájl résztvesz a műveletekben, egyébként pedig csak az, amelyiken asávkurzor áll.



Delete : a fájlok törlésére szolgál.

Copy : a fájlokat a végrehajtás előtt kijelölt helyre (meghajtóra/alkönyvtárba) másolja.

Move : a fájlokat a végrehajtás előtt kijelölt helyre (meghajtóra/alkönyvtárba) átviszi úgy, hogy azokat az eredeti helyükön megszünteti.

Rename : a fájlok átnevezésére szolgál. Csak egy fájlra működik.

View : a kiválasztott fájl tartalmát megjeleníti a képernyőn. Ez az opció csak egy fájlra működik.

Edit : a beépített szövegszerkesztő segítségével módosíthatunk a kijelölt fájl tartalmán.

4.3.4.6. Visszatérés - Exit

Visszatérés a Control Center-be.

4.4. A Control Panel

A korábbiakban már említettük, hogy a Control Panel a Control Center munkafelületén látható hat, egymás mellett elhelyezett oszlop összessége.

Az oszlopokat, panelrészeket is panelnek fogjuk hívni a továbbiakban, de mindig megjelöljük, hogy melyikről van szó.

Első közelítésben csak annyit láthatunk, hogy mindegyik panel más-más típusú fájlok neveit tartalmazza. A következő alfejezetekben részletesen ismertetjük az egyes elemek rendeltetését, s a velük kapcsolatos kezelési tudnivalókat.

Minden panelhez más-más munkafelület és menürendszer tartozik. Zavaró lehet viszont, hogy sok esetben ugyanaz a menüopció (pl.: GoTo) szerepel több, különböző panelnél is, ráadásul mindenütt egy kicsit más jelentéssel. Így a könnyebb megértés kedvéért az alfejezetek elején felsoroljuk, hogy az aktuális panelhez megjelenő menü milyen választékot kínál, s csak utána bocsátkozunk a részletek megtárgyalásába.

Mielőtt azonban továbbmennénk, néhány általános tudnivalóra hívjuk fel a figyelmet.

Ha a Control Center-rel dolgozunk, minden panel esetén igaz, hogy a legtöbb időt két, egymást kiegészítő tevékenységgel töltjük, nevezetesen:

- az **adatok kezelésével** (ide tartozik mindenfajta adatmódosító, megjelenítő funkció), amelyhez a képernyőn a "*Data*" szó kapcsolódik, illetve
- a különböző **megjelenítési** módok (lekérdezés, jelentés, címke) formátumának **megtervezésével**, amelyet a "*Design*" angol szó jelöl.

A két üzemmódot bármely panelben kényelmesen és gyorsan váltogathatjuk az **F2** (*Data*) és a **Shift-F2** (*Design*) billentyűk leütésével.

Ez egyben azt is jelenti, hogy minden panelben két munkafelületre juthatunk: a *tervezésére* és az *adatmegjelenítésére.*

A **megjelenítés munkafelületei**ről elöljáróban csak annyit érdemes megemlíteni, hogy némelyik panel nem is rendelkezik saját megjelenítő felülettel, így a **Shift-F2** hatására a Data panel megjelenítője (BROWSE, EDIT) lép működésbe. Ezek a modulok általában nyomtatnak, vagy nyomtatási formátumban jelenítenek meg adatokat a képernyőn. Ezeket nem lehet munkafelületnek nevezni, hiszen "dolgozni" rajtuk nem tudunk.

A **tervezés munkafelületei**ben sok közös vonás van, de persze a különböző céloknak megfelelő lényeges eltéréseket is találunk. Legfontosabb közös vonásuk az, hogy mindegyiken vizuális, grafikus megjelenítés segíti a munkát, és mindegyiken használhatók a teljes-képernyős szerkesztőbillentyűk, különös tekintettel a blokkkijelölés, -áthelyezés és -másolás funkciókra.

A tervezés során különböző fájlok keletkeznek. Minden panel tervezőfunkciója rendszerint több (leginkább három) fájlt hoz létre. Az első tartalmazza a definiciókat (például a képernyőformátum fájl esetén az **.scr** kiterjesztésű), egy másik az ennek megfelelő programkódot (**.fmt**), és végül a harmadik, a lefordított, futtatható változatot (**.fmo**). A következő táblázatban feltüntettük ezt a fájl hármast a különböző fájltípusok esetén.

Állománytípus	Panel	Definíció	Programkód	Lefordított
Adatfájl	Data			.DBF
Nézet	Queries	.QBE		.QBO
Módosító nézet	Queries	.UPD		.UPO
Formátum	Forms	.SCR	.FMT	.FMO
Riport	Reports	.FRM	.FRG	.FRO
Cimkefájl	Labels	.LBL	.LBG	.LBO
Program	Applications		.PRG	.DBO
SQL program	Applications		.PRS	.DBO
Alkalmazás	Applications	.APP	.PRG	.DBO

4.5. Az adatfájltervezés, -kezelés - Data panel

Az adatfájl a nyilvántartandó adatokat tartalmazza, rekordokba és mezőkbe szervezve. Ezek felépítéséről, szerkezetéről szóló részletesebb információk az erre vonatkozó fejezetben megtalálhatóak. A Data panel segítségével számos adatfájltervező és adatkezelő funkció megoldható az utasítások különösebb ismerete nélkül. Így például:

- új adatfájl létrehozása, vagy meglévő szerkezet módosítása
- egy adatfájl rekordjainak fizikai illetve logikai (indexelés) rendezése
- rekordok hozzáadása, módosítása, törlése
- az adatfájl szerkezetének kinyomtatása.

Minden más panelhez hasonlóan itt is világosan elkülönül a tervezés és a konkrét adatok kezelésével kapcsolatos munkafelület, igaz ugyan, hogy egyikből néha át lehet lépni a másikba.

Az adatfájlokkal végzendő tervezési vagy kezelési műveletekhez ki kell választanunk először azt az adatfájlt, amelyikkel dolgozni akarunk. Ehhez egyszerűen a sávkurzort a bennünket érdeklő adatfájlra kell "húznunk", majd az **Enter**-t megnyomni. Ilyenkor a következő menü jelenik meg:

Use file	Modify stucture/order	Display data

A **Use file** opció megnyitja az adatfájlt és az éppen aktuálisat lezárja. Ha a sávkurzor éppen az aktuális adatfájlon van (vonal felett) az *Use file* helyett a **Close file** (fájl lezárás) jelenik meg. Hatására egyébként semmi nem történik.

A **Modify stucture/order** az adatfájltervezés munkafelületét nyitja meg.

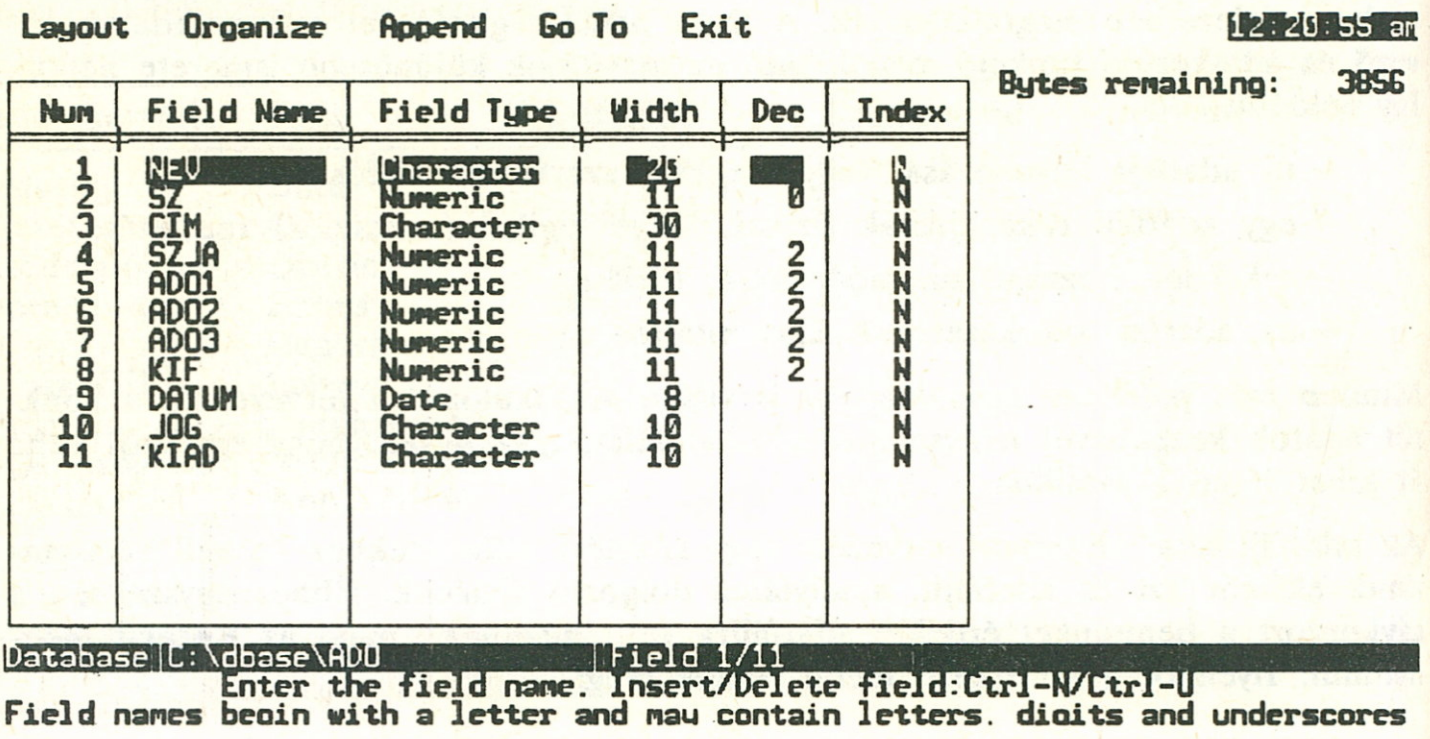
A **Display data** pedig az adatmegjelenítés, -módosítás műveletekhez nyit utat.

Elérhetjük ugyanezt másként is. Ha a sávkurzort a kívánt fájlra húzzuk az **F2** billentyűvel az **adatmegjelenítés, Shift-F2** billentyűkre pedig a **tervezés** képernyője jelenik meg.

Ha **új adatfájlt** akarunk megtervezni, létrehozni, álljunk a Data panel **< create >** opciójára. Az **Enter** lenyomására máris a tervezés munkafelületén vagyunk.

4.5.1. Adatfájl tervezése, létrehozása

Akár új adatfájlt hozunk létre, akár egy létező fájl nevét kiválasztva akarjuk a fájl szerkezetét módosítani a képernyőn mindkét esetben a 4.3. ábra jelenik meg.



4.3. ábra

A menürendszer itt is az **F10** billenyű segítségével aktivizálható. A menü az alábbi opciókat tartalmazza.

Layout : A kialakított adatfájl szerkezetének az elmentését és nyomtatását, valamint leírásának módosítását segíti elő.

Organize : Ez az opció az adatfájlokhoz történő közvetlen hozzáférést támogatja. Itt valósítható meg pl.: a rekordok fizikai és logikai (indexelés) rendezése, rekordok fizikai törlése, stb...

Append : Bár új rekordok beillesztésére legalább két módszert is ismerünk (pl.: APPEND, BROWSE), ez a menüelem mégis hasznos, mert lehetővé teszi egész rekordcsoportok egyidejű átmozgatását.

Go To : Az adatfájltervező képernyőn történő gyors mozgást segíti.

Exit : Visszatérés a Control Center-be.

A megjelenő táblázat rovatai sorrendre a *mezők sorszámát*, *nevét*, *típusát*, *hosszát*, *tizedesjegyeinek számát*, *indexjelzőjét* tartalmazzák illetve várják új fájl esetén.

4.5.1.1. Új adatfájl szerkezetének tervezése, létrehozása

A Data panel < *create* > opcióját kiválasztva azonnal megjelenik a fájltervezés munkafelülete, rajta a táblázattal, amelynek kitöltésével (a mezők és tulajdosnságaik kijelölésével) a rekordszerkezet máris meghatározható. A kijelölés a **Layout • Save this database file structure** opciójával, vagy az utolsó mező megnevezése utáni **Ctrl-End** leütésével menthető el.
A megjelenő ablakban az adatfájl nevét kell megadnunk.

A képernyőn a szerkesztőbillenytűk nagy része segíti a munkát.
Az adatfájl szerkezetébe a **Ctrl-N** billentyűk alkalmazásával *szúrhatunk* be újabb mezőket. Mező *törlése* a **Ctrl-U** leütésével történik. A kurzort mindkét esetben a megfelelő mezőre kell mozgatnunk.

Amennyiben megjelöltünk mezőt indextagként (*Index*="Y"), úgy az adatfájl nevével egyező, összetett indexfájl, az *alapindexfájl* automatikusan létrejön.
Az adatfájl szerkezetének a meghatározása után a menürendszer segítségével rekordokat vihetünk be (szándékunkról a rendszer is faggat), és az adatokat is rendezhetjük. (Lásd: Adatok megjelenítése, módosítása).

4.5.1.2. Fájlszerkezet módosítása

A Data panelből a fájlkiválasztás (sávkurzor) és a **Modify structure/order** opció megjelölése (vagy egyszerűen a Shift-F2 megnyomása) után ismét az adatfájltervező munkafelület aktivizálódik, de azonnal megjelenik az **Organize** menüpont rolómenüje is. Az **Esc** billentyű megnyomására ez eltűnik és hozzáférhetünk a fájlszerkezet kitöltött táblázatához, amelynek módosítására az új létrehozásánál megismertek vonatkoznak.

4.5.1.3. Fájlszervezés, rendezettség meghatározása

A sávkurzorral válasszuk ki a kívánt adatfájlt a Data panelből. Ismét a fájlszerkezet módosításánál ábrázolt menü jelenik meg, amelyből megintcsak a **Modify structure/order** opciót kell kijelölnünk. Megjelenik a fájltervező munkafelület, és láthatóvá válik az **Organize** menüpont rolómenüje is. Ez szolgál a különféle adatrendezési funkciók elérésére.

A könnyebb megértés érdekében fel lehet lapozni a dBASE IV programozás alapjai c. fejezetet és a függelékben az INDEX és SORT parancsokat.

Create new index
Modify existing index
Order records by index
Activate .ndx index file
Include .ndx index file
Remove unwanted index tag
Sort database on field list

Create new index : kiválasztásával új egyszerű indexfájl (.ndx) vagy új indextag hozható létre az összetett indexfájlban. A rendezés kulcsát irányát és az indexfájl vagy indextag nevét kell megadnunk.

Modify existing index : már régebben létrehozott indexfájl vagy -tag kulcsa és neve módosítható.

Order records by index : kijelölhető a megjelenő listából az elsődleges index neve.

Activate .ndx index file : az aktuális adatfájlhoz tartozó indexfájlok megnyitására szolgál.

Include .ndx index file : választásával az aktuális katalógusfájlba belekerülő indexfájlokat jelölhetjük ki.

Remove unwanted index tag : az összetett indexfájl nemkívánt eleme törölhető.

Sort database on field list : fizikailag rendezi az adatfájlt a megadott mezők és rendezési irány szerint.

A rendezési adatok beviteléhez a kapcsos zárójelek az **Enter** leütésére nyílnak.

4.5.1.4. Az adatfájlszerkezet megjelenítése

A Data panel menürendszere támogatja az adatfájl szerkezetének a **kinyomtatását**, meghatározott formák között. Ennek érdekében válasszuk ki a **Layout • Print database structure** opcióját.
A nyomtatás a **Begin printing** opció megválasztásával történik. Előtte beállíthatjuk a nyomtatási paramétereket. A beállítási művelet a listaformátum-fájlnál leírtakkal azonos, ezért az útmutatót lásd ott.

Kiírhatjuk a rekordszerkezetet **szöveges fájlba** is. Ez esetben a **Print database structure** kijelölése után a **Destination** opciónál a *szóköz* billenyű segítségével kijelölhetjük a kimenet helyét. A fájlba írás itt is a **Begin printing** megválasztásával történik.

4.5.2. Adatok kezelése

4.5.2.1. Adatfájl megnyitása, zárása

A sávkurzorral válasszuk ki a kívánt adatfájlt a Data panelből. Ismét a fájlszerkezet módosításánál ábrázolt menü jelenik meg.

A **Use/Close file** opció a kiválasztott adatfájlt megnyitja (használatba veszi) és ezzel le is zárja az előzőleg használatban lévőt (ha volt ilyen).

A *Display data* opció is megnyitja a kijelölt adatfájlt, erről bővebben az adatok megjelenítésénél szólunk.

4.5.2.2. Adatok megjelenítése, módosítása

A megjelenítés első mozzanata az aktuális adatfájl megnyitása. Ehhez elgendő a fájl kiválasztása után a **Display data** opciót megjelölni. Beléphetünk ugyanide a tervezés munkafelületről is, ha ott az *Append • Enter records from keyboard* opcióját választjuk.

A megjelenítéssel az adatok módosítása is lehetővé válik. Erre két formát kínál a dBASE IV.
Az egyik formában egy rekord jelenik meg mezőneveivel és adataival a képernyőn, amely az **EDIT** utasításnak megfelelő formátum. A másik esetben táblázatos formában láthatjuk az adatokat, ahol egy rekord adatait egy sor tartalmazza. Ez a **BROWSE** utasítás megjelenítési módja. Ilyenkor ha a táblázat a képernyőn nem fér el, függőlegesen és vízszintesen is "görgethetjük" a képernyőt. A két mód között az **F2** billentyűvel menetközben is válthatunk.

A két forma menürendszere mindössze egyetlen menüpontban különbözik – ez a **Fields** –, a többi működése teljesen azonos. Nézzük ezeket sorra:

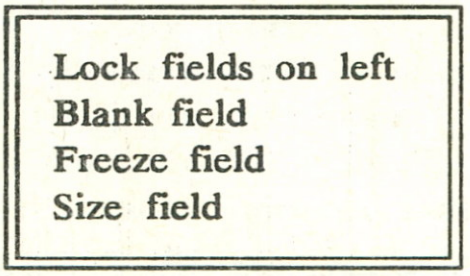
Records : Az adatok törlésére, bevitelére és az eredeti állapot visszaállítására szolgál.

Fields : Meghatározhatjuk a Browse táblázat megjelenő mezőit, azok rögzítését a görgetés során, stb.

Go to : A rekordmutató mozgatására szolgál.

Exit : Visszatérés a Control Center alapfelületére.

A *Fields* kiválasztásával új ablak jelenik meg:



A **Lock fields on left** után megadhatjuk, hogy a vízszintes görgetésnél balról számítva hány mezőt akarunk rögzíteni. Ezek mindig a helyükön maradnak, csak a többi mező mozog.

A **Blank field** kiüríti az éppen aktuális mezőt.

A **Freeze field** segítségével korlátozhatjuk a használható mezőket. A kijelöltön kívüli mezőket a kurzorral nem érhetjük el.

Ha nem kívánunk egyes mezőket teljes szélességében látni, a **Size field** segítségével korlátozhatjuk ezt.

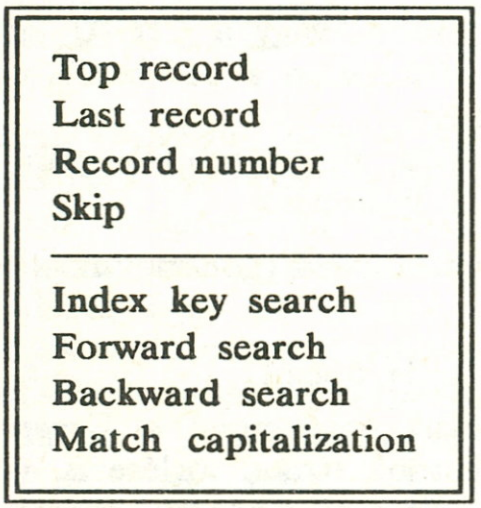
A megjelenő adatokat egyszerű begépeléssel módosíthatjuk, új rekordokat is vihetünk az adatfájlba. A módosítás során a szerkesztőbillentyűket használhatjuk az adatbevitelhez és a képernyőn való mozgáshoz.

A **Records • Blank record** opciójával kiüríthetjük az aktuális rekord minden mezőjét.

A módosítások visszaállíthatók a **Records • Undo change to record** opcióval.

4.5.2.3. Az adatok keresése.

Nagyobb méretű adatfájlok adatainak megjelenítésénél, módosításánál is szükségünk lehet arra, hogy a benünket érdeklő rekordra gyorsan rátaláljunk. Erre szolgál a megjelenítés (Display data) menürendszerében a **Go To** menüpont. A kiválasztásakor a következő lehetőségeket kínálja:



Top record : az adatfájl első rekordjára ugrik. Ha van index, annak figyelembevételével.

Last record : az adatfájl utolsó rekordjára ugrik. Ha van index, annak figyelembevételével.

Record number : után megadhatjuk a bennünket érdeklő rekord sorszámát. A rekordmutatót erre állítja.

Skip : után lépésszámot adhatunk meg előre és hátra (negatív előjellel). Az éppen érvényes rekordról ennyivel fog tovább lépni a rekordmutató, figyelembevéve az esetleges indexet is.

Index key search : opciót csak akkor választhatjuk, ha van aktuális (nyitott és kijelölt) index. Ilyenkor a keresési kulcsot kell megadnunk.

Forward/Backward search : Index híján kereső kifejezést is megadhatunk. Ilyenkor mindig az aktuális mezőben történik a keresés előre (forward) vagy hátra (backward). A kifejezésnél figyelembe kell venni a mező adattípusát.

Match capitalization : keresésél beállítható a kis és a nagybetűk megkülönböztetése.

A keresés végrehajtása után a rekordmutató a keresett rekordra áll, amely az *EDIT* módban a rekord megjelenését, a *BROWSE* módban a megjelenítés mellett a kurzor rekordra állítását is jelenti.

4.5.2.4. Rekordok törlése.

Az aktuális adatfájlból a rekordokat két lépésben lehet kitörölni.

Először meg kell jelölni a törölni kívánt rekordokat. Ezt a **Records • Mark record for deletion** opció segítségével tehetjük meg, de a **Ctrl-U** is ugyanezt eredményezi (ha a rekord már nem volt korábban is törlésre jelölve).

A törlésjel vissza is állítható a **Clear deletion mark**, vagy a **Ctrl-U** megnyomásával. Ha a státusz-sor látható, a jobb oldali blokkjában a *del* betűk utalnak arra, hogy az aktuális rekord éppen törlésre kijelölt.
Minden rekord törlésjelét egyszerre állíthatjuk vissza, ha a tervezés munkafelületén az **Append • Unmark all records** opcióját választjuk.

A *törlés második lépcsőfoka* a rekordok fizikai eltávolítása (gondoljunk a PACK parancsra). A gyakorlati végrehajtás menete :

- A törlésre kijelölés után válasszuk ki az **Exit** opciót.
- Lépjünk át a tervezési felületre, és válasszuk az **Append • Erase marked records** opcióját. Ezzel a kijelölt rekordok fizikai törlése is megtörténik, ami magától érthetődően a rekordlétszám és rekordsorszámok megváltozását is maga után vonja.

4.5.2.5. Új rekordok csoportos bevitele

A csoportos rekordbevitel más fájlokból kivételesen a tervezés munkafelületéhez kapcsolódik. A Data panelből a sávkurzorral válasszuk ki a kívánt fájlt, majd a megjelenő menüből a **Modify structure/order** opciót. A tervezésnél már megismert képernyő menüjéből ezúttal az **Append** menüelemet jelöljük meg. Ezzel az alábbi lehetőségeket választhatjuk:

Enter records from keyboard
Append records from dBASE file
Copy records from non-dBASE file

Append records from dBASE file opció megválasztása után megjelenik az aktuális katalógusfájl összes adatfájlja. A sávkurzorral meghatározott dBASE IV fájl tartalma bekerül az aktuális adatfájlba. Ügyelni kell az adatfájlok szerkezetének egyezésére.

Copy records from non-dBASE file az aktuális adatfájlba, más típusú adatkezelő rendszerekből importálhatunk adatokat. A lehetséges adatkezelő rendszerek listája megjelenik képernyőn az alapfelület *Tools • Import* opciójával azonosan.

Az *Enter records from keyboard* gyakorlatilag a *Display data* megjelenítés és beviteli munkafelületet aktivizálja, és csak kényelmi szempontokból hívható innét is.

4.6. Nézetfájlok tervezése - Query panel

Gyakran előfordul, hogy a feldolgozás valamely fázisában a rekordoknak és mezőknek csak egy bizonyos csoportjára van szükségünk. Például egy alkalmazás során csak a személyzeti nyivántartásból csak a hölgyekkel kívánunk foglalkozni, és róluk is csak az adatok bizonyos része érdekel bennünket. Ez tulajdonképpen azt jelenti, hogy az eredeti adatfájlok egyfajta kivonatára vagyunk kíváncsiak. Bizonyos tekintetben hasonlíthatjuk ezt az ábrázoló geometria (alul-, felül-, stb) "nézet"-eihez, ahol az ábrázolt tárgynak csak egy része, oldala látszik. Ennek nyomán mi is **nézetfájl**nak fogjuk nevezni az eredeti adatfájlokból különféle módokon előállított kivonatokat. Maga a "*query*" szó lefordítása sem túl félrevezető, ugyanis az egyik jelentése: **kérdőív**.

Az adatok kiválogatása a nézetfájlba kétféle módon történhet. Válogathatunk **horizontálisan**, azaz válogathatunk a megjelenítendő **mezők** között.

Szelektálhatunk **vertikálisan** is, azaz válogathatunk a rekordok között. Ez utóbbi valójában azt jelenti, hogy a rekordok egy részét "*kiszűrjük*" az általunk meghatározott **szűrőfeltételek** segítségével. (Személyzeti példánk esetében azokat a rekordokat szűrjük ki, ahol a NEME = "nő".)

A *query panel* segítségével ez a feladat (és persze ennél lényegesen bonyolultabb is) különösebb előképzettség nélkül megoldható. Míg az előző fejezetben tárgyalt Data panel főként az adatfájlok, indexfájlok szerkezetének felépítését támogatta, itt a már létező adataink különféle csoportokba szervezését segítő eszközökkel találkozhatunk.

A továbbiak megértéséhez pontosítanunk kell a nézet fogalmát. A dBASE IV kétféle jellegű nézetfájl használatát engedi: az egyik *passzív nézet*, – amelyet a továbbiakban egyszerűen nézetnek vagy nézetfájlnak fogunk hívni – ; a másik (természetesen) az *aktív*, amit majd **módosító nézet**nek nevezünk.

Most pedig nézzük meg, miben térnek el egymástól:

Nézet

Lehetőséget ad egy vagy **több adatfájl** rekordjai közül kiválasztani azokat, amelyek megfelelnek az általunk megadott feltételeknek. Az így kiszűrt rekordok (és meghatározott mezőik) megjeleníthetők. Ezek a folyamatok a memóriában játszódnak le, ezért gyorsabbak, mint ha ugyanezt a kigyűjtést fizikailag is megcsi-

nálnánk, sőt így elkerüljük a redundáns adattárolást is. A szűrőfeltételrendszer és a mezőkijelölések (*nem a kigyűjtött rekordok*!) tárolhatóak egy fájlban —, ez a *nézetfájl* —, így később pusztán a nevére történő hivatkozással (SET VIEW TO) vagy a Control Center-ből egyszerű kiválasztással bármikor aktivizálható anélkül, hogy újra meg kellene határoznunk a sokszor elég bonyolult feltételek sokaságát. Hasonlóan a dBASE III PLUS-hoz az egyszerű nézet használata során **adatokat nem módosíthatunk.**

Módosító nézet

Az előbbivel szemben a módosító nézet **csak egyetlen adatfájlra** vonatkozhat. Célja az, hogy az előzőekhez hasonlóan kijelölt rekordcsoportokon és mezőkön gyorsan és hatékonyan tudjuk a következő négy **művelet** valamelyikét elvégezni:

- új rekordok hozzáadása (APPEND)
- rekordok kijelölése törlésre (DELETE)
- rekordok tartalmának megváltoztatása (REPLACE)
- a törlésjelölések eltávolítása (RECALL)

A legfontosabb különbség a két fajta nézet közt az, hogy míg az első csak megjelenítésre képes, addig a másik csak adattartalmak megváltoztatására alkalmas. Az eltérő funkciók ellenére mindkét típust ugyanazon a munkafelületen kell megterveznünk.

A többi panelhez hasonlóan itt is világosan elkülönül a tervezés és a konkrét adatok kezelésével kapcsolatos felület. Olyannyira így van ez, hogy a Query panelnek tulajdonképpen nincs is adatmegjelenítő, adatmódosító munkafelülete, hanem a Data panelnél megismert BROWSE, EDIT megjelenítőket használja.

A nézetfájlokkal végzendő tervezési vagy kezelési műveletekhez ki kell választanunk először azt a nézetfájlt, amelyikkel dolgozni akarunk. Ehhez egyszerűen a sávkurzort a bennünket érdeklő nézetre kell "húznunk", majd az **Enter**-t megnyomni. Ilyenkor a következő menü jelenik meg:

Use view	Modify query	Display data

Use view opció megnyitja a nézetfájlt és az éppen aktuálisat lezárja. Ha a sávkurzor éppen az aktuális nézeten van (vonal felett) az *Use view* helyett a **Close view** (nézet lezárása) jelenik meg. Hatására egyébként semmi nem történik.

Modify query a nézetfájltervezés munkafelületét nyitja meg.

A **Display data** pedig az adatmegjelenítés, -módosítás műveletekhez nyit utat, és teljesen azonos a működése a Data panel ugyanezen opciójával. Az ismertetését lásd ott.

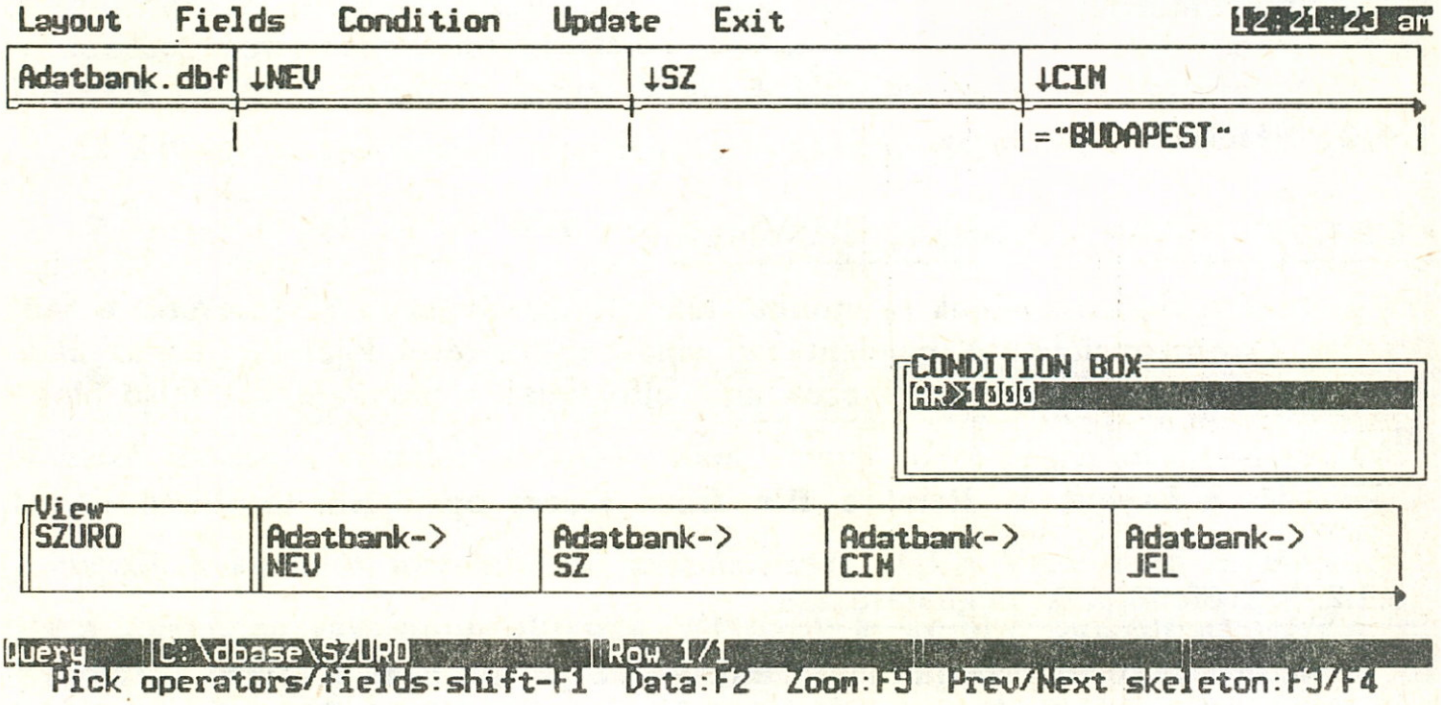
Alternatívaként a Data panelnél megismert **Shift-F2** (tervezés), **F2** (megjelenítés) itt is használható.

Ha **új nézetfájlt** akarunk megtervezni, létrehozni, álljunk a Query panel **< create >** opciójára. Az **Enter** lenyomására máris a tervezés munkafelületén vagyunk.

Most tehát nézzük meg magát a tervezési folyamatot.

4.6.1. A nézetfájl-tervezés munkafelülete

Akár új nézetet hozunk létre, akár egy létező fájl nevét kiválasztva akarjuk a nézet szerkezetét módosítani a képernyőn mindkét esetben a következő kép jelenik meg.



4.4. ábra

A menürendszer itt is az **F10** funkciós billentyű segítségével aktíválható, amelynek a következő menüpontjai vannak:

Layout : Az aktuális adatfájl kijelölésére, törlésére és a megfelelő fájlok létrehozására szolgál.

Fields : A horizontális válogatásra, azaz a nézetfájl mezőinek a kijelölésére és törlésére használható.

Condition : A vertikális, azaz a rekordok közötti válogatófeltételek meghatározását támogatja.

Update : Az aktuális adatfájlon néhány adatkezelő művelet elvégzését teszi lehetővé.

Exit : Visszalépés a Control Centerbe.

A munkafelületen (képernyőn) két blokk különíthető el világosan:

A *felső blokk*ban megjelenik az aktuális adatfájl összes mezője. Ha több adatfájl van, akkor a felső blokk is fájlonkénti részekre tagolódik. Minden mező alatt ablak látható, amely a (vertikális) rekordok közötti válogatást végző **szűrőfeltételek** bevitelére szolgál. Természetesen nemcsak egy, hanem több mezőhöz is határozhatunk meg feltételeket.

A *második (alsó) rész*ben jelölhetjük ki azoknak a **mezőknek a körét**, amelyekre a továbbiakban szükségünk van. Ezzel a nézetfájl mezőit határozzuk meg.

A blokkok ill. részek között az **F3**, **F4**, a mezők közt pedig a **Tab**, **Shift-Tab** billentyűkkel mozoghatunk.

4.6.2. Nézetfájl tervezése

4.6.2.1. Adatfájlok kiválasztása, eltávolítása

Ha nincs aktuális adatfájlunk (a munkafelület üres), válasszuk ki a **Layout • Add file to Query** opcióját. A bejelentkező ablak segítségével kijelölhetjük az *aktuális adatfájlt*, amelynek mezői ezek után elfoglalják a munkafelület felső blokkját.

A kijelölés a **Layout • Remove file from Query** opciójával törölhető.

4.6.2.2. Szűrőfeltételek meghatározása

Ha a rekordok között válogatni akarunk — ez persze nem kötelező — a vertikális válogatáshoz szükséges szűrőfeltételeket az alábbi lépésekben határozhatjuk meg:

● Mozgassuk a kurzort a **Tab** és a **Shift-Tab** billentyűk segítségével a mező alá, amelyhez feltételt akarunk kapcsolni. Ha több adatfájlunk is van, az **F3**, **F4** billentyűkkel válthatunk köztük.

● Határozzuk meg a feltételt. A feltétel lehet bármilyen kifejezés, matematikai művelet vagy függvény. A kijelölést elősegíti a **Shift-F1** leütése, amely választható mező-, művelet- és függvénylistát kínál fel. Ugyanezt elérhetjük a **Condition • Add condition** opciójával is, amelynek elsősorban a *módosító nézetnél* van fontos szerepe.

Tetszés szerinti számú feltételt jelölhetünk ki a fenti lépések ismétlésével.

Példa:

Nev	Cim	Telefon	Beosztás
	BUDAPEST		

Az aktuális adatfájlból ezek után csak a budapestiek rekordja érhető el.

A Condition box alkalmazásával ugyanez:

```
Nev = "Budapest"
```

4.6.2.3. A mezők kijelölése

A szűrőfeltétel meghatározása után a felület alsó blokkjában kijelölhetjük azokat a mezőket, amelyekre a továbbiakban szükségünk van. Az adatok megjelenítésekor csak ezeket a mezőket fogjuk látni. A kijelölés az alábbi lépésekből áll:

- Álljunk a kiválasztott adatfájlra a felső blokkban.
- Ha az *összes mezőt* meg akarjuk jeleníteni, álljunk a kurzorral az adatfájl nevével címzett rovatba, ha csak valamelyiket, akkor a kívánt mező alá lépjünk a **Tab**, **Shift-Tab** billentyűk segítségével.
- Válasszuk ki a **Fields • Add field to view** opcióját — vagy nyomjuk meg a **F5** gombot —, és a kívánt mező(k) megjelenik (-nek) az alsó blokkban. A mező törlése a nézetből **Remove field from view** opció alkalmazásával történik.

A mezők a kijelölés sorrendjében jelennek meg alul a View című blokkban.

Ha a sorrenden változtatni akarunk, válasszuk ki az alsó blokkot az **F3/F4** billentyűkkel. Álljunk rá a mozgatni kívánt mezőre és nyomjuk le az **F7** billentyűt. A kurzorral haladjunk arra a pozícióra, amelyen el akarjuk helyezni a kijelölt mezőt, utána üssünk **Entert**.

Ha nézetből mezőket akarunk törölni, mozgassuk egyenként a kurzort a nemkívánt mezőkre és az **F5** leütésével a mezők eltűnnek a nézetből.

Példa:

Nev	Cim

A fájl csak az emberek nevét és címét fogja csak tartalmazni, azok közül is csak a budapestiekét.

4.6.2.4. Számított mezők

Az adatfájl mezőiből gyakran hozunk létre számított, származtatott adatokat. Gondoljunk például az ismert *nettó érték + általános forgalmi adó = bruttó érték* összefüggésre. Ez megoldható a nézetfájl létrehozása esetén is. Kijelöljük a megfelelő mezőket a műveleti jelekkel (operátorokkal) együtt, és az így számított adat belekerül a nézetfájlba, egy külön ún. **számított mezőbe**. A számított mező az eredeti mezők értéke és a műveleti jelek által diktált értéket automatikusan felveszi.

Számított mezőt az alábbiak szerint hozhatunk létre:

- A **Fields Edit field name** opciójával határozzuk meg annak a mezőnek a nevét, amelybe a számítás eredménye kerül.

- A **Fields Create calculated field** opciója segítségével megadjuk a megfelelő mezőket és a közöttük alkalmazandó műveleti jeleket, operátorokat. Ezt segíti a **Shift-F1** alkalmazása.

- Ha a számított mezőn állunk, az **F5** billentyű működtetésével az előzőek szerint konstruált számított mezőt hozzáfűzzük a nézetfájlhoz.

4.6.2.5. Több adatfájl összekapcsolása a nézetfájlban

Több adatfájl felhasználásával is létrehozhatunk nézetfájlt.

A kijelölt egyetlen adatfájl mellé (a képernyőn alá) a **Layout • Add file to Query** opció segítségével új adatfájlokat is "behozhatunk".

A szűrőfeltételek és mezők kijelölése a már ismertetett módon történik, ahol az adatfájlok között az **F3/F4** billentyűkkel navigálhatunk.

Több fájl esetén a munkát csak akkor tudjuk bevezetni, ha a fájlokat összekapcsoltuk.

Az összekapcsolás mezőkön keresztül történik. (A lehetőség jobb megértése érdekében érdemes elolvasni a függelékben a SET RELATION TO utasítást.)

- Álljunk az alapfájlnak –, amihez mást kapcsolni akarunk – arra a mezőjére, amellyel a kapcsolatot meg akarjuk teremteni a másik fájlban.

- Válasszuk a **Layout • Create link by poiting** opcióját – erre megjelenik a mező rovatában a *LINK* jel –, majd mozgassuk a kurzort a másik fájlban arra a mezőre, amelynek (az alapfájl mezőértékével) egyező értéke fogja majd a rekordot az alapfájl rekordjához kapcsolni. Nyomjunk **Entert** és itt is feltűnik *LINK* jel, amely a kapcsolatra utal.

4.6.2.6. A nézetfájl elmentése, kilépés

Válasszuk ki a **Layout** • **Save this Query** opcióját a nézetfájl elmentése érdekében.

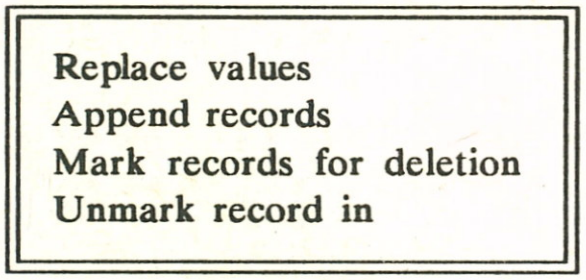
Az elmentés során egy **.qbe** kiterjesztésű fájl jön létre, amely a tervezésünk definícióit (szűrőfeltételek, mezőkijelölések, kapcsolók, stb) tartalmazza. Ez az adatok megjelenítésére nem is használható, csak akkor, ha a SET VIEW TO utasítás (vagy a Query panelnél az F2 gomb) hatására a rendszer **.qbo** kiterjesztésű tárgykódú fájlra fordítja le.
A módosító nézetfájl **.upd** illetve **.upo** kiterjesztést kap.

A munka befejezése az **Exit** menüpont kijelölésével történik, amelyben az **Abandon...** opcióval el is vethetjük a módosításainkat a Control Centerbe való visszatérés előtt.

4.6.3. A módosító nézet: csoportos műveletek egy adatfájlon

Sokszor kerülünk abba a helyzetbe, hogy az adatfájlokon néhány, több rekordot is érintő ún. *csoportos* műveletet kell elvégeznünk. Például minden rekordban 0-ra kell állítanunk az egyik mezőt, stb. Erre szolgál a **módosító nézetfájl**, amelynek tulajdonképpen semmi köze az igazi nézetfájlhoz. Segítségével **csak egy adatfájlon** végezhetők műveletek, és az adatfájl tartalma is változik a nézetfájl-lal ellentétben.

Használatához a nézetfájlnál megismert módon ki kell választani az adatfájlt, majd a műveletek az **Update** • **Specify update operation** opció megválasztásával jelölhetők ki. A megjelenő menü ezeket kínálja fel:



Replace values : A kijelölt mező értékének a megváltoztatására szolgál.

Append records : Rekordokat fűz a fájl végéhez.

Mark records for deletion : Rekordok kijelölése törlésre.

Unmark record in : Törléskijelölés megszüntetése.

A mezők tartalmának megváltoztatásakor (**Replace values**) az eddig szűrőfeltételnek használt rovatokba a mező kívánt új értékét előállító *kifejezést* kell begépelni.

A szűrőfeltételeket ezúttal a **Condition • Add condition box** opció megválasztásakor megjelenő feltétel-ablakba (*Condition box*) kell begépelnünk.

A kijelölt műveleteket csak az **Update • Perform update** opció jóváhagyására hajtja végre a rendszer. Ezt azonban, ha a módosító nézetfájlt elmentettük, későbbre is halaszthatjuk.

Figyelem ! A műveletek végrehajtása az eredeti adatfájlon történik, így annak adattartalma megváltozik.

A fájl elmentésével és a tervezéssel kapcsolatos minden tudnivaló az "igazi" nézetfájlokéval azonos.

4.6.4. A nézetfájl adatainak megjelenítése, tárolása adatfájlként

Az **F2** leütésével megjeleníthetjük a megfelelő adatokat a képernyőn.
Ilyenkor – ha módosítottunk a nézetfájlon – a rendszer először lefordítja **.qbo** kiterjesztésű tárgyprogrammá, majd ennek felhasználásával a nézet eredményezte adatokat a *BROWSE* képernyőn jeleníti meg.

Ebből a **Shift-F2** segítségével térhetünk vissza a tervezéshez.

Ha a nézetfájl eredményeként született adatokat külön adatfájlban is tárolni akarjuk, válasszuk ki a **Layout • Write view as database file** opcióját, amely létrehozza a szűrőfeltételt kielégítő és a kijelölt mezőket tartalmazó adatfájlt, a **.dbf** kiterjesztésű *nézet-adatfájl*t.

4.7. Képernyőformátumok - Forms panel

Ha adatfájlok tartalmát képernyőn szeretnénk megjeleníteni vagy módosítani, alapvetően két lehetőség közül választhatunk. Az egyik a *Browse* utasítás, de ennek megjelenési formája kötött (táblázat, amelyben egy sor egy rekordot reprezentál). A másik, az *Edit* által megjelenített képernyőn (amely önmagában meglehetősen szegényes, de elrendezése a Forms panel segítségével rugalmasan kialakítható) egyszerre csak egy rekord látható. A Forms panel segítségével ízlésünknek megfelelő formában helyezhetjük el a képernyőn a rekord mezőit, különböző típusú keretekkel, vonalakkal emelhetjük ki a lényegesnek vélt részeket, s még ún. számított (a fájlban fizikailag nem szereplő, valamiyen általunk megadott algoritmussal képzett) mezőket is használhatunk. Az ily módon megtervezett formátumokat nézetfájlokra építve egyetlen adatfájlhoz teljesen eltérő célú és megjelenésű képernyőket hozhatunk létre.

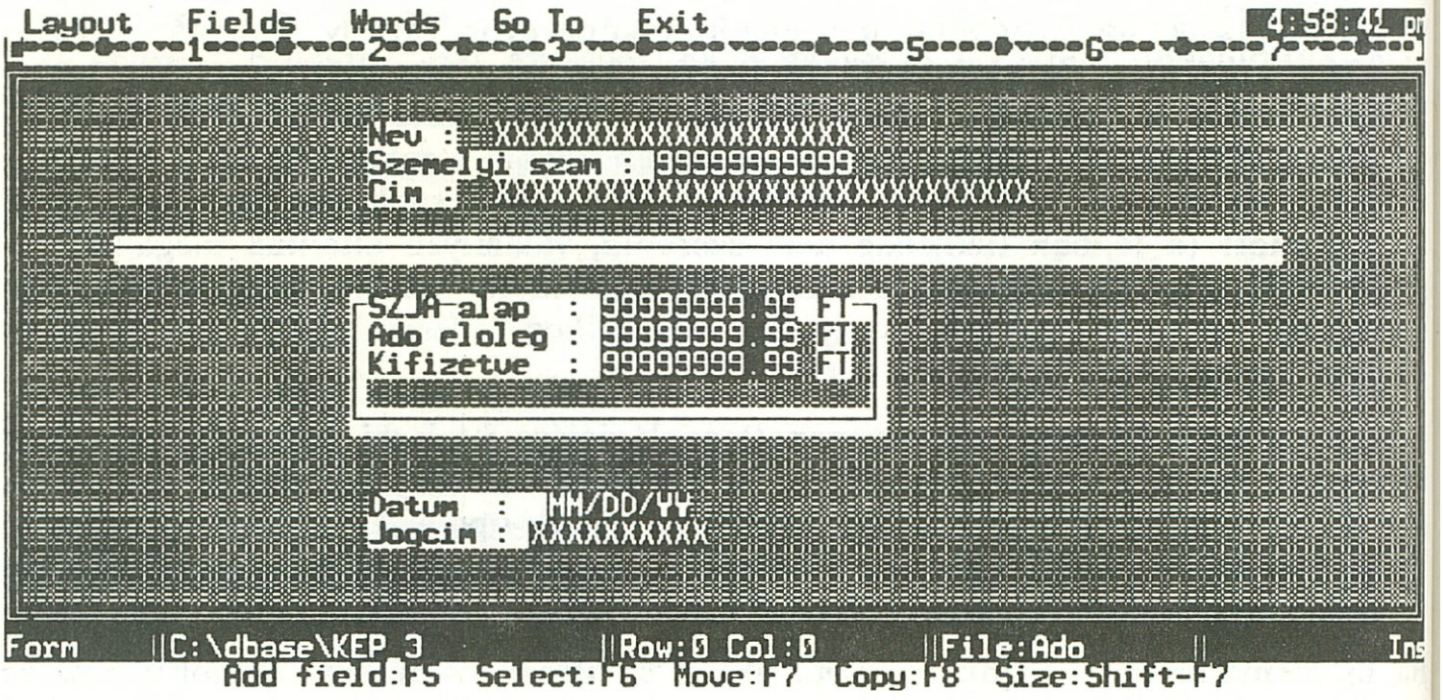
A korábbiak (Data panel, Query panel) analógiájára itt is ki kell választanunk a bennünket érdeklő **formátumfájlt** – a rövidség kedvéért így fogjuk nevezni a képernyőformátum-fájlt – , majd a **Display data** opcióval az adatok megjelenítése mellett, a **Modify layout** választásakor pedig a tervezés mellett dönthetünk. Itt is választhatjuk a **F2/Shift-F2** billentyűket is.

Ha új formátumot akarunk tervezni, akkor az ismert **<create>** opciót választva a tervezés munkafelületére jutunk.

Meg kell jegyeznünk, hogy az adatmegjelenítés nem lesz sikeres, ha korábban nem vettünk használatba egy adatfájlt, vagy nézetfájlt. Célszerű ezt megtenni új formátum létrehozása előtt is.

4.7.1. A formátumtervezés munkafelülete

Ha új formátumot tervezünk, akkor az üres munkafelület, egyébként a kiválasztott formátum jelenik meg a menüsorral. A panel menürendszere a szokásos F10 leütésével aktiválható.



4.5. ábra

A menüsor elemei:

Layout : Az adatfájl kijelölése, a mezők gyors elhelyezése, keret készítése és a formátum elmentése.

Fields : Mezők és változók képernyőre helyezése, formátumozása, beviteli előírások, műveletek meghatározása.

Words : Szövegszerkesztő jellegű funkciók, kijelölt képernyőblokkok színezése és egyéb jellemzők beállítása (pl: betűtípusok, margók, lapbeosztás stb.).

Go to : Adott sorra pozicionálás, adott szövegrészlet megkeresése, szövegrészletek kicserélése.

Exit : Kilépés a Forms panelből a tervezett formátum elmentésével vagy anélkül.

Most nézzük meg egy formátum megtervezésének lehetséges lépéseit.

A tervezéshez akkor kezdhetünk hozzá, ha adat- vagy nézetfájlt választottunk. Ha ezt nem tettük meg belépés előtt, pótolhatjuk a **Layout • Use different database file or view** opció kijelölésével.

4.7.2. A mezők és változók a képernyőn

A formátumfájl tervezése során háromfajta adat megjelenítését tervezhetjük meg a képernyőn: adatfájlok **mezőit**, **változókat** és ún. **számított mezőket**. Tekintettel arra, hogy a menürendszer számos opciója – így például a megjelenítési formázó – nem tesz különbséget ezek között, így az **adat** kifejezés bármelyik fajtát is takarhatja. Meg kell jegyeznünk azonban, hogy a számított mező szerkesztése a majdani adatbevitel során lehetetlen. Ez a számított mező természetéből fakadóan – úgy gondoljuk – annyira logikus, hogy magyarázatot nem igényel.

4.7.2.1. Mezők kiválasztása, eltávolítása

Leggyakoribb az az eset, amikor az adatfájl valamennyi vagy majdnem minden mezőjére szükség van a bevitel során. A mezők egyenkénti, lassú kijelölése helyett célszerű a **Layout • Quick layout** opciójának használata. Ebben az esetben a dBASE IV az aktuális adatfájl valamennyi mezőjét kijelöli és megjeleníti az *Edit* képernyő megszokott elrendezésének megfelelően. Ilyenkor a fölösleges mezőket törölhetjük.

Egyenként is hozhatunk a képernyőre mezőket :

• Ilyenkor célszerű mindjárt a képernyő kívánt helyére állni a kurzorral és utána a **Fields • Add field** opcióját kiválasztani. Erre az alábbi kétrészes ablak jelenik meg a képernyőn:

fájlnév	CALCULATED
mező 1	<create >
mező 2	*szám.-mező 1*
mező 3	*szám.-mező 2*
mező 4	....
....	*szám.-mező n*
mező n	

• A mezőlistából jelöljük ki a kívánt mezőt. (A számított mezőkről később lesz szó.) A választás után megjelenő almenürendszer segítségével a bemeneti adatokra vonatkozó előírások és a megjelenítés formátumának a beállítása (lásd

később részletesen) következhet, de a kiválasztott mezőt enélkül is a **Ctrl-End** leütésével a képernyőre vihetjük.

Az aktuális (sávkurzorral jelölt) mező **törlése** a képernyőről közvetlenül a **Backspace** vagy **Del** billentyű leütésével történik. Más mezők törléséhez választhatjuk a **Fields • Remove** opcióját is, ahol a megjelenő mezőlistából jelölhetjük ki a törölni kívántakat.

4.7.2.2. Változók kijelölése, eltávolítása

A formátum tartalmazhat változókat is. A változó használata érdekében válasszuk ki a **Fields • Insert memory variable** opcióját. (Ezelőtt célszerű mindjárt a képernyő kívánt helyére állni a kurzorral.) A következő ábra jelenik meg a képernyőn:

```
Name :
Variable type :
-------------------
Template
Picture functions
Edit options
```

Name : A változó nevét várja.

Variable type : A *szóköz* leütésével kereshető meg a kívánt adattípust.

A *Picture.. és Edit options* opciókat lásd lejjebb.

Az így kijelölt változót a **Ctrl-End** leütésével a képernyőre vihetjük.

A változó **törlésének** módja azonos a mező eltávolításáéval.

4.7.2.3. Számított mezők.

Gyakran előfordulhat, hogy adatbevitel vagy -megjelenítés során szükség van olyan mezőre, amelyet valamilyen algoritmus szerint más mezőkből származtatunk. Említettük már példaként *nettó érték + általános forgalmi adó = bruttó érték* összefüggést. Ez a formátumtervezés során is megoldható. Kijelöljük a megfelelő mezőket a műveleti jelekkel (operátorokkal) együtt, és az így számított adat bekerülhet a formátumba, egy külön **ún. számított mezőbe**. A számított mező az eredeti mezők és a műveleti jelek által diktált értéket automatikusan felveszi. *Adatbevitelre nem használható.*

Számított mezőt a **Fields • Add field** opciója megválasztásával konstruálhatunk. A mezők kiválasztásánál bemutatott ablak **Calculated** részében a **< create >** opciót választva megjelenik a következő ablak:

```
Name :
Description :
Expression :
_____________________

Template
Picture functions
```

Name : A számított mező nevét kell megadnunk. Pl: *brutto*

Description : A mezőhőz emlékeztetőül megjegyzést kapcsolhatunk.

Expression : Ide kell írnunk a kifejezést. Pl: *netto + afa*

Ezt a kijelölést is a **Ctrl-End** leütésével zárhatjuk.

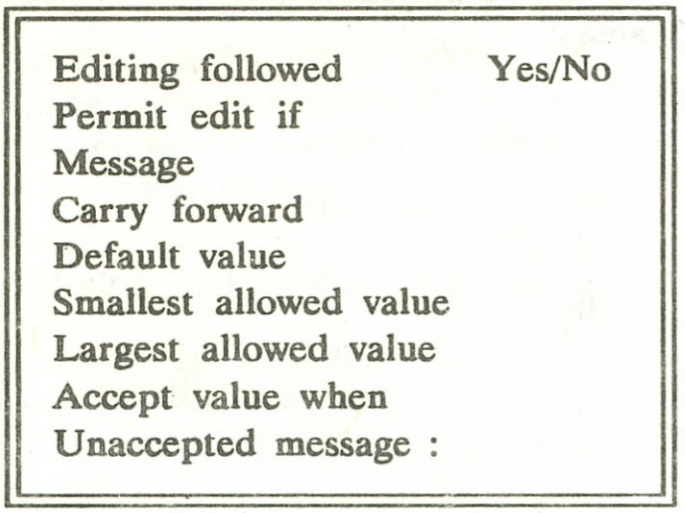
4.7.3. Előírások és formátum kijelölése az adatokhoz

4.7.3.1. Előírások, üzenetek a bemenő adatokhoz

Az eddig tárgyalt háromfajta adatból a mező és változó esetén egyéb, járulékos információkat, beviteli szabályokat is megadhatunk (feltételtől függő adatbevitel, a bevitelhez kapcsolódó üzenetsor, az adat bemeneti formátuma stb.). Ez gyakorlatilag azonos a @ ...SAY ..GET utasítások által nyújtott lehetőségekkel. (A rendszer felismeri az adatok fajtáját és a számított mezőhöz nem is kínálja fel az Edit options almenüjét.)

Amikor éppen a kívánt mezőn – vagy új mező esetén annak helyén – áll a kurzor válasszuk ki a **Fields • Add/Modify field • Edit options** almenürendszerét.

A következő ábra jelenik meg a képernyőn :



Editing followed : *No* esetén a kijelölt mező adatai adatbevitelkor nem módosíthatóak.

Permit edit if : A mező csak az itt megadott *feltétel* teljesülése esetén szerkeszthető.

Message : Az üzenetsor tartalma állítható be, amely a mezőbe történő adatbevitelkor a képernyő alján megjelenik.

Carry forward : A mező az előző rekord ugyanezen mezőjének az értékét veszi fel, és a bevitelhez így jelenik meg a képernyőn. (Továbbviszi az előző rekord értékét.) A bevitel során ez módosítható.

Default value : A kijelölt mezőnek kezdeti értéket adhatunk.

Smallest allowed value : Megadható a minimumérték, amelynél a mező értékének nagyobbnak vagy egyenlőnek kell lennie.

Largest allowed value : Megadható a maximumérték, amelynél a mező értékének kisebbnek vagy egyenlőnek kell lennie.

Accept value when : Ha a megadott logikai kifejezés *hamis* értéket eredményez az adatbevitelt a mezőnél meg kell ismételni. A kifejezésben – amelyben függvények is szerepelhetnek – különböző ellenőrzésekkel lehet eldönteni, hogy az adat hibás vagy elfogadható.

Unaccepted message : Ha a bevitt adat hibás (pl: az előző logikai kifejezés nem teljesül, vagy a min. max. értékeken az adat kívülesik, stb) kiírja az itt megadott üzenet tartalmát.

A rendszer intelligens, és az adattípusoktól függően csak az értelmes opciókat engedi kiválasztani.
Lásd még a függelékben Az adatok bevitele, megjelenítése fejezetek ?/?? és @ .. GET parancsait.

4.7.3.2. Adatok formázása

Beállíthatjuk az adat megjelenítésének jellemzőit is. Például megadható, hogy valamennyi alfanumerikus adat csak nagybetűvel jelenjék meg, stb. Kiválasztva a **Fields • Add/Modify field • Template** almenüjét megadhatunk a bevitelhez és a megjelenítéshez a megfelelő *mintákat*. A mintában a következő karaktereket szerepelhetnek: **9, #, A, N, Y, L, X, ! .**

A kijelölt adat megjelenítése befolyásolható a **Fields • Add/Modify field • Picture functions** almenüből a megfelelő megjelenítési funkciók kiválasztásával.

Ezekről részletesen a függelék ad tájékoztatót *Adatmegjelenítés, adatbevitel* alfejezet **?/?? és @ ..** utasításoknál.

A rendszer intelligens, és az adattípusoktól függően csak az értlemes opciókat engedi kiválasztani.

4.7.3.3. Az adatok jellemzőinek és elhelyezésének módosítása

Ha útközben valami eszünkbe jutott és mégis változtatni akarunk egy adat jellemzőin, válasszuk **Fields • Modify field** opcióját. Megjelenik a képernyőn az aktuális adatfájl összes mezője, a változók és számított mezők, amelyekből a kurzor segítségével választhatjuk ki a módosítani kívánt mezőt. A módosítás menete teljesen megegyezik az új adatoknál leírt eljárásokkal.

Áthelyezhetjük a mezőket, objektumokat a képernyőn, ha előbb a megfelelő területet az **F6**, a **navigálóbillentyűk** és az **Enter** segítségével kijelöltük, majd a képernyő kívánt pontjára pozícionálva az **F7** és **Enter** billentyűkkel a objektum áthelyezése megtörténik.

4.7.4. Szövegek a képernyőn

A formátumban természetesen bármilyen szöveg is elhelyezhető, például megjegyzések, fejlécek, mező megnevezések, stb.

4.7.4.1. Szövegbevitel billentyűzetről

A szövegek a szövegszerkesztőkhöz hasonló módon vihetők a billentyűzetről a képernyőre. Karakterek és sorok beszúrásához az "Insert" üzemmódot kell beállítani az **Ins** gomb megnyomásával. Az Enter billentyűt lenyomva üres sort szúrhatunk be a szövegbe, karakterek és sorok törlése pedig a **Backspace** vagy a **Del** billentyű leütésével történhet, stb.

Sorok törlése és beszúrása megvalósítható a menürendszer segítségével is. Ehhez válasszuk ki a **Words • Remove line** vagy **Add line** opcióját.

4.7.4.2. Szövegbevitel fájlból, -kivitel fájlba

Kívánságra a képernyő kijelölt részének lévő szöveg a **Words • Write text..** opciójával szabványos szövegfájlba kiírható. Ugyanígy a kijelölt képernyőrészbe szöveget is olvashatunk be külső szövegfájlból a **Words • Read ..** opció megválasztásával.

4.7.4.3. Szövegigazítás

A képernyőn megtalálható szövegek egyszerű eljárással igazíthatók jobbra, balra, stb. Ennek érdekében válasszuk ki a **Words • Position** opcióját. Egy ablak jelenik meg a következő tartalommal :

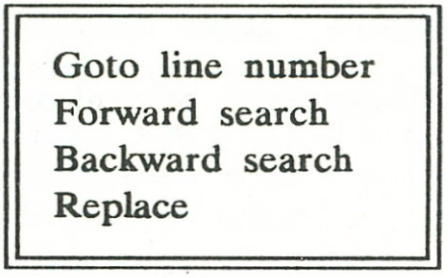
Left : A kurzor sorában található szöveget a balra igazítja.

Center : A szöveget középre igazítja.

Right : Jobbra igazít.

4.7.4.4. Mozgás, keresés, csere a szövegekben

Ha bonyolultabb képernyőn szövegrészeket akarunk megkeresni, vagy másra kicserélni, válasszuk a **Go to** menüpontot, amely a gyorsabb navigálásra is lehetőséget ad.



Goto line number : A kurzor a megadott számu sor elejére ugrik.

Forward search : Kijelölhetünk karaktersorozatot, amelyet a képernyőn elhelyezkedő szövegben előrefelé megkeres, és a kurzort rápozíçionálja.

Backward search : Az előző keresés visszafelé.

Replace : A kijelölt sztringet megkeresi a szövegben, és kicseréli a másodikként megadott sztringre.

4.7.4.5. Blokkok másolása, mozgatása

Időnként szükségessé válik, hogy a képernyőn szövegeket mezőket áthelyezzünk vagy átmásoljunk egyik helytől a másikra. Erre szolgálnak a más szövegszerkesztőknél is jólismert blokkműveletek.
Blokknak nevezzük az általunk megjelölt szövegrészeket illetve képernyőrészeket.

A **blokk-kijelölés** menete az alábbi:

- Mozgassuk a kurzort a kijelölni kívánt szöveg- vagy képernyőrész bal felső sarkába.
- Üssük le az **F6** billentyűt.
- Vigyük a kurzort a kijelölni kívánt rész jobb alsó sarkába és üssünk **Entert**. Ezzel a rész színe is inverzre változik.

A **blokk áthelyezése:**

- Pozicionáljuk a kurzort a kívánt új hely bal felső sarkába.
- Használjuk az **F7** billentyűt.
- Nyomjunk **Entert**, amire a blokk az eredeti helyéről eltűnik és az új helyen megjelenik.
- Nyomjuk meg ismét az **F6** billentyűt és a kijelölés az inverz színnel együtt megszűnik.

Blokk másolása:

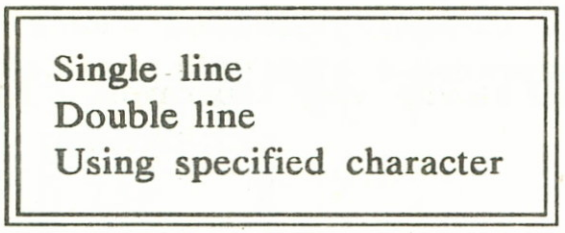
Megegyezik az áthelyezéssel, de az F7 helyett az **F8** billenytűt kell használni, és a kijelölt rész az eredeti helyén is megmarad.

4.7.5. Grafika és színek a képernyőn

4.7.5.1. Keret- és vonalrajzolás

Az adatok éles elválasztása és a tetszetős külső érdekében célszerű az egyes képernyőrészek köré kereteket, vagy közéjük vonalakat rajzolni.

Keret a **Layout** • **Box**, vonal pedig a **Layout** • **Line** opciójának megválasztásával érhető el. Mindekettőre a következő almenü jelenik meg a képernyőn :



Single : Egy vonalból álló keretet vagy vonalat rajzol.

Double : Két vonalból álló keretet vagy vonalat rajzol.

Using specified character : A képernyőn megjelenik az összes ASCII karakter, amelyből a kiválaszthatjuk az elképzelésünknek megfelelő keret- vagy vonalkaraktereket.

A rajzolás menete a következő :

- Mozgassuk a kurzort a felrajzolandó keret bal felső sarkába illetve a vonal kívánt kezdetéhez, majd üssünk **Entert.**
- Ezután mozgassuk a leendő keret jobb alsó sarkába illetve a vonal kívánt végéhez a kurzort és üssünk **Entert.**

4.7.5.2. A színek beállítása.

A képernyő jobb áttekinthetősége érdekében valamennyi képernyőelemet színezhetjük.

Álljunk a kurzorral a kívánt képernyőelemre, illetve jelöljük azt ki (F6). Utána válasszuk ki **Words** • **Display** opcióját. A megjelenő ablak két részből áll. Az egyik ablak segítségével magát a karaktert (*Foreground*)a másikkal pedig a hátteret (*Background*) színezhetjük. A színkiválasztás a navigálóbillentyűkkel lehetséges. A kiválasztott színpár a **Ctrl-End** megnyomásával véglegelhető.

4.7.6. A formátum elmentése, adatok megjelenítése

Miután valamennyi szövegbevitelt és kijelölést befejeztünk válasszuk ki a menürendszer **Exit • Save changes and exit** opcióját. Megtehetjük ezt menetközben kilépés nélkül is, ha a **Layout • Save ..** opcióját választjuk. Új formátum esetén meg kell adnunk a formátumfájl nevét is. **Enter** lenyomására a formátum definicióinak tárolása illetve lefordítása történik meg az alábbi fájlokba:

.scr - A formátumdefiníciókat tartalmazza.

.fmt - A formátumot létrehozó programkódot tartalmazza, amely szerkeszthető is.

.fmo - A futtatható programot, azaz a tárgykódra fordított *.fmt* fájlt tartalmazza.

Az így keletkező formátumfájl adatbevitelkor vagy megjelenítéskor használható dBASE IV parancsokat tartalmaz, melyek a tervezett képernyőforma létrehozására alkalmasak.

A képernyőformátum használata interpreterben és programjainkban egyaránt lehetséges. Ehhez a **SET FORMAT TO** utasítást kell kiadnunk.

Az adatok **megjelenítéséhez** lépjünk vissza a Control Centerbe. Ezután válasszuk ki a Forms panelből a most felépített formátumfájl nevet. Ezúttal a **Display data** opciót válasszuk, vagy nyomjunk egyszerűen **F2**-t és az adatok a tervezett formátumban jelennek meg. Ez a lehetőség a tervezés során is rendelkezésünkre áll, azaz az **F2/Shift-F2** a menetközbeni megjelenítést és visszatérést teszi lehetővé.

4.8. Listatervezés, -nyomtatás - Reports panel

A legtöbb adatfeldolgozás eredményeként valamiféle nyomtatón megjelenített lista is születik. Az adatfeldolgozás eredményeként születő listákat az angol terminológia **jelentésnek** nevezi. A listák — jelentések — jelentős része az adatfájlokat dolgozza fel, azokból jelenít meg eredeti és származtatott adatokat. Az adatok a listán általában valamilyen szempontból rendezettek, hiszen így alkalmasak keresésre, "emberi fogyasztásra". Legtöbb listában bizonyos rendezett csoportosításban összesítő, átlagoló, stb adatok is szerepelnek. Gondoljunk csak a cikkenkénti rendelésállomány listájára, ahol a tételek mellett a cikkenkénti naturális és értékbeni összegekre is kíváncsiak lehetünk, sőt nem közömbös az sem, hogy összesen hány forintnyi (netán dollárnyi) árut rendeltek cégünktől. Bonyolíthatjuk még a kívánságokat azzal is, hogy a listán termelőegységenként is látni akarjuk a rendelések értékét. Nyilvánvaló, hogy ilyenkor a listán az adatoknak termelőegységenként, azon belül cikkenkénti sorrendben kell szerepelnie. Az is természetes követelménynek tűnik, hogy a listának jól olvashatónak, esztétikusnak kell lennie.

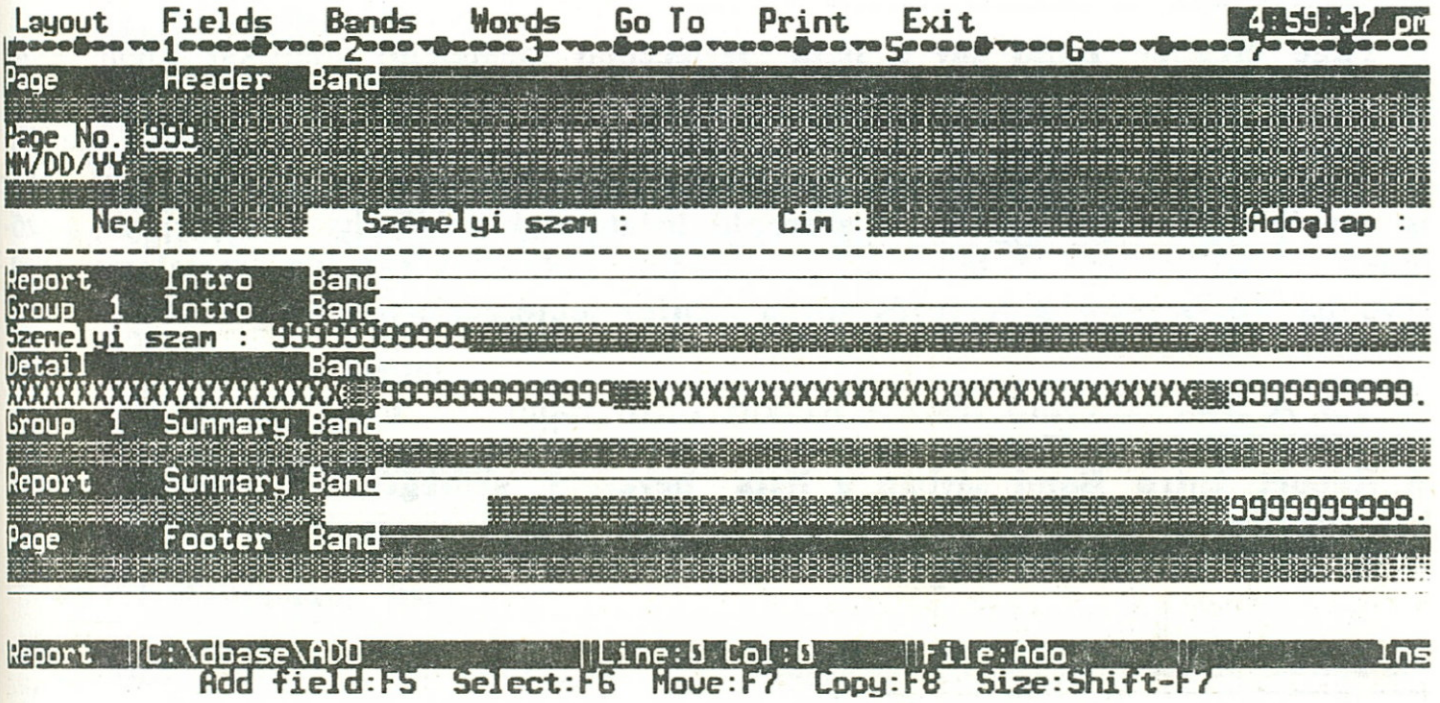
A dBASE IV-ben az ilyen feladatok megoldására alkalmas a "jelentés-", vagy ahogy mi nevezzük: **listatervező** modul.

Hasonlóan a többi panelhoz, itt is a már létező listaformátum kijelölésével és a **Modify Layout** opció kiválasztásával juthatunk a tervezés munkafelületére, vagy új fájl esetén ugyanerre a **<create>** opció mellett kell. A **Print report** az adatok kiválasztott listaformátum szerinti kinyomtatását (ill.nyomtatási formátumban a képernyőn való megjelenítését) indítja.

Mielőtt azonban bármelyik opciót választanánk, feltétlenül ki kell jelölnünk azt az adat- vagy nézetfájlt, amelyik adatait ki akarjuk nyomtatni. Ezt a módosíthatjuk a **Layout • Use different database file or view** opció kiválasztása után.

4.8.1. A listatervezés munkafelülete

A listatervezés munkafelülete némiképp eltér az eddig megismertektől. Legfőbb kükönbség az, hogy a munkafelület több "sáv"-ra (**Band**) oszlik, követve a legáltalánosabb listaszerkezetet. Így különül el a fejléc, a cím, az összegek sávja, stb. Minden sávba kijelölhetjük azt, hogy ott milyen adatok (mezők, számított mezők, konstansok, stb), szövegek, grafikai elemek jelenjenek. Természetesen a sávok használata nem kötelező.



4.6. ábra

A sávok címsorai (.....**Band**----------...) csak vizuális tájékoztatásunkra szolgálnak, de kinyomtatásra nem kerülnek.

Szokás szerint a menürendszerrel foglalkozunk először, majd sorravesszük a listasávokat.

4.8.1.1. A menürendszer

A menüsor elemeit az **F10** billentyű segítségével aktíválhatjuk.

Layout : Gyors lista tervezés különböző listatípusok szerint, grafikai elemek meghatározása, az aktuális adatfájl kijelölése és a jelentés, listaformátum elmentése.

Fields : A listán szerepeltetni kívánt mezők kijelölése, módosítása.

Bands : A lista logikailag elkülönülő részeinek, sávjainak kezelése

Words : Szövegigazítás, színek, betűstílusok, szövegfájl betöltése a formátumba, stb.

Go to : Szövegkeresés, csere, stb.

Print : A lista kinyomtatása és a nyomtatási jellemzők beállítása.

Exit : Visszatérés a Control Centerbe.

4.8.1.2. A listaoldal fejléce - Page Header Band

A **Page Header Band** sáv a **lista fejlécében** megjelenő adatokat fogja össze. Az adatok konstansok lehetnek tetszés szerintiek, de rendszerint a lista neve, dátum, oldalszám, és az oszlopokat azonosító szövegek (adatok fejléce) szokott megjelenni, de pénzügyi jellegű listáknál itt jelenhet meg az ún "*áthozat*" rovat is, amely az korábbi oldalakon szereplő listatételek valamelyik numerikus mezőjének eddigi összege. A fejléc a lista **minden oldalának tetején** megjelenik. (Feltéve ha ezt a sávot a tervezés során (**Enter** leütésével) megnyitjuk.)

4.8.1.3. A lista bevezető része - Report Intro Band

A **Report Intro Band** sávban a **lista bevezető szövegét**, ill. bevezető adatait kell elhelyezni. Ezek az adatok csak az **első oldalon** jelennek meg. Az adatokra itt sincs különösebb megkötés.

4.8.1.4. A listatétel - Detail Band

A **Detail Band** a **lista tétele**, azaz a lista leglényegesebb eleme. Ide kerülnek a tételes adatok. **Minden oldalon több** listatétel szerepel, ha az oldalra több is ráfér, vagy nem rendelkezünk másképp.

4.8.1.5. A listazáró, -összegző rész - Summary Band

A **Summary Band** a **lista végösszegének** sávja. Ide kerülhetnek az összes listatétel numerikus mezőinek végösszegei, átlagai, a tételek darabszámai, átlagtól való eltérés mutatószámai, stb. Ezek az adatok a **lista végén** jelennek meg.

4.8.1.6. A lábléc - Page Footer Band

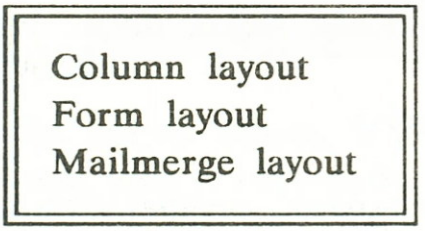
A **Page Footer Band** sáv a **lista láblécében** megjelenő adatokat fogja össze. Néha ide kerül az oldalszám, de kerülhet ide például pénzügyi listáknál az ún "*átvitel*", amely az adott oldalon lévő tételek valamelyik numerikus mezőjének és az oldal fejlécében lévő "*áthozat*"-nek az összege. A lábléc a lista **minden oldalának alján** jelenik meg. (Feltéve ha ezt a sávot a tervezés során **Enter** leütésével megnyitjuk.)

4.8.1.7. A csoportsávok - Group .. Band

A rendezett (indexelt) adatok csoportosíthatók a listán, és ezeknek a csoportoknak a képzett adatai (a csoport címadatai, csoportösszegek, stb) az erre rendelt csoportsávokban jeleníthetők meg. A csoportsávok csak a csoportok kijelölése után jelennek meg a munkafelületen. Részletesen lásd az adatok csoportosítását tárgyaló bekezdést.

4.8.2. Listatípusok

A listatervezővel sok, különféle listaformátum létrehozható, mégis három típust érdemes külön megemlíteni. A **Layout • Quick Layouts** menüpont ugyan a listasávok mezőkkel, egyéb adatokkal törénő gyors feltöltésére szolgál, ezelőtt azonban választásra kínálja a három leggyakrabban használt listatípust:



4.8.2.1. Táblázatos lista

Column layout : Táblázatos, oszlopos megjelenítési forma, ahol a rekordok egy-egy sorba, a mezők egymás alá egy-egy oszlopba kerülnek.

Példa :

Név	Ország	Beosztás
Kiss János	Magyarország	Csapatvezető
Horvát Dénes	Magyarország	Bíró.

4.8.2.2. Soros lista

Form layout : A mezők egymás alá kerülnek a papíron. Akkor használatos ha túl sok adatunk van, amelyek nem férnek el egy sorban.

Példa :

Név : Kiss János.
Ország : Magyarország.
Beosztás : Bíró.

4.8.2.3. Formalevél

Mailmerge layout : Több címzettnek elküldendő levéljellegű lista, amely minden tétele egy-egy oldal és egy tétel az azonos levélszöveg mellett a címzett nevét és a címzettnek szóló (dedikált) adatokat tartalmazza.

Példa :

Értesítés fizetésemelésről

Kiss János csapatvezető részére
Magyarország

Értesítem, hogy fizetését a vállalatvezetés az inflációra való tekintettel **26**, jó munkáját értékelve pedig további **6** százalékkal emelte. Új alapbére: **12345** Ft.

Utópiafalva, 1989. szeptember 26.

Verne Gyula igazgató sk.

4.8.3. Az adat megjelenési helyének meghatározása

Mielőtt a listában megjelenítendő adatokat kiválasztanánk, meghatároznánk, el kell döntenünk, hogy azok a lista melyik részében, sávjában és azon belül melyik ponton jelenjenek meg.

4.8.3.1. Listasáv nyitása, zárása

Ha a kurzort a kívánt listasáv címsorára (Page Header Band, Detail Band, stb) visszük, és **Entert** ütünk, akkor a listasáv **megnyílik**, és a kurzor a sávon belülre kerül.
Minden listasávot egyszerre megnyithatunk, ha a **Bands • Open all bands** opciót választjuk.

Ha a listasáv nyitva volt, akkor **Enterre zárul**, és az oda kijelölt adatok nyomtatáskor nem jelennek meg. A sáv zárása nem jelenti az adatkijelölés elvesztését. A kijelölt adatokat ismételt megnyitáskor a helyükön találjuk.

4.8.3.2. A sáv bővítése, szűkítése

Ha a listasávon belül vagyunk, akkor tetszés szerinti helyezhetjük el a szövegeket, adatokat, grafikus elemeket. Ha erre a hely szűknek bizonyul, a **Words • Add line** opcióval *bővít*hetjük egy sorral a sávot. Megtehetjük ugyanezt a sor végén az **Enter** leütésével is, ha éppen a beszúró (*Ins*) mód érvényes.

Sorok *eltávolítása* a **Words • Add line** kiválasztásával történhet. Ilyenkor az aktuális sor adatokkal együtt eltűnik (a kijelölések is törlődnek), és a sáv ennyivel szűkül.

4.8.4. Az adatok kiválasztása

Ha a megfelelő listasávot kiválasztottuk, kijelölhetjük az ott megjelenítendő adatokat. Ha gyors eredményt akarunk, akkor válasszuk a korábban is említett **Layout • Quick Layouts** opciót, és a listatípus eldöntése után a rendszer a listasávokat automatikusan feltölti az aktuális adatfájl vagy nézetfájl mezőivel, illetve a listazáró összegsávot a numerikus mezők összegző mezőivel. Ezeket aztán tetszés szerint megváltoztathatjuk.

Ha egyenként jelölünk ki az adatokat, mielőtt ehhez hozzákezdenénk, álljunk arra a pontra, ahol az adat kezdődjön. Tegyünk így annak ellenére is, hogy az adat sávon belüli helyét később is módosíthatjuk. (Blokkműveletek: F6 kijelölés, F7 áthelyezés)

Ha az adat helyén állunk, a **Fields • Add field** opcióját kiválasztva az alábbi ablak jelenik meg a képernyőn

fájlnév	CALCULATED	PREDEFINED	SUMMARY
mező 1	<create >	Date	Average
mező 2		Time	Count
mező 3		Recno	Max
mező 4		Pageno	Min
....			Sum
mező n			Std
			Var
			SUM(*mező i*)
			SUM(*mező j*)

4.8.4.1. Mezők

Az aktuális adat- vagy nézetfájl mezőit a *fájlnév* alatti oszlopból választhatjuk ki a kurzorral történő rámutatással.

A megfelelő mező kiválasztása után az alábbi ablak jelenik meg:

Field name:	*mezőnév*
Type:	Character
Length:	10
Decimals:	0
Template	{XXXXXXXXXX}
Picture functions	{T}
Suppress repeated values	NO

Field name : A kiválasztott mező neve.

Type : A kiválasztott mező típusa.

Length : Mezőhossz.

Decimals : Tizedesjegyek száma.

Ezeket (a vonal felettieket) itt most nem módosíthatjuk.

Template : A mező mintája, amely a típusnak megfelelő. Ez módosítható. Az **Enter** megnyomására felkínálja egy ablakban a választható mintakaraktereket. Lásd a függelékben a @ .. utasítás PICTURE cikkely mintáit.

Picture functions : A megjelenítést vezérlő függvények módosíthatóak. Az **Enter** megnyomására felkínálja egy ablakban a választható függvényeket. Lásd a függelékben a @ .. utasítás FUNCTION cikkelyt.

Suppress repeated values : *NO* esetén elnyomja az ismétlődő értékeket, azaz, ha a köbetkező megjelenített sorban is ugyanaz a mező értéke, azt nem jeleníti meg. Ezzel a táblázat jobban áttekinthetővé válik.

4.8.4.2. Számított adatok

Ha a mezők nyers adatai nem felelnek meg, akkor a mezőkből alkotott valamilyen kifejezést is megjeleníthetünk. A kifejezésnek nevet is kell adnunk és ezzel lényegében egy memóriaváltozó keletkezik.

Ilyenkor a **CALCULATED** oszlop **<create>** opcióját kell választanunk, amelyre a már (a mezőknél) megismert ablakhoz hasonló jelentkezik.

A mezőktől eltérően itt az ablak felső részét is ki kell töltenünk:

Name : A kifejezés neve, a változónév

Description : Rövid emlékeztető megjegyzést fűzhetünk a kifejezéshez. (Ez a generált programban kommentárként szerepel.)

Expression : A kifejezést kell meghatároznunk, amely legális dBASE kifejezés lehet létező mező és változónevekkel, operátorokkal, függvényekkel.

Hidden : Mód van arra is, hogy a számított adat megjelenítését megakadályozzuk, mert csak egy közbülső számítást, átalakítást tartalmaz, amelynek eredményét csak a nyomtatás (megjelenítés) későbbi fázisában használunk fel. Ilyenkor válasszuk a *YES* opciót. (Az **Enter** leütése vált.)

A többi opció a mezőknél megismertek szerint működik.

4.8.4.3. Kötött rendszeradatok

Lehetőségünk van előre meghatározott — **Predefined** — adatokat is választani:

Date : A rendszerdátum

Time : Rendszeridő

Recno : Az aktuális, éppen a sor nyomtatásakor érvényes rekordsorszám.

Pageno : Az éppen érvényes lapszám.

Ilyenkor a kiválasztás után megjelenő ablak értelemszerűen nem tartalmaz néhány adatjellemzőt, de a megjelenés képét (Template, Picture), és az ismétléselnyomást módosíthatjuk.

4.8.4.4. Összegző adatok

Gyakran szükséges lehet, hogy a nyomtatás végén, de közben is különböző adatok valamiféle összegzését, summázó adatát megjelenítsük. Különösen így van ez az egy csoportba tartozó adatoknál, ahol a csoportok összegző adataira szinte mindig kíváncsiak vagyunk. (A csoportok képzéséről, kezeléséről később lesz szó.) A summázó, összegző kifejezést itt tágabban értelmezzük: takarhatja a matematikai szumma, az átlag, a minimum és maximum, stb műveleteket is. Ezt kínálja a **SUMMARY** ablak is, ahol a felsorolt függvények bármelyikét választhatjuk. Ezek magyarázatát a függelékben részletesen megtalálhatjuk a függvények címszó alatt.

A kiválasztás után megjelenő ablakban az eddigiekhez képest három ismeretlen címszót találunk:

Operation : Itt a kiválasztott függvény neve látható.

Field to summarize on : Ide kell kiválasztanunk azt a mezőt, amelyre a summázó műveletet el akarjuk végezni. A megnyíló újabb ablak felkínálja a választható mezőket.

Reset every : Meg kell határoznunk azt is, hogy az összegző művelet mikor fejeződjön be és egyben kezdődjön újra a lista megjelenítésének folyamán.
A **végösszegzéseknél** ez egyszerű, hiszen a művelet a lista elején kezdődik, és a végén befejeződik. Ezt kínálja fel a megnyíló újabb ablakban a **< REPORT >** opció.
Van olyan is, amikor **laponként** akarunk valamilyen összegzést látni. Gondoljunk vissza a korábban említett *átvitel*, *áthozat* adatokra. Ilyen hatást gyakorol a **< PAGE >** opció.
Leggyakoribb azonban az, amikor valamilyen mező értékének változásakor kell az összegző adatot kiírni, és a műveletet újrakezdeni. Ilyenkor a megnyíló ablakból valamelyik **mező** nevét kell kiválasztanunk. (Ez a módszer a csoportosított adatokra jellemző, amelyről később lesz szó.)

Érdekességként megemlítjük, hogyha a summázó műveleteket a listatétel sávjában jelöljük ki, akkor halmozott, azaz a művelet kezdetétől az éppen aktuális listatételig elkészült összegző információt kaphatjuk eredményül. Így például a lista végéig összeadó művelet a lista tételsorában annak elejétől a kinyomtatott tételsorig készült összeget fogja mutatni. Jól használható ez akkor, ha egy cég összbevételeinek napról napra történő növekedését akarjuk bemutatni.

4.8.5. Csoportok képzése

A korábbiakban sokszor szó esett az adatok csoportosításáról. Nézzük ezt részletesebben.

Gyakran előfordul, hogy listáinkban az egyszerű felsorolás és végösszegek (átlagok, stb) mellett az adatok egy csoportjának összegző információit is látni kívánjuk, sőt megköveteljük, hogy a jobb áttekinthetőség érdekében egymás közvetlen közelében lássuk az egy csoportba tartozó adatokat. Ilyen lehet például, ha egy vállalat termelési adatait listázzuk, de látni akarjuk ezen belül az egyes termelőegységek adatait is. A példák – úgy gondoljuk – annyira közismertek, hogy nem érdemes tovább sorolni őket.

4.8.5.1. A csoportosítás előkészítése rendezéssel, indexeléssel

A csoportosító összegző műveleteknél elegendő, ha az egy csoportba tartozó adatok szigorúan követik egymást. Ezt többféle módon is elérhetjük, de a kézenfekvő leginkább az, ha indexeljük az adatfájlt az adott szempont szerint. Megtehetjük ezt a fizikai rendezéssel is. Mindezt a Data panel tervezési felületéről érhetjük el, de az interpreterből is használhatjuk a SET INDEX, SET ORDER,

SORT utasításokat. A jelentés tervezéséhez ez nem szükséges, de elengedhetetlen a nyomtatáshoz (képernyőn történö listamegjelenítéshez).

4.8.5.2. A csoport létrehozása

A csoportok létrehozása annyit jelent, hogy listarészt, sávot kell kialakítanunk a csoport bevezető és összegzőadatai számára.

Ezt a **Bands • Add a group band** opciójával érhetjük el.

A megjelenő ablak a csoportosító szempont meghatározását várja:

Field value	{}
Expression value	{}
Record count	{}

Field value : Ha ezt az opciót választjuk, akkor a megadott mező lesz a csoportosítás alapja, azaz egy csoportba azok a rekordok tartoznak, amelyeknél a mező értéke azonos.

Expression value : Ebben az esetben kifejezést határozhatunk meg csoportosító szempontként, azaz egy csoportba azok a rekordok tartoznak, amelyeknél a megadott kifejezés értéke azonos.

Record count : Ezt válasszuk, ha a rekordokat darabszám szerint akarjuk kötegelni. A megadott darabszám szerinti rekordok tartoznak egy csoportba. Így például, ha 10-et adunk meg, akkor egymást követő tíz rekordonként keletkezik egy csoport, ha 14-et, akkor 14 rekordonként, stb.

A kjelölés után megjelenik a két csoport-sáv a **Group n Header Band** és a **Group n Summary Band**. "n"-nel a csoport sorszámát jelöltük tekintettel arra, hogy ezekből több is lehet.
Ha ez az első csoportsáv, amit kijelöltünk, akkor az a listatételt fogja "körülvenni" és az n értéke 1 lesz.
Ha volt már kijelölt csoport, akkor ez az előzőleg meghatározott csoport sávjait veszi "héjként" körül. Ilyenkor az n (a csoportszám) értéke 1-gyel magasabb lesz.

Több **csoport** kialakításánál ügyelnünk kell arra, hogy a legalacsonyabbrendű szempont szerinti csoport legyen a legelső, azaz ez kerüljön "legbelülre", az eggyel magasabbrendű legyen a második, stb.
Ilyenkor figyelni kell arra is, hogy csak a rendezési indexelési szempont alapján képezhetünk csoportokat. Így korábbi példánknál a vállalat termelési adatait nemcsak termelőegységenként, hanem azon belül team-enként (régebbi kifejezéssel élve: brigádonként) akarjuk látni, akkor az indexkifejezésben termelőegység kódjá-

hoz hozzzá kell fűzni a team-kódot is, az első csoport pedig csak a team lehet, és csak a második a termelőegység.

A csoport jellemzőit a **Bands • Modify group** opció kiválasztásával módosíthatjuk, amely megegyezik az új csoport létrehozásánál szükséges teendőkkel.

A csoportokat a **Bands • Remove group** opcióval törölhetjük, ha azok fölöslegessé válnak.

Példa:

Lássunk egy egyszerű csoportosított listát csoportösszegekkel.

Név	Ország	Beosztás	Összeg
Kiss János	HUN	Bíró	1000
Nagy Károly	HUN	Bíró	1000
Bírók összesen:			2000
Varga Tamás	HUN	Edző	1500
Pintér Kálmán	HUN	Edző	1500
Edzők összesen:			3000
Brian John	USA	Csapatvezető	1000
. . .			
. . .			
Mindösszesen:			4850000

4.8.5.3. A csoport fejléce - Group Header Band

A megnyíló csoport fejléc sáv a csoport bevezető adatait tartalmazhatja, amelyekre különösebb megkötés nincs. Ide célszerű elhelyezni a termelési adatok példánál maradva a team kódját, nevét illetve az efeletti csoportsávba a termelőegység kódját, nevét, stb.

A sávot nem kötelező használni.

4.8.5.4. A csoport összegző sáv - Group Summary Band

Ez is opcionális sáv. Ide általában a csoport summázó információi kerülnek, néha a csoport azonosító adatával együtt.

A summázó adatok kijelölésének menetét már ismerjük. A helyes eredményhez arra kell ügyelnünk, hogy az **Add field • CALCULATED • ... • Reset every** opciónál a csoportra jellemző mezőt adjuk meg.

4.8.6. Szövegszerkesztés, grafika

A képernyőformátum tervezésénél ismertetett módszerek itt is ugyanúgy alkalmazhatók. **A Words** és **Go To** menüpontok azonosan működnek, segítségükkel a sorok beszúrása, eltávolítása, szövegek cseréje, szövegigazítás, betűtípusok kiválasztása, stb megoldható. Az ismert módon működik a blokk kijelölés, áthelyezés, másolás is. (F6, F7, F8)
(Lásd : Képernyőformátumok - Szövegek a képernyőn)

A **Layout • Box** és **Layout • Line** segítségével ugyanolyan módon rajzolhatunk keretet, vonalat.(Lásd : Képernyőformátumok - Grafika és színek a képernyőn)

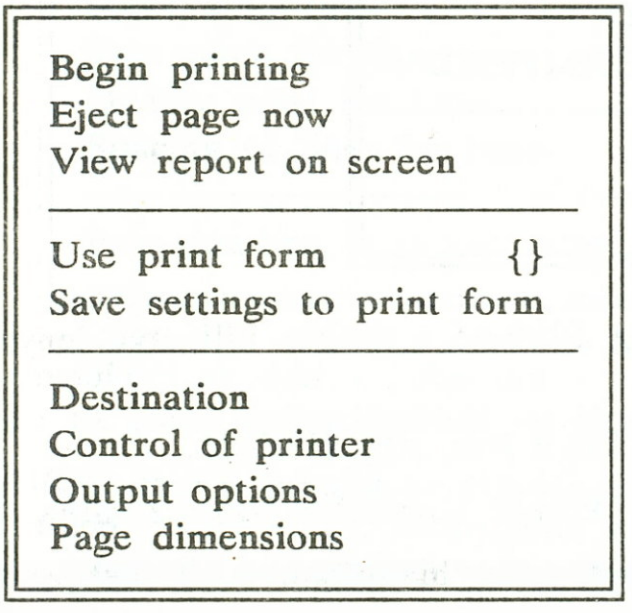
Grafikarajzolást, blokkműveleteket csak a sávokon belül végezhetünk.

4.8.7. Adatok megjelenítése

A listaformátum tervezés elsőrendű célja az adatok nyomtatásban történő megjelenítése. Ez esetben a *Shift-F2* az eredeti adat- vagy nézetfájlt jeleníti meg a Data panelnél megismert BROWSE, EDIT formákban.

A nyomtatás és megjelenítést alapvetően a **Print** menüpont kínálta lehetőségek vezérlik, de jelentős szerepe van a sávok megjelenítését szabályozó (*Bands* menüpont) opcióknak is.

A Print rolómenüje az alábbiak szerint fest:



Az első három opció nem a jellemzők beállítására való, erről a *Nyomtatás, megjelenítés* bekezdésben esik szó.

Use print form : Használhatunk egy korábban beállított kombinációt a nyomtatási jellemzőkként.

Save settings to print form : A most beállított nyomtatási paramétereket elmenthetjük későbbi felhasználás céljából.

Destination : A megnyíló ablakban meghatározhatjuk, hogy hova történjen a kiírás (*Write to*). Ha az DOS fájlba történik akkor megadhatjuk annak nevét (*Name of DOS file*), a nyomtató fajtáját (*Printer model*) és megtilthatjuk, hogy nyomtatás közben a képernyőn is megjelenjenek az adatok (*Echo to screen* NO).

Output options : A bejelentkező ablakban meghatározhatjuk, hogy a lista hányadik oldalától (*Begin of page*) hányadik oldaláig (*End after page*) kerüljön kinyomtatásra. Megadható az első oldalszám (*First page number*) és a példányszám (*Number of copies*).

Page dimensions : A megnyíló ablakban a laphossz (*Lenght of page*), az alapmargó (*Offset from left*) és sorhúzás (Line spacing) állítható.

4.8.7.1. A nyomtató vezérlése

A **Print • Control of printer** opciójával a nyomtatót kezelhetjük:

Text pitch	DEFAULT
Quality print	DEFAULT
New page	BEFORE
Wait between pages	NO
Advance page using	FORMFEED
Starting control codes	{}
Ending control codes	{}

Text pitch : A karakterszélesség, -sűrűség állítható a szóköz billentyű lenyomásával.

Quality print : Az írásminőség (levélminőségű írás, stb) állítható.

New page : Beállítható a listát megelőző, követő lapemelés (szóköz váltja).

Wait between pages : Leporelló vagy lapbehúzó hiányában a lap befűzéséhez oldalankénti megszakítást kérhetünk.

Advance page using : A lapemelés normál, és sorokból számított módját válthatjuk.

Starting control codes : Különleges nyomtatóvezérlő karaktereket küldhetünk a nyomtatás előtt a nyomtatóra, amikor olyan lehetőségeket akarunk használni, amelyre a nyomtató képes, de a nyomtatómeghajtó (PRINTER DRIVER) nem. Ehhez a nyomtató kézikönyvét ismerni kell.

Ending control codes : Célja azonos az előző opcióéval, de a vezérlőkaraktereket a nyomtatás végén küldi. Rendszerint a nyomtató eredeti állapotának visszaállítására használatos.

A választható értékek a *szóköz* billentyű ismételt leütésével válthatóak.

4.8.7.2. Sávok megjelenítésének vezérlése

A nyomtatás (megjelenítés) egyik fontos kérdése, hogy a lista sávjai hogyan, és mikor jelenjenek meg a listán. A Bands menüpont legtöbb opciója ennek vezérlésével foglalkozik.

Add a group band	
Remove group	
Modify group	
Group intro on each page	NO
Open all bands	
Begin band on new page	NO
Word wrap band	NO
Text pitch for band	DEFAULT
Quality print for band	DEFAULT
Spacing of lines for band	DEFAULT
Page heading in report intro	YES

Az opciókból az *Add* - , *Remove* - , *Modify group* és az *Open all bands* már ismert az előző bekezdésekből, és nem a nyomtatás vezérlésével foglalkozik.

Az alábbi opciók mindig az aktuális sávra határozzák meg a megjelenítés jellemzőit, tehát arra a sávra, amelyikben a tervezés alatt éppen vagyunk. A lehetőségeket a **szóköz** billentyűvel váltogathatjuk. A *DEFAULT* az alapértelmezést jelenti.

Begin band on new page : *YES* esetén az aktuális sávot nyomtatáskor mindig új lapon kezdi.

Word wrap band : *YES* esetén az aktuális sávban azok a szavak, amelyek éppen a sor végén elválasztásra kerülnének, új sorba kerülnek, és jobbraigazítás (*Words* • *Position* • *Right*) esetén a sorkiegyenlítés is megtörténik.

Text pitch for band : Az aktuális sáv karaktereinek szélességét, -sűrűségét határozhatjuk meg.

Quality print for band : Változtathatjuk az írásminőséget (normál, szépírás, stb) az adott sávban.

Spacing of lines for band : A soremelés változtatható az alapértelemezés, 1, 2 és 3 soremelés között.

Page heading in report intro : *NO* esetén a lap fejléce nem jelenik meg a lista első oldalán.

4.8.7.3. Nyomtatás, megjelenítés

Ha beállítottuk a megfelelő paramétereket, vagy ennek hiányában megelégedtünk az alapértelmezés szerinti nyomtatási jellemzőkkel és lehetőségekkel, a **Print** • **Begin printing** opciójával a nyomtatás elindítható.

Ha a nyomtatást megelőzően új lap elejére akarunk állni, akkor ezt Begin printing előtt a **Print** • **Eject page** opcióval tehetjük.

Ha a listát a nyomtatási formátumban a képernyőn akarjuk megjeleníteni, akkor a **Print** • **View report on screen** opciót válasszuk. A képernyőn történő megjelenítésnél nem kell meglepődnünk, ha a 80 karakternél szélesebb lista egynél több sorban jelenik meg, valamint az is természetes, hogy a betűtípusok jelentős része nem látható, csak azok, amelyeket az éppen aktuális képernyő-meghajtó és grafikus kártya lehetővé tesz.

4.8.8. Listaformátum elmentése későbbi használatra

Az elkészített listaformátumot kétféleképpen is elmenthetjük későbbi használat céljából:
Választhatjuk a **Layout** • **Save this report** opcióját, de az **Exit** • **Save changes...** is ugyanezt eredményezi. A különbség mindössze annyi, hogy az utóbbival visszatérünk a Control Center-be, míg az előbbi után folytathatjuk a formátum tervezését. Az elkészített listaformátumot programjainkban és az interpreter alatt a **REPORT FORM** utasítással használhatjuk.

A listatervezés elmentése után három fájl keletkezik:

A **.frm** kiterjesztésű a lista elkészítéséhez szükséges információkat hordozza.
A **.frg** kiterjesztéssel a meghatározásainknak megfelelő forrásprogram jön létre.
A **.fro** a végrehajtható tárgykódra lefordított *.frg* forrásprogram.

4.9. Címketervezés, -nyomtatás - Labels panel

A gyakorlati életben nemegyszer előfordul az igény, hogy valamilyen dolgokon, tárgyakon azoknak jellemző és azonosító adatait fel kell tüntetnünk. Gondoljunk csak arra, hogy egy videókölcsönzőben illik felcímkézni a kazettákat, de mindenki találkozhatott már azzal a problémával is, amikor egy formalevelet sok címre kell elküldeni. Nem kis feladat ilyenkor a borítékok megcímzése.

Az ilyen jellegű problémákat sokféleképpen megoldhatjuk, akár tollal is felírhatjuk a szükséges információkat, de használhatunk nagyon bonyolult nyomdai technológiát is, olyat, amilyet a dobozos sörök dobozainak előállításánál alkalmaznak. Ezt általában a sorozatnagyság dönti el.

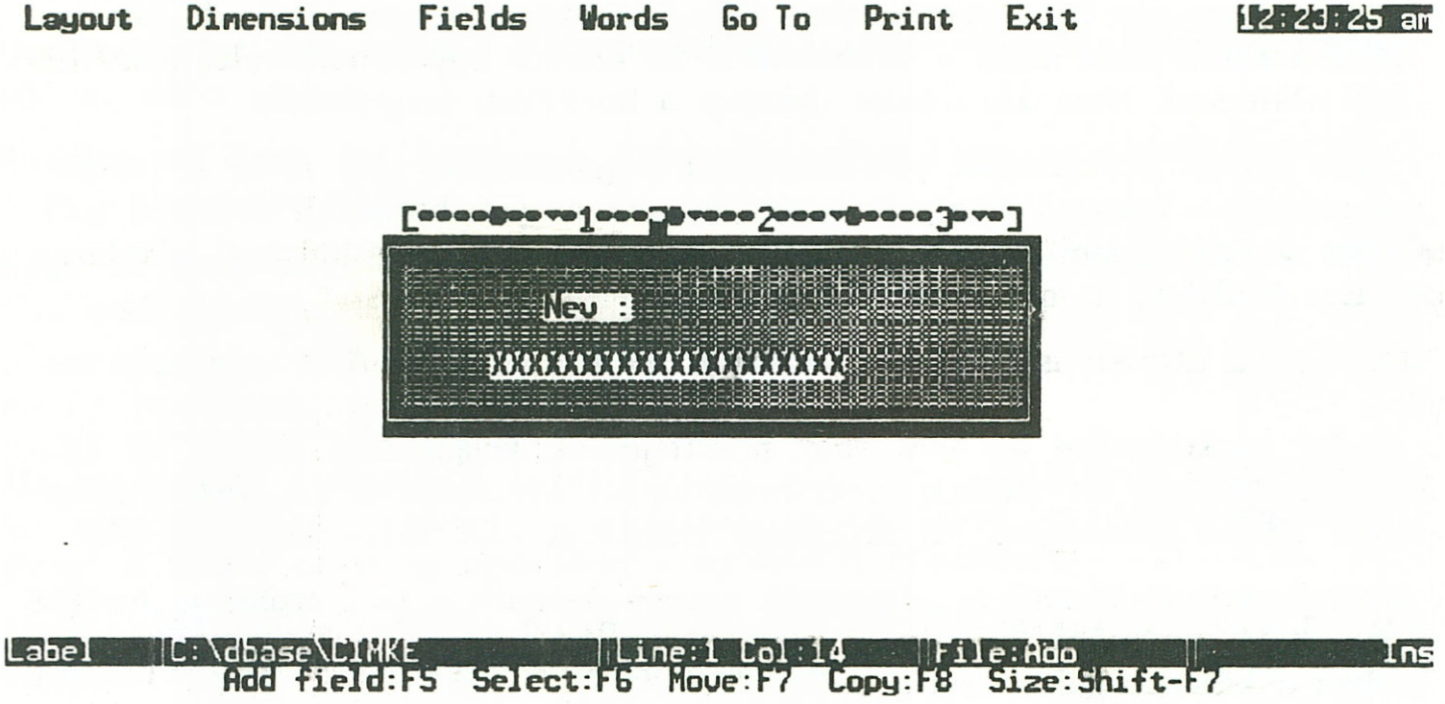
Címkézésre a dBASE is kínál technnológiát: ez a címketervező és -nyomtató modul.
A modul gyakorlatilag az adat vagy nézetfájljaink kiválasztott adatait az általunk meghatározott sorrendben címkeformátumban nyomtatja ki. (A címkék nyomtatására létezik különleges, ún öntapadós címkeleporelló, másnéven "etikett".)

A címkeformátum-készítő és -nyomtató nagyon hasonlít a listaformátum tervező és -nyomtató modulhoz, ezért erről csak szűkszavú ismertetést adunk, amely elsősorban a különbségekre szorítkozik.

A Labels panelből a tervező munkafelületre belépni a Reports panellel teljesen analóg módon lehet. (Lásd: *Modify Layout, <create>* opciók). Itt is célszerű előzőleg kijelölni a kívánt adatfájlt.

4.9.1. A címketervezés munkafelülete

A munkafelület elsősorban abban tér el a többitől, hogy annak közepén egy címke körvonalait látjuk.



4.7. ábra

A menürendszer szinte teljesen megegyezik a Reports panelével természetesen azzal a különbséggel, hogy a menüelemekben *report* helyett *label* szerepel. A Bands menüpont helyett egy ismeretlen pontot láthatunk, a **Dimensions**-t, amely a címkék méreteinek beállítására, és a címkeelhelyezésre szolgál.

4.9.2. A címkeméret és elhelyezés beállítása

Az alkalmazott címkeleporelló (egyoszlopos, kétoszlopos, stb) szerkezetéhez és címkeméreteihez a nyomtatást a **Dimensions** menüpont segítségével állíthatjuk be.

Predefined Size 15/16 x 3 1/2 by 1	
Width of label	{35}
Height of label	{5}
Indentation	{0}
Lines between labels	{1}
Spaces between label columns	{0}
Columns of labels	{1}

Predefined Size : Szabvány címkeméretek választhatóak a megnyíló ablakból. Ha azonos méretet találunk az ablak több sorában is, akkor amellett az egy sorban lévő címkék száma is feltüntetésre került. A méretek collban értendők, a szabványok nemzetköziek, de érdemes őket előbb kipróbálni a magyar címkeleporellókhoz.

Width of label : A címke szélessége adható meg karakterben. Ennek megfelelően változik a képernyő közepén lévő keret is. Legfeljebb 255 lehet.

Height of label : A címke magassága, azaz sorainak száma állítható.

Indentation : A sorban az első címke bal éle és a leporelló bal széle közötti távolságot befolyásolja. A itt megadott érték összadódik a *Print • Page dimensions* opciónál megadott alapmargó értékével. Legfeljebb 250 lehet.

Lines between labels : Beállítható az egymás alá kerülő címkék távolsága. Legfeljebb 16 sor lehet.

Spaces between label columns : Meghatározható két egymás mellett lévő címke távolsága. Ez legfeljebb 120 lehet.

Columns of labels : Az egymás mellett (egy sorban) lévő címkék számát határozza meg. Legfeljebb 15 lehet.

4.9.3. A címkére kerülő adatok kiválasztása

Ha eldöntöttük, hogy az adatfájl alapján milyen adatokat akarunk megjeleníteni, álljunk a kurzorral az adat kívánt helyére a címkén, és az **F10**-re aktivált menü **Fields • Add field** opcióját. A Reports panelnél megismert ablak jelenik meg. (Ebből csak a *Summary* oszlop hiányzik, mert címkékre összegző adatokat nem határozhatunk meg.) A **mezők, számított adatok, kötött rendszeradatok** kiválasztásával kapcsolatos teendők teljesen azonosak a listatervezésnél is szükségesekkel. Ha meggondoltuk magunkat, és javítani vagy törölni akarjuk a kiválasztott adatokat, akkor a **Fields • Modify field** illetve **Fields • Remove field** opciókat válasszuk, ha a kivánt mezőre álltunk a kurzorral.
Részletes információkat lásd ott. (Listatervezés - Adatok kiválasztása)

4.9.4. Szövegek szerkesztése

A szövegek bevitele, szerkesztése a korábbi paneloknál leírtakkal azonos. Címkékre grafikai elemeket, kereteket, vonalakat nem rajzolhatunk. A szövegszerkesztést segítő **Words** és **Go To** menük kezelését, működését lásd a képernyőtervezés alfejezetben. (Képernyőformátumok - Szövegek a képernyőn)

4.9.5. Címkék megjelenítése, nyomtatása

A nyomtatást vezérlő, és annak jellemzőit beállító **Print** menüpontot a listatervezéssel azonosan kell kezelni. Részletes információkat lásd a Listatervezés Adatmegjelenítés bekezdésében. (A *report* szó helyett a menüpontokban a *labels* szó szerepel.)

Egyedül a **Generate sample labels** opció nem szerepel a listatervezés **Print** menüjében. Ez arra szolgál, hogy próbaként nyomtathassunk egy címkét ha kétségeink vannak afelől, hogy tervezésünk megfelel a címkeleporelló beosztásának, vagy elképzelésünknek.

4.9.6. Címkeformátum elmentése későbbi használatra

Az elkészített címkeformátumot kétféleképpen is elmenthetjük későbbi használat céljából:

Választhatjuk a **Layout • Save this label design** opcióját, de az **Exit • Save changes...** is ugyanezt eredményezi. A különbség mindössze annyi, hogy az utóbbival visszatérünk a Control Center-be, míg az előbbi után folytathatjuk a formátum tervezését.

Az elkészített listaformátumot programjainkban és az interpreter alatt a LABEL FORM utasítással használhatjuk.

A címketervezés elmentések után három fájl keletkezik:

Az **.lbl** kiterjesztésű a címke elkészítéséhez szükséges információkat hordozza.

Az **.lbg** kiterjesztéssel a meghatározásainknak megfelelő forrásprogram jön létre.

Az **.lbo** a végrehajtható tárgykódra lefordított *.lbg* forrásprogram.

4.10. Az alkalmazás-generátor - Applications panel

Az eddigiekben megismerkedhettünk az adatok kezelését, ezen belül különböző formában történő megjelenítését támogató modulokkal. A számítógép céljainknak megfelelő alkalmazása során azonban igényeink valószínüleg ebben nem merülnek ki. Szükségünk lehet az így megtervezett adataink feldolgozásához számos más eljárásra is.

Az *alkalmazásgenerátorral* az eddig megismert eszközöket fűzhetjük egy felhasználói feleadatcsoportot (bérszámfejtést, raktárnyilvántartást, stb) támogató rendszerbe, ami gyakorlatilag az adatfeldolgozó **eljárások készítését** és rendszerbe kapcsolását támogató tervező rendszer. Bár a Control Center egyik (Applications) paneljéről léphetünk be ide is, könyvünknek mégis külön fejezetében foglalkozunk vele.

5. Az alkalmazás-generátor

5.1. Bevezetés

Az alkalmazás-generátor az eddigi dBASE verziókban egyáltalán nem létező, teljesen új fejlesztés. Célja már a nevéből is következik: segítségével tetszőlegesen bonyolult programrendszereket hozhatunk létre úgy, hogy eközben alig kell dBASE IV utasítást leírnunk. Azonban ez ne tévesszen meg senkit. Nem csodaszerről van szó, amely gondolatainkat kitalálva sorozatban gyártja a jobbnál-jobb rendszereket.

5.1.1. Mi az alkalmazás ?

A Control Center fejezetben számos olyan eszközzel megismerkedhettünk, amely adataink kezelését, bizonyos mértékű feldolgozását lehetővé tette. A számítógép céljainknak megfelelő alkalmazása során azonban igényeink valószínüleg ebben nem merülnek ki. Több ok miatt sem lehet így:

● Gyakran előfordul, hogy olyan felhasználó akarja használni ezeket a lehetőségeket, aki nem akarja megismerni a dBASE IV működését, lehetőségeit. Őt csak az érdekli, hogyha leül a számítógép elé, akkor annak segítségével különösebb számítógépes ismeretek nélkül végezhesse munkáját. Erre az eddig megismert módszerek, lehetőségek nem elegendőek.

● A megismert modulok széles és rugalmas eszköztárat kínálnak a leggyakrabban előforduló adatkezelési feladatokhoz, de nem nyújtanak arra lehetőséget, hogy egy ezeket munkánk egy jól körülhatárolható feladatcsoportjának támogatására egybekapcsoljuk. Az ilyen feladatcsoportok általában sok feladatrészből állnak, amelyeknek kapcsolata, összefüggései előre meghatározhatóak, és mindig ugyanazok. Gondoljunk csak egy raktárosra, akinek feladataiban állandóan ismétlődik a

- cikkek beérkezése,
- elhelyezése a tárolóhelyeken,
- cikkek előkeresése a tárolóhelyekről,
- a cikkek kiadása,
- leltár,
- rovancs,
- válaszadás a "hány herkentyű van éppen?" kérdésekre,

Szinte mindegyik feladatot megoldhatjuk az ismert modulok segítségével, de ez láthatóan egységes egészbe, rendszerbe kapcsolt adatkezelést kíván, amelyre a dBASE általánosított eszközei önmagukban nem adnak megoldást.

Az alkalmazás fogalma

Egy-egy feladatcsoport számítógépes támogatására összeállított, rendszerként működő dBASE IV utasításokat és modulokat nevezzük dBASE **alkalmazás**nak.

A hétköznapi beszéd során gyakran felcseréljük az "alkalmazás" és a "program" szavakat, ezért most pontosítjuk jelentésüket. A program valamely programnyelven (esetünkben ez a dBASE IV) megírt utasítások összessége, amely valamilyen meghatározott feladatot hajt végre. Meghatározásunkat így tovább pontosíthatjuk: Alkalmazásnak általában a programokból felépített rendszert nevezzük, amely több, egymástól különböző, de a felhasználó szempontjából összefüggő feladatot old meg.

Az alkalmazás bonyolultsága mindig a felhasználói igényektől, a céltól függ.

5.1.2. Az alkalmazás komponensei

Az adatfeldolgozó alkalmazások különböző típusú komponensek szerves együtteséből állnak. Ilyen komponenstípusok: a fájlok, a menürendszer, az interaktív adatbevitel, a kötegelt feldolgozóeljárások. Természetesen egy-egy típusból több konkrét előfordulás is található egy alkalmazásban, sőt egy-egy típuson belül a komponensek mindig különböznek egymástól.

Az alkalmazás definícióját kicsit pontosítva, és ehhez a fejezethez közelítve azzal is kiegészíthetjük, hogy egy alkalmazás az őt alkotó komponensek működő összessége.

5.2. Általános tudnivalók az alkalmazás-generátorról

Az dBASE IV **alkalmazás-generátora** egy olyan tervező rendszer, amellyel a felhasználói céloknak megfelelő komponensek elkészíthetők, és ezek a komponensek összekapcsolhatóak egységes egésszé. A dBASE IV alkalmazás alatt végülis az egységes egésszé kapcsolt komponensek összességét érthetjük.

5.2.1. Az alkalmazás-generátor objektumai - a komponensek

5.2.1.1. Komponenstípusok

Egy alkalmazás legfontosabb komponenseiből az adatokat illetve ezeket valamilyen formában megjelenítő **fájlokat** (nézetfájl, lista-, címke-, képernyőformátumfájl) az alkalmazás-generátor csak felhasználja. Ezeket megtervezni a Control Center megfelelő paneljei segítségével lehet.

Az alkalmazásgenerátor által kezelt komponenstípusok az alábbiak:

Menü — Menu : Vonal- és rolómenük, és ezekből kapcsolt menük, amelyeknek elemei előre meghatározottak.

Listamenü — List : Olyan menük, amelyeknek elemei nem előre meghatározottak, hanem változóak. Több fajtája van, így:
Files list, amelynek elemei valamilyen tárolón lévő fájlok nevei,
Fields list, amelynek elemei egy fájl mezőinek neveiből állnak,
Values list, amelynek elemei egy fájl egy mezőjének rekordonkénti értékeit reprezentálják.

Eljárásköteg — Batch : A felhasználó beavatkozása nélkül, a "tudtán kívül" végrehajtott tetszés szerinti utasításokból álló utasítássorozat. (Pl.: fájlok másolása, tömörítése, egyéb utasításokat tartalmazó eljárások, stb.).

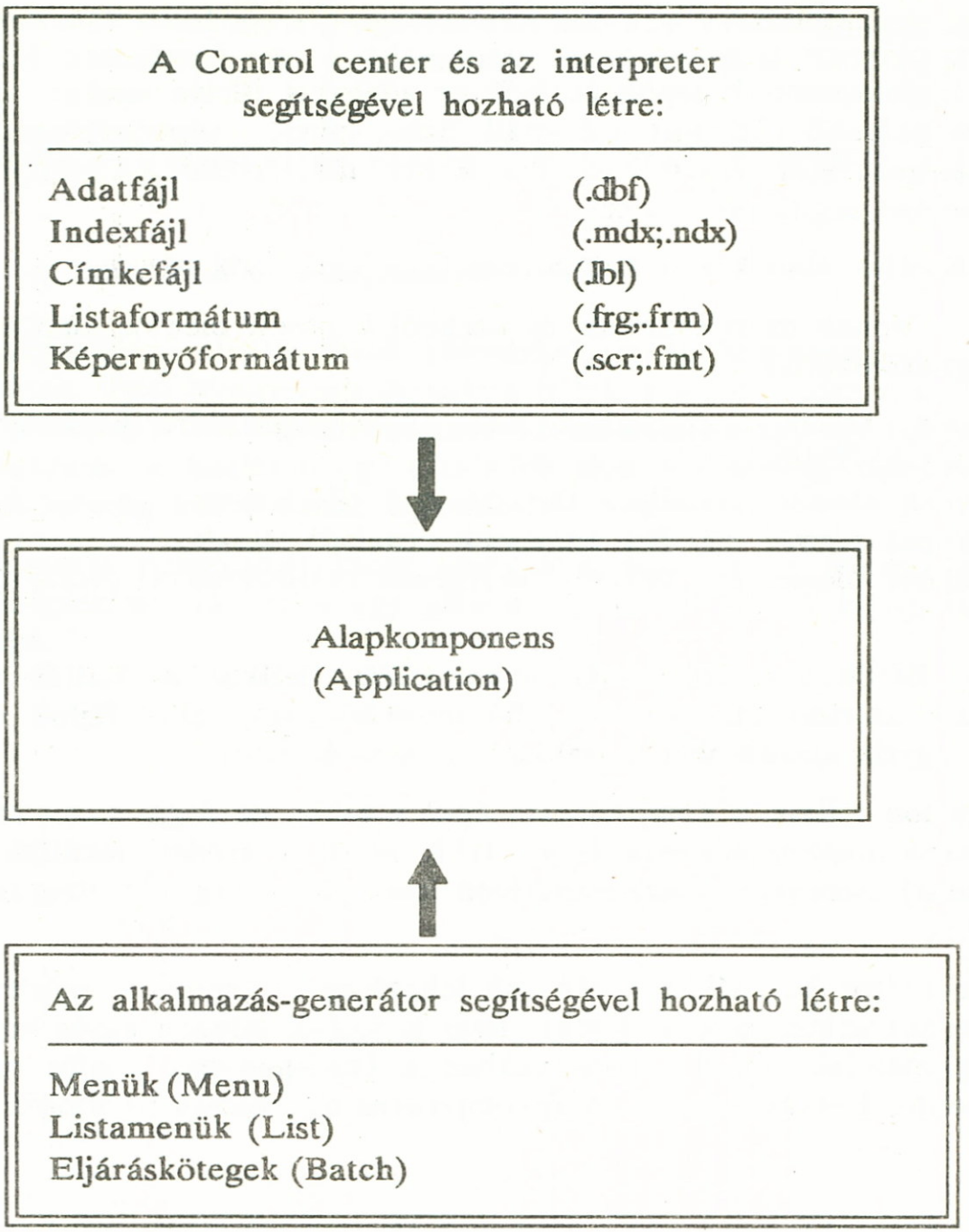
Alap — Application : Erre a komponensre épül a többi, ez fogja össze azokat. Itt most önkényesen *alapkomponensnek* nevezzük, mert az eredeti fordítása (alkalmazáskomponens) esetenként összetéveszthető lenne. Ebből egy alkalmazás esetén csak egy lehet.

Az alkalmazás-generátor használata során sok lehetőség és tennivaló szinte azonos minden komponenstípus esetén. Ezeket nem is fogjuk minden egyes komponens esetén megismételni. Ezért a továbbiakban a **{komponens}** jelölés mindig a Menü-, Listamenü-, Eljárásköteg- és Alapkomponens típusoknak bármelyikét takarhatja.

5.2.1.2. A komponensek kapcsolatai

A komponensek közül az egyik kiemelt jelentőségű. Ez az ún. **alapkomponens** (Application), amely a generált alkalmazás törzsének is tekinthető. Ehhez kapcsolódik az összes többi komponens, amelyeknek nagyobb részét az alkalmazás-generátorral, míg egyet (Fájl) a Control Center segítségével hozhatunk létre. Az alapkomponens különbözik még a többitől abban is, hogy amíg a többi típusból bármennyi előfordulhat egy alkalmazásban, addig ebből csak egy.

Az elmondottakat illusztrálja az alábbi összefoglaló ábra, amelyen jól látható a komponensek keletkezési helye és kapcsolatrendszerük is.



5.2.2. Az alkalmazás-generátor tulajdonságai

5.2.2.1. Rálátás a dBASE IV-re

Az alkalmazás-generátor látszólag nem más, mint a Control Center egyik panelje, a valóságban azonban ennél lényegesen több, tehát nem véletlen, hogy külön fejezetet szenteltünk neki. Tekinthetjük úgy, mint a dBASE IV felhasználói csatolóinak koronája, hiszen a nyelv valamennyi elemét egybefogva mintegy felülről "látja" a dBASE IV egész eszköztárát. Ennek az eszközrendszernek egyfajta csoportosításaként foghatók fel a bevezetésben említett komponensek, amelyek (mint látni fogjuk) a tervezés globális egységei lesznek. Az átfogó tulajdonság abban is megnyilvánul, hogy az alkalmazás-generátor használata közben a dBASE szinte minden tervezőrendszere (adatfájtervezés, lista- és képernyőformátumtervezés, stb) elérhető.

5.2.2.2. Vizuális eszközök

Az alkalmazás-generátor a tervezési munkához egy ún **what you get is what you see, azaz azt kapod, amit látsz** környezetet teremt. Ez azt jelenti, hogy a tervezett komponenseket úgy tudjuk elhelyezni a képernyőn, méretüket oly módon változtathatjuk, hogy eközben látjuk is munkánk eredményét, vagyis a komponensek futás közbeni megjelenési formáit.

5.2.2.3. Rugalmasság

Előfordulhat, hogy az alkalmazás-generátor által biztosított szolgáltatások valamilyen oknál fogva nem elegendőek feladatunk megoldásához vagy vannak régebben megírt "kedvenc" programrészleteink, amelyekről nem szeretnénk lemondani. Nos, a rendszer szerencsére elég rugalmas ahhoz, hogy az ilyen és ehhez hasonló problémáinkat át tudjuk hidalni. Módunk van ugyanis ún. beágyazott programkódot illeszteni a tervezett alkalmazáshoz (később pontosan látni fogjuk ennek megvalósítását). Ez azt jelenti, hogy gyakorlatilag bármilyen (szintaktikusan helyes) program korlátozás nélkül beépíthető rendszerünkbe.

Az alkalmazás tervezése során pontosan elő kell írnunk mindazon feltételeket, amelyek megszabják a felhasználó lehetséges tevékenységeit futtatás közben (pl.: mely fájlokkal milyen műveleteket végezhet, illetve kell végeznie). Ha mégis lazítani szeretnénk eme kötöttségeken, ezt a **&listval** makró sgítségével tehetjük meg. Így például ahelyett, hogy előre rögzítenénk, mely képernyő formátum szerint kell megadni az adatokat, a felhasználó maga választhatja ki a megfelelőt egy listából. A &listval makró az **Item** menü minden olyan ablakában alkalmazható, ahol egyébként fájlnevet, mezőnevet vagy valamilyen értéket kellene megadnunk. A makró használatának pontos leírása az Item menünél olvasható.

Ha a rendszerterv készítése közben elég ügyesek voltunk (lásd a dBASE IV programozás alapjairól szóló fejezetben a modularitással foglalkozó részeket), akkor rendszerünkbe korábban generált alkalmazások komponenseit is beépíthetjük.

5.2.3. Néhány tanács alkalmazások tervezéséhez

5.2.3.1. A tervezést megelőző munkafázis

Szeretnénk mindenkit megóvni attól, hogy túlértékelje az alkalmazás-generátor nyújtotta lehetőségeket és minden előzetes meggondolást nélkülözve nekilásson egy bonyolultabb rendszer elkészítésének. Ha mégis megpróbálja, igen nagy valószínűséggel bele fog keveredni a menük és egyéb komponensek szövevényébe. Az egyre tökéletesebb eszközök sajnos (vagy szerencsére) még nem helyettesíthetik az emberi gondolkodást. Ennyiben nagyon hasonlít ez az autóversenyzésre. A FORMA-1-es versenyző is méterről méterre fejében kell tartsa az egész pályát, a kanyarok íveit, s csak ezután kezdhet neki a látszólag fejvesztett száguldásnak. Ha ezt nem tenné, a sokszáz lóerős csodagép már az első kanyarban elszállna vele. Így tehát **nem úszhatjuk meg rendszereink igen alapos előzetes megtervezését, mielőtt a számítógépet bekapcsolnánk.**

Bármilyen témakörben is szülessen meg alkalmazói rendszerünk, fő vonalaiban azonos tervezési lépéseknek kell megelőznie. Először is át kell gondolnunk milyen információkra van szükségünk. Ennek ismeretében meghatározzuk a bemenő adatokat, ezek képernyőformátumait, a kívánt eredményeket (listák, keletkező fájlok stb.), majd az eddigiek alapján tervezhetjük meg a megfelelő adatfájlokat, ezek kapcsolatait, indexfájljait. A következő lépcsőfok a menürendszer felépítése, s csak utána következhet a megfelelő eljárások (programok) elkészítése. A dBASE IV programozás alapjai szóló fejezetben erről a témáról egy kicsit bővebben is olvashatunk.

5.2.3.2. Az alkalmazás tervezés lépései

A fent említett lépések jellemzik az alkalmazás-generátorral történő rendszerfejlesztés munkafolyamatát is. Ha az eddigieket becsületesen végrehajtottuk, akkor már a kezünkben van a megvalósítandó alkalmazás rendszerterve. Ennek alapján további munkánk az alábbi lépésekre tagolható:

1. A dBASE IV feladatunknak megfelelő konfigurálása.
2. Adatfájl(ok) létrehozása.
3. Indexfájl(ok) kialakítása.
4. Képernyőformátum fájl (.fmt) felépítése.
5. Listaformátum (report) elkészítése.
6. Címkeformátum (label) megtervezése.
7. Felhasználói segédprogramok (pl Assembly rutin) elkészítése.

8. A kód létrehozása az alkalmazás-generátorral.
9. Fordítás, összefűzés (COMPILE, dBLINK).
10. Tesztelés.
11. Alkalmazás.

Az 1-6. pontokat a Control Center-rel (lásd az erről szóló fejezetet) vagy közvetlenül az interpreterből kiadott dBASE IV utasításokkal valósíthatjuk meg. A sorrend természetesen nem kötött.

Nem szabad sajnálni a rendszerek világos, pontos megtervezéséhez szükséges időt, ugyanis ez a későbbiek során busásan megtérül.

5.3. Tudnivalók a használathoz

5.3.1. A szükséges felkészültségről

Az alkalmazás-generátort kezdők és gyakorlott programozók egyaránt (igaz más-más céllal) használhatják. Kezdőknek még viszonylag egszerűbb alkalmazások elkészítése is gondot okozna a hagyományos módon, az alkalmazás-generátorral viszont (megfelelő odafigyeléssel) egész komoly rendszerek megvalósításába is belekezdhetnek (hiszen a legfőbb gondot, az utasítások, függvények szintaxisának pontos ismeretének hiányát az alkalmazás-generátor leveszi a vállukról). A programozásban jártas fejlesztők valószínűleg azért használják majd ezt a segédeszközt, mert tetemes mennyiségű unalmas rutinmunkától szabadulhatnak meg általa, s így elegendő lesz csak a "finomságokra" pazarolni energiájukat.

Használatához elengedhetetlenül szükséges a dBASE legfontosabb elemei **lényegének** (különböző típusú fájlok, ezek létrehozási módjai, kapcsolataik, a menük kezelése stb.) és a Control Center legalább alapszintű ismerete. A tervezéshez nem szükséges feltétlenül a dBASE utasítások pontos ismerete, de – mint azt látni fogjuk – bizonyos feladatok (Eljárásköteg, beágyazott kód) megoldásához ez mégsem nélkülözhető.

A dBASE-ben járatlan olvasónak ezért javasoljuk, hogy mielőtt belekezdene e fejezet tanulmányozásába, ismerkedjék meg a könyv többi részével. A dBASE III PLUS ismeretekkel rendelkező olvasóknak pedig célszerű legalább a dBASE IV és a dBASE III PLUS c. fejezetet végigolvasniuk.

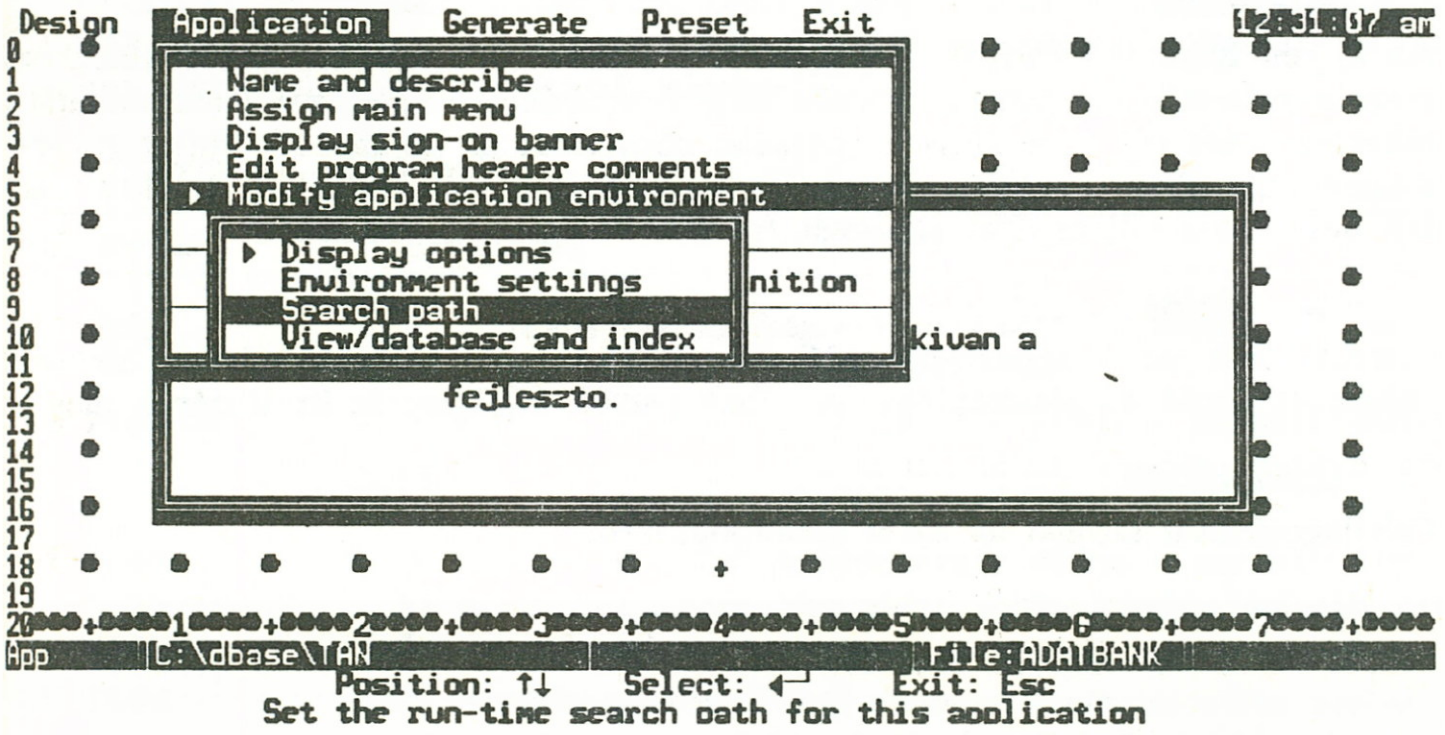
5.3.2. Az alkalmazás-generátor indítási módjai

Az alkalmazás-generátort elindíthatjuk a Control Centerből az Application panel kiválasztásával. Álljunk rá a sávkurzorral a Create mezőre és üssünk **Enter**-t.

Ekkor megjelenik a 5.1. ábra a képernyőn. Amennyiben egy létező alkalmazást akarunk módositani, úgy az Application panelből a megfelelő alkalmazás nevet kell kiválasztanunk.

Az indítás történhet közvetlenül megadott utasítással is (CREATE / MODIFY APPLICATION). Ebben az esetben is az említett ábra jelenik meg a képernyőn.

5.4. Az alkalmazás-generátor munkafelülete



5.1. ábra Az alkalmazás-generátor munkafelülete

A képen felülről lefelé haladva az alábbi részek különíthetők el :

Menüsor.

A választható menük neveit tartalmazza. Az alkalmazás-generátor menüvezérelt, így valamennyi tevékenységünket csak a menük megfelelő kezelésével hajthatjuk végre. Mind a menüsorban, mind az almenükben a már megismert navigáló billentyűk segítségével mozoghatunk.

Almenürendszer.

A kijelölhető almenük neveit tünteti fel. A menüsor további menüket, almenüket aktivizálhat, amelyek szintén meghívhatnak újabb menüket.

Munkafelület

A menüsorok és az állapotsor közötti részen történik a párbeszédes, információs adatforgalom. Az alkalmazásról érkező információ, a program menürendszerének meghatározása és a munkavégzéshez szükséges segítőrendszer (HELP) is itt jelentkezik. Ugyanezt a részt használják a teljesképernyős műveletek is (például a segítőrendszer szerkesztése vagy a dBASE IV kód beillesztése). A képernyőnek ezen a részén folyik a komponensek megtervezése. A munkafelületen látható pontrács az objektumok könnyebb elhelyezését segíti. A képernyő tetején a számok az oszlop koordinátákat, míg a bal szélén lévők a megfelelő sor koordinátákat

jelzik. A munkafelület alaphelyzetben a 0. és a 20. sor között helyezkedik el, de "kinagyíthatjuk" az **F9-Zoom** funkciós billentyű leütésével.

Állapotsor

Az éppen folyó műveletről és állapotról szolgáltat információt. A sor különböző részein más-más állapotjelző jelenik meg. Sorrendben részletezve : megmutatja a művelet nevét (pl generálás) az aktuális könyvtárat és az aktuális rutint (pl menu.gen vagy dbase), szerkesztés esetén a sor, oszlop koordinátákat, az aktív adatfájl nevét és a billentyűzet állapotát (pl INS = beszúró üzemmód).

Navigációs sor

Instrukciókat ad a képernyőn való mozgásokhoz, az opcióválasztáshoz, illetve a választott opció kiegészítéséhez. A hibaüzenetek kijelzése is itt történik meg.

Üzenet sor

Felvilágosítással szolgál az aktív menüpontokról.

5.4.1. A komponensek kezelése

Mint azt láttuk, az alkalmazás-generátor komponens-szemléletű, azaz minden logikai eszköze a komponensekre létrehozására, "felszerelésére" és ezek összekapcsolására szolgál. Ugyanezt tükrözik a technikai eszközök is. A munkafelületen elhelyezett komponenseket addig másolhatjuk, mozgathatjuk, nagyíthatjuk, kicsinyíthetjük, amíg a kívánt méretben a megfelelő helyre nem kerülnek.

Mivel általában sok komponens látszik a képernyőn egyidejűleg, ahhoz, hogy az imént felsorolt műveletek bármelyikét el tudjuk végezni, **először mindig ki kell jelölnünk az aktuális komponenst.** Ezt az **F3** (előző) illetve az **F4** (következő) billentyűkkel tehetjük meg. Ezek után az aktuális komponens másolását az **F8**, mozgatását (az eredeti példány törlődik) az **F7**, méretének változtatását a **Shift-F7** billentyűk leütésével kezdeményezhetjük. A funkciós billentyűket összefoglalva külön részben tárgyaljuk.

A képernyőn egyidőben tartható komponensek száma a rendelkezésre álló szabad memória nagyságától függ. Ha több komponenst próbálunk elhelyezni, mint amennyit az alkalmazás-generátor egyidejűleg kezelni képes, figyelmeztető üzenet jelenik meg, s munkánkat csak akkor folytathatjuk, ha előzőleg néhány komponenst eltávolítunk a munkafelületről (lásd a {*komponens*}.*Put away current* {*komponens*} opciót, ahol {komponens} a Menü, Listamenü, Eljárásköteg komponensek valamelyikét jelöli).

Minden meghatározás és adatbevitel a Ctrl-End billentyűk leütésével véglegesíthető illetve menthető el. Ezt a továbbiakban nem fogjuk külön említeni.

5.4.1.1. A tervezés során használható szerkesztőbillentyűk

A használható vezérlő és navigálóbillentyűzet egységességére elég jól odafigyeltek a dBASE IV fejlesztésénél. Ezért más tervező modulokból már ismerhetjük a billentyűk funkcióját. Szükségesnek láttuk azonban ide is ismételten beszúrni a táblázatot, az alkalmazás-generátorra "élesítve".

Funkciós billentyűk.

Billentyűk	Értelmezés
F1 Help	Az aktuális tevékenységre vonatkozó információt ad, bárhol is járunk az Alkalmazás-generátorban.
F3 Previous	A kurzort az előző komponensre állítja a munkafelületen, és az a komponens lesz aktív, tehát minden művelet erre fog vonatkozni.
F4 Next	A kurzort az előző komponensre állítja a munkafelületen, és az a komponens lesz aktív, tehát minden művelet erre fog vonatkozni.
F5 Field	A (vízszintes) vonalmenü elemeinek elhelyezésére szolgál a képernyőn. A menüelemek kezdő- és befejező pozícióját jelölhetjük ki vele.
F7 Move	A komponensek elmozgatása a munkafelületen (az eredeti példány törlődik).
F8 Copy	A komponensek másolása a munkafelületen (az eredeti példány a helyén marad).
F9 Zoom	Teljesképernyős üzemmód ki-be kapcsolója. Letörli az alkalmazás-generátor menüsorát,üzenet sorát, állapot sorát és navigációssorát, hogy az összes képernyősort elérhessük (alapértelmezésben csak a 20. sorig terjed a munkafelület). Ismételt leütésekor az információs sorok és a menüsor ismét láthatóvá válik.
F10 Menus	Az alkalmazás-generátor menürendszerének aktivizálása.
Shift-F1 Pick	A menürendszer használata közben megjelenő ablakok kitöltésekor a közvetlen beírás helyett sok esetben egy listából kiválaszthatjuk a megfelelő elemet (pl.: fájlnevek, mezőnevek, függvényazonosítók stb.). Ezzel csökkenthetjük a begépelés közben elkövethető hibák számát. A listát ezzel a billentyűvel aktivizálhatjuk.
Shift-F2 Design	A Control Center sávkurzora által kijelölt alkalmazás módosítása.
Shift-F7 Size	A komponenseket körülfogó keretek méretének változtatása.

Navigációs billentyűk.

Billentyű	Hatása
← (LeftArrow)	Komponensben, szerkesztő üzemmódban és az ablakban a kurzort egy hellyel balra mozgatja. A vonalmenükben a sávkurzort egy menüponttal balra mozgatja és megnyitja az esetleg hozzátartozó almenüt.
→ (RightArrow)	Komponensben, szerkesztő üzemmódban és az ablakban a kurzort egy hellyel jobbra mozgatja. A vonalmenükben a sávkurzort egy menüponttal jobbra tolja és megnyitja az esetleg hozzátartozó almenüt.
↑ (UpArrow)	Komponensben, szerkesztő üzemmódban és a ablakban a kurzort egy hellyel felfelé mozgatja. A rolómenüsorokban a sávkurzort egy hellyel felfelé tolja.
↓ (DownArrow)	Komponensben, szerkesztő üzemmódban és a ablakban a kurzort egy hellyel lefele mozgatja. A rolómenüsorokban a sávkurzort egy hellyel lefele tolja.
PgUp	A menürendszer Item opciójában a kurzort a kijelölt komponens aktuális eleme előtti elemre mozgatja. Használatával gyorsabban válthatunk a komponenselemek között.
PgDn	A menürendszer Item opciójában a kurzort a kijelölt komponens aktuális eleme utáni elemre mozgatja. Használatával gyorsabban válthatunk a komponenselemek között.
BackSpace	Az aktuális kurzorpozíciótól balra lévő karaktert törli.
Del	Az aktuális kurzorpozícióban található karaktert kitörli.
Home	Ablakban a kurzort az első opcióra mozgatja. Szerkesztő üzemmódban pedig a kurzor az aktuális sor első karakterére ugrik.
Ins	A beszúró üzemmód, ki-be kapcsolója.
End	Ablakban és listában a kurzort az utolsó mezőre illetve opcióra mozgatja. Szerkesztő üzemmódban pedig a kurzor az aktuális sor utolsó karakterére ugrik.
Tab	Szerkesztő üzemmódban a kurzor a következő tabulátor pozícióra mozog. Menüknél a következő opció első karakterére ugrik a kurzor.
Shitf-Tab	Szerkesztő üzemmódban a kurzor a megelőző tabulátor pozícóra mozog. Párbeszédes ablakban, az előző opció első karakterére ugrik a kurzor.
Enter	A menürendszerben végrehajtja a kiválasztott elemet. Szerkesztő üzemmódban soremelést vált ki.

Billentyű	Hatása
Spacebar	Üres helyet hagy az aktuális pozicióban, illetve bizonyos mezőkben az opciók közötti választás eszköze.
Esc	Az aktuális tevékenység megszakítása.
Alt + billentyű	Az alkalmazás-generátor menürendszere elemei közül történő gyorsabb választást segíti elő. Pl Alt-D kiválasztja a Design opciót.
Ctrl-End	**A szerkesztett komponens véglegesítésére, mentésére szolgál.**
Ctrl-H	Teljesképernyős szerkesztő üzemmódban törli az előző karaktert.
Ctrl-N	Szerkesztő üzemmódban sort szúr be az aktuális kurzorpozició elé.
Ctrl-T	A szerkesztő üzemmódban törli az aktuális szó hátralévő részét
Ctrl-W	A szerkesztő üzemmód és a párbeszédes kommunikáció valamennyi adatát elmenti és visszatér a hívó menübe.
Ctrl-Y	A kurzor által megjelölt sort törli szerkesztő üzemmódban.
Ctrl-RightArrow	Szerkesztő üzemmódban a kurzort a következő szó elejére mozgatja.
Ctrl-LeftArrow	A kurzort a szó elejére mozgatja szerkesztő üzemmódban.

5.4.1.2. Az alkalmazás-generátor végtermékei

A tervezés befejezése után a dokumentált forráskód készül (Generate).

A programkód generálásakor az alábbi két fájl keletkezik:

<alkalmazásnév>.prg

<főmenünév>.prg

Az *alkalmazásnév.prg* a programunk indítását, a fő környezeti jellemzők beállítását, és néhány közös, a generátor által készített általános rutint (pl: hibakezelés, stb) valamint a tervezés során általunk meghatározott komponensek kijelölő utasításait (DEFINE POPUP, DEFINE WINDOW, DEFINE BAR, stb) tartalmazza. A főmenünév.prg indítását is ez a program végzi. Ezért néhol **alapprogramnak** is nevezzük.

A *főmenünév.prg* tartalmazza gyakorlatilag az összes általunk tervezett szerkezeti és művelet jellegű, tehát nem kijelölő, definíciós utasítást. Ezért ezt **szerkezetprogramnak** hívjuk.

A későbbi eredményes használathoz mindkét fájlnak ugyanabban az alkönyvtárban kell jelen lennie.

Ha dokumentációt is generálunk, akkor a

<alkalmazásnév>.doc

fájl jön létre, amely ASCII szövegfájl típusú. Tartalmazza az alkalmazás fontos paramétereit, képernyőképeit, szerkezetét angol nyelvű kommentekkel.

5.4.1.3. Az elkészült alkalmazás használata

Az elkészült, tehát generált alkalmazás, közönséges programnak tekinthető, amelyben szövegszerkesztővel módosíthatunk (saját felelősségünkre) és ugyanúgy tesztelhetjük (Debugger), mintha sajátkezűleg írtuk volna. Az elkészült alkalmazást a Control Center-ből vagy közvetlenül az interpreterből is elindíthatjuk.

5.4.1.4. Az alkalmazás-generátor és a lokális hálózatok

Mivel az alkalmazás-generátor automatikusan elvégzi a szükséges zárolásokat (lásd a hálózatról szóló fejezetet), a generált alkalmazásokat minden megszorítás nélkül használhatjuk hálózati környezetben is.

5.5. Az alkalmazás-generátor menürendszere

Mivel az alkalmazás-generátor kezelése túlnyomórészt a menükön keresztül történik, célszerű az ezzel kapcsolatos lehetőségeket alaposan megismerni.

Az általánosan megjelenő menüsor a következő:

Design {komponens} Item Generate Preset Exit

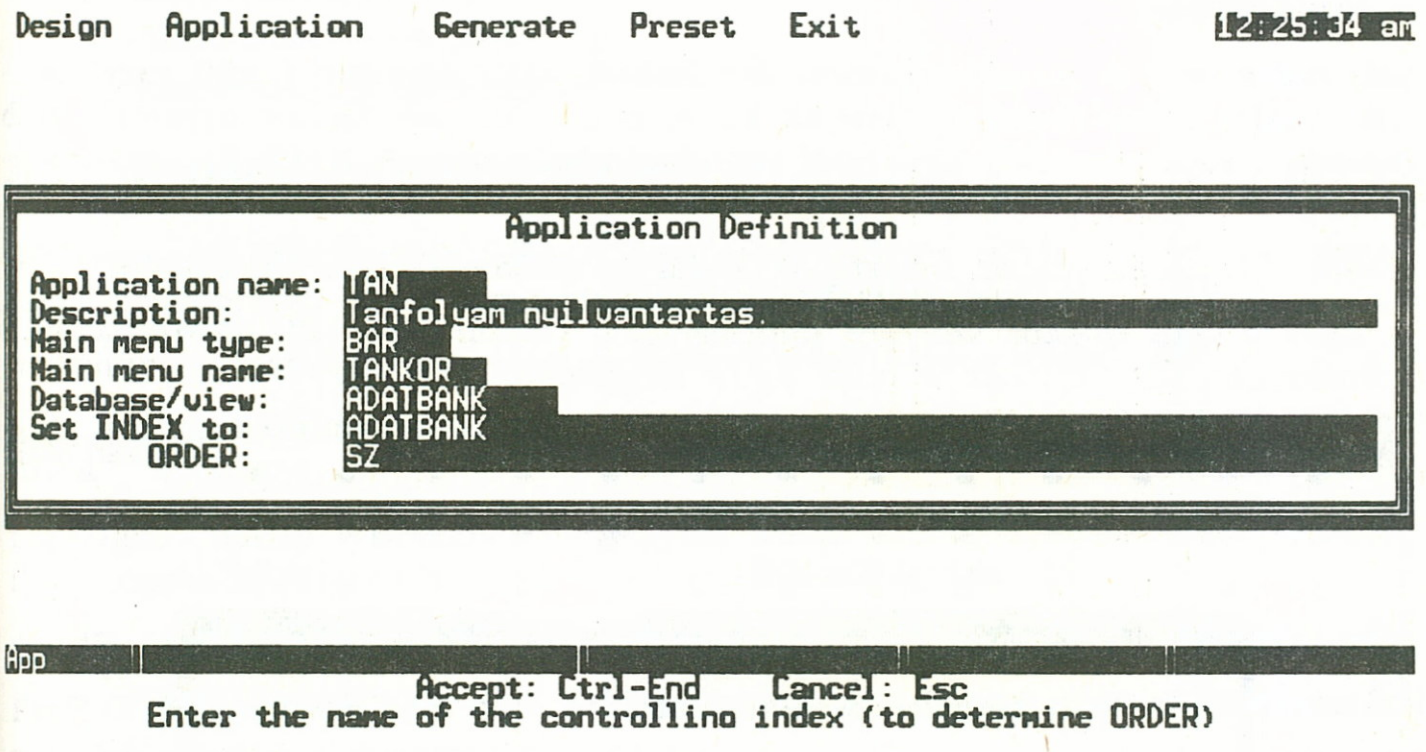
A könnyebb követhetőség kedvéért a menüelemeket nem megjelenési sorrendben ismertetjük, jobbnak látjuk, ha ezeket az egyes komponensek (mint logikai egységek) szerint csoportosítva tárgyaljuk, méghozzá egy olyan lehetséges sorrendben, ahogy a gyakorlati használat során találkozhatunk velük. Így tehát elsőként az alkalmazás működését meghatározó környezet (színek, környezeti jellemzők értékei, aktuális meghajtó stb.) beállítását végző **Preset** menüvel, majd az alapkomponens bejelentkezési képernyőjével ismerkedünk meg, utána pedig bemutatjuk a komponensek kijelölésére szolgáló **Design** menüt. Az itteni választásunk szabja meg, hogy a továbbiakban a fennmaradó három komponens (emlékeztetőül: menü, listamenü és eljárásköteg) közül melyiknek a tervezésével kívánunk foglalkozni. Ez fogja meghatározni a **{komponens}** menü aktuális nevét (és tartalmát is).

Az egyes komponensekhez nagyon hasonló felépítésű menü tartozik, ezért külön tárgyaljuk ezek közös részét, majd rendre a komponensek specifikus menüelemeit.

Ezek után következik a komponensek elemeihez rendelhető műveletek felsorolása az **Item** menü szerint, majd a kód- és dokumentációgenerálás **Generate** folyamatát mutatjuk be. Végül a kilépés, az **Exit** menü használatának módjairól ejtünk néhány szót.

5.6. Az alapkomponens létrehozása új alkalmazásnál

Új alkalmazás tervezésének megkezdésekor kötelező érvénnyel az alkalmazás alapadatait kell megadnunk. Ez gyakorlatilag az **alapkomponens** létrehozását is jelenti. Ilyenkor — ha akarjuk, ha nem — a következő képernyő jelenik meg:



5.2. ábra Az új alkalmazásnál megjelenő ablak

Application name : Ebben a mezőben kell megadni az alkalmazás nevét. A rendszer létrehoz egy **.app** kiterjesztésű fájlt az itt megadott névvel. Ebben a fájlban szerepelnek az alkalmazás meghatározásai.

Description : Az alkalmazáshoz megjegyzést fűzhetünk, például a program felhasználási lehetőségeinek leírását.

Main menu type : Ebben a mezőben határozzuk meg a főmenü típusát. A főmenü lehet *BAR* - vízszintes, vonalmenü vagy *POP-UP* - függőleges, rolómenü vagy *BATCH* - eljárásköteg típusú. (Ez utóbbi nem is igazi menü, hiszen választási lehetőséget nem kínál.) Az egyes típusokból a **szóköz** leütésével választhatunk.

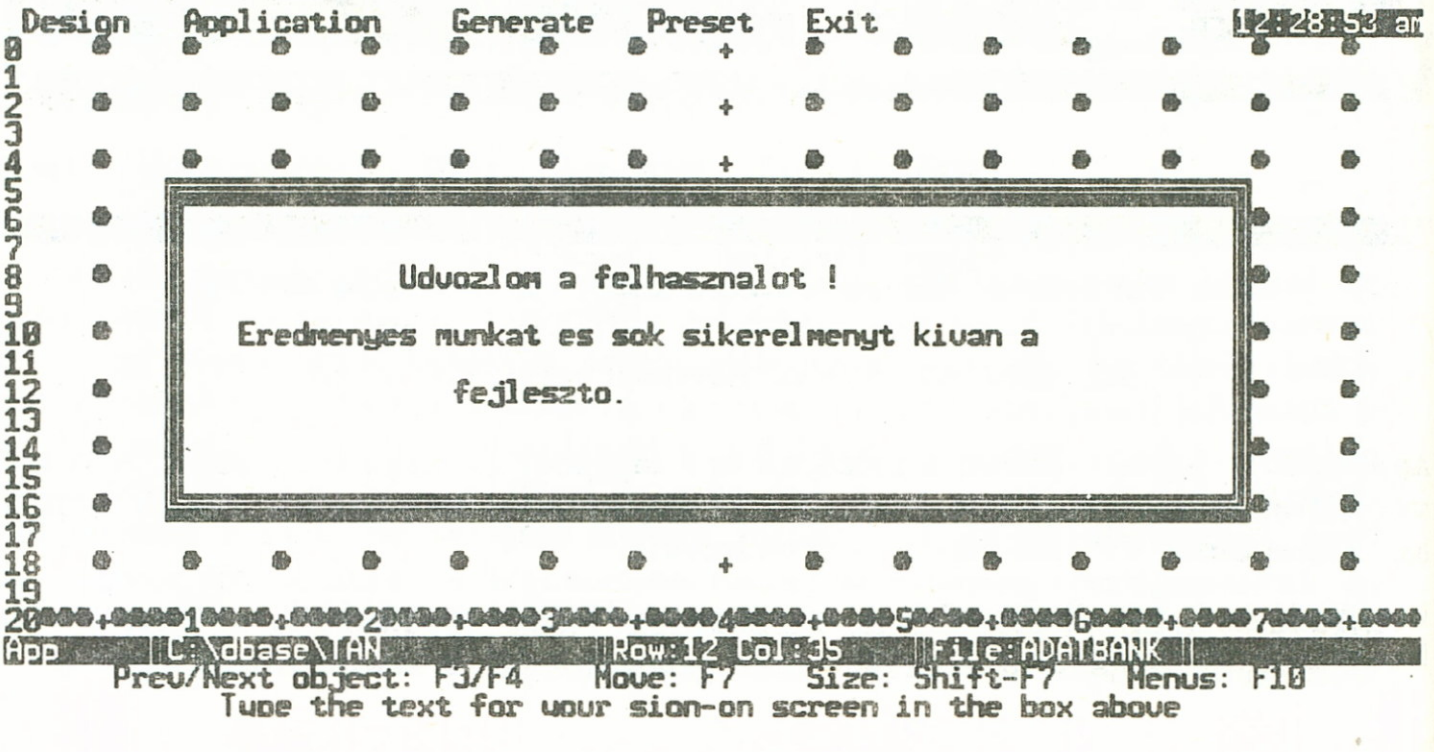
Main menu name : Itt adhatjuk meg a főmenü (főprogram) nevét. Az alkalmazás-generátor a programírás után két utasításfájlt hoz létre. Az egyik programfájl (.prg) neve az alkalmazás nevével, a másik programfájl neve pedig, a főmenü nevével fog megegyezni. A rendszer indítása az alkalmazás nevének begépelésével történik. A **Shift-F1** egyszeri lenyomása után egy főmenü lista jelenik meg a képernyőn, amelyek közül a sávkurzorral választhatunk.

Database/view : Ebben a mezőben jelölhetjük ki a programindításakor aktuális adatfájlt, azaz az alkalmazáshoz tartozó **elsődleges adatfájlt**. A **Shift-F1** leütése után egy ablakban megjelenik az összes fájl, amelyből a sávkurzorral lehet választani. Ugyanezt a megnyitási formát használhatjuk minden egyes almenüben, azaz komponensnél is, tehát az elsődlegesen kívül szükség esetén más adatfájlok is használhatók.

Set index to : A programrendszer indulásakor aktív egyszerű (.ndx) vagy összetett (.mdx) indexfájlokat jelölhetjük ki. A megnyitott adatfájlhoz egyszerre több indexfájlt kapcsolhatunk, egymástól vesszővel elválasztva. A **Shift-F1** egyszeri leütésével a megjelenő listából választhatunk.

Order : Kijelölhetjük az indexfájlok közül az elsődleges fájlt (indextagot).

A mentés után ablakba szöveges információkat vihetünk be. (Pl: üdvözölhetjük a felhasználót, stb). Ez lesz a futás során megjelenő nyitókép.



5.7. Az alapkörnyezet beállítása - Preset

A **Preset** menü olyan jellemzők értékeinek beállítására szolgál, amelyek minden ezután tervezett alkalmazás esetén állandóak maradnak. Így, ha több alkalmazást generálunk egymás után, valamennyi keletkező programrendszer közös jellemzőkkel rendelkezhet, tehát nem szükséges az újabb alkalmazásokat ismét paraméterezni. Az itt beállított jellemzőket (például színek, keret típusok stb.) kínálja fel az alkalmazás-generátor az alkalmazás tervezése során. Természetesen ezek a jellemzők szükség esetén felülírhatók, valamennyi alkalmazói programban. A menüpontot kiválasztva a következő ábra jelenik meg a képernyőn :

Preset

Sign-on defaults
Display options
Environment settings
Application drive/path

5.7.1. A nyitókép - Sign-on default

A tervezendő rendszerünk felhasználása során először megjelenő kép, az úgynevezett bejelentkező **nyitókép** tartalmát határozhatjuk meg.

A menüpont megválasztása után megjelenő ablak három részből áll:

Author : Az alkalmazás szerzőjének- tehát a programozónak a nevét kell itt beírnunk.

Copyright notice : Bevezető megjegyzés írható esetleg az alkalmazás használatának jogosultságáról, üdvözölni lehet a felhasználót, stb.

dBASE version : A dBASE család megfelelő változatszámát tüntethetjük fel.

Ezek az információk nem csak a képernyőn jelennek meg, hanem dokumentáció generálása során a dokumentációban is.

Amennyiben az aktuálisan szerkesztendő felhasználói programban alkalmazni kívánjuk a most kijelölt szalagcímet, az **Application • Display sign-on banner** menüpontban feltett kérdésre **Y** (yes) választ kell adnunk. Az egyes alkalmazásoknál a szalagcímet, az **Application • Edit program header comments** kiválasztásával írhatjuk felül.

5.7.2. A képernyőjellemzők - Display options

Az alkalmazások futtatása során általánosan érvényes képernyőjellemzőket állíthatjuk be. A helyi képernyőjellemzőket a megfelelő komponens **Modify display options** menüelemének segítségével változtathatjuk meg (lásd később a komponenseknél).

5.7.3. A környezeti jellemzők - Environment settings

A környezeti paramétereket (lásd SET utasítások) állíthatjuk be itt valamennyi alkalmazásra vonatkozóan. Az ablakban megjelenő környezeti változókon az **Enter**-rel mozoghatunk, értékeiket pedig a szóköz leütésével változtathatjuk meg. Az így beállított értékeket egy konkrét alkalmazás esetén az *Application • Modify application environment* menüpont segítségével változtathatjuk meg.

5.7.4. Meghajtó, elérési út - Application drive/path

Az alapértelmezés szerinti lemezmeghajtó illetve elérési út meghatározására szolgál minden ezután létrehozandó alkalmazás esetére. A drive mezőbe tehát azt a meghajtónevet kell beírnunk, amelyen az összes művelet végrehajtásra kerül, és a rendszerhez szükséges objektumok (adatfájlok, formátumok stb.) futás közben megtalálhatók. A **Shift-F1** lenyomásakor egy ablakban megjelennek a lehetséges meghajtónevek. (lásd a SET DEFAULT utasítást).

A **search path** mezőben meg kell adnunk azt az alkönyvtár elérési utat, amely alapján a dBASE IV megtalálja az alkalmazáshoz szükséges objektumokat.

Egy konkrét alkalmazáshoz az így megadott értékeket az *Application • Modify application environment* opció megválasztásával módosíthatjuk.

5.8. A komponenesek kijelölése - Design

Ez a menü a tervezés során mindig jelen van a képernyőn és mindig ugyanazokat az elemeket tartalmazza. Itt kezdeményezhetjük az egyes komponensek (Menü, Listamenü, Eljárásköteg) kijelölését az Alapkomponens kivételével, hiszen azt az alkalmazás-generátorba történő belépéskor kell elkészítenünk (lásd feljebb). A dBASE IV minden komponensbeli elemet egy-egy fájlban tárol, amelynek nevét mi határozzuk meg a tervezés során, kiterjesztését azonban a típustól függően az alkalmazás-generátor adja. Az alábbi táblázatban ezt foglaltuk össze:

Komponensnév	Kiterjesztés
Application (Alkalmazás)	.APP
Horizontal bar menu (Vonalmenü)	.BAR
Pop-up menu (Rolómenü)	.POP
Files list (Fájlok listája)	.FIL
Structure list (Mezők listája)	.STR
Values list (Mezőértékek listája)	.VAL
Batch process (Kötegelt utasítások)	.BCH

Miután kiválasztottunk egy elemet a menüből, megjelenik a képernyőn a második menüsor
Design, {komponens}, Item, Generate, Preset, Exit menüpontok.

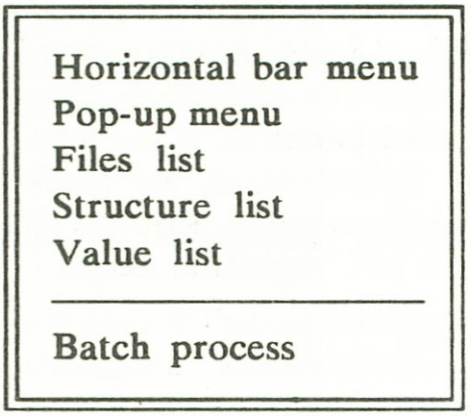
A {komponens} konkrét megjelenési formája Menu (Menü), List (Listamenü) illetve Batch (Eljárásköteg) lehet az alábbi táblázat szerint:

Kijelölés	Komponens neve
Horizontal bar menu, Popup menu.	Menu
Files list, Structure list, Values list.	List
Batch process	Batch

A **{komponens} menü** tehát a kijelöléstől függően, a felsorolt neveket tartalmazhatja.

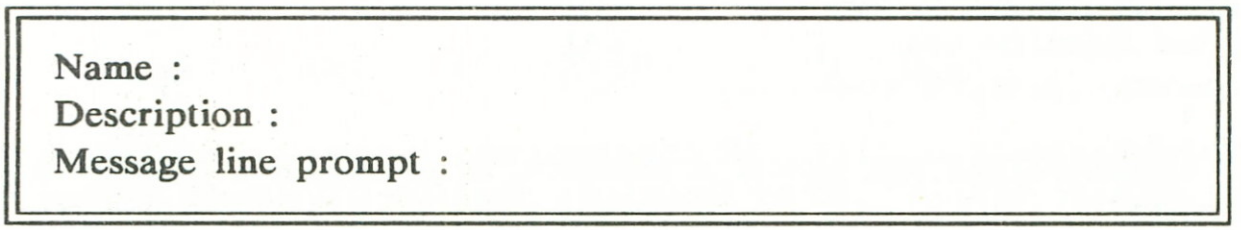
A menüpont kiválasztásakor az alábbi rolómenü jelenik meg:

Design



A komponensek alapadatai

Bármelyik komponenst is választjuk, mindegyiknél az alábbi ablak jelenik meg:



Name : A komponens nevét kell megadnunk.

Description : Magyarázó szöveget vihetünk be, amelynek a dokumentációnál, a forráskódú programnál és a későbbi módosításoknál lehet emlékeztető, tájékoztató szerepe..

Message line prompt : A felhasználónak szóló üzenet magyarázat vihető be, amely a futtatás során jelenik meg a komponens felhasználó által történő kiválasztásakor. Ez legfeljebb 76 karakter hosszú lehet.

5.8.1. Vonalmenü - Horizontal bar menu

Vízszintes elrendezésű, azaz vonalmenü létrehozására vagy a megfelelő kiválasztására szolgál. Ez a menü összekapcsolható egy rolómenüvel, így a vonalmenü lesz a főmenü, a hozzákapcsolt rolómenü pedig az almenü. Az így kapott összetett menüt hívjuk *kapcsolt menünek.*

Válasszuk ki a menürendszerből a **Horizontal bar menu** elemet, amire megjelenik a következő ablak:

Ha a sávkurzorral **create** opciót választjuk, akkor új menüt hozhatunk létre. Egy régebben kijelölt menü is beépíthető programunkba, ha a menülista valamelyik elemére mutatunk. Gyorsíthat a tervezésen, ha egy ilyent módosítunk.

Az alapadatok (Name, stb) meghatározása után megszerkeszthetjük a menüpontokat. Ahová helyezzük megjelenő menüablakban a menüpontokat, úgy fognak azok elhelyezkedni felhasználói programunkban is. Minden egyes menüpont elejét és végét meg kell jelölni az F5 funkciós billenyűvel.

Az aktuális komponens neve **Menu** lesz.

Figyelem ! Az alapkomponensben és a Menü komponensben a főmenü nevének egyezni kell.

Példa:

5.8.2. Rolómenü kijelölése - Pop-up menu

A menüpont rolómenük létrehozására vagy kiválasztására szolgál. Kapcsolható a vonalmenühöz.

Válasszuk ki az alkalmazás-generátor menürendszeréből a rolómenüt és üssünk **Enter**-t, amire megjelenik a következő kitöltendő ablak :

Ha a sávkurzorral a **create** opciót választjuk ki, akkor új menüt hozhatunk létre. Programunkban módosítással vagy anélkül, alkalmazhatunk már régebben létrehozott hasonló formájú menüt, a gyorsabb tervezés érdekében.

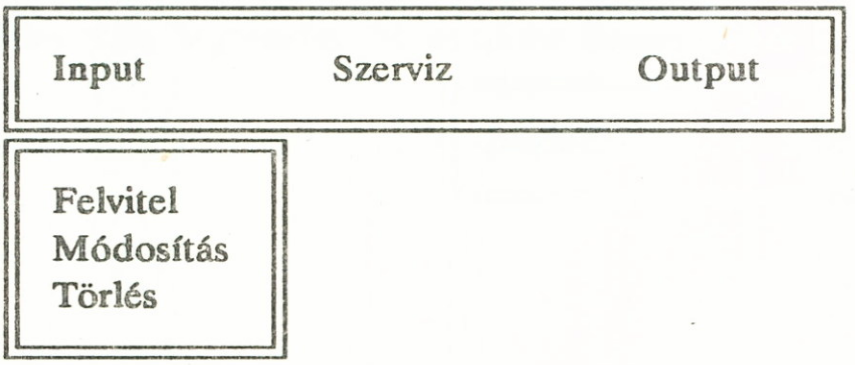
Az alapadatok (Name, stb) megadása után következik a menüpont szövegeinek (prompt) meghatározása.

A rolómenü ablakának a mérete a **Shift-F7** billenyűk lenyomásával változtatható. Az ablakot a felhasználói programunk által igényelt helyre lehet mozgatni. A mozgatás az **F7** funkciós billentyű segítségével oldható meg.

Az aktuális komponens neve itt is **Menu** lesz.

Példa:

Kapcsolt rolómenü:



5.8.3. Listamenü fájlnevekkel - Files list

Előfordulhat, hogy az alkalmazás során nem előre meghatározott menüpontokból kell választani, hanem lehetővé kell tennünk, hogy a felhasználó fájlokból választhasson. Erre szolgál az olyan menü, amelyeknek elemei valamilyen típusú fájlok neveiből állnak. (Lásd DEFINE POPUP utasítás PROMPT FILES cikkelyét). Példa lehet erre egy adatfájlból többféle listát készítő különböző listaformátumok (Report) közüli választás.

Itt is megjelenik a menüknél már megismert ablak, amelyből a **create** opciót választva új fájl-listát határozhatunk meg, illetve a sávkurzorral régebbit is felhasználhatunk.

Az alapadatok meghatározása után megjelenő ablakot itt is mozgathatjuk, nagyíthatjuk, stb, de eltérően a menüktől csak egy elemet látunk, amelyet nem módosíthatunk. Ez mintaként jelképezi az alkalmazás során megjelenő listaelemeket.

Az aktuális komponens neve **List** lesz.

5.8.4. Listamenü mezőnevekkel - Structure list.

Előfordulhat az is, hogy az alkalmazás során valamilyen megjelenítéskor meg kell engednünk a felhasználónak, hogy ő válasszon, mit akar látni egy adat- vagy nézetfájl mezői közül. Erre szolgál az olyan menü, amelyeknek elemei egy fájl mezőinek neveiből állnak. (Lásd DEFINE POPUP utasítás PROMPT STRUCTURE cikkelyét).

Itt is megjelenik a menüknél már megismert ablak, amelyből a **create** opciót választva új mezőlistát határozhatunk meg, illetve a sávkurzorral régebbit is felhasználhatunk.

Az alapadatok meghatározása után megjelenő ablakot itt is mozgathatjuk, nagyíthatjuk, stb, de eltérően a menüktől csak egy elemet látunk, amelyet nem módosíthatunk. Ez mintaként jelképezi az alkalmazás során megjelenő listaelemeket.

Az aktuális komponens neve **List**.

5.8.5. Listamenü mezőértékekkel - List values

Bizonyos esetekben arra lehet szükség, hogy egy adat- vagy nézetfájl mezőinek értékeit kínáljuk fel választásra. Ilyen eset lehet pédául, ha egy számla írásakor meg akarjuk könnyíteni a felhasználó dolgát, és a vevő adatainak kitöltésekor a vevőtörzsadatfájl név mezőjének értékeit kínáljuk fel egy ablakban választásra. Ez a listamenü ilyen esetekre szolgál, azaz olyan menüt hoz létre, amelyeknek

elemei egy fájl meghatározott mezőjének értékeiből állnak. (Lásd DEFINE POPUP utasítás PROMPT FIELDS cikkelyét).

Itt is megjelenik a menüknél már megismert ablak, amelyből a **create** opciót választva új mező szerinti értéklistát jelölhetünk ki, illetve sávkurzorral régebbit is felhasználhatunk.

Az alapadatok meghatározása után megjelenő ablakot itt is mozgathatjuk, nagyíthatjuk, stb, de eltérően a menüktől csak egy elemet látunk, amelyet nem módosíthatunk. Ez mintaként jelképezi az alkalmazás során megjelenő listaelemeket.

Az aktuális komponens neve **List**.

5.8.6. Hozzáférés a listákból történt választáshoz - LISTVAL változó

A listamenük különleges menük, amelyeknél a menüpontoknak csak a körét határozhatjuk meg, de magukat a menüpontokat nem. Ha a futás során a felhasználó választását nem tudjuk megismerni és kezelni, valószínüleg az egésznek semmi értelme nincs. Könnyen belátható, hogy a kiválasztott listaelem az alkalmazás további menetét fogja meghatározni, így a tervezéskor is meg kell tudnunk határozni mi történjen, ha a felhasználó az egyik vagy másik listaelemet választotta.

Erre szolgál a **listval** változó, amely a listamenüben megjelenő és a felhasználó által kiválasztott elemet tartalmazza a választás után. Ezt az alkalmazás során felhasználhatjuk úgy, hogy a megfelelő helyekre nem konkrét értékeket írunk, hanem a listval változó értékét makró formában, azaz: **&listval**

Így járhatunk el például akkor, ha egy fájl-listában a listaformátumokat kínáltuk fel, és az ezt követően a kiválasztott fájl szerint akarjuk a listát, jelentést kinyomtatni. Ilyenkor az **Item • Change action • Display or print • Report** megválasztásakor megjelenő ablakban a **Form name** után egyszerűen a **&listval** makrót írjuk. Ennek hatására a kiválasztott listaformátum szerint indul a nyomtatás.

5.8.7. Eljáráskötég - Batch process

A színfalak mögötti munka határozható meg segítségével. Olyan műveletek összefoglalására szolgál, amelyek nem interaktív jellegűek, azaz a futás során a végrehajtásba a felhasználó nem avatkozik bele, azaz az utasítássorozatok végrehajtása **egy kötegben** zajlik. Ez természetesen nem szigorú megkötés, hiszen az eljáráskötegben bármilyen utasítások szerepelhetnek. Tipikus példák: a listázás, fájlmásolás, indexelés.

Itt is megjelenik a menüknél már megismert ablak, amelyből a **create** opciót választva új eljárásköteget jelölhetünk ki, illetve sávkurzorral régebbit is felhasználhatunk.

Az alapadatok meghatározása után megjelenő ablakban több elemet határozhatunk meg, amelyek végrehajtása a futás során megszakítás nélkül, sorrendben történik. Mindegyik elem gyakorlatilag egy utasítássorozatot jelképez, illetve arra utal.

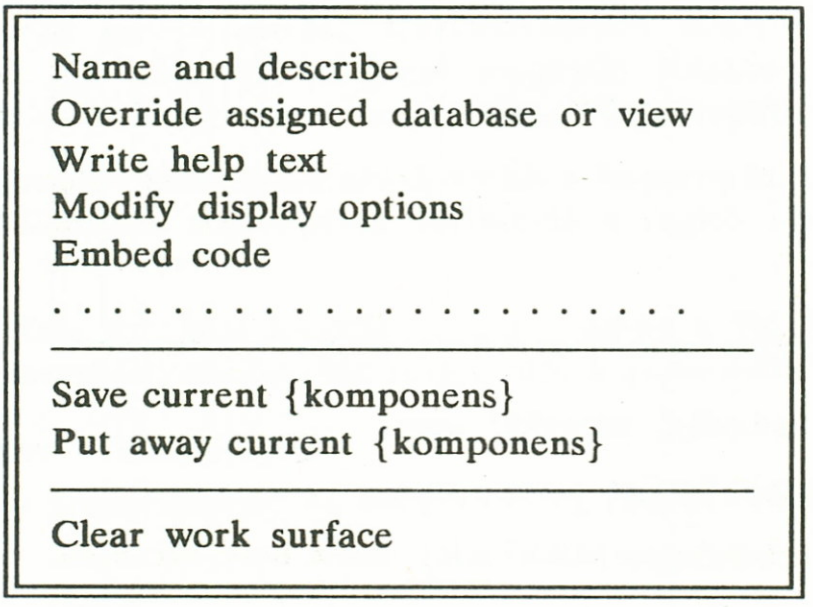
Az aktuális komponens neve **Batch** lesz.

5.9. A komponensjellemzők beállítása- {komponens}

A komponensek jellemzői és környezete minden komponensnél szinte azonos módon állítható be. Amelyik komponensnél egyes menüpontok nem használhatók, ott arra külön kitértünk. A csakis egy komponenseknél használható különleges menüpontokat külön alfejezetben tárgyaljuk.

A komponens menü az alábbi menüpontokkal jelenik meg:

{komponens}



A {komponens} helyén az alapkomponens kivételével mindig az aktuális komponenstípus neve, tehát **Menu / List / Batch** jelenik meg. A kipontozott helyen a Jolly Joker menüpont áll rendszerint. Ez az, amelyik csak egy komponenstípusra érvényes.

5.9.1. Név és leírás - Name and describe

A komponens neve, a hozzátartozó tájékoztató megjegyzés és a felhasználónak kiírandó üzenetsor tartalma változtatható meg. A módosításra is ugyanazok vonatkoznak, amit a komponenskijelölésnél (Design) a komponens alapadatai címszó alatt említettünk.

A komponens nevének a megváltoztatásával a meghatározott jellemzők nem változnak meg. Pl: a rolómenü neve Adatbevitel-ről Törzsadatok bevitele-re változtatható. Ezt más menüpontok opcióival tehetjük meg.

5.9.2. Adatfájl kapcsolása - Override assigned database or view

A segítségével kijelölhetjük az adott komponenshez tartozó adat- vagy nézetfájlt. A generált program indításánál az alapkomponens meghatározásakor kijelölt adatfájl lesz az aktuális, azonban ily módon akár minden komponenshez más-más adatfájlt rendelhetünk. Így nem csak az elsődlegesként megjelölt adatfájl használható rendszerünkben.

```
Database/view :
Set index to  :
ORDER         :

For this {komponens} you may use values . . . . . .

Database/view :
Set index to  :
ORDER         :
```

Database/view : Az aktuális adatfájl, nézetfájl neve.

Set index to : Az aktív indexek listáját tartalmazza.

ORDER : Itt az éppen aktív (elsődleges-MASTER) index neve szerepel.

Az első részben megjelenik az alkalmazás meghatározásakor megjelölt adatfájl neve, a hozzátartozó indexlista és az elsődleges index neve.

For this {komponens} you may use values ..... : "Ehhez a {komponens}-hez a következő opciók használhatók". A kipontozott helyen álló opciók
IN EFFECT AT RUN TIME / ABOVE / ENTERED BELOW
közül a **szóköz** billenyű leütésével választhatunk.

ABOVE

Fordításban: A fenti. Ilyenkor nem határozhatunk meg más fájlt, de a komponenssel kapcsolatban lévő más komponensek fájljai elérhetők.

IN EFFECT AT RUN TIME

A futtatás során aktuálissá váló fájl lesz érvényes. Ilyenkor is elérhető a kapcsolt komponensek összes fájlja. Ezt célszerű használni, ha futás közben egy Listamenüvel a felhasználó választja ki az adott fájlt más komponensben.

ENTERED BELOW

"Az alant beírt", azaz itt (az ablak alsó részében) adhatjuk meg az aktuális komponensben megnyitandó fájl nevét, a hozzátartozó indexek listáját és az elsődleges index nevét. A **Shift-F1** lenyomása a megfelelő fájlok listáját jeleníti meg, amelyből választhatunk. Ha a futtatás során a felhasználó egy korábbi lépésben valamilyen Listmenü alapján jelöli ki a használandó fájlokat, indexeket, ehhez használható mindenütt a **&listval** makró.

Az **alapkomponens (Application) kivételével** az összes komponensnél alkalmazható.

5.9.3. Segítő információk készítése - Write help text

Az aktuális komponenshez szerkeszthetünk segítőinformációkat, amely a *futás során* az **F1** billentyű leütésére megnyíló ablakban jelenik meg, tájékoztatva a felhasználót a komponens használatának rejtelmeiről.

A menüpontot kiválasztva ablak nyílik a képernyőn, ahova a teljesképernyős szerkesztő billentyűk segítségével bevihetjük a segítő adatokat. Erre 19 sor használható.

Amennyiben ezt nem tesszük, a *futás során* a **No help defined** (Nincs kijelölt segítőinformáció) üzenet fog megjelenni a képernyőn az **F1** billentyű leütése után.

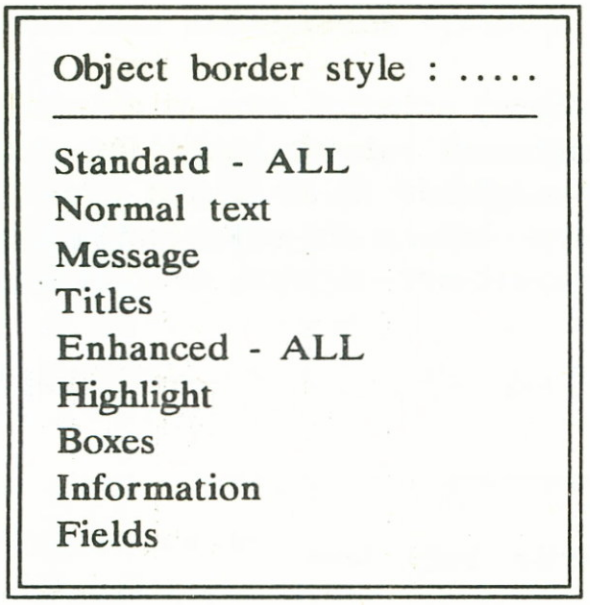
Nem használható:

Az **alap-** (Application) és **Eljárásköteg** (Batch) komponenseknél.

5.9.4. Aktuális képernyőjellemzők beállítása - Modify display options.

A komponensnél érvényes (helyi) képernyőjellemzőket állíthatjuk be. Erre akkor lehet szükség, ha valamilyen ok miatt az éppen aktuális komponenst ki akarjuk emelni a többiek közül, azért hogy ez vizuálisan is jobban vezesse a felhasználót.

A menüpont megválasztásakor a következő ablak jelenik meg a képernyőn:



Object border style : ..... : A keretrajzoló vonalakat állíthatjuk be. A kipontozott helyen álló DOUBLE (duplavonalas), SINGLE (egyvonalas), PANEL (inverz) és NONE (nincs keret) opciók közül a **szóköz** billentyű leütésével választhatunk.

A következő opciókkal a képernyő színeit tudjuk változtatni:

Standard - ALL : Az összes szöveg, üzenet és cím színe azonos módon meghatározható. Így nem szükséges az egyes képernyőrészeket külön meghatározni, de minden egyforma színű lesz.

Normal text : A képernyőn megjelenített normál szövegek színe határozható meg.

Messages : Az üzenetsor a navigációs sor és az általunk megadott üzenet színpárja választható ki.

Titles : A címek és fejlécek színpárja állítható be.

Enhanced - ALL : Valamennyi kiemelt szöveg (inverz, GET mező) színe egyszerre beállítható.

Boxes : Megváltoztatható az ablakok és menük kereteinek a színe.

Highlight : A menük aktuális menüpontját kiemelő színpár határozható meg. Ezt nevezzük sávkurzornak.

Information : Az állapot sor, az óra és a hibaüzenet keretének színét állíthatjuk be.

Fields : A beviteli (GET, INPUT, stb) mezőket kiemelő színpár határozható meg.

A színbeállításhoz akármelyik menüpontot választjuk, a következő ablak jelenik meg a képernyőn :

Foreground	Background
színek	színek

Foreground : előtér

Background : háttér

A navigáló billentyűk segítségével választhatunk a színek között, és választásunkat azonnal ellenőrizhetjük, mert a színek ennek megfelelően megváltoznak.

Nem használható:

Az **alap-** (Application) és **Eljárásköteg** (Batch) komponenseknél. Az alapkomponensnél erre a célra a *Modify application environment* menüpont szolgál.

5.9.5. Végrehajtandó programrészlet készítése - Embed code

A kiválasztott komponenshez programrészletet írhatunk, amely a futás során kerül végrehajtásra. Erre akkor van szükség, ha az alkalmazás-generátor által kínált lehetőségek nem elegendőek feladatunk megoldásához. Ez az ún. *beágyazott kód* (Embed code), amelyről már a bevezetésben is szó esett.

A menüpont kiválasztásakor megjelenő ablak két opciót kínál:

Before
After

Before : A *futtatás során* a programrészlet végrehajtása még az *előtt* megtörténik, mielőtt a vezérlés a komponensre kerülne. Tipikus példa lehet néhány környezeti szolgáltatás (SET AUTOSAVE, stb) beállítása.

After : A *futtatás során* a programrészletre a vezérlés a komponens végrehajtása *után* kerül. A példa akár a környezeti szolgáltatás (SET AUTOSAVE, stb) visszaállítása is lehet.

Az opciók valamelyikének megválasztása után megnyílik a szerkesztőablak és máris hozzáfoghatunk a programrészlet utasításainak begépeléséhez. Az utasításokra nincs különösebb megkötés, így természetesen a DO <programnév> is megengedett. Ez azt jelenti, hogy korábban elkészített eljárásainkat is használhatjuk (ha ezt a tervezés végén generált kódhoz hozzáfűzzük!).

A programrészletet az alkalmazás-generátor nem ellenőrzi le szintaktikailag, így nagyon ügyeljünk a pontosságra.

Nem használható:

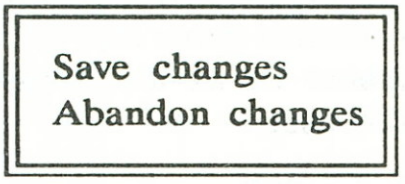
Az **alapkomponensnél** (Application).

5.9.6. A komponens elmentése - Save current {komponens}

A megtervezett komponens aktuális állapotát elementhetjük a háttértárolóra. A komponens a képernyőn marad, és folytathatjuk tovább a munkát.

5.9.7. A komponens eltávolítása - Put away current {komponens}

Eltávolíthatjuk a képernyőről az aktuális komponenst a menüpont kijelölésével, amire a kövekező opciók jelennek meg:



Save changes : Az utolsó mentés óta végrehajtott változtatásokat menti el az aktuális komponensre vonatkozóan úgy, hogy közben törli is a munkafelületről.

Abandon changes : Az aktuális komponenst mentés nélkül törli a munkafelületről.

Nem használható:

Az **alapkomponensnél** (Application).

5.9.8. A munkafelület törlése - Clear work surface

Az **alapkomponens kivételével** törli a teljes munkafelületet, azaz a komponenseket. Használata előtt célszerű menteni. Ha ezt mégsem tennénk meg, a megváltozott komponensekkel összefüggésben egyenként figyelmeztetés jelenik meg a képernyőn, s a mentés ekkor is pótolható.

5.10. Az Alapkomponens menüje - Application

Akár új alkalmazást tervezünk, akár már meglévőt módosítunk a képernyőn megjelenik az alapkomponens, és a menüben is a második pont az **Application** lesz. Ennek rolómenüje az alábbi menüpontokkal rendelkezik:

Application

Name and describe
Assign main menu
Display sign-on banner
Edit program header comments
Modify application environment
Generate quick application

Save current application definition

Clear work surface

Amint látjuk, már ismert menüpontokkal is találkozhatunk. Itt most csak szigorúan az alapkomponenshez tartozó menüpontokat ismertetjük.

Bármelyik menüpontot választjuk is, a Preset menüben megadott jellemzők fognak megjelenni, s ezeket itt megváltoztathatjuk. Az APPLICATION menü a következő, még nem ismertetett menüpontokkal rendelkezik (a minden komponens esetén elérhető opciókat az előző fejezetben tárgyaltuk):

5.10.1. A főmenü megváltoztatása - Assign main menu

Megváltoztathatjuk az aktuális alkalmazás főmenüjének a nevét. Ügyelnünk kell arra, hogy a főmenü csak **Menu** vagy **Batch** típusú komponens lehet. Gyakorlatilag az az ablak jelenik meg, amelyiket az új alkalmazás esetén már megismertünk. Lásd az ott feltüntetett ábrát.

5.10.2. A nyitókép megjelenítése - Display sign-on banner.

Dönthetünk arról, hogy a Preset menüvel kijelölt illetve külön az aktuális alkalmazáshoz megtervezett nyitókép megjelenjen vagy sem az alkalmazás futtatásakor.

A menüpont megválasztása után a következő kérdés jelenik meg:
Display frame as "sign-on banner" at run time ?

Igen, Y (yes) válasz esetén a futtatás során elsőként megjelenő kép a nyitókép lesz.
Nem, N (no) esetén pedig a *főmenü* jelenik meg először.

5.10.3. A forrásprogram fejléce - Edit program header comments.

Megadhatjuk illetve megváltoztathatjuk a tervezésünk eredményeként születő forrásprogramok fejrészében feltüntetendő alábbi adatokat:

Author : A szerző neve

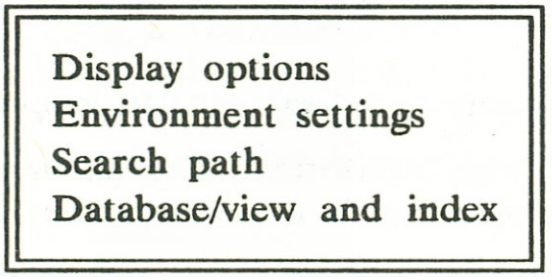
Copyright notice : A felhasználás jogosultságával, szerzői jogvédelemmel kapcsolatos megjegyzés.

Version : Az alkalmazott dBASE IV változat száma.

Amennyiben ez a kijelölés elmarad, a szalagcímben szereplő szöveg kerül a dokumentációba.

5.10.4. A környezet módosítása - Modify application environment

Megváltoztathatjuk az aktuális alkalmazásra érvényes környezetet is, azaz eltérhetünk a Preset menüben meghatározott, minden alkalmazásra általánosan kijelölt környezettől:



Display options : Azonos a működése a többi komponensre érvényes *Modify display option* menüpont működésével.

Environment settings : Az alkalmazás környezeti jellemzőinek, szolgáltatásainak beállítására szolgál. A beállítható paramétereket az alábbiakban felsoroljuk, de részletes ismertetésükre a függelékben kerül sor.

SET BELL
SET CARRY
SET CENTURY
SET CONFIRM
SET ESCAPE
SET SAFETY
SET DELIMITERS

5.10.5. Meghajtó és elérési út meghatározása - Search path

Kijelölhetjük külön az aktuális alkalmazáshoz kötött meghajtót és elérési utat is. A futtatás során a programrendszer mindig itt keresi a fájlokat (hacsak a generált forráskódba bele nem javítunk). A **Shift-F1** itt is a választható listát jeleníti meg.

5.10.6. Elsődleges adatfájl kijelölése - Database/view and index

Az alkalmazáshoz kijelölt elsődleges adat- vagy nézetfájlt és a kapcsolódó indexeket változtathatjuk meg. A megjelenő ablak adatainak kitöltése megegyezik az új alkalmazásnál megismertekkel.

5.10.7. Gyorsgenerálás típusfeladatokra - Generate quick application

A legtipikusabb alkalmazások egy adatfájl karbantartását oldják meg. A leggyakoribb ún. törzsadatkezelő funkciók ilyenkor: az adatbevitel, módosítás, törlés és listázás.

A menüpont segítségével gyorsan létrehozhatunk egy egyszerű felhasználói programot, amely megoldja ezeket a típusfeladatot.

Ha kijelöltünk indexet, úgy a rendszer tartalmazni fog újraindexelés menüt is.

Ha kijelöltünk képernyőformátumot, címkefájlt vagy listaformátumot, akkor az alkalmazás-generátor azokat is beépíti a programjainkba és a menürendszerbe.

A menüpont aktivizálásakor mindössze a következő információkat kell meghatároznunk:

Database file: Adatfájl neve
Report format file: Listaform. neve
Screen format file: Kép.form. neve
Label format file: Címkeform. neve

Set INDEX to: Kapcsolódó indexfájlok neve
ORDER: Elsődleges indexfájl neve

Application author: Szerző neve
Application menu heading: A menü fölött álló cím

Ha a **Ctrl-End** lenyomásával befejeztük az adatok begépelését, újabb ablak jelenik meg, amelyben a feltett kérdésre (Történjen e generálás?) **Y** (yes) választ adva megkezdődik a generálás. **N** válasz esetén ez elmarad.

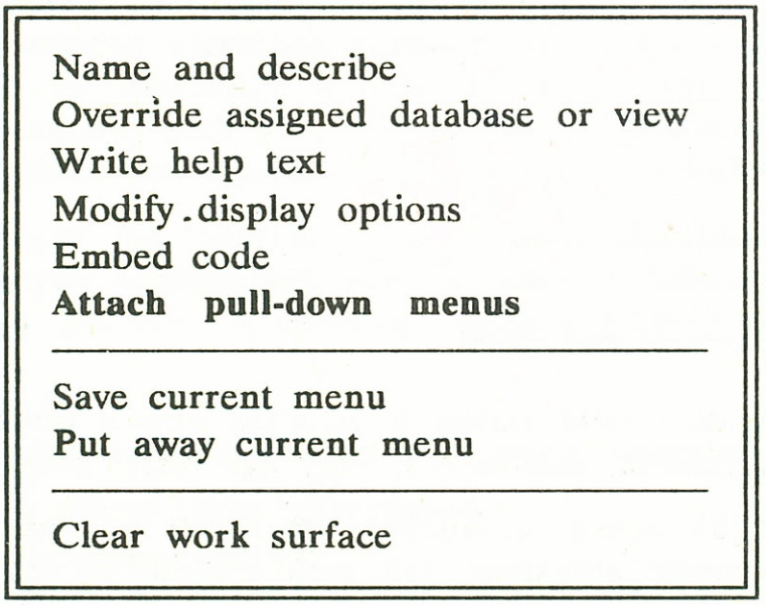
Természetesen a megjelenő menüpontok, üzenetek angol nyelvűek lesznek, így ezeket elég fáradságos munka átírni a generált forráskódban, ha magyarosítani akarjuk. Az alkalmazás-generátor használatakor sok esetben ezt (pl: hibaüzenetek) amúgy sem kerülhetjük el.

5.11. Más komponensek különleges menüpontjai

5.11.1. Menükapcsolás - Menu • Attach pull-down menu

Igen gyakori a két- vagy többszintű menürendszer az alkalmazásoknál. Ilyenkor általában a vonalmenü egyes menüpontjaira mozgatva a kurzort azonnal megjelenik egy rolómenü. Ezt a kapcsolási lehetőséget kínálja a **Menu** komponens **Attach pull-down** menüpontja is. Itt jelezhetjük, hogy az éppen aktuális vonalmenühöz rolómenüket fogunk kapcsolni.

Menu



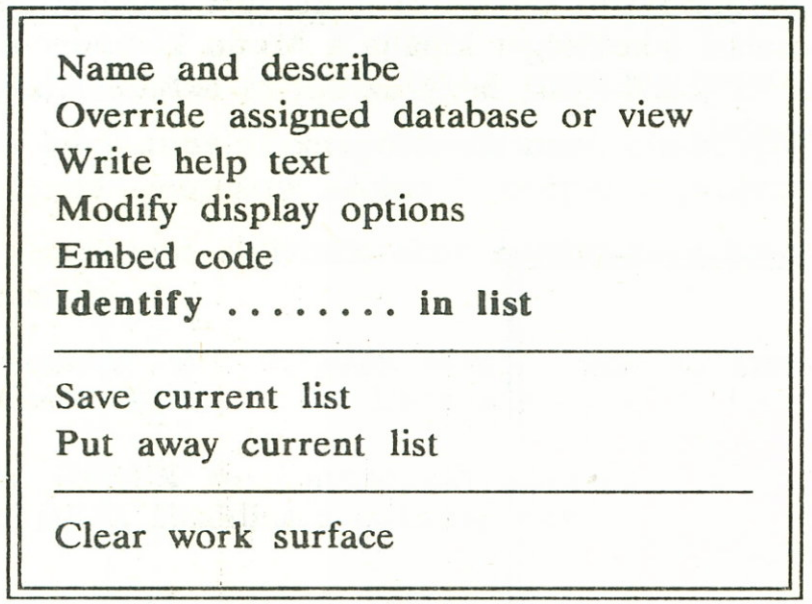
A menüpont csak akkor választható, ha az aktuális komponens a **Horizontal bar menu**. Máshoz nem lehet rolómenüt kapcsolni.

Ha kiválasztás után egy kérdés jelenik meg némi magyarázattal, Erre igent, azaz Y-t kell válaszolni, ha azt akarjuk, hogy az általunk tervezett menü menüpontjaihoz később (*Item • Change action • Open a menu*) rendelt rolómenük a *futtatás során* már akkor megjelenjenek, ha a kurzor a menüponton áll.

5.11.2. A Listamenük elemeinek kijelölése - List • Identify .....

A következő menüopciók akkor jelennek meg ha az aktuális komponens **Files list**, **Structure list** vagy **Values list**.

List



A kipontozott helyen a files / fields / field values kifejezések egyike állhat attól függően, hogy a listaelemek fájlnevek, mezőnevek, vagy egy mező értékei.

Identify files in list : Kijelölhetjük azokat a fájlokat, amelyeket a felhasználó elérhet, azaz a *futtatás során* megnyíló ablakban csak ezek a fájlnevek jelennek meg. A kijelölést minta segítségével adhatjuk meg, amelyben a * és a ? helyettesítő karakterek használhatóak. A meghatározásunk egy *.fil* kiterjesztésű fájlba kerül elmentéskor. Például a **.dbf* begépelésére a futás során a listamenüben az adatfájlok nevei tünnek fel. A menüpont nyilvánvalóan csak akkor jelenik meg, ha az aktuális komponenstípus: *Files list*.

Identify fields in list : Kijelölhetjük az aktuális adat- vagy nézetfájlnak azokat a mezőit, amelyeket a felhasználó elérhet, azaz a *futtatás során* megnyíló ablakban csak ezek a mezőnevek jelennek meg. A meghatározásunk egy .str kiterjesztésű fájlba kerül elmentéskor. Az aktuális adatfájl mezőinek a listája a **Shift-F1** segítségével csalogatható elő. A menüpont nyilvánvalóan csak akkor jelenik meg, ha az aktuális komponenstípus: *Structure list*.

Identify field values in list : Kijelölhetjük az aktuális adat- vagy nézetfájlnak azt a mezőjét, amelynek rekordonkénti értékei jelennek meg a *futtatás során* megnyíló ablakban. A meghatározásunk egy .val kiterjesztésű fájlba kerül elmentéskor. Az aktuális adatfájl mezőinek a listája a **Shift-F1** segítségével csalogatható elő, amelyből egyet választhatunk. A menüpont nyilvánvalóan csak akkor jelenik meg, ha az aktuális komponenstípus: *Values list*.

5.12. A komponensek elemeinek kezelése - Item

Miután létrehoztuk a menürendszert (*Design menü*) és beállítottuk a helyi jellemzőket (*{komponens} menü*), sor kerülhet az egyes elemekhez tartozó eljárások, akciók kijelölésére (*Item*).

A műveletek széles eszköztárát kínálja fel az alkalmazás-generátor, így írhatunk segítő információkat, kijelölhetünk más adatfájlokat, másolhatunk, ablakot nyithatunk, stb. Ha a készen kínált lehetőségek nem elégségesek, magunk is írhatunk programrészeket.

Vannak olyan komponensek, amelyeknek minden eleméhez más-más műveletet és környezetet rendelhetünk, de a Listamenük (*Files list, Structure list, Values list*) minden eleméhez ugyanazt az akciót kell kapcsolnunk. Ez utóbbi esetben hívjuk fel ismételten a figyelmet a konkrét értékek helyett **&listval** makró alkalmazására, ahol a futtatás során a felhasználó által választott érték függvényében rendelhető eljárás illetve környezet a különböző listaelemekhez.

A menüt úgy használhatjuk, hogy kiválasztjuk a kívánt komponenst az **F3/F4** billentyűk segítségével, majd a navigáló billentyűkkel a **megfelelő elemre állítjuk a kurzort.** A menü a szokásos **F10**-re megjelenik:

Item

Show item information
Change action
Override assigned database or view
Embed code
Bypass item on condition
Position record pointer
Reassign index order
Define logical window
Write help text
Assign message line prompt

A menü három pontja: az **Override assigned database or view, Embed code** és a **Write help text** nemcsak nevében, hanem működésében is azonos a komponensmenüével, de értelemszerűen itt nem a komponensre, hanem annak kiválasztott elemére vonatkozik. Ezzel lehetővé válik, hogy ne csak a komponenshez kapcsoljunk más adatfájlt illetve előtte-utána végrehajtandó programrészt vagy segítő-információt, hanem magához a komponens eleméhez is.

Show item information : A menüpont tájékoztató jellegű információkat jelenít meg a *tervezés során* az éppen kiválasztott elemről. Megmutatja a komponens elemének nevét, a kapcsolt adatfájlt és indexek nevét, a kapcsolt eljárás nevét, illetve kapcsolt akció típusát (Change action).

A **Change action** menüpont jelöli ki a komponenselemhez tartozó eljárást. Az összes többi menüpont által meghatározott esetleges tevékenységek (kivétel: Embed code • After) mindig ez előtt kerülnek végrehajtásra. Így például az ablakkijelölés, adatfájlkijelölés, aktuális index váltása, stb.

5.12.1. Akció hozzárendelés - Change action

Egy komponens egy eleméhez számos tevékenység, művelet rendelhető. Ezt a sokféleséget a valóságos szituációk megkövetelik. Az alkalmazás-generátor készen kínálja a menünyitást, adatmegjelenítést és -módosítást, listanyomtatást, fájlműveleteket. Más egyedi programok, eljárások indítására is lehetőség van, erről azonban nekünk kell gondoskodnuk.

A választható lehetőségeket a megjelenő almenü tartalmazza.

Change action

Text (no action)	
Open a menu	ACTIVATE MENU
Browse (add, delete, edit)	BROWSE
Edit form (add, delete, edit)	EDIT
Display or print	
Perform file operation	
Run program	
Quit	

5.12.1.1. Felhasználót tájékoztató szöveg - Text (no action)

A menürendszer létrehozása után, az alkalmazás-generátor az egyes menüpontokhoz ezt az opciót kapcsolja automatikusan mindaddig, amíg ki nem jelölünk valami mást. Ez az eljárás szöveges információt tartalmazhat. Például az Adatbevitel modul megválasztásakor kiíratható a következő szöveg : Az adatok bevitelét elősegítő programrész fejlesztés alatt áll, később használható.

5.12.1.2. Menünyitás - Open a menu

Ha ezt a menüpontot választjuk, a *futattás során* a menüpont felhasználó által történő kiválasztása (**Enter** lenyomása) után, megnyílik az általunk meghatározott menü. Ha a *Menu • Attach pull-down* menus opcióval jeleztük korábban, hogy az egyik menü pontjaihoz rolómenüket csatolunk, akkor a csatolt menü a futás során már akkor is megjelenik, ha .a felhasználó a menüpontra navigál.

Válasszuk ki ehhez *Item • Open a menu* menüpontot.

A megjelenő ablakban válasszuk ki a kapcsolandó menü típusát a **szóköz** segítségével. A következő mezőbe gépeljük be a kiválasztott rolómenü vagy vonalmenü nevét vagy válasszunk a **Shift-F1**-re megjelenő listából.

5.12.1.3. Adatok módosítása - Browse/Edit form

Az adott elemet követheti a *futtatás során* az aktuális adatfájl adatainak módosítása. Erre szolgálnak a **Browse (add, delete, edit)** illetve az **Edit form (add, delete, edit)** menüpontok. A lényeges különbség közöttük gyakorlatilag csak annyi, hogy a Browse táblázatos formában jeleníti meg az adatokat, így a képernyőn több rekord is látható, míg az Edit egymás alatt jeleníti meg a mezőket, és csak egy rekord látható egyszerre.

A megjelenő ablakban az Browse és Edit utasítások cikkelyei és opciói jelennek meg. Ezek részletesen megtalálhatók a függelékben, ezért magyarázat helyett többnyire csak a kulcsszóra utalunk.

Fields : A megjelenítendő mezők köre jelölhető ki. A **Shift-F1** segít a kiválasztásban. Ha semmit sem adunk meg, az összes mezőhöz hozzá lehet férni a futtatás során.

Filter : A vertikális válogatást megvalósító szűrőfeltételeket tartalmazó nézetfájlt, és/vagy szűrőfeltételt adhatunk meg. A feltételeknek megfelelő rekordok adatai jelennek meg.

Fields to LOCK onscreen : lásd **LOCK** cikkely.

FREEZE : lásd **FREEZE** cikkely.

Maximum column WIDTH : lásd **WIDTH** cikkely.

FORMAT file : Kijelölhetünk képernyőformátumot (.fmt) is az adatbevitelhez. Az Edit esetén a formátum szerinti képernyőelhelyezés is érvényesül, míg Browse esetén csak a formátum egyéb előírásai. A formátum kiválasztását a **Shift-F1** lenyomásával könnyíthetjük. Ilyenkor a **create** opció a képernyőtervezőt aktíválja, amivel új formátumot hozhatunk létre.

Allow record ADD : lásd **NOAPPEND** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

Allow record EDIT : lásd **NOEDIT** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

Allow record delete : lásd **NODELETE** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

KEEP image on exit : lásd **NOCLEAR** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

Display browse MENU : lásd **NOMENU** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

Use PREVIOUS Browse table : lásd **NOINIT** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

FOLLOW record after update : lásd **NOFOLLOW** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

COMPRESS display : lásd **COMPRESS** opció. A választ (Yes, No) a **szóköz** nyomkodásával váltogathatjuk.

5.12.1.4. Listanyomtatás - Display or print • Report

Szükség esetén listát nyomtathatunk előre megtervezet listaformátum szerint. Ehhez a menü Report opcióját kell választanunk.

Display or print

Report	**REPORT FORM**
Labels	LABEL FORM
Display/list	DISPLAY/LIST

Ha ezt a menüpontot választottuk, a képernyőn megjelenik egy ablak, amely a következő információkat várja :

Form name : A listaformátum nevét kell megadnunk. Itt is használhatjuk a **Shift-F1**-re megjelenő formátumfájl-listát. Ha ebből a **create** opciót választjuk, a listatervezés munkafelületére jutunk, ahol új formátumot hozhatunk létre.

Heading : Fejléc tartalmának a meghatározására szolgál. Nyomtatáskor valamennyi oldal első sorában megtalálható.

Report format : Kétféle nyomtatási forma közül választhatunk a **szóköz** billenyű segítségével.
A *FULL DETAIL* kijelölése esetén a nyomtatás minden részletre kiterjed, a *SUMMARY ONLY* esetén pedig csak az összegzések kerülnek papírra.

Heading format : Választhatunk, hogy a lista fejlécében megjelenítjük a lapszámot és a dátumot - *INCLUDE DATE AND PAGE*, vagy nem - *PLAIN*.

Before printing : A **szóköz** billentyűvel beállíthatjuk, hogy a nyomtatás előtt legyen lapdobás vagy sem.

Send out to : Meghatározhatjuk a kimenő adatok rendeltetési helyét a következők szerint:

PRINTER - nyomtató
DISK FILE - Szövegfájl
SCREEN - Képernyőn történő megjelenítés
ASK AT RUN TIME - Futás közben tesz fel kérdést a rendszer felhasználónak a rendeltetési helyre vonatkozóan.

5.12.1.5. Címkenyomtatás - Display or print • Labels

Ha borítékokat, árukat kell címkéznünk, választhatjuk az öntapadós címke nyomtatását is.

Display or print

Report	REPORT FORM
Labels	**LABEL FORM**
Display/list	DISPLAY/LIST

A meghatározandó adatok szinte azonosak a listázásnál megismertekkel, de értelemszerűen a listaformátum helyett címkeformátumra vonatkoznak. Lásd még a Control Center címketervezéssel foglalkozó alfejezetét.

5.12.1.6. Adatmegjelenítés adatfolyamban - Display or print • Display/list

A menüpont megválasztásával az adat- vagy nézetfájl adatfolyam típusú, *futtatás során történő* megjelenítését érhetjük el. A megjelenő ablak kérdései már jól ismertek az alábbiak kivételével:

PAUSE at full page/screen : A **szóköz**zel választhatunk a képernyőnkénti (szünettel történő) megjelenítés - *YES*, illetve a megszakítás nélküli között - *NO*.

Include RECORD NUMBERS : NO esetén a rekord sorszámok nem jelennek meg.

5.12.1.7. Fájlkezelő műveletek - Perform file operation

A menüpont segítségével a komponenselemhez csatolható (annak végrehajtását a futtatás során követő) fájlkezelő műveletek egész tárházából válogathatunk.

Perform file operation

File copy	COPY FILE
Add records from file	APPEND FROM
Copy records to file	COPY TO
Substitute field values	REPLACE
Mark records for deletion	DELETE
Unmark records	RECALL
Discard marked records	PACK
Generate index	INDEX ON
Reindex database	REINDEX
Physically sort database	SORT TO
Import foreign file	IMPORT
Export foreign file	EXPORT

File copy : Tetszőleges típusú fájl másolására szolgál. A megjelenő ablak kitöltése értelemszerű, a másolandó fájlok kiválasztását megkönnyíti a **Shift-F1** alkalmazása.

Add records from file : Az aktív adatfájl végéhez rekordokat fűz egy másik fájlból. A megjelenő ablak kitöltéséhez részletesebb információkat lásd a függelékben az APPEND FORM utasításnál.

Copy record to file : Az aktív adatfájlból kijelölt rekordokat, átmásolhatjuk egy másik fájlba. A megjelenő ablak kitöltéséhez részletesebb információkat lásd a függelékben a COPY TO utasításnál.

Substitute field values : Az aktív adatfájl mezőinek a tartalmát írhatjuk felül. A megjelenő ablak kitöltéséhez részletesebb információkat lásd a függelékben a REPLACE utasításnál.

Mark records for deletion : Rekordok törlésre kijelöléséhez szolgál. Lásd a függelékben a DELETE utasítást.

Unmark records : Törlés kijelölés visszaállítására alkalmazható. Lásd még a RECALL utasítást.

Discard marked records : Az aktív adatfájlból törli az összes törlésre kijelölt rekordot és tömöríti azt. Lásd még a PACK utasítást a függelékben.

Generate index : Egyszerű- (.ndx), és összetett (.mdx) indexfájlok illetve indextagok létrehozására szolgál.
Az *Index KEY expression* után kell a kulcskifejezést, indexkifejezést begépelnünk.
A *UNIQUE* opció a duplikált kulcsú rekordokat figyelmen kívül hagyja.
Az *INDEX file* mögé az egyszerű indexfájl nevét kell írnunk. Ha összetett indexfájlt generálunk ezt a mezőt üresen kell hagyni. A következő két mezőt csak az összetett indexfájl létrehozásakor kell kitölteni. Lásd még a függelékben az *INDEX* utasítást.

Reindex database : Az indexek felfrissítésére, újra rendezésére szolgál. Lásd még a függelékben a REINDEX utasítást.

Physically sort database : Az opció segítségével új, valamilyen szempont szerinti rendezett adatfájl hozható létre. A ablak kitöltése könnyen megoldható a *SORT* utasítás ismeretében, amelyről a függelékben olvashatunk.

Import foreign file : Más típusu adatkezelő rendszer adatait is használhatjuk a felhasználói programunkban ennek a menüpontnak kiválasztásával. Részletesebben lásd az IMPORT utasítást a függelékben.

Export foreign file : A kijelölt adatfájlt más adatkezelő rendszerek által értelmezhető formátumú fájlba másolja. Lásd az EXPORT utasítást.

5.12.1.8. Más eljárás, program futtatása - Run program

Ennek a menüpontnak a segítségével dBASE vagy DOS programokat, Eljáráskötegeket, gépi kódú (.bin) programokat és még a billentyűmakrókat is futtathatunk. A megfelelő eljárás kiválasztásához a következő almenü segít:

Run program

Do dBASE program
Execute BATCH process
Insert dBASE code
Run DOS program
Load/call binary file
Play back macro
Quit

Do dBASE program : Futtathatunk egy dBASE III PLUS vagy egy dBASE IV programot. A **Shift-F1** leütése a választható programfájlokról listát ad. Ha **create** opciót választjuk, a szövegszerkesztő lép működésbe, amellyel új programot írhatunk. A *Parameters* mezőbe a programnak átadandó értékek listája kerül.

Execute BATCH process : Egy korábban létrehozott Eljárásköteg (Batch process) kerül végrehajtásra. Az Eljárásköteg nevét kell megadnunk, vagy a **Shift-F1** leütése után megjelenő listából választhatunk.

Insert dBASE code : Helyben is írhatunk programrészt, ha ezt a menüpontot választjuk. A szerkesztőablak azonnal megnyílik és begépelhetjük az utasításokat.

Run DOS program : Kapcsolhatunk egy végrehajtható (.com, .exe, .bat kiterjesztésű) DOS programot is az aktuális komponenselemhez. A DOS fájl nevét kell emgadnunk, de a **Shift-F1** megkönnyíti a kijelölést. Itt is meg kell adnunk az esetleges paramétereket. Pl: *Format* a DOS fájl neve, *a:* pedig a paraméter.

Load/call binary file : Kijelölhetünk egy assembly nyelven megírt, és lefordított (.bin) programot is. Feladatunk most is a futtatandó (.bin) fájl nevének a meghatározása és a szükséges paraméterlista megadása. *Figyelem !* A paraméterek nem adhatók meg a dBASE PARAMETERS analógiájára. A paraméterátadásnak összhangban kell lennie a programfájl típusával. Például, ha a binárisan kódolt fájlnak a 8 numerikus értéket kívánjuk átadni, úgy ezt chr(8) formában tehetjük meg.

Play back macro : Ez a menüpont a megválasztása után megjelölt billentyűmakrót játssza le. A kijelölést a **Shift-F1** használata könnyítheti.

Quit : Visszaléphetünk oda, ahonnét a programot a felhasználó elindította - *Return to calling program*, de kiléphetünk közvetlenül a DOS operációs rendszerbe is - *Quit to DOS*.

5.12.2. A komponenselem megbénítása - Bypass item on condition

Esetenként szükség lehet arra, hogy bizonyos feltételtől függően egy komponenselemet ne lehessen végrehajtani, kiválasztani. Tipikus eset a rolómenü egy:egy menüpontjának "megbénítása".

A megjelenő ablakban a **Skip this item if** után begépelhetünk bármilyen kifejezést, ami logikai értéket (.T., .F.) eredményez. *Hamis* érték esetén a komponenselem nem kerül végrehajtásra, nem választható. Ez a futtatás során egy menüelemnél úgy jelentkezik, hogy azt a sávkurzor "átugorja".

5.12.3. A rekordmutató pozicionálása - Position record pointer

Gyakran előfordul, hogy a szükséges tevékenységhez – annak indítása előtt – valamilyen szempont szerinti rekordot kell megkeresni az aktuális adatfájlban, azaz a rekordmutatót pozicionálni kell.

Ez többféleképpen is megvalósítható. A módszerek közül a megjelenő almenü segít választani:

Position record pointer

```
Display POSITIONING MENU at RUN TIME? :
SEEK first occurrence of key :
GOTO :
LOCATE SCOPE :
FOR :
WHILE :
```

Display POSITIONING MENU at RUN TIME? : A **szóköz** leütésével dönthetünk abban a kérdésben, hogy a rekordpozícionálás módját itt határozzuk meg, vagy ezt a futtatás közben a felhasználóra bízzuk. Ha Y-t válaszolunk, akkor a futtatás közben fog hasonló ablak megjelenni, amelyet a felhasználónak kell kitölteni.

SEEK first occurrence of key : A begépelt kulcskifejezés szerinti első rekordot keresi meg. Lásd függelék SEEK.

GOTO : A kívánt rekord sorszámot adhatjuk meg. Lásd függelék GO TO.

LOCATE SCOPE : Meghatározhatjuk a keresés, pozícionálás hatáskörét. (Pl: NEXT 10) Lásd függelék Jelölések és szimbólumok.

FOR : A beírt feltételnek megfelelő rekordra áll a mutató. Lásd függelék LOCATE.

WHILE : A keresés csak addig folyik, amíg az itt megadott feltétel igaz.

A pozícionálás az előtt történik meg, mielőtt a kiválasztott eljárásra (*Change action*) kerül a vezérlés.

Felhívjuk a figyelmet arra, hogy itt is alkalmazható a **&listval** makró!

5.12.4. Reassign index order

Megváltoztathatjuk a komponenselemhez az aktuális indexet, azaz kijelölhetjük az egyszerű indexfájlok-, illetve az indextagok közül az elsődlegeset, mielőtt a *Change action* pontban kijelölt eljárás megindulna. Az adatfájlhoz kapcsolt .ndx fájlok közül a leírási sorrendnek megfelelő sorszámot kell megadni a kijelöléshez.

Ez a sorszám 0 és 7 közötti érték lehet.

Például *USE ado INDEX nev,cim,szja,kif* esetén, *SET ORDER TO 3* a helyes válasz. Ebben az esetben a szja indexfájl lesz aktív.

Összetett indexfájl elemei közül a következő módon jelölhetünk ki aktívnak : **tag név OF .mdx név**

Például: *SET ORDER TO sz_szam OF ado*

5.12.5. Ablakkijelölés - Define logical window

A kijelölt komponenselem eljárásának indítása előtt meghatározhatunk egy ablakot is, amelyet az eljárás során (pl: Browse) aktíválhatunk.

Meg kell adnunk az ablak nevét, a keret típusát, amelyet a
SINGLE, DOUBLE, PANEL, CUSTOM és NONE
opciók közül választhatunk ki a **szóköz** használatával.

Ha a CUSTOM opciót választottuk, meg kell adnunk a keretet alkotó karaktereket is.

A következő mezőben az ablak színeiről kell rendelkeznünk. Kitöltése a SET COLOR utasításnak megfelelően történik.

Végül az ablak koordinátáit kell megadnunk

Részletes eligazítást találhatunk a függelékben a DEFINE WINDOW utasításnál.

A kijelölt ablakot használhatjuk például a BROWSE ablakaként is, ha éppen a Change action menüpontban a Browse-t jelöltük ki.

5.12.6. Assign message line prompt

A komponenselemhez kapcsolhatunk üzenetszöveget is, amely a futás során az elem kiválasztáskor az üzenetsorban középre igazítva jelenik meg. Az ablak első részében a kijelölt menühöz kapcsolódó üzenetsor határozható meg. A második részben pedig az adott menühöz tartozó eljárások (pl almenük) üzenetsora állítható be.

5.13. Generálás - Generate

Ha készen vagyunk a komponensek megtervezésével, ezt a menüpontot kell választanunk ahhoz, hogy létrejöjjön a meghatározásainknak megfelelő forrásprogram illetve — ha azt kívánjuk — dokumentáció.

A programgenerálás után a fordítás, tesztelés ugyanúgy történik, mintha sajátkezüleg írtuk volna a programot.

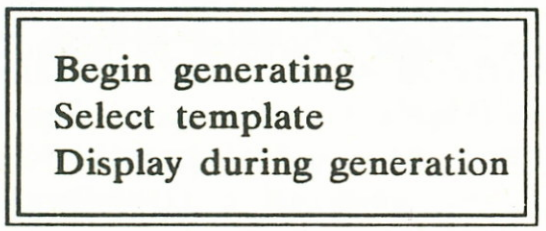
Az alkalmazás-generátor két **.prg** kiterjesztésű programfájlt hoz létre. A fájlnevük megegyezik az alkalmazás, illetve a tervezés során meghatározott főmenü nevével.

A program indításához az alkalmazás nevét kell megadnunk.

A **.doc** kiterjesztésű dokumentációfájl az alkalmazás jellemzőit tartalmazza ékes angol nyelven. Ennek is azonos a fájlneve az alkalmazás nevével.

A generálás időtartama függ a hardver eszközeink sebességétől, illetve az alkalmazás méretétől. Ha a későbbiek során akár csak egyetlen komponenst is megváltoztatunk, természetesen újra kell generálni a kódot, hogy a változás futtatás közben is érvényesüljön.

Generate



Begin generating : Ezt az opciót kiválasztva a második menüpontban beállított értéktől föggően elkészíti a programrendszer forráskódját vagy a dokumentációt. Ha elfelejtettük volna meghatározni a főmenüt (erre felhívja a figyelmünket), úgy ezt itt is megtehetjük még.

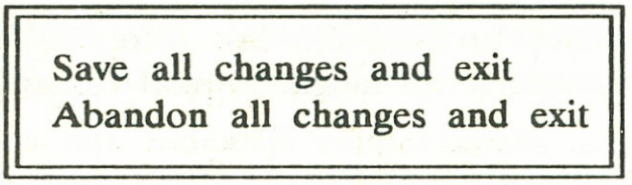
Select template : Itt választhatunk a programkód vagy dokumentációgenerálás között. Ha a megjelenő ablakba a programgenerátor fájlnevét, a **MENU.GEN** nevet adjuk meg, akkor a programgenerálás indul. Ha a **DOCUMENT.GEN** nevet adjuk, akkor a dokumentáció készítése indul.

Display during generation : Ha azt kívánjuk, hogy a képernyőn megjelenjenek a generált sorok, akkor válaszoljunk igent, azaz **Y**-t ennek a menüpontnak megválasztásakor megjelenő kérdésre.

5.14. Mentés, kilépés - Exit

Ha a tervezést befejeztük, vagy később akarjuk folytatni, vagy esetleg meggondoltuk magunkat, és a változtatásainkat mégsem akarjuk megtartani, ebben a menüben kell kiválasztanunk a megfelelő menüpontot.

Exit



Save all changes and exit : A munkafelületen elhelyezkedő komponensek a komponensnek megfelelő kiterjesztésű fájlba kerülnek. A változtatások felülírják ezeket, ha már korábban léteztek. A mentés után kilépés történik a Control Center-be vagy az interpreterbe attól függően, hogy az alkalmazás-generátort honnét indítottuk.

Abandon all changes and exit : Ebben az esetben a belépésünk óta végrehajtott változtatások nem kerülnek mentésre, hacsak menetközben a komponens menü *Save this* .... menüpontjának használatával egyes komponensekre ezt nem tettük meg. A mentés nélküli kilépés előtt jóváhagyólag, azaz Y-nal kell válaszolnunk a rendszer által feltett biztonsági kérdésre, majd a kilépés az indító környezetbe megtörténik.

6. SQL a dBASE IV-ben

Az SQL, azaz **Structured Query Language** (Struktúrált Lekérdező Nyelv) a relációs adatbáziskezelés szabványosítási törekvéseinek egyik – talán a legsikeresebb – eredménye. Fejlesztése az IBM égisze alatt a 70-es évek közepén kezdődött el. Először kísérlet képpen az egyfelhasználós változat készült el. Miután ez a verzió megfelelt a várakozásnak, tovább léptek a többfelhasználós változat irányába. Az SQL 1979-ben jelent meg a számítástechnikai piacon. Bevezetése óta sok társaság szabvány adatbáziskezelő nyelvként vette át. A nagy- és miniszámítógépek után, a személyi számítógépeken is sor került a kifejlesztésére.
Az ASHTON-TATE is csatlakozott ehhez a szabványhoz, és beépítette a szabvány SQL utasításokat és függvényeket a dBASE IV rendszerbe, úgy, hogy lehetőséget biztosít SQL és a mikrogépes dBASE adatbáziskezelő – amely már maga is szabványnak fogadható el – kombinált használatára is.

Maga az SQL táblákkal dolgozik. A tábla nem más mint adathalmazok (tulajdonságok) közötti relációk halmaza. Másképpen, a tábla oszlopokból és sorokból épül fel, azaz analóg a dBASE adatfájl fogalmával. A sorok és oszlopok metszéspontjában találhatók az adatok. Egy konkrét reláció tehát nem más mint a tábla egy sora. **Alaptáblának** nevezzük az olyan táblákat, amelyek a tényleges adatokat tartalmazzák. A virtuális tábla – amelyet **nézettáblának** fogunk nevezni – nem tartalmaz adatot, ehelyett csak az alaptáblákban levő sorokra, oszlopokra hivatkozik. Ebből az is következik, hogy a nézettáblában az adat dinamikusan jelenik meg, azaz az alaptáblában módosuló adatot a virtuális táblában is módosulva láthatjuk. Ugyanez fordítva is működik, azaz a nézettáblában levő sorok megváltozása a hozzájuk tartozó alaptábla adatainak megváltozását is jelenti.

Maga az SQL, mint nyelv, ún. *eljárásmentes* nyelv. Ez azt jelenti, hogy nem az adatok elérésének a módját, lépéseit kell megadni, hanem a kívánt információ tulajdonságait kell leírni.

Végezetül következzen egy táblázat, amely az SQL és a dBASE IV egymásnak megfelelő fogalmait tartalmazza.

SQL terminológia	dBASE IV terminológia
adatbázis	összekapcsolt adatfájlok
tábla	adatfájl (.dbf) (adatbázis)
oszlop	mező
logikai oszlop	számított mező
sor	rekord
nézettábla, virtuális tábla	nézetfájl (view)

6.1. táblázat

6.1. A dBASE IV/SQL alkalmazásáról

A dBASE IV két különböző módot szolgáltat az adatok kezelésére. Az egyik a hagyományos módszer, amikor is a teljes dBASE IV utasításkészlet a felhasználó rendelkezésére áll. A másik, az SQL üzemmód, amikor az összes SQL utasítás és függvény elérhető. Ebben üzemmódban is használhatjuk a dBASE leszűkített utasítás- és függvénykészletét, nevezetesen azokat, amelyek nincsenek konfliktusban az SQL utasításokkal. Az interpreterben a SET SQL ON, illetve a SET SQL OFF utasításokkal lehet a két üzemmódot váltogatni. Az SQL-t programokban is használhatjuk, de ekkor nem szabad a SET SQL utasítást kiadni.

A programoknál a kiterjesztés utal az SQL ill. dBASE eljárásokra. A **.prs** kiterjesztés jelzi az SQL, a .prg kiterjesztés pedig a dBASE IV utasításkészlet használatát. A program végrehajtásakor a dBASE IV automatikusan átvált a megfelelő üzemmódba a kiterjesztéstől függően.

Egy SQL utasítás mindig kulcsszóval kezdődik, mint például UPDATE vagy SELECT, és pontosvesszővel (;) végződik. A kulcsszavakhoz még egy vagy több cikkely tartozhat, amely az utasítás működését befolyásolja. Például:

```
SELECT <oszloplista>
    FROM <táblalista>
    [WHERE <feltétel>] ;
```

Az SQL utasítások beírásánál a dBASE IV minden szolgáltatása rendelkezésünkre áll. Használható a Ctrl-Home és a Ctrl-End billentyű kombináció. Ekkor legfeljebb 1024 karakteres sorokat írhatunk. A RETURN és TAB billentyűkkel könnyűszerrel strukturálhatjuk az utasításokat. SQL üzemmódban is használható az utasítás puffer, és így lehetőség van az utasítások ismétlésére, illetve átszerkesztésére (SET HISTORY ON/OFF). F1 billentyűre itt is mankóinformációkhoz jutunk, mégpedig az SQL utasításokról.

Az SQL hálózati alkalmazása sem bonyolult. Nincs rekord- vagy fájlzárolással kapcsolatos utasítása, de nincs is rá szükség, mert az SQL a konkurens adathozzáférést programozói beavatkozás nélkül automatikusan kezeli.

E fejezet (SQL) további részeiben több példát sorakoztatunk fel. A példák mind a dBASE IV-hez tartozó SAMPLES adatbázisra vonatkoznak.Feltétlenül javasoljuk a példák kipróbálását számítógépen, ennek érdekében azonban a következő előkészületeket kell megtenni:

Először kapcsoljunk SQL üzemmódba az interpreterben,

```
.SET SQL ON
```

majd aktivizáljuk az adatbázist.

```
SQL. START DATABASE SAMPLES ;
```

Ezekre az utasításokra nincs szükség ha a *config.db* fájl tartalmazza a következő bejegyzéseket.

```
COMMAND    = SET SQL ON
SQLDATABASE    = SAMPLES
```

A tömörség érdekében bizonyos, több utasításhoz egyaránt kapcsolódó információkat csak az első utasításnál fogjuk említeni. Ezért ajánljuk a lépésről-lépésre haladás módszerét.

6.2. Adatbázis létrehozása és karbantartása

6.2.1. Adatbázis kezelése

A táblák elszakíthatalanok az adatbázistól (ellentétben a dBASE-zel). Ezért a táblák (adatfájlok) létrehozása előtt létesíteni kell egy új, vagy aktivizálni kell egy már létező adatbázist. Minden adatbázist csak egy DOS könyvtárhoz rendelhetünk, azaz egy könyvtárba kell kerülnie az adatbázishoz tartozó összes táblának és a táblákhoz kapcsolódó egyéb fájloknak. Itt jönnek létre a **rendszerkatalógusok vagy rendszertáblák**, amelyek a táblákat, nézettáblákat, indexeket, másodlagos neveket tartják nyilván és tartalmazzák a táblákhoz való hozzáférési jogokat és a táblák kezelését meggyorsító statisztikai információkat is.

A SQL **rendszer-főkatalógus** magukat az adatbázisokat tartja nyilván. Az adatbázisok *elérési útja* (DOS könyvtár), a *tulajdonos*, a *létrehozás dátuma* szerepelnek benne.
A rendszer-főkatalógus neve **SYSDBS.DBF** és a **\DBASE\SQLHOME könyvtárban** található.

Új adatbázist a **CREATE DATABASE** paranccsal lehet létrehozni. Ez az utasítás nem programozható, tehát csak interaktív üzemmódban adható ki:

```
CREATE [<út>] <adatbázisnév>;
```

Példa:

```
SQL. CREATE DATABASE sajatdb ;
```

Ekkor az aktuális könyvtárba belekerül a *sajatdb* nevű *alkönyvtár*, az alkönyvtárban létrejönnek a szükséges SQL rendszertáblák és a sajatdb adatbázis aktuálissá válik. Ezzel együtt az előzőleg aktív adatbázist a rendszer lezárja. Ha az *út*at is megadjuk, akkor természetesen az út által kijelölt helyen (alkönyvtárban) jön létre az adatbázis. A sajatdb adatbázis bejegyzésre kerül a SYSDBS.DBF fő rendszerkatalógusba is.

A létező adatbázisokat a SHOW DATABASE ; paranccsal listázhatjuk ki.

Példa:

```
SQL. SHOW DATABASE ;

Existing databases are:

NAME        CREATOR   CREATED   PATH
SAMPLES     SYSTEM    02/20/89  C:\DBASE\SAMPLES
SAJATDB               01/04/89  C:\DBASE\SAJATDB
```

Már **létező adatbázist** a **START DATABASE** utasítás aktivizál a következőképpen:

START DATABASE <adatbázisnév>;

Példa:

```
SQL. START DATABASE samples;
```

Ha kiadjuk ezt az utasítást, csak azokra a táblákra és fájlokra hivatkozhatunk, amelyek az aktivált adatbázisban vannak. Egyszerre csak egy adatbázis lehet aktív, így a START utasítás a megnyitással együtt a korábban aktív adatbázist lezárja.

A **DROP** utasítás segítségével **törölhetünk** adatbázist:

DROP DATABASE <adatbázisnév>;

Az utasítás kiadása után jóváhagyást kérő üzenet jelenik meg, ha mégis meggondolnánk magunkat, akkor az **Esc** billentyű lenyomására a törlés elmarad.

A DROP az adatbázis minden alkotóelemét törli, így a táblák, az indexek, az adatbázishoz tartozó rendszertáblák is megsemmisülnek. Az adatbázisra vonatkozó bejegyzést is megszűnik a rendszer-főkatalógusban (SYSDBS.DBF). Egyedül az adatbázis (DOS) alkönyvtára marad meg.

Aktív adatbázis nem törölhető. Ilyenkor előbb használjuk a **STOP DATABASE** utasítást, amely **az aktív adatbázist lezárja.**

A felsorolt utasítások lényegében az adatbázisokkal kapcsolatos minden alapműveletet tartalmaznak. Természetesen több utasítás támogatja az adatbázisokkkal kapcsolatos egyéb funkciókat. Ilyen például az adatbázisok ellenőrzése, a dBASE IV adatfájlok SQL táblákká való átalakítása stb. Ezekről a lehetőségekről később ejtünk szót.

6.2.2. Táblák

A táblák az SQL adatbázisok alap-alkotóelemei, megfelelnek a dBASE adatfájl fogalmának. A táblák szerkezetét a **CREATE TABLE, ALTER TABLE** vagy a DBDEFINE utasításokkal határozhatjuk meg. A **CREATE az új tábla** létrehozására, az **ALTER már meglévő táblák** szerkezetének módosítására szolgál. A DBDEFINE, – amelyről később lesz szó – a híd a dBASE kezelő és az SQL között.

```
CREATE TABLE <táblanév>
        (<oszlopnév> <adattípus>
        [,<oszlopnév> <adattípus> ...] );
```

Ebben az utasításban meg kell adnunk a létrehozandó tábla nevét és zárójelek között a táblát alkotó oszlopokat. Két azonos nevű tábla nem lehet az adatbázisban és a táblán belül az oszlopneveknek is egyedieknek kell lenniük. Az oszlopok az oszlopnév után megadott típusú adatokat tartalmazhatnak. Az **SQL adattípusai** a következők:

Adattípus	Leírás
SMALLINT	Kis egész számok. Legfeljebb 6 számjegyből állhatnak, beleértve az előjelet. A legkisebb szám a -99999, a legnagyobb a 999999.
INTEGER	Egész számok. Legfeljebb 11 számjegy az előjellel együtt. Intervalluma: -9999999999 - 99999999999.
DECIMAL(n,m)	*n* hosszúságu, *m* tizedesjegyű fixpontos decimális számok. A szám hosszába az előjel beleszámít. *n* 1-től 19-ig változhat, *m* 0 és 18 között lehet. Például DECIMAL(5,2) értéke -99.99 és 999.99 között lehet.
NUMERIC(n,m)	*n* hosszúságu, *m* tizedesjegyet tartalmazó fixpontos decimális adattípus. A szám hosszába beleszámít az előjel és a tizedespont is. *n* 1 és 20, *m* pedig 0 és 18 között állhat. Például NUMERIC(5,2) értéke-9.99 és 99.99 lehet.
FLOAT(n,m)	Lebegőpontos számok. Az adat *n* hosszal, *m* tizedesjeggyel jelenik meg. A szám megjelenési hosszába beleszámít az előjel és a tizedespont. Az *n* 1 és 20, az *m* pedig 0 és 18 között állhat. A lebegőpontos számok 0,1E-307 és 0,9E+308 között lehetnek. Ha a szám hagyományos módon nem jeleníthető meg n pozición akkor a kijelzés lebegőpontos formában történik meg. Ha lebegőpontos formában sem jeleníthető meg, akkor helyette csillagokat láthatunk a mezőben.

Adattípus	Leírás
CHAR(n)	Karakteres adat *n* hosszal. *n* 1-től 254-ig terjedhet. Nagy és kis betűket, számjegyeket, írásjeleket (például !,:,., stb.) és általában bármely nyomtatható karaktert tartalmazhat.
DATE	Naptári dátumok. Hossza 8 bájt. A megjelenési formája hó/nap/év, például 12/31/89 illetve 12/31/1989. A megjelenési formát a SET DATE, SET CENTURY, SET MARK utasítások szabályozzák. Dátum típusú adatot a CTOD() függvény illetve a {} zárójel pár segítségével lehet előállítani, például {"12/31/89"}, CTOD("12/31/89"). Értékadásnál ha az évszázad hiányzik, akkor a dBASE IV/SQL automatikusan 19-et szúr be az év elé.
LOGICAL	Logikai (kétértékű) adat. Értékei lehetnek: .T., .t., .Y., .y. (igaz); .F., .f., .N.,.n. (hamis). A mezőbe pont nélkül csak a jellemző betűk kerülnek.

6.2. táblázat

A **táblanév** legfeljebb 8 karakter lehet, betűvel kell kezdődnie és betűket, számokat, aláhúzás-jelet tartalmazhat.Az **oszlopnév** 10 karakterből állhat, a karakterek ugyanolyanok lehetnek, mint a táblanévnél.A tábla oszlopait később törölni, nevét, típusát módosítani nem lehet.

Példa:

Hozzuk létre a SAMPLES adatbázis STAFF táblájának megfelelő szerkezetű saját adat fájlunkat.

```
SQL. CREATE TABLE alkalmzt
     (kodszam CHAR(6),
     vezetknev  CHAR(15),
     kersztnev  CHAR(10),
     fizdatum DATE,
     mhely CHAR(15),
     fonokkod CHAR(6),
     fizetes NUMERIC(6,0));
```

Ebben a példában az utasítás több sorból épül fel. Az ilyen SQL utasításokat a **Ctrl-Home** billenyűk egyidejű lenyomása után gépelhetjük be. A **pontosvesszőt** kötelező kitenni, ez jelzi az **utasítássor végét.**

Amikor létrejön egy SQL tábla, akkor a dBASE IV tulajdonképpen egy dBASE adatfájlt konstruál (.dbf kiterjesztéssel), amely szerkezetileg azzal meg is egyezik. Ebből következően a dBASE IV/SQL táblák legfeljebb 255 oszlopból állhatnak és a sorhossz sem haladhatja meg a 4000 bájtot

A táblákat új oszlopokkal az **ALTER** utasítás segítségével egészíthetjük ki:

```
ALTER TABLE <táblanév>
      ADD ( <oszlopnév> <adattípus>
      [,<oszlopnév> <adattípus> ...] );
```

Példa:

Az előbbi példában az *alkalmzt* táblából kimaradt a *jutalom* oszlop. Addjuk hozzá a táblához a hiányzó oszlopot.

```
SQL. ALTER TABLE alkalmzt
      ADD (jutalom NUMERIC(4,1));
```

Ha nagyobb mérvű módosítást akarunk, akkor új táblát kell létrehozni, és az adatokat át kell másolnunk. Az eredeti táblát szükség szerint törölhetjük. Később láthatunk erre a módszerre is példát.

Táblát a **DROP** utasítással törölhetünk:

```
DROP TABLE <táblanév>;
```

Példa:

```
DROP TABLE regi;
```

Megjegyzés: Ilyen tábla nincs az adatbázisban, a törlést a dBASE IV/SQL vissza fogja utasítani.

6.2.3. Indexek

Minden adatbáziskezelő rendszer legfontosabb célja lehetővé tenni a tárolt adatok gyors visszakeresését. Az indexelés az egyik leghatékonyabb segédeszköz erre.

Az SQL is használ indexeket, de fontos jellemzője, hogy **index nélkül** is képes a kívánt szempont (rendezettség) szerint az adatokat elérni, megjeleníteni. Maga az SQL nyelv nem is rendelkezik olyan utasításokkal amelyek kijelölnék valamelyik index használatát. Ehelyett a szempontokat, a rendezési sorrendet kell meghatároznunk és az SQL **automatikusan kiválasztja** és alkalmazza a létező **indexek közül a szempontjainknak megfelelőt,** illetve ha ilyen nincs akkor a meghatározásnak megfelelően rendezi a tábla sorait.

Az index meggyorsítja a tábla egyes sorainak a megtalálását, ugyanakkor lelassítja a feldolgozást ha a tábla sorait módosítjuk. Ezért ügyeljünk arra, hogy csak a **legszükségesebb szempontokat indexeljük.** Ajánlatos az indexeket törölni, ha a rendezendő tábla kicsi vagy ha már nem az index szerinti sorrendben dolgozzuk fel a tábla sorait.

A tábla törlésekor a táblához tartozó összes index is megsemmisül.

Indexeket csak alaptáblákhoz határozhatunk meg, nézettáblákhoz nem. Nézettáblák esetén a dBASE IV/SQL a nézettáblához tartozó alaptábla indexeit használja az adatelérés meggyorsítására.

SQL-ben indexeket a **CREATE INDEX** utasítással hozhatunk létre:

```
CREATE [UNIQUE] INDEX <indexnév>
    ON <táblanév>
    ( <oszlopnév> [ASC/DESC]
    [,<oszlopnév> [ASC/DESC] ...] );
```

Az utasítás adott adatbázisban az adott táblához létrehozza a meghatározott indexet. Az elsődleges rendezési szempontot az első oszlop határozza meg, az **azon belüli rendezési** szempontot a második oszlop és így tovább. Az ASC kulcsszó határozza meg a növekvő sorrendet, a DESC kulcsszó pedig a csökkenő ellenkező rendezettséget definiál. Az alapértelmezés az ASC. Az utasítássorban csak az egyik rendezési irány szerepelhet, tehát egy utasításon belül az ASC és DESC kulcsszavakat vegyesen nem használjhatjuk. Az INDEX utasításban szereplő oszlopok együttes hossza a 100 karaktert nem lépheti túl. A UNIQUE kulcsszóval a duplikációt akadályozhatjuk meg, azaz sem módosításkor, sem új rekord bevitelekor két azonos indexű (kulcsú) rekord nem keletkezhet. Ha a tábla már az utasítás kiadásakor duplikált sorokat tartalmaz, akkor az indexelést nem tudjuk elvégezni.

A létrehozott SQL indexek megfelelnek a dBASE adatfájlhoz tartozó többszörös indexfájl (.mdx) tagindexének. Egy táblához legfeljebb 47 index tartozhat.

Példa:

Indexeljük a SAMPLES adatbázisban az *alkalmzt* táblánkat *kodszam* szerint.

```
SQL. CREATE UNIQUE INDEX kodindex
       ON alkalmzt
       ( kodszam ASC );
```

Ezután nem vihetünk a táblába két egyforma kódszámú sort.

Feleslegessé vált indexeket a **DROP INDEX** utasítás törli:

```
DROP INDEX <indexnév>;
```

Példa:

```
SQL. DROP INDEX kodindex ;
```

6.3. Másodlagos nevek

A másodlagos név – röviden **másodnév** – az SQL táblák és nézettáblák helyettesítő neve. A másodnevek bármelyik SQL utasításban megjelenhetnek ahol tábla vagy nézettábla szerepelhet. Alkalmazásuk célja elsősorban a táblanevek egyszerűsítése és rövidítése. Másodneveket a **CREATE SYNONYM** utasításokkal adhatunk.

CREATE SYNONYM < másodnév > FOR < táblanév >;

Példa:

Jelöljük ki az *alkalmzt* táblanév mellé az *alk* másodnevet:

SQL. CREATE SYNONYM alk FOR alkalmzt ;

Ezekután *alk* névvel hivatkozhatunk az *alkalmzt* táblára a SELECT, CREATE INDEX, INSERT, UPDATE, DELETE, DROP utasításokban. **Vigyázat!** A másodnevek a DROP TABLE, DROP VIEW utasításokban is használhatók, ami egyet jelent a tábla vagy nézettábla törlésével !

A másodneveket a **DROP SYNONYM** utasítás törli.

DROP SYNONYM < másodnév >;

Példa:

SQL. DROP SYNONYM alk ;

6.4. Táblákon végezhető műveletek

6.4.1. Az utasítások alapeleme: a SELECT

A **SELECT** utasítás az SQL legfontosabb utasítása. A szó fordítása tükrözi is a lényeget: szelektálni, kiválasztani. Segítségével jeleníthetjük meg adatainkat és más utasítások részeként az adatbáziskezelő műveletekben résztvevő adatok körét határozhatjuk meg. A SELECT-tel részletesen a későbbiekben foglalkozunk, itt és most csak a soronkövetkezők könnyebb megértése miatt ejtünk néhány szót róla. Az utasítás szintaktikája:

```
SELECT */<oszlopnév-lista>
        FROM <táblanév-lista>
        [WHERE <feltétel>] ;
```

Példa:

```
SQL. SELECT * FROM staff;
```

A példában szereplő utasítás kiadása után a staff tábla minden sora és minden oszlopa megjelenik a képernyőn. A * azt jelenti, hogy a tábla összes oszlopa részt vesz a műveletben, a hiányzó WHERE <feltétel> pedig azt, hogy minden sort meg kell jeleníteni. A FROM után található névvel a táblát azonosítjuk.

Példa:

```
SQL. SELECT staff_no, lastname
        FROM staff
        WHERE location = "LOS ANGELES" ;
```

Ennél a példánál a *staff* táblából csak a *staff_no* és *lastname* oszlopok jelennek meg azokból a sorokból ahol a *location* oszlop tartalma "LOS ANGELES".

A SELECT más utasítás részeként is alkalmazható, ilyenkor meghatározza, hogy az adott utasítás a tábla mely soraira vonatkozzon.

6.4.2. Sor hozzáadás - Insert

Táblákat új sorokkal az **INSERT** és a LOAD DATA (erről az utasításról a függelékben olvashatunk részletesen) utasítások segítségével bővíthetünk. A hozzáadott sorok lehetnek önálló adatok, de lehetnek más táblák sorai is, így az INSERT utasításnak is két formája van. Az önálló adatokkal történő bővítésre az következő szolgál:

```
INSERT INTO <táblanév>
        [(<oszloplista>)]
        VALUES (<értéklista>);
```

Az új sor tartalmát az értéklistában szereplő konstansok, változók és dBASE IV függvények határozzák meg. Az értékeknek ugyanolyan sorrendben kell szerepelniük, ahogy az oszloplistában az oszlopok (mezőnevek) követik egymást. Azok az oszlopok, amelyek az oszloplistában nem szerepelnek az adattípusuknak megfelelően kapnak értéket a következő táblázat szerint:

típus	érték
CHAR (n)	SPACE(n)
SMALLINT	0
INTEGER	0
DECIMAL()	0
NUMERIC()	0
FLOAT()	0
DATE	{ / / }
LOGICAL	.F.

Ahol:

n	a karakter típusú oszlop hossza.
{, }	a dátum típust jelölő kapcsos-zárójel pár.

6.3. táblázat

Ha úgy adunk a táblához sort, hogy a tábla minden oszlopának (mezőjének) értéket adunk, az oszloplistát el is hagyhatjuk. Ilyenkor természetesen az értékeknek a tábla oszlopsorrendjében kell követni egymást, és az értékek típusának is egyeznie kell az oszlop típusával.

Példa:

Adjunk egy új sort az *alkalmzt* táblához.

```
SQL. INSERT INTO alkalmzt
        VALUES  ("000001","Zambini","Rick",
        {02/15/80},"LOS ANGELES","000000",6000,5.0);
```

Az INSERT utasítás másik változata a két tábla közötti adatmozgatást teszi lehetővé:

```
INSERT INTO <táblanév>
        [(<oszloplista>)]
        <select mondat>;
```

Példa:

Töltsük fel adatokkal az *alkalmzt* nevű táblát az angol megfelelőjéből (*staff* tábla).

```
SQL. INSERT INTO alkalmzt
       SELECT *
       FROM staff
       WHERE lastname < > "Zambini" ;
```

Ez az utasítás hozzáadja az *alkalmzt* táblához a *staff* tábla összes sorát kivéve azt amelyikben a *lastname* oszlop tartalma egyenlő "Zambini"-vel. (A "< >" operátor jelzi a "nem egyenlő" feltételt.) A csillaggal jelezzük azt, hogy a *staff* tábla minden oszlopát figyelembe kell venni az áttöltésnél. Fontos megjegyezni, hogy ebben az esetben a két tábla sorrendben megfelelő oszlopainak típusban, és az oszlopoknak létszámban egyeznie kell.

Az alkalmzt tábla tartalmát a következő utasítással listázhatjuk ki.

```
SQL. SELECT * FROM alkalmzt ;
```

6.4.3. Sorok módosítása - Update

Tábla sorainak a módosítását az **UPDATE** paranccsal tehetjük meg:

```
UPDATE <táblanév>
        SET <oszlopnév> = <kifejezés>
        [,<oszlopnév> = <kifejezés> ...]
        [WHERE <feltétel>];
```

Példa:

Emeljük a vezető beosztású dolgozók fizetését 10 százalékkal.

```
SQL. UPDATE alkalmzt
        SET fizetes = (fizetes * 1.1)
        WHERE kodszam < "000100";
```

6.4.4. Sor törlése - Delete

A **DELETE** utasítás segítségével törölhetők a feleslegessé vált sorok:

```
DELETE FROM <táblanév>
        [WHERE cikkely];
```

Példa:

Egyik alkalmazottunk se szó, se beszéd, fogta a munkakönyvét és kilépett. Töröljük a nyilvántartásból!

```
SQL. DELETE FROM alkalmzt
        WHERE kodszam = "058012";
```

A kilépettet a 058012-es kódszám azonosítja (Megjegyzés: a *kodszam* nevű oszlop karakteres típusú.).

Ha az utasításból kimarad a WHERE kulcsszó és ezzel együtt a törlési feltétel, akkor a tábla összes sora megszűnik, azaz az egész nyilvántartást töröljük! Ehhez a rendszer — figyelmeztető üzenettel — jóváhagyásunkat kéri. Ha meggondoltuk magunkat, akkor az Esc billentyűre a törlés elmarad.

6.4.5. Nézettáblák

Külön utasítás szolgál a nézettáblák, másképpen virtuális táblák létrehozására. A nézettábla nem tartalmaz adatot, ehelyett az alaptáblákban levő sorokra, oszlopokra hivatkozik. A nézettáblában az adat *dinamikusan* megjelenik meg. Ez azt jelenti, hogy az alaptáblában módosuló adat a nézettáblában is módosul és fordítva.

A legegyszerűbb nézettábla egyetlen egy alaptáblára hivatkozik. Kreálhatunk nézettáblát acélból, hogy csökkentsük a megjelenítendő oszlopok számát, vagy azért, hogy szűkítsük sorok halmazát egy megadott feltétel szerint. (Kiszűrjük azokat a sorokat, amelyekre nem vagyunk kíváncsiak.)

Létesíthető nézettábla több táblából is. Ez a fajta nézettábla egy-egy (azonos tulajdonságot jellemző) oszlopon keresztül kapcsol össze különböző táblákat mégpedig úgy, hogy a kapcsoló oszlopban azonos értékekkel rendelkező, de különböző táblákban lévő sorok egy sorban jelennek meg.

Nézettáblát a **CREATE VIEW** utasítás segítségével hozhatunk létre:

```
CREATE VIEW <nézettáblanév> [(<oszloplista>)]
    AS <select mondat>
    [WITH CHECK OPTION];
```

Az utasítás első sorában a nézet nevét és opcionálisan a nézettáblát alkotó oszlopok neveit adhatjuk meg. Az oszlop listát el is hagyhatjuk. Ilyenkor a a nézettáblát felépítő oszlopok nevei megegyeznek a select mondatban meghatározott nevekkel.

Az utasítás második sorában választhatjuk ki a nézettáblát alkotó táblák oszlopait és sorait. A nézettábla szerkezete, tartalma olyan bonyolult lehet, amilyent csak a SELECT utasítással létre tudunk hozni. A nézettáblát úgy is tekinthetjük, mintha egy SELECT utasításban meghatározott kiválasztási feltételeket rögzítenénk. A WITH CHECK OPTION biztosítja, hogy a későbbi adatmódosulások esetén se kerülhessen a nézettáblába olyan sor, amely nem felel meg a SELECT-ben leírt feltételeknek.

Példa:

Alkossunk egy olyan nézettáblát amelyben csak a Los Angeles-ben dolgozó alkalmazottak jelennek meg, azoknak is csak a kódszáma, vezetékneve és fizetése, majd ellenőrizzük az eredményt:

```
SQL. CREATE view losang
     AS SELECT kodszam, vezetknev,fizetes
     FROM alkalmzt
     WHERE mhely = "LOS ANGELES";

SQL. SELECT * FROM losang;
```

KODSZAM	VEZETKNEV	FIZETES
000003	Vidoni	5780
000006	Thomas	5875
000011	Michaels	4927
000012	Charles	5945
000015	Roddick	5493
000016	Long	5190
000020	Sanders	3783

A CREATE VIEW utasítással még **alaptáblák szerkezetét** is **átalakíthatjuk**.(Persze csak látszólag.)

Példa:

Ha az alkalmazott nevét egy oszlopban szeretnénk látni és a várható éves fietésre vagyunk kíváncsiak akkor begépelhetjük a következő utasítást:

```
SQL. CREATE VIEW evesfiz
     (kodszam, nev, evesfizu)
     AS SELECT kodszam, vezetknev + kersztnev, fizetes*12
     FROM alkalmzt ;
```

A **nézettáblák** alkalmasak két vagy **több tábla összekapcsolására** is.

Példa:

Hozzunk létre egy nézettáblát amelyben szerepel a rendelés szám, az eladó neve, és az eladási dátum. A dBASE IV/SQL-hez mellékelt példa adatbázisban a *sales* táblában vannak az eladással kapcsolatos adatok, de ebben a táblában nem szerepel az eladó neve.

```
SQL. CREATE VIEW
     (rend_szam, elad_dat, elado)
     AS SELECT order_no, sale_date, vezetknev
     FROM sales, alkalmzt
     WHERE staff_no = kodszam;
```

Ebben a példában az eladások táblában (*sales*) levő *staff_no* oszlop határozza meg az eladó kódszámat. A létrehozott nézettáblában a WHERE cikkelyben szereplő feltétel a válogató szempont, ez biztosítja, hogy a nézettáblában a megfelelő eladó neve jelenjen meg.

6.5. Lekérdezés - SELECT alapfokon

6.5.1. A SELECT utasítás

A teljes SELECT utasítás a következőképpen néz ki:

```
SELECT [ALL/DISTINCT] {*/<kifejezés-lista>}
    [INTO <változólista>]
    FROM <táblalista>
    [WHERE <feltétel>]
    [GROUP BY <oszloplista>
        [HAVING <feltétel>]]
    [UNION <select mondat>]...
    [ORDER BY <oszlopnév>/<oszlopszám> [ASC/DESC]
        [,<oszlopnév>/<oszlopszám> [ASC/DESC]...]]
    [FOR UPDATE OF <oszloplista>]]
    [SAVE TO TEMP <táblanév> [(<oszloplista>)] [KEEP]]
```

Mielőtt részleteiben megismernénk a paramétereket, címszavakban rövid áttekintést adunk a lényegesebb cikkelyekről.

INTO	Ha csak egy sort választunk ki, tartalmát változókba tölthetjük.
FROM	A résztvevő táblákat határozhatjuk meg.
WHERE	Feltételekkel kiválaszthatjuk az eredménytáblában megjelenő sorokat.
GROUP BY	A megadott oszlopok szerint duplikált sorokat vonja össze.
HAVING	Ugyanaz, mint a WHERE, de az összevont sorokra vonatkozó feltétel.
UNION	Két vagy több tábla összevonása, összemásolása
ORDER BY	Az eredménytáblát a megadott oszlopok szerint rendezi.
FOR UPDATE OF	Meghatározhatjuk melyik oszlopok módosíthatók (a felhasználó által).
SAVE TO TEMP	A SELECT-tel meghatározott táblát elmenthetjük ideiglenesen vagy véglegesen (KEEP) egy fájlba.

Az **INTO** és a **FOR UPDATE OF csak programban** használható.

A SELECT utasítást több más utasítás részeként is használhatjuk, hiszen ez választja ki számunkra a kívánt adatokat és rendezettséget. Ha az utasítást a *CREATE VIEW*, *INSERT*, *DELETE*, *UPDATE*, *WHERE* vagy *UNION* részeként használjuk **nem tartalmazhatja** az **INTO, UNION, ORDER BY, FOR UPDATE OF, SAVE TO TEMP** cikkelyeket.

6.5.2. Oszlopok és sorok kiválasztása

Alapvetően a SELECT utasítás a FROM után szereplő táblákra vonatkozik, azaz a SELECT után álló paraméterek segítségével felhasználandó oszlopokat ezekből a táblákból választhatjuk ki. Az itt felsorolt oszlopok vesznek részt a SELECT utasítás kimenő adataiban.

Példa:

```
SQL. SELECT kodszam, vezetknev, mhely
     FROM alkalmzt ;
```

Ha egy tábla minden oszlopára szükség van akkor az összes oszlop felsorolása helyett egy csillagot is tehetünk.

Példa:

```
SQL. SELECT * FROM alkalmzt ;
```

Nagy táblák megjelenítésénél a listázást a Ctrl-S billentyűk lenyomásával lehet felfüggeszteni illetve továbbfolytatni. A listázás automatikusan megszakad képernyőnként, ha a SET PAUSE ON utasítást kiadjuk. A SET HEADINGS OFF paranccsal a fejléc megjelenítését lehet kikapcsolni.

ALL, DISTINCT

Az **ALL** és **DISTINCT** opciók a SELECT-ben résztvevő sorokat határozzák meg. ALL esetében, – ez az alapértelmezés, – minden sor megjelenik. DISTINCT-nél a táblában levő duplikált sorokból csak az egyik szerepel, jelenik meg.

Példa:

```
SQL. SELECT ALL part_no, descript
     FROM inventry ;

SQL. SELECT DISTINCT part_no, descript
     FROM inventry ;
```

A második esetben a 001038, 001007 , stb. sorszámmal rendelkező sorok csak egy példányban jelennek meg a képernyőn.

WHERE

A megfelelő sorok további kiválasztására szolgál a **WHERE** kulcsszó után megadható feltétel. Ez a forrástábla (táblák) minden sorára kiszámítja a feltételt, és csak azokat jeleníti meg, amelyeknél ez a feltétel igaznak bizonyul

Példa:

```
SQL. SELECT *
     FROM alkalmzt
     WHERE mhely = "NEW YORK";
```

A feltétel tetszőlegesen bonyolult lehet, amelyet relációs operátorok segítségével adhatunk meg. A dBASE IV/SQL a következő relációs operátorokat ismeri:

operátor	leírás
=	Egyenlőség jel
<	Kisebb mint
	Nagyobb mint
<=	Kisebb vagy egyenlő
=	Nagyobb vagy egyenlő
<>	Nem egyenlő
! <	Nagyobb vagy egyenlő
! >	Kisebb vagy egyenlő
! =	Nem egyenlő
#	Nem egyenlő

6.4. táblázat

Az össszehasonlított értékeknek nem kell szigorúan azonos adattípusúaknak lenni, de a "típuskompatibilitás" elengedhetetlen. Így összehasonlíthatunk különböző numerikus típusú adatot (pl: SMALLINT, FLOAT), de numerikust és dátum típusut már nem. Azt is meg kell jegyeznünk, hogy karakteres típus esetén a kis- és nagybetűk nem tekinthetők azonosnak.

Lehetőség van a különböző feltételek variálására. A feltételeket összekapcsoló logikai operátorokat (műveleti jeleket) a következő táblázatban láthatjuk:

operátor	leírás
NOT	*Tagadás*. A mögötte álló logikai kifejezés értékének ellentéte. Igaz = NOT Hamis, Hamis = NOT Igaz.
AND	*És*. Igaz értéket eredményez ha a tőle balra és jobbra eső logikai kifejezések mindegyikének értéke igaz. Igaz = Igaz AND Igaz, Hamis = Hamis AND Igaz.
OR	*Vagy*. Igaz értéket eredményez ha a tőle balra és jobbra eső logikai kifejezések közül valamelyik értéke igaz. Igaz = Hamis OR Igaz, Igaz = Igaz OR Hamis, Hamis = Hamis OR Hamis, Igaz = Igaz OR Igaz.

6.5. táblázat

Példa:

```
SQL. SELECT *
       FROM alkalmzt
       WHERE NOT kodszam = "000001"
             OR fizetes > 5500 AND jutalom < 5;
```

A logikai operátorok nem egyenrangúak, a rangsorrend közöttük csökkenő sorrendben: NOT, AND, OR. A kiértékelésük is rangsorrendben és nem az elhelyezkedésük sorrendjében történik. Így a példánkban azok kerülnek kiválasztásra, akiknek a kódszámuk nem 000001 vagy a fizetésük nagyobb 5500-nál és a jutalmuk (viszont) kisebb 5-nél (bármilyen is a kódszámuk!).

Ettől eltérő sorrendet a feltételek zárójelezésével írhatunk elő.

Példa:

```
SQL. SELECT *
       FROM alkalmzt
       WHERE NOT ( (kodszam = "000001" OR fizetes > 5500 )
             AND jutalom < 5 );
```

Ekkor először a zárójelen belüli logikai kifejezés kerül kiértékelésre, majd csak ezután a többi, azaz most azokat válogatjuk ki, akiknek a jutalmuk nem kisebb 5-nél, a kódszámuk semmiképp sem 000001, de a fizetésük sem nagyobb 5500-nál.

6.5.3. Kifejezések

Az SQL kifejezések hasonlóak a dBASE IV-ben megszokottakhoz. Egy-egy kifejezés oszlopból, konstansból, változóból, és függvényből épülhet fel. A kifejezés alkotóelemei között álló aritmetikai operátorok (matematikai műveleti jelek) határozzák meg az aritmetikai (matematikai) műveleteket. Ezek a következők lehetnek:

operátor	leírás
**,^	hatványozás
+	pozitív előjel
-	negatív előjel
*	szorzás
/	osztás
+	összeadás (numerikus), összekapcsolás (karakter)
-	kivonás (numerikus), összekapcsolás (karakter)

6.6. táblázat

A táblázatban a műveleteket rangsorrendben (precedencia) tüntettük fel, tehát végrehajtásra is ilyen sorrendben kerülnek. Akár csak a logikai operátoroknál, itt is lehetséges a zárójelek használata a számítási sorrend megváltoztatására. Ezek a kifejezések állhatnak a SELECT, a WHERE és a HAVING kulcsszó után.

Az oszloplistában nemcsak oszlopnevek, hanem kifejezések is szerepelhetnek. Ezeket a kifejezéseket **logikai oszlopnak vagy számított oszlopnak** nevezzük. Ezek az oszlopok semmilyen alaptáblában sem szerepelnek, de többnyire létező oszlopokra hivatkoznak.

Példa:

```
SQL. SELECT kodszam, vezetknev, kersztnev,
            fizetes*12, " - éves fizetés -"
     FROM alkalmzt ;
```

KODSZAM	VEZETKNEV	KERSZTNEV	EXP1	EXP2
000001	Zambini	Rick	72000	- éves fizetés -
000003	Vidoni	Cheryl	69360	- éves fizetés -
000004	Coudray	Sandy	74844	- éves fizetés -
000006	Thomas	Pat	70500	- éves fizetés -
000008	McLester	Debbie	57504	- éves fizetés -
000011	Michaels	Delores	59124	- éves fizetés -
000012	Charles	Ted	71340	- éves fizetés -
000013	Marin	Mark	57624	- éves fizetés -
000015	Roddick	Mary	65916	- éves fizetés -
000016	Long	Nicole	62280	- éves fizetés -
000019	Rolfes	Chuck	55032	- éves fizetés -
000020	Sanders	Kathy	45396	- éves fizetés -

6.5.4. SQL függvények

A SELECT kulcsszó után az oszloplistában állhatnak dBASE IV függvények is kifejezés gyanánt. Természetesen itt is érvényesek némi megszorítások, amelyekről később összefoglaló listát adunk. A hagyományos függvények az SQL-ben kiegészültek ún. **gyűjtő függvényekkel**. A gyűjtő függvény argumentuma az oszlop.

Az SQL függvényekkel a SELECT utasítással kiválasztott sorok megadott oszlopának az összegét, átlagát, maximumát, minimumát , illetve létszámát számíthatjuk ki. A gyűjtő függvények csak a SELECT paranccsal együtt használhatók, attól függetlenül, önállóan nem.
A függvény argumentumában az oszlop neve előtt állhat DISTINCT illetve az ALL kulcsszó. Az ALL az alapértelmezés, és azt jelenti, hogy a WHERE cikkellyel kiválasztott sorok mindegyike résztvesz a műveletben.
A DISTINCT azt erdeményezi, hogy a duplikált értékeket tartalmazó sorok csak egyszer számítanak a függvény végértékének kiszámításánál. Ez az paraméter ha-

sonlóan hat, mint a SELECT kulcsszó után megjelenő ALL, DISTINCT kulcsszavak.

Példa:

Az *alkalmzt* tábla ismételt sorokat tartalmaz. Irassuk ki a tábla összes sorának számát és az egyedi sorok számát!

```
SQL. SELECT COUNT(mhely)
     FROM alkalmzt  ;

     COUNT1
          12

SQL. SELECT COUNT( DISTINCT mhely)
     FROM alkalmzt ;

     COUNT1
          3
```

Az első utasítás eredménye a cégnél dolgozó alaklmazottak száma lesz, mert az összes sort megszámolja. A második utasítás azt mutatja meg, hogy a cég dolgozói hány helyen dolgoznak.
Írassuk ki még a dolgozók fizetésének minimumát, maximumát, átlagát és teljes mennyiségét is.

```
SQL. SELECT MIN (fizetes), MAX (fizetes), AVG (fizetes),
            SUM (fizetes)
     FROM alkalmzt ;

     MIN1  MAX2    AVG3      SUM4
     3783   62375284.17      63410
```

gyűjtő függvény	leírás
COUNT()	A kiválsztott sorok számát adja vissza.
SUM()	Numerikus oszlop összegét képzi.
MIN()	Karakter, dátum vagy numerikus oszlop minimális értékét szolgáltatja.
MAX()	Karakter, dátum vagy numerikus oszlop maximális értékét szolgáltatja.
AVG()	Numerikus oszlop átlagértékét számítja ki.

6.7. táblázat

A **SUM, MAX, MIN, AVG** függvény paramétereként olyan kifejezés is szerepelhet, amelyben oszlopra hivatkozunk.

A COUNT függvénynél a * is szerepelhet paraméterként. A * itt az ALL <oszlopnév> helyett áll, és azt jelenti, hogy az összes sort figyelembe kell venni.

A MIN és MAX függvények karakter típusú oszlopokat az ASCII kódértékük alapján hasonlítják össze. Ez azt jelenti, hogy a nagy betűk nem egyenlőek a kis betűkkel, és a nagy betűk megelőzik a kis betűket.

A gyűjtő függvények és az oszlopok nem szerepelhetnek vegyesen a SELECT utasításban. Ez alól a szabály alól kivétel, ha a SELECT utasítás GROUP BY cikkelyt is tartalmaz.

Ennek a technikának a leírását lásd később.

6.5.5. BEETWEN, IN, LIKE

Az SQL a feltételek meghatározásának megkönnyítésére három speciális operátor használatát engedélyezi a WHERE és a HAVING cikkelyekben. Ezek a kulcsszavak a BETWEEN, IN és LIKE, mind a három operátor bonyolult kifejezések egyszerűsítésére szolgál.

BETWEEN	megvizsgálja, hogy egy érték a megadott intervallumban fekszik e.
IN	megvizsgálja, hogy egy érték a felsorolt listában listában szerepel-e.
LIKE	Karakter sztringet hasonlít egy megadott mintához.

A BETWEEN szintaktikája:

<kifejezés> BETWEEN <alsó határ> AND <felső határ>

amely a következő hagyományos feltétellel azonos:

<kifejezés> >= <alsó határ> AND <kifejezés> <= <felső határ>

A BETWEEN numerikus, karakteres, és dátum típusú kifejezések összehasonlítására használható. A hagyományos módon leírt változatból kiderül, hogy a BETWEEN igaz értéket ad vissza akkor is ha a kifejezés egyenlő valamelyik határértékkel. Szokásosan a kifejezés helyén oszlop áll, míg a határok helyett egy-egy konstans.

Példa:

```
SQL. SELECT *
       FROM alkalmzt
       WHERE fizetes BETWEEN 4000 AND 5000;
```

Az IN szintaktikája:

< kifejezés > IN (< értéklista >)

amely a következő hagyományos feltétellel azonos:

< kifejezés > = < érték1 > OR < kifejezés > = < érték2 > OR
... OR < kifejezés > = < értékN >

Az értéklistát konstansok, dBASE IV függvények és változók alkothatják. A kifejezés helyén rendszerint egy oszlop áll, míg az érték listában állandók szerepelnek.

Példa:

```
SQL. SELECT *
       FROM alkalmzt
       WHERE mhely IN ("LOS ANGELES","CHICAGO");
```

A LIKE operátor karakteres értékek öszehasonlítására szolgál. A hagyományos egyenlőségvizsgálatnál azonban tágabb lehetőségekkel rendelkezik.

A LIKE szintaktikája:

< karakteres oszlop > LIKE < minta >

ahol a *minta* karakter-konstans, változó, USER dBASE IV/SQL kulcsszó lehet. A mintába speciális ún. "joker" karaktereket is tehetünk. Az aláhúzás jel "_" helyén a keresendő sztringben bármilyen karakter állhat. A százálék jel "%" több tetszőleges karaktert heylettesít.

```
SQL. SELECT vezetknev, kersztnev
       FROM alkalmzt
       WHERE vezetknev LIKE "M%";

SQL. SELECT *
       FROM alkalmzt
       WHERE lakhely LIKE  "____A%";
```

Az első példában az utasítás azokat az alkalmazottakat listázza ki, amelyek neve M betűvel kezdődik, míg a második azokat, akiknek a munkahelye nevében az ötödik karakter A.

Megjegyzés: az A betű előtt négy aláhúzás karakter áll.

6.5.6. Rendezés

A megelőző példákban a tábla sorai abban a sorrendben jelentek meg a képernyőn, ahogy azokat az adatbevitelkor a táblába kerültek. A sorrend, rendezettség megváltoztatására szolgál az ORDER BY cikkely.

```
ORDER BY <oszlopnév/oszlopszám> [ASC/DESC]
    [,<oszlopnév/oszlopszám> [ASC/DESC]...];
```

A SELECT utasítás eredményadatait az ORDER BY kulcsszó után szereplő oszlopok szerinti rendezettségben kapjuk. Azoknak az oszlopoknak, amely szerint a táblát rendezni akarjuk, feltétlenül szerepelniük kell a SELECT kulcsszó után az oszloplistában. **Számított oszlopot nem lehet rendezni.** Az oszlopra hivatkozhatunk a neve helyett a számával is. Az oszlopszám az oszlopnak a SELECT-ben elfoglalt helyét jelzi.

Az **ASC**, – ez az alapértelmezés, – határozza meg a növekvő sorrendet, a **DESC** a csökkenőt. Az ORDER BY cikkelyben több oszlop is kombinálható egymással, és a csökkenő és növekvő sorrend vegyes alkalmazása is lehetséges.

A rendezési kulcsként megadott oszlopok összes hossza nem haladhatja meg a 100 karaktert.

A rendezettséget kifejezetten csak az ORDER BY cikkely határozza meg, így ha vannak is indexeink a táblához, azok a sorrendet nem befolyásolják. Az **indexek** mégis lényeges **hatással vannak** a rendezésre, egészen pontosan **a rendezés sebességére.** Ha az ORDER BY-nál előírt rendezési szempont megegyezik valamelyik létező index-szel, akkor a rendezés az index felhasználásával történik meg.

Például :

```
SQL. SELECT *
       FROM alkalmzt
       ORDER BY mhely, vezetknev;

SQL. SELECT fizetes, vezetknev + keresztnev, kodszam
       FROM alkalmzt
       ORDER BY fizetes DESC, 2 ASC ;
```

Az első példában a sorok munkahely és azon belül vezetéknév szerint növekvő sorrendben fognak megjelenni. A második példában a sorok fizetes szerint csökkenő, azon belül teljes név szerint növekvő rendezettség szerint fognak a képernyőre kerülni.

6.5.7. Sorok összevonása

Az SQL lehetőséget ad a tábla sorainak csoportosítására. A létrehozott csoportokon a gyűjtőfüggvények segítségével különböző műveleteket lehet végrehajtani. Ilyenkor a függvények hatásköre nem terjed ki az egész táblára, hanem csak az egyes csoportokra. A **GROUP BY** cikkely használatával gyűjthetjük a sorokat csoportokba, mégpedig azokat, amelyekben a kiválasztott oszlopok azonos értékkel rendelkeznek. A SELECT utasítás kimenetében minden csoportot egy-egy sor fog képviselni. Az eredménysor egy eleme (oszlopa) vagy egy csoportot azonosít – azaz a közös értéket mutatja –, vagy a csoporton a gyűjtőfüggvényekkel végrehajtott műveletek eredményét tartalmazza.

GROUP BY <oszloplista> [HAVING <feltétel>]

Sok megszorítás vonatkozik a GROUP BY használatára. Csak olyan oszlopok szerint lehet a táblát csoportosítani, amelyek a select mondatban szerepelnek. Ez fordítva is igaz, a SELECT kulcsszó után szereplő oszlopoknak a GROUP BY cikkelyben is meg kell jelenniük.
A SELECT után az oszlopok között, a gyűjtő függvényeken kívül, nem szerepelhet olyan kifejezés, amely a csoportosításból kimaradt oszlopokkal valamilyen kapcsolatban van.
A csoportosítás csak rendezett oszlopokon történhet, ezért a GROUP BY kijelölése előtt a táblát a csoportosító szempontok szerint valamilyen módon – kevés kivétellel az ORDER BY cikkely használatával – rendezni kell.

Példa:

Irassuk ki, hogy a cég alkalmazottai milyen városokban dolgoznak, és azt, hogy egy városban hányan vannak.

```
SQL. SELECT mhely, COUNT(*)
       FROM alkalmzt
       GROUP BY mhely;
```

Nézzük meg azt is, hogy a cég dolgozói milyen jutalom-osztályokba vannak sorolva, egy osztályba hány ember tartozik, és mennyi az egy osztályba tartozók átlag fizetése.

```
SQL. SELECT jutalom, COUNT(*), AVG(fizetes)
       FROM alkalmzt
       GROUP BY jutalom
       ORDER BY 1 DESC ;
```

A HAVING cikkely a GROUP BY által létrehozott csoportokat szűri meg. Funkciója azonos a WHERE-évei, de amíg a WHERE az egész táblára vonatkozik, addig a HAVING csak az egyes csoportokra.

Példa:

Bonyolítsuk tovább az előző példát, listázzuk ki azokat a jutalmi csoportokat ahol az átlag fizetés nagyobb mint 5000 dollár.

```
SQL. SELECT jutalom, COUNT(*), AVG(fizetes)
       FROM alkalmzt
       GROUP BY jutalom
              HAVING AVG(fizetes) > 5000
       ORDER BY 1 DESC ;
```

6.6. Lekérdezés több táblából - SELECT haladóknak

6.6.1. Táblák összemásolása

Több táblát az **UNION** cikkellyel másolhatunk össze. Természetesen az egyesítendő táblák között kompatibilitásnak kell lenni.

UNION <select mondat> [UNION <select mondat>...]

A SELECT-ekben kijelölt **oszlopok létszámának** meg kell **egyezniük**, és az egymásnak megfelelő oszlopoknak **azonos típusúaknak** és azonos hosszúaknak kell lenniük. **Numerikus típusoknál** a hossz mellett a **tizedesjegyek számának** is **egyezni** kell. A **fixpontos és a lebegőpontos** típusú oszlopok **nem kompatibilisek** egymással. A különböző numerikus típusok az előző megkötésekkel vegyesen is használhatók.
Az UNION-hoz tartozó SELECT mondat **nem tartalmazhatja** az *INTO, ORDER BY, FOR UPDATE OF, SAVE TO TEMP* cikkelyeket, de magát az *UNION* cikkelyt sem.

Ez áthidalható, ugyanis a legmagasabb szintű (legkülső) SELECT több UNION kulcsszóval is rendelkezhet.

További megkötés, hogy az ORDER BY kulcsszónál **oszlopra** való **hivatkozásként** csak **egész szám** alkalmazható.
Az esetleges **ismétlődő sorok kimaradnak** a műveletből, így az utasítás eredménye mentes lesz a duplikált soroktól.

Példa:

Példaként listázzuk ki az adatbázisunkban levő összes várost. A helység neveket a *customer*, *staff*, *inventry* táblák rendre a *city*, *location* és *location* nevű oszlopokban tartalmazzák.

```
SQL. SELECT city
     FROM customer
          UNION
     SELECT location
          FROM staff
          UNION
     SELECT location
          FROM inventry
     ORDER BY 1;
```

6.6.2. Kapcsolat táblák között

Az eddigi fejezetekben áttekintettük a SELECT utasítás használatát egyedi táblák esetén. A kívánt infomáció azonban leggyakrabban több táblában található. A SELECT utasítás a legtermészetesebb módot biztosítja a táblákban lévő adatok elérésére. Két tábla esetén egyszerűen csak fel kell sorolni a FROM cikkelyben a táblákat.

```
SQL. SELECT *
     FROM staff, sales ;
```

Ekkor a SELECT utasítás eredményeként egy olyan szerkezet jön létre, amelyben az első tábla oszlopai után kerülnek a második tábla oszlopai, az erdményben (kimenetben) pedig az első tábla minden sora a második tábla minden sorával együtt (kombinálva) jelentkezik. A két tábla kombinációját a WHERE cikkelyben megadható feltétellel lehet szabályozni.

Példa:

Példa adatbázisunkban a *staff* a dolgozókat nyilvántartó tábla, *sales* pedig az eladásokat tartalmazza. Mindkét táblában van egy *staff_no* nevű oszlop, amely a dolgozók kódszámát tartalmazza. Módosítsuk az előbbi példát!

```
SQL. SELECT order_no, staff.staff_no, sales.staff_no, lastname
     FROM staff, sales
     WHERE sales.staff_no = staff.staff_no ;
```

A képernyőn a megrendelés sorszáma, az ügyintéző kódszáma, és az ügyintéző neve jelenik meg. Láthatjuk, hogy a WHERE kulcsszó után megadott feltétel hatására csak azok a sorok kapcsolódnak össze, ahol a dolgozó kódszáma mind a két táblában megegyezik. Meggyőződhetünk arról is, hogy az összekapcsolás ténylegesen eleget tesz a feltételnek. Ha a táblákban **azonos nevű oszlopok** vannak,

akkor megkülönböztetésül az oszlop neve elé ponttal elválasztva ki kell tenni a **tábla nevét,** amelyhez tartozik.

Több tábla összekapcsolása esetén ugyan ilyen elvek alapján kell eljárni, a FROM cikkelyben tetszőleges számú tábla felsorolható.

A táblák összeskapcsolásánal nem csak egyenlőséget tartalmazó feltételt határozhatunk meg. Lehetőség van a kisebb és nagyobb (<,>) operátorok, vagy bármilyen egyéb összetett feltétel megfogalmazására is.

Példa:

Listázzuk azokat a helyeket, ahol a raktáron álló áru mennyisége kielégítheti a 020021-es rendelést!

```
SQL. SELECT items.part_no, qty, on_hand, location
       FROM items, inventry
       WHERE items.part_no = inventry.part_no AND
             on_hand > qty AND items.order_no = "020021" ;
```

ITEMS->PART_NO	ITEMS->QTY	IINVENTRY->ON_HAND	INVENTRY->LOCATION
001013	8	157	CHICAGO
001013	8	35	LOS ANGELES
001025	8	63	LOS ANGELES
001025	8	47	NEW YORK
001024	6	140	CHICAGO
001024	6	56	NEW YORK

Magyarázat:

inventry	Készlet-tábla.
items	Az áruk táblája, a megrendelt áruk kódját és menyiségét tartalmazza.
part_no	A rendelt áru kódszáma.
qty	Rendelt mennyiség.
location	A raktár helyének megnevezése.
on_hand	Raktáron levő készlet.
order_no	Rendelésszám.

A GROUP BY, ORDER BY, WHERE, gyűjtő függvények és az előző fejezetek egyéb lehetőségei ugyanúgy alkalmazhatók összekapcsolt tábláknál, mint egyedi táblák esetében.

6.6.3. Tábla összekapcsolása önmagával

A dBASE IV/SQL megengedi táblák önmagukkal való összekapcsolását, létrehozását. Erre akkor lehet szükség, ha az egyes sorokban levő információt ugyan abban a táblában, de a tábla más sorában található adattal kell kiegészíteni. Ek-

kor a SELECT utasításban egyetlen tábla, de az két vagy több példányban, fog megjelenni a FROM cikkelyben. A SELECT ilyen alkalmazása felveti az oszlopok megjelölésének kérdését. A korábbi fejezetekben láthattuk, hogy a kétértelmű oszlopneveket a tábla nevével kiegészítve különböztethetjük meg. Ez most nem segíthet, hiszen a tábla neve is ugyanaz lesz. A probléma megoldását a **másodnevek** alkalmazása jelenti. (Ez a másodlagos név **nem keverendő össze a CREATE SYNONYM** utasítással létrehozható **másodlagos nevekkel.**). A FROM cikkelyben a tábla neve után space-szel elválasztva kell állni a másodnévnek. A másodnév az **A, B, C, D, E, F, G, H, I, J** betűk **kivételével** bármilyen név lehet. Ekkor a másodnév fogja helyettesíteni a tábla nevét az oszlopok megkülönböztető jelzéseként.

Példa:

Listázzuk ki azokat az alkalmazottakat és velük együtt a főnökük nevét!

```
SQL. SELECT beoszt.vezetknev, beoszt.kersztnev,"főnöke:",
            fonok.vezetknev
     FROM alkalmzt beoszt,alkalmzt fonok
     WHERE beoszt.fonokkod=fonok.kodszam
     ORDER BY 1, 2;
```

6.6.4. SELECT a SELECT-ben

A feldolgozás során sokszor előadódhat olyan helyzet, hogy a tábla sorának kiválasztását – bármilyen bonyolult feltételrendszert konstruálunk is – nem lehet megoldani, csak akkor, ha egy másik SELECT utasítást is alkalmazhatunk. A dBASE IV/SQL lehetőséget nyújt erre is. A WHERE vagy a HAVING cikkelyben a kiválasztó-feltétel megkonstruálásakor használhatunk további SELECT utasításokat.
A beágyazott SELECT utasítást **belső SELECT**-nek, míg a másikat **külső SELECT**-nek nevezzük. A külső SELECT a belső SELECT által szolgáltatott értékektől függően készíti el a kimenetét.
A belső SELECT utasítás szokásosan egy oszlopból áll.

A külső SELECT utasítást más- és másféleképpen kell felépíteni attól függően, hogy a belső SELECT hány értéket eredményez. Két eset lehetséges:

- A belső SELECT egy értéket szolgáltat.
- A belső SELECT több értéket ad vissza.

Egyszerű az eset, ha a **belső SELECT egy értéket** produkál. Ilyenkor a külső SELECT WHERE (vagy HAVING) cikkelyében a belső SELECT úgy viselkedik mint bármilyen más feltétel-kifejezés, amely ezekben az cikkelyekben alkalmazható. A belső SELECT utasítást megkülönböztetésül – mint egy kifejezést – mindig zárójelek közé kell tenni.

Példa:

Például irassuk ki azokat a dolgozókat akiknek a fizetése az átlagot meghaladja.

```
SQL. SELECT vezetknev, fizetes
     FROM alkalmzt
     WHERE fizetes >
          (SELECT AVG(fizetes)
          FROM alkalmzt)
     ORDER BY vezetknev ;
```

A hagyományos kifejezésekhez hasonlóan, **tetszőleges relációs operátorok** használhatók a belső SELECT utasításokkal kapcsolatban.

Példa:

```
SQL. SELECT order_no, sale_date
     FROM sales
     WHERE cust_no =
          (SELECT cust_no
          FROM customer
          WHERE company = "American Business Supply")
     OR staff_no =
          (SELECT kodszam
          FROM alkalmzt
          WHERE vezetknev = "Zambini") ;
```

Magyarázat:

order_no	Rendelés szám
sale_date	Dátum
sales	Eladások táblája
cust_no	Megrendelő kódja
customer	Megrendelők táblája
company	Megrendelő cég neve
staff_no	Alkalmazott kódszáma

Ebben a példában azok megrendelések kerülnek kilistázásra, ahol a megrendelő az American Business Supply vagy az eladót Zambini-nek hívják.

Több értéket szolgáltató **belső SELECT**-k esetén bonyolultabb a helyzet. Ilyen esetekben az IN, ANY, ALL kulcsszavak segítségével lehet feltételeket konstruálni a beágyazott SELECT utasításokból.

Az IN működése megfelel korábban leírtaknak azzal a kiegészítéssel, hogy az értéklista helyett szerepelhet SELECT utasítás is. Ilyenkor az IN azt vizsgálja, hogy a megadott érték szerepel-e a SELECT által szolgáltatott értékek között.

Példa:

Példaként listázzuk ki azokat az eladásokat amelyeket los angelesi vagy chicagói alkalmazottak végeztek.

```
SQL. SELECT order_no, sale_date
     FROM sales
     WHERE staff_no IN
          (SELECT kodszam
          FROM alkalmzt
          WHERE mhely IN ("LOS ANGELES","CHICAGO")
     ORDER BY 2 DESC;
```

Az **ANY** és az **ALL** kulcsszavak mindig olyan relációs operátorokkal állnak, amelyek belső SELECT utasításra vonatkoznak, és bizonyos értelemben megsokszorozzák a relációs operátorok hatását. A következő szimbólumok alakalmazhatók műveleti jelként: =, <, >, <=, >=, <>, !<, !>, !=, #.

Ha az **ANY** kulcsszó szerepel, akkor eredményünk már akkor is **igaz** lesz, **ha** a **hasonlítás** a belső SELECT által szolgáltatott legalább (**akármelyik**) egyetlenegy **értékkel** is **igaz**. Hamis értéket csak akkor kapunk, ha a belső-SELECT értékkel való hasonlítás egyetlenegy esetben sem volt igaz.

Az **ALL** kulcsszónál az eredmény csak akkor lesz **igaz, ha** relációs operátor előtt álló érték a SELECT által szolgáltatott értékek összehasonlítása **mindegyik esetben igaz.**

Példa:

Listázzuk ki azokat a dolgozókat akiknek kisebb a jutalékuk akármelyik chicago-i dolgozónál.

```
SQL. SELECT kodszam, vezetknev, mhely, jutalom
     FROM alkalmzt
     WHERE jutalom < ANY
          (SELECT jutalom
          FROM alkalmzt
          WHERE mhely="CHICAGO")
     ORDER BY 1 ASC;
```

Most nézzük meg azokat a dolgozókat, akiknek nagyobb a jutalékuk minden chicago-i alkalmazottnál.

```
SQL. SELECT kodszam, vezetknev, mhely, jutalom
     FROM alkalmzt
     WHERE jutalom > ALL
          (SELECT jutalom
          FROM alkalmzt
          WHERE mhely="CHICAGO")
     ORDER BY 1 ASC;
```

A feldolgozás során előadódhat olyan helyzet, amikor nem az a feltétele egy tábla sorai megjelenítésének, hogy más SELECT utasítás milyen eredményt szolgáltat, hanem az, hogy egyáltalán van-e eredménye a beágyazott SELECT utasításnak. Ilyen problémák áthidalására, az **EXISTS** – magyarul: létezik – kulcsszó szolgál.
Az eredmény igaz lesz, ha a belső SELECT legalább egyetlen értéket szolgáltat, és hamis, ha egyet sem.

Példa:

Listázzuk ki azokat az üzletfeleket, akiknek legalább egy megrendelése van a cégnél.

```
SQL. SELECT cust_no, company
     FROM customer
     WHERE EXISTS
          ( SELECT *
          FROM sales
          WHERE customer.cust_no = sales.cust_no)
     ORDER BY 1 ASC ;
```

Magyarázat:

cust_no	Üzletfél azonosító
company	Cég neve
customer	Üzletfeleket tartalmazó tábla
sales	Eladásokat tartalmazó tábla

Természetesen egy SELECT utasításban több EXISTS is szerepelhet és a feltételt meg is fordíthatjuk a NOT kulcsszóval.

Összefoglalva a belső SELECT utasítás és a hozzátartozó operátorok ugyanúgy kombinálhatók mint a hagyományos feltétel-kifejezések.

6.6.5. SQL rendszertáblák

SQL üzemmódban nem használható a DIPLAY STRUCTURE és a DIR utasítás. E hiány sok kényelmetlenséget okozhatna, hiszen sem a tábláink listáját, sem azok szerkezetét nem tudnánk megjeleníteni. Szerencsére – de nem véletlenül – a dBASE SQL rendszertábláiból mindezt, sőt még ennél több információt is megtudhatunk.

A rendszertáblák minden SQL könyvtárban megtalálhatók és az adott adatbázisra vonatkoznak. Az SQL rendszertáblákat sorolja fel a következő táblázat.

Tábla neve	Tábla tartalma
SYSAUTH	Az adatbázis tábláira vonatkozó hozzáférési jogokat tartalmazza.
SYSCOLAU	Az adatbázis tábláinak oszlopaira vonatkozó hozzáférési jogokat tartalmazza.
SYSCOLS	Az adatbázisban található táblák oszlopait írja le.
SYSDBS	A SQL adatbázisok leírását tartalmazza. Ez a tábla csak az SQLHOME könyvtárban található meg.
SYSIDXS	Az adatbázis indexeinek leírását tartalmazza.
SYSKEYS	Egy indexnek a rendezési szempontjait tartalmazza.
SYSSYNS	Az adatbázisban létező másodneveket leírását tartalmazza.
SYSTABLS	Az adatbázis tábláinak leírását tartalmazza.
SYSTIMES	Az adatbázis tábláinak dátumát tartalmazza.
SYSVDEPS	Leírja, hogy egy nézettábla milyen alaptáblákból épül fel.
SYSVIEWS	Az adatbázis nézettábláinak leírását tartalmazza.

6.8. táblázat

Példa:

Ha ki akarjuk listázni az *alkalmzt* tábla oszlopait, azaz a tábla szerkezetére vagyunk kíváncsiak, akkor kiadhatjuk a következő utasítást:

```
SQL. SELECT *
       FROM syscols
       WHERE tbname="ALKALMZT" ;
```

A SQL **rendszertáblák szerkezete azonos az adattáblákéval**, így információnyerésre használhatjuk a közönséges SQL utasításokat is.

Példa:

Ha sűrűn van szükségünk táblák oszlopainak nevére, típusára, akkor a következő nézettáblát hozhatjuk létre:

```
SQL. CREATE VIEW oszl
     AS SELECT colname, coltype, collen, tbname
          FROM syscols
          WHERE NOT tbname LIKE "SYS%" ;
```

Így a nézettábla segítségével kevesebbet kell begépelni.

Ha az adatbázist alkotó összes táblára vagy nézettáblára vagyunk kíváncsiak, akkor a következő utasításokat használhatjuk:

```
SQL. SELECT *
     FROM systabls ;

SQL. SELECT *
     FROM sysviews  ;
```

Az első példában úgy a táblák, mint a nézettáblák megjelennek. A nézettábláknál a TBTYPE (táblatípus) oszlopban V szerepel, míg táblák esetén T.
A második példában csak a nézettáblák jelennek meg.

Az SQL **rendszertáblák** információkat tárolnak **önmagukról** is.

Példa:

```
SQL. SELECT colname, coltype, collen, tbname
     FROM syscols
     WHERE tbname LIKE "SYS% ;
```

utasítás kilistázza a rendszertáblákat alkotó oszlopokat.

Érdeklődjünk a jogosultságok felől:

```
SQL. SELECT * FROM sysauth ;

SQL. SELECT * FROM syscolau ;
```

Az első példa kilistázza, hogy az adatbázisban levő táblákra vonatkoztatva az egyes felhasználóknak milyen jogai vannak. A második utasítás arra mutat példát, hogyan kell megjeleníteni azt, hogy az egyes táblák oszlopaira kinek van UPDATE jogosultsága.

A rendszertáblák legfontosabb oszlopainak nevei és jelentésük:

Oszlopnév	Leírás
COLNAME	Oszlopnév.
COLTYPE	Oszlop típus.
COLLEN	Oszlop hossz.
TBNAME	Táblanév.
TNAME	Táblanév.
TBTYPE	Tábla típus.
CREATOR	Tábla tulajdonosa, a létrehozó.
TBCREATOR	Tábla tulajdonosa, a létrehozó.
GRANTOR	A jogosultságot átadó neve.
SQLTEXT	A SYSVIEWS táblában a nézettábla definicióját tartalmazza.
VIEWNAME	Nézettábla neve.
IXNAME	Index neve.
ORDERING	Rendezettség.

6.9. táblázat

6.7. A dBASE IV és az SQL

6.7.1. SQL adatvédelem

A adatbáziskezelőknek fontos feladata, hogy a tárolt információhoz csak az arra jogosult férhessen hozzá. A dBASE IV és a dBASE IV/SQL egyaránt lehetővé teszi, hogy az adathozzáférés jogosultságát többféle szinten is meghatározhassuk. dBASE IV-ben a hozzáférési jogokat a PROTECT utasítással lehet szabályozni. SQL nyelv más módon kezeli a jogosultságokat. Technikai okokból azonban **csak akkor használhatók az SQL adatvédelmi utasításai, ha a dBASE IV PROTECT** utasítással a védelmi rendszert **előzőleg üzembehelyezzük.**

SQL üzemmódban az adott táblára hozzáférési joggokal a **táblagazda,** a tábla létrehozója rendelkezik. Alapértelmezésben senki sem férhet hozzá más tábláihoz, adataihoz. Egyetlen felhasználó használhatja az összes SQL adatbázist, a **rendszergazda.** A rendszergazdát a bejelentkezési neve azonosítja. Az SQL rendszergazda mindig egy különleges névvel, az SQLDBA névvel rendelkezik. Ezt a

nevet éppúgy mint a többi felhasználó nevét, a PROTECT utasítással lehet kijelölni.

Az SQL a következő jogosultságokkal ruházza fel a tábla tulajdonosát illetve a rendszergazdát:

- Létező tábla oszlopait az ALTER utasítással bővítheti.
- Tábla, nézettábla sorait törölheti.
- Indexet hozhat létre adott táblához.
- Táblát új sorokkal bővítheti.
- Használhatja a táblát vagy nézettáblát.
- Módosíthatja a tábla sorait.

A **jog eredeti tulajdonosa átruházhatja** a **jogait** más felhasználóra is. A jog új tulajdonosa is továbbadhatja a jogosultságait, ha az eredeti tulajdonos ezt megengedi. A **GRANT** utasítás szolgál a jogok átruházására.

```
GRANT ALL [PRIVILEGES]/<jogosultséglista>
      ON [TABLE] <táblalista>
      TO PUBLIC/<felhasználólista>
      [WITH GRANT OPTION];
```

Az SQL az alábbi jogosultságokat különbözteti meg:

kulcsszó	jogkör
ALTER	ALTER utasítás használatával új oszlopok hozzáadása a táblához.
DELETE	Sorok törlése a táblából vagy nézettáblából.
INDEX	Index létrehozása a táblán.
INSERT	Új sorokat bevitele (hozzáadása) a táblához.
SELECT	A tábla vagy nézettábla használata SELECT utasítással.
UPDATE [(<oszloplista>)]	A tábla sorainak módosítása. Ha az oszloplista hiányzik, akkor a tábla összes oszlopát lehet módosítani, egyébként csak az oszlop listában szereplő oszlopokat.

6.10. táblázat

Az utasítás a már meglévő jogokat bővíti újabbakkal. Az átadott jogok természetesen akkor is érvényesek, ha a táblára másodnevével hivatkozunk. A **WITH GRANT OPTION** opció szolgál annak jelzésére, hogy a jog új tulajdonosa másra is átruházhatja jogait.

Példa:

```
SQL. GRANT ALL
     ON TABLE alkalmzt
     TO joska
     WITH GRANT OPTION;

SQL. GRANT INDEX, SELECT, INSERT, UPDATE (fizetes, jutalom)
     ON TABLE alkalmzt
     TO PUBLIC;
```

Az első példában a *joska* felhasználó minden jogot megkapott az *alkalmzt* táblával kapcsolatban és ezeket a jogokkal mást is felruházhat. A második példában minden felhasználónak engedélye van az *alkalmzt* tábla használatára, indexelésére, új sorokkal való bővítésére és a *fizetes* meg a *jutalom* oszlopok tartalmának megváltoztatására.

Az átruházott jogokat szükség esetén a **REVOKE** paranccsal lehet visszavonni.

```
REVOKE ALL [PRIVILEGES]/<jogosultságlista>
     ON [TABLE] <táblalista>
     FROM PUBLIC/<felhasználólista>;
```

A REVOKE utasítás a GRANT utasítás ikerpárjaként működik. Ugyanazokat a paramétereket lehet megadni, csak – értelemszerűen – a WITH GRANT OPTION opció hiányzik.

Az **ALL PRIVILEGES** kulcsszó teljes "jogfosztottságot" eredményez. A jogok és a jogokat reprezentáló kulcsszavak a lásd a GRANT utasításnál.

PUBLIC esetén **mindenkitől**, **felhasználólista** esetén **csak** a **megemlített felhasználóktól** kerülnek visszavonásra a különböző hozzáférési jogok. Ha olyan felhasználótól vonunk vissza jogot, aki jogát másnak is átadta, – bármennyire nem tűnik elég demokratikusnak – a REVOKE utasítással a másodlagos (harmadlagos, stb) jogtulajdonost is megfosztjuk jogaitól.
A **REVOKE ... FROM PUBLIC** utasítás csak azokra a jogokra vonatkozik, amelyeket a **GRANT ... TO PUBLIC** paranccsal ruháztunk át. Az olyan felhasználók, akik a jogaikat egyénileg kapták, megmaradnak nekik továbbra is. Ezeket csak egyedileg lehet visszavonni.

Példa:

```
SQL. REVOKE ALL PRIVILEGES
     ON TABLE alkalmzt
     FROM joska ;

SQL. REVOKE INDEX, UPDATE, INSERT
     ON TABLE alkalmzt
     FROM PUBLIC;
```

Az első példában a *joska* felhasználótól minden jogot visszavontunk az *alkalmzt* táblával kapcsolatban. Azok is elvesztették hozzáférési jogaikat az adott táblához, akik azokat a *joska* felhasználótól kapták.
A második példában minden felhasználótól megvontuk az indexelés, a módosítás és a bővités jogát az *alkalmzt* táblával kapcsolatban.

6.7.2. dBASE IV adatainak használata

A dBASE IV/SQL két gyökeresen különböző módszert nyújt nem SQL-beli adatok kezelésére.
Az egyik módszer szerint a kívánt adatokat a **LOAD** utasítással **konvertálhatjuk** dBASE IV/SQL formátumuvá, illetve **kitölthetjük** azokat az **UNLOAD** paranccsal más adatkezelő rendszerekbe (dBASE, Framework II, Rapidfile, Visicalc, Lotus 1-2-3, Multiplan, stb.). A **SELECT utasítás külön opcióval** rendelkezik, hogy az eredményül létrehozott sorokat **dBASE IV adatfájlba** (.dbf) kimentse. Az alkalmazásuk egyszerű részletes ismertetésük a függelékben megtalálható.

A másik módszer lényege, hogy az **adatokat kezelhetővé** (elérhetővé, módosíthatóvá, törölhetővé, stb) **teszi dBASE IV és dBASE IV/SQL üzemmódban egyaránt.**

A dBASE IV által kezelt .dbf és .mdx fájlok dBASE IV/SQL üzemmódban is használhatók. Tekintettel arra, hogy az SQL üzemmódban az adatkezelés szinte minden műveletéhez az SQL a rendszertábláit használja, addig lehetetlen a dBASE IV adatfájlok használata, amíg ezeket a rendszertáblákba nem jegyeztük (jegyeztettük) be. Erre szolgál a **DBDEFINE** utasítás, amelyről részletesebben a következő alfejezetben lesz szó.

Az SQL csak a .dbf és .mdx fájlokat kezeli, az *.ndx fájlokat nem!* Ha egyszerű index fájlokat akarunk SQL módban használni, akkor azt előbb integrálnunk kell egy összetett (.mdx) indexfájlba.

A DBDEFINE alkalmazása után az adatfájlt mindkét üzemódban lehet használni. SQL üzemmódban létrehozott táblákat dBASE IV-ben adatfájlként (.dbf) minden további nélkül használhatunk.

Ha mindkét üzemmódban rendszeresen használni akarjuk adatainkat, akkor több megszorításnak szigorúan eleget kell tennünk:

- Az adatfájl szerkezete dBASE IV módban nem változtatható meg, azaz tilos a MODIFY STRUCTURE utasítás alkalmazása.
- Tilos a CONVERT TO utasítás használata a .dbf állománnyal kapcsolatban.
- Nem hozható létre SQL táblával megegyező nevű .dbf fájl.

- Az adatfájlhoz tartozó .mdx fájlt, amelynek neve az adatfájl nevével negegyezik, nem módosíthatjuk dBASE IV üzemmódban.
- Nem módosíthatjuk magát az adatfájlt (.dbf) sem dBASE IV alatt, ha ahhoz SQL módban egyedi indexet határoztunk meg (CREATE UNIQUE INDEX).
- Ne módosítsuk az SQL rendszertáblákat.
- PROTECT-tel titkosított SQL táblák nem érhetők el dBASE IV üzemmódban. Ez fordítva is igaz. A titkosítást a dBASE IV végzi, de ezek az utasítások SQL üzemmódban is alkalmazhatók. A titkosítás menetét, kezelését (PROTECT, SET ENCRYPTION) a dBASE IV-gyel foglalkozó fejezetek tárgyalják.

6.7.3. Különleges adatbáziskezelő utasítások

Az egyik lehetőség a dBASE IV és SQL közötti kapcsolat megteremtésére a DBDEFINE. Segítségével már létező .dbf fájlokat, vagy egész alkalmazást felölelő fájl-halmazt lehet SQL-ben is elérhetővé tenni.

```
DBDEFINE [<.dbf fájlnév>];
```

Hatására az összes, az aktuális adatbázis könyvtárában lévő .dbf és a hozzátartozó .mdx fájlt csatolja az aktuális adatbázishoz és bejegyzi a rendszertáblákba a létezésüket. Ha a .dbf fájlnév szerepel az utasításban, akkor csak az adott adatfájl lesz a dBASE IV/SQL adatbázis része. Ahhoz, hogy egy alkalmazás összes adata SQL számára elérhető legyen a következőket kell tenni:

- Létre kell hozni a CREATE DATABASE utasítással egy új adatbázist. Ez nem szüséges ha az fájlokat egy létező SQL adatbázishoz akarjuk hozzáadni.
- Másoljuk át az összes .dbf, .mdx fájlt az SQL adatbázis könyvtárába.
- Aktivizáljuk az adatbázist a START DATABASE paranccsal.
- Adjuk ki a DBDEFINE utasítást.

Ismételnünk kell a korábbiakat:

A DBDEFINE num tud titkosított fájlokat kezelni. Nem kezeli az egyszerű indexfájlokat (.ndx) sem, ezért ezeket előbb az összetett indexbe kell integrálnunk, aztán alkalmazhatjuk a DBDEFINE utasítást.

Példa:

Tegyük az SQL üzemmódban még nem elérhető *vendors* adatfájlt használhatóvá:

```
SQL. DBDEFINE vendors ;
```

A másik különleges adatbáziskezelő utasítás a **DBCHECK**. Ellenőrzi, hogy az SQL katalógusai, rendszertáblái helyesen írják-e le az adatbázis aktuális állapotát. Ha bármilyen eltérés tapasztalható a valós helyzet és a rendszertáblák között, akkor hibaüzenetet kapunk.

DBCHECK [<táblanév>];

Hiba esetén a következő a teendő:

- A hibaüzenetben felsorolt .dbf fájlokat és a hozzájuk tartozó .mdx fájlokat másoljuk át egy ideiglenes könyvtárba.
- DROP utasítással töröljük ki az eredeti táblákat az adatbázisból.
- Másoljuk vissza az adatfájlokat és a hozzájuk tartozó egyéb fájlokat az adatbázis könyvtárába.
- Futtassuk a DBDEFINE utasítást a táblák használhatósága érdekében.
- A feleslegessé vált ideiglenes könyvtárat tartalmával együtt töröljük.

A **RUNSTATS** utasítás szolgál az adatbázis rendszertábláinak aktualizálására.

RUNSTATS [<táblanév>];

Aktualizálja az adatbázis statisztikai információit. Ajánlatos végrehajtani ezt az utasítást valahányszor új táblát vagy indexet hozunk létre, vagy a tábla sorainak legalább tíz százaléka megváltozik. Amennyiben a dBASE IV/SQL statisztikai információi nem pontosak az adatbázis műveletek végrehajtási ideje jelentősen is csökkenhet. Ha táblanév szerepel az utasításban, akkor csak a megadott tábla információi frissülnek.

Példa:

```
SQL. RUNSTATS ;
```

6.7.4. SQL-ben tiltott dBASE IV utasítások és függvények

Az SQL üzemmódban nem használhatjuk az alábbi utasításokat:

APPEND [BLANK]	APPEND FROM
APPEND MEMO	ASSIST
AVERAGE	BROWSE
CALCULATE	CHANGE
CONTINUE	CONVERT TO
COPY INDEXES/TAG	COPY MEMO
COPY STRUCTURE	COPY TO ARRAY
COUNT	CREATE FROM
CREATE/MODIFY	CREATE/MODIFY APPLICATION
CREATE/MODIFY LABEL	CREATE/MODIFY QUERY
CREATE/MODIFY REPORT	CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW	DELETE/DELETE TAG
DISPLAY	DISPLAY STRUCTURE
EDIT	EXPORT TO
FIND	GO/GOTO [GOTO]
IMPORT FROM	INDEX
INSERT	JOIN
LABEL FORM	LIST
LIST STRUCTURE	LOCATE
MODIFY STRUCTURE	PACK
RECALL	REINDEX
REPLACE	REPORT FORM
RESET	ROLLBACK
SCAN	SEEK
SELECT	SET
SET BLOCKSIZE	SET CARRY
SET CATALOG	SET DESIGN
SET FIELDS	SET FILTER
SET INDEX	SET INSTRUCT
SET MEMOWIDTH	SET NEAR
SET ORDER	SET RELATION
SET SKIP	SET TITLE
SET UNIQUE	SET VIEW
SET WINDOW OF MEMO	SKIP
SORT	SUM
UNLOCK	TOTAL ON
UPDATE	USE
ZAP	

6.11. táblázat

Az SQL utasításokban tiltott függvények listája. A csillaggal megjelölt függvények használhatók SQL üzemmódban az SQL utasításokon kívül.

&*	ACCESS()	ALIAS()
BAR()*	BOF()	CALL()*
CHANGE()	COL()*	COMMAND()
COMPLETED()*	DBF()	DELETED()
DISKSPACE()*	EOF	ERROR()*
FIELD()	FILE()*	FKLABEL()*
FKMAX()*	FLOCK()	FOUND()
GETENV()*	IFF()*	INKEY()*
ISALPHA()*	ISCOLOR()*	ISLOWER()*
ISMARKED()	ISUPPER()*	LASTKEY()*
LIKE()*	LINENO()*	LKSYS()
LOOKUP()	LUPDATE()	MAX()*
MDX()	MEMLINES()	MEMORY()*
MENU()*	MESSAGE()*	MIN()*
MLINE()	NDX()	NETWORK()*
ORDER()	OS()*	PAD()*
POOL()*	POPUP()*	PRINSTATUS()*
PROGRAM()*	PROMPT()*	PROW()*
READKEY()*	RECCOUNT()	RECNO()
RECSIZE()	RLOCK()	ROLLBACK()*
ROW()*	SEEK()	SET()*
TAG()	TYPE()*	VARREAD()*
VERSION()*		

6.12. táblázat

6.8. Az SQL programban

6.8.1. Általános tudnivalók

A fejezetben eddig csak az SQL interaktív üzemmódját taglaltuk. Természetesen lehetőség van arra is, hogy az SQL utasításokat programban használjuk.

SQL programok fejlesztése közben alkalmazhatjuk a dBASE IV szinte teljes eszköztárát (pl.: DEBUG), és ezek kezelése is a megszokott módon történik. A dBASE üzemmódban létező lehetőségek ismertetésére nem térünk ki, azok a kőnyv megfelelő fejezeteiben megtalálhatók. Itt csak az SQL utasítások programban való használatának sajátosságairól szólunk.

Az SQL utasításrendszer kizárólag az adatok manipulálására szolgál. Az egyéb funkciók, mint például a képernyő kezelése, hiányoznak. Ez azért van így, mert az SQL interpretációk szinte kivétel nélkül *befogadó nyelvűek*, azaz **más programnyelvekbe ágyazódnak be.** Ez azt jelenati, hogy az adatkezelésen kívüli funkciókat más programnyelv utasításaival kell megoldani. A dBASE IV/SQL-t "befogadó" programnyelv nem más, mint maga a dBASE IV. Ezzel lehetőség nyílik minden igényt kielégítő programok készítésére. (A megszorításokról már korábban beszéltünk.)

SQL utasításokat tartalmazó programot nem csak SQL üzemmódban írhatunk. Programunk kiterjesztése mutatja majd meg, hogy abban milyenfajtaz utasítások találhatók. A **.prs** kiterjesztésű program tartalmaz SQL utasításokat, és futtatásakor automatikusan SQL üzemmódba kapcsol át. Éppen ezért SQL programban a SET SQL utasítást nem szabad használni.

A programba ágyazott SQL kiegészül a táblasorok egyedi kezelésének lehetőségével. Ez azt jelenti, hogy az interaktív üzemmódtól eltérően lehetséges a kiválasztott **sorokat egyenként**, egymástól függetlenül **feldolgozni**, szükség szerint más és más eljárást alkalmazva az egyes sorokra.

A probléma kezelésére bevezették a rekordmutató, az ún. kurzor fogalmát. Bár ebben a fejezetben nem keveredik a kétféle kurzor (ti. a képernyőé) mégis megkülönböztetésül az **sql-kurzor** nevet fogjuk használni a továbbiakban.

Az sql-kurzor a megadott feltétel szerinti sorok kiválasztására szolgál. A sorokat csak sorban egymás után lehet elérni, de ez a sorrend szabadon meghatározható. Amikor az sql-kurzor kiválaszt egy sort, akkor a sor minden oszlopa (mezője) változókba kerül. A programban a sor tartalmát ezeken a változókon keresztül vizsgálhatjuk, műveleteket végezhetünk velük, és segítségükkel a sor tartalmát módosíthatjuk, a sort törölhetjük, stb.

6.8.2. Az sql-kurzort kezelő utasítások

A következő utásítások szolgálnak a kuzor kezelésére:

DECLARE	Kijelöli, meghatározza az sql-kurzor által elérhető sorokat.
OPEN	Aktivizálja a megadott sql-kurzort.
FETCH	Az sql-kurzort a következő (a feltételnek megfelelő) sorra mozgatja.
CLOSE	Lezárja a megadott sql-kurzort.

Ezeknek az utasításoknak az együttműködése a következő:

A **DECLARE** paranccsal lehet a különböző sql-kurzorokat meghatározni, kjelölni.Ezekből több is lehet. Az így kijelölt sql-kurzort a programban később bárhol, akár többször aktivizálni lehet. Az sql-kurzor által **elérhető sorokat** a DECLARE utasításban egy **belső SELECT** utasítás írja le.

Ha a megadott sorokat el akarjuk érni, akkor az **OPEN** parnccsal kell a kiválasztott sql-kurzort **aktivizálni**. Egyszerre legfeljebb 10 sql-kurzor lehet aktív. Az OPEN utasítás kiadása után az sql-kurzor a kiválasztott sorok közül az **első elé (!) áll.**

A **FETCH** utasítás hatására az sql-kurzor a **következő**, azaz kezdetben az első **sorra** fog **mutatni**, majd beolvassa a sor oszlopait a megadott változókba. A kiválasztott sort módosítani, törölni lehet.

Miután az összes sor elfogyott (fájl vége), az sql-kurzort a **CLOSE** paranccsal le kell zárni. Ezt megtehetjük előbb is.

A **hibakezeléshez** két különleges változó, az **SQLCODE** és az **SQLCNT** ad támogatást. Az első tartalmazza a végrehajtott SQL utasítás **hibakódját**, a második a kiválasztott **sorok sorszámát** adja vissza. Ez a két változó automatikusan létrejön az első kiadott SQL utasítás után úgy interaktív üzemmódban, mint programban. Az SQLCNT mutatja meg, hogy az utasításban hány sor vesz részt.

Ha egy SQL utasítást nem lehet végrehajtani valamilyen hiba bekövetkezése miatt, akkor azt a **ON ERROR**, illetve az **ERROR() függvénnyel** lehet lekezelni. Az utasítás végrehajtása esetén az SQLCODE tartalmazza az SQL hibakódót. Értéke 0, ha nem volt hiba. Ha az utasítás nem eredményzett egyetlen sort sem, akkor a tartalma 100. Ilyen helyzet előállhat például a SELECT és FETCH utasítások végrehajtásakor. Ekkor természetesen az SQLCNT tartalma 0 lesz.

Lássuk sorban az sql-kurzorutasításokat:

```
DECLARE <sql-kurzornév> CURSOR
    FOR <select mondat>
    [FOR UPDATE OF <oszloplista>/<ORDER BY cikkely>];
```

Ha az **ORDER BY** cikkely szerpel, akkor az sql-kurzor a tábla sorain a megadott rendezési szempont szerint fog végigjárni.
Ha a **FOR UPDATE OF** cikkely **hiányzik** az utasításból, akkor a kiválasztott sorok **nem módosíthatók**. A későbbiekben is csak azok az oszlopok változtathatóak meg UPDATE utasítással, amelyek neve szerepel a kulcsszó után.
A FOR UPDATE OF és az ORDER BY cikkelyek *kölcsönösen kizárják* egymást.
A select mondatban nem szerepelhet SAVE TO TEMP és INTO cikkely.

```
OPEN <sql-kurzornév>;
```

Az sql-kurzor a DECLARE utasítás által kijelölt első sor elé áll és sehova sem mutat.

```
FETCH <sql-kurzornév>
    INTO <változólista>;
```

Az előzőleg DECLARE CURSOR és OPEN utasításokkal aktivizált sql-kurzort a következő sorra viszi. Az így aktuálisá vált sor oszlopainak tartalmát az INTO kulcsszó után felsorolt változókba teszi. A változók és az oszlopok létszámának egyezni kell.

```
CLOSE <sql-kurzornév>;
```

Lezár egy korábban OPEN-nel aktivizált sql-kurzort. Hatására az sql-kurzor aktivizálásakor keletkezett munkafájlt törli és felszabadítja az sql-kurzor által használt munkaterületet.

6.8.3. Egyéb sql-kurzorutasítások

Az sql-kurzorral kijelölt sort két utasítással kezelhejük: a **DELETE** és az **UPDATE** utasításokkal, amelyek formája némileg eltér a korábban megismertektől.

```
DELETE FROM <táblanév>
    WHERE CURRENT OF <sql-kurzornév>;
```

Az sql-kurzor által kijelölt **sort törli** a táblából. Ehhez a DECLARE CURSOR utasításnak tartalmaznia kell az FOR UPDATE OF cikkelyt, ellenkező esetben sorokat nem tudjuk törölni. A sor törlése után az sql-kurzor nem mutat "sehová", ezért az sql-kurzort egy újabb FETCH utasítással aktualizálni kell.

```
UPDATE <táblanév>
    SET <oszlopnév> = <kifejezés>
        [, <oszlopnév> = <kifejezés> ...]
    WHERE CURRENT OF <sql-kurzornév>
```

A tábla egy sql-kurzor által kijelölt **sorának tartalmát módosítja**. Csak azok az oszlopok módosíthatók, amelyek a DECLARE CURSOR utasítás FOR UPDATE OF cikkelyjében szerepelnek. A kifejezés és a hozzátartozó oszlop típusának meg kell egyezni. Ez a módosító utasítás **nem mozgatja az sql-kurzort**, ezért az utasítás használata után még újra módosítható vagy akár törölhető is a kijelölt sor.

Bár nem kapcsolódik szorosan az sql-kurzor használatához az INSERT INTO utasítás, de van egy változata, amelyik egyedi sor beszúrására szolgál. Az új sor mindig a tábla végére kerül.

```
INSERT INTO <táblanév>
    [(<oszloplista>)]
    VALUES (<értéklista>);
```

Az értéklistában szereplő konstansok, változók és dBASE IV függvények által meghatározott értékek kerülnek a megadott oszlopokba (mezőkbe). Programban többnyire függvények vagy változók szerepelnek az értéklistában.

6.8.4. SELECT INTO

Az előző alfejezetben láthattuk, hogy lehetséges egyedi sorok kezelése SQL üzemmódban. Egy kiválasztott sor elérése azonban nagyon bonyolultnak tűnik (esetenként az is), több utasítás végrehajtása szükséges hozzá (DECLARE, OPEN, FETCH, CLOSE). Ha csak arra van igényünk, hogy a kiválasztott sor tartalmát megvizsgáljuk vagy megjelenítsük, akkor egyszerűbb lehetőség is rendelkezésünkre áll.
A SELECT utasítás rendelkezik az INTO cikkellyel, amely csak programban alkalmazható. Az INTO cikkelyben a SELECT utasításban szereplő oszlopnak megfelelő változók állnak. A SELECT utasítás többi cikkelye a szokásos és ugyanúgy kell alkalmazni őket, mint ahogy azt eddigi alfejezetekben megismertük.

```
SELECT [ALL/DISTINCT] {*/<kifejezéslista>}
    INTO <változólista>
```

A WHERE cikkelyben a keresési feltételt úgy kell megszerkeszteni, hogy az utasítás csak egy sort válasszon ki a táblából. Az oszlopok és a változók létszámának meg kell egyezni, a változók felveszik a hozzájuk tartozó oszlopok típusát.

Például:

```
SQL. SELECT DISTINCT kodszam, nev, fizetes
     INTO mkod, mnev, mfizu
     FROM alkalmzt
     WHERE kodszam=71976;
```

6.8.5. Példaprogram az sql-kurzor használatára

```
SET SCORE OFF
SET STATUS OFF
SET TALK OFF
START DATABASE samples ;
DECLARE mutato CURSOR
     FOR SELECT * FROM alkalmzt
          WHERE fizetes < 5000
          FOR UPDATE OF fizetes ;
OPEN mutato;
@ 2,5 SAY "A kivalasztott sorok szama:"
@ 2,34 SAY sqlcnt
@    4,5    SAY    "-------------------------------------------------------"
WAIT "Nyomjon meg egy gombot..."
CLEAR
FETCH mutato INTO mkod,mvez,mker,mfizdat,mmhely,mfo-
nok,mfiz,mjut ;
chrr= " "
CLEAR
DO WHILE sqlcode = 0
     @ 1,5 SAY "SQLCODE:"
     @ 1,14 SAY sqlcode
     @ 2,5 SAY "SQLCNT :"
     @ 2,14 SAY sqlcnt
     @    4,5    SAY    "-------------------------------------------------------"
     @ 6,7 SAY "Kodszam:"
     @ 6,17 SAY mkod PICTURE "XXXXXX"
     @ 6,34 SAY "Nev:"
     @ 6,39 SAY mvez PICTURE "XXXXXXXXXXXXXXX"
     @ 6,54 SAY mker PICTURE "XXXXXXXXXX"
     @ 8,0 SAY "Fonok kodszama:"
     @ 8,17 SAY mfonok PICTURE "XXXXXX"
     @ 10,0 SAY "Fizetes datuma:"
     @ 10,16 SAY mfizdat
     @ 10,30 SAY "Fizetes:"
     @ 10,39 SAY mfiz PICTURE "999999"
     @ 10,48 SAY "Jutalek:"
     @ 10,57 SAY mjut PICTURE "99.9"
     @ 12,5 SAY "Munkahely:"
     @ 12,16 SAY mmhely PICTURE "XXXXXXXXXXXXXXX"
     @ 19,5 SAY "Lehetoseg K=kovetk., T=torles, F=fiz.emeles"
     @ 20,10 SAY "Valasztasa:" GET chrr PICTURE "@M K,F,T"
     READ
```

```
        DO CASE
              CASE chrr="T"
                    DELETE FROM alkalmzt
                          WHERE CURRENT OF mutato;
              CASE chrr="F"
                    UPDATE alkalmzt SET fizetes=fizetes*1.1.
                          WHERE CURRENT OF mutato;
        ENDCASE
        FETCH mutato INTO mkod,mvez,mker,mfizdat,mmhely,
                          mfonok,mfiz,mjut ;
ENDDO
CLEAR
@ 1,5 SAY "SQLCODE:"
@ 1,14 SAY sqlcode
@ 2,5 SAY "SQLCNT :"
@ 2,14 SAY sqlcnt
@    4,5    SAY    "-------------------------------------------------"
CLOSE mutato ;
STOP DATABASE ;
SET SCORE ON
SET STATUS ON
SET TALK ON
```

7. A dBASE IV és a lokális hálózatok

7.1. A lokális hálózatokról általában

A klasszikus, nagy- és középkategóriájú számítógépeken fejlesztett alkalmazások általában valamilyen többfelhasználós (multiuser) és/vagy többfeladatos (multitasking) rendszerekre épültek. A 80-as évek elején Magyarországon is kezdett tért hódítani egy újfajta szemlélet, amelyet a professzionális személyi számítógépek eleinte óvatos, majd rohamosan meginduló alkalmazása tett lehetővé. Sokan rájöttek, hogy a rengeteg pénzbe kerülő, bérmunkába kiadott ügyviteli adatfeldolgozásnak egy jelentős része megoldható az új gépekkel is, ráadásul ezeknek az ára lehetővé tette saját gépek beszerzését s így a belső igény szerinti programrendszerek kifejlesztését. Az események tehát az egyfelhasználós folyamatok gyors terjedését hozták magukkal.

Ezek a gépek azonban (nevükből is következően) nem kifejezetten nagyvállalatok ügyviteli adatainak feldolgozására készültek, s a nagyobb kényelemhez szokott felhasználók komoly korlátokba ütköztek a sebességet és a háttértárolók kapacitását illetően. Erre az időre azonban már oly mértékben elterjedtek, hogy nem maradt más megoldás, mint a lokális hálózatok megalkotása, s ezzel tulajdonképpen megkezdődött a visszatérés az eredeti koncepció felé. A tendencia olyan erős, hogy egyes nyugati szaklapok előrejelzése szerint 1995-re minden második PC valamely hálózat részeként fog működni.

7.1.1. Alapfogalmak

A lokális (helyi) hálózat két vagy több számítógép összekapcsolása útján jön létre, ahol az összeköttetést nagy sebességű adatátviteli vonalak teremtik meg. A hálózatban résztvevő gépek mindegyike saját központi egységgel rendelkezik, így az egyfelhasználós rendszerek esetleges előnyei változatlanul megmaradnak. Ugyanakkor a hálózati elemekhez kapcsolt perifériák (lemezek, nyomtatók, közös néven: **erőforrások**), sőt a programok és az adatok is megoszthatóak.

A hálózat azért **lokális**, mert egyrészt a gépek közti kapcsolat nem telefonvonalakon keresztül valósul meg, másrészt annak a körnek a sugara, amelyen belül az összekapcsolt gépek elhelyezkednek, általában néhányszor 100 m lehet maximum (a pontos érték az aktuális hálózati hardvertől függ).

Egy hálózatban legalább egy mikroszámítógépet ún. **file server**-ként kell kijelölni. Ezt a továbbiakban **hálózatvezérlőnek** nevezzük, mert feladata többek között a hálózati erőforrások megosztása és a gépek közötti kommunikáció kezelése. A

hálózatvezérlő által kiszolgált egyéb számítógépeket **munkaállomásoknak**, munkahelyeknek vagy ritkábban hálózati csomópontoknak nevezzük.

Lokális hálózatban egy megosztott alkalmazói program a hálózatvezérlőről kerül a munkaállomás memóriájába, s mivel ott is fut, nem lassítja a többi munkahely feldolgozásait (itt most csak a programfutásról van szó, a közös adatokon végzett manipulációk természetesen lassíthatják a feldolgozást, sőt elvétve kialakulhat ún *holtpont,* patthelyzet is, amikor akár végtelen ideig is várakozhatunk, ha van kedvünk).

A lokális hálózatoknak világszerte számos megvalósítása létezik. Hazánkban a **Novell** cég hálózati programcsomagja terjedt el leginkább, ezért a továbbiakban a dBASE IV hálózati működtetésével kapcsolatos tudnivalókat a Novell NetWare keretei közt ismertetjük.

7.1.2. Az adatbiztonsággal kapcsolatos problémák

Mint láttuk, a hálózat legfőbb előnye az adatok megoszthatóságából fakad. Ez veti fel egyben a legélesebb követelményt is, ti. az adatok **integritásának**, sértetlenségének megőrzését a munka során. Tipikus veszélyforrás az, amikor két munkaállomás ugyanazon adatfájl egy meghatározott részét akarja egyidőben módosítani. Ez nyilván elkerülendő mégpedig úgy, hogy a közösen használt adatokhoz egyidőben csak egyetlen munkaállomás férhessen hozzá, s a többi igényt sorba kell állítani, amíg az érintett adathalmaz (ami általában egy fájl vagy egy rekord) felszabadul.

Előfordulnak azonban másfajta problémák is. Például ha az adatfájl tartalmát (vagy annak egy részét) csak bizonyos felhasználók számára kívánjuk hozzáférhetővé tenni, akkor valamilyen módon meg kell különböztetnünk őket egymástól. Arra is szükségünk lehet, hogy egy fájlhoz különböző felhasználóknak más és más típusú hozzáférési jogot engedélyezzünk.

Megint más jellegű adatvédelemmel állunk szemben akkor, amikor egy konkrét adatfájl vagy program jellemzőit határozzuk meg az összes munkaállomással szemben.

Végül képzeljük el azt ,hogy a munkaállomásokat sokan, sokféle célra használhatják. Van aki csak adatrögzítést végez egy konkrét program segítségével, van aki dBASE IV programokat fejleszt és a legfelső szinten áll a **rendszergazda** (supervisor), aki az egész hálózat működéséért felelős. Nyilvánvaló, hogy a munkájukhoz szükséges adatok köre lényegesen eltér egymástól.

Az eddigiek alapján a hálózat adatvédelmi rendszere az alábbi 4 szintre tagolódik:

7.1.2.1. Bejelentkezés a hálózatba (LOGIN/password)

Minden felhasználónak rendelkeznie kell egy ún **bejelentkezési névvel**, amelyet a különleges jogokkal felruházott SUPERVISOR, más néven a rendszergazda oszt ki az adott hálózatvezérlőhöz. A jelszó hozzárendelése opcionális, de ha van, akkor azt is meg kell adni, és addig nem tudunk bejelentkezni, amíg a megfelelő nevet és jelszót be nem írtuk.

A rendszergazda megszabhatja a következőket is:

- adott felhasználó csak meghatározott időszakban jelentkezhet be
- adott felhasználó csak meghatározott munkaállomáson jelentkezhet be
- stb

7.1.2.2. Adathozzáférés (trustee)

Ez a védelmi szint egy adott felhasználó jogait határozza meg egy adott alkönyvtár fájljaival kapcsolatban. Ezek a következők lehetnek:

R engedély az adott könyvtárban létező megnyitott fájlok tartalmának megtekintésére

W engedély az adott könyvtárban létező megnyitott fájlok tartalmának módosítására

O engedély az adott könyvtárban létező fájlok megnyitására

C engedély az adott könyvtárban még nem létező fájl létrehozására. A létrehozott fájl meg is nyílik, így nem kell az **O** joggal rendelkeznünk. Ha a **C** mellett a **P** jogot is megkaptuk, új alkönyvtárak létrehozása is megengedett.

D engedély az adott könyvtárban még létező fájl törlésére. Ha a **C** mellett a **P** jogot is megkaptuk, úgy alkönyvtárak törlése is megengedett.

P engedély alkönyvtárak létrehozására, törlésére, átnevezésére (de ehhez kell a **C**, **M**, **D** is), valamint a jogok más felhasználóknak történő átadására.

S engedély az adott könyvtárban lévő fájlok közti keresésre, azaz pl: a DIR vagy NDIR utasítások kiadására

M engedély az adott könyvtárbeli fájlok szabvány DOS jellemzőinek (*attribútumainak*) illetve a kiterjesztett hálózati jellemzőknek módosítására, illetve alkönyvtárak és fájlok átnevezésére, ha a **P** jogot is megkaptuk. Az jellemzők részletezése a későbbiekben megtalálható.

7.1.2.3. Az alkönyvtárak elérése

Míg az előző pontban egy felhasználónak az alkönyvtárakhoz való viszonyát tárgyaltuk, most egy alkönyvtár és az összes felhasználó viszonyát vizsgáljuk. Egy alkönyvtár létrehozásakor az előbb említett 8 jog mindegyike érvényes. Ez azt jelenti, hogy bármely felhasználó a felsorolt összes tevékenységet elvégezheti az adott alkönyvtáron, ha ezekkel a jogokkal egyébként rendelkezik. Alkönyvtár védelemről akkor beszélünk, ha bizonyos jogokat letiltunk az adott könyvtárra vonatkozóan. Ettől kezdve a letiltott jogokat senki sem gyakorolhatja, függetlenül attól, hogy ezeket korábban megkapta vagy sem. A rendszergazda természetesen ez alól is kivétel.

7.1.2.4. A fájlok adatvédelmi jellemzői

Ez a védelmi szint egy konkrét fájlra vonatkozóan állapítja meg a következő dolgokat:

R/W — a fájl írható, olvasható, átnevezhető és törölhető, ha ezek a jogok egyébként is biztosítottak.

R/O — a fájl csak olvasható, ha ez a jog biztosított. A többi tevékenység nem megengedett függetlenül a felhasználó egyéb jogaitól.

S — a fájl megosztható, azaz egyidőben több felhasználó is olvashatja. Általában az R/O jellemzővel együtt használatos.

NS — a fájl nem megosztható, azaz egyidőben csak egy felhasználó férhet hozzá.

N — a fájl normál jellemzőjű, azaz nem megosztható és R/W. Ez az alapértelmezés.

Ezek a védelmi szintek egyenként és egymással kombinálva is használhatók. Meg kell jegyeznünk, hogy egy felhasználó tényleges jogai az Adathozzáférés és az Alkönyvtárak elérése részekben megfogalmazott jogok közös részeként állnak elő. A fájlok adatvédelmi jellemzői viszont elsőbbséget élveznek még a tényleges jogokkal szemben is. Hiszen hiába van írási jogosultságunk egy alkönyvtárra, ha abban egy adott fájl R/O jellemzővel van ellátva. A rendszergazda az összes joggal rendelkezik minden alkönyvtár felett.

7.2. A dBASE IV használata hálózatban

A dBASE IV hálózati alkalmazhatósága nagymértékben hasonlít a dBASE III PLUS adatbáziskezelő rendszer hasonló célú működéséhez. A megvalósítás módjában (néhány új utasítás és függvény megjelenése, valamint a PROTECT belső utasítássá tétele folytán) igen, de filozófiájában nem tartalmaz lényeges újdonságokat. Azok az olvasóink tehát, akik a dBASE III PLUS rendszer hálózati üzemeltetésében már gyakorlottak, lapozzanak a 4. alfejezetre.

Mivel a dBASE IV egy adatbáziskezelő rendszer, természetes igényként merül fel az, hogy viszonylag nagy mennyiségű adatot minél rövidebb időn belül rögzíthessünk, valamint ezekből a lehető legtöbb irányú információkat nyerhessük a későbbi feldolgozások során. Mindannyian tudjuk, hogy egy vállalat ügyviteli folyamatainak számítógépre vitele során egyetlen IBM XT/AT hamar "megtelik". Ez a probléma megoldható úgy, hogy újabb és újabb gépeket állítunk csatasorba, s ezeken egymástól függetlenül végezzük a feldolgozásokat. Ekkor azonban hamarosan kiderül, hogy a különböző gépeken használt adatoknak egy igen tekintélyes része majdnem mindenütt előfordul (gondoljunk csak egy bérszámfejtő rendszer és egy tőle függetlenül működő munkaügyi rendszer személyi törzsadataira, vagy egy anyagnyilvántartó- és egy raktári-pénzügyi rendszer cikk, készlet stb. adataira). Ez nyilvánvalóan gazdaságtalan megoldás egyrészt az adatok redundáns tárolása, a függetlenül működő rendszerek adatkapcsolatainak nehézkessége, másrészt az adatok rögzítésének és ellenőrizhetőségének hiányosságai miatt. Az ilyen és ehhez hasonló problémák hatékony megoldása lehet az említett számítógépek lokális hálózatba kapcsolása oly módon, hogy a dBASE IV rendszer az alkalmazói programokkal és a mindenki által használt adatfájlokkal együtt egy (vagy több) hálózatvezérlő merevlemezére kerül, s a munkaállomásokra csak azokat az adatokat tesszük, amelyeket nem akarunk másokkal megosztani.

Az eredetileg egyfelhasználós környezetre készített dBASE IV programok elvileg minden változtatás nélkül használhatóak hálózatban is, mivel minden egyfelhasználós utasítás és függvény megengedett hálózati viszonyok között is, a gyakorlatban azonban a biztonság érdekében ezeket a programokat fel kell készíteni az ütközések kivédésére, a *holtpont* elkerülésére és egyéb később részletesen kifejtendő adatvédelmi funkciók ellátására is.

7.2.1. A dBASE IV telepítése többfelhasználós környezetben

A dBASE IV hálózati üzemeltetésének feltételei:

- merevlemezes háttértárral és legalább 640 Kb központi memóriával (RAM) felszerelt hálózatvezérlő, amelyet a hálózati rendszer sebességének elviselhetősége végett célszerű legalább egy 286-os processzorra alapuló IBM AT-nek vagy ezzel kompatibilis gépnek választani,
- A számítógépek kapcsolatát biztosító csatolókártyák, kábelek (pl: Orchid, Ethernet, stb).
- Hálózati programcsomag (pl: Novell Network, IBM PC Network, stb).
- legalább egy hajlékony lemezes háttértárolóval (floppy drive) és 512 Kb központi memóriával rendelkező legalább 1 db munkaállomás
- DOS 3.1 vagy ennél magasabb verziószámú operációs rendszer

Emlékeztetőül megemlítjük, hogy a dBASE III PLUS-hoz tartozik egy ún. hálózati programcsomag, amely az alábbi elemekből áll:

- dBASE ADMINISTRATOR
- dBASE ACCESS
- PROTECT

Az első kettő nélkülözhetetlen a dBASE III PLUS hálózati installálásához, a PROTECT használata, amely az adatvédelmi szintek létrehozására és karbantartására szolgál, viszont nem kötelező, csak ajánlott.

A dBASE IV esetén nincs ilyen a rendszer egészétől elkülönülő hálózati programcsomag, így a hálózati működéséhez nem kell mást tennünk, csak a rendszert tartalmazó lemezek valamelyikén megtalálható DBSETUP nevű programot kell elindítanunk, majd a **Multiuser installation** menüpont választásával a rendszer hálózatba telepítése párbeszédes módon megtörténik. Ennek részletes ismertetésével itt nem foglalkozunk.

7.2.2. A rendszergazda feladatai

A lokális hálózatban történő adatfeldolgozás fő célja számítógépes rendszerünk hardware- és software **erőforrás**ainak megosztása a munkaállomások között. Ilyen erőforrások a következők:

- a dBASE IV adatbáziskezelő rendszer és az egyéb akár dBASE IV-ben, akár más nyelven készült programtermékek, amelyekhez így minden munkaállomás hozzáférhet, holott csak 1 példányban tároljuk ezeket.
- a közösen használt adatfájlok, indexfájlok, amelyek nagymértékben csökkentik az adatok redundanciáját, kiküszöbölik az adatok többszörös karbantartását és szükségtelenné teszik a számítógépek között lemezmásolással megoldott adatátvitelt, amely lassú és adatvesztések forrása lehet.
- a közös néven perifériákként emlegetett lemezmeghajtók, merevlemezek, szalagos háttértárak, nyomtatók, rajzgépek stb.

Már a felsorolásból is kitűnhet, hogy ezen eszközök kezdeti elosztása, majd a munka során felmerült pillanatnyi igények szerinti újrafelosztás a folyó munkák erőforrásigényének, az adatfájlok hálózaton belüli elhelyezkedésének és magának az aktuálisan használt hálózati szoftvernek a pontos ismeretét követeli meg. Tehát szükség van egy olyan felhasználóra aki ezeket a feladatokat ellátja, viszont azt is biztosítani kell, hogy más ezt ne tudja megtenni. Ezt a speciális jogokkal felruházott felhasználót az angol terminológia szerint hívhatjuk rendszer-felügyelőnek (supervisor), de a könyvben inkább a Magyarországon elterjedt **rendszergazda** kifejezést használjuk.

Meg kell jegyeznünk, hogy a hálózat, és a hálózati üzemmódban működő dBASE IV rendszergazdája nem ugyanaz a fogalom, bár mindkét feladatot elláthatja egy személy is.

7.2.3. A munkaállomások kezelőinek feladatai

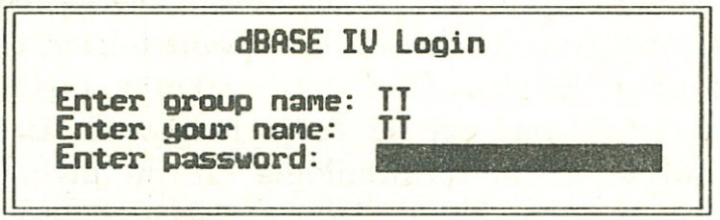
A munkaállomások kezelőit általában a **felhasználó** (user) névvel illetjük. Felhasználó mindenki, aki valamelyik hálózati csomóponton dolgozik. Ebben az alfejezetben áttekintjük azokat az alapfeladatokat, amelyekkel minden dBASE IV felhasználó találkozik hálózati környezetben.

7.2.3.1. Bejelentkezés

A dBASE IV indítása hálózatban ugyanúgy történik, mint egyfelhasználós környezetben:

```
DBASE
```

Ha a PROTECT rendszert nem használtuk még, akkor megjelenik a Control Center (ha úgy konfiguráltuk). Ha a védelmi rendszer aktív, azaz a PROTECT segítségével meghatároztunk bizonyos védelmi szinteket, akkor egy ún. bejelentkezési képernyő jelenik meg az alábbi formában:



7.1. ábra

Mint látható, itt egy csoport nevet, egy felhasználó nevet és egy jelszót kell megadnunk. Ezt a hármast együtt megfelelően tömör magyar megfelelő híján "**login**"-nak nevezzük. A rendszert csak érvényes login megadása után használhatjuk. A login-t a rendszergazda állítja elő minden felhasználó számára. Nevet minden felhasználó kap, de a jelszó hozzárendelése nem kötelező. A jelszó természetesen nem látható beírás közben, s megadása csak akkor kötelező, ha a rendszergazda ezt elrendelte (lásd a PROTECT-nél).

A **csoport** fogalma már egy másik védelmi szinttel kapcsolatos. A felhasználók bizonyos szempontok szerinti csoportokba sorolását szintén a rendszergazda vég-

zi azzal a céllal, hogy a különböző csoportok más és más fájlokhoz férhessenek hozzá.

Ha a login megadása közben hibázunk, újra kell kezdeni az egészet, s három sikertelen próbálkozás után a dBASE IV visszaad bennünket az operációs rendszernek. Sikeres bejelentkezés után a képernyőn megjelenik a Control Center (ha úgy konfiguráltuk).

7.2.3.2. Az ütközés, a zárolás és a holtpont

Hálózati környezetben az adatfájlok egy része más felhasználók által is hozzáférhető, így gyakran előfordulhatnak olyan helyzetek, amikor egy fájlt vagy annak egy részét ugyanabban az időben többen is módosítani akarják. Ezt a továbbiakban **ütközésnek** hívjuk. Tekintettel arra, hogy egy ilyen ütközés fatális hibákat eredményezhet, a hálózatban működő programtermékekkel szemben alapkövetelmény, hogy kizárják ennek lehetőségét az adatfájl vagy egyetlen rekord **zárolásával**. Zárolt fájlhoz/rekordhoz más felhasználó az olvasás kivételével nem férhet hozzá. A dBASE IV a zárolást automatikusan elvégzi pl. a BROWSE, EDIT, CHANGE utasítások végrehajtásakor, s a tevékenység befejezésekor a zárolást megszünteti. Programok írásakor azonban a programozó dolga, hogy a zárolásokról és azok feloldásáról gondoskodjék.

A zárolások nem eléggé körültekintő alkalmazása azonban újabb hibaforrás lehet. Tipikus eset az, amikor két felhasználó mindegyike fogva tart (zárol) egy-egy fájlt vagy rekordot, s mindkettő szeretné megnyitni (módosítani) éppen a másik által zárolt fájlt vagy rekordot. Ez természetesen egy *patthelyzet*, így a keresztbe-hivatkozás miatt akár végtelen ideig várakozhatnak egymásra. Az ilyen és ehhez hasonló állapotokat nevezzük **holtpont**nak vagy ritkábban halálos zárolásnak (deadlock). A programozónak törekednie kell arra, hogy holtpontmentes rendszereket készítsen.

7.2.3.3. Néhány alapvető interaktív utasítás

Ebben a részben a dBASE IV-et interaktív módon használókat segítő alapvető hálózati utasításokról és a zárolás interaktív módjairól lesz szó.

Ha olyan fájlt próbálunk megnyitni, amelyet egy másik felhasználó már kizárólagos használatra megnyitott, vagy ha már zárolt fájlt akarunk zárolni, a következő üzenet jelenik meg a képernyőn:
File is in use by *felhasználónév*. Retrying lock, press Esc to cancel
Ez annyit tesz, hogy a dBASE IV addig próbálja újra zárolni illetve megnyitni a fájlt, amíg nem sikerül vagy az **Esc** billentyű leütésével meg nem szakítjuk a folyamatot.
Ha egy rekordot próbálunk zárolni az EDIT, BROWSE, CHANGE utasítások végrehajtása közben, és ez nem sikerül, az alábbi üzenetet írja ki:
Record is in use by *felhasználónév*. Retrying lock, press Esc to cancel.

Meg kell jegyeznünk, hogy egy rekord zárolása maga után vonja a SET RELATION-nel kapcsolódó összes rekord zárolását is. A zárolás feloldására ugyanez vonatkozik.

DISPLAY/LISTSTATUS — Ez az utasítás hálózati környezetben a megszokott információkon kívül az egyes munkaterületeken megnyitott fájlok zárolt rekordjainak számát is feltünteti. Ha maga a fájl zárolt, akkor a Lock list: database locked üzenet jelenik meg. Ha a TO PRINT opciót is megadtuk, a kimenő adatok nem a képernyőre, hanem vagy a lokális nyomtatóra vagy bármely megosztott hálózati nyomtatóra kerülnek attól függően, hogy a SET PRINTER paranccsal mit határoztunk meg.

SETPRINTER — Ez az utasítás itt arra szolgál, hogy a nyomtatásra szánt adatokat átirányíthassuk a lokális nyomtatóról a hálózati eszközökre és fordítva.

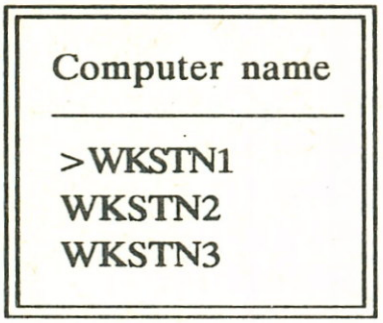
DISPLAYUSERS — Ezzel a paranccsal a dBASE IV-be aktuálisan bejelentkezett felhasználók login-ban megadott neveit tekinthetjük meg. A ">" jelöli az aktuális munkaállomáson dolgozó felhasználót.

```
.DISPLAY USERS
```

```
Currently Selected Database:
Select area:  1, Database in Use: C:\DBASE\TT.DBF   Alias: TT

File search path:
Default disk drive: C:
Print destination:  PRN:
Margin =      0
Refresh count =     0
Reprocess count =     0
Number of files open =     4
Current work area =     1

ALTERNATE  - OFF   DELIMITERS - OFF   FULLPATH   - OFF   SAFETY     - ON
AUTOSAVE   - OFF   DESIGN     - ON    HEADING    - ON    SCOREBOARD - ON
BELL       - ON    DEVELOP    - ON    HELP       - ON    SPACE      - ON
CARRY      - OFF   DEVICE     - SCRN  HISTORY    - ON    SQL        - OFF
CATALOG    - OFF   ECHO       - OFF   INSTRUCT   - ON    STATUS     - ON
CENTURY    - OFF   ENCRYPTION - ON    INTENSITY  - ON    STEP       - OFF
CONFIRM    - OFF   ESCAPE     - ON    LOCK       - ON    TALK       - ON
CONSOLE    - ON    EXACT      - OFF   NEAR       - OFF   TITLE      - ON
Press any key to continue...
Command | C:\dbase\TT          | Rec None          | File |
```



7.3. A PROTECT utasítás funkciói

A PROTECT használata nem kötelező. Felmerülhet a kérdés, hogy akkor miért hozták létre egyáltalán. Erre próbálunk meg választ adni néhány mondatban. Az előző néhány alfejezetben láttuk, hogy az adatok zárolása megoldja a hálózati alkalmazások legnagyobb problémáját, kiküszöböli az ütközések lehetőségét. Ezzel az eszközzel tehát az adatok sértetlenségét biztosíthatjuk, s minimalizálhatjuk a helytelen használatból következő adatvesztések lehetőségét. Van azonban az adatbiztonság problémakörének egy másik területe is, mégpedig az **illetéktelen hozzáférés megelőzése**. Ez nemcsak a misztifikált titkos kartokékok külvilág előli elrejtésekor lehet fontos, hanem tényleges gyakorlati haszna is van. Egy hálózatvezérlő merevlemezén ugyanis általában több, egymástól független alkalmazás adatfájljait, programjait tároljuk. Egyrészt kézenfekvő az az igény, hogy ezek a rendszerek (az esetleg egyébként kijelölt kapcsolódási pontok kivételével) egymás adatfájljait ne érhessék el. Másrészt a munkaállomásokon eltérő tudásszintű emberek különböző célokkal dolgoznak (adatrögzítés, programfejlesztés, rendszerkarbantartás, stb.), a biztonság érdekében célszerű lenne igényeik szerint differenciáltan megállapítani az adatokhoz való hozzáférési lehetőségeiket. Az eddigiek szerint az adatbiztonsággal kapcsolatos elvárásainkat összefoglalhatjuk egy mondatban: szeretnénk megszabni azt, hogy hálózatunkban mely felhasználók, mely adatokhoz, milyen módon férhetnek hozzá. Nos, a PROTECT nagyjából ennek a hármas funkciónak a megvalósítására szolgál.

Mint korábban már láttuk, maga a hálózati szoftver (esetünkben a Novell NetWare) is rendelkezik hasonló eszközökkel, így a két rendszer közt tapasztalható némi átfedés. A Novell ilyen jellegű szolgáltatásai azonban általánosak, míg a PROTECT kifejezetten dBASE specifikus védelmi rendszer létrehozására alkalmas.

7.4. A PROTECT védelmi szintjei

A PROTECT az alábbi három adatbiztonsági szint létrehozását és karbantartását valósítja meg:

- a bejelentkezés ellenőrzése (login)
- a fájl- és mező hozzáférések ellenőrzése
- az adattitkosítás

Az alábbi táblázat összefoglalva mutatja be a három típust, az egyes típusok eredményeit, valamint azt, hogy mit kell megadnunk létrehozásukkor.

Típus	Mit kell megadni	Eredmény
Login	A felhasználó neve és jelszava	A dBASE IV korlátozott elérése
Fájl- és mező-hozzáférés	Elérési szintek	Fájlok és mezők korlátozott elérése
Adattitkosítás	Felhasználó és fájl csoportok	Az adatok automatikus kódolása, dekódolása

7.1 Táblázat A PROTECT védelmi szintjeinek összefoglalása

A felsorolt védelmi szintek prioritási viszonyban állnak egymással, ezért létrehozási sorrendjük is a táblázatnak mefelelően kötött. Amíg a login-t nem adtuk meg, nem készíthetjük el az adatok elérésére vonatkozó korlátozásokat, s hasonlóan, míg az említett két szint nem létezik, nem hozhatjuk létre a harmadikat, a titkosítást. A teljes három szint megadása egyébként nem kötelező, így sok esetben elegendő lehet csupán a login kitöltése is.

7.4.1. A bejelentkezési szint (login)

Ha a védelmi rendszer aktív, a felhasználók nem tudják elérni a dBASE IV szolgáltatásait addig, amíg egy érvényes login-t meg nem adnak. A login a következő három részből tevődik össze: csoport név, felhasználó név (login név) és a jelszó.

7.4.2. A fájl- és mező hozzáférések szintje

7.4.2.1. A fájlok elérése

Az adatfájlok és mezőik elérését két irányból, egyrészt a felhasználó oldaláról, másrészt a fájlok felől megközelítve szabályozhatjuk, s így azt, hogy mely felhasználó mely adatcsoportokon milyen műveleteket végezhet, a két feltételrendszer sajátos viszonya határozza meg.

A felhasználói oldal:

Minden felhasználónak rendelkeznie kell egy ún. elérési szinttel, amely nem más, mint egy 1 és 8 közé eső egész szám. Ez tulajdonképpen csak arra szolgál, hogy segítségével kategóriákba soroljuk a felhasználókat. Ez azt jelenti, hogy azonos elérési szinttel bíró felhasználók nyilván azonos kategóriába tartoznak a fájlok elérését tekintve. Minél kisebb egy felhasználó elérési szintje, annál több előjoggal rendelkezik. A kategóriákba sorolást a rendszergazda végzi el.

A fájlok felőli megközelítés:

Minden fájlhoz elkészíthetünk egy ún. előjog-táblázatot, amelyben az adott fájlon végrehajtható műveletek körét szabályozhatjuk, mégpedig úgy, hogy minden művelethez szintén egy 1 és 8 közé eső egész számot rendelünk. Egy fájlon az itt felsorolt műveleteket végezhetjük el:

- az adatok olvasása (olvasási előjog)
- az adatok módosítása (módosítási előjog)
- új adatok hozzáfűzése (bővítési előjog)
- az adatok törlése (törlési előjog)

Sejthető, hogy végülis valamilyen módon ezek a számok teremtik meg a kapcsolatot a felhasználói elérési szintek és a fájlokhoz rendelt előjogok között.

Nézzük meg ezt pontosabban:
A műveleti előjogokhoz rendelt szám tulajdonképpen az adott művelet elvégzésére jogosult **felhasználói kategóriák** kijelölésére szolgál. Minden felhasználó, akinek elérési szintje kisebb vagy egyenlő a bármely fájlra vonatkozó tetszőleges művelethez rendelt számnál, végrehajthatja a szóbanforgó műveletet. Ebből azonnal következik, hogy a fájlokra vonatkozóan a 8-as szint a legkevésbé korlátozó. Az érthetőség kedvéért lássunk egy példát:

Ha egy fájl törlési előjogát 2-re, bővítési előjogát 1-re, a másik kettőt pedig 8-ra állítjuk be, akkor minden olyan felhasználó, aki 1-es vagy 2-es elérési szinttel rendelkezik, törölhet rekordokat, új rekordokat viszont csak az 1-es kategóriájú felhasználók fűzhetnek a fájlhoz. A 8-as szint miatt olvasni és módosítani minden felhasználó jogosult.

Az eddigieket összefoglalva elmondható, hogy a felhasználói kategóriák (elérési szintek) közül az 1-es rendelkezik a legtöbb előjoggal, azaz elméletileg (lásd.: Megjegyzés) bármely fájlon bármely művelet elvégzésére jogosult, hiszen az 1 bármely 1 és 8 közé eső számnál kisebb vagy egyenlő. A számok rendezettségéből adódó precedencia viszonyok miatt az elérési szint számának növekedése az előjogok körének csökkenését vonja maga után, így tehát a 8-as kategória lehetőségei a legkorlátozottabbak (a fájlokra ez természetesen fordítva igaz).

Megjegyzés:
Említettük, hogy az 1-es elérési szinttel rendelkező felhasználók elméletileg bármely fájlhoz tetszőleges módon hozzáférhetnek. Magyarázatra szorul az "elméletileg" kifejezés használata. Azt is írhattuk volna, hogy "bizonyos feltételek mellett" igaz az állításunk. Alapvetően két ilyen feltételt kell megemlítenünk:
- A hálózati szoftver (esetünkben a Novell) tárgyalásánál szóba kerültek a fájl jellemzők is. Nos, a Novell NetWare által beállított jellemzők (pl. a "csak olvasható" R/O) magasabb prioritásúak a dBASE IV által használtaknál. Ez azt jelenti, hogy ha egy fájl a Novell szerint "csak olvasható" jellemzővel ellátott, akkor a dBASE IV-ben hiába engedélyezzük a fájl módosítását, törlését vagy bővíthetőségét.
- A következő alfejezetben lesz szó a felhasználói csoportokról és az adatok titkosításáról. Az ott elmondottak alapján egy felhasználó csak a saját csoportjához tartozó fájlokat érheti el. Így az egyéb csoportokhoz rendelt fájlokat az 1-es elérési szinttel rendelkező felhasználók sem érhetik el.

7.4.2.2. A mezők elérése

Míg a fájlok elérését a felhasználói kategóriák és az adott fájlhoz rendelt előjog-táblázat egymáshoz való viszonya határozta meg, addig a mezők hozzáférhetőségét minden egyes elérési szintre külön kell (illetve lehet) meghatározni. A dBASE IV három, mezőkre vonatkozó előjogot különböztet meg:

- olvasás/írás (FULL) ez az alapértelmezés;
- csak olvasás (R/O);
- sem írás, sem olvasás (NONE) a mező "láthatatlan"

A mezőelőjogok létrehozásával tulajdonképpen a fájlelőjogok által meghatározott hozzáférési lehetőségeket korlátozhatjuk tovább. Például ha egy fájlra 2-es szintű módosítási előjogot szabtunk meg, ez azt jelenti, hogy az 1-es, 2-es kategóriájú felhasználók módosíthatják az egész fájlt. De ha az 1-es elérési szintre az egyik mezővel kapcsolatban az R/O előjogot állítjuk be, akkor ettől kezdve az 1-es szintű felhasználó ezt a mezőt csak olvashatja, függetlenül attól, hogy a fájlelőjog táblázata módosítási lehetőséggel ruházza fel. A 2-es kategóriájú felhasználó természetesen továbbra is rendelkezik az egész fájl módosítási jogával. Másrészt az is igaz, hogy a fájlelőjogok magasabb precedenciával rendelkeznek, mint a mezőelőjogok, azaz, ha valamely művelethez tartozó fájlelőjog korlátozza a felhasználói kategóriákat, akkor a korlátozás alá eső elérési szintek mezőkhöz való hozzáférése is ehhez igazodik, függetlenül a megadott mezőelőjogoktól.

A most bemutatott három fogalom (elérési szint, fájlelőjogok, mezőelőjogok) szemléletessé tétele érdekében célszerű ezeket táblázatos formában ábrázolni egy pél-

da keretein belül. E táblázatok elkészítése, a könnyebb áttekinthetőség végett, hálózati alkalmazások gyakorlati tervezésekor is ajánlott.

Példa:

Tegyük fel, hogy egy állománnyal kapcsolatban a következőket szeretnénk kikötni:

- csak a 4-nél kisebb vagy egyenlő kategóriájú felhasználók módosíthatnak benne;
- az "M1" nevű mező csak a 7-es kategóriájú felhasználók számára legyen látható;
- az "M2" mezőt csak a 2-es típusú felhasználók módosíthatják.

Az előjog-táblázat a következőképpen fog kinézni:

Fájl-előjogok	Felhasználói elérési szintek 1	2	3	4	5	6	7	8
Olvasás							*	
Módosítás				*				
Bővítés							*	
Törlés							*	

A táblázatból látható, hogy csak a módosításra adtunk korlátozást, a többi funkciót bármely felhasználó végrehajthatja.

A mezőelőjogok táblázata:

Mezőnevek	Elérési szintek 1	2	3	4	5	6	7	8
M1	NONE	NONE	NONE	NONE	NONE	NONE	FULL	NONE
M2	R/O	FULL	R/O	R/O	FULL	FULL	FULL	FULL

Az M1 mezőt egyáltalán senki sem módosíthatja, ugyanis az első táblázat szerint módosítást csak a 4-nél kisebb vagy egyenlő kategóriájú felhasználók végezhetnek egyáltalán, a második táblázat szerint viszont az "M1"-re vonatkoztatva azt a feltételt szabja, hogy csak a 7-es elérési szinttel rendelkező felhasználók módosíthatják. S mivel, mint fentebb már említettük, a fájlokra meghatározott előjogok a nagyobb prioritásúak, az "M1"-re vonatkozó írási/olvasási (FULL) előjog értelmét és érvényét veszti. Ugyanezért nincs értelme és hatása az "M2"-re vonatkozó FULL előjogoknak az 5,6,7,8-as kategória esetén (bár ez az alapértelmezés). Hasonlóan meggondolható, hogy az "M2"-re kiadott R/O előjogok viszont szükségesek ahhoz, hogy a harmadikként megfogalmazott feltétel teljesüljön.

7.4.2.3. A titkosítási szint

Az adatok titkosítása (encryption) azt jelenti, hogy a dBASE IV (amennyiben a védelmi rendszer aktív,) automatikusan olyan formátumra konvertálja az adatfájlokat és a hozzájuk tartozó index- és memo-fájlokat, amely illetéktelenek számára olvashatatlan. Ahhoz, hogy definiálni tudjuk az illetékességet, pontosítanunk kell a csoport fogalmát. Mint már említettük, minden felhasználót ell kell látni egy ún. login-nal, amely többek közt egy csoportnevet is tartalmaz. Ennek a jelentősége most válik kézzelfoghatóvá. Ugyanis a fájlelőjogok kiosztásakor minden egyes fájlhoz szintén meg kell adni egy csoportnevet is. Ily módon egy újabb kapcsolat teremthető a felhasználók és az adatok között azzal a megkötéssel, hogy egy felhasználó több csoport tagja is lehet, de egy fájlt csak egyetlen csoporthoz rendelhetünk.

Most már visszatérhetünk az illetékesség kérdéséhez. A titkosított fájlok tartalmát csak az a felhasználó tudja elolvasni, aki ahhoz a csoporthoz tartozik, amelyet az adott fájlhoz rendeltek. Így tulajdonképpen az adatfájlok egymást kizáró részhalmazokra oszthatók a felhasználói csoportok által. A titkosítás feloldását, tehát az olvashatóság helyreállítását a dBASE IV a sikeres bejelentkezés után teszi meg, de csak azokra a fájlokra, amelyek a megadott csoporthoz tartoznak. Tehát egy titkosított fájl használatára illetéktelen az a felhasználó, akinek csoportja nem egyezik meg a fájlhoz rendelt csoporttal.

Ezzel a dBASE IV védelmi rendszerének mindhárom elemét megismertük. Összefoglalva az elmondottakat:

Az **első védelmi szint a bejelentkezés**. Ekkor dől el az, hogy ki használhatja egyáltalán a dBASE IV szolgáltatásait az adott hálózati környezetben a csoport, a név, és a jelszó megadása által.

Sikeres bejelentkezés után a dBASE IV a megadott csoporthoz tartozó fájlokat dekódolja, így azok (és csak azok) a bejelentkezett felhasználó számára olvashatóvá válnak.

A **fájl- és mezőelőjogok** meghatározzák azt, hogy a dekódolt fájlok közül melyeken milyen műveletek elvégzése megengedett.

7.5. A DBSYSTEM.DB fájl

Mint azt már említettük, a védelmi rendszert a PROTECT utasítással hozhatjuk létre. Az ezzel kapcsolatos információkat a dBASE IV a DBSYSTEM.DB fájlban tárolja. Minden felhasználóhoz hozzárendeljük a következő adatokat:

- login név;
- teljes név;
- jelszó;
- csoportnév;
- elérési szint.

Ezeket együtt **felhasználói profil**nak nevezzük. Egy felhasználói profil felel meg a DBSYSTEM.DB egy rekordjának. A fájlt a PROTECT hozza létre és tartja karban titkosított formában. Nem dekódolható, a benne rejlő információkat ezért csak a rendszergazda képes megjeleníteni.
Ha a dBASE IV indításakor ez a fájl létezik, a bejelentkezési eljárás automatikusan elindul.

Megjegyzés:
A fájlokra vonatkozó védelmi feltételeket (fájl- és mezőelőjogok, titkosítás) a dBASE IV más módon tartja nyilván, ezért a DBSYSTEM.DB törlése, módosítása az említettekre hatástalan. Ugyanakkor azt is el kell mondanunk, hogyha csak a bejelentkezési szintet aktivizáltuk, a DBSYSTEM.DB törlése a védelmi rendszert kikapcsolja.

7.6. A PROTECT használata

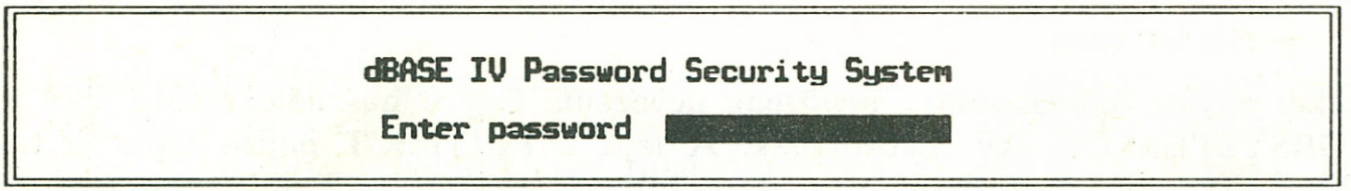
A PROTECT (a dBASE III PLUS rendszernél még külön program volt) szerves része a dBASE IV-nek, és mind hálózatos, mind egyfelhasználós környezetben működtethető. Mint említettük, használata opcionális (ha nem használjuk, az említett védelmi szintek nem jönnek létre, így az adatok bárki által tetszőleges módon elérhetőek), de egyszeri használata után az adatok hozzáférhetősége a létrehozott védelmi rendszer ellenőrzése alá kerül. A védelmi rendszer létrehozása az alábbi munkafázisokra tagolható:

- A PROTECT indítása
- A rendszergazda jelszavának kijelölése
- A felhasználói profilok létrehozása
- A fájl- és mezőelőjogok meghatározása
- A védelmi információk mentése

7.6.1. A PROTECT indítása.

A PROTECT csak akkor indítható el, ha minden adatfájl le van zárva. Az indításnak két módja is lehetséges: egyrészt az interpreterben beírt

PROTECT karaktersorozattal, másrészt a Control Center Tools menüjének Protect Data funkciójának kiválasztásával. Mindkét esetben az alábbi képernyő jelenik meg.



7.3. ábra dBASE IV bejelentkezési képernyő

Ez a képernyő a védelmi rendszer kialakítását végző személy (általában a rendszergazda vagy másnéven adminisztrátor) azonosítására szolgál. Egy legfeljebb 16 hosszúságú, tetszőleges alfanumerikus karaktereket tartalmazó jelszót kell megadnunk. Ez az **adminisztrátori jelszó.**

A kis- és nagybetűket nem különbözteti meg a rendszer, azok tehát egyenértékűek, s a begépelt jelszó természetesen nem látható. Az adminisztrátori jelszó megállapításakor a legnagyobb biztonság érdekében (a jelszó próbálgatásokkal való véletlen eltalálását kizárandó) célszerű a rendelkezésünkre álló mind a 16 karaktert kihasználni. A PROTECT első indítításakor (pontosabban fogalmazva akkor, ha a DBSYSTEM.DB nevű fájl nem létezik) a jelszó megadása után az alábbi üzenet jelenik meg:

Please reenter password to confirm.

Ez azt jelenti, hogy a jelszót (megerősítés végett) újra be kell írni. Minden további bejelentkezés alkalmával viszont már csak egyszer kell megadnunk a jelszót (ha a DBSYSTEM.DB létezik). Az érvényes jelszó megadására három lehetőségünk van. Ha ezt harmadszorra sem sikerül elérnünk, a PROTECT az *Unauthorized login* hibaüzenet kiírása után befejeződik.

Az adminisztrátori jelszó ismerete nélkül a védelmi rendszer megváltoztatására nincs mód, mivel ez a jelszó a dBASE IV rendszeren belül hozzáférhetetlen. Ezért **a jelszót nem célszerű elfelejteni. Ha ez az eset mégis bekövetkezne,** nem marad más választásunk, mint a **DBSYSTEM.DB fájl törlése.** Ekkor ugyanis a

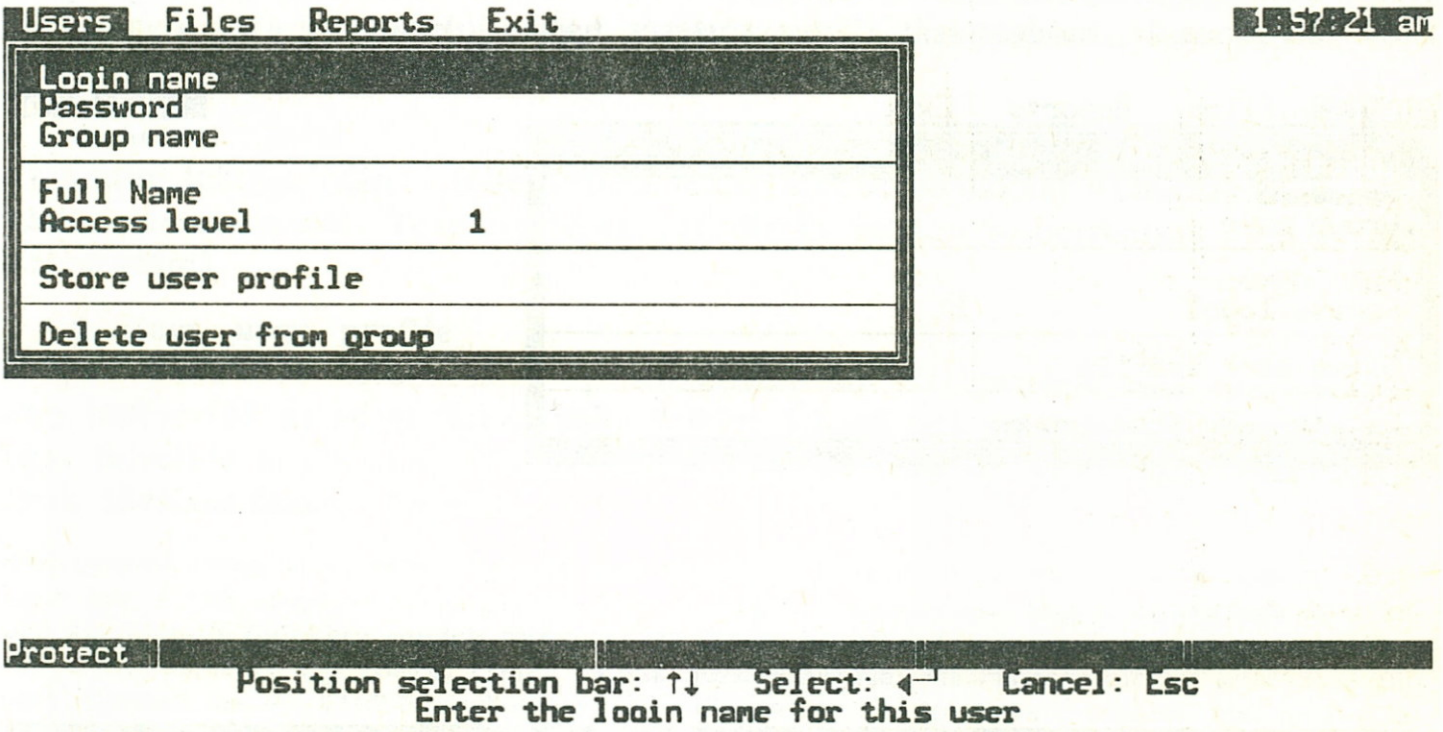
PROTECT úgy viselkedik, mintha először indítanánk el, s ennélfogva persze a korábban fáradságos munkával megkonstruált védelmi rendszerből semmi sem marad.

Vigyázat!

A titkosított állományok a DBSYSTEM.DB törlése után végképp hozzáférhetetlenekké válnak, még akkor is ha új védelmi rendszert építünk fel. Ezért a jelszó elfelejtése előtt javasoljuk a titkosított állományok eredeti formába történő helyreállítását. (SET ENCRYPTION OFF, USE, COPY TO)

7.6.2. A PROTECT munkafelülete

Az adminisztrátori jelszó sikeres megadása után a PROTECT szolgáltatásai már korlátozás nélkül rendelkezésünkre állnak, s ezeket egy áttekinthető, logikailag jól csoportosított menürendszeren keresztül aktivizálhatjuk. A képernyő felosztását szemlélteti s az alábbi ábra.



7.4. ábra A PROTECT képernyőszerkezete

Az első sorban (menüsor) találhatjuk a PROTECT négy fő funkciójának nevét. Ezek sorrendben:

Users : felhasználói profilok kialakítása

Files : fájl- és mezőelőjogok

Reports : a védelmi információk megjelenítése

Exit : a kilépés módozatai

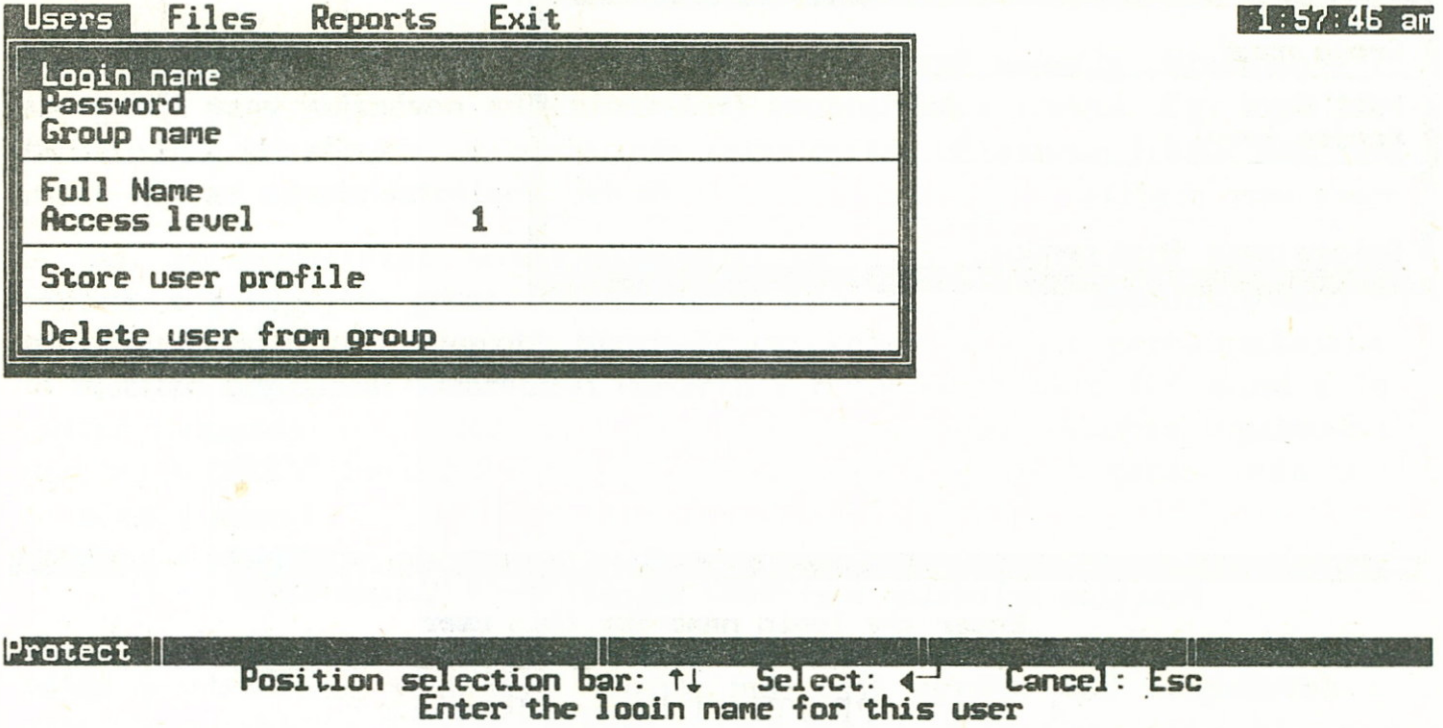
A 2-22. sorokban a fenti négy fő funkcióhoz tartozó almenük jelennek meg. Ezek szerkezete és használata megegyezik a dBASE IV más részeiben alkalmazott menükével. Ha ettől mégis el kell térni, azt külön megemlítjük.

A 23. sor (státusz) a billentyűzet állapotát jelzi ki.

A 24. sorban a menük használatára vonatkozó általános tudnivalókat láthatjuk, míg a 25. sor az aktuális menüpont használatával kapcsolatos információkat mutatja meg.

7.6.3. A felhasználói profilok létrehozása - Users

A felhasználói profil, mint tudjuk, nem más, mint az adott felhasználóra vonatkozó, a védelmi rendszerrel kapcsolatos információk gyűjteménye, amelyet a DBSYSTEM.DB fájlban tárol a dBASE IV. A felhasználói menüvel (Users) e profilok létrehozását, módosítását illetve törlését hajthatjuk végre.



7.5. ábra Felhasználói (Users) menü

7.6.3.1. Új felhasználói profil

Új felhasználói profilt az alábbi lépésekben hozhatunk létre:

- **Login name**

A kurzor az első menüsoron áll. Az **Enter** leütésével megnyílik egy input mező. Ide írjuk be a felhasználónak szánt login nevet (1-8 karakteres). A PROTECT ezt a bejegyzést csupa nagybetűssé alakítja. Végül az ismételt megnyomásával fejezzük be a login név megadását.

- **Password**

A felhasználóhoz rendelni kívánt jelszó megadása következik (1-16 karakteres).

- **Group name**

A csoportnév megadása hasonló módon történik (1-8 karakteres). Ezt a bejegyzést a PROTECT szintén nagybetűsre konvertálja.

- **Full name**

A felhasználó teljes nevének megadása nem kötelező (1-24 karakteres), de érdemes az adott felhasználó jellemzőinek pontosabb leírására felhasználni. Ide tulajdonképpen bármit írhatunk, mivel a felhasználó azonosításában nem játszik szerepet.

- **Access level**

Az **Enter** leütése után a felfele ill. a lefele mutató nyilakkal válasszuk ki a megfelelő elérési szintet. Természetesen ezt direkt módon is beírhatjuk (1-8 természetes szám).

- **Store user profile**

A most kialakított felhasználói profil elmentése. Itt igazából csak azt jelezzük, hogy befejeztük az adott felhasználó védelmi jellezőinek megadását. A profil tényleges felvétele a DBSYSTEM.DB fájlba az "Exit" menü "Save" vagy "Exit" pontjának kiválasztásakor történik.

Megjegyezzük, hogy a felhasználói profil csak akkor jön létre, ha legalább az első három jellemzőt (login név, jelszó, csoportnév) megadtuk. Már itt meg kell említenünk, hogy a csoportnév és a később kijelölendő, fájlokhoz rendelt csoportnevek fontos összefüggésben állnak, ti. a titkosított fájlok csak a csoportnevek egyezősége esetén válnak hozzáférhetővé. Másrészt az elérési szint és a fájlokra jellemző, szintén később meghatározandó-előjogok közti kapcsolatra is célszerű már most odafigyelni.

7.6.3.2. Felhasználói profil módosítása.

Ha a felhasználói menü első három elemét megadtuk, a PROTECT ellenőrzi, hogy ez tényleg egy új profil vagy korábban már meghatározott. Ha az utóbbi igaz, akkor az alábbi kérdés jelenik meg:

User already exists, do you want to edit ?

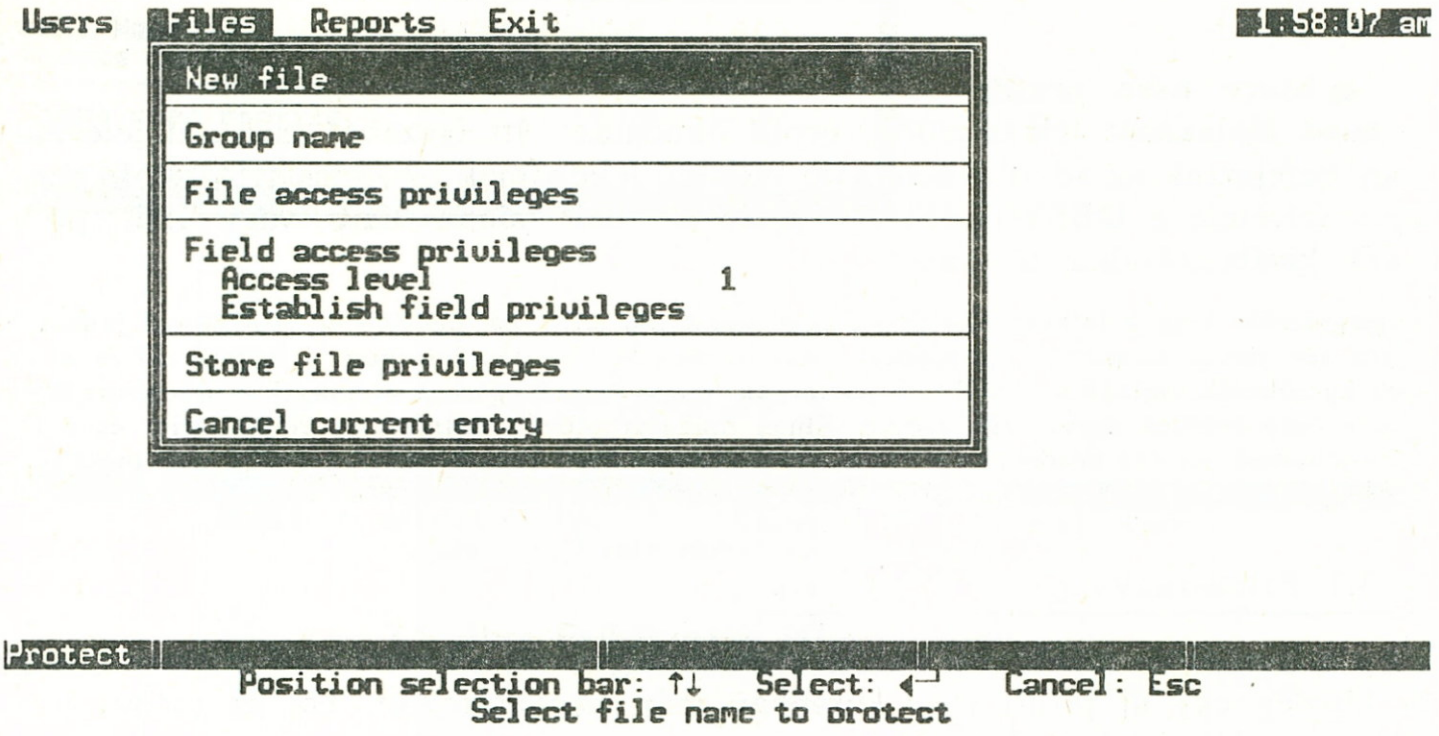
"Y" válaszra a menü többi sora a létrehozáskor vagy az előző változtatáskor meghatározott formában jelenik meg, és bármelyik sor módosítható. A megváltoztatott adatokat természetesen ugyanúgy el kell mentenünk, mintha egy új profilt alakítottunk volna ki. Bár lehetséges, mégsem ajánlott a csoportnév módosítása, ugyanis az eredeti névhez rendelt titkosított fájlokat semmilyen módon nem érhetjük el, ha a csoportnév megváltozott (persze ez alól is van kibúvó, mint látni fogjuk a SET ENCRYPTION utasításnál).

7.6.3.3. A felhasználói profil törlése

Itt is, akárcsak a módosításnál, meg kell adni a már korábban meghatározott profil első három tételét. A felhasználói menü utolsó sorát kiválasztva a profil törlődik.

7.6.4. A fájl- és mezőelőjogok meghatározása - Files

A **Files** menüben a fájl- és mezőelőjogok szintjét állíthatjuk be. Mint korábban már említettük, ezek szoros összefüggésben állnak a **Users** menüben kijelölt felhasználói elérési szintekkel. Az itt beállított védelmi információkat a PROTECT az adott adatfájl szerkezetében tárolja, ezért a DBSYSTEM.DB változtatása nem befolyásolja az így kialakított védelmet.



7.6. ábra Fájlok (Files) menü

Az ábra alapján ez a menü az alábbi három funkciót valósítja meg:

- adatfájl felhasználói csoporthoz rendelése
- fájlelőjogok szintjének beállítása
- mezőelőjogok megállapítása minden fájlelőjog szintre

Mielőtt e lépések részletezésébe kezdenénk, néhány fontos megjegyzést kell tennünk.

Mint tudjuk, a védelmi rendszer hatáskörébe tartozó adatfájlok titkosítottak lesznek, így azokat csak a megfelelő csoportba tartozó felhasználók érhetik el. Nos, ez a titkosítás egy konkrét fájl esetén akkor jön létre, ha a **Files • New file** menüben kiválasztjuk és kijelöljük tárolásra (**Store file privileges**) és az **Exit** menüben elmentjük (Exit vagy Save).

Ha ezenkívül semmi egyebet nem teszünk, akkor az előjogszintek alapértelmezés szerinti értékeit fogadjuk el, amelyek gyakorlatilag semmiféle korlátozást nem jelentenek az adott fájlra nézve, ugyanis ezek az értékek fájlelőjogok esetén 8-ra, mezőelőjogok esetén pedig FULL-ra vannak beállítva. Ezesetben a fájlok elérését csak a hálózati rendszerben (tehát a dBASE IV-en kívül, pl. a Novell által) beállított paraméterek szabályozzák.

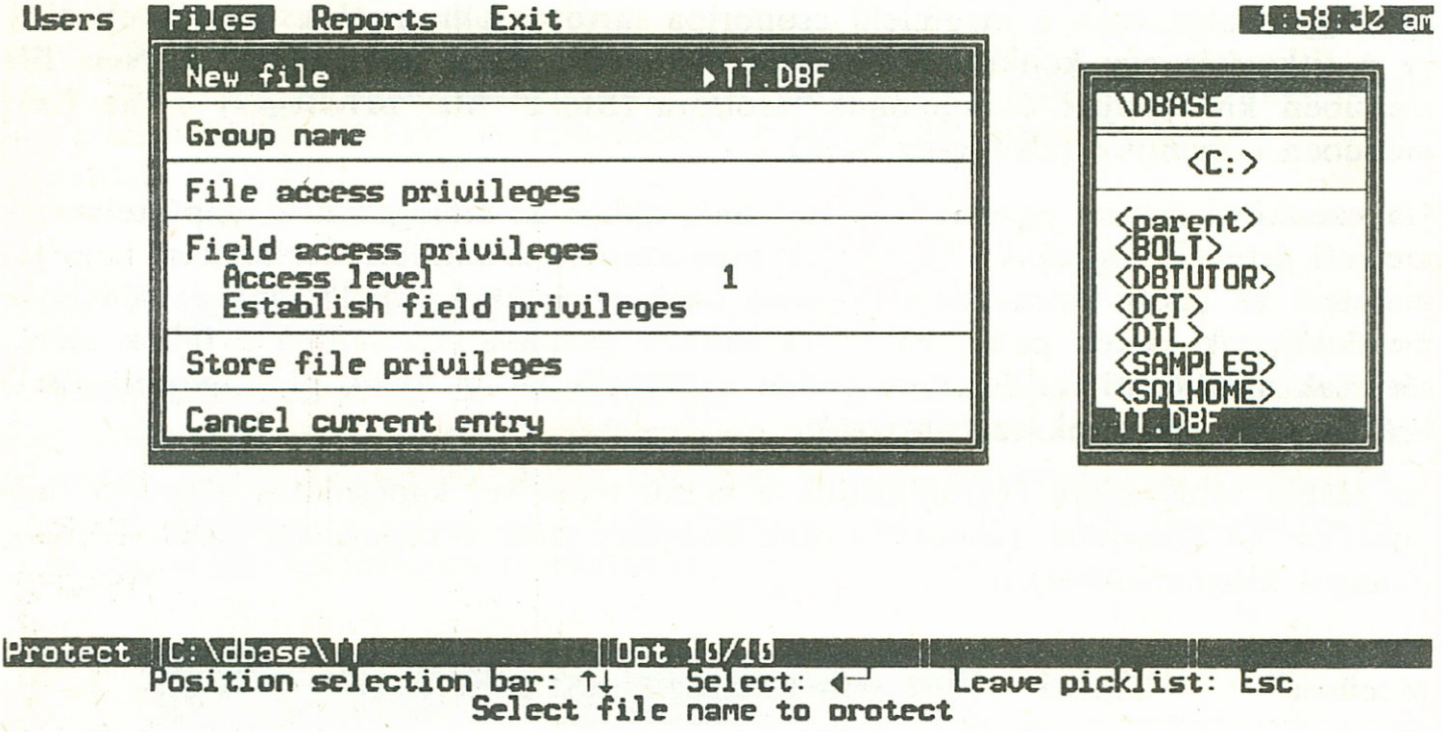
Az alábbi táblázatban összefoglaljuk a menü tételeivel kapcsolatos alapvető tudnivalókat (a könnyebb azonosíthatóság kedvéért most a képernyőn látható eredeti angol kifejezésekkel).

Menüsor	Típusa	Értéke	Alapértelmezés
New file	karakter	a választott fájl neve	az aktuális meghajtó azonosítója
Group name	karakter		-
Read privilege	egész	1-8	8
Update privilege	egész	1-8	8
Extend privilege	egész	1-8	8
Delete privilege	egész	1-8	8
Access level	egész	1-8	1
Field access privilege	-	FULL R/O	FULL NONE

Ezek után vizsgáljuk meg az egyes menüpontokat részletesen. A menütételek sorrendje egyben kijelöli a végrehajtási sorrendet is.

7.6.4.1. Fájlkijelölés - New file

Ezzel a menüponttal jelölhetjük ki azt a fájlt, amelynek védelmi jellemzőit be akarjuk állítani. Az megnyomása után egy ablakban megjelenik egy fájlnév-lista. Ez a lista a rendszerben fellelhető összes adatfájl (*.dbf*) nevét tartalmazza, beleértve az alkönyvtárakat is. Az ablakon belül az inverz csík mozgatásával választhatjuk ki a megfelelő fájlt, s ha ez megtörtént, az ablak eltűnik, és a név bekerül a Files menü első sorába.



7.7. ábra Fájl kijelölése

Itt kell megjegyeznünk, hogy egyidőben legfeljebb 9 adatfájl védelmi rendszerét állíthatjuk be a File menüvel. Ha megpróbálnánk egy tizedik fájlt is bevonni, a következő hibaüzenet jelenne meg:

Too many files are open.

ami ugye annyit tesz, hogy túl sok fájlt akarunk egyidőben nyitva tartani. Ez persze nem túl meglepő, hiszen korábban mondtuk, hogy a **File** menü védelmi információi az aktuális fájl szerkezetében tárolódnak, így nyilván meg is kell azokat nyitni. Tehát a kilencedik fájl védelmi rendszerének létrehozása után feltétlenül válasszuk ki az Exit menü (lásd később) Exit vagy Save funkcióját (ekkor a szükséges információk tárolása és a fájlok lezárása megtörténik), s csak utána folytathatjuk a többi fájlelőjog-rendszer felépítését.

7.6.4.2. Fájlcsoport - Group name

Itt valósul meg a már kijelölt felhasználói csoportok és a fájlokból alkotott csoportok közti kapcsolat. Ez a 8 karakter tulajdonképpen bármi lehet, de igazából annak van értelme, ha a fájlokat logikai együvé tartozásuk szerint szervezzük csoportokba, s az így kialakított csoportot az itt magadott néven keresztül hozzárendeljük egy felhasználói csoporthoz (vö. 10.3.2.3)

Egy fájl csak egy csoporthoz rendelhető.

7.6.4.3. Fájl-előjogszintek - File access privileges

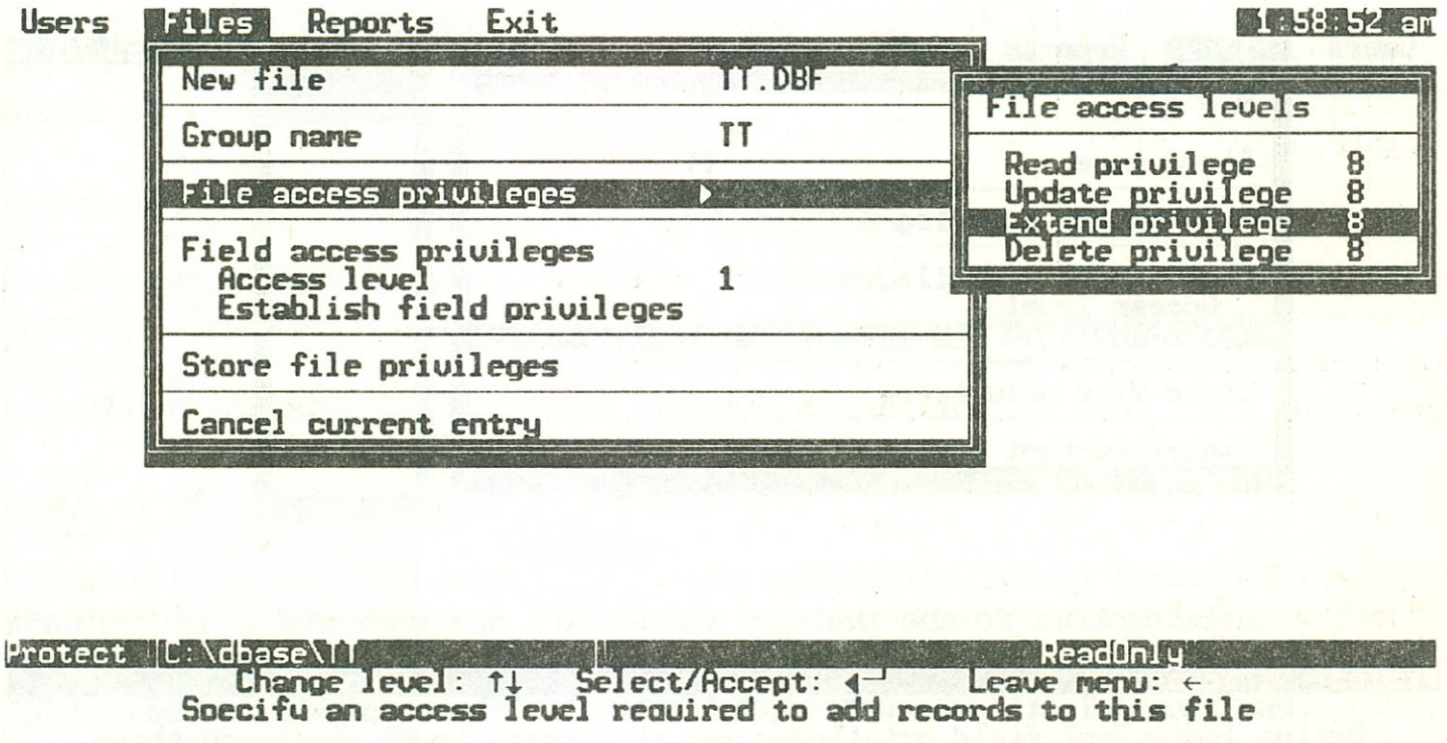
Ezt a tételt választva megjelenik egy **File access levels** című almenü, amelyben az alább felsorolt négy művelet előjogszintjét egymástól függetlenül lehet beállítani.

READ : a fájl tartalmának megtekintése (olvasás)

UPDATE : létező rekordok módosítása

EXTEND : új rekordok hozzáadása

DELETE : rekordok törlése a fájlból



7.8. ábra Fájl-előjogszintek beállítása

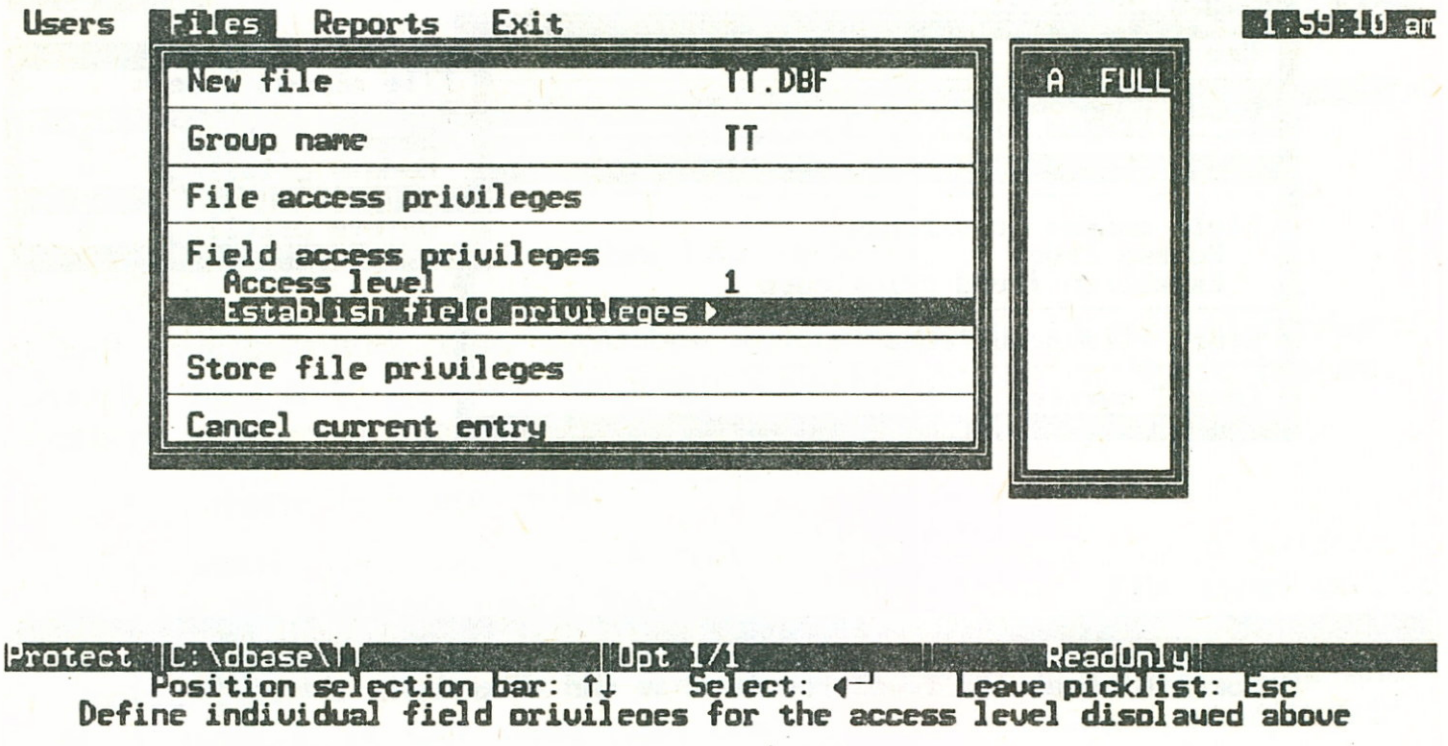
Az előjogszintek értéke egy-egy 1-8 közötti egész szám lehet, amelyet vagy a le/fel nyilakkal vagy közvetlen beírással állíthatunk be.

Ismételten meg kell jegyeznünk, hogy az itt látható elérési jogok nem bírálják felül az operációs rendszer által beállított jellemzőkat. Tehát ha egy fájl az operációs rendszer szerint R/O, akkor ezt csak olvasni lehet függetlenül attól, hogy a **Files** menüben esetleg elláttuk az UPDATE, EXTEND, DELETE előjoggal is.

Emlékeztetőül felidézzük, hogy a felhasználókhoz rendelt hozzáférési szint volt az itt meghatározandó fájlelőjogok fogalompárja (lásd: Users menü). Láttuk, hogy egy felhasználó egy fájlon a fenti négy művelet valamelyikét akkor végezheti el, ha hozzáférési szintje kisebb vagy egyenlő, mint az adott műveletre beállított előjogszint. Tekintettel arra, hogy a fájlelőjogok közül a 8-as (ez az alapértelmezés is) a legkevésbé korlátozó, leszögezhetjük, hogy a felhasználókhoz rendelt hozzáférési szintek közül az 1-es a legerősebb. Fordítva, ha azt szeretnénk, hogy egy konkrét fájlhoz bárki, bármely műveletre hozzáférhessen, akkor az előjogszintek alapértelmezését (8) kell meghagynunk, mivel ennél bármely lehetséges felhasználói hozzáférési szint kisebb vagy egyenlő. Hasonló meggondolások alapján egy fájlra nézve az 1-es a legkorlátozóbb előjogszint.

7.6.4.4. Mezőelőjogok - Field access privileges

A mezőelőjogokat minden felhasználói elérési szinthez külön-külön kell megállapítani. Ezért először mozgassuk a sávkurzort az **Access level** menütételre, majd nyomjuk meg azt. A megfelelő hozzáférési szint (1-8) beállítása után menjünk egy sorral lejjebb az **Establish field privilege** címszóra, majd ismét nyomjuk meg az Entert. Ekkor a **Files** menü mellett megjelenő ablakban az adott fájl mezőneveit láthatjuk az alapértelmezés szerinti FULL bejegyzésekkel.



7.9. ábra Mezőelőjogok megállapítása

A mezőelőjogok jelentése:

FULL : Teljeskörű hozzáférést tesz lehetővé, az adott mező írása, olvasása is megengedett.
R/O : A mező csak olvasható.
NONE : A mező az adott felhasználói kategória számára egyáltalán nem hozzáférhető, azaz úgy tűnik, mintha nem is lenne benne a fájlban.

Az előjogok megváltoztatása úgy történik, hogy a sávkurzort a módosítani kívánt sorra állítjuk, majd a *szóköz* többszöri megnyomásával váltogatjuk az ott megjelenő, előre beállított értékeket. Az ablakot a balra mutató nyíl vagy az **Esc** billentyű segítségével hagyhatjuk el. Ne feledkezzünk meg arról, hogy ezt az eljárást minden felhasználói elérési szintre meg kell ismételnünk.

A fájlelőjogok szélesebb körű korlátozást jelentenek a mezőelőjogoknál. Például ha a fájlelőjog READ (amely az adott fájlra nézve csak az olvasást engedélyezi), akkor az érintett felhasználói kategóriákban nincs értelme a FULL mezőelőjognak egyetlen mező esetén sem. Bizonyos egész fájlt érintő dBASE IV utasítások (pl.: DELETE ALL, PACK, ZAP) ellen időnként célszerű megvédenünk a különösen fontos adatfájlokat. A mezőelőjogok R/O-ra vagy NONE-ra állítása a fájlelőjogok korlátozása nélkül azonban természetesen hatástalan.

A **Store file privileges** menüpontot válasszuk ki, ha úgy érezzük, hogy már valamennyi fájl- és mezőelőjog helyes beállítását elvégeztük, és ezeket tárolni akarjuk. Itt is megjegyezzük, hogy ez a menüpont csak megjelölést végez, a fizikai tárolás az **Exit** menüben történik.

7.6.4.5. A kijelölés törlése - Cancel current entry

Az előjogok meghatározása (akár első beállításról, akár módosításról van szó) bármely pillanatban megszakítható, ha az utolsó sorra állva engedélyezzük a menüpont végrehajtását. Ekkor természetesen csak az aktuális adatok szűnnek meg, a korábban rögzített és eltárolt információk megmaradnak.

7.6.4.6. A fájl- és mezőjogok módosítása

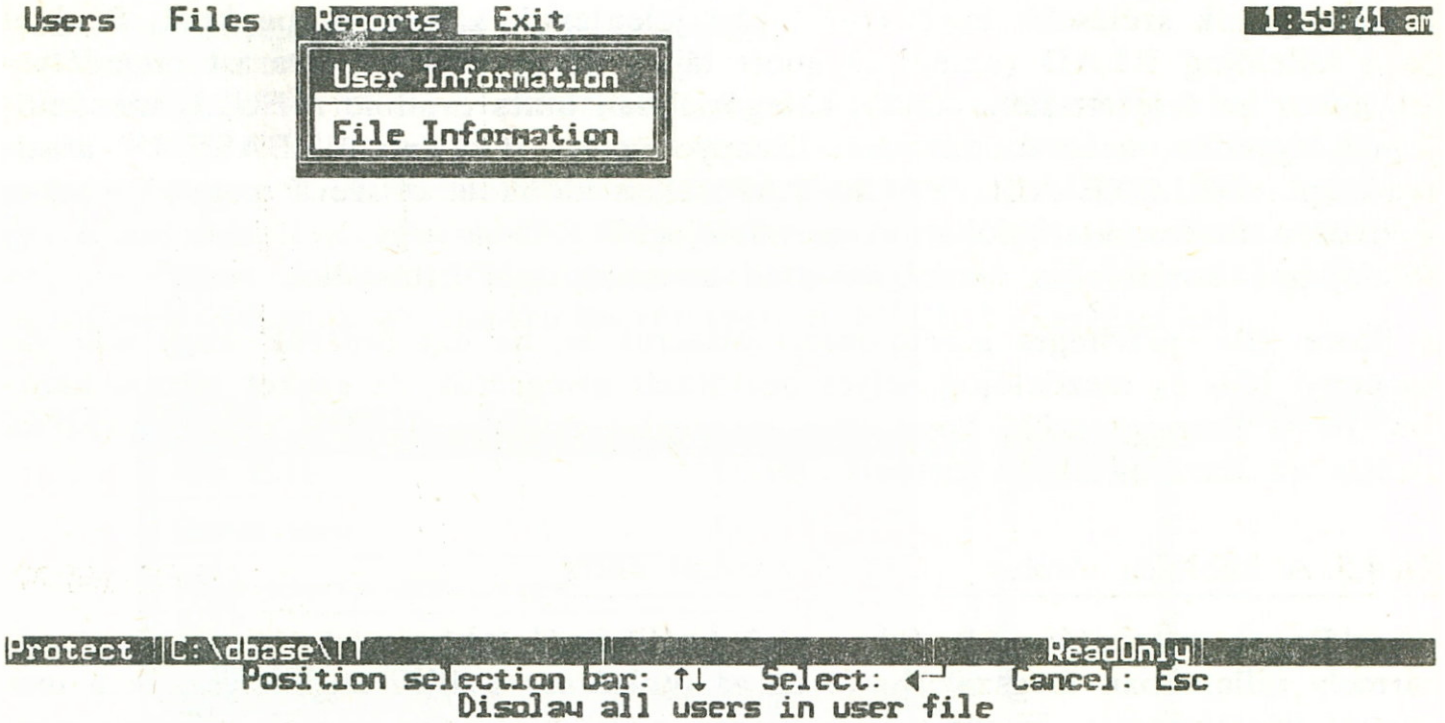
Végezetül még egy rövid megjegyzés erejéig szólunk arról, hogy hasonlóan a **Users** menühöz, itt is lehetőség van a korábban rögzített adatok módosítására. A PROTECT ugyanis minden fájl kiválasztáskor (**New file**) ellenőrzi azt, hogy az adott fájlhoz meghatároztunk-e már korábban előjogrendszert. Ha igen, akkor a menü többi eleme az aktuális tartalommal töltődik fel az alapértelmezések helyett, és azok bármely eleme szabadon módosítható. Amennyiben a változtatásokat ismét eltároljuk a **Store file privileges** megválasztásával, a következő kérdés jelenik meg:

fájlnév.CRP already exists, overwrite it ? (Y/N)

"Y"-t válaszolva a PROTECT tárolja a módosításokat. Ne feledkezzünk meg azonban arról, hogy a fizikai tároláshoz az **Exit** menü megfelelő sorát is végre kell hajtanunk.

7.6.5. A védelmi információk megjelenítése - Reports

A PROTECT a védelmi információkat a DBSYSTEM.DB-ben tárolja. Ez a fájl titkosított, tehát sem a rendszergazda sem pedig más felhasználók nem tudják elolvasni a tartalmát. Ez pedig kellemetlen lehet olyan esetekben, amikor egy felhasználó elfelejti valamelyik login-értékét (név, csoport, jelszó). Szerencsére ezt a hiányosságot szünteti meg a Reports menü.



7.10. ábra Védelmi információk megjelenítése

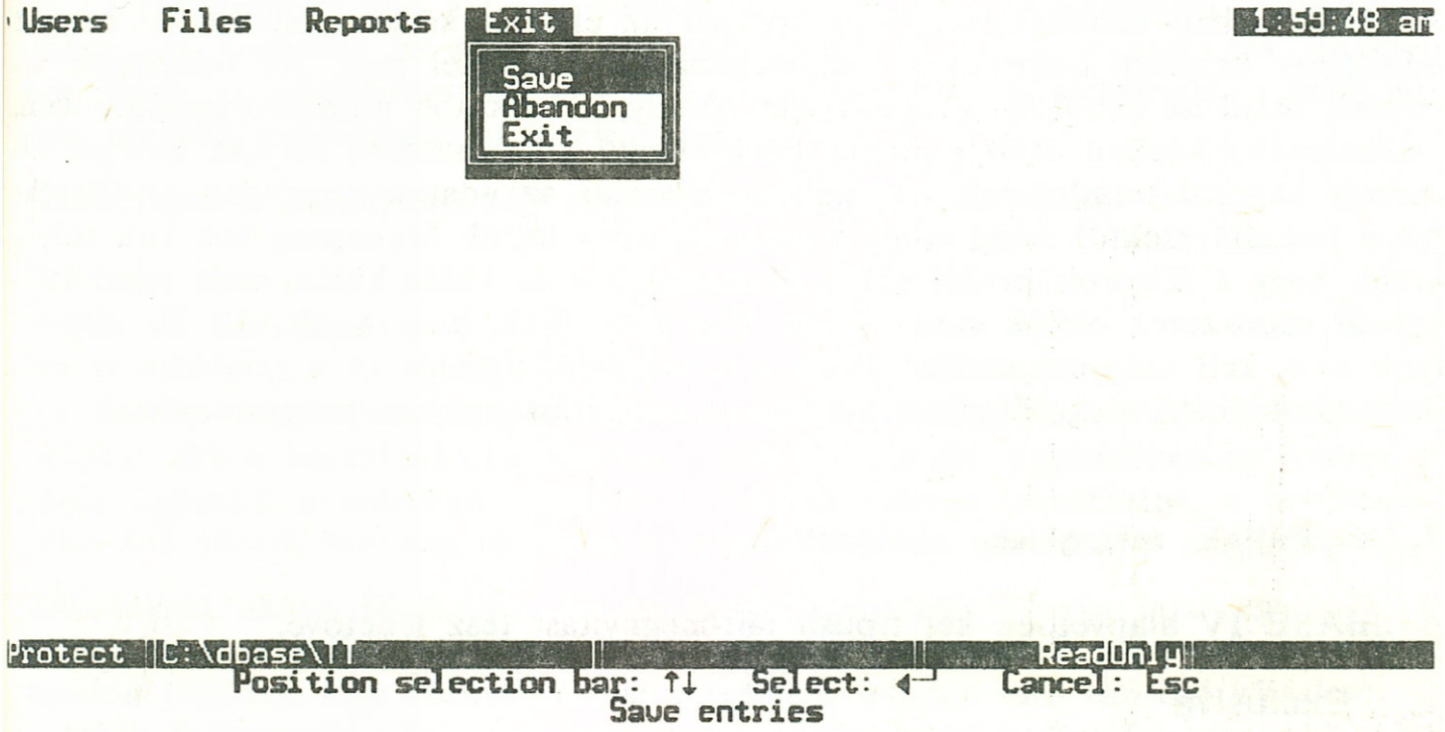
User information : választásával a rendszer összes felhasználójának a Users menüben beállítható adatai (jelszó, csoportnév, teljes név és az elérési szint) jelennek meg egy táblázatban. A *Send report to the printer ? (Y/N)* kérdésre "Y"-t válaszolva a nyomtatón jelennek meg az információk.

File information : Ezt választva először a **.crp** kiterjesztésű fájlok listája jelenik meg egy ablakban. Kiválasztva egy fájlt az ablakban, a következő üzenettel találkozunk: *Enter Group Name:* Ide a megfelelő csoportnevet kell beírnunk. Helyes csoportnév megadása esetén a fájlra jellemző védelmi információk (név, csoport, fájl- és mezőelőjogok) kívánság szerint képernyőre vagy nyomtatóra kérhetőek.

Emlékeztetőül megjegyezzük, hogy a .crp fájlokat a normál adatfájlokból a PROTECT hozza létre az adott fájlra vonatkozó védelmi információk tárolásakor. Ezek titkosított változatai a megfelelő adatfájloknak és csak az illetékes csoportba tartozó felhasználók érhetik el őket. A CRP egyébként az angol "encrypted" szóra utal.

7.6.6. A védelmi információk mentése - Exit

Mint már többször is szóba került, a felhasználói profilok és a fájl- és mezőelőjog rendszer **fizikai tárolása** a DBSYSTEM.DB-ben illetve a megfelelő adatfájl szerkezetében az **Exit** menü feladata.



7.11. ábra Védelmi információk mentése

Save : Az aktuálisan létrehozott vagy módosított, és a Users vagy Files menüben tárolásra kijelölt felhasználói profilok és előjogrendszerek fizikai tárolását valósítja meg. A felhasználói profilok a DBSYSTEM.DB fájlba kerülnek, míg az előjogrendszer beépül az érintett fájl szerkezetébe. A fájlok titkosított változata (.CRP) is ekkor jön létre. A PROTECT befejezése után mentsük el, majd töröljük ki az eredeti adatfájljainkat. Ezután a megfelelő **.crp** fájlokat nevezzük át **.dbf** kiterjesztésűre. A profilokat a védelmi rendszer létrehozása közben bármikor elmenthetjük, de az előjogrendszert minden nyolcadik fájl után kötelező menteni.

Abandon : A PROTECT elindítása óta létrehozott, módosított de el nem mentett profilok és előjogok törlése valósul meg itt. Azaz az aktuálisan előidézett változtatásokat tehetjük meg nem történtté ezzel a funkcióval.

Exit : Ezt a sort akkor használjuk, ha ki akarunk lépni a PROTECT-ből. Ha előzőleg a "Save" funkciót nem hajtottuk végre, akkor az ott leírtak automatikusan megtörténnek, mielőtt a PROTECT befejezné működését.

7.7. Programozás hálózati környezetben

Mint korábban már utaltunk rá, a hagyományos körülmények közt fejlesztett dBASE IV programok elvileg változtatás nélkül futtathatóak hálózati környezetben is. A gyakorlatban azonban nem célszerű ezt az eljárást követni, hiszen az osztott adatfájlok kezelése alapvetően más megközelítést követel meg. Az ütközések kivédése, majd az ebből fakadó holtpont-veszély, a hosszabb munkafolyamatok (ún. tranzakciók) közben esetleg mégis bekövetkező adatvesztések folytán fellépő eltorzult kapcsolatrendszerek korrigálása, valamint az adatok megóvása az illetéktelen hozzáférésektől mind-mind a programozóra hárul. Manapság sok vita folyik arról, hogy a felsorolt problémákat összességében és külön-külön mely adatbáziskezelő rendszerek oldják meg jobban. Úgy gondoljuk, hogy a dBASE IV alkotóinak nem kell szégyenkezniük ebben a versenyben, ugyanis ez a rendszer az említett problémák megoldásához hatékony támogatást nyújt a programozónak.

7.7.1. Fájlok megnyitása hálózatban

A dBASE IV alapvetően két típusú fájl-megnyitást tesz lehetővé.

Exclusive

Megnyitás kizárólagos módon. Ez esetben a megnyitott fájl csak az aktuális felhasználó számára hozzáférhető. A többi felhasználó a fájlt semmiféle módon nem nyithatja meg (olvasásra sem). Ha az adott fájlt már egy másik munkaállomáson (akár megosztott módon) korábban megnyitották, akkor a mi kizárólagos megnyitási kísérletünk nem fog sikerülni egészen addig, amíg a fájlt le nem zárták.

Shared

Megnyitás megosztott módon. Az így megnyitott fájl bárki által hozzáférhető írásra, olvasásra egyaránt (ha egyéb korlátozások nincsenek érvényben).

Kizárólagos módon rendszerint a felhasználó saját fájljait nyitjuk meg. Hálózatban ez a ritkább eset, de néhány dBASE IV utasítás ezt a fajta megnyitást feltételezi, ellenkező esetben az utasítás végrehajtása hibaüzenettel megszakad.

Ilyen utasítások az alábbiak:

COPY TAG/INDEXES	MODIFY STRUCTURE
INDEX	REINDEX
PACK	ZAP

A megnyitás módja szabályozható egyrészt az egész dBASE IV környezetet tekintve az EXCLUSIVE környezeti változó értékének megfelelő beállításával, másrészt egy-egy konkrét fájlra nézve a USE utasítás EXCLUSIVE opcióját használva.

Például a SET EXCLUSIVE OFF kiadása után minden megnyitás osztott lesz, de ha egy konkrét fájlt a USE <fájlnév> EXCLUSIVE paranccsal nyitunk meg, akkor ezt kizárólag az adott felhasználó érheti el. Talán természetes, de ideírjuk, hogy az adatfájlhoz tartozó index- és memo-fájlok megnyitási módja minden esetben megegyezik az adatfájléval.

Kizárólagosan megnyitott fájl esetén nincs szükség zárolásra.

7.7.2. A zárolásról

Az osztott adatkezelés alapvető problémája az ütközések kivédése. **Ütközésről** például akkor beszélünk, ha két vagy több felhasználó ugyanabban az időben kívánja ugyanazt a rekordot módosítani. Mint korábban olvashattuk, e probléma a rekordok illetve bizonyos esetekben egész fájlok ún. zárolásával megoldható.

Zárolásról akkor beszélünk, ha az egyik felhasználó "fogva tart" egy rekordot (fájlt) azzal a céllal, hogy mások ne kezdeményezhessenek írási műveletet a rekordon (fájlon). Egy sikeres zárolás meghiúsít minden más zárolási kísérletet, így a többi felhasználó várakozni kényszerül addig, amíg a zárolást meg nem szüntetik.

Rekordok zárolása az **RLOCK()**, míg **fájlok zárolása** az **FLOCK()** függvénnyel történik, s ezeket direkt zárolásnak nevezzük. Hangsúlyozzuk, hogy zárolt adatokat olvasni lehet, csak írni nem.

A zárolás **feloldására** az **UNLOCK** utasítás szolgál, de felold a fájl lezárása és a dBASE IV-ből történő kilépés is. Egy rekord zárolásakor az összes vele relációban lévő rekord zárolása is megtörténik (legfeljebb 50 rekord egyidőben). Ha a zárolás sikertelen, a dBASE IV hibaüzenettel tér vissza (programozás közben az RLOCK()/FLOCK() függvények visszaadott értékei alapján kell megtervezni a következő lépéseket). A SET REPROCESS utasítással meghatározható, hogy a dBASE IV hány alkalommal próbálkozzon a zárolással, mielőtt hibát jelezne.

Teljes-képernyős (full-screen) utasítások használata közben is mód van direkt rekordzárolásra. Az aktuális rekordon állva a **CTRL-O** billentyűk egyidejű lenyo-

mására a **zárolás** megtörténik, **feloldására** pedig a **CTRL-R** vagy **CTRL-C** billentyű-kombinációt használhatjuk.

A dBASE IV-ben **alapértelmezésben** minden olyan **művelet** előtt, amely **adatmódosulást eredményez**, a **zárolás automatikusan** megtörténik. A BROWSE és EDIT utasítások végrehajtásakor például csak az aktuális rekord zárolása történik meg (és az is csak akkor, ha a felhasználó nem kurzormozgató billentyűt üt le) azért, hogy más felhasználók zavartalanul hozzáférhessenek a többi rekordhoz.

Egész fájlt érintő műveletek előtt (pl. APPEND FROM) azonban magát a **fájlt zárolja automatikusan** a dBASE IV.

A műveletek befejezése után mindkét esetben **automatikusan fel is oldódik** a zárolás. Ez alól egyedül a *REPLACE utasítás* kivétel. Ugyanis ha az érvényességi köre RECORD, akkor a zárolás csak akkor oldódik fel, ha a következő utasítás nem egy újabb REPLACE. Erre azért van szükség, hogy az adott rekordra vonatkozó összes egymást követő REPLACE biztonságosan befejeződjék, mielőtt a rekordot más felhasználók számára felszabadítjuk (ugyanis az újabb REPLACE előtti újabb zárolás nem biztos, hogy sikerülne, és ez a rekord kitöltésének félbeszakadását jelenti). Az alábbi táblázatban összefoglaljuk azokat az utasításokat, amelyek **automatikus zárolást** hajtanak végre.

dBASE IV utasítás	Zárolás jellege	A SET LOCK hatással van-e rá
APPEND FROM	F	N
APPEND [BLANK]	R	N
AVERAGE	F	I
BROWSE	R	N
CALCULATE	F	I
CHANGE	R	N
COPY TAG/INDEX	F	I*
COPY [STRUCTURE]	F	I*
COUNT	F	I
DELETE/RECALL	FR	N
EDIT	R	N
INDEX	F	I*
JOIN	F	I*
LABEL	F	I
REPLACE	FR	N
REPORT	F	I
SET CATALOG ON	F	N
SORT	F	I*
SUM	F	I
TOTAL	F	I*
UPDATE	F	N

A táblázat jelölései:

F	fájl zárolás
R	rekord zárolás
FR	rekord/fájl zárolás
	az érvényességi körtől
	függően
I	Igen
N	Nem
*I**	Csak olvasáskor

A táblázat jobb szélső oszlopa magyarázatra szorul. A LOCK környezeti változó megfelelő beállításával bizonyos mértékig mi döntjük el, hogy az eddigiekben ismertetett automatikus zárolást megengedjük vagy letiltjuk. A SET LOCK OFF kiadása után a táblázatban felsorolt utasítások egy része (az "I"-vel jelöltek) nem hajt végre zárolást működésének megkezdése előtt. Ezek az utasítások csak olvassák a fájlokat. Az "I*" jelű utasítások olvasást/írást egyaránt végeznek, a "*" arra utal, hogy csak az olvasási fázisban szűnik meg az automatikus zárolás, mielőtt az írásra sor kerülne, az utasítás azonnal zárolja a rekordot vagy szükség szerint az egész fájlt a SET LOCK állásától függetlenül.

Programok írása közben arra kell törekednünk, hogy a munkaállomások lehetőleg minél kevesebbet várjanak egymásra. (A holtponthelyzetek általában a hibás zárolási stratégiából fakadnak.) Ez igazából az adatok zárolási időtartamának minimalizálását jelenti. Ezért ahol csak lehet, kerüljük egy egész fájl lefoglalását, mindig csak az éppen feldolgozás alatt álló rekordo(ka)t zároljuk, így biztosítva a többi rekord más felhasználók általi elérését. Figyeljünk továbbá arra is, hogy egy RLOCK()/FLOCK() és UNLOCK utasításpár között csak a valóban szükséges utasítások álljanak, tehát a zárolást amint lehet, oldjuk fel.

A *legnagyobb veszélyforrás* az, amikor a @ GET/READ utasításokkal közvetlenül a rekordba olvasunk. Az adatrögzítés amúgy is egy viszonylag lassú művelet. Képzeljük el, hogy két, egymástól távoli munkaállomáson folyik az adatbevitel, és mindkét rögzítő kislány ugyanazt a rekordot szeretné módosítani. Valamelyikük (a gyorsabb) zárolja a szóbanforgó rekordot, így a másik várakozni kényszerül. Tegyük fel, hogy az első kislány (aki fogva tartja a rekordot) abbahagyja a munkát és elmegy ebédelni. Világos, hogy ezért a másik sem tud dolgozni addig, amíg a rekord fel nem szabadul (amíg az ebédidő tart). Az ilyen és ehhez hasonló helyzetek elkerülése érdekében soha ne írjunk olyan programot, amely közvetlenül rekordba rögzíti az adatokat, még akkor sem, ha ez programozási szempontból kényelmesebbnek látszik. Ehelyett minden mezőnek feleltessünk meg egy-egy változót, ezekbe kerüljön az input, s így a rekordot csak a REPLACE idejére kell lefoglalnunk.

Megjegyezzük, hogy a rekordba rögzítés egyébként sem túl elegáns megoldás.

7.7.3. Gyakorlati tanácsok

A PROTECT bemutatásánál szó volt a titkosított fájlokról. Tudjuk, hogy tartalmukat csak a megfelelő csoportba tartozó felhasználók képesek elolvasni. Előfordulhat azonban, hogy bizonyos fájljainkat szeretnénk átadni más felhasználói csoportok tagjainak. Tekintettel arra, hogy a titkosított formához ők nem férhetnek hozzá, kell egy eszköz, amelynek segítségével a fájlok "titkosítatlan" változata visszanyerhető. Nos, ez az eszköz a **SET ENCRYPTION** utasítás. OFF állásban (ez az alapértelmezés is) ugyanis a COPY,JOIN,TOTAL utasítások eredménye-

ként keletkező fájlok már titkosítatlanok, tehát bárki által hozzáférhetőek lesznek.

Programírásnál célszerű a **CONVERT** utasítást használni, amely egy adatfájlt olyan formára alakít, amelyre néhány nagyon hasznos, a rekordok zárolásával kapcsolatos információkat szolgáltató függvény alkalmazható. A CONVERT a fájlt először elmenti **.cvt** kiterjesztéssel, majd létrehoz egy új adatfájlt, amely csak annyiban tér el az eredetitől, hogy tartalmaz egy **_dbaselock** nevű új mezőt is, amely a rekordról az alábbi információkat tárolja:

- ellenőrző szám
- utolsó zárolás időpontja
- utolsó zárolás dátuma
- a zárolást végrehajtó felhasználó login neve.

Az ily módon átalakított fájl rekordjain az alábbi két függvény értelmezett:

CHANGE()

Ezzel a függvénnyel megtudhatjuk, hogy egy adott rekord tartalma megváltozott-e a fájl megnyitása óta. A függvény a _dbaselock mező megfelelő részében található ellenőrző számot hasonlítja össze a mezők aktuális értéke alapján (egy belső algoritmus szerint) képzett értékkel. Ha eltérést tapasztal, akkor .T.-val tér vissza, ami azt jelenti, hogy a rekord tartalma megváltozott.

LKSYS()

A _dbaselock mező adatai alapján megfelelő paraméterezéssel visszaadja a zárolás dátumát, idejét, végrehajtójának nevét.

Emlékezzünk arra, hogy a PROTECT segítségével minden felhasználóhoz hozzárendelhetünk egy elérési szintet. Az **ACCESS()** függvény ezt a számot adja vissza nekünk. Ez egy hasznos dolog, mert így a programozás szintjén egy más jellegű adatvédelmet is megvalósíthatunk úgy, hogy az elérési szinttől függően bizonyos programrészek végrehajtását nem engedélyezzük.

Példa:

```
Accept "Válassza ki a megfelelő funkciót" To valasz
Do Case
      Case valasz = "1"
            If Access() < 3
                  Do AlProgam
            Else
                  ? "A végrehajtás nem megengedett !!!"
                  Wait
                  Return
            EndIf
      Case stb...
```

Ha a dBASE IV induláskor nem találja a DBSYSTEM.DB fájlt, akkor a bejelentkezési (login) eljárás elmarad. Ilyen esetben az ACCESS() mindig 0-át ad vissza eredményül. A 0 elérési szintű felhasználó viszont nem tudja elolvasni a titkosított fájlokat. Ezért egyrészt a DBSYSTEM.DB fájlt a dBASE IV alkönyvtárában célszerű tartani, másrészt a felhasználók elérési szintjét még a program elején tisztázni kell az ACCESS() segítségével.

Jó tudni azt, hogy a programunk éppen egyfelhasználós vagy hálózati környezetben fut (ugyanis az első esetben jó néhány feltételvizsgálat, utasításcsoport feleslegessé és értelmetlenné válik). Ezt az információt szolgáltatja nekünk a **NETWORK()** függvény, amely .T.-t ad vissza, ha hálózati környezetben vagyunk.

Végezetül megemlítjük, hogy a dBASE IV-ben (az eddig ismert dBASE verzióktól eltérően) végre megvalósított tranzakció kezelés (BEGIN/END TRANSACTION, ROLLBACK) az ON ERROR paranccsal kombinálva igen hatékony eszköz az esetleg mégis előforduló adatvesztések korrigálására. Bővebb magyarázat helyett álljon itt egy rövid példaprogram vázlat.

```
Set Reprocess To 30
On Error Do Hiba
Begin Transaction
      <tranzakciós utasítások>
End Transaction
If .Not. RollBack()
      ? "Az eredeti állapot visszaállítása nem sikerült."
      ? "Kérem, töltse vissza a kimentett fájlokat."
      Return
EndIf
.
stb...
Return
Procedure Hiba
v = " "
      Do Case
            Case Error() = 108 && Valaki zárolta a fájlt.
                 ? "Az egyik fájl foglalt. Próbáljam újra ?"
                 Wait To v
                 If Upper(v) = "I"
                       Retry
                       Return
                 Else
                       If Completed()
                             ? "A művelet sikeresen lezárult."
                             Return To Master
                       EndIf
                       ? "Az eredeti állapot visszaállítása ..."
                       RollBack
                 EndIf
```

```
                    Return To Master
                    .
                    .
                    . stb.
            EndCase
    Return
```

7.8. A hálózati üzemmód hibaüzenetei

Az alább felsorolt hibaüzenetek csak hálózati környezetben és közvetlenül a dBASE IV indítása után fordulhatnak elő. Ezek a függelékben nem szerepelnek, ugyanis célszerűbbnek tartottuk itt ismertetni őket. Az üzenetek ABC sorrendben követik egymást. Mindegyik mellett közöljük a magyar fordítást és ahol lehetséges, a hiba elhárításának lehetőségeit is.

Check network drive specification
Ellenőrizze a hálózati lemezegység megadását
A dBASE IV indításakor a dBASE IV és a DBNETCTL.300 fájlok nincsenek a hálózatvezérlőn.

Control file cannot be processed
A vezérlő-fájl nem feldolgozható
A dBASE IV nem találja a DBASE.CTL fájlt a DBNETCTL.300 alkönyvtárban vagy a fájl megsérült.

Control file LOCK failure
Vezérlő-fájl zárolási hiba
A DBASE.CTL fájl ellenőrzése közben belső hiba történt.

Load failed
A betöltés nem sikerült
A DBNETCTL.300 alkönyvtárban lévő fájlok olvashatatlanok.

Maximum network users reached (xx) Please try later
Elértük a maximális számú (xx) felhasználót. Próbálkozzon később
A zárójelben jelzett számú felhasználó az utolsó, aki még bejelentkezhetett a dBASE IV-be. Vámi kell, amíg valaki kilép.

Memory allocation error
Memória foglalási hiba
A dBASE IV indulásakor belső hiba történt.

Network drive not supplied
A rendszer nincs ellátva hálózati lemezegységgel
Valószínűleg rosszul adta meg a hálózati egység paramétereit vagy a DBNETCTL.300 alkönyvtár fájljai nincsenek a megjelölt helyen.

Network load failure
Hálózat betöltési hiba
A dBASE IV vagy a DBNETCTL.300 megsérült. Próbálja újra indítani a dBASE IV-et. Amennyiben ismét előfordul ez a hiba, újra kell installálni a rendszert.

Unauthorized duplicate
Illetéktelen másolat
A DBASE.LOD nincs a DBNETCTL.300 könyvtárban.

8. A dBASE IV és a dBASE III PLUS

A dBASE IV a Magyarországon is közkedvelt dBASE programcsalád legújabb változata. Alapvetően elődei filozófiájára épül; de fejlesztői elfogadták a kor kihívását, és "felszerelték" mindazzal a tudással, amellyel kategóriájában az élmezőny jogos tagja lehet, ha éppen nem az első. (Szakmai berkekben rebesgetik, hogy talán éppen az alapfilozófiához való ragaszkodás az akadálya annak, hogy "utcahossznyi" előnyre tegyen szert.) Mindenesetre elődjének sok korlátját ledöntötte, lehetőségei, szolgáltatásai bővültek, sőt olyan elemekkel is kiegészült, amelyeknek a dBASE III PLUS-ban nyoma sem volt (SQL).

Ebben a fejezetben rövid áttekintést adunk a különbségekről mellőzve a részletes magyarázatokat.

8.1. Mezők, változók

8.1.1. Korlátok feloldása

A korábbi legfeljebb 256 aktív változó helyett 15000 használható egyidőben.

Mezők lehetséges száma egy adatfájlon belül 128-ról 255-re nőtt.

Korábban nehézségekbe ütközött, ha a mezőkkel azonos nevű változókat alkalmaztunk, mert ha a mező aktív volt (éppen az az adatbázis volt aktív, amelyiknek a változóval azonos nevű mezője volt), akkor csak a mezővel végezhettünk műveletet. A dBASE IV-ben a változó elé illesztett >előtagM->előtag feloldja ezt a korlátot. Ezzel a megkülönböztetéssel már a változót is használhatjuk a műveletekhez.

8.1.2. Új adattípusok

Bővült a **numerikus adattípus** a BCD numerikus típussal, amelynek nem volt korábban megfelelője, annak ellenére, hogy ezt a dBASE IV N kóddal jelöli és Numeric-nek nevezi. A dBASE III PLUS numerikus adattípusának a dBASE IV-ben a F (FLOAT), azaz a lebegőpontos típusa felel meg. A két numerikus típus között minkét irányba létezik átalakító függvény. (FIXED(), FLOAT())

A dBASE IV lehetőséget ad a dátum típusú mezők mellett a dátum típusu változók alkalmazására is. Így nincs szükség a dátumműveletekhez a karakteres változók konvertálására. (CTOD())

Példa:

```
SET DATE ANSI
Date = {89.05.16}
```

8.1.3. Tömbök

Új és figyelemreméltó lehetőség az egy- és kétdimenziós tömbök használata.

Tömbkezelő utasítások:

APPEND FROM ARRAY	PUBLIC
AVERAGE	SUM
COPY TO ARRAY	STORE
DECLARE	@ és ? / ??

8.2. Adatleírás, adatkezelés

8.2.1. Korlátok

Mezők lehetséges száma 128-ról 255-re nőtt.

A megnyitható fájlok száma 99-re bővült.

Az egyszerre megnyitható és karbantartott indexek száma 470-re nőtt. (10 összetett indexfájl fájlonként 47 db index)

minden egyes összetett fájl egy fájl megnyitásnak számít.

Egy adatfájlhoz több másik adatfájl is kapcsolható a SET RELATION utasítással. (többszörös reláció) A kapcsolások egymásbaágyazása is megengedett, azaz a kapcsolt adatfájlhoz is lehet újabbat kapcsolni, stb.

8.2.2. Összetett indexfájl

Kényelmes kezelést jelent az összetett indexfájl használata, amely fájlonként (legfeljebb) 47 indexet integrál.

8.2.3. Új mezőjellemző (Index-e?)

A fájlszerkezet meghatározásánál megadhatjuk minden mezőnél azt is, hogy kívánjuk e kulcsmezőként, indexként használni.
Ezt az "INDEX" rovatban a Y (Yes = Igen) illetve N (No = Nem) értékek valamelyikének kiválasztásával tehetjük. Ilyenkor a dBASE IV az adatfájl nevével, de .mdx kiterjesztéssel automatikusan létrehozza az összetett indexfájlt.

8.2.4. Adattárolás és -hozzáférés

Megakadályozhatjuk az adatfájlokban a **nem kívánt módosításokat** a READ ONLY fájl-jellemző beállításával.

Lehtőséget kapunk az esetleges üzemzavarok okozta nagyobb mérvű **adatvesztések kiküszöbölésére** a SET AUTOSAVE ON utasítás alkalmazásával. Ilyenkor egy rekord szerkesztésének befejezése után az adatok azonnal elmentésre kerülnek és az operációs rendszer fájl-adminisztrációjának (FAT) aktualizálása is megtörténik.

Az adatvédelem területén új és hasznos lehetőséget biztosítanak BEGIN TRANSACTION, END TRANSACTION és a ROLLBACK utasítások, amelyek egy (akár ciklusban szervezett) tranzakciósorozat (adatfájlban történő módosítás-sorozat) közben bekövetkezett hibák esetén módot adnak az adatfájlok **eredeti állapotának visszaállítására.**

8.2.5. A memofájlok optimalizálása

A dBASE III PLUS esetében a memofájlokban feleslegessé vált blokkok haszontalanul foglalják a lemezterületet. A dBASE III PLUS új memo-adat esetén mindig a fájl végéhez kapcsolja az új információt. Emiatt növekednek a dBASE III PLUS memofájlok szünet nélkül akkor is, ha közben más adat törlésre került. A dBASE IV a memofájlban törölt adat helyét, a kiürült blokkot újra felhasználja, amikor új adat kerül a fájlba.

8.2.6. Kapcsolat más adatkezelőkkel

A LOTUS, Multiplan, VisiCalc és a PFS rendszerek adatain túl hozzáférhetünk (írhatunk, olvashatunk) a dBASE II, FRAMEWORK II és a Rapidfile adatkezelő adataihoz is.

A kapcsolat létrehozását a következő utasítások támogatják:

APPEND FROM	COPY TO
EXPORT TO	IMPORT FROM

8.3. Felhasználói csatoló

A dBASE IV számos új lehetőséget azok számára, akik különösebb programozói tudás nélkül kívánnak egyszerűbb adatkezelési feladatokat elvégezni. Bővültek és emberközelibbé váltak az ilyen célú, már a dBASE II PLUS-ban is megtalálható eszközök. Természetesen ezek jól használhatók a programozó számára is.

8.3.1. Control Center.

A Control Center a dBASE III PLUS "assist" nevű felhasználói csatolójának továbbfejlesztése. (Itt is az assist paranccsal hívható.) A minőség javulása és a szolgáltatások bővülése azonban olymérvű, hogy nyugodtan nevezhetjük újnak is. Alkalmas dBASE IV eleme "leírás nélküli" önálló kezelésére. Használatával létrehozhatunk, megnyithatunk, megjeleníthetünk, módosíthatunk és lezárhatunk (nemcsak adat-) fájlokat. (adatfájl, formátumfájl, jelentésfájl, stb). Beléphetünk a programszerkesztőbe, sőt az alkalmazásgenerátorba is. Programok, generált alkalmazások futtatását is indíthatjuk innét. A "Vezérlő Központ" elnevezés tehát nem is túlzás.

8.3.2. Nézetfájl

Lényegesen könnyebb bizonyos feltételt teljesitő rekordok megjelenítése. A nézetfájl létrehozásához könnyen megadható a mezők listája és a szűrőfeltételek. Ezeket a paramétereket a dBASE IV külön fájlban megőrzi, így a későbbi megjelenítésnél csak ezt a fájl kell aktivizálnunk.

8.3.3. Jelentés készítése

Adatainkból könnyebben készíthetünk (nyomtatón vagy képernyőn is) megjeleníthető jelentést. A jelentés paramétereinek beviteléhez nagyon szemléletes, párbeszédes csatoló készült. Komfortosságára jellemző, hogy a jelentés egyes elemeihez még különböző betűtípusokat (normál, kövér, stb) is meghatározhatunk. A jelentésformátum megőrzi paramétereinket.

8.3.4. Váltás a fájlszerkesztési módok között

Az adatfájlszerkesztés két formája, a BROWSE és EDIT utasítások között az **F2** billentyű megnyomásával egyszerűen válthatunk.

8.3.5. Átfogó segítő-rendszer (Full-help)

A felhasználói csatolók hátterében lévő segítő-rendszer figyeli az aktuális (éppen végrehajtás alatt lévő) funkciókat. Az **F1** billentyű megnyomására az éppen aktuális segítő-információk (utasítás szintaktika, alkalmazási példa, stb) egy külön ablakban megjelennek a képernyőn, különböző szempontok szerint az összes segítő-információ is elérhető, sőt kívánságra ki is nyomtatható.

8.4. Programozástechnológiai eszközök

Jobbak, komfortosabbak lettek a dBASE III PLUS-ban is már (esetenként csak csíráiban) meglévő eszközök, amelyek a hatékony, gyors programozást támogatták, de jelentősen bővült is ez az eszköztár. Különösen nagy súlyt helyeztek a felhasználóbarát programok készítőinek kényelmét szolgáló lehetőségekre

8.4.1. Korlátok feloldása

Egy program és "interpreter" **utasítássor** lehetséges hossza 1024 karakterre nőtt, sőt ez még további 254 karakterrel is bővíthető a interpreter kiterjesztett (teljes képernyős) használatakor.

Nemcsak az aktuális adatfájlon operálhatunk az alábbi utasításokkal és függvényekkel, ha az adatfájl hivatkozási nevét is megadjuk:

utasítások

DISPLAYSTRUCTURE	SKIP
GO / GOTO	UNLOCK
LISTSTRUCTURE	USE
RESET	

függvények

BOF()	LOOKUP()	CHANGE()
LUPDATE()	DBF()	MDX()
DELETED()	NDX()	EOF()
RECCOUNT()	FIELD()	RECNO()
FLOCK()	RECSIZE()	FOUND()
RLOCK()	ISMARKED()	SEEK()
KEY()	TAG()	LOCK()

Adatfájl megnyitásánál karakteres kifejezést is használhatunk. A karakteres kifejezés kezelése lényegesen gyorsabb, mint a makró-hivatkozásé.

8.4.2. Alkalmazás-generátor

Az alkalmazás-generátor menürendszere az utasítások konkrét ismerete nélkül használható programok vázának, sőt a legfőbb adatkezelő eljárásoknak a létrehozására. Az alkalmazás megtervezése után hibátlan forrásprogramot készít, amely a "manuálisan" előállított programokhoz hasonlóan szerkeszthető, módosítható.

Az alkalmazás-generátorral készült programrendszer struktúrájáról dokumentációt is készíthetünk a DOCUMENT.GEN felhasználásával.

8.4.3. Programtesztelés

Nagyon hatékony eszköz a beépített hibakereső rendszer (debugger), amely menetközben még a forrásprogram szerkesztését is lehetővé teszi. A debugger négy, élesen elválasztott ablakot használ a jellemzők kiírására, a program javítására, a megjelenítésre és a megszakítási pontok kijelölésére.

Könnyen áttekinthető információt kapunk a program futásának állapotáról a jellemzők-ablakban. Láthatjuk az aktuális eljárást forráslistáját kiemelve az éppen végrehajtásra kerülő sort. Információt ad az adatbázisokról, és a rekordmutató pillanatnyi helyzetéről.

A megjelenítő-ablakban információt kaphatunk változóink pillanatnyi értékéről, és a környezeti paraméterek állásáról. A megszakítási-ablakban 10 feltételt szabhatunk meg, amelyek bármelyikének bekövetkezésekor felfüggeszti a program futását.

8.4.4. Ablakkezelés

Látványossá és áttekinthetővé teszi az adatok megjelenítését az ablakkezelés. Ez nem más mint kis képernyőrészek létrehozása, amelyek az igazi képernyővel megegyező módon működnek. Például kimenő adatfolyam (? utasítás) esetén az adatok csak a kijelölt részen jelennek meg és "gördülnek". Egyszerre 20 ablakot határozhatunk meg, amelyek egymással lefedhetők, mozgathatók és tetszés szerint színezhetők. Az ablakkezelés utasításai:

ACTIVATE SCREEN
ACTIVATE WINDOW
CLEAR WINDOW
DEACTIVATE WINDOW
DEFINE WINDOW
MOVE WINDOW
RELEASE WINDOW
RESTORE WINDOW
SAVE WINDOW
SET WINDOW OF MEMO TO

8.4.5. Menükezelés

Komoly támogatást élvez a menürendszer létrehozására, működtetésére irányuló programozói munka. A dBASE IV kétféle menütípust kínál: az ún. egyszerű vagy **vonalmenüt** (BAR MENU), amelynek elhelyezkedése általában vízszintes, és a függőleges vagy **rolómenüt** (POPUP). A vonalmenühöz rolómenü kapcsolható, így a kurzorvezérlő billentyűk segítségével a menürendszerben "minden irányban" haladhatunk. A menüutasítások és függvények a következők :

utasítások

DEFINE POPUP
DEFINE BAR
ACTIVATE POPUP
SHOW POPUP
DEACTIVATE POPUP
CLEAR POPUPS
RELEASE POPUPS
DEFINE MENU
DEFINE PAD
ACTIVATE MENU
SHOW MENU
DEACTIVATE MENU
CLEAR MENUS
RELEASE MENU

függvények

POPUP()
BAR()
PROMPT()
MENU()
PAD()

8.4.6. Adatbevitel és -megjelenítés paraméterei

A @ sor, oszlop SAY ... GET adatbeviteli, adatmegjelenítő parancs nagyon hasznos cikkellyel, opcióval bővült.

Ki kell emelnünk az érvényesség ellenőrzéséhez használható VALID cikkelyt, amely feltételének helyén egy logikai értéket visszadó felhasználói függvény (eljárás) is állhat. Addig nem tudunk kilépni az (adott mező, változó) beviteléből, amíg olyan adatot nem viszünk be, amely a VALID mögött megfogalmazott feltétel(eke)t nem elégíti ki. Ha a VALID feltétele nem teljesül, újra kell a bevitelt ismételni, csak akkor enged hozzáférni a következő GET változó szerkesztéséhez, ha a feltétel igaz.

Nem kevésbé fontos a színállítási és ablak kijelölési lehetőségek sem.

8.4.7. Felhasználói függvények.

Új lehetőség, hogy a felhasználó (programozó) is konstruálhat saját függvényeket az alábbi formában:

```
FUNCTION
        utasítások
RETURN  kifejezés
```

A függvény egy értékkel tér vissza, amelyet a RETURN után álló kifejezés határoz meg. Látszólag nem sok eltérés van az eljárás (PROCEDURE) és a felhasználó által meghatározható függvények között. Ha azonban arra gondolunk, hogy egy függvényt bármilyen matematikai, sztring kifejezés egy eleme lehet, a különbség mindjárt nyilvánvaló. Példának elegendő a GET új VALID cikkelyét említeni, ahol eljárást biztosan nem hívhatunk, de egy logikai értéket adó függvényt igen.

8.4.8. Billentyű-makró

Hatékonyan használható a programozásnál az elmentett és újra lejátszható billentyű leütés-sorozatok, amelyek mintegy szimulálják a valóságos billentyűleütéseket. Tipikus alkalmazásuk olyan esetekben fordul elő, amikor olyan funkciókat akarunk programból működtetni, amelyek egyébként csak sok leütéssel aktivizálhatók.

8.4.9. Eseménycsapdák

A dBASE III PLUS által is jól ismert eseményfigyelő utasítások jelentősen bővültek. Az új "csapdák" többsége a menüfigyeléssel kapcsolatos, ilyenek az ON PAD, ON SELECTION PAD, ON SELECTION POPUP utasítások

Újdonságnak számít a nyomtatást és az adatbevitel érvényességét figyelő ON READERROR, ON PAGE utasítás.

8.4.10. Memo-mező szerkesztés

A memo-mezők adattartalmának bevitelét és szerkesztését több új utasítás és függvény is támogatja:
APPEND MEMO, REPLACE, COPY MEMO, SET BLOCKSIZE TO, SET WINDOW OF MEMO TO, AT(), MLINE(), LEN(), LEFT(), SUBSTR(), RIGHT(),MEMLINES()

8.4.11. Rekordtartalomtól függő ciklusszervezés

A SCAN-ENDSCAN a DO WHILE-ENDDO analógiájára működik, de a megadott feltételeket kielégítő rekordokat automatikusan sorbavéve hajtja végre a ciklusmag utasítássorozatát.

8.4.12. Keresés nem aktuális adatfájlban

A LOOKUP() függvény lehetőséget ad arra, hogy (akár a nem aktuális adatfájlban) megadott érték szerint megkeressünk egy rekordot a SEEK mintájára, és a megtalált rekord egy tetszés szerinti mezőjének értékét megkaphassuk. Külön előnye az is, hogy a keresést szevenciálisan is elvégzi, tehát a SEEK-kel ellentétben nem szükséges az adatfájl indexelése sem az adott mezőre. Természetesen indexelt adatfájlokon a keresés gyorsabb.

8.4.13. Egyszerűsített statisztikai számítások

A kényelmet és a gyorsabb futást szolgálja a CALCULATE utasítás, amelynek segítségével összevonható néhány matematikai, statisztikai és pénzügyi művelet.

8.5. Utasítás- és függvénytár bővítése

Új függvényekkel egészült ki a dBASE IV, amelyek részben matematikai jellegűek, részben a pénzügyi életben a megtakarítások és befizetések kamatos-kamat számítására használhatók. Ez utóbbiak: FV(), PAYMENT(), PV() .

A matematikai függvények:

SIN()	RTOD()	ASIN()
DTOR()	COS()	SIGN()
ACOS()	CEILING()	TAN()
FLOOR()	ATAN()	LOG10()
ATN2()	RAND()	PI()

8.6. SQL

Az SQL (Structured Query Language) teljesen új utasítás és függvénykészlet, amely a dBASE adatfájlokat kezeli, dBASE IV környezetében. Ez a szabványosítási törekvések egyik terméke, amelynek beépítése egyfajta hídat jelenthet más SQL szabványú adatbáziskezelőkhöz.

Végeredményben választhatunk, hogy az adatkezelési funkciókat a dBASE IV eredeti utasításkészletével, vagy az SQL eszköztárral valósítjuk meg. Ez utóbbi új szemléletével talán szokatlannak tűnik, de meglehetősen hatékony és integrált lehetőségeket nyújt.

8.7. Segédeszközök

8.7.1. Fordítás, összefűzés

A dBASE IV-ben már nem közvetlenül a forrásprogramunkat értelmezi és hajtja végre az interpreter, hanem előbb egy ún. közbülső kódra (.dbo kiterjesztéssel) fordítás történik. Az interpreter felismeri, ha a programunkban változtatásokat eszközöltünk, és végrehajtás, tesztelés előtt automatikusan elindítja a fordítást, amit egyébként a COMPILE paranccsal indíthatunk. A közbülső kódra fordítás a futási sebesség növelése érdekében történt.

A dBASE IV programcsomag eleme a dBLINK külső utasítás. Ez az önálló program képes összekapcsolni 256 összefüggésben álló tárgyprogramot (.dbo) egy tárgy-

programba (.dbo) a gyorsabb futás érdekében. A dBLINK automatikusan összefűz minden kapcsolatban álló fájlt.

8.7.2. dBSETUP segédprogram.

A dBSETUP a dBASE IV belső környezetének beállítására szolgál. Alkalmazható installálásra (dBASE IV üzembehelyezésére), hardver környezet módosítására, és a **config.db** fájl létrehozására, módosítására. Mindezenfelül információt szolgáltat a dBASE IV és a DOS környezetről.

8.7.3. Rendszerváltozók.

A dBASE IV-ben a rendszerváltozók segítségével beállíthatjuk a megfelelő nyomtatási, megjelenítési módot, kijelölhetjük a nyomtatás helyét, beállíthatjuk a példányszámot, stb.

8.7.4. Nyomtatási lehetőségek

Esztétikusabb, áttekinthetőbb nyomtatott listákat készíthetünk. A nyomtató meghajtó (PRINTER DRIVER) lehetőséget nyújt kövér, kurzív, aláhúzott betűk használatára. Lehetőség van a lapszám, sorköz stb beállítására is.

A ?/?? utasítás meghívásakor a STYLE paraméter gondoskodik a megfelelő nyomtatási mód megválasztásáról. Az új ??? utasítás segítségével vezérlő karaktereket küldhetünk a nyomtató felé.

8.8. Kompatibilitás, konverzió

A dBASE IV felülről kompatibilis a dBASE III PLUS-szal. Ez azt jelenti, hogy sem az adatainkat, sem a programjainkat nem kell "kidobni", mindegyik használható dBASE IV alatt is. A dBASE III PLUS-ból átvett számok automatikusan N (BCD) típusu számokká alakulnak át.

Általában a dBASE IV-ben készült adatok és programok is használhatóak dBASE III-ban, ha nem tartalmaznak olyan adattípust (pl: FLOAT), utasítást, amely a dBASE III-ban nincs. A dBASE IV képernyő-formátum fájljai is használhatók, ha azok nem tartalmaznak a @ utasításon kívül más utasítást és ha @ utasítás nem tartalmaz új dBASE IV kulcsszavakat.

ÁLLOMÁNYKAPCSOLATOK ÉS KOMPATIBILITÁS

Típus	dBASE IV kiterjesztés	Leford. fájl	Alulról kompatibilitás	Átnevezett dBASE III PLUS fájl
katalógus	.cat		igen	
adatfájl	.dbf		igen	
memo	.dbt		igen	
képernyőform. meghatározások	.scr		nincs	.sc3
képernyő-formátum	.fmt	.fmo	nincs	
listaformátum meghatározások	.frm		nincs	.fr3
listaformátum	.frg	.fro	nincs	
címkeformátum meghatározások	.lbl		nincs	.lb3
címkeformátum	.lbg	.lbo	nincs	
nézet	.qbe	.qbo	nincs	
módosító nézet	.upd	.dbo	nincs	
SQL program	.prs	.dbo	nincs	
minta(PICT)	.gen		nincs	
makro	.key		nincs	
program	.prg	.dbo	nincs	

8.1.táblázat

8.9. Bővítések listája

Az alábbiakban rövid, részletezés nélküli felsorolást adunk a dBASE III PLUS-ban nem létező, vagy másként működő (kevesebb lehetőséggel rendelkező) utasításokról és függvényekről. Kövér betűkkel szedtük az új, csak a dBASE IV-ben lévő utasításokat és függvényeket.

8.9.1. Utasítások.

? / ??	utasítások támogatják az AT, FUNCTION és STYLE cikkelyek használatát, amelyek a megjelenítés formáját és a nyomtatási módot határozzák meg.
???	Vezérlő karaktereket lehet küldeni a nyomtatónak.
@ ...SAY ...GET [[OPEN]WINDOW < ablaknév >] [VALID < feltétel >] [ERROR < C-kifejezés >] [WHEN < feltétel >] [DEFAULT < kifejezés >] [MESSAGE < C-kifejezés >] [COLOR < normál >, < kiemelt >]	Az új paraméterek gondoskodnak sorrendben: az ablakban történő megjelenítésről, az adatok érvényességének ellenőrizhetőségéről, feltételtől függő bevitelről, megadható hibaüzenetről, a kezdőérték beállításáról, tájékoztatóüzenet kijelzéséről, színezésről.
@.. FILL TO	Képernyőrész kiszínezésére szolgál.
@.. TO .. [DOUBLE/PANEL /< keretsztring >] [COLOR < színkifejezés >]	Az új paraméterek használata lehetővé teszi a felhasználó igényeinek megfelelő keret megrajzolását a kívánt színben.
ACTIVATE MENU	Aktivál (működésbe helyez) egy vonalmenüt.
ACTIVATE POPUP	Aktivál egy rolómenüt.
ACTIVATE SCREEN	Az ablakokban történő megjelenítésről visszavált a teljes képernyőre.
ACTIVATE WINDOW	Aktivál egy ablakot.
APPEND FROM ARRAY < tömbnév > /[DBASEII/RPD/FW2]	Az új opciók segítségével az adatbázis végéhez tömböt, vagy a feltüntetett "új fájltípusok" valamelyikét fűzi hozzá.
APPEND MEMO	Kijelölt memomező bővítése.
AVERAGE TO ARRAY < tömbnév >	Az átlagszámítás eredménye tömbben is tárolható.

BEGIN TRANSACTION	Az utasítás létrehoz a kijelölt adatfájlról egy másolatot, amely a ROLLBACK utasítás használatával visszaállítható.
BROWSE [NOINIT] [NOEDIT] [NODELETE] [NOCLEAR] [COMPRESS] [FORMAT] [WINDOW < ablaknév >]	Lehetőség nyílik többek között arra, hogy megtiltsuk a rekordok szerkesztését és törlését, megváltoztassuk a megjelenítés módját, stb.
CALCULATE	Matematikai számításokat integrálhatunk.
CHANGE	Lásd BROWSE utasítás új paramétereit.
CLEAR MENUS	Törli az összes menüt a képernyőről és menüdefiníciót a memóriából is.
CLEAR POPUPS	Törli az összes rolómenüt a képernyőről és rolómenü-definíciót a memóriából.
CLEAR WINDOWS	Törli az összes ablakot a képernyőről és ablakdefiníciót a memóriából.
COMPILE	Lefordítja a forrásprogramot tárgykódúra (.dbo).
CONVERT	Egy mezőt fűz az aktív adatfájl végére, amely többfelhasználós üzemmódban fájl-információkat tartalmaz.
COPY INDEXES	A felsorolt indexfájlokat összetett (multi) indexfájl-lá konvertálja.
COPY MEMO	Memo-mezőt másol fájlba.
COPY TAG	Az összetett (.mdx) indexfájlban lévő tagot másolja ki egyszerű indexfájlba (.ndx).
COPY TO ARRAY < tömbnév > [FIELDS < mezőlista >]	Adatfájl segítségével feltölt egy tömböt.
COPY TO [DBASEII/DBMEMO3/RPD/FW2]	Másolatot készít az aktív adatfájlról a megjelölt adatkezelő rendszerek fájlformátumában is.
CREATE APPLICATION	Az utasítás indítja az Alkalmazás-generátort.
DEACTIVATE MENU	Érvényteleníti, törli a képernyőről a vonalmenüt.
DEACTIVATE POPUP	Érvényteleníti, törli képernyőről a rolómenüt.
DEACTIVATE WINDOW	Törli a képernyőről, és érvényteleníti a kiválasztott ablako(ka)t.

DEBUG	Programkövetést indítja.
DECLARE	Tömbdeklaráció, egy- vagy kétdimenziós tömb kijelölése, deklarálása.
DEFINE BAR	Rolómenü definíció (rolómenü elemeinek meghatározása).
DEFINE BOX FROM	Nyomtatáskor a szöveg köré rajzolandó keret meghatározása.
DEFINE MENU	Vonalmenü kijelölése.
DEFINE PAD	A vonalmenü elemeinek meghatározása.
DEFINE POPUP	Rolómenü kijelölése
DEFINE WINDOW	Ablak kijelölése.
DELETE TAG	A megjelölt indextagot törli az összetett indexfájlból.
DISPLAY TO FILE < fájlnév >	Az új cikkely használatával a kimenő adatok szövegfájlba kerülnek.
EDIT	Lásd a BROWSE utasítás opcióit.
EJECT PAGE	A következő lap elejére áll, vagy pedig az ON PAGE utáni utasításban meghatározott módon halad előre a nyomtatófej.
ENDPRINTJOB	A nyomtatási köteg végét jelző utasítás, amely után érvényét veszti a beállított nyomtatási mód.
ENDSCAN	A SCAN ciklusutasítás vége.
END TRANSACTION	Lezárja a tranzakciósorozatot, az adatfájl helyreállítása ezután nem lehetséges.
EXPORT TO FW2/RPD/DBASEII [FIELDS < mezőlista >] [< hatáskör >][FOR < feltétel >] [WHILE < feltétel >]	Az aktuális adatfájl tartalmát, a kijelölt adatkezelő rendszerek fájljaiba is bemásolja.
FUNCTION	Felhasználó által konstruált függvény utasításainak kezdete.
GO / GOTO [IN < másodnév >]	Nemcsak az aktuális, hanem a másodnévvel kijelölt adatfájlban is történhet a rekordmutató pozicionálása.
IMPORT FROM [RPD/FW2/WK1/DBASEII]	A megadott adatkezelő rendszerek fájljaiból is létrehoz dBASE IV adatfájlt.

INDEX ON TAG < tagnév > [OF < mdx-fájlnév >] [DESCENDING]	Az indexet az összetett indexfájl (mdx-fájl) egyik tagindexeként határozza meg. DESCENDING opció hatására fordított ASCII sorrendben történik a rendezés.
LIST TO FILE < fájlnév >	A kimenő adat szövegfájlba kerül.
MODIFY APPLICATION	Lásd CREATE APPLICATION.
MODIFY COMMAND WINDOW < ablaknév >	Az általunk definiált ablakban történhet meg a programszerkesztés.
MOVE WINDOW	Az ablakot új helyre mozgatja a képernyőn.
ON KEY [LABEL < billentyűfelirat >] [< utasítás >]	LABEL cikkely esetén csak a kiválasztott billentyű leütése után hajtja végre a megfelelő utasítást.
ON PAD	Összekapcsolja a vonalmenüpontot egy rolómenüvel.
ON PAGE	Adatfolyam megjelenítésnél a sor koordinátát figyeli.
ON READERROR	Érvénytelen adat bevitelét figyeli (rossz dátum, RANGE, VALID, stb).
ON SELECTION PAD	Egy vonalmenüpont kiválasztását figyeli.
ON SELECTION POPUP	Rolómenüpont kiválasztását figyeli.
PLAY MACRO	A makró-könyvtárban rögzített billentyűmakrókból hajt végre egyet.
PRINTJOB	A nyomtatási köteg elejét jelzi, amelyre azonos nyomtatási jellemzők érvényesek.
PROTECT	Hozzáférési jog kijelölése.
PUBLIC .. ARRAY < tömb elem lista >	Tömbök hatáskörének kiterjesztése.
RELEASE MENUS	Menü törlése a memóriából és a képernyőről.
RELEASE POPUPS	Rolómenü törlése a memóriából és a képernyőről.
RELEASE WINDOWS	Ablakok törlése a memóriából és a képernyőről.
REPLACE .. WITH .. [ADDITIVE]	Adatok hozzáfűzése egy memo-mező tartalmához, rekord mező módosítás.
RESET	Eltávolítja az adatfájlból a fájl szerkesztés előtti állapotát.
RESTORE MACROS	Betölti a makró-fájlt a memóriába.
RESTORE WINDOW	Ablakbetöltés memóriába fájlból.

RETURN < kifejezés >	Felhasználói függvény visszatérő utasítása. A kifejezés a függvény visszatérési értéke.
ROLLBACK	Helyreállítja adatfájl a tranzakciós feldolgozás megkezdése előtti (BEGIN TRANSACTION) állapotát.
SAVE MACROS	Elmenti a memória makró-könyvtárát egy fájlba.
SAVE WINDOW	Ablakot menti fájlba.
SCAN	Adatfájl tartalmától függő ciklusszervező utasítás. A ciklus kezdete.
SHOW MENU	Aktíválás nélkül megmutatja a vonalmenüt.
SHOW POPUP	Aktíválás nélkül megmutatja a rolómenüt.
SKIP[IN < másodnév >]	A másodnév megjelölése esetén nemcsak az aktuális, hanem attól különböző adatfájlban is történhet léptetés.
SORT TO [ASCENDING/DESCENDING]	Lehetőséget ad emelkedő illetve csökkenő sorrendbe rendezéshez.
SUMTOARRAY < tömbnév >	Az összegképzés eredménye tömbbe kerül.
TYPE[NUMBER]	Az új opció hatására a szövegfájl tartalmának megjelenítésekor látható a sorszám is.
UNLOCK IN < másodnév >	Felszabadítja a zárolt a rekordot vagy fájlt az aktuális adatfájlon kívül is.
USE IN < másodnév > INDEX < ndx- vagy mdx-fájl > ORDER < ndx-fájl vagy indextag > [EXCLUSIVE][NOUPDATE]	Az aktuális index az összetett indexfájl indextagjaként vagy egyszerű indexfájlként egyaránt meghatározható. Az EXCLUSIVE opció többfelhasználós üzemmódban kizárólagos használatra nyitja meg a fájlt, így azt más felhasználó nem érheti el. A NOUPDATE opció pedig csak olvasást tesz lehetővé.

8.9.2. Függvények.

ACOS()	Arkusz koszinusz függvény.
ALIAS()	Visszaadja a megadott munkaterület hivatkozási nevét, azaz másodnevét.
ASIN()	Arkusz szinusz függvény.
AT(<C-kifejezés1>, <C-kifejezés2> /<memo-mező>)	Az első sztring keresését a memo-mezőben is elvégzi.
ATAN()	Arkusz tangens függvény.
ATN2()	Arkusz tangens függvény. A függvény paraméterei: a szög szinusza és koszinusza.
BAR()	Visszaadja a kiválasztott rolómenüpont sorszámát.
CALL()	dBASE IV-en kívüli, gépi kódú program végrehajtására szolgál.
CEILING()	Legkisebb egész szám meghatározása, amely nagyobb vagy egyenlő mint a megadott érték.
CHANGE()	Logikai függvény, amely többfelhasználós üzemmódban megmutatja, hogy az adott rekord módosításra került-e a megnyitás óta.
COMPLETED()	Megmutatja, hogy a BEGIN TRANSACTION utasítással megkezdődött tranzakció befejeződött-e.
COS()	Radiánban megadott szögnek kiszámítja a koszinuszát.
DIFFERENCE()	Meghatározza két C-kifejezés SOUNDEX() függvény által képzett értékeinek különbségét.
DMY()	A dátum kifejezést adja vissza karakterláncként.
DTOR()	Fokot radiánná alakítja át.
DTOS()	A dátum kifejezést karakterlánccá alakítja át.
FIXED()	Lebegőpontos számot fixpontossá konvertálja.
FLOAT()	Fixpontos számot lebegőpontossá alakítja át.
FLOOR()	Visszaadja azt a legnagyobb számot, amely kisebb vagy egyenlő, mint a megadott.
FV()	Pénzügyi függvény. Kiszámítja a megtakarítás jövőbeni értékét a kamatokat figyelembevéve.
ISMARKED()	Többfelhasználós üzemmódban jelzi, hogy az adatfájl éppen módosítás alatt áll-e.
KEY()	A megadott indexfájl kulcskifejezését adja vissza.

LASTKEY()	Az utolsóként lenyomott billentyű ASCII kódját adja vissza.
LEFT(<memo-mezőnév>, <N-kifejezés>)	A memo-mezőből adott számu karaktert ad vissza balról.
LEN(<memo-mezőnév>)	A memo-mező tartalmának hosszát adja vissza.
LIKE()	Megmutatja, hogy egy karaktersorozat egy másikra illeszthető-e.
LINENO()	Az éppen értelmezésre került utasítássor sorszámát adja vissza.
LKSYS()	Információt ad a zárolt rekordról, fájlról.
LOG10()	10-es alapú logaritmus.
LOOKUP()	Hatékony rekordkeresés úgy indexelt, mint szekvenciális fájlban egyaránt.
MDX()	Az adatfájlhoz tartozó összetett indexfájl nevét adja vissza.
MDY()	A dátum kifejezést hónap, nap, év karakterlánccá alakítja át.
MEMLINES()	Megmutatja, hogy az megadott memo-mezőnév hány sorban fér el.
MEMORY()	A szabad RAM memória méretet adja vissza.
MENU()	Visszaadja az aktív vonalmenü nevét.
MLINE()	A megadott memo-mezőnév megadott sorát adja vissza.
NDX(<N-kifejezés> [,<másodnév>])	A kijelölt munkaterületen megnyitott egyszerű indexfájl nevét adja vissza.
NETWORK()	Megmutatja, hogy a dBASE IV hálózatban fut-e.
ORDER()	Visszaadja az elsődleges indexfájl nevét.
PAD()	Az aktív vonalmenü kiválasztott menüpontját adja vissza.
PAYMENT()	A kölcsöntörlesztések mértékét számolja ki a kamatokat figyelembevéve.
PI()	A Ludolf féle szám 3.14159 .
POPUP()	Visszaadja az aktív rolómenü nevét.
PRINTSTATUS()	Megmutatja, hogy a printer nyomtatásra kész állapotban van-e.
PROGRAM()	Annak a programnak vagy eljárásnak a nevét adja vissza a függvény, amelyikben hiba történt.
PROMPT()	A függvény visszaadja a kiválasztott menüpontot.

PV()	Pénzügyi funkció, kiszámolja a megtakarítás értékét bizonyos idő után.
RAND()	Véletlen szám függvény.
RIGHT(<memo-mezőnév>, <N-kifejezés>)	A kijelölt memo-mezőből adott számu karaktert ad vissza jobbról.
ROLLBACK()	Kijelzi, hogy az utolsó ROLLBACK utasítás sikeres volt-e.
RTOD()	Radiánt fokká alakít át.
SEEK()	Megmutatja, hogy a megadott értékkel található-e rekord az adatfájlban.
SELECT()	A függvény visszaadja a szabad munkaterületek közül a legnagyobb számát.
SET()	Visszaadja a megadott SET paraméter állapotát.
SIGN()	Előjel függvény.
SIN()	Radiánban megadott szögnek kiszámítja a szinuszát.
SOUNDEX()	Kódot szolgáltat, amely az alapja az angol fonetika szerint hasonló kiejtésű szavak keresésének.
SUBSTR(<memo-mezőnév>, <kezdőpozíció>, <hossz>)	A megadott memo-mező kijelölt részét képezi.
TAG()	Visszaadja az összetett indexfájl megfelelő elemét.
TAN()	Tangens függvény.
USER()	Az aktuálisan bejelentkezett felhasználó bejelentkezési nevét adja vissza.
VARREAD()	Az éppen szerkesztés alatt álló változó vagy mezőnév nevét adja vissza.

8.9.3. A környezeti jellemzők, szolgáltatások

SET ALTERNATE TO [ADDITIVE]	Szövegfájlban felülírás helyett hozzáfűzést eredményez.
SET AUTOSAVE on/OFF	Minden változtatás után a lemezre mentés azonnal megtörténik/ nem történik meg.
SET BELL TO	Hangjelzés beállítása.
SET BLOCKSIZE TO	Memo-mezők és összetett indexfájlok blokkméretének beállítására szolgál.
SET BORDER TO	Az ablakok, keretek és menük rajzolásának formáját szabja meg.
SET CARRY TO	Az új rekord értékeit, az előző rekordbeli értékkel tölti fel.
SET CLOCK on/OFF	Rendszeridő-megjelenítés ki/bekapcsolása.
SET CLOCK TO	Rendszeridő képernyő-koordinátáinak meghatározása.
SET COLOR OF	A képernyőn megjelenő szövegek, táblázatok és alakzatok színeinek elkülönített meghatározása.
SET CURRENCY LEFT/right	Pénznem helyének kijelölése.
SET DATE AMERICAN	mm/dd/yy jellegű rendszerdátum beállítása.
SET DESIGN ON/off	Tervezési, kijelölési funkciók engedélyezése, letiltása.
SET DEVELOPMENT ON/off	Beállítható a forrásprogram automatikus újrafordítását, ha abban változás történt.
SET DEVICE TO FILE < fájlnév >	A @ .. SAY utasítások eredménye a kijelölt fájlba kerül.
SET DISPLAY TO	Monitortípus kiválasztása.
SET EXCLUSIVE on/off	Többfelhasználós környezetben lehetővé teszi a kizárólagos fájlhasználatot.
SET FIELDS TO [< mező > [/R] /< számított mező > ...] /... /ALL [LIKE/EXCEPT < maszk >]	Csak a megadott mezőlistához lehet hozzáférni, illetve R paraméter esetén pedig csak olvasni lehet őket. Az ALL LIKE és az ALL EXCEPT opció a rövidebb kijelölést segíti elő.
SET FULLPATH on/OFF	Fájlnév kiírás elérési úttal, vagy anélkül.
SET HOURS TO	A rendszeridő megjelenési formáját szabja meg.

SET INDEX TO [ORDER[TAG] < ndx-fájl > /< mdx-fájl > [OF < mdx-fájl >]]	Kijelölhető megnyitásra az összetett indexfájl is és az elsődleges indextag.
SET INSTRUCT ON/off	Engedélyezi/letiltja a promptok megjelenését.
SET LOCK ON/off	Többfelhasználós környezetben a rekordok zárolásának módját állítja.
SET MARK TO	Rendszerdátum elválasztó karaktereit állítja be.
SET NEAR on/OFF	Keresés esetén, ha nem talál megfelelő rekordot, nem a fájl végére áll, hanem a kereső kifejezéshez legjobban hasonlító rekordra áll.
SET ORDER TO TAG < indextag > OF < mdx-fájl >	Az elsődleges indexfájl kijelölésére szolgál az összetett indexfájlban.
SET POINT TO	Tizedespontot jelölő karakter megváltoztatására szolgál.
SET PRECISION TO	Matematikai számítások pontosságát írja elő.
SET REFRESH TO	Többfelhasználós rendszerben azt az időtartamot határozhatjuk meg, amely idő alatt a más felhsználóknál történt adatváltoztatások megjelenítésére, adatfrissítésre igényt tarunk.
SET RELATION TO [< N-kifejezés > INTO < másodnév >] [,...] [,....]	Összekapcsolhatunk egy fájlt több fájllal. A kapcsolatok egymásbaágyazhatók.
SET REPROCESS TO	Többfelhasználós környezetben, az adatokhoz való hozzáférés próbálkozásainak a száma állítható be.
SET SEPARATOR TO	A háromhelyiértékenkénti elválasztó karaktert határozhatjuk meg.
SET SKIP TO	Ha egy fájl rekordjához egy másik fájl több rekordja is kapcsolódik (tuljadonos-tag viszony), mindegyik kapcsolt rekordhoz hozzáférhetünk a tulajdonos-adatfájl rekordmutatójának mozdulása nélkül.
SET SPACE ON	?/?? utasításokkal nyomtatott, egymástól vesszővel elválasztott kifejezések között automatikusan egy üres helyet hagy.
SET SQL on/OFF	SQL üzemmód bekapcsolása.

SET TRAP on/OFF	Programhiba vagy az Esc billentyű lenyomására aktivizálja a debug utasítást.
SET WINDOW OF MEMO TO	Kijelöli a memo-mező szerkesztésének aktuális ablakát.

1-2. kötet ára: 488 Ft