

# Magiszter

Számítástechnikai szerkesztőség

## MIQUEL

C nyelvű lekérdező rendszer MISAM adatállományokhoz egymunkahelyes

vagy NOVELL hálózati rendszerekhez

Programvédelem nélkül!

Szoftver  
IBM PC, XT, AT  
kompatibilis mikroszámítógépekre







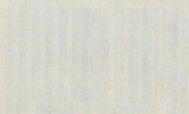


GONDOLA-KISS  
KRAFFT-SCHILL-SZEGEDI

**MIQUEL**

C nyelvű lekérdező rendszer

NTAK





**Szoftver-dokumentáció**

**Sorozatszerkesztő: Pintér Tibor**

**A kiadványt szakmailag ellenőrizte: Benedikti István**

**© Akadémiai Kiadó és Nyomda Vállalat**

**GONDOLA-KISS-KRAFFT-SCHILL-SZEGEDI, 1990**

**Minden jog fenntartva**

**A szoftver-dokumentáció a DOG programcsomaggal készült**

**© Akadémiai Kiadó és Nyomda Vállalat · Schill – Szegedi**

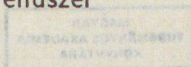


56348

GONDOLA ZOLTÁN-KISS GYÖRGY  
KRAFFT WALTER-SCHILL RÓBERT-SZEGEDI JÁNOS

## MIQUEL

C nyelvű lekérdező rendszer



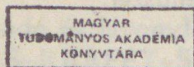
Akadémiai Kiadó, Budapest 1990

MTAK





677242



AKADÉMIAI KIADÓ ÉS NYOMDA VÁLLALAT

Felelős kiadó: Hazai György főigazgató

Felelős szerkesztő: Nagy Zoltán

A borítót Szabadi Zoltán tervezte

ISBN 963 05 5749 5

ISSN 0238-3896

A kézirat lezárva: 1990. május 30.





## T a r t a l o m

|                                            |    |
|--------------------------------------------|----|
| 1. Bevezetés.....                          | 1  |
| 2. A lekérdező rendszer alapelve.....      | 3  |
| 3. A MIQUEL könyv felépítése.....          | 5  |
| 4. Alapfogalmak és definíciók.....         | 7  |
| 4.1. Alias név.....                        | 7  |
| 4.2. Illesztési pontok.....                | 7  |
| 4.3. Intervallum.....                      | 7  |
| 4.4. Lekérdezési mód.....                  | 8  |
| 4.5. Lekérdező rendszer.....               | 8  |
| 4.6. Linkelt állomány.....                 | 9  |
| 4.7. Link kapcsolat.....                   | 9  |
| 4.8. Link kulcs.....                       | 9  |
| 4.9. Masterállomány.....                   | 11 |
| 4.10. Master kulcs.....                    | 11 |
| 4.11. Master kulcsérték.....               | 11 |
| 4.12. Misam állomány.....                  | 11 |
| 4.13. Munkakönyvtár.....                   | 12 |
| 4.14. Query, lekérdezés.....               | 12 |
| 4.15. Query definíció.....                 | 12 |
| 4.16. Set kapcsolat.....                   | 13 |
| 4.17. View.....                            | 13 |
| 5. Queryleíró nyelv.....                   | 15 |
| 5.1. Misam állomány hivatkozások.....      | 15 |
| 5.1.1. isam_no=.....                       | 15 |
| 5.1.2. master=.....                        | 16 |
| 5.1.3. isam=.....                          | 16 |
| 5.2. Link kapcsolatok kijelölése.....      | 16 |
| 5.2.1. linked=.....                        | 17 |
| 5.2.2. linked_no=.....                     | 17 |
| 5.2.3. isam_name=.....                     | 18 |
| 5.2.4. partno=.....                        | 18 |
| 5.2.5. parts=.....                         | 19 |
| 5.2.6. key=.....                           | 19 |
| 5.3. A view egy sorának megadása.....      | 20 |
| 5.3.1. detailfields=.....                  | 20 |
| 5.3.2. field=.....                         | 21 |
| 5.4. A view rendezettségének megadása..... | 22 |
| 5.4.1. tmplib=.....                        | 22 |
| 5.4.2. partno=.....                        | 22 |
| 5.4.3. parts=.....                         | 23 |
| 5.4.4. first key.....                      | 23 |
| 5.4.5. last_key=.....                      | 24 |
| 5.4.6. intervallum=.....                   | 24 |
| 5.4.7. sel_func=.....                      | 24 |
| 5.5. Lekérdezési mód megadása.....         | 25 |



|                                           |    |
|-------------------------------------------|----|
| 6. Query editor.....                      | 29 |
| 6.1. A QRYEDIT funkciója.....             | 29 |
| 6.2. A QRYEDIT hívása és opciói.....      | 29 |
| 6.3. A QRYEDIT kezelése.....              | 30 |
| 6.3.1. Magyarázó szöveg.....              | 31 |
| 6.3.2. Misam állományok kiválasztása..... | 31 |
| 6.3.3. Master misam kijelölése.....       | 33 |
| 6.3.4. Link kapcsolatok kijelölése.....   | 33 |
| 6.3.5. A view megadása.....               | 35 |
| 6.3.6. Munkakönyvtár megadása.....        | 37 |
| 6.3.7. Master kulcs összeállítás.....     | 37 |
| 6.3.8. Intervallum megadása.....          | 38 |
| 6.3.9. Az intervallum kezelési mód.....   | 39 |
| 6.3.10. Lekérdezési mód megadása.....     | 40 |
| 6.3.11. Szelekciós függvény.....          | 40 |
| 6.3.12. Misam leírók könyvtára.....       | 41 |
| 6.3.13. A leírófile kiírása.....          | 42 |
| 6.3.14. Konzisztencia vizsgálat.....      | 43 |
| 7. A MIQUEL könyvtár.....                 | 45 |
| 7.1. Előkészítő modulok.....              | 50 |
| 7.1.1. q_open.....                        | 50 |
| 7.1.2. q_one_file.....                    | 51 |
| 7.1.3. q_set_new_key.....                 | 51 |
| 7.1.4. q_set_first.....                   | 52 |
| 7.1.5. q_set_last.....                    | 53 |
| 7.1.6. q_clear_first.....                 | 53 |
| 7.1.7. q_clear_last.....                  | 53 |
| 7.1.8. q_work_lib.....                    | 54 |
| 7.1.9. q_int_mod.....                     | 54 |
| 7.1.10. q_select_key.....                 | 54 |
| 7.1.11. q_clear_ndx.....                  | 55 |
| 7.1.12. q_clear_all_ndx.....              | 55 |
| 7.1.13. q_init.....                       | 56 |
| 7.2. Aktív view állítás.....              | 57 |
| 7.2.1. q_select.....                      | 57 |
| 7.2.2. q_close.....                       | 58 |
| 7.3. Query információk.....               | 58 |
| 7.3.1. q_get_detail.....                  | 58 |
| 7.3.2. q_get_act.....                     | 59 |
| 7.3.3. q_get_misam.....                   | 59 |
| 7.3.4. q_get_fdf.....                     | 59 |
| 7.3.5. q_det_size.....                    | 60 |
| 7.3.6. q_is_match.....                    | 60 |
| 7.3.7. q_get_first.....                   | 61 |
| 7.3.8. q_get_last.....                    | 61 |
| 7.3.9. q_link_sernum.....                 | 62 |
| 7.3.10. q_link_name.....                  | 62 |
| 7.3.11. q_query_sernum.....               | 62 |
| 7.3.12. q_isam_number.....                | 63 |
| 7.4. Query végrehajtás.....               | 63 |
| 7.4.1. q_exec.....                        | 63 |



|                                       |     |
|---------------------------------------|-----|
| 7.4.2. q_next_link.....               | 64  |
| 7.4.3. q_link_init.....               | 64  |
| 7.5. A view mezőinek kezelése.....    | 65  |
| 7.5.1. q_field_name.....              | 65  |
| 7.5.2. q_field_sernum.....            | 65  |
| 7.5.3. q_field_offset.....            | 66  |
| 7.5.4. q_field_size.....              | 66  |
| 7.5.5. q_field_length.....            | 66  |
| 7.5.6. q_field_type.....              | 67  |
| 7.5.7. q_field_dec.....               | 67  |
| 7.5.8. q_field_set.....               | 67  |
| 7.5.9. q_field_get.....               | 68  |
| 7.5.10. q_field_is_valued.....        | 68  |
| 7.5.11. q_field_number.....           | 68  |
| 7.6. Interaktív lekérdezés.....       | 69  |
| 7.6.1. q_build key.....               | 69  |
| 7.6.2. q_scan_key.....                | 70  |
| 7.7. Módosító modulok.....            | 70  |
| 7.7.1. q_upd_det.....                 | 70  |
| 7.7.2. q_upd_rec.....                 | 71  |
| 8. Lekérdező programok készítése..... | 73  |
| 8.1. Standard lekérdező program.....  | 73  |
| 8.2. Keretprogramok írása.....        | 85  |
| 8.2.1. Egy file lekérdezése.....      | 86  |
| 8.2.2. Több file listázása.....       | 90  |
| 8.2.3. Több file lekérdezése.....     | 96  |
| 9. Query dokumenter.....              | 105 |
| 10. Lekérdezés hálózatban.....        | 109 |
| 11. A disztributív anyag.....         | 111 |







## 1. Bevezetés

A hálózati adatfeldolgozási programrendszerek fejlesztésének egyik, sok szempontból előnyös, módszere a hagyományos programnyelvekből indul ki. A leginkább használt nyelvek a COBOL és a C. Az IBM XT/AT gépeknél, a NOVELL hálózatoknál és a UNIX rendszereknél a C nyelv tűnik előnyösebbnek. Az általános célú programnyelvek előnye a "csomagolt" rendszerekkel (pl. a dBase, Dataflex) szemben a fejlesztés során a kötetlenség, a fejlesztett rendszereknél a gyorsaság, a jobb átvihetőség. Ezek az előnyök más szempontból komoly hátrányokat jelentenek: tekintettel arra, hogy a hagyományos programnyelvek az adatfeldolgozást nem támogatják annyira, mint a csomagolt rendszerek, a kötetlenségért és a gyorsaságért komoly fejlesztési munkatöbblettel kell fizetni. Ezt a nehézséget úgy tudjuk megkerülni, ha az adatfeldolgozási funkciók elvégzésére hagyományos programnyelvekből hívható adatbáziskezelő tárgykönyvtárakat használunk. Használatukkal a fejlesztési munka a csomagolt rendszerekkel azonos, megmarad ugyanakkor a programnyelvi közelítés összes előnye.

A MISAM, C nyelvből hívható olyan indexelt szekvenciális file-kezelő rendszer, amely az adatállományok kényelmes kezelését teszi lehetővé. A MISAM és a hozzá hasonló file-kezelő rendszerek (BTRIEVE, C-TREE stb.) problémája az, hogy az általuk kezelt adatállományok nem alkotnak egy egységes adatbázist. Az adatállományok közötti logikai kapcsolatok egyszerűbb kezelését tudjuk megvalósítani, ha a file-kezelő mellett egy lekérdező rendszert is használunk. A MIQUEL, a MISAM file-kezelőnek a lekérdezőrendszere. Csak a MISAM-mal létrehozott állományok hálózatos vagy egyfelhasználós lekérdezésére használható. A MIQUEL, követve a MI Software termékek (MISAM, MFORM, MIREP) elvét, leírófile által vezérelt rendszer. A leírófile-ban kell megadni a lekérdezés adattartalmát, a rendezettségét és a megjelenítési formáját. A leírófile és a futtatórendszer együttesen biztosítják a kényelmes és rugalmas lekérdezést. A leírófile-ban definiált file-kapcsolatok megteremtik a különálló adatállományok között azt a logikai kapcsolatot amelyek egységes adatbázist hoznak létre.

A MIQUEL futtatórendszer a lekérdező programmal memória buffereken keresztül tart kapcsolatot, így a lekérdező rendszertől kapott információk könnyen átadhatók más rendszereknek (MFORM, MIREP), amelyek az adatok megjelenítését végzik.

A futtatórendszer állandó kapcsolatban van a lekérdezett adatállományokkal, így a lekérdezés során az állományok módosítása is lehetséges.







## 2. A lekérdező rendszer alapelve

A lekérdező rendszer célja:

a./ egyszerűsítse a programozási munkát,

b./ lehetőséget adjon a felhasználónak arra, hogy a lekérdezésbe bevont adatokat és azok rendezettségét tetszés szerint változtassa.

A MIQUEL lekérdező rendszer csak a MISAM file-kezelővel létrehozott állományokon működik.

A lekérdezést a leírófile vezérli. A leírófile-t a QRYEDIT nevű query editorral hozhatjuk létre. A programok a MIQUEL tárgy-könyvtár moduljain keresztül a leírófile-t használják. A táblázatos közelítésmód miatt a leírófile változtatása általában nem igényel programváltoztatást.

A leírófile kulcsszavas, karakteres formájú, közönséges szövegszerkesztővel is kezelhető.

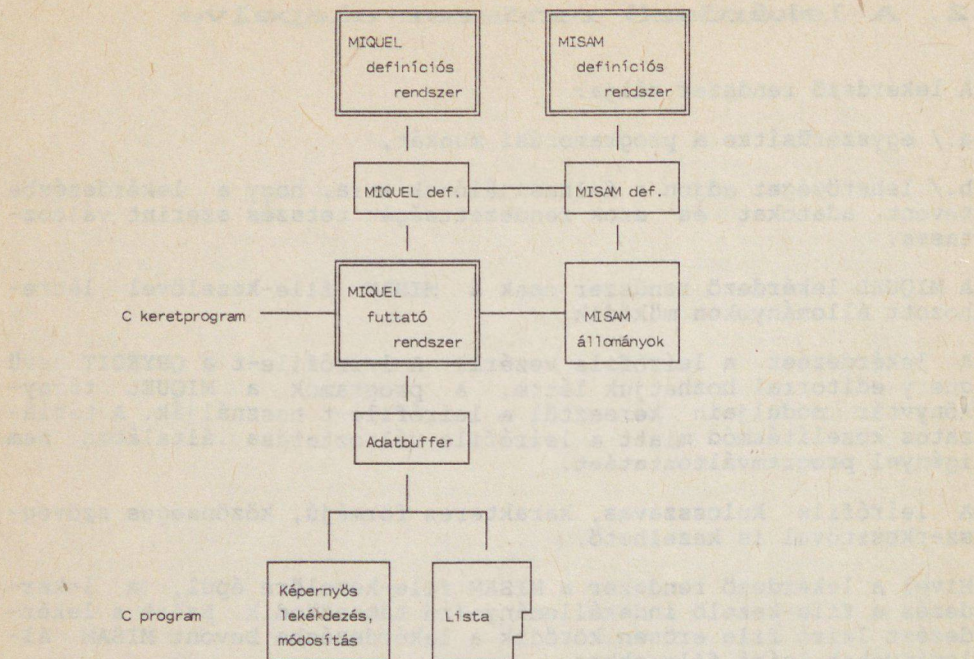
Mivel a lekérdező rendszer a MISAM file-kezelőre épül, a lekérdezés a file-kezelő indexállományaira támaszkodik. Ezért a lekérdezést leíró file erősen kötődik a lekérdezésbe bevont MISAM állományokat leíró file-okhoz.

A rendszer lehetővé teszi egy vagy több MISAM állomány együttes lekérdezését. Az igazi erejét a több MISAM állományból kialakított adatbázis együttes kezelése adja. Az ily módon létrehozott lekérdező programokat kezelő operátor számára az adatbázis fizikai elhelyezkedése rejtve marad.

A MIQUEL rendszer a kapott információkat egy elemi adatbufferben adja át a programnak. Ezzel az adatbufferrel biztosítható a csatlakozás az MFORM vagy a MIREP rendszerekhez.

A MIQUEL koncepcióját szemlélteti az 1. ábra.





1. ábra  
A MIQUEL használatának alapelvei.



### 3. A MIQUEL könyv felépítése

A MIQUEL dokumentációjában először az alapfogalmakat ismertetjük. Ezt követi a lekérdezést leíró file szerkezetének leírása, vagyis annak a kulcsszavas szerkezetű "nyelvnek" az ismertetése, amellyel a lekérdezés jellemzőit adjuk meg. A leírófile szerkezetének a megismerése azért is fontos, mert a lekérdező rendszer lehetőségeit részletesen bemutatja. Ugyanilyen mélységű, de más közelítésű tárgyalást ad a Query EDitor segédprogram ismertetése. Az egyik nyelvleírás szerűen, a másik egy interaktív eszközön (értelmezőn) keresztül mutatja be ugyanazt a témát.

A hetedik fejezetben részletesen tárgyaljuk a MIQUEL tárgykönyvtár moduljait.

Ezután a MIQUEL használatának lehetőségeit ismertetjük. Ezen belül szó lesz a standard lekérdező programról, a programozott standard megoldásról, több állomány együttes lekérdezéséről valamint az illesztett C modulok kezeléséről

A kilencedik fejezet a MIQUEL leírófile-ről dokumentációt készítő QRYDOC program ismertetését tartalmazza.

A tizedik fejezetben a MIQUEL rendszer hálózatos alkalmazásával foglalkozunk.







## 4. Alapfogalmak és definíciók

A dokumentációban gyakran használunk olyan fogalmakat, amelyeknek a jelentése nem mindenki számára azonos. Az következő kis szótár ezen fogalmakat teszi egyértelművé. A fogalmak alfabetikus sorrendben követik egymást.

### 4.1. Alias név

A lekérdezésbe bevont MISAM állományokat a MIQUEL rendszer egy névvel azonosítja. Ennek a névnek egyedinek kell lennie. Alapértelmezésben ezt a nevet a MISAM állományt leíró file tartalmazza. Ha ugyanazt az állományt a link kapcsolatokban több helyen is akarjuk szerepeltetni, akkor adunk az állománynak egy alias nevet. Így egy MISAM állományt különböző neveken többször is bevonhatunk a lekérdezésbe.

A továbbiakban a MISAM állomány nevén mindig az alias nevet értjük. Ha nem adtunk alias nevet, akkor ez természetesen meg egyezik a MISAM eredeti nevével.

### 4.2. Illesztési pontok

A lekérdezés bizonyos pontjain C függvények hívhatók. E pontokat nevezzük illesztési pontoknak.

Kétféle C függvény illeszthető:

- A szelekciós függvény segítségével a view sorai az intervallum kijelölésnél kifinomultabban is szelektálhatók. A függvény jelzi a futtatórendszer számára, hogy az adott sort elhagyja vagy ne a lekérdezésből.
- A mezőbetöltő függvények a view egy sorának a feltöltésekor hívódnak meg. Így lehetőség van a view sorain való manipulálásra, származtatott értékek képzésére.

### 4.3. Intervallum

Több lehetőség van arra, hogy a view bizonyos sorait elhagyjuk a lekérdezésből. Ennek a legegyszerűbb módja egy intervallum kijelölése. Az intervallumot a master kulcs kezdő és végső értékeivel lehet megadni. A lekérdezésbe csak azok a sorok fognak bekerülni, amelyeknél a master kulcs a két kulcs értéke közé esik.

Az intervallum kezelésének két módja van. A két mód közötti kü-



lönbségnek, akkor van jelentősége, ha a master kulcs több darabból áll. Ilyenkor az intervallum alsó és felső határához való hasonlításakor kezelhetjük a master kulcsot úgyis mintha egy kulcsdarabból állna, és végezhetjük a hasonlítást kulcsdarabonként is. A hasonlítás sok esetben eltérő eredményre vezet a kétféle kezelési móddal.

#### 4.4. Lekérdezési mód

A MIQUEL rendszer kétféle módon tudja végrehajtani a lekérdezést: interaktív és batch módban.

A futtatórendszer alapértelmezésben interaktív módban működik. Ez azt jelenti, hogy minden egyes view-sorkérésnél olvassa a szükséges MISAM állományokat és az aktuális MIASM rekordokból összeállítja a view-sort. Ily módon a felhasználói program közvetlen kapcsolatban van a lekérdezésbe bevont MISAM állományokkal, azokon módosításokat is el tud végezni. Hátránya ennek a módnak, hogy a view-t csak a masterfile alapján lehet rendezni.

Batch módban mielőtt megkapnánk az első view-sort a rendszer egy ideiglenes MISAM állományban felépíti az egész view-t, és a továbbiakban ebből kapjuk a view-sorokat. Az ideiglenes állomány a felhasználói program futása után törlődik. Ezzel a módszerrel elvesztjük a közvetlen kapcsolatot az eredeti MISAM állományokkal, de lehetőség van a rendezésére olyan oszlopok alapján is, amelyeket nem tartalmaz a masterállomány.

A batch mód másik előnye akkor jelentkezik, ha a view olyan rendezettségére van szükségünk, amihez a futtatórendszernek ideiglenes indexállományt kell építenie, ugyanakkor csak a view egy intervallumára vagyunk kíváncsiak. Ha nem batch módban használjuk a MIQUEL-t, akkor függetlenül az intervallumtól a futtatórendszer az egész masterállományra épít indexet. Batch mód esetén azonban csak az intervallumba eső rekordokra épül indexfile. Ily módon jelentősen csökkenhet az index építésre fordított idő.

#### 4.5. Lekérdező rendszer

A lekérdező rendszer a lekérdezéseket automatizáló, illetve segítő software. A MIQUEL lekérdező rendszer lényegében két részből áll: a lekérdezési definícióból és az azt értelmező részből. Az értelmező rész egy önálló C program.

A MIQUEL-hez tartozik egy önálló végrehajtó program (STDQRY), de ez csak viszonylag egyszerű lekérdezésekre képes. A lekérdezé-



seket általában olyan C programok hajtják végre, amelyek a MIQUEL tárgykönyvtár moduljait használják.

A lekérdezés menetét a megfelelő helyen illesztett saját C modulokkal tudjuk befolyásolni.

Az lekérdezés tipikus feldolgozási ciklusa a 2. ábrán látható.

#### 4.6. Linkelt állomány

Azokat az állományokat nevezzük linkelt állományoknak, amelyek valamilyen index alapján hozzá vannak rendelve egy másik állományhoz. A linkelt állományokhoz is lehet hozzárendelni másik állományt. Ily módon az állományok között egy hierarchikus kapcsolat jöhet létre.

A masterállományt nem lehet semmihez hozzárendelni.

#### 4.7. Link kapcsolat

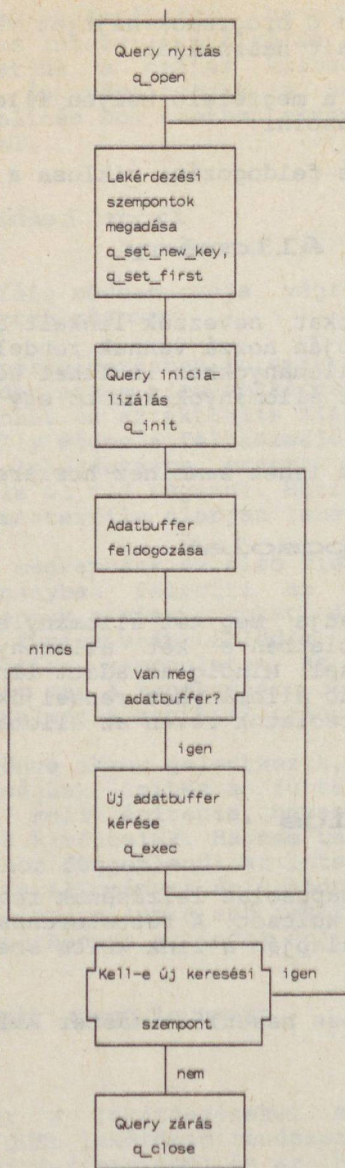
A link kapcsolat adja meg két állomány egymáshoz rendelésének módját. A link kapcsolatban a két állomány alárendeltségi viszonyban van egymással. Mindig az adott link kapcsolatban master állományként viselkedő állományhoz rendeljük hozzá a linkelt állományt. A link kapcsolatok révén az állományokból egy fa struktúra jön létre.

#### 4.8. Link kulcs

A link kulcs a link kapcsolat leírásának része. A linkelt állományon definiál egy kulcsot. A futtatórendszer a master kulcsértékből kapott kulcs alapján a link kulcs szerint keres rekordot a linkelt állományban.

A kulcsérték kijelölése hasonló a master kulcs kijelöléshez.





2. ábra  
Tipikus lekérdezési ciklus.



#### 4.9. Masterállomány

A lekérdezésbe egy vagy több MISAM állományt lehet bevonni. Több állomány esetén az állományok között az indexfile-jaik teremtenek kapcsolatot. Az egyik állomány rekordjából képzett kulcs alapján lehet elérni a másik állományból hozzá tartozó rekordot. Azt az állományt nevezzük masterállománynak, amelynek a rekordjai alapján érjük el a többi állományt.

#### 4.10. Master kulcs

A master kulccsal lehet megadni a view sorainak a rendezettségét. Ez gyakorlatilag egy kulcs kijelölése a masterállományon. A kulcs megadása némileg eltér a MISAM file-definícióban használatos módszertől. A kulcs itt is kulcsdarabok megadásának soroztából áll, de a kulcsdarabot a hozzá tartozó mező neve, a kulcsdarab hossza és típusa azonosítja.

A leírtak alapján látszik, hogy a view csak a masterállomány rekordjainak mezőiből képzett kulcs alapján rendezhető. Az ettől eltérő rendezettséggel a lekérdezési mód tárgyalásánál foglalkozunk.

#### 4.11. Master kulcsérték

A master kulcsérték (nem keverendő össze a master kulccsal, ami a view rendezettségét adja meg) a link kapcsolat leírásának a része. A kapcsolatban masterként viselkedő állomány rekordján jelöl ki egy kulcsot, amely kulcs aktuális értéke alapján lehet a hozzá rendelt linkelt állomány megfelelő rekordját elérni.

A kulcsérték kijelölése hasonló a master kulcs kijelöléshez.

#### 4.12. Misam állomány

A MISAM file-kezelő rendszerrel létrehozott, névvel azonosított adatállomány. Fix hosszúságú rekordokból áll és a rekordok névvel azonosított mezőkből épülnek fel. Az állományok különböző indexállományok szerint rendezve vannak.

Minden MISAM állományhoz tartozik leírófile, amely a MISAM állomány összes jellemzőjét tartalmazza.



Több, egymással indexeik alapján logikailag összefüggő MISAM állomány alkothat egy adatbázist.

A MIQUEL rendszer a MISAM állományok rekordjaiból képes adatokat szolgáltatni az indexállományok alapján.

#### 4.13. Munkakönyvtár

A MIQUEL rendszer futás közben létrehozhat olyan ideiglenes állományokat, amelyeket a futás befejeztével eltöröl. A munkakönyvtár az a DOS könyvtár, ahol ezek az állományok képződnek.

#### 4.14. Query, lekérdezés

Egy adatbázisból különböző információk elérése. Jellemző rá:

- adattartalma és
- rendezettsége.

Az adattartalom alatt azokat az információkat értjük, amelyekre kíváncsiak vagyunk az adott adatbázisból.

A rendezettség a kapott információk sorrendjét adja meg.

A lekérdezés eredménye általában nyomtatón vagy képernyőn jelenik meg.

#### 4.15. Query definíció

Táblázatos vagy nyelvi formában leírja a lekérdezés szempontjait.

A definíció áll:

- a lekérdezésbe bevont MISAM állományok megadásából,
- az állományok közötti kapcsolat leírásából,
- az adatleírásból,
- az adatok rendezettségének megadásából,
- a lekérdezési mód megadásából.



#### 4.16. Set kapcsolat

A set kapcsolat része a link kapcsolat leírásának. Egy adott master kulcsértékhez tartozhat egy vagy több rekord is a linkelt állományban. Ha több ilyen rekord is van, akkor a master kulcsértékhez egy úgynevezet set tartozik. Ha azt akarjuk, hogy a futtatórendszer az összes ilyen rekordot automatikusan feldolgozza, akkor ezt kell a set kapcsolatban jelezni.

#### 4.17. View

A lekérdezés eredménye a view (vjú). A view sorokból és névvel azonosított oszlopokból áll. A sor tekinthető egy MISAM állomány olyan rekordjaiként is amelynek a mezői a view oszlopai. A view sorai valamilyen index szerint rendezettek. Ily módon a view felfogható úgyis mint egy MISAM állomány, azzal a különbséggel, hogy a view általában fizikailag nem létezik.

A view definíciója a lekérdezési definícióban található. A view oszlopai vagy a lekérdezésbe bevont MISAM állományok rekordmezőiből kaphatnak értéket, vagy származtatott mezők esetében, felhasználói C függvénytől. A view egy oszlopát a neve azonosítja, valamint annak a MISAM állománynak a neve amelyikből az adat származik. Az oszlop neve megegyezhet annak a mezőnek a nevével amelyikből az adatot nyerjük.

A query egyszeri végrehajtása a view következő sorát tölti a C keretprogram által kezelt adatbufferbe.







## 5. Queryleíró nyelv

A query leírása kulcsszavas. Általános formája:

<kulcsszó>=<argumentum>

Pl.  
ISAM\_NO=4

A leírási parancsokat csoportokra oszthatjuk:

- MISAM állomány hivatkozások.
- Link kapcsolatok kijelölése.
- A view egy sorának megadása.
- A view rendezettségének megadása.
- A lekérdezési mód megadása.

A \* jellel kezdett sorok a magyarázat sorok.

A leírófile elején, dokumentációs célból, egy magyarázó szöveget helyezhetünk el. Ezt a szöveget a QRYDOC program értelmezi.

Formája:

DESCR=<magyarázó szöveg>

Példa:

DESCR=Példa leírófile

### 5.1. Misam állomány hivatkozások

A MISAM állomány hivatkozások parancsaival a lekérdezésbe bevont MISAM állományokat kell definiálni. Meg kell adni a bevont állományok leírófile-ját az abszolút vagy a relatív path-al, valamint az állomány alias nevét. Ha nem adunk alias nevet, akkor ez a MISAM leírófile-ban található név lesz. A lekérdezési definícióban ezután a MISAM állományokra az alias név alapján lehet hivatkozni.

#### 5.1.1. isam\_no=

Funkciója:

A lekérdezésbe bevont MISAM állományok számának megadása.



Formája:

ISAM\_NO=<a lekérdezésbe bevont állományok száma>

Példa:

ISAM\_NO=5

### 5.1.2. master=

Funkciója:

A masterállomány leírójának megadása.

Formája:

MASTER=<a file-leíró neve abszolút vagy relatív path-al>,  
[,<alias név>]

Példa:

MASTER=C:\QLIB\\_FILEDEF\CIKK.FDF,CIKK

### 5.1.3. isam=

Funkciója:

A többi állomány file-leírójának megadása.

Formája:

ISAM=<a file-leíró neve abszolút vagy relatív path-al>,  
[,<alias név>]

Példa:

ISAM=C:\QLIB\\_FILEDEF\VEVO.FDF,VEVO  
ISAM=C:\QLIB\\_FILEDEF\VERE.FDF,VERE  
ISAM=C:\QLIB\\_FILEDEF\ESE.FDF,ESE  
ISAM=C:\QLIB\\_FILEDEF\TEREL.FDF,TEREL

## 5.2. Link kapcsolatok kijelölése

Ebben a részben az állományok közötti link kapcsolatokat kell egyenként leírni. Mindig a masterállománytól kell elindulni és



lefelé haladni. Csak olyan állományokhoz lehet más állományokat hozzárendelni, amelyet már elhelyeztünk a link struktúrában.

A link kapcsolatok megadásánál először jelezni kell, hogy melyik állományhoz akarunk más állományokat kapcsolni. Ezt a LINKED kulcsszóval tudjuk megadni. A hozzárendelt állományok számát a LINKED NO kulcsszó jelöli. Ezután következik az egyes kapcsolatok leírása a kijelölt állományra.

Ennek tartalmaznia kell a csatolt állomány nevét (ISAM\_NAME), a master kulcsérték és a link kulcs leírását egymás után, valamint azt, hogy a link kulcs egyedi kulcs-e. A master kulcsérték és a link kulcs megadása hasonló. Először a PARTNO kulcsszóval megadjuk, hogy hány darabból áll a kulcs, ezután a PARTS kulcsszóval egyenként leírjuk a kulcsdarabokat. Ezután a kulcsszó után négy érték következik, amelyek vesszővel vannak elválasztva. Az első érték annak a rekordmezőnek a neve, amelyben az adott kulcsdarab található, a második érték a kulcsdarab mezőn belüli kezdő pozícióját jelöli, a harmadik pedig a kulcsdarab hosszát adja meg. A negyedik érték jelzi a kulcsdarab típusát a MISAM file-leíró konvenciója szerint.

Ha a KEY kulcsszó értéke UNIQUE, akkor a futtatórendszer a kijelölt link indexet egyedi indexként kezeli. Ha SET-et adunk, akkor a futtatórendszer mielőtt tovább lépne a master rekordon előbb feldolgozza az azonos link indexeket.

### 5.2.1. linked=

Funkciója:

Az adott link kapcsolatban masterként viselkedő állomány nevének megadása.

Formája:

LINKED=<az állomány neve>

Példa:

LINKED=CIKK

### 5.2.2. linked\_no=

Funkciója:

A hozzárendelt állományok számának megadása.



Formája:

LINKED\_NO=<a hozzárendelt állományok száma>

Példa:

LINKED\_NO=1

### 5.2.3. isam\_name=

Funkciója:

Egy hozzárendelt állomány nevének megadása.

Formája:

ISAM\_NAME=<a hozzárendelt állomány neve>

Példa:

ISAM\_NAME=VERE

### 5.2.4. partno=

Funkciója:

A kucsdarabok számának megadása.

Formája:

PARTNO=<a kulcsdarabok száma>

Példa:

PARTNO=2



### 5.2.5. parts=

Funkciója:

Az egyes kulcsdarabok leírása.

Formája:

```
PARTS= <a hozzátartozó rekordmező neve>,
      <a mezőn belüli kezdő pozíció>,
      <a kulcsdarab hossza>,
      <a kulcsdarab típusa>
```

Példa:

```
PARTS=CIKK KOD,0,2,A
PARTS=VEVOKOD,0,2,A
```

### 5.2.6. key=

Funkciója:

Megadja, hogy a link kulcs egyedi, vagy több ilyen kulcsú rekord is lehet.

Formája:

```
KEY=<a kulcs típusa>
```

A kulcs típusa lehet UNIQUE vagy SET.

Példa:

```
KEY=UNIQUE
```

A következő példa egy teljes link struktúra leírása.

\*\*\*\*\*

\* Link definitions

\*

LINKED=CIKK

LINKED\_NO=1

\*

ISAM\_NAME=VERE

PARTNO=1

PARTS=CIKK\_KOD,0,2,A

PARTNO=2

PARTS=CIKK\_KOD,0,2,A



```

PARTS=VEVOKOD,0,2,A
KEY=SET
*
LINKED=VEREL
LINKED_NO=2
*
ISAM_NAME=TEREL
PARTNO=2
PARTS=CIKK_KOD,0,2,A
PARTS=VEVOKOD,0,2,A
PARTNO=2
PARTS=CIKK_KOD,0,2,A
PARTS=VEVOKOD,0,2,A
KEY=SET
*
LINKED=TEREL
LINKED_NO=1
*
ISAM_NAME=ESE
PARTNO=1
PARTS=ESEKOD,0,3,A
PARTNO=1
PARTS=ESEKOD,0,3,A
KEY=UNIQUE
*
ISAM_NAME=VEVO
PARTNO=1
PARTS=VEVOKOD,0,2,A
PARTNO=1
PARTS=VEVOKOD,0,2,A
KEY=UNIQUE

```

### 5.3. A view egy sorának megadása

A sor megadásakor a view oszlopait kell leírni. A DETAILFIELDS kulcsszóval kell megadni, hogy hány oszlopa van egy view-sornak. Ezután a FIELD kulcsszóval kell egyenként felsorolni az oszlopokat.

#### 5.3.1. detailfields=

Funkciója:

A view-oszlopok számának megadása.



Formája:

DETAILFIELDS=<az oszlopok száma>

Példa:

DETAILFIELDS=6

### 5.3.2. field=

Funkciója:

Leírja az egyes oszlopokat.

Formája:

FIELD= <soron belüli offset>,  
 <hossz>,  
 <dimenzió>,  
 <típus>,  
 <kiírási szélesség>,  
 <decimális pont>,  
 <MISAM állomány neve>,  
 <rekordmező neve>,  
 <rekordmezőn belüli offset>,  
 <oszlop név>,  
 <C függvény neve>

- A kiírási szélesség csak az általános STDQRY lekérdező program által használat információ.
- A decimális pontnak csak float és double típusú oszlopoknál van értelme.
- A MISAM állomány neve származtatott oszlop esetén \$USER.
- A rekordmező neve származtatott oszlop esetén tetszőleges, de egyedi név.
- Az oszlopnév alapértelmezésben megegyezik a rekordmező nevével, de megadhatunk más nevet is. A program által generált C struktúrában ezen a néven fog a mező szerepelni, és a futtatórendszer is ezen a néven azonosít.
- C függvénynevet nem kötelező megadni. Jelentősége főleg a származtatott oszlopoknál van.

Példa:

DETAILFIELDS=4

FIELD=0,15,1,A,15,0,FOKONYV,FOKONYV,0,fokonyv

FIELD=15,30,1,A,30,0,FOKONYV,MEGNEV,0,megnev

FIELD=45,8,1,D,12,2,FOKONYV,NYITO,0,nyito

FIELD=53,8,1,D,14,2,FOKONYV,TARTOZIK,0,tart\_sum



## 5.4. A view rendezettségének megadása

A view rendezettségét úgy adjuk meg, hogy kijelölünk egy kulcsot a masterállományra. A kulcs megadása hasonló mint a link kapcsolatok definiálásában a master kulcsérték és a link kulcs kijelölése. A master kulcs megadása előtt megadhatjuk egy DOS könyvtárnak a nevét, ahová a futtatórendszer által képzett ideiglenes állományok kerülnek.

A master kulcs után jelölhetjük ki a keresés intervallumát a FIRST KEY és a LAST KEY kulcsszavakkal. Ha a master index több darabból áll, akkor az intervallum megadásánál az indexdarabok értékeit vesszővel kell elválasztani egymástól.

Az INTRVALLUM kulcsszóval az inetrvallum kezelésének módját adhatjuk meg. Ha ALL-t adunk, akkor a futtatórendszer a kezdő és végértéket egy kulcsértéknek fogja venni. Ha SELECTED módban kérjük az intervallumba esés kiértékelését, akkor a rendszer a kezdő és végértéket kulcsdarabonként fogja vizsgálni.

### 5.4.1. tmplib=

Funkciója:

A munkakönyvtár nevének megadása.

Formája:

TMPLIB=<a könyvtár neve>

Példa:

TMPLIB=C:\TMP

### 5.4.2. partno=

Funkciója:

A master kulcsdarabok számának megadása.

Formája:

PARTNO=<a kulcsdarabok száma>



Példa:

PARTNO=1

### 5.4.3. parts=

Funkciója:

Leírja az egyes kulcsdarabokat.

Formája:

PARTS= <a hozzá tartozó rekordmező neve>,  
 <a mezőn belüli kezdő pozíció>,  
 <a kulcsdarab hossza>,  
 <a kulcsdarab típusa>

A típus megadásánál ugyanazok az értékek szerepelhetnek mint a MISAM file definíciókban. Ha a típus nagybetű, akkor a rendezettség növekvő, kisbetű esetén csökkenő.

Példa:

PARTS=CIKK\_KOD,0,2,A  
 PARTS=VEVOKOD,0,2,A

### 5.4.4. first\_key

Funkciója:

A view soraira kezdő kulcsérték megadása. Összetett kulcs esetén a kulcsdarabokat vesszővel kell elválasztani.

Formája:

FIRST\_KEY=<a kulcsérték>

Példa:

FIRST\_KEY=22,10.56



#### 5.4.5. last\_key=

Funkciója:

A view soraira végső kulcsérték megadása. Összetett kulcs esetén a kulcsdarabokat vesszővel kell elválasztani.

Formája:

LAST\_KEY=<a kulcsérték>

Példa:

LAST\_KEY=55,100.56

#### 5.4.6. intervallum=

Funkciója:

Az intervallum kezelési módjának megadása.

Formája:

INTERVALLUM=<a kezelési mód>

A kezelési mód lehet SELECTED vagy ALL.

Példa:

INTERVALLUM=ALL

#### 5.4.7. sel\_func=

Funkciója:

A felhasználói szelekciós függvény nevének megadása.

Formája:

SEL\_FUNC=<a C függvény neve>

Példa:

SEL\_FUNC=check\_view



## 5.5. Lekérdezési mód megadása

Funkciója:

Megadja a lekérdezés interaktív vagy batch módját.

Formája:

PROCESSING=<a lekérdezési mód>

A lekérdezési mód lehet INTERACTIVE vagy BATCH. Batch mód esetén meg kell adni a view rendezettséget, amely a master kulcshoz hasonló kulcs megadását jelenti. Egy különbség van, ami abból adódik, hogy itt nem csak a masterállományból származtatott oszlopok kerülhetnek be a kulcsba, hanem a view bármelyik oszlopa. Ezért a PARTS kulcsszó első paramétere annak a MISAM állománynak a neve lesz, amelyikből az oszlop származik (származtatott oszlop esetén természetesen \$USER). A többi értéket ugyanúgy kell megadni mint a master kulcsnál.

Batch mód esetén megadhatunk intervallumot és szelekciós függvényt is úgy mint a masterállományra. Ezek a generált ideiglenes állományra vonatkoznak.

Példa a batch módú lekérdezés megadására:

```
*****
*
*      Processing
PROCESSING=BATCH
PARTNO=2
PARTS=FORG_TORZS, ANYAGSZAM, 0, 2, A
PARTS=FORG_TORZS, FORG, 0, 8, D
FIRST_KEY=22,      100.00
LAST_KEY=55,      500.00
INTERVALLUM=ALL
SEL_FUNC=sel_func
```

Végül a következő példa egy teljes lekérdezési definíció.



\*\*\*\*\*

\* "CIKK.QTX" QUERY leírófile.  
\* Generated by Query Editor (MI Software)  
\* on Oct/05/1989 18:08:01

\*\*\*\*\*

\*  
DESCR=Példa leírófile

\*  
\* MISAM files  
\*

ISAM\_NO=5  
MASTER=C:\QLIB\FILEDEF\CIKK.FDF,CIKK  
ISAM=C:\QLIB\FILEDEF\VEVO.FDF,VEVO  
ISAM=C:\QLIB\FILEDEF\VERE.FDF,VERE  
ISAM=C:\QLIB\FILEDEF\ESE.FDF,ESE  
ISAM=C:\QLIB\FILEDEF\TEREL.FDF,TEREL

\*\*\*\*\*

\* Link definitions

\*  
LINKED=CIKK  
LINKED\_NO=1

\*  
ISAM\_NAME=VERE  
PARTNO=1  
PARTS=CIKK\_KOD,0,2,A  
PARTNO=2  
PARTS=CIKK\_KOD,0,2,A  
PARTS=VEVOKOD,0,2,A  
KEY=SET

\*  
LINKED=VERE  
LINKED\_NO=2

\*  
ISAM\_NAME=TEREL  
PARTNO=2  
PARTS=CIKK\_KOD,0,2,A  
PARTS=VEVOKOD,0,2,A  
PARTNO=2  
PARTS=CIKK\_KOD,0,2,A  
PARTS=VEVOKOD,0,2,A  
KEY=SET

\*  
LINKED=TEREL  
LINKED\_NO=1

\*  
ISAM\_NAME=ESE  
PARTNO=1  
PARTS=ESEKOD,0,3,A  
PARTNO=1  
PARTS=ESEKOD,0,3,A  
KEY=UNIQUE



```

*
ISAM_NAME=VEVO
PARTNO=1
PARTS=VEVOKOD,0,2,A
PARTNO=1
PARTS=VEVOKOD,0,2,A
KEY=UNIQUE
*****
*
*      Detail row
*
DETAILFIELDS=7
FIELD=0,2,1,A,2,0,CIKK,CIKK_KOD,0,CIKK_KOD
FIELD=2,10,1,A,10,0,CIKK,CIKKNEV,0,CIKKNEV
FIELD=12,2,1,A,2,0,VEVO,VEVOKOD,0,VEVOKOD
FIELD=14,10,1,A,10,0,VEVO,VEVONEV,0,VEVONEV
FIELD=24,4,1,M,8,0,TEREL,DATUM,0,DATUM
FIELD=28,10,1,A,10,0,ESE,ESENEV,0,ESENEV
FIELD=38,8,1,D,8,2,VERE,RENDELES,0,RENDELES
*****
*
*      Master key
*
TMPLIB=C:\TMP
PARTNO=1
PARTS=CIKK_KOD,0,2,A
FIRST_KEY=aa
LAST_KEY=cc
INTERVALLUM=ALL
SEL_FUNC=select_cikk
*****
*
*      Processing
PROCESSING=INTERACTIVE
*
*      end of CIKK.QTX
*****

```







## 6. Query editor

### 6.1. A QRYEDIT funkciója

A Query Editor programmal párbeszédessé módon hozhatunk létre vagy módosíthatunk egy lekérdezési definíciót. A programnak meg kell adni annak a DOS könyvtárnak a nevét, ahol az érintett MISAM állományok leírói találhatóak. A leírókban lévő információkból kiindulva tudjuk a programmal a lekérdezési definíciót létrehozni. A QRYEDIT-el a leírófile-on kívül két C header file-t is generálhatunk. Az egyikben definiálhatunk konstansokat a querybe bevont MISAM állományokra és a view-sor mezőire. A másikban a felhasználó által definiált C függvények táblája található.

### 6.2. A QRYEDIT hívása és opciói

Ha a programot -h opcióval hívjuk meg, akkor kiírja a hívási opcióit:

QErYeDit Version 1.00 Nov/18/1988 Produced by MI Software

Usage: >QRYEDIT [-o headerpath] [-d fdpath] [-i] [-b] queryfile

|       |            |                                     |
|-------|------------|-------------------------------------|
| where | headerpath | is the directory of .QH output file |
|       | fdpath     | is the directory of .FDF files      |
|       | -i         | ignore the start picture            |
|       | -b         | means Black and White mode          |
|       | queryfile  | is the name of queryfile            |

- A -o opció után megadhatjuk egy DOS könyvtár nevét, ahová a QRYEDIT-el generált header file-okat akarjuk tenni. Alapértelmezésben ez az aktuális könyvtár.
- A -d opció után annak a DOS könyvtárnak a nevét kell megadni, ahol a MISAM állományok leírói találhatóak. Alapértelmezésben ez az aktuális könyvtár.
- A -i opció megadása esetén nem jelenik meg a bejelentkező képernyő.
- A -b opcióval jelezhetjük a programnak, hogy nem színes monitoron akarjuk futtatni.
- Ha a megadott queryleíró létezik, akkor a program ennek a módosítását teszi lehetővé. Ha a leíró még nincs a könyvtárban, akkor egy új leíróhoz hozhatunk létre.

Ha nem tiltottuk le, akkor indítás után a képernyőre beúszik a következő bejelentkező kép.



```
QRYEDIT V1.00
Produced by MI Software

Press any key to start
and
use F1 for help.
```

Bármilyen billentyű leütése után jelenik meg a kezdő képernyő.

### 6.3. A QRYEDIT kezelése

A program helyes meghívása után a képernyőn a 3. ábrán látható ablak jelenik meg. Ebben választhatjuk ki a QRYEDIT megfelelő funkcióját.

```
QUERY EDITOR
A: query description
B: select misams
C: select master misam
D: link definition
E: detail row definition
F: work directory
G: master key definition
H: first key value
I: last key value
J: handling of intervall
K: batch method definition
L: select function
M: directory of misam descriptions
N: save to file
O: save header file
P: save function table
R: check query description
```

3. ábra  
QRYEDIT fő menü

A kiválsztás két módon történhet. Vagy ráállunk a kurzorvezérlő billentyűkkel a kívánt funkcióra és leütjük az ENTER billentyűt, vagy a szövegek előtt található nagybetűvel választunk. A kiválasztott funkciókból az ESC billentyűvel tudunk a fő menübe visszatérni.

A QRYEDIT programból az ESC billentyűvel léphetünk ki. Ha a program futása közben változtattunk az indulási állapoton és ezt



nem mentettük, akkor a program kiszállás előtt megkérdezi, hogy mentjük-e az új állapotot.

A QRYEDIT program futása közben az F9 billentyűvel ki tudunk lépni a DOS-ba, ahonnan az EXIT utasítással térhetünk vissza a programba.

### 6.3.1. Magyarázó szöveg

#### query desription

Itt egy dokumentációs célú magyarázó szöveget adhatunk meg amit a QRYDOC program értelmez. A szöveget a következő ablakban adhatjuk meg:

Description: Példa leírófile

### 6.3.2. Misam állományok kiválasztása

#### Select misam

Ezzel a funkcióval lehet kiválasztani a lekérdezésbe bevont állományokat. A képernyő jobb sarkában megjelennek a QRYEDIT hívásakor megadott könyvtárban található MISAM leírók (ld. 4. ábra). Az ENTER billentyűvel kijelölt állományok a képernyő bal sarkában láthatók. Az ESC billentyűvel tudunk átmenni a baloldali ablakba, ahol a DEL billentyűvel törölhetünk állományokat. Ha a funkció választásakor már vannak kiválasztott állományok, akkor csak a baloldali ablak jelenik meg, és az INS billentyűvel tudunk a jobboldali ablakba átlépni.

A baloldali ablakban a MISAM leírók mellett megjelenik a MISAM állományok alias nevei is. A program ide az állomány eredeti nevét teszi be. Ha alias nevet akarunk adni, akkor rá kell állni a kívánt sorra és az ENTER billentyűvel adhatunk nevet (ld. 5. ábra).



| Selected MISAMS | Alias   |
|-----------------|---------|
| CIKK            | , CIKK  |
| VEVO            | , VEVO  |
| VERE            | , VERE  |
| ESE             | , ESE   |
| TEREL           | , TEREL |

| MISAM names |
|-------------|
| AFORG       |
| ANYAG       |
| BIZ         |
| CIKK        |
| ESE         |
| FOKONYV     |
| FORGALOM    |
| KESZLET     |
| RAKT        |
| SZALL       |
| TELEP       |
| TEREL       |
| TERM_ANY    |
| TFORG       |
| TORZS       |
| TYPES       |
| UTOR        |
| VERE        |
| VEVO        |

F1,ESC

4. ábra  
MISAM állományválasztás.

| Selected MISAMS | Alias   |
|-----------------|---------|
| CIKK            | , CIKK  |
| VEVO            | , VEVO  |
| VERE            | , VERE  |
| ESE             | , ESE   |
| TEREL           | , TEREL |

Alias name: VERE

ESC

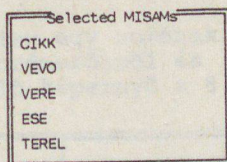
5. ábra  
Alias név adása.



### 6.3.3. Master misam kijelölése

#### select Master misam

A képernyő baloldalán megjelennek a kiválasztott MISAM állományok. Az ENTER billentyűvel tudjuk a masterállományt kiválasztani. A master mindig a legelső állomány az ablakban. Ha már definiáltunk link kapcsolatokat, akkor a masterállomány már nem változtatható.



6. ábra  
Master MISAM választás.

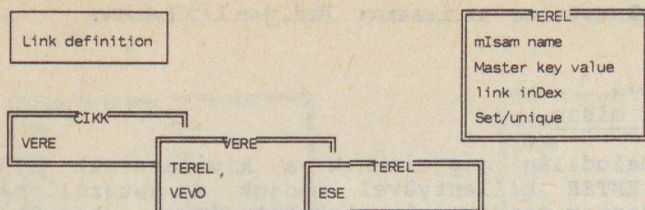
### 6.3.4. Link kapcsolatok kijelölése

#### Link definition

A link struktúrát a masterállományból kiindulva lefelé tudjuk definiálni. Amikor először lépünk ide be, akkor a masterhez kapcsolhatunk állományokat. Az egyes linkek definiálását a jobb felső sarokban lévő menüvel valósíthatjuk meg. A menü tetején annak az állománynak a neve látható, amelyik az adott link kapcsolatban masterként viselkedik.

A már definiált link struktúra aktuális ága mindig látható a képernyőn (ld. 7. ábra).





|             |                                |
|-------------|--------------------------------|
| INS, CTRL_A | : New link on the actual level |
| ALT_M       | : Modify actual link           |
| ENTER       | : New link to the actual MISAM |

7. ábra  
Link kapcsolatok megadása.

- mIsam name: Itt kell megadni annak az állománynak a nevét, amelyiket csatolni akarjuk.
- Master key value: Itt kell kijelölni a master rekordon azokat a részeket, amelyekből a master kulcsértéket kell számolni. A master megnevezés itt természetesen relatív, mert lehet, hogy csak az adott link kapcsolaton belül minősül masternek a rekord. A kulcskijelölés részletes leírása a master kulcs összeállításnál található.
- link inDex: Itt kell kijelölni a csatolt file rekordján azokat a részeket amelyek a link kulcsot alkotják. A kulcs-kijelölés részletes leírása a master kulcs összeállításnál található.
- Set/unique: Itt kell megadni, hogy a link kulcsot UNIQUE vagy SET módon kezelje a futtatórendszer. Az alapértelmezés UNIQUE.

A link kapcsolat megadását az ESC-el tudjuk befejezni. Ekkor a képernyőn megjelenik a hozzárendelt állomány név a link struktúrában és új állományt kapcsolhatunk az adott szinthez. Ha nem akarunk az adott szinten több link kapcsolatot definiálni, akkor ESC-el hagyhatjuk el a jobb felső sarokban lévő menüt. Ha alsóbb szintű link kapcsolatot akarunk definiálni, akkor a link struktúrában ráállunk arra az állományra, amelyikhez újat akarunk hozzárendelni és leütjük az ENTER billentyűt. A link kapcsolatot definiáló menü tetején ennek az állománynak a neve fog megjelenni, mint aktuális masterfile.



Az ily módon létrejött link struktúra a képernyő bal felső sarkában látható, amelyen az ENTER billentyűvel lefelé, az ESC billentyűvel pedig felfelé mozoghatunk. Az aktuális link kapcsolat az ALT M billentyű kombinációval módosítható, a DEL billentyűvel törölhető a link struktúrából. Az INS billentyűvel az adott szintre tudunk új link kapcsolatot beszúrni.

### 6.3.5. A view megadása

#### Detail row definition

Itt kell definiálni a view egy sorának felépítését a lekérdezésbe bevont állományok rekordmezőiből és az esetleges származtatott mezőkből. A definiáló képernyő a 8. ábrán látható.

| Detail row definition: cikk.GTX |            |             |      |      |      |       |     |
|---------------------------------|------------|-------------|------|------|------|-------|-----|
| MISAM name                      | Field name | Column name | Type | Ofs. | Len. | Disp. | Usf |
| VEVO                            | VEVOKOD    | VEVOKOD     | Str  | 0    | 2    | 2.0   | N   |
| VEVO                            | VEVONEV    | VEVONEV     | Str  | 2    | 10   | 10.0  | N   |
| TEREL                           | DATUM      | DATUM       | Date | 7    | 4    | 8.0   | N   |
| ESE                             | ESENEV     | ESENEV      | Str  | 3    | 10   | 10.0  | N   |
| VERE                            | RENDELES   | RENDELES    | dOub | 4    | 8    | 8.2   | N   |
|                                 |            |             |      |      |      |       | N   |

| Misam names |
|-------------|
| CIKK        |
| VEVO        |
| VERE        |
| ESE         |
| TEREL       |
| \$USER      |

8. ábra  
View összeállítás.

A felső ablakban a view-sor elemei láthatók. Az oszlopok jelentése a következő:

- MISAM name: Az adatot tartalmazó MISAM állomány neve. Származtatott adat esetén ez a név \$USER.
- Field name: Az adatot tartalmazó rekordmező neve. Származtatott adat esetén ezt a nevet a felhasználónak kell megadni.
- Column name: Alapértelmezésben ide bekerül az a név amit a Field name oszlopban adtunk meg, de megváltoztatható. Szükséges a változtatás, ha több MISAM állományban ugyanolyan nevű mező szerepel.



- Type: Az adat típusa.
- Ofs.: Az adatnak a MISAM rekordon belüli pozíciója. Származtatott adat esetén -1.
- Len.: Az adat view-soron belüli hossza.
- Disp.: Az adat megjelenítési szélessége. A pont után lehet, float vagy double típusú adatok esetén, a tizedes helyek számát megadni.
- Usf.: Van-e az adathoz felhasználói C függvény rendelve.

A mezőnév, a típus, a pozíció, a hossz, a szélesség és a decimális helyek száma a MISAM leírókból származik, kivéve a származtatott adatokat, ahol ezt külön meg kell adni. A típus megadását a 9. ábrán látható kiválasztó képernyő segíti.

|      |
|------|
| Type |
| Byte |
| Ubyt |
| Int  |
| uInt |
| Long |
| Date |
| Floa |
| dOub |
| Char |
| Str  |

9. ábra  
View-mező típusának megadása.

Az alsó ablakban a válsztható MISAM nevek vannak. Az ENTER billentyűvel kell kiválasztani egy állományt. Ekkor az alsó ablakban a kijelölt állomány rekordmezőinek a nevei tűnnek fel. Ezek közül választhatunk mezőket a view-sorba. A kiválasztott mező jellemzői bekerülnek a felső ablakba. ESC-el elhagyva a rekordmezők ablakát új MISAM állományt választhatunk. Kalkulált mezőt a \$USER név kijelölésével tudunk definiálni. Ekkor a program megkérdezi a mező és a hozzárendelt C modul nevét a következő ablakban:

User function name:

A MISAM állományok ablakát ESC-el elhagyva a felső ablakba jutunk, ahol a kiválasztott mezők jellemzőit módosíthatjuk. A kalkulált mezők típusát és hosszát mindenképpen meg kell adni. Az ablakban a kurzorvezérlő billentyűkkel tudunk mozogni. A DEL billentyűvel mezőt törölhetünk a view-sorból. Az INS billentyűvel az aktuális mező elé új mezőt szúrhatunk be. Az utolsó mező után a lefelé nyíllal illeszthetünk új mezőt.

Az F2 billentyűvel nyugtázhatjuk a detil sor összeállítást. Ha



ESC-el hagyjuk el ezt a funkciót, akkor a program nem mentett változtatás esetén megkérdezi, hogy nyugtázza-e a változást.

### 6.3.6. Munkakönyvtár megadása

#### Work directory

A futtatórendszer szükség esetén ideiglenes indexállományokat épít magának. Itt megadhatjuk, hogy melyik könyvtárba helyezze ezeket az állományokat. Erre a következő ablak szolgál:

Work directory:C:\TMP

Ha nem adunk meg semmit, akkor az ideiglenes állományok az aktuális könyvtárában fognak épülni.

### 6.3.7. Master kulcs összeállítás

#### master Key definition

Itt lehet megadni a view rendezettségét, ami tulajdonképpen egy kulcs összeállítása a masterállományra (ld. 10. ábra). Ugyan így lehet a link kapcsolatok definiálásánál a master kulcsértéket és a link kulcsot megadni.



| Index definition: CIKK |       |        |        |            |       |
|------------------------|-------|--------|--------|------------|-------|
| Field name             | Type  | Offset | Length | Asc./Desc. |       |
| CIKK_KOD               | Str   | 0      | 2      | Asc.       | Asc.  |
| _____                  | _____ | _____  | _____  | _____      | _____ |
| _____                  | _____ | _____  | _____  | _____      | _____ |
| _____                  | _____ | _____  | _____  | _____      | _____ |
| _____                  | _____ | _____  | _____  | _____      | _____ |

| Field names |
|-------------|
| CIKK_KOD    |
| CIKKNEV     |

10. ábra  
Master kulcs összeállító képernyő.

A felső ablakban az kulcsdarabok adatai láthatók:

- A Field name annak a neve a mezőnek a neve, amelyben a kulcsdarab található.
- A Type (típus) ugyanaz mint a MISAM leíróban.
- A Offset oszlop a kulcsdarab rekordon belüli kezdő pozíciója.
- A Length oszlopban a kulcsdarab hossza található.
- Az utolsó oszlopban azt adhatjuk meg, hogy a kulcsdarabra növekvő vagy csökkenő rendezés történjen.

Az alsó ablakban a rekordmezők nevei vannak. Közülük az ENTER billentyűvel tudunk választani. Az ESC billentyűvel jutunk a felső ablakba. Itt a kulcsdarabok adatait lehet módosítani. Az ablakban a kurzorvezérlő billentyűkkel tudunk mozogni. A DEL billentyűvel törölhetünk kulcsdarabot. Az INS billentyűvel az aktuális darab elé újat szúrhatunk be. Az utolsó mező után a lefelé nyíllal illeszthetünk új indexdarabot. Csak a karakter típusú kulcsdarabok adatai módosíthatók.

Az F2 billentyűvel nyugtázhatjuk a kulcs összeállítását. Ha ESC-el hagyjuk el ezt a funkciót, akkor a program nem mentett változtatás esetén megkérdezi, hogy nyugtázza-e a változást.

### 6.3.8. Intervallum megadása

First key value és lAst key value

A kezdő és végső kulcs megadásával tudunk intervallumot kijelölni



a view-ra. Mind a két esetben ugyanaz a képernyő jelenik meg:

| FIRST KEY         |                    |
|-------------------|--------------------|
| CIKK_KOD          | :                  |
| F2: Save and exit | F3: Drop key value |
| ESC: Exit         |                    |

11. ábra  
Intervallum megadás.

Ki kell tölteni a kívánt kulcsdarabokat és az F2 billentyűvel nyugtázzhatjuk az intervallum megadását. Ha ESC-el jövünk ki ebből a funkcióból, akkor a QRYEDIT program nem veszi figyelembe amit beírtunk. Az F3-as billentyűvel elhagyhatjuk a kulcs értéket. Adhatunk meg csak kezdő vagy csak végső kulcsot is.

### 6.3.9. Az intervallum kezelési mód

#### Handling of intervall

Itt lehet megadni a futtatórendszernek, hogy az intervallumot ALL vagy SELECTED módon kezelje:

| Interval |
|----------|
| All      |
| Selected |

12. ábra  
Intervallum kezelés módja.

Az alapértelmezés ALL.

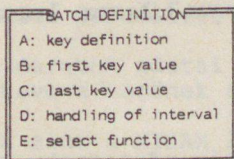


### 6.3.10. Lekérdezési mód megadása

#### Batch method definition

Itt írhatjuk elő, hogy a futtatórendszer batch módban üzemeljen. A 13. ábrán látható menüből választhatunk.

A kulcs összeállításakor kell az ideiglenes állomány rendezettségét megadni. Mivel az ideiglenes állomány minden rekordja egy view-sor lesz, az itt megadott kulcs a view rendezettségét határozza meg. A kulcs összeállítása hasonló a master kulcs összeállításához, csak itt a kulcsdarabokat a már definiált view-oszlopokból tudjuk kiválogatni: Az almenü többi része jelentésében és működésében azonos a főmenü megfelelő részeivel, csak az ideiglenes állományra vonatkoznak (ld. 13. ábra).



13. ábra  
Batch mód megadása.

### 6.3.11. Szelekciós függvény

#### select fUnction

Ha az intervallum kijelölésén kívül más szempontok alapján is akarunk szelektálni a lekérdezésbe bekerülő view-sorok között, akkor itt kell megadni a szelekciót végző C modul nevét a következő ablakban:

Select function:check\_view



| Index definition: cikk.QTX |       |        |        |            |
|----------------------------|-------|--------|--------|------------|
| Field name                 | Type  | Offset | Length | Asc./Desc. |
| _____                      | _____ | _____  | _____  | Asc.       |
| _____                      | _____ | _____  | _____  | _____      |
| _____                      | _____ | _____  | _____  | _____      |
| _____                      | _____ | _____  | _____  | _____      |
| _____                      | _____ | _____  | _____  | _____      |

| MISAM name | Field name |
|------------|------------|
| CIKK       | CIKK_KOD   |
| CIKK       | CIKKNEV    |
| VEVO       | VEVOKOD    |
| VEVO       | VEVONEV    |
| TEREL      | DATUM      |
| ESE        | ESENEV     |
| VERE       | RENDELES   |

14. ábra  
Batch módú lekérdezés rendezettsége.

### 6.3.12. Misam leírók könyvtára

#### directory of misam description

A queryleíróban teljes file-névvel szerepelnek a lekérdezésbe bevont MISAM állományok leírói. Ha itt nem adunk meg semmit, akkor a MISAM leírók nevében az a könyvtárnév fog szerepelni, amit a QRYEDIT hívásakor a -d opcióval megadtunk, mégpedig a gyökértől számítva. Itt ezt a könyvtárnevet módosíthatjuk a következő ablakban:

Directory of misam descriptions: \_filedef



### 6.3.13. A leírófile kiírása

save to file, save header file és save function Table

A Query Editor programnak három output file-ja van:

- a lekérdezési definíciót tartalmazó file,
- a view-oszlopok és MISAM állományok konstansait tartalmazó header file,
- a felhasználói C modulok függvénytábláját tartalmazó header file.

Mindegyik file kírása előtt lehetőség van a nevük módosítására.

A lekérdezési definícióra az ötödik fejezet végén láttunk példát. Itt header file-ra és a függvénytáblára mutatunk példát.

```

/*
 *      "PELDA.QH" generated by Query Editor (MI Software)
 *      on Aug/31/1989 15:16:12
 */

/*      MISAM files      */
#define QI_CIKK 0
#define QI_VEVO 1
#define QI_VERE 2
#define QI_ESE 3
#define QI_TEREL 4
/*      Detail fields      */
#define QF_CIKK_KOD 0
#define QF_CIKKNEV 1
#define QF_VEVO_KOD 2
#define QF_VEVOEV 3
#define QF_DATUM 4
#define QF_ESENEV 5
#define QF_RENDELES 6
#define QF_USER 7
/*      Detail row      */
struct Q_SPELDA
{
    char      cikk_kod[2];
    char      cikksnev[10];
    char      vevokod[2];
    char      vevonev[10];
    long      datum;
    char      esenev[10];
    double     rendeles;
    double     user;
};
/* end of PELDA.QH */

```



```

/*
 * "PELDA.QCT" generated by Query Editor (MI Software)
 * on Aug/31/1989 15:16:15
 */
extern int check_view();
extern int count_user();

#ifdef CFUNC_STRUCT
#define CFUNC_STRUCT
struct CFUNC {
    char *Cfname;
    int (*Cfp)();
};
#endif

static struct CFUNC PELDA_QCT[] = {
    {"check_view", check_view},
    {"count_user", count_user},
    {0, 0} };

/* end of PELDA.QCT */

```

### 6.3.14. Konzisztencia vizsgálat

#### check query description

A queryleíró erősen támaszkodik a lekérdezésbe bevont MISAM állományok leírójára. Ha a queryleíró létrehozása után megváltozik valamelyik rekordmező neve, akkor a Query Editor nem tudja a queryleírot beolvasni, mivel a hivatkozott mezőket a nevük alapján azonosítja. Ugyan ez a helyzet, ha a MISAM állományok nevét változtatjuk meg. Ebben az esetben kénytelenek vagyunk a queryleíróban szövegszerkesztővel kijavítani a megváltozott neveket.

Ha a MISAM leírókban más jellemzőket változtatunk meg (pl. típus, hossz), akkor a Query Editor beolvassa a leírot, de valószínű, hogy a view és az index leírásokon változtatni kell, hogy a queryleíró ismét összhangba kerüljön a MISAM leírókkal. A konzisztencia vizsgálat azokat az eltéréseket keresi meg ahol az esetleges változtatásokat el kell végezni. A vizsgálat olyan eltéréseket is kiszűr, amelyek nem feltétlenül igényelnek javítást. Ezek az üzenetek csak figyelmeztető jellegűek.







## 7. A MIQUEL könyvtár

A modulokat nem fontossági sorrendben tárgyaljuk. Célszerűbbnek tűnt a modulok funkció szerinti csoportosítása a következő kategóriák szerint:

- Előkészítő modulok.
- Aktív view állítás.
- Query információk.
- Query végrehajtás.
- A view mezőinek kezelése.
- A lekérdezés interaktív változtatása.
- Módosító modulok.

A MIQUEL könyvtár moduljait használó C programokhoz minden esetben két header file-t kell illeszteni:

- a MISAM file-kezelőhöz tartozó misam.h file-t,
- és a MIQUEL-hez tartozó miquel.h file-t.

A misam.h-nak meg kell előzni a miquel.h-t, mert az utóbbi hivatkozik az előzőre.

A miquel.h file tartalmazza a futtatórendszer által használt táblák definícióját, valamint néhány globális változót. A felhasználó számára a q\_errmsg globális változó a hibakezelés szempontjából lehet érdekes. A futtató rendszer bizonyos moduljai olyan hibakódokkal térhetnek vissza, amelyek valamilyen file-művelet sikertelenségére utalnak. Több file együttes lekérdezése esetén nem egyértelmű, hogy melyik file-lal kapcsolatos a hiba. Ilyenkor az érintett file neve a q\_errmsg változóban található.

A modulok visszatérési értékeire definiált konstansok is a miquel.h-ban találhatók.

A miquel.h file tartalma:

```
/*
 *      MIQUEL.H - Produced by MI Software 1990
 *
 *      Constant and type declaration
 */

/*
 *      Error codes
 */
#define Q_LKEY      1          /* Intervall upper bound */
#define Q_FKEY      2          /* Intervall lower bound */
#define Q_EOF       3          /* End of master file */
#define Q_BOF       4          /* Top of master file */
```



```

#define Q_LERR      5          /* Lock error          */
#define Q_ERROR    6          /* MISAM I/O error     */
#define ISBUILD     7          /* Index file creation error */
#define MEMORY     8          /* Not enough memory   */
#define OPEN       9          /* Open error          */
#define NO_MATCH   10         /* Missing key in link file */
#define NO_QUERY   11         /* There is no query to select */
#define MISSING_LINK 12        /* Invalid link name    */
#define MISSING_ISAM 13        /* Invalid isam name    */
#define TOO_MANY_SET 14        /* Too deep link structure */
#define BAD_FUNC   15         /* Bad user function name */
#define SET_END    16         /* End of set           */
#define CLOSE     17         /* Close error          */

```

```

/*
 *      Type definitions
 */

```

```

#ifndef CFUNC_STRUCT
#define CFUNC_STRUCT
struct CFUNC {
    char    *Cfname;
    int     (*Cfp)();
};

```

```

#endif

```

```

#ifndef ENGLISH
#define ENGLISH      0
#endif
#ifndef HUNGARIAN
#define HUNGARIAN    1
#endif
#ifndef GERMAN
#define GERMAN       2
#endif

```

```

typedef int (*XFP)();          /* pointer to a function */

```

```

/*
 *      Dynamic string array structure for internal use
 */

```

```

typedef struct SARR
{
    int    strn;          /* number of strings      */
    char   **strs;        /* string array           */
} *PSARR;

```



```

/*
 *      Structure for link connections
 */
struct SLNODE
{
    int      link_ind;      /* Index of linked MISAM      */
    int      m_key_partno;  /* Number of master key parts */
    KPARTS   *m_parts;      /* Master key parts           */
    char     *master_key;   /* Master key                 */
    int      master_keylen; /* Length of master key      */
    int      l_key_partno;  /* Number of link key parts   */
    KPARTS   *l_parts;      /* Link key parts             */
    char     *link_key;     /* Link key                   */
    int      link_keylen;   /* Length of link key        */
    char     match_flag;    /* Found link record          */
    char     is_set;        /* set or unique              */
    struct SLINK *links;    /* Sub links                  */
};

typedef struct SLNODE LNODE,*PLNODE;

struct SLINK
{
    int      master_ind;    /* Index of master MISAM      */
    int      link_no;       /* Number of linked MISAMs    */
    struct SLNODE *lnodes;  /* Link descriptions           */
};

typedef struct SLINK LINK,*PLINK;

/*
 *      Structure for detail row description
 */
struct SDÉDESC
{
    char     col_name[16];  /* Name of the column         */
    int      start;         /* Start position in det. row  */
    int      length;        /* Length in detail row       */
    int      dim;           /* From MISAM description     */
    char     typ;           /* Field type (MISAM like)    */
    int      width;         /* Field format width         */
    int      dec;           /* Field format dec. pos.     */
    int      ind;           /* Index of MISAM buffer      */
    int      offset;        /* Start pos. in MISAM buffer */
    char     *uname;        /* Name of user field         */
    char     *us_func;      /* User function name         */
};

typedef struct SDÉDESC DÉDESC,*PDÉDESC;

```



```

/*
 * Description of temporary file
 */
typedef struct S_TMPTAB
{
    int      nparts;           /* Number of key parts      */
    KPARTS   *iparts;          /* Key parts                 */
    char     *first_key;       /* First key                 */
    char     *last_key;        /* Last key                  */
    char     sel_mod;          /* Handling of interval     */
} TMPTAB, *PTMPTAB;

/*
 * QUERY descriptor
 */
struct SQRYDSC
{
    char     *descr;           /* Query description        */
    int      isam_no;          /* Number of selected MISAMs */
    FDESC    **q_fdsc;         /* Array of MISAM descriptors */
    char     **q_buffers;      /* Array of record buffers  */
    char     **alias;          /* Alias MISAM names        */
    char     iname[16];        /* Name of master index     */
    char     fullname[80];     /* Path                      */
    int      nparts;           /* Number of master key parts */
    KPARTS   *iparts;          /* Master key parts         */
    char     *first_key;       /* First key                 */
    char     *last_key;        /* Last key                  */
    char     sel_mod;          /* Handling of interval     */
    char     *act_key;          /* Actual master key value   */
    int      act_keylen;       /* Length of master key     */
    TMPTAB   *batch_mod;       /* Batch or interactive mode */
    LINK     *links;           /* Link structure            */
    char     *detailbuf;       /* View buffer               */
    int      fieldno;          /* Number of view columns    */
    DETDESC   *detdsc;         /* View description         */
    XFP       *func;           /* Function table            */
    int      set_no;           /* Number of set connections */
    char     *set_chain[5];     /* Array of set connections  */
    PSARR     tmp_ind;         /* Array of temporal files   */
};

typedef struct SQRYDSC QDSC, *PQDSC;

/* Array of opened queries */
struct SQRYTAB
{
    char     qryname[13];       /* Name of description file  */
    char     init_flag;         /* Is it initialized?        */
    PQDSC    qdsc;             /* Query descriptor          */
};

```



```
typedef struct SQRYTAB QRYTAB,*PQRYTAB;
```

```
/*
```

```
 *   For internal use only
```

```
*/
```

```
typedef struct S_TETEL
```

```
{
```

```
    char        name[17];
```

```
    int         offset;
```

```
    int         length;
```

```
    char        type;
```

```
} TETEL, *PTETEL;
```

```
/*
```

```
 *   Global variables
```

```
*/
```

```
extern int      q_errno;                /* Actual return value */
```

```
extern char     q_ermmsg[];            /* Message about error */
```

```
extern PQDSC    qdsc;                 /* Query descriptor */
```

```
extern PQRYTAB  qrytab;               /* Table of opened queries */
```

```
extern int      act_qdsc;              /* Serial number of active query*/
```

```
extern int      opened_qdsc;          /* Number of opened queries */
```

```
/* Lint definitions */
```

```
struct KEYPARTS *q_build_key(int *new_partno);
```

```
void      q_clear_all_ndx(void );
```

```
void      q_clear_first(void );
```

```
void      q_clear_last(void );
```

```
int        q_clear_ndx(int);
```

```
int        q_close(void );
```

```
int        q_det_size(void );
```

```
int        q_exec(int dir);
```

```
int        q_field_dec(int sernum);
```

```
void       q_field_get(int sernum,void *val);
```

```
int        q_field_is_valued(int sernum);
```

```
int        q_field_length(int fldnum);
```

```
char       *q_field_name(int sernum);
```

```
int        q_field_number(void);
```

```
int        q_field_offset(int sernum);
```

```
int        q_field_sernum(char *col_name);
```

```
void       q_field_set(int sernum,void *val);
```

```
int        q_field_size(int fldnum);
```

```
char       q_field_type(int sernum);
```

```
char       *q_get_act(int sernum);
```

```
char       *q_get_detail(void );
```

```
struct FILEDESC *q_get_fdf(int sernum);
```

```
void       q_get_first(char *key);
```

```
int        q_get_keylen(int partno,struct KEYPARTS *parts);
```

```
void       q_get_last(char *key);
```

```
char       *q_get_misam(int sernum);
```



```

int      q_init(void );
void     q_int_mod(char mod);
int      q_isam_number(void);
int      q_is_int(void );
int      q_is_match(int sernum);
int      q_link_init(int sernum, char * key);
char     *q_link_name(int sernum);
int      q_link_sernum(char *lname);
int      q_next_link(int sernum,int dir);
int      q_one_file(char *fn,char *worklib);
int      q_open(char *fn,struct CFUNC *func);
int      q_query_sernum(char *name);
int      q_scan_key(int actisam,char *key,char *head,int fldno);
int      q_select(int sernum);
int      q_select_key(int ind);
void     q_set_first(char *key);
void     q_set_last(char *key);
int      q_set_new_key(struct KEYPARTS *iparts,int partno);
int      q_upd_det(char *newdet);
void     q_upd_rec(char *newbuf,char *detail,int isamno);
void     q_work_lib(char *worklib);
/* end of MIQUEL.H */

```

## 7.1. Előkészítő modulok

### 7.1.1. q\_open

**Funkció** Megnyitja a queryleírót.

**Használat**

```

#include <MISAM.H>
#include <MIQUEL.H>
      int q_open(fname,ftab)
      char *fname;  queryleíró neve
      CFUNC *ftab;  függvénytábla

```

**Leírás**

A `q_open()` függvény az `fname` által azonosított leírófile-t beolvassa. Helyet allokal a query leírotáblájának és feltölti azt a leírófile alapján. Beolvassa a megadott MISAM leíró állományokat a MISAM `isgetfdf()` függvényével. Mivel a táblába itt kerülnek be az összes többi függvény számára szükséges információk, ennek a hívása meg kell, hogy előzzön minden más MIQUEL függvényhívást. Ha felhasználói függvényekkel akarunk a view sorain manipulálni, akkor a felhasználói függvénytábla címét az `ftab` paraméterben kell átadni. Ha nem használunk függvényeket, akkor itt egy "null pointer"-t kell átadni.



Visszatérés Sikeres művelet esetén visszaadja a megnyitott query sorszámát. Egyébként a felhasználói program hibaüzenettel abortál, mivel sikertelen querynyitás után nincs értelme a program további futásának.

### 7.1.2. q\_one\_file

Funkció query nyitása leírófile nélkül.

Használat

```
#include <MISAM.H>
#include <MIQUEL.H>
    int q_one_file(fname,tmplib)
        char *fname; a MISAM leírófile
        char *tmplib; munkakönyvtár neve
```

Leírás Ha csak egy MISAM állományt akarunk lekérdezni, és nincs szükségünk link kapcsolatok kezelésére, akkor leírófile nélkül is megnyithatjuk a queryt a q\_one\_file() függvénnyel. Az fname a lekérdezendő MISAM állomány leírójának a neve. A keresési szempontot a MISAM leíró első indexe adja meg. A view-sorba az állomány egy rekordja kerül. Ha a függvény hívása után megváltoztatjuk a keresési szempontot a q\_set\_new\_key() függvénnyel, akkor szükség lehet egy ideiglenes indexállomány létrehozására. Ez az állomány a tmplib paraméterrel megadott könyvtárba fog képződni.

Visszatérés Sikeres művelet esetén visszaadja a megnyitott query sorszámát. Egyébként a felhasználói program hibaüzenettel abortál, mivel sikertelen querynyitás után nincs értelme a program további futásának.

### 7.1.3. q\_set\_new\_key

Funkció Új keresési szempont megadása a masterfile-ra.

Használat

```
#include <MISAM.H>
#include <MIQUEL.H>
    int q_set_new_key(kdsc,partno)
        KPARTS kdsc; az új kulcs leírója
        int partno; kulcsdarabok száma
```

Leírás A q\_set\_new\_key() függvénnyel lehet a view sorainak a rendezettségét változtatni. A paraméterek



a masterállományra vonatkozó kulcsot írnak le. A függvény ezzel a kulccsal felülírja a leírófile-ban megadottat, és a további keresést már eszerint a kulcs szerint végzi. Ha az új kulcshoz nem tartozik indexállomány, akkor a függvény megkísérli annak felépítését a `q_init()`-nél leírtak szerint.

A függvény hívása után az előzőleg beállított intervallumhatárok érvényüket veszítik, ezért ezeket a `q_set_first()` és a `q_set_last()` függvényekkel újra meg kell adni, ha szükséges. A függvény hatására a masterfile aktuális pozíciója a masterállomány eleje lesz (nem az intervallum eleje).

Visszatérés      SUCCESS Sikeres művelet.  
                   ISBUILD Sikertelen ideiglenes indexfile építés.

#### 7.1.4. `q_set_first`

Funkció            Beállítja a keresés intervallumának alsó határát.

Használat        `#include <MISAM.H>`  
                   `#include <MIQUEL.H>`  
                   `void q_set_first(key)`  
                   `char *key;`      a beállítandó érték

Leírás            A `key` változóba a masterfile aktuális indexének megfelelő értéknek kell lennie. A függvény ezt az értéket helyezi el a querytábla `first_key` elemébe. Ha a leírófile-ban adtunk meg kezdő kulcsot, akkor ez felülbíráلódik. Ha a `q_exec()` függvény által beolvasott master rekord kulcsa kisebb lesz mint az itt megadott érték, akkor a `Q_FKEY` értéket adja vissza. Ha a `q_set_first()` függvényt a `q_init()` függvény előtt hívjuk meg, akkor a `q_init()` erre a kulcsra fog ráállni. Ha a `q_init()` után állítottuk a kezdő kulcsot, akkor a `q_init()` ismételt hívásával tudunk az első rekordra állni. Ilyenkor a `q_init()` már nem allokál területeket csak a pozícionálást hajtja végre.

Visszatérés      Nincs.



### 7.1.5. q\_set\_last

Funkció Beállítja a keresés intervallumának felső határát.

Használat 

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_set_last(key)
char *key;    a beállítandó érték
```

Leírás A key változóban a masterfile aktuális indexének megfelelő értéknek kell lennie. A függvény ezt az értéket helyezi el a querytábla last\_key elemébe. Ha a leírófile-ban adtunk meg kezdő kulcsot, akkor ez felülbíráلódik. Ha a q\_exec() függvény által beolvasott master rekord kulcsa nagyobb lesz mint az itt megadott érték, akkor a Q\_LKEY értéket adja vissza.

Visszatérés Nincs.

### 7.1.6. q\_clear\_first

Funkció Eltörli az intervallum alsó határát.

Használat 

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_clear_first()
```

Leírás A rutin eltörli az intervallum alsó határát. Hívása után a q\_init() függvény a view első sorára fog pozícionálni.

Visszatérés Nincs.

### 7.1.7. q\_clear\_last

Funkció Eltörli az intervallum felső határát.

Használat 

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_clear_last()
```

Leírás A rutin eltörli az intervallum felső határát.

Visszatérés Nincs.



7.1.8. `q_work_lib`

Funkció Beállítja a ideiglenes indexfile-ok könyvtárát.

Használat

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_work_lib(wrklib)
char *wrklib; a munkakönyvtár neve
```

Leírás Ha a quereleíró alapján szükséges, akkor a futatórendszer ideiglenes indexfile-okat épít a leírófile-ban megadott munkakönyvtárban. Ezzel a függvénnyel a `q_init()` rutin hívása előtt felül lehet bírálni a leírófile-ban megadott munkakönyvtárt.

Visszatérés Nincs.

7.1.9. `q_int_mod`

Funkció Megváltoztatja az intervallum kezelésének módját.

Használat

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_int_mod(mod)
char mod; 'A' vagy 'S'
```

Leírás Az intervallum kezelését lehet SELECTED vagy ALL módra állítani S vagy A paraméterekkel.

Visszatérés Nincs.

7.1.10. `q_select_key`

Funkció Új keresési szempont megadása a masterfile-ra.

Használat

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_select_key(sernum)
int sernum; a kulcs sorszáma
```

Leírás A `q_set_new_key()` függvényhez hasonlóan ezzel a rutinnal is a view sorainak rendezettségét lehet állítani. Használata akkor célszerű, ha a masterállományt leíró MISAM leírófile-ban szereplő kulcsok valamelyikét akarjuk használni. A sernum



a kívánt kulcs sorszáma a MISAM leíróállományban.  
A rutin hatása egyébként megegyezik a `q_set_key()` függvényével.

Visszatérés      SUCCESS Sikeres művelet  
                  ISBUILD Sikertelen az ideiglenes indexfile építése.

### 7.1.11. `q_clear_ndx`

Funkció           A megnyitott indexállományok számának csökkentése.

Használat        

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_clear_ndx(sernum)
int     sernum; az állomány sorszáma
```

Leírás           A futtató rendszer a lekérdezésbe bevont állományokhoz tartozó indexállományt megnyitja. Így az egyszerre nyitva tartott állományok száma igen nagy lehet, pedig a linkelt állományok csak egy indexfile alapján kapcsolódnak a lekérdezéshez. Ezzel a rutinnal kérhetjük, hogy a futtató rendszer a `sernum`-mal jelzett állományhoz csak a lekérdezéshez szükséges indexfile-t nyissa meg.

Visszatérés      SUCCESS Sikeres művelet.  
                  Egyébként Nem létező állomány.

### 7.1.12. `q_clear_all_ndx`

Funkció           A megnyitott indexállományok számának csökkentése.

Használat        

```
#include <MISAM.H>
#include <MIQUEL.H>
void q_clear_all_ndx(sernum)
```

Leírás           A futtató rendszer a lekérdezésbe bevont állományokhoz tartozó indexállományt megnyitja. Így az egyszerre nyitva tartott állományok száma igen nagy lehet, pedig a linkelt állományok csak egy indexfile alapján kapcsolódnak a lekérdezéshez. Ezzel a rutinnal kérhetjük, hogy a futtató rendszer a linkelt állományokhoz csak a lekérdezéshez szükséges indexfile-t nyissa meg.



Visszatérés        nincs

### 7.1.13. `q_init`

**Funkció**            Inicializálja a query tábláját és feltölti a view első sorát.

**Használat**        `#include <MISAM.H>`  
                      `#include <MIQUEL.H>`  
                      `int q_init()`

**Leírás**            A `q_init()` függvény több funkciót lát el:

- Memóriaterületet allokál a querytábla elemeinek.
- Megnyitja a MISAM állományokat.
- Ideiglenes indexállományokat épít, ha szükséges.
- Feltölti az első view-sort.

Mivel a függvény maga nyitja meg a queryleíróban szereplő MISAM állományokat, ezekre az állományokra nem kell külön meghívni az `isopen()` függvényt. Ha a felhasználói programban szükség van valamelyik MISAM állomány leírotáblájára, akkor ezt a `q_get_fdf()` függvény segítségével tudjuk elérni.

Ammenyiben a view feltöltéséhez szükséges indexállományok valamelyike nem létezik, a `q_init()` függvény ideiglenes indexállományt épít, amelyet a `q_close()` függvény eltöröl. Az ideiglenes állományok a queryleíróban megadott munkakönyvtárban, ennek hiányában az aktuális könyvtárban képződnek.

A függvény hívása után, a master kulcs és az intervallum alapján a view-tábla az első sorra áll rá, a bufferek pedig az ehhez tartozó rekordokat fogják tartalmazni.

**Visszatérés**        **SUCCESS** Sikeres művelet.  
                      **OPEN**    File-nyitási hiba, a file neve a globális `q_errmsg`-ben.  
                      **ISBUILD** Sikertelen az ideiglenes indexfile építése.  
                      **Q\_LERR**    Nem hozzáférhető állomány, az állomány neve a globális `q_errmsg`-ben.



Q\_LKEY A master rekord kulcsa nagyobb mint a végső kulcs.  
 Q\_EOF Az állomány végére értünk.  
 NO\_MATCH Sikertelen link illesztés, nem talált a link definíciónak megfelelő rekordot az állományban, az állomány neve a globális q\_errmsg-ben.  
 Q\_ERROR File-olvasási hiba, az állomány neve a globális q\_errmsg-ben.

## 7.2. Aktív view állítás

### 7.2.1. q\_select

**Funkció** Megváltoztatja az aktív view-t.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_select(sernum)
    int *sernum; a query sorszáma
```

**Leírás**

A futtatórendszer egyszerre több view-t is tud kezelni. A q\_open() vagy a q\_one\_file() modulok valamelyikével legutoljára megnyitott view lesz az aktuális view. A q\_select() modullal a már megnyitott view-k közül bármelyiket aktívvá tehetjük. A sernum az a sorszám, amit a q\_open() vagy a q\_one\_file() rutinok adtak vissza. A sorszámot megkaphatjuk a leírófile neve alapján, a q\_query\_sernum() függvénnyel is. A megnyitott view-k nevei a globális qrytab struktúrában találhatók. Ha egyszerre több view-t akarunk nyitva tartani, akkor könnyen előfordulhat, hogy több file-t nyitunk meg mint amit a DOS megenged.

**Visszatérés**

SUCCESS Sikeres művelet.  
 NO\_QUERY A megadott nevű query nincs megnyitva.  
 Nem történt szelektálás.



### 7.2.2. q\_close

|             |                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | Lezárja az aktív queryt.                                                                                                                                                                         |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_close()</pre>                                                                                                                      |
| Leírás      | Felszabadítja a q_open() és a q_init() függvények által allokált területeket, valamint lezárja a q_init()-ben megnyitott állományokat. Ha voltak ideiglenes indexállományok, akkor ezeket törli. |
| Visszatérés | SUCCESS Sikeres művelet.<br>OPEN File-zárási hiba, a file neve a globális q_errmsg-ben.                                                                                                          |

### 7.3. Query információk

#### 7.3.1. q\_get\_detail

|             |                                                                                    |
|-------------|------------------------------------------------------------------------------------|
| Funkció     | Visszaadja a view-tábla címét.                                                     |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; char *q_get_detail()</pre> |
| Leírás      | Visszaadja azt a címet, ahova a q_exec() függvény view aktuális sorát tölti.       |
| Visszatérés | Az aktuális sor címe.                                                              |

#### 7.3.2. q\_get\_act

|           |                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció   | Visszaadja egy adatállomány aktuális kulcsát.                                                                                                     |
| Használat | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; char *q_get_act(sernum) int sernum; az adatállomány sorszáma</pre>                        |
| Leírás    | A sorszámmal azonosított adatállomány aktuális rekordjából képzett aktuális kulcsot adja vissza. Az állomány sorszámát a QRYEDIT programmal gene- |



rált konstansokból, vagy pedig a `q_link_sernum()` függvénnyel kaphatjuk meg.

Visszatérés      A kulcs címe.

### 7.3.3. `q_get_misam`

Funkció            Visszaadja egy adatállomány aktuális rekordját.

Használat        `#include <MISAM.H>`  
                 `#include <MIQUEL.H>`  
                 `char *q_get_misam(sernum)`  
                 `int     sernum;` az adatállomány sorszáma

Leírás            A névvel azonosított adatállomány aktuális rekordját adja vissza. Az állomány sorszáma a QRYEDIT programmal generált konstansokból, vagy pedig a `q_link_sernum()` függvénnyel kaphatjuk meg.

Visszatérés      A rekord buffer címe.

### 7.3.4. `q_get_fdf`

Funkció            Visszaad egy MISAM leírotáblát.

Használat        `#include <MISAM.H>`  
                 `#include <MIQUEL.H>`  
                 `FDESC *q_get_fdf(sernum)`  
                 `int     sernum;` az adatállomány sorszáma

Leírás            A sorszám alapján azonosított MISAM leíróstruktúra címét adja vissza. A struktúrát a `q_open()` függvény tölti fel a MISAM `isgetfdf()` függvényével. Az állomány sorszáma a QRYEDIT programmal generált konstansokból, vagy pedig a `q_link_sernum()` függvénnyel kaphatjuk meg.

Visszatérés      A MISAM leíró címe.

### 7.3.5. `q_det_size`

Funkció            Visszaadja a view-sor hosszát.

Használat        `#include <MISAM.H>`



```
#include <MIQUEL.H>
int q_det_size()
```

**Leírás** Gyakran kell valahová átmásolni az aktuális view-sort. Mivel a view-sor nem C stringként viselkedik (nincs lezárva null karakterrel), ezért csak memória másolással tudjuk ezt megoldani. Ehhez kell a view-sor hosszát ismerni.

**Visszatérés** A view-sor címe.

### 7.3.6. q\_is\_match

**Funkció** Megadja, hogy az adott linkelt állományon talált-e a rendszer megfelelő rekordot.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_is_match(sernum)
    int sernum; a állomány sorszáma
```

**Leírás** A lekérdezést végrehajtó modulok (q\_exec(), q\_next\_link() és a q\_init()) visszatérési értéke lehet NO\_MATCH. Ez azt jelenti, hogy valamelyik linkelt állományban a rendszer nem talált a master kulcsértéknek megfelelő rekordot. A q\_is\_match() rutinnal megvizsgálhatjuk, hogy az adott link kapcsolatban volt-e megfelelő rekord. Az állomány sorszámát a QRYEDIT programmal generált konstansokból, vagy a q\_link\_sernum() függvénnyel kaphatjuk meg.

**Visszatérés**

- 1 Van megfelelő rekord.
- 0 Nincs megfelelő rekord.

### 7.3.7. q\_get\_first

**Funkció** Adja a beállított intervallum alsó határát.



Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `void q_get_first_key(key)`  
                  `char    *key;        a kulcsérték helye`

Leírás            A kapott paraméter egy allokált terület mutatója. Erre a területre tölti a modul az intervallum alsó határát jelölő kulcsértéket. Ha nincs intervallum beállítva az átadott memóriaterületet feltölti null karakterekkel.

Visszatérés      Nincs.

### 7.3.8. `q_get_last`

Funkció            Átadja a beállított intervallum felső határát.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `void q_get_last(key)`  
                  `char    *key;        a kulcsérték helye`

Leírás            A kapott paraméter egy allokált terület mutatója. Erre a területre tölti a modul az intervallum felső határát jelölő kulcsértéket. Ha nincs intervallum beállítva az átadott memóriaterületet feltölti null karakterekkel.

Visszatérés      Nincs.

### 7.3.9. `q_link_sernum`

Funkció            Visszaadja az állomány sorszámát.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `int q_link_sernum(lname)`  
                  `char    *lname;    Az adatállomány neve`

Leírás            Az adatállományokat a link struktúrában egy sorszám azonosítja. A `q_link_sernum()` ezt a sorszámot adja vissza az adatállomány neve alapján.

Visszatérés      Az állomány sorszáma.  
                  -1 Ha nincs ilyen nevű állomány.



### 7.3.10. q\_link\_name

|             |                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | Sorszám alapján visszaadja egy linkelt állomány nevét.                                                                        |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; char *q_link_name(sernum) int *sernum; a linkelt állomány száma</pre> |
| Leírás      | A QRYEDIT programmal létre hozott header file-ban található sorszám alapján visszaadja a linkelt állomány nevét.              |
| Visszatérés | Az állomány neve.<br>0 Ha a megadott sorszámhoz nem tartozik állomány.                                                        |

### 7.3.11. q\_query\_sernum

|             |                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | A leírófile alapján visszaadja egy megnyitott query sorszámát.                                                                                   |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_query_sernum(qname) char *qname; a leírófile neve</pre>                            |
| Leírás      | A qname, vagy egy queryleíró neve (.QTX), vagy egy MISAM leíró neve (.FDF). A megnyitott queryk nevei a globális grytab struktúrában találhatók. |
| Visszatérés | A query sorszáma.<br>-1 Ha nincs megnyitott queryt.                                                                                              |

### 7.3.12. q\_isam\_number

|             |                                                                                       |
|-------------|---------------------------------------------------------------------------------------|
| Funkció     | Visszaadja az állományok számát.                                                      |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_isam_number(void)</pre> |
| Leírás      | Visszaadja a querybe bevont állományok számát.                                        |
| Visszatérés | A MISAM állományok száma.                                                             |



## 7.4. Query végrehajtás

### 7.4.1. q\_exec

**Funkció** Beltölti a következő view-sort.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_exec(dir)
    int dir;          a keresés iránya
```

**Leírás** A dir által megadott irányban veszi a masterfile következő rekordját. A link definíciók alapján beolvassa a hozzá tartozó rekordokat, és ezekből képez egy view-sort. A dir változó lehetséges értékei, a MISAM.H-ban az isread() függvényénél megadott konstansok lehetnek. Ha a dir változó értéke CURRENT, akkor a rutin az aktuális master rekord alapján feltölti a linkelt állományokat. Ha SET kapcsolatokat jelöltünk ki a linkelt állományokon, akkor így tudunk a SET-ek elejére állni. Hívása után a q\_get misam() és a q\_get\_act() függvényekkel elérhetjük a querybe bevont állományok aktuális rekordjait és aktuális kulcsait.

Ha a link kapcsolatok leírásánál a KEY kulcszónak SET értéket adtunk, akkor a q\_exec() előbb mindig ezeken az állományokon próbál azonos kulcsú rekordot olvasni. A próbálkozás mindig a legelső SET jellegű link kapcsolatnál kezdődik, és ha ott nem volt sikeres, akkor lép egy szintel feljebb.

**Visszatérés**

```
SUCCESS Sikeres művelet.
Q_LERR Nem hozzáférhető állomány, az állomány neve a globális q_errmsg-ben.
Q_FKEY A master rekord kulcsa kisebb mint a végső kulcs.
Q_LKEY A master rekord kulcsa nagyobb mint a végső kulcs.
Q_EOF Az állomány végére értünk.
NO_MATCH Sikertelen link illesztés, nem talált a link definíciónak megfelelő rekordot az állományban, az állomány neve a globális q_errmsg-ben.
Q_ERROR File-olvasási hiba, az állomány neve a globális q_errmsg-ben.
```



### 7.4.2. q\_next\_link

**Funkció** Alsóbb szintű állomány továbbléptetése azonos kulccsal.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_next_link(sernum,dir)
    int    sernum; a csatolt állomány száma
    int    dir;    a keresés iránya
```

**Leírás** Ha a link definíciókban megadott kulcshoz a csatolt állományban több rekord tartozik, és nem adtuk meg, hogy SET kapcsolatról van szó, akkor a q\_exec() az azonos kulcsú rekordok közül az első hozza be. A q\_next\_link() rutinnal ezeket a rekordokat programozottan dolgozhatjuk fel. A függvény a masterállományból nem vesz új rekordot, hanem a sernummal azonosított és az ahhoz csatolt állományokat kezeli. A q\_next\_link() működése addig sikeres, amíg azonos kulcsú rekordokat talál a jelzett állományban. Az állomány sorszámát a QRYEDIT programmal generált konstansokból, vagy pedig a q\_link\_sernum() függvénnyel kaphatjuk meg.

**Visszatérés**

SUCCESS Sikeres művelet.

Q\_LERR Nem hozzáférhető állomány, az állomány neve a globális q\_errmsg-ben.

NO\_MATCH Sikertelen link illesztés, nem talált a link definíciónak megfelelő rekordot az állományban, az állomány neve a globális q\_errmsg-ben.

Q\_ERROR File-olvasási hiba, az állomány neve a globális q\_errmsg-ben.

### 7.4.3. q\_link\_init

**Funkció** Pozicionálás valamelyik linkelt állományon.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
int q_link_init(sernum,key)
    int    sernum; a csatolt állomány száma
    char    *key;  a keresés kulcsa
```

**Leírás** A q\_init rutinnal tudunk a masterállományon pozicionálni. Ilyenkor a linkelt állományok pozí-



cióját a master rekordból képzett kulcsértékek határozzák meg. Ha valamelyik linkelt állományon ettől eltérő rekordra akarunk pozícionálni, akkor a `q_link_init` rutint használjuk. A `sernum` az állomány sorszáma a key pedig a kívánt kulcsérték. Újabb `q_exec`, vagy `q_init` hívás után a linkelt állomány pozícióját megint a master rekord határozza meg.

|             |                                                                                                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Visszatérés | SUCCESS Sikeres művelet.                                                                                                                                      |
|             | Q_LERR Nem hozzáférhető állomány, az állomány neve a globális <code>q_errmsg</code> -ben.                                                                     |
|             | NO_MATCH Sikertelen link illesztés, nem talált a link definíciónak megfelelő rekordot az állományban, az állomány neve a globális <code>q_errmsg</code> -ben. |
|             | Q_ERROR File-olvasási hiba, az állomány neve a globális <code>q_errmsg</code> -ben.                                                                           |

## 7.5. A view mezőinek kezelése

### 7.5.1. `q_field_name`

|             |                                                                                                                                   |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | Visszaadja a view-sor egy mezőjének a nevét.                                                                                      |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; char *q_field_name(sernum)     int sernum;          a mező sorszáma</pre> |
| Leírás      | A mező sorszáma alapján visszaadja a view-sor nevét.                                                                              |
| Visszatérés | A mező neve.                                                                                                                      |

### 7.5.2. `q_field_sernum`

|           |                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| Funkció   | Visszaadja a view-sor egy mezőjének a sorszámát.                                                                            |
| Használat | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_field_sernum(name)     char *name;          a mező neve</pre> |
| Leírás    | A mező neve alapján visszaadja a view-sor adott mezőjének a sorszámát.                                                      |



Visszatérés      Mező sorszáma.  
                      -1 egyébként.

### 7.5.3. `q_field_offset`

Funkció            Visszaadja a view-sor egy mezőjének a bufferen belüli címét.

Használat        `#include <MISAM.H>`  
                      `#include <MIQUEL.H>`  
                      `int q_field_offset(sernum)`  
                          `int        sernum;`            a mező sorszáma

Leírás            A mező sorszáma alapján visszaadja a mezőnek a futtató rendszer bufferének kezdetétől számított kezdőpozícióját.

Visszatérés      A belső bufferbeli relatív pozíció.

### 7.5.4. `q_field_size`

Funkció            Visszaadja a view-sor egy mezőjének a méretét.

Használat        `#include <MISAM.H>`  
                      `#include <MIQUEL.H>`  
                      `int q_field_size(sernum)`  
                          `int        sernum;`            a mező sorszáma

Leírás            A mező sorszáma alapján visszaadja a view-sor adott mezőjének a méretét.

Visszatérés      A mező mérete.

### 7.5.5. `q_field_length`

Funkció            Visszaadja a view-sor egy mezőjének a szélességét.

Használat        `#include <MISAM.H>`  
                      `#include <MIQUEL.H>`  
                      `int q_field_length(sernum)`  
                          `int        sernum;`            a mező sorszáma

Leírás            A mező sorszáma alapján visszaadja a view-sor adott mezőjének a kiírási szélességét.



Visszatérés      A mező szélessége.

### 7.5.6. q\_field\_type

Funkció            Visszaadja a view-sor egy mezőjének a típusát.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `char q_field_type(sernum)`  
                  `int     sernum;`            a mező sorszáma

Leírás            A mező sorszáma alapján visszaadja a mező típusát a MISAM-ban megszokott formában.

Visszatérés      A mező típusa.

### 7.5.7. q\_field\_dec

Funkció            Decimális helyek számát adja vissza.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `int q_field_dec(sernum)`  
                  `int     sernum;`            a mező sorszáma

Leírás            Float és double típusú mezőknél visszaadja a decimális helyek számát.

Visszatérés      A decimális helyek száma.

### 7.5.8. q\_field\_set

Funkció            Értékkadás a view-sor egy mezőjének.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `void q_field_set(sernum,addr)`  
                  `int     sernum;`            a mező sorszáma  
                  `void *addr;`            az új érték

Leírás            A sorszámmal azonosított view-mezőbe beteszi az addr paraméterben kapott értéket.

Visszatérés      Nincs.



### 7.5.9. q\_field\_get

|             |                                                                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | View-sor egy mezőjének beolvasása.                                                                                                                                             |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; void q_field_get(sernum,addr)     int    sernum;      a mező sorszáma     void *addr;         a beolvasott érték</pre> |
| Leírás      | A sorszámmal azonosított view-mező értékét az addr paraméter által mutatott helyre másolja.                                                                                    |
| Visszatérés | Nincs.                                                                                                                                                                         |

### 7.5.10. q\_field\_is\_valued

|             |                                                                                                                                                                      |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | Visszaadja, hogy a mező kapott-e értéket.                                                                                                                            |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_field_is_valued(sernum)     int    sernum;      a mező sorszáma</pre>                                  |
| Leírás      | Visszaadja, hogy a sorszámmal azonosított mezőnek tudott-e a q_exec rutin a legutóbbi híváskor értéket adni. A függvénynek \$USER típusú mezők esetén nincs értelme. |
| Visszatérés | 1 A mező kapott értéket.<br>0 Egyébként.                                                                                                                             |

### 7.5.11. q\_field\_number

|             |                                                                                        |
|-------------|----------------------------------------------------------------------------------------|
| Funkció     | Visszaadja a view oszlopainak számát.                                                  |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_field_number(void)</pre> |
| Leírás      | Visszaadja a view oszlopainak számát                                                   |
| Visszatérés | A view oszlopainak száma.                                                              |



## 7.6. Interaktív lekérdezés

A következő két rutin a lekérdező programok számára biztosít interaktív szolgáltatásokat.

Ezeknek a moduloknak a használata esetén a C keretprogramhoz hozzá kell szerkeszteni a CEX, a CEX2 és az MFORM tárgykönyvtárakat.

A rutinok által generált képernyő lehet magyar vagy angol nyelvű. Ezt a felhasználói programban definiált, integer típusú, globális MI\_language változóval tudjuk beállítani.

```
#include <miquel.h>
.
.
.
int      MI_language = HUNGARIAN;
```

A konstansok a miquel.h-ban vannak definiálva.

A modulok használata még egy globális változó definiálását teszi szükségessé. A generált képernyő színei számára kell egy ötelemű glcolors nevű karaktertömböt definiálni.

```
char      glcolors[] =
{
    COLOR(CYAN,BLUE),      /* alap szín */
    COLOR(YELLOW,BLUE),    /* kiemelt szín */
    COLOR(YELLOW,BLACK),   /* edit szín */
    COLOR(GRAY,RED),        /* üzenet szín */
    COLOR(BLACK,GREEN)     /* help szín */
};
```

A színek a CEX könyvtárhoz tartozó setattr.h header file-ban vannak definiálva.

### 7.6.1. q\_build\_key

**Funkció** Interaktív módon összeállítja egy kulcs leírását a masterállományra.

**Használat**

```
#include <MISAM.H>
#include <MIQUEL.H>
      KPARTS *q_build_key(partno)
      int      *partno;  a kulcsdarabok száma
```

**Leírás** A függvény segítségével a master rekordmezőiből



egy kulcsot tudunk interaktív módon összeállítani. A kulcs megadása Qery EDitor kulcs definiációs funkciójához hasonló (10. ábra). A definiált kulcsdarabok száma a partno változóba kerül. Az így definiált kulcsra a `q_set_new_key()` függvénnyel tudjuk a queryt állítani.

Visszatérés      0    Nem definiáltunk kulcsot.  
Egyébként a kulcs címe.

### 7.6.2. `q_scan_key`

Funkció            Egy MISAM állomány aktuális kucsának kitöltése form képernyőn.

Használat        `#include <MISAM.H>`  
                  `#include <MIQUEL.H>`  
                  `int q_scan_key(isamno, key, head, partno)`  
                  `int    isamno;    a MISAM sorszáma`  
                  `char   *key;       a beolvasott kulcsérték`  
                  `char   *head;      a képernyő fejszövege`  
                  `int    partno;     a kulcsdarabok száma`

Leírás            Az isamno a queryleíróban megadott sorrend alapján azonosítja a MISAM állományt ( 0 a masterfile). Erre a kulcsra nyit egy formot és a kitöltött értéket a key változóban adja vissza. A head-ban megadott szöveg a képernyő tetején meg fog jelenni. A partno-val megadott sorszámu kulcsdarabra fog a kurzor állni. A képernyő formája a 10. ábrán látható.

Visszatérés      1    Adtunk meg kulcsot.  
                  0    Nem adtunk meg kulcsot.  
                  -1    Eltöröltük a kulcsot.

## 7.7. Módosító modulok

### 7.7.1. `q_upd_det`

Funkció            A módosított view-sor alapján aktualizálja a futatórendszer buffereit.



|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; int q_upd_det(buffer) char *buffer; a módosított view-sor</pre>                                                                                                                                                                                                                                                                                                                                                                                                         |
| Leírás      | Ha a programból módosítottuk az aktuális view-sor tartalmát, akkor szükség lehet a megadott file-kapcsolatok alapján a futtatórendszer buffereinek az aktualizálására. A rutin hívásával tudjuk azt is elérni, hogy a következő view-sort a módosított állapotnak megfelelően kapjuk.                                                                                                                                                                                                                                           |
| Visszatérés | <p>SUCCESS Sikeres művelet.</p> <p>Q_LERR Nem hozzáférhető állomány, az állomány neve a globális q_errmsg-ben.</p> <p>Q_FKEY A master rekord kulcsa kisebb mint a végső kulcs.</p> <p>Q_LKEY A master rekord kulcsa nagyobb mint a végső kulcs.</p> <p>Q_EOF Az állomány végére értünk.</p> <p>NÖ_MATCH Sikertelen link illesztés, nem talált a link definíciónak megfelelő rekordot az állományban, az állomány neve a globális q_errmsg-ben.</p> <p>Q_ERROR File-olvasási hiba, az állomány neve a globális q_errmsg-ben.</p> |

### 7.7.2. q\_upd\_rec

|             |                                                                                                                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Funkció     | A módosított view-sor alapján előállít egy MISAM buffert.                                                                                                                                                                     |
| Használat   | <pre>#include &lt;MISAM.H&gt; #include &lt;MIQUEL.H&gt; void q_upd_rec(buffer,detbuf,isamno) char *buffer; a MISAM buffer char *detbuf; a módosított view-sor int isamno; a MISAM sorszáma</pre>                              |
| Leírás      | Ha a programból módosítottuk az aktuális view-sor tartalmát, a rutin előállít a view-sor alapján egy MISAM buffert, amellyel a MISAM állomány módosítható. A módosított MISAM buffert a buffer nevű változóban kapjuk vissza. |
| Visszatérés | Nincs.                                                                                                                                                                                                                        |







## 8. Lekérdező programok készítése

A dokumentáció eddigi része csaknem teljes egészében a lekérdezési definícióval foglalkozott. A lekérdezés leírásából azonban csak akkor lesz adatokat tartalmazó kész lekérdezés, ha létezik egy olyan eszköz, amely értelmezni tudja ezeket a leírásokat.

Ez az eszköz lehet egy kész általános célú lekérdező program vagy egy külön erre a célra készült lekérdező C nyelvű program.

A C nyelvű program is különböző bonyolultságú lehet, függően attól, hogy milyen egyéb segédeszközöket használunk, illetve mennyire kívánjuk módosítani a standard lekérdezési folyamatot.

### 8.1. Standard lekérdező program

A standard lekérdező program funkciója:

Programozás nélkül lehetővé teszi MISAM állományok lekérdezését és módosítását. Megkötései a következők:

- A standard lekérdezési logikával dolgozik.
- Számított mezők nincsenek.
- Csak a masterállomány rekordjait lehet módosítani.

Hívása:

Standard Query, Nov/30/1989 Produced by MI Software

Usage: >STDQRY options

where options are:

|          |                                                |
|----------|------------------------------------------------|
| -b       | black and white colors                         |
| -d       | deletion is allowed                            |
| -f qfile | Misam description file (prohibits -q)          |
| -l l     | language (E: English, H: Hungarian, G: German) |
| -m       | modification is allowed                        |
| -q qfile | MIQuel description file (prohibits -f)         |
| -w wdir  | directory of temporary files                   |

- A -m opció lehetővé teszi a master rekordok módosítását.
- A -d opció lehetővé teszi a master rekordok törlését.
- A -f opció esetén egy állomány lekérdezéséről van szó, és a program nem lekérdezési, hanem file-leírásból dolgozik. Ezt a file-leíró kell az opció után megadni. Nem lehet a -q opcióval együtt használni.
- A -q opció esetén lekérdezési definícióból dolgozik a



- program. Ezt a leíró kell az opció után megadni. Nem lehet a -f opcióval együtt használni.
- A -w opció után egy DOS könyvtár nevét adhatjuk meg. A program futása során keletkezett munkaállományok ebben a könyvtárban fognak képződni. Ha nem használjuk ezt az opciót akkor az ideiglenes állományok vagy a queryleíróban megadott könyvtárba kerülnek, vagy az aktuális könyvtárba.
  - A -l opcióval lehet beállítani, hogy milyen nyelvű üzeneteket adjon a program.

A következő példában egy file lekérdezését mutatjuk be a standard lekérdező programmal. A file leírása (FOKONYV.FDF):

```
#
#      Data file description
#
FOKONYV
dat\fokonyv.dbf
A főkönyvi törzsállomány
#
#      Indexfile descriptions
#
1
FOK
dat\fokonyv1.ndx
Kulcs a főkönyvi szám szerint
1
0      15      A
#
#      Data record field descriptions
#
5
fokonyv
Főkönyvi számlaszám
0      15      1      A      15      0
megnev
Megnevezés
15      30      1      A      30      0
nyito
Nyitó egyenleg
45      8      1      D      12      2
tartozik
Az éves kumulatív tartozik forgalom
53      8      1      D      14      2
kovetel
Az éves kumulatív követel forgalom
61      8      1      D      14      2
```

A program hívása a következő:

```
STDQRY -m -d -f fokonyv
```



A -m és -d opció miatt az állomány módosítása is lehetséges. A hívás után a master kulcs összeállítása következik. A képernyő formája a 15. ábrán látható.

| Keresési szempontok           |  |
|-------------------------------|--|
| Kulcs a főkönyvi szám szerint |  |
| Új index                      |  |

15. ábra  
Keresési szempontok megadása.

Az ablakban fel vannak sorolva azok a lehetséges kulcsok, amelyekre a masterállomány indexelve van. Ha ezek közül nekünk egyik sem felel meg, akkor az Új index kiválasztásával tetszőleges kulcsot állíthatunk össze a master rekordmezőiből. A kulcs összeállítás hasonló a Query Editor master kulcs összeállító funkciójához. Ha új indexet adtunk meg, akkor a program a tényleges lekérdezés előtt felépíti ezt az indexállományt, és a programból való kiszálláskor törli ezeket az állományokat. Ha a lekérdezés után visszatérünk ehhez az ablakhoz, akkor a felépített indexállományok már szerepelni fognak ebben az ablakban, azon a néven, amelyiken a program létrehozta őket.

A kulcs kiválasztása után a keresés kezdőkulcsát adhatjuk meg, a Query Editor intervallum megadási funkciójához hasonlóan.

A kulcs kiválasztása után kezdődik a lekérdezés a megadott szempontok szerint. A lekérdező képernyő egy automatikusan generált form képernyő:

| Standard Query      |                       |
|---------------------|-----------------------|
| fokonyv             | : 121                 |
| megnev              | : Gepek, berendezések |
| nyito               | : 0.00                |
| tartozik            | : 100.00              |
| kovetel             | : 0.00                |
| F1: Help, ESC: Exit |                       |

16. ábra  
Standard lekérdező képernyő.

A view sorai között a PGDN/PGUP billentyűkkel tudunk mozogni. A Ctrl-PgUp és a Ctrl-PgDn billentyűkkel a view első és utolsó sorát kérhetjük a képernyőre. Több állomány együttes lekérdezése esetén, azok a mezők, amelyek nem a masterállomány rekordjaiból



származnak úgynevezett védett mezők. A többi mező szabadon módosítható a képernyőn. A képernyőn végzett módosítások mentése az állományba az F2 billentyűvel történik. Az F3-as billentyűvel az aktuális master rekord törölhető az állományból. Ha a képernyőn olyan mezőket módosítottunk, amelyek beletartoznak a kiválasztott kulcsba, akkor az F10-es billentyű hatására a program az új kulcsérték szerinti view-sorra pozícionál.

Ha leírófile alapján dolgozunk, és a leírófile batch módú lekérdezést ír elő, akkor a program előbb felépíti az ideiglenes állományt, és egyből ezt a képernyőre adja. Ebben az esetben a módosító és törlő funkció értelmét veszti, és az ESC leütése után a program befejezi működését.

Ha nem batch módban fut a program akkor az ESC leütése után visszatér a kulcs kiválasztó képernyőre, és a lekérdezés más szempontok szerint mindaddig ismételhető, amíg innen nem ESC-el szállunk ki.

A standard lekérdező program lekérdező részének C nyelvű forrásából jól látható a standard interaktív lekérdezés menete.

```
/******Standar query******/
```

```
#include <stdio.h>
#include <cex2.h>
#include <setattr.h>
#include <misam.h>
#include <form.h>
#include <kbget.h>
#include <miquel.h>
```

```
#define NO          0
#define YES         1
#define XFNAME      81

#define cbase       glcolors[0]          /* alapszin          */
#define chigh       glcolors[1]          /* kiemelt szín      */
#define cedit       glcolors[2]          /* edit szín         */
#define onsg        glcolors[3]          /* error szín        */
#define chelp       glcolors[4]          /* help szín         */
#define CLNUM       5
```

```
char glcolors[] =
{
    COLOR(CYAN,BLUE),
    COLOR(YELLOW,BLUE),
    COLOR(YELLOW,BLACK),
    COLOR(GRAY,RED),
    COLOR(BLACK,GREEN)
};
```

```
char *fbuf;          /* Form buffer      */
```



```

char *detbuf;          /* Query buffer          */
int  mflag = 0;        /* Van-e modosito funkcio */
int  dflag = 0;        /* Van-e torlo funkcio    */

```

```

int  MI_language = ENGLISH;

```

```

main(argc,argv)

```

```

int argc;

```

```

char **argv;

```

```

{
char  fname[XFNAME+1];    /* Leiro file neve        */
char  tmp1ib[XFNAME+1];   /* Munkakonyvtar neve     */
int   qflag = 0;          /* Lekerdezes query leirobol */
int   fflag = 0;          /* Lekerdezes misam leirobol */
int   wflag = 0;          /* Munkakonyvtar opcio     */
KPARTS *new_key;          /* Kulcs definicio        */
int    partno;             /* Kulcsdarabok szama     */
char  key[129];           /* Intervallum erteke     */
int    i;                  /* Ciklus valtozo         */

```

```

/*
 * Parancs sor argumentumok feldolgozasa
 */

```

```

if(scf("qfwmh", "qfw", &argc, &argv) == 0)

```

```

{
    print_usage();
    exit(1);
}

```

```

for(i = 1; stromp(argv[i], "--"); i++)

```

```

{
    switch(argv[i][1])
    {
        case 'h':
            print_usage();
            exit(0);

        case 'q':
            qflag++;
            strcpy(fname, argv[++i]);
            break;

        case 'f':
            fflag++;
            strcpy(fname, argv[++i]);
            break;

        case 'w':
            wflag++;
            strcpy(tmp1ib, argv[++i]);
            break;

        case 'm':
            mflag++;

```



```

        break;
    case 'd':
        dflag++;
        break;
    case 'l':
        switch(toupper(*argv[i+1]))
        {
            case 'E':
                MI_language = ENGLISH;
                break;
            case 'H':
                MI_language = HUNGARIAN;
                break;
            case 'G':
                MI_language = GERMAN;
                break;
            default:
                print_usage(1);
        }
    }
}

scfend();

/*
 * Kepernyo torles
 */

getcolors(glcolors);
erase();

/*
 * Query nyitas
 */

if(qflag)
{
    if(q_open(fname, (struct CFUNC *)NULL) != SUCCESS) /* Query */
    {
        strcat(q_errmsg, " nem létező vgy hibás leíró");
        shmsg(q_errmsg, NULL);
        exit(1);
    }
}
else
{
    if(!fflag)
    {
        fprintf(stderr, "Nincs leíró file\n");
        print_usage();
        exit(0);
    }
    setfext(fname, ".FDF");
}

```



```

if(q_one_file(fname,wflag ? tmplib : NULL) != SUCCESS) /* Misan */
{
    strcat(q_errmsg," nem létező vagy hibás leíró");
    shmsg(q_errmsg,NULL);
    exit(1);
}

}

if(wflag)
    q_work_lib(tmplib);          /* Beallitjuk a munkakönyvtarat */

again:

/*
 *   Mater kulcs összeallitas
 */

if( (new_key = q_build_key(&partno)) == NULL)
{
    if(q_erno != SUCCESS)
    {
        sherror("Hiba a kulcsválasztásban",NULL);
        q_close();
        exit(3);
    }

    else
    {
        q_close();
        exit(0);
    }
}

/*
 *   Kulcs beallitas
 */

if(q_set_new_key(new_key,partno) != SUCCESS)
{
    sherror("Sikertelen kulcs állítás",NULL);
    q_close();
    exit(3);
}

/*
 *   Kezdo kulcsertek bekerese
 */

memset(key,'\0',128);
if((i = q_scan_key(0,key,FIRST_KEY[MI_language],0)) < 0)
    q_clear_first();
else if(i == 1)

```



```

        q_set_first(key);
else
    goto again;

/*
 *   Inicializálás és query indítás
 */

switch(q_init())
{
    case SUCCESS:
    case Q_LKEY:
    case NO_MATCH:
        show();
        break;
    case Q_LERR:
        smmsg("Foglalt állomány", NULL);
        break;
    case Q_ERROR:
        smmsg("Hiba a QUERY inicializálása közben", NULL);
        q_close();
        exit(3);
    case Q_EOF:
        smmsg("Üres állomány vagy nem létező kulcs", NULL);
        break;
    case OPEN:
        smmsg("A ", q_errmsg, " állomány nem nyitható meg", NULL);
        exit(1);
}

goto again;
}

/*
 *   Definíciók a form képrenyő generalasához
 */
int  det_help();
int  modify();
int  delete();
int  step();

static struct ASSKEY d_keys[] = {
    {F1<<8,      "det_help"},
    {F2<<8,      "modify"},
    {F3<<8,      "delete"},
    {PGDN<<8,    "step"},
    {PGUP<<8,    "step"},
    {F7<<8,      "step"},
    {F8<<8,      "step"},
    {-1, 0}
};

static struct CFUNC DKEY_CFT[] = {

```



```

    {"det_help",    det_help},
    {"modify",      modify},
    {"delete",      delete},
    {"step",        step},
    {0, 0} };

show()
/*
 *   A lekérdezőt vergerhajto modul
 */
{
    char          head[80];
    int           actform;
    struct ASSKEY *keys;
    int           i;

/*
 *   Form képernyő nyitása
 */
    keys = (struct ASSKEY *)malloc(8 * sizeof(struct ASSKEY));
    for(i = 0; i < 8; i++)
    {
        keys[i].keycode = d_keys[i].keycode;
        keys[i].keyfp = strdup(d_keys[i].keyfp);
    }
    memset(head, '\0', 80);
    sprintf(head, "Standard Query");
    actform = det_form(
        head,
        "F1: Help, ESC: Exit",
        gcolors,
        keys,
        DKEY_CFT);

    if (actform < 0)
    {
        shmsg("Hiba a képernyő generálása közben", NULL);
        q_close();
        exit(3);
    }

/*
 *   A detbuf tartalmazza az aktuális view-sort
 */

    detbuf = q_get_detail();

/*
 *   Képernyő inicializálás
 */

    Form_Init();
    Form_Clear();

```



```
Form_Text();
fbuf = Form_Buffer();
```

```
/*
 *      Az aktualis view-sor a form bufferbe
 */
```

```
movbytes(fbuf, detbuf, q_det_size());
Field_Def_All(YES);
Form_Fields();
```

```
Form_Exec(0);
erase();
Form_Close();
free(keys);
return;
}
```

```
static char deleted = NO;      /* Torolt-e a kepernyon levo rekord */
```

```
step(pressed, fldno, actrec)
```

```
/*
 *      A form altal hivott rutin. Megadott iranybol betolti
 *      a form bufferbe a kovetkezo view-sort
 */
```

```
int    pressed;
int    fldno;
int    actrec;
{
int    dir;
```

```
switch(pressed)
```

```
{
    case SCAN(PGDN):
        dir = NEXT;
        break;
    case SCAN(PGUP):
        dir = PREVIOUS;
        break;
    case SCAN(F7):
        dir = FIRST;
        break;
    case SCAN(F8):
        dir = LAST;
        break;
}
```

```
switch(q_exec(dir))
```



```

{
case SUCCESS:
case Q_LKEY:
case Q_FKEY:
case NO_MATCH:
    break;
case Q_EOF:
    shmsg("Állomány vége",NULL);
    return(NO);
case Q_BOF:
    shmsg("Állomány eleje",NULL);
    return(NO);
case Q_LERR:
    shmsg("Foglalt állomány",NULL);
case Q_ERROR:
    return(NO);
}

deleted = NO;
movbytes(fbuf,detbuf,q_det_size());
Field_Def_All(YES);
Form_Fields();
return(NO);
}

```

```

modify()
/*
 *   A form által hívott rekord. Módosítja a képernyő tartalmával
 *   az aktuális master rekordot
 */
{
FDESC *fdsc;          /* A master misam tablaja */
char *buffer;          /* Eredeti rekord buffer */
char *newbuf;          /* Módosított rekord buffer */
int error;             /* Hiba kód */

if(!mflag)
    return(NO);

if(deleted)
{
    shmsg("A rekordot már törölte",NULL);
    return(NO);
}

fdsc = q_get_fdf(0);
buffer = q_get_misam(0);
newbuf = myalloc(q_get_recrlen(fdsc));
q_upd_rec(newbuf,fbuf,0);

```

```

if(!swlock(fdsc) != SUCCESS)

```



```

    {
        shmsg("Foglalt az állomány",NULL);
        free(newbuf);
        return(0);
    }

error = isupdate(fdsc,newbuf,buffer);
iswunlock(fdsc);
if(error == SUCCESS && fill_form())
    shmsg("A rekord módosítva",NULL);
else
    shmsg("Módosítás nem sikerült",NULL);
free(newbuf);
return(0);
}

delete()
/*
 *      A form által hívott rekord. Torli az aktualis master rekordot
 */
{
    FDSC *fdsc;          /* A master misem tablaja */
    char *newbuf;         /* Módosított rekord buffer */
    int error;            /* Hiba kód */

    if(!dflag)
        return(NO);
    if(deleted)
    {
        shmsg("A rekordot már eltörölte",NULL);
        return(NO);
    }
    if(!shgetans("Töröljük a rekordot ?",NULL))
        return(NO);
    fdsc = q_get_fdf(0);
    newbuf = myalloc(q_get_reclen(fdsc));
    q_upd_rec(newbuf,fbuf,0);
    if(iswlock(fdsc) != SUCCESS)
    {
        shmsg("Foglalt az állomány",NULL);
        free(newbuf);
        return(0);
    }
    error = isdelete(fdsc,newbuf);
    iswunlock(fdsc);

    if(error != SUCCESS)
        shmsg("A törlés nem sikerült",NULL);
    else
    {
        shmsg("A rekord törölve",NULL);
        deleted = YES;
    }
}

```



```

    }
    free(newbuf);
    return(0);
}

fill_form()
/*
 *   Aktualizálja a form buffer tartalmát a módosításoknak
 *   megfeleleoen
 */
{
    if(q_upd_det(fbuf))
        return(NO);
    movbytes(fbuf,detbuf,q_det_size());
    Form_Fields();
    return(YES);
}

print_usage()
{
    Printf("Standard Query, Nov/30/1989 Produced by MI Software\n\n");
    Usage: >STDQRY options\n\n
    where options are:\n\
        -b           black and white colors\n\
        -d           deletion is allowed\n\
        -f qfile     Misam description file (prohibits -q)\n\
        -l l         language (E: English, H: Hungarian, G: German)\n\
        -m           modification is allowed\n\n");
    Printf("\
        -q qfile     MiQuel description file (prohibits -f)\n\
        -w wdir      directory of temporary files\n\
    ");
    return;
}

/*--> end of STDQRY.C <--*/

```

## 8.2. Keretprogramok írása

Az alábbiakban mintákat adunk meg lekérdező C programokra. Az első egy MISAM állomány egyszerű interaktív lekérdezése. A második két MISAM állomány együttes listázása saját szelekciós függvényvel. A harmadik ugyanennek a két állománynak az interaktív lekérdezése, saját mezőbetöltő függvény illesztésével.



### 8.2.1. Egy file lekérdezése

Ez a mintaprogram az előbb látott főkönyvi törzsfile interaktív karbantartó programja. A program lehetőséget ad a törzsfile rekordjainak a képernyőn való nézegetésére, módosítására és törlésére. A képernyőn egyszerre csak egy rekord látható, és ezt lehet módosítani vagy törölni. A rekordok között a főkönyvi szám szerint lehet lapozni, ami a törzsfile indexe. Mivel csak egy állományról van szó, amelynek rendezettsége megegyezik a kívánt rendezettséggel, és az egész rekordot akarjuk a képernyőre kérni, használhatjuk a `q_one_file` rutint a lekérdezéshez. Így nincs szükség `quyrelíróra`.

A rekordok képernyőn való megjelenítésére az MFORM adatbeviteli rendszert használjuk. A MIQUEL és az MFORM futtatórendszere között a kapcsolatot a két rendszer adatbufferei teremtik meg. Az MFORM adatbufferét a `Form_Buffer` rutinnal lehet elérni. Ebbe a bufferbe kell bemásolni a MIQUEL `q_get_detail` rutinja által visszaadott buffert.

A mintaprogram forráslistája:

```

/*          PELDA1.C
 *          Főkönyvi törzs karbantartó mintaprogram
 */

#include <stdio.h>      /* Standard C header */
#include <cex2.h>
#include <misam.h>      /* MISAM header */
#include <miquel.h>      /* MIQUEL header */
#include <form.h>        /* M-FORM header */
#include "fokonyv.fct"  /* FORMEDIT által generált C függvény tábla */

/* Billentyű kódok */
#define ESC             '\33'
#define PGUP           73
#define PGDN           81
#define CTR_PGUP       (-124)
#define CTR_PGDN       118
#define F2              60
#define F3              61

int MI_language = HUNGARIAN;

main(argc,argv)
int   argc;
char  *argv[];
{
    int   rc;          /* Retun kód */

```



```

erase(); /* Képernyő törlés */

/* Query megnyitása egy állományra query leíró nélkül */

if(q_one_file("_FILEDEF\FOKONYV", (char *)0))
{
    printf("%s nem nyitható meg!\n", "_FILEDEF\FOKONYV.FDF");
    exit(-1);
}

/* Query inicializálása és az első view-sor beolvasása */

switch(q_init())
{
    case SUCCESS:
        break;
    case Q_LERR:
        printf("Az állomány nem hozzáférhető!\n");
        exit(0);
    case Q_ERROR:
        printf("Fatális hiba!\n");
        exit(-1);
    case Q_EOF:
        printf("Üres állomány!\n");
        exit(0);
}

/* Form nyitása és inicializálása */
if(Form_Read("_form\fokonyv") < 0)
{
    printf("Form nyitási hiba!\n");
    exit(-1);
}

Set_Form_Funcs(FOKONYV_FCT);
Form_Init();
Form_Clear();
Form_Text();

movbytes(Form_Buffer(), q_get_detail(), Form_Size());
again:
Field_Def_All(1);
Form_Fields();

/* Form végrehajtás */
rc = Form_Exec(0);
switch(rc)
{
    case ESC:
        break;
    case SCAN(PGUP):

```



```
        get_next(rc);
        goto again;
    case SCAN(PGDN):
        get_next(rc);
        goto again;
    case SCAN(CTR_PGDN):
        get_next(rc);
        goto again;
    case SCAN(CTR_PGUP):
        get_next(rc);
        goto again;
    case SCAN(F2):
        if(modify())
            get_next(SCAN(PGDN));
        goto again;
    case SCAN(F3):
        if(delete())
            get_next(SCAN(PGDN));
        goto again;
    default:
        goto again;
}
```

```
/* Query és Form zárás */
```

```
Form_Clear();
```

```
Form_Close();
```

```
q_close();
```

```
exit(0);
```

```
}
```

```
get_next(dir)
```

```
/* Beolvassa a kívánt view-sort */
```

```
int dir;
```

```
{
```

```
switch(dir)
```

```
{
```

```
case SCAN(PGDN):
```

```
    dir = NEXT;
```

```
    break;
```

```
case SCAN(PGUP):
```

```
    dir = PREVIOUS;
```

```
    break;
```

```
case SCAN(CTR_PGUP):
```

```
    dir = FIRST;
```

```
    break;
```

```
case SCAN(CTR_PGDN):
```

```
    dir = LAST;
```

```
    break;
```

```
}
```



```

switch(q_exec(dir))
{
    case SUCCESS:
    case NO_MATCH:
        break;
    case Q_EOF:
        shmsg("Állomány vége.",NULL);
        return;
    case Q_BOF:
        shmsg("Állomány eleje.",NULL);
        return;
    case Q_LERR:
        shmsg("Az állomány nem hozzáférhető.",NULL);
        return;
    case Q_ERROR:
        shmsg("Fatális hiba!",NULL);
        return;
}

movbytes(Form_Buffer(),q_get_detail(),Form_Size());
return;
}

modify()
/* Módosítja az aktuális rekordot a képernyőn lévő adatokkal */
{
    FDESC *fdsc;
    fdsc = q_get_fdf(0);
    if(!isupdate(fdsc,Form_Buffer(),q_get_misam(0)))
    {
        sherror("Módosítás nem sikerült!",NULL);
        return(0);
    }

    shmsg("Rekord módosítva.",NULL);
    return(1);
}

delete()
/* Törli az aktuális rekordot */
{
    FDESC *fdsc;

    fdsc = q_get_fdf(0);
    if(!isdelete(fdsc,q_get_misam(0)))
    {
        sherror("Törlés nem sikerült!",NULL);
        return(0);
    }

    shmsg("Rekord törölve",NULL);
    return(1);
}

```



```

int my_break(pressed, fieldnum, actrec)
/* Hot key function of key 'PgUp' */
int   pressed;      /* ascii+scan code of the pressed key */
int   fieldnum;     /* the serial number of the actual field */
int   actrec;       /* actual subrecord (-1, if not subrecord) */
{
  Form_Break(pressed); /* leave Form_Exec() returning the 'pressed' */
  return(0);           /* pressed key ; 0 for again ; -1 for screen */
}
/* end of peldal */

```

A módosításnál a képernyőn módosított rekord adatait az MFORM bufferéből kapjuk, az eredeti rekordot a MIQUEL q\_misam\_buf rutinjával kapjuk meg. Ezt a két buffert kell átadni a MISAM isupdate rutinjának.

### 8.2.2. Több file listázása

Ez a mintaprogram két file együttes listázásra ad példát. A bemutatott főkönyvi törzsfile-hoz tartozik egy forgalmi file, amely a különböző főkönyvi számokhoz tartozó könyvelési tételeket tartalmazza. Ez az állomány is a számlaszámra van indexelve. A kapcsolatot a törzsfile és a forgalmi állomány között, a számlaszám teremti meg. A forgalmi állomány MISAM leírója a következő:

```

#
#   Data file description
#
FORG
dat\forg.dbf
Folyószámla forgalmi file
#
#   Indexfile descriptions
#
2
BIZSZ
dat\forg1.ndx
Kulcs a bizonylatszám szerint
1
0      8      A
SZLASZAM
dat\forg2.ndx
Kulcs a főkönyviszám szerint
1
8      15     A
#
#   Data record field descriptions
#
6

```



```

bizszám
A bizonylat száma
0      8      1      A      8      0
szlaszam
Főkönyvi szám
8      15      1      A      15      0
bizdat
A bizonylat kelte
23     4      1      M      8      0
hiv
Hivatkozási szöveg
27     10     1      A      10     0
mozgmen
A mozgásnem: t. ha tartozik, k. ha követel
37     1      1      A      1      0
forgalom
A forgalom abszolút értéke
38     8      1      D      14     2

```

A mintaprogram célja egy forgalmi kimutatás készítése. A kimutatás minden főkönyvi számhoz kilistázza a hozzá tartozó forgalmi tételeket. A lekérdezést a queryleíró alapján végzi a program. A leírófile a következő:

```

*****
* "FORKIM.QTX" QUERY leiro file.
* Generated by Query Editor (MI Software)
* on Oct/10/1989 09:12:28
*****
* MISAM files
*
ISAM_NO=2
MASTER=_filedef\FOKONYV.FDF
ISAM=_filedef\FORGALOM.FDF
*****
* Link definitions
*
LINKED=FOKONYV
LINKED_NO=1
*
ISAM_NAME=FORG
PARTNO=1
PARTS=FOKONYV,0,15,A
PARTNO=1
PARTS=SZLASZAM,0,15,A
KEY=SET
*****
*
* Detail row
*
DETAILFIELDS=10

```



```

FIELD=0,15,1,A,15,0,FOKONYV,FOKONYV,0,fokonyv
FIELD=15,30,1,A,30,0,FOKONYV,MEGNEV,0,megnev
FIELD=45,8,1,D,12,2,FOKONYV,NYITO,0,nyito
FIELD=53,8,1,D,14,2,FOKONYV,TARTOZIK,0,tartozik
FIELD=61,8,1,D,14,2,FOKONYV,KOVETEL,0,kovetel
FIELD=69,8,1,A,8,0,FORG,BIZSZAM,0,bizszam
FIELD=77,4,1,M,8,0,FORG,BIZDAT,0,bizdat
FIELD=81,10,1,A,10,0,FORG,HIV,0,hiv
FIELD=91,8,1,D,14,2,FORG,FORGALOM,0,forgalom
FIELD=99,1,1,A,1,0,FORG,MOZGNEM,0,mozgnem
*****
*
*      Master key
*
TMPLIB=C:\TMP
PARTNO=1
PARTS=FOKONYV,0,15,A
FIRST_KEY=0000000000000000
LAST_KEY=9999999999999999
INTERVALLUM=ALL
SEL_FUNC=chk_forg
*****
*
*      Processing
PROCESSING=INTERACTIVE
*
*      end of FORKIM.QTX
*****

```

A törzsfile lesz a masterfile, és hozzá kapcsolódik a főkönyvi szám alapján a forgalmi állomány. Mivel egy főkönyvi számhoz több forgalmi tétel is tartozik, a link kapcsolatot SET típusúnak adtuk meg, hogy a futtatórendszer dolgozza fel az összes forgalmi rekordot. Így a masterállományon mindaddig nem fogunk tovább lépni, amíg van a master rekordhoz tartozó forgalmi rekord.

A kimutatásba csak azokat a főkönyvi számokat akarjuk bevenni, amelyekhez tartozik forgalmi rekord. Az ilyen törzsrekordoknál mind a tartozik, mind a követel forgalom nulla. Ezeket a rekordokat a lekérdezésből szelekciós függvénnyel hagyhatjuk el. A függvény a leíróban `chk_forg` néven van definiálva. A felhasználói programban a `q_open` rutinnak át kell adni paraméterként a `QRYEDIT` által generált függvénytáblát, és meg kell írni a `chk_forg` nevű rutint. Ezt a rutint a futtatórendszer minden alkalommal meghívja, amikor összeállított egy view-sort. A szelekciós függvénynek kell eldönteni, hogy ezt a view-sort a `q_exec` rutin visszatér-e a programnak, vagy próbálja meg a következő sort összeállítani. Ha a függvény visszatérési értéke 0, akkor a futtatórendszer nem hagyja ki ezt a view-sort a lekérdezésből, egyébként kihagyja. A futtatórendszer a szelekciós függvényt paraméter nélkül hívja meg.



A mintaprogram lehetővé teszi, hogy intervallumot jelöljünk ki a kimutatásba bekerülő főkönyvi rekordokra. Ezt a `q_first_key`, a `q_last_key`, a `q_get_key`, a `q_set_first` és a `q_set_last` rutinokkal végzi. A leírófile-ban az intervallum úgy van kijelölve, hogy minden rekord beleessen.

A listázás technikai részeit a MIREP reportgenerátor rendszerrel oldottuk meg. A MIREP és a MIQUEL közötti adatforgalmat a `qbuf` és az `rbuf` globális változókon keresztül valósítjuk meg.

A mintaprogram listája a következő:

```

/*                                PELDA2.C
 *      Főkönyvi forgalmi kimutatás mintaprogram
 */

#include <stdio.h>      /* Standard C header                */
#include <ctype.h>      /* Standard C header                */
#include <setattr.h>     /* Header of global colors          */
#include <misam.h>      /* MISAM header                    */
#include <miquel.h>     /* MIQUEL header                   */
#include <mirep.h>      /* MIREP header                    */
#include "forkim.rh"    /* REPEDIT által generált header    */
#include "forkim.qh"    /* QRYEDIT által generált header    */
#include "forkim.qct"   /* QRYEDIT által generált függvény tábla */

char  glcolors[] =      /* Színek tömbje az interaktív query rutinoknak */
{
    COLOR(CYAN,BLUE),
    COLOR(YELLOW,BLUE),
    COLOR(YELLOW,BLACK),
    COLOR(GRAY,RED),
    COLOR(BLACK,GREEN)
};

int  MI_language = HUNGARIAN; /* Magyar nyelvű program */

main(argc,argv)
int  argc;
char  *argv[];
{
    struct Q_SFORKIM  *qbuf;      /* Query buffer */
    struct R_FORKIM  *rbuf;      /* Report buffer */
    int              rc;          /* Return kód */
    char             key[sizeof(qbuf->fokonyv)]; /* Intervallum */

    erase();                    /* Képernyő törlés */

    /* Report megnyitása */

    if(r_read("_REPORT\\FORKIM"))

```



```
{
    printf("%s nem nyitható meg!\n", "_REPORT\\FORKIM.RTX");
    exit(-1);
}

/* Hova kérjük a listát */

if(argc < 2)
    usage();

switch(toupper(argv[1][0]))
{
    case 'C':
        /* consolra */
        r_set_prntype("CONSOL");
        r_set_device(CONSOLE);
        break;
    case 'D':
        /* file-ba */
        r_set_prntype("CONSOL");
        r_set_device(DISKFILE);
        r_set_lstfdir("C:\\TMP");
        break;
    case 'L':
        /* nyomtatóra */
        r_set_prntype("EPSON");
        r_set_device(LOCALPRINTER);
        break;
    default:
        usage();
}

if(r_init(NULL))
{
    printf("Sikertelen report init!\n");
    exit(-1);
}

/* Query megnyitása query leíróból */

if(q_open("_QUERY\\FORKIM", FORKIM_QCT))
{
    printf("%s nem nyitható meg!\n", "_QUERY\\FORKIM.QTX");
    exit(-1);
}

/* Intervallum bekérése */

q_get_first(key);
if( (rc = q_scan_key(0, key, "Kezdő számla szám", 0)) < 0)
    q_clear_first();
else if(rc == 1)
    q_set_first(key);
else
```



```

    {
        q_close();
        exit(0);
    }

q_get_last(key);
if( (rc = q_scan_key(0, key, "Végső számla szám", 0)) < 0 )
    q_clear_last();
else if(rc == 1)
    q_set_last(key);
else
    {
        q_close();
        exit(0);
    }

/* Query inicializálása és az első view-sor beolvasása */

switch(q_init())
{
    case SUCCESS:
    case NO_MATCH:
        break;
    case Q_LERR:
        printf("Az állomány nem hozzáférhető!\n");
        exit(0);
    case Q_ERROR:
        printf("Fatális hiba!\n");
        exit(-1);
    case Q_LKEY:
        printf("Nincs az interвалumba eső rekord\n");
        exit(0);
    case Q_EOF:
        printf("Üres állomány!\n");
        exit(0);
}

/* Listázás */

qbuf = (struct Q_SFORKIM *)q_get_detail();
rbuf = (struct R_FORKIM *)r_d_buffer();
do
    {
        movbytes(rbuf, qbuf, r_d_bufilen());
        r_exec();
        rc = q_exec(NEXT);
    }
while(rc == SUCCESS || rc == NO_MATCH);
r_total_print(1);
r_close();
q_close();

```



漢書卷之六十五

\*\*\*\*\*

\* MISAM files

MASTER= filedef\FOKONYV.FDE

TSAM: filedef\FORGALOM.FDE

\*\*\*\*\*

\* Link definitions



```

LINKED=FOKONYV
LINKED_NO=1
*
ISAM_NAME=FORG
PARTNO=1
PARTS=FOKONYV,0,15,A
PARTNO=2
PARTS=SZLASZAM,0,15,A
PARTS=BIZDAT,0,4,M
KEY=UNIQUE
*****
*
*      Detail row
*
DETAILFIELDS=11
FIELD=0,15,1,A,15,0,FOKONYV,FOKONYV,0,fokonyv
FIELD=15,30,1,A,30,0,FOKONYV,MEGNEV,0,megnev
FIELD=45,8,1,D,12,2,FOKONYV,NYITO,0,nyito
FIELD=53,8,1,D,14,2,FOKONYV,TARTOZIK,0,tart_sum
FIELD=61,8,1,D,14,2,FOKONYV,KOVETEL,0,kov_sum
FIELD=69,8,1,A,8,0,FORG,BIZSZAM,0,bizszam
FIELD=77,4,1,M,8,0,FORG,BIZDAT,0,bizdat
FIELD=81,10,1,A,10,0,FORG,HIV,0,hiv
FIELD=91,8,1,D,12,2,$USER,tartozik,0,tartozik,get_forgalom
FIELD=99,8,1,D,12,2,FORG,FORGALOM,0,kovetel
FIELD=107,1,1,A,1,0,FORG,MOZGNEV,0,mozgnev
*****
*
*      Master key
*
TMPLIB=C:\TMP
PARTNO=1
PARTS=FOKONYV,0,15,A
INTERVALLUM=ALL
*****
*
*      Processing
PROCESSING=INTERACTIVE
*
*      end of forlek.GRF
*****

```

A leírófile-ban a link kapcsolatot nem vettük SET típusúnak. Ezt azért tettük, mert az egy főkönyvi számhoz tartozó forgalmi tételeket programból vezérelve gyűjtjük össze egy dinamikus tömbbe a `q_next_link` rutinnal. Ha SET kapcsolatot definiáltunk volna, akkor a visszafelé lapozásnál a programunk nem működne helyesen.

Figyeljük meg, hogy a link kulcs hosszabb mint a master kulcs. Ennek eredményeként a forgalmi rekordok a bizonylat dátuma



szerint rendezetten fognak megjelenni a képernyőn. Mivel a forgalmi állomány nem rendezett a bizonylat dátuma szerint, a futatórendszer `q_init` rutinja egy ideiglenes indexállományt fog építeni, amelyet a `q_close` rutin töröl el. Az ideiglenes állomány a `C:\TMP` nevű DOS könyvtárban fog képződni. Ezért a program futásához ennek a könyvtárnak léteznie kell.

A lekérdezési képernyőn külön oszlopban jelentetjük meg ha a forgalmi tételen tartozik vagy követel forgalom van. Ezért a view-sorban is felvettünk egy tartozik és egy követel oszlopot. Mivel a forgalom állományban ilyen mezők nincsenek, egy user mezőt kell felvenni. A tartozik és a követel oszlopoknak a mezőbetöltő függvényekkel tudunk értéket adni. A betöltő függvény neve `get_forg`. A `q_open` függvényt a generált függvénytáblával kell meghívni. A `get_forg` rutinra mindig akkor kerül a vezérlés, ha a futatórendszer a többi nem user típusú mezőt már feltöltötte. A mezőbetöltő függvényt két paraméterrel hívja meg a futatórendszer. A betöltő függvény prototípusa a következő:

```
void load_field(fieldnum,addr)
int   fieldnum;      /* Mező sorszáma */
char  *addr;         /* Mező címe */
{
  q_field_set(fieldnum,"USER");
}
```

A mintaprogram betöltő függvénye a mozgásnemtől függően tölti fel a tartozik és a követel oszlopokat a forgalom értékével.

A mintaprogram listája a következő:

```
/*          PELDA3.C
 *          Főkönyvi forgalom lekérdező mintaprogram
 */

#include <stdio.h>      /* Standard C header */
#include <cex2.h>
#include <ainser.h>     /* Dinamikus struktúra tömb kezelése */
#include <misam.h>      /* MISAM header */
#include <miquel.h>     /* MIQUEL header */
#include <form.h>       /* M-FORM header */
#include "forlek.fct"   /* FORMEDIT által generált C függvény tábla */
#include "forlek.qct"   /* QED által generált függvény tábla */
#include "forlek.fh"    /* FORMEDIT által generált header */
#include "forgalom.fh"  /*      -"-      */
#include "forlek.qh"    /* QED által generált header */

/* Billentyű kódok */
#define ESC      '\33'
#define PGUP     73
#define PGDN     81
```



```

#define CTR_PGUP      (-124)
#define CTR_PGDN      118

struct SFORGALOM
int
    *tet; /* A számlához tartozó tételek */
    tetno; /* A számlához tartozó tételek száma */

int
    MI_language = HUNGARIAN;

main(argc,argv)
int    argc;
char   *argv[];
{
    int    rc; /* Retun kód */

    erase(); /* Képernyő törlés */

    tet = (struct SFORGALOM *)0;
    tetno = 0;

    /* Query megnyitása query leíróból */

    if(q_open("_QUERY\\FORLEK",FORLEK_QCT))
    {
        printf("%s nem nyitható meg!\n", "_QUERY\\FORLEK.QRF");
        exit(-1);
    }

    /* Query inicializálása és az első view-sor beolvasása */

    switch(q_init())
    {
        case SUCCESS:
            break;
        case Q_LERR:
            printf("Az állomány nem hozzáférhető!\n");
            exit(0);
        case Q_ERROR:
            printf("Fatális hiba!\n");
            exit(-1);
        case Q_EOF:
            printf("Üres állomány!\n");
            exit(0);
    }

    /* Form nyitása és inicializálása */
    if(Form_Read("_form\\forlek") < 0)
    {
        printf("Form nyitási hiba!\n");
        exit(-1);
    }

```



```
Set_Form_Funcs(FORLEK_FCT);
Form_Init();
Form_Clear();
Form_Text();

fill_form();          /* Form feltöltése */
again:
Field_Set(F_FORGALOM, (char *)&tetno);
Field_Def_All(1);
Form_Fields();
```

```
/* Form végrehajtás */
rc = Form_Exec(0);
switch(rc)
{
    case ESC:
        break;
    case SCAN(PGUP):
        get_next(rc);
        goto again;
    case SCAN(PGDN):
        get_next(rc);
        goto again;
    case SCAN(CTR_PGDN):
        get_next(rc);
        goto again;
    case SCAN(CTR_PGUP):
        get_next(rc);
        goto again;
    default:
        goto again;
}
```

```
/* Query és Form zárás */
Form_Clear();
Form_Close();
q_close();
exit(0);
}
```

```
get_next(dir)
/* Beolvassa a kívánt view-sort */
int dir;
{
    switch(dir)
    {
        case SCAN(PGDN):
            dir = NEXT;
            break;
        case SCAN(PGUP):
            dir = PREVIOUS;
```



```

        break;
    case SCAN(CTR_PGUP):
        dir = FIRST;
        break;
    case SCAN(CTR_PGDN):
        dir = LAST;
        break;
}

switch(q_exec(dir))
{
    case SUCCESS:
    case NO_MATCH:
        break;
    case Q_EOF:
        shmsg("Állomány vége.", NULL);
        return;
    case Q_BOF:
        shmsg("Állomány eleje.", NULL);
        return;
    case Q_LERR:
        shmsg("Az állomány nem hozzáférhető.", NULL);
        return;
    case Q_ERROR:
        shmsg("Fatális hiba!", NULL);
        return;
}

```

```

fill_form();          /* Form feltöltése */
return;
}

```

```

fill_form()
{
    struct SFORLEK      *fbuf;          /* Form buffer */
    struct Q_SFORLEK    *qbuf;          /* Query buffer */
    struct SFORGALOM    tbuf;          /* Egy forgalmi tétel */
    FDESC               *fdsc;          /* Forgalmi állomány leírója */
    int                  rc;             /* Retun kód */
}

```

```

fbuf = (struct SFORLEK *)Form_Buffer();
qbuf = (struct Q_SFORLEK *)q_get_detail();

```

```

/* Főkönyvi rekordmezők áttöltése a form bufferbe */
movbytes(fbuf->fokonyv, qbuf->fokonyv, sizeof(fbuf->fokonyv));
movbytes(fbuf->megnev, qbuf->megnev, sizeof(fbuf->megnev));
fbuf->nyito = qbuf->nyito;
fbuf->tartozik = qbuf->tart_sum;
fbuf->kovetel = qbuf->kov_sum;

```

```

/* Forgalmi rekordok feltöltése dinamikus tömbbe */
if(tetno)

```



```

{
    free(tet);
    tet = (struct SFORGALOM *)0;
    tetno = 0;
}

fdsc = q_get_fdf(QI_FORG);
do
{
    movbytes(tbuf.bizszam,qbuf->bizszam,sizeof(tbuf.bizszam));
    tbuf.bizdat = qbuf->bizdat;
    movbytes(tbuf.hiv,qbuf->hiv,sizeof(tbuf.hiv));
    tbuf.tartozik = qbuf->tartozik;
    tbuf.kovetel = qbuf->kovetel;
    AINSERT(struct SFORGALOM *,tet,tetno,tetno,&tbuf);
    rc = q_next_link(q_link_sernum(fdsc->dname),NEXT);
}while(rc == SUCCESS || rc == NO_MATCH);
if(rc != SET_END)
{
    if(tetno)
    {
        free(tet);
        tet = (struct SFORGALOM *)0;
        tetno = 0;
    }
    sherror("Keresési hiba a ",fdsc->dname," állományon",NULL);
}
return;
}

```

/\* "FORLEK.FCC" generated by FormEdit (MI Software) \*/

int FORGALOM\_load(actitem,function)

/\* Load function of subform field 'FORGALOM' \*/

```

int  actitem;      /* serial number of the subrecord */
int  function;     /* 0: load,
                    ** 1: insert request,
                    ** -1: position to the átrecl,
                    ** -2: entering into the subform,
                    */

```

```

{
    struct SFORGALOM  *tbuf;      /* Aktuális forgalmi tétel */

```

switch(function)

```

{
    case -1:
    case -2:
    case -3:
        return(1);
    case 0:
        tbuf = tet + actitem;

```



```

        movbytes(Form_Buffer(),(char *)tbuf,Form_Size());
default:
    Field_Def_All(1);
    return(1);
}

}

int FORGALOM_store(actitem,function)
/* Store function of subform field 'FORGALOM' */
int    actitem;          /* serial number of the subrecord      */
int    function;         /* 1: inserting, -1: deleting, 0: edited      */
{
    return(0);           /* Alrekordok módosítása nem megengedett    */
}

int my_break(pressed,fieldnum,actrec)
/* Hot key function of key 'PgUp' */
int    pressed;          /* ascii+scan code of the pressed key      */
int    fieldnum;          /* the serial number of the actual field     */
int    actrec;            /* actual subrecord (-1, if not subrecord)    */
{
    Form_Break(pressed); /* leave Form_Exec() returning the 'pressed' */
    return(0);           /* pressed key : 0 for again ; -1 for screen */
}

/* Query által hívott felhasználói függvény */

get_forgalom(fieldno,addr)
int    fieldno;           /* Mező sorszáma */
char    *addr;            /* Mező query címe */
{
    char    mozgnem;        /* Tartozik vagy követel forgalom */
    double  forgalom;       /* Forgalom abszolút értéke */

    q_field_get(QF_mozgnem,(char *)&mozgnem);
    q_field_get(QF_kovetel,(char *)&forgalom);
    if(toupper(mozgnem) == 'T')
    {
        q_field_set(QF_tartozik,(char *)&forgalom);
        forgalom = 0.0;
        q_field_set(QF_kovetel,(char *)&forgalom);
    }
    else
    {
        q_field_set(QF_kovetel,(char *)&forgalom);
        forgalom = 0.0;
        q_field_set(QF_tartozik,(char *)&forgalom);
    }
}

/* end of pelda3 */

```



Ez a program is az MFORM adatbeviteli rendszert használja a lekérdezés megjelenítésére, mégpedig úgy, hogy a forgalmi tételek egy subform tételeiként jelennek meg. A subform tételei a tet nevű dinamikus tömből kapnak értéket. Ezt a tömböt mindig újra kell tölteni, ha új főkönyvi számot kérdezzünk le. Ezt a fill\_form rutin végzi.



## 9. Query dokumenter

A QRYDOC program a queryleírókról készít egy dokumentációs anyagot, amelyet be lehet illeszteni nagyobb rendszerek dokumentációjába.

A program hívása a következő:

QryDoc Version 1.00 Sep/27/1989 Produced by MI Software

Usage: >QRYDOC qryfile [-o output] [-d] [-i] [-a] [-c c] [-l l]

where qryfile is the name of the queryfile  
 output is the output file (def.: .QDC)  
 -d formatted Document format (def.: IBM display format)  
 -g generate only link structure  
 -i is it an included Document source? (def.: NO)  
 -a append to the existing output  
 -c document Chapter level (1-5, def.: 1 - \$CPT.)  
 -l language (E:English, H:Hungarian, G:German, def.: E)

A -l opció megadásával tudunk magyar nyelvű listát generálni. Alapértelmezésben a lista angol nyelvű. A többi opciónak csak akkor van értelme, ha a DOG szövegszerkesztőt használjuk.

Az ötödik fejezetben megadott leírófile dokumentációja:

Query: "CIKK"

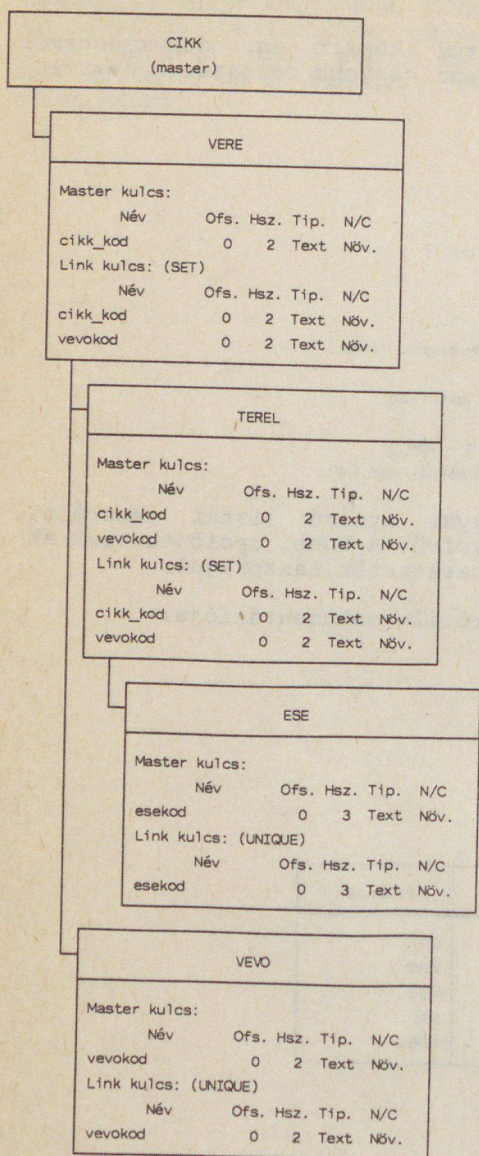
Query leírás: Példa leírófile

Misam leírók:

| Leíró file        | Alias név |
|-------------------|-----------|
| filedef\CIKK.FDF  | CIKK      |
| filedef\VEVO.FDF  | VEVO      |
| filedef\VERE.FDF  | VERE      |
| filedef\ESE.FDF   | ESE       |
| filedef\TEREL.FDF | TEREL     |



Link struktúra:





View oszlopok:

1.: CIKK\_KOD

Misam név : CIKK  
Mező név : cikk\_kod  
Offset : 0  
Hossz : 2  
Típus : Text  
Szélesség : 2

2.: CIKKNEV

Misam név : CIKK  
Mező név : cikknev  
Offset : 0  
Hossz : 10  
Típus : Text  
Szélesség : 10

3.: VEVOKOD

Misam név : VEVO  
Mező név : vevokod  
Offset : 0  
Hossz : 2  
Típus : Text  
Szélesség : 2

4.: VEVONEV

Misam név : VEVO  
Mező név : vevonev  
Offset : 0  
Hossz : 10  
Típus : Text  
Szélesség : 10

5.: DATUM

Misam név : TEREL  
Mező név : datum  
Offset : 0  
Hossz : 4  
Típus : Date  
Szélesség : 8

6.: ESENEV

Misam név : ESE  
Mező név : esenev  
Offset : 0  
Hossz : 10  
Típus : Text  
Szélesség : 10



## 7.: RENDELES

Misam név : VERE  
Mező név : rendeles  
Offset : 0  
Hossz : 8  
Típus : Doub  
Szélesség : 8, dec.: 2

A view rendezettsége:

Master index:

1.:  
Mező név : cikk\_kod  
Offset : 0  
Hossz : 2  
Típus : Text  
Csökk.\_Növ. : Növ.

Munka könyvtár : C:\TMP  
Kezdő kulcs : aa  
Végző kulcs : cc  
Intervallum kezelése : ALL  
Szelekciós függv. : select\_cikk



## 10. Lekérdezés hálózatban

A MIQUEL rendszer hálózatos használata nem igényel külön programozási munkát. A `q_open` rutin megvizsgálja, hogy a lekérdező program hálózatos környezetben fut-e. Ha hálózatban dolgozunk, akkor a query végrehajtó rutinok minden file-olvasás előtt olvasásra zárják az állományokat.

Ha a lekérdező program módosításokat is végez a lekérdezett állományokon, akkor ezeknek az állományoknak az írási zárásáról a programozónak kell gondoskodnia.

A hálózati felhasználás akkor okozhat problémákat, ha a futtatórendszer ideiglenes állományokat hoz létre. Ez akkor fordul elő, ha a futtatórendszert batch módban használjuk, vagy ha olyan rendezettségű van szükségünk, amelyhez indexfile-t kell építeni. A futtatórendszer az ideiglenes állományok nevét a létrehozásuk sorrendjében maga képi. Hálózatban a különböző munkaállomásokról létrehozott állományok a nevek azonosága miatt felülrírhathatják egymást. Ezt a problémát úgy lehet megoldani, hogy a lekérdező program minden munkaállomásnak különböző munkakönyvtárat állít be a `q_work_lib` függvénnnyel. A következő programrészlet egy ilyen megoldást mutat be.

```
#include <stdio.h>
#include <stdlib.h>
#include <misam.h>
#include <miquel.h>

main(argc,argv)
int    argc;
char   *argv[];
{
    int    stnumber;           /* munkaállomás száma */
    char   tmplib[80];         /* munkakönyvtár */

    if(q_open("test",(struct CFUNC *)0))
    {
        printf("Query nyitási hiba\n");
        exit(-1);
    }

    stnumber = atoi(argv[1]);   /* munkaállomás száma argv[1]-ben */
    get_tmp_lib(stnumber,tmplib);
    q_work_lib(tmplib);
    q_init();
    .
    .
    exit(0);
}
```



A program paraméterként megkapja a munkaállomás számát, és a `get_tmp_lib` rutin a kapott szám alapján `tmplib` tömbbe betölti az állomáshoz tartozó munkakönyvtár nevét. A munkakönyvtárak nevét célszerű egy file-ban elhelyezni, és így a `get_tmp_lib` ebből a file-ból dolgozhat.



## 11. A disztributív anyag

A MIQUEL disztributív anyag egy felhasználói kézikönyvből és két hajlékony lemezből áll.

Az első lemez tartalma:

QRYDOC .EXE: lista leírás dokumentálása  
 QRYEDIT .EXE: a query editor  
 LMIQUEL .LIB: Microsoft 5.x futtató rendszer (Large memória-modell)  
 SMIQUEL .LIB: Microsoft 5.x futtató rendszer (Small memória-modell)

A második lemez tartalma:

STDQRY .EXE: a standard lekérdező program  
 MIQUEL .H : MIQUEL header-file  
 MIQUELL .LIB: TurboC 2.x futtató rendszer (Large memória-modell)  
 MIQUELS .LIB: TurboC 2.x futtató rendszer (Small memória-modell)

A második lemez tartalmazza a példaprogramok C nyelvű forrásait, a hozzájuk tartozó leírókat és az adatokat a következő alkönyvtárakban:

DAT : a példa programok által használt MISAM-állományok  
 PL1 : az első példa program  
 PL2 : a második példa program  
 PL3 : a harmadik példa program  
 \_FILEDEF: a példa programokhoz tartozó MISAM-leírók  
 \_FORM : a példa programokhoz tartozó MFORM-leírók  
 \_QUERY : a példa programokhoz tartozó MIQUEL-leírók  
 \_REPORT : a példa programokhoz tartozó MIREP-leírók





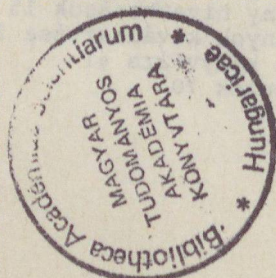


I n d e x

- Aktív view állítás 57
- Alapfogalmak és definíciók 7
- Alias név 7
- Bevezetés 1
- detailfields= 20
- A disztributív anyag 111
- Egy file lekérdezése 86
- Előkészítő modulok 50
- field= 21
- first key 23
- Illesztési pontok 7
- Interaktív lekérdezés 69
- Intervallum 7
- Az intervallum kezelési mód 39
- Intervallum megadása 38
- intervallum= 24
- isam= 16
- isam\_name= 18
- isam\_no= 15
- Keretprogramok írása 85
- key= 19
- Konzisztencia vizsgálat 43
- last key= 24
- A leírófile kiírása 42
- Lekérdezés hálózatban 109
- Lekérdezési mód 8
- Lekérdezési mód megadása 25, 40
- Lekérdező programok készítése 73
- Lekérdező rendszer 8
- A lekérdező rendszer alapelve 3
- Link kapcsolat 9
- Link kapcsolatok kijelölése 16, 33
- Link kulcs 9
- linked= 17
- linked no= 17
- Linkelt állomány 9
- Magyarázó szöveg 31
- Master kulcs 11
- Master kulcs összeállítás 37
- Master kulcsérték 11
- Master misam kijelölése 33
- master= 16
- Masterállomány 11
- A MIQUEL könyv felépítése 5
- A MIQUEL könyvtár 45
- Misam állomány 11
- Misam állomány hivatkozások 15
- Misam állományok kiválasztása 31
- Misam leírók könyvtára 41
- Módosító modulok 70



Munkakönyvtár 12  
Munkakönyvtár megadása 37  
partno= 18, 22  
parts= 19, 23  
A QRYEDIT funkciója 29  
A QRYEDIT hívása és opciói 29  
A QRYEDIT kezelése 30  
Query definíció 12  
Query dokumenter 105  
Query editor 29  
Query információk 58  
Query végrehajtás 63  
Query, lekérdezés 12  
Queryleíró nyelv 15  
q\_build\_key 69  
q\_clear\_all\_ndx 55  
q\_clear\_first 53  
q\_clear\_last 53  
q\_clear\_ndx 55  
q\_close 58  
q\_det\_size 59  
q\_exec 63  
q\_field\_dec 67  
q\_field\_get 68  
q\_field\_is\_valued 68  
q\_field\_length 66  
q\_field\_name 65  
q\_field\_number 68  
q\_field\_offset 66  
q\_field\_sernum 65  
q\_field\_set 67  
q\_field\_size 66  
q\_field\_type 67  
q\_get\_act 58  
q\_get\_detail 58  
q\_get\_fdf 59  
q\_get\_first 60  
q\_get\_last 61  
q\_get\_misam 59  
q\_init 56  
q\_int\_mod 54  
q\_isam\_number 62  
q\_is\_match 60  
q\_link\_init 64  
q\_link\_name 62  
q\_link\_sernum 61  
q\_next\_link 64  
q\_one\_file 51  
q\_open 50  
q\_query\_sernum 62  
q\_scan\_key 70  
q\_select 57





q\_select\_key 54  
q\_set\_first 52  
q\_set\_last 53  
q\_set\_new\_key 51  
q\_upd\_det 70  
q\_upd\_rec 71  
q\_work\_lib 54  
sel\_func= 24  
Set\_kapcsolat 13  
Standard lekérdező program 73  
Szelekciós függvény 40  
tmplib= 22  
Több file lekérdezése 96  
Több file listázása 90  
View 13  
A view egy sorának megadása 20  
A view megadása 35  
A view mezőinek kezelése 65  
A view rendezettségének megadása 22











# MIQUEL

## C nyelvű lekérdező rendszer MISAM adatállományokhoz

Az egymunkahelyes vagy Novell hálózati rendszereken a fájlkezelést egyszerűsíti a MISAM programcsomag. Az egy vagy több MISAM állományokból kialakított adatbázis együttes lekérdezését könnyíti meg a MIQUEL lekérdező programcsomag:

A MIQUEL lekérdező rendszer jellemzői:

- Lehetővé teszi egy vagy több MISAM állományból álló adatbázis intelligens lekérdezését.
- Lehetőséget ad a felhasználónak arra, hogy a lekérdezésbe bevont adatokat és azok rendezettségét tetszés szerint határozza meg.
- Segédprogramokkal támogatja a lekérdezési jellemzők megtervezését és automatizált dokumentálást biztosít.
- Nagy mértékben leegyszerűsíti a programozási munkát.
- Standard lekérdező programmal lehetővé teszi adatbázisok lekérdezését és módosítását.

A lekérdezést egy leíró file vezérli. A leíró file-t a QED query editorral hozhatjuk létre. A programok a MIQUEL tárgykönyvtár moduljain keresztül a leíró file-t használják. A táblázatos közelítésmód miatt a leíró file változtatása általában nem igényel programváltoztatást.

A rendszerhez tartozik egy standard lekérdező program, amellyel bármely MISAM alapú adatbázis lekérdezhető, illetve módosítható és dinamikus párbeszéd-s paraméterezési lehetőségeivel tetszőleges rendezettséget és értékhatár-beállítást megenged.

