GONDOLA ZOLTÁN – KISS GYÖRGY – KRAFFT WALTER

SCHILL RÓBERT – SZEGEDI JÁNOS

MISAM 2.00

Kiegészítés

Akadémiai Kiadó, Budapest 1989

GONDOLA — KISS — KRAFFT — SCHILL — SZEGEDI

MISAM 2.00

Kiegészítés

Szoftver — dokumentáció
Sorozatszerkesztő: Pintér Tibor
A kiadványt szakmailag ellenőrizte: Juhász Lehel

A szoftver — dokumentáció a MATEX programcsomaggal készült

1. fejezet

Tartalomjegyzék

1. Bevezetés 5

1.1. A módosítások áttekintése 5

1.1.1. Módosítások a MISAM futtatórendszerében 6

1.1.2. A segédprogramok módosításai 6

2. A futtatórendszer módosításai 8

2.1. Többféle C környezet támogatása 8

2.2. Egyedi kulcsértékek kikényszerítése 9

2.3. Rendezettség indexdarabonkénti megadása 10

2.4. Karakteres kulcsdarabok hasonlítása 11

2.5. Az aktuális pozíci kezelése 12

2.6. Tranzakció kezelés 12

3. A MISAM futtatórendszer új függvényei 15

3.1. SBEGTRANS 15

3.2. ISBREAKTRANS 17

3.3. ISBUILDRANGE 18

3.4. ISENDTRANS 19

3.5. ISFREEFDF 21

3.6. ISGETPOS 22
3.7. ISSETCMP 24
3.8. ISSETPOS 26

4. A segédprogramok módosításai 28
4.1. FDFED: interaktív .FDF file editor 28
4.1.1. A program funkciója 28
4.1.2. A program indítása és opciói 28
4.1.3. A kezelői párbeszéd 29
4.2. TRANS: tranzakció-befejező program 29
4.2.1. A program funkciója 29
4.2.2. A program indítása és opciói 30
4.2.3. A kezelői párbeszéd 30
4.3. QUERY: lekérdező és módosító 30
4.3.1. A program indítása és opciói 31
4.4. LIST: állomány listázó program 31
4.4.1. A program indítása és opciói 32
4.5. FDFTOREP: lista leíró generátor 33
4.5.1. A program funkciója 33
4.5.2. A program indítása és opciói 33

5. Függelék 34
5.1. A lista leíró file 34
5.1.1. Az elemi változók megadása 35
5.1.2. Listafejléc megadása 38
5.1.3. Az elemi sorok megadása 38
5.1.4. Összegző sorok kiíratása 39
5.2. A query leíró file 40

1. fejezet

Bevezetés

A MISAM rendszer 2.00 változata az 1988-ban készült 1.00 változat javított és továbbfejlesztett kiadása. A módosítások részben az 1.00 változatban talált hibák javításai, részben olyan fejlesztések, amelyek lehetővé teszik a rendszer szélesebb körű és hatékonyabb felhasználását.

A jelen dokumentáció az 1.00 változathoz készített leírás kiegészítése. A rendszernek csak azokat a részeit tárgyalja, amelyek megváltoztak vagy újak a 2.00 változatban. A MISAM 2.00 rendszer használatához ezért szükséges a 1.00 felhasználói leírása is.

1.1. A módosítások áttekintése

A MISAM rendszeren elvégzett változtatások érintik mind a MISAM könyvtárakat, mind a MISAM segédprogramokat. A könyvtárakon végzett módosítások biztosítják, hogy amit az 1.00-val létrehoztunk, az változtatás nélkül használható a 2.00-val is. A segédprogramok közül néhány megszűnt, illetve megváltozott a használatuk és vannak újak is, amelyek a 2.00 tágabb lehetőségei miatt készültek el. Az eltérő használatból származó kényelmet-

lenségeket a segédprogramok által nyújtott tágabb lehetőségek kompenzálhatják.

1.1.1. Módosítások a MISAM futtatórendszerében

- Többféle C környezet támogatása:
  - MSC 5.1 változat,
  - Quick C,
  - Turbo C 2.00 változat,
  - Aztec C 3.20 változat.
- Egyedi kulcsértékek kikényszerítése.
- Indexdarabonkénti rendezettség (csökkenő-növekvő) megadása.
- Karakter típusú kulcsdarabok tábla alapján történő hasonlítása.
- Új függvények az aktuális pozíció kezelésére.
- Tranzakció kezelés.

1.1.2. A segédprogramok módosításai

- Új segédprogram az .FDF file-ok interaktív definiálásához (FDFED.EXE).
- Új segédprogram a befejezetlen tranzakciók kezelésére (TRANS.EXE).
- A FED program kibővült a futtató rendszer által nyújtott új lehetőségekkel.
- A QUERY program paraméterezése kibővült egy query leíró file-lal. Ezáltal az információ több állományból is összeválogatható.

- A LIST program paraméterlistájában szintén megadható egy query leíró file. Ezáltal a lista is összeválogatható több állományból.
- A LIST programhoz tartozó lista leíró file formája változott. Az eddigi .RPD és .RPF file-ok helyett egy .REP állománnyal lehet leírni a lista formáját.
- Az FDFTORPD és FDFTORPF programok funkcióját az FDFTOREP program vette át.

2. fejezet

A futtatórendszer módosításai

2.1. Többféle C környezet támogatása

A MISAM 1.00 változat csak az MSC 4.0 fordítóprogramot támogatta. A 2.00 változat valamennyi Magyarországon gyakran használt C fordítóhoz elérhető:

- MSC 5.1 változat,
- Quick C,
- Turbo C 2.00 változat,
- Aztec C 3.20 változat.

A programozói kapcsolat mind a négy C rendszerben azonos, így a MISAM rendszer szempontjából a kifejlesztett programok korlátozás nélkül átvihetők egyik C rendszerből a másikba. Az egyetlen megkötés az, hogy a rekordbuffereknek tömörítettnek kell lenniük, vagyis a megfelelő fordítóprogram opcióval felül kell bírálni a struktúraelemek a hardver számára optimális határra igazítását. (Pl. az MSC 5.1 -Zp opciója.)

Az 1.00 változattal kifejlesztett programok a MISAM szempontjából változtatás nélkül vihetők át bármelyik 2.00 változat alá.

A MISAM rendszer a négy C környezet mindegyikében a small, medium, compact és large könyvtárverziókat szolgáltatja.

A könyvtárnevek (méret jelölése) mindig az adott C környezetnek megfelelő névkonvenciót követik. (Pl. SMISAM.LIB illetve MISAMS.LIB az MSC, illetve a Turbo C környezetekben.)

2.2. Egyedi kulcsértékek kikényszerítése

Az 1.00 változat a kulcsértékekre vonatkozóan semmiféle vizsgálatot nem végzett. Így esetenként a felhasználónak kellet az írás és felülírás (**iswrite()**, ill. **isupdate()**) előtt a kulcsérték egyediségéről meggyőződnie.

A 2.00 változatban az indexleírásánál lehetőség van annak jelölésére, hogy az indexfile csak egyedi kulcsértékeket tartalmazhat. Az **iswrite()** és **isupdate()** függvények az E_DUPLICATE hibakóddal térnek vissza, ha valamelyik egyedinek deklarált indexfileban az adatrekordból generált kulcsérték már létezik.

Az indexfile-építő függvények (**isbuild()** és **isbuildrange()**) nem vizsgálják a kulcsértékek egyediségét.

A kulcsérték egyediségét az index hivatkozási neve után írt, attól vesszővel elválasztott UNIQ kulcsszóval jelölhetjük.

Példa:

```
##################################################
# Index definicio
#
1
FOKONYV_I1,UNIQ
c:\mikridat\fo001.ndx
Kulcs a fokonyvi szam szerint
1
0       15       A
```

A példában az index hivatkozási neve: FOKONYV_I1. Az utána következő UNIQ kulcsszó azt jelöli, hogy a főkönyvi törzsállományban egy főkönyvi szám csak egyszer szerepelhet.

2.3. Rendezettség indexdarabonkénti megadása

Az 1.00 változat csak a növekvő rendezettséget ismerte. Előfordulhatnak azonban olyan (elsősorban listázási, lekérdezési) feladatok, amikor nem kényelmes a növekvő rendzettség.

A 2.00 változatban lehetőség van a rendezettség kulcsdarabonkénti meghatározására, melyet a .FDF file-ban kell leírni. A korábbi változattal való formai kompatibilitás miatt nem a kulcsdarabok leírását bővítettük, hanem a kulcsdarab típusmegadásánál megkülönböztetjük a kis- és nagybetűt. A kisbetűs típus csökkenő, a nagybetűs típus növekvő rendezettséget jelent.

Fontos: Ha egy indexfile rendezettségét megváltoztatjuk, akkor az adatállomány minden további használata előtt újra kell építeni az adott indexfile-t.

Példa:

```
################################################
# Index definicio
1
FOKONYV_I2
c:\mikridat\fo002.ndx
Kulcs bizonylatszam +, datum szerint -
2
0        15        A
52        8        m
```

A példában az index két darabból áll: a bizonylatszámon belül növekvő sorrendben dátum szerint csökkenő sorrendben rendezett. Így egy bizonylatszámhoz először a legutóbbi, majd az egyel korábbi stb. rekordot kapjuk meg.

2.4. Karakteres kulcsdarabok hasonlítása

Az 1.00 változat a karakteres kulcsdarabokat (B,S,C,A típusok) csak a standard ASCII kódtábla alapján tudta hasonlítani. Ez kényelmetlenségeket okozhatott, ha nem volt szükség pl. a kis- és nagybetűk megkülönböztetésére, vagy ékezetes neveket akartunk écé sorrendben elérni.

Az új **issetcmp()** függvénnyel (lásd később) lehetőség van tetszőleges hasonlítási tábla megadására. A táblát indexfile-onként, nyitás után kell megadni. Természetesen a tábla megadása önmagában nem változtatja meg a rendezettséget. Ha különleges rendezettséget kívánunk elérni, akkor a megfelelő hasonlító táblát az írások során is következetesen kell használni.

A hasonlító tábla egy 256 elemű karakter tömb. A tömb i-edik eleme megmondja, hogy az ASCII kódtábla i-edik karaktere az új sorrendben a hányadik helyet foglalja el.

Példa:

```
unsigned char    tabla[] = "
 /*0:      */      0,
        . . .
 /*97:    a*/     97,
        . . .
 /*160:   á*/     97,
        . . .
 /*255:     */   255";
```

A példa egy hasonlító tábla részletét mutatja be. A tábla alapján nem teszünk különbséget a 160-as (a képernyőn „á" betűként megjelenő) karakter és a kis „a" betű között.

A hasonlító tábla használatának részletes ismertetése az **issetcmp()** függvény leírásánál található.

2.5. Az aktuális pozíció kezelése

A törlés és felülírás műveletek (**isdelete()** és **isupdate()** függvények) mindig a kiválasztott index aktuális pozíciója alapján meghatározott rekordra vonatkoznak. Az aktuális pozíciót az **isread()** függvény állítja be. Ha egy rekord olvasása és felülírása között pozícionálnunk kellett az állományban, akkor az eredeti rekordra való visszaállás bonyolult feladatot jelentett.

Az új **isgetpos()** függvénnyel elmenthetjük, az **issetpos()** függvénnyel pedig beállíthatjuk egy indexfile aktuális pozícióját. Ezzel az előbbi kényelmetlen helyzet kiküszöbölhető.

Az aktuális pozíció kezelése az **isgetpos()** és **issetpos()** függvények leírásánál található.

2.6. Tranzakció kezelés

Az 1.00 változat nem nyújtott semmiféle adatvédelmet hardver- vagy szoftver-hibák (pl. áramkimaradás, kevés memória stb) esetére. Emiatt nehezen lehetett megoldani több állomány biztonságos párhuzamos karbantartását. (Hiba esetén nagy valószínűséggel megbomlott az adatok következetessége.)

A 2.00 változatban az írási kérelmeket tranzakciókba foghatjuk össze. A MISAM rendszer biztosítani tudja, hogy egy tranzakció összes művelete ténylegesen kikerül az állományba.

A tranzakciót **isbegtrans()** függvénnyel indíthatjuk el. Egy alkalmazás egyszerre csak egy tranzakciót futtathat. Az **isbreaktrans()** függvénnyel megszakíthatjuk a tranzakciót, az állományok változatlanul maradnak. Az **isendtrans()** függvény a tranzakció összes írási kérését kiírja a megfelelő állományba.

A tranzakció kezdete előtt az összes, a tranzakcióban szereplő állományt kizárólagos joggal zárni kell (write lock), és ezt csak a tranzakció vége után szabad elengedni (**isbreaktrans()** vagy **isendtrans()**). A tranzakcióban minden, a tranzakció kezdete és vége között érkezett írási kérés automatikusan részt vesz:

- **iswrite()**
- **isupdate()**
- **isdelete()**
- **isclose()**

Tranzakció közben tilos indexfile-t építeni (**isbuild()** vagy **isbuildrange()**), vagy MISAM leíró blokknak allokált helyet felszabadítani.

Mivel a tranzakció közben fogni kell az állományokat, ezért a tranzakciót a lehető legrövidebbre kell tervezni. A tranzakciók méretének a szabad memória is korlátot szabhat. Ha tranzakció közben valamelyik írási művelet E_MEMORY hibakóddal tér vissza, akkor valószínűleg túl nagy a tranzakció.

Ha az **isendtrans()** függvény hívása előtt kapunk hibaüzenetet, akkor az állományok a tranzakció kezdete előtti változatlan állapotban maradnak. Ha az **isendtrans()** függvény hívása közben keletkezik hiba, akkor a program leállítása után a TRANS segédprogrammal be kell fejezni a tranzakciót. Amíg a tranzakciót nem sikerült befejezni, addig tilos az állományok bármilyen használata.

Hálózatos környezetben az állományok elengedése nélkül kell kiszállni az alkalmazásból. Ezzel biztosítani tudjuk azt, hogy a tran-

zakció sikeres befejezéséig más munkaállomás nem fér hozzá az állományokhoz. A TRANS segédprogram az állományokat a tranzakció sikeres befejezése után elengedi.

A tranzakciókat kezelő függvényeket a 3. fejezet ismerteti.

3. fejezet

A MISAM futtatórendszer új függvényei

3.1. ISBEGTRANS

Funkció: Tranzakció kezdete

Használat:

int **isbegtrans** (*name*)

char * *name* ; köztes file neve

Leírás: Az **isbegtrans()** függvénnyel egy tranzakció kezdetét jelölhetjük. A tranzakció kezdete után a tranzakció befejezéséig bármely állományra kiadott írási kérés eleme a tranzakciónak. A tranzakcióban résztvevő állományokat a tranzakció kezdete előtt írásra kell zárni és azt nem szabad a tranzakció végéig feloldani.

A *name* paraméter egy köztes file neve. Munkaállomásonként egyedinek kell lennie és nem létezhet az **isbegtrans()** függvény hívásakor. A tranzakció megszakítása vagy a sikeres befejezés törli a köztes file-t.

Egy programban egyszerre csak egy tranzakció lehet aktív. Hiba az **isbegtrans()** függvénynek a korábbi tranzakció befejezése előtti ismételt hívása.

Visszatérés: SUCCESS — sikeres művelet
E_TRSTARTED — már van aktív tranzakció
E_IO — hiba a köztes file létrehozása közben

Lásd még: **isbreaktrans()** , **isendtrans()** , **iswlock()** , **iswunlock()**

Példa: Tranzakció indítása konkurrens környezetben

```
#include <MISAM.H>

    .

    .

struct FILEDESC     *fdp;
int                 rc;

    .

    .

if ((rc = iswlock(fdp)) != SUCCESS)
  {
    printf("Foglalt allomany !\n");
    return(-1);
  }
if ((rc = isbegtrans("MYTRANS.TRA")) != SUCCESS)
  {
    iswunlock(fdp);
    printf("Hibas tranzakcio kezdes !\n");
    return(-1);
  }
```

3.2. ISBREAKTRANS

Funkció: Tranzakció megszakítása

Használat: int **isbreaktrans()**

Leírás: Az **isbreaktrans()** függvénnyel egy tranzakciót szakíthatunk meg anélkül, hogy az állományokba kiírnánk a változtatásokat. Az **isbreaktrans()** hívás után az állományok állapota az **isbegtrans()** előtti állapot lesz.

Visszatérés: SUCCESS — sikeres művelet
E_IO — hiba a köztes file törlése közben
E_NOTRANS — nincs aktív tranzakció
E_BADFORM — sérült állomány

Lásd még: **isbegtrans()** , **isendtrans()** , **iswlock()** , **iswunlock()**

Példa: Az előző tranzakció megszakítása

```
#include <MISAM.H>
struct FILEDESC     *fdp;
int                 rc;

    .
    .
    .
rc = isbreaktrans();
iswunlock(fdp);
if (rc != SUCCESS)
{
    printf("Hibas tranzakcio megszakitas!
    \n");
    return(-1);
}
```

3.3. ISBUILDRANGE

Funkció: Részleges indexfile építése

Használat:

int **isbuildrange**(*fdp*, *ixname*, *start*, *end*)
struct FILEDESC **fdp* ; MISAM leíró blokk
char **ixname* ; index hiv. név
char **start* ; kezdő kulcs
char **end* ; végső kulcs

Leírás: Az **isbuildrange()** függvénynek az **isbuild()** függvényhez hasonló a funkciója, a felépített indexfile azonban csak a megadott intervallumba eső

kulcsértékeket taralmazza. Így ezen az indexfile-on keresztül csak az adatállomány egy részét érhetjük el. Előnye az **isbuild()** függvénnyel szemben, hogy — a megadott intervallumtól függően — kisebb az indexfile helyigénye és rövidebb a felépítési idő. Különösen ez utóbbi tényező indokolhatja az **isbuildrange()** függvény használatát, elsősorban batch jellegű feldolgozások során (pl. listázás).

fdp egy nyitott MISAM file leíró blokkjának mutatója. *ixname* a felépítendő index hivatkozási neve. A felépítendő index leírását előzőleg be kell szúrni a MISAM file leírásába. Ha az *ixname* paraméter NULL, akkor az építés a kiválasztott (preferált) indexre vonatkozik.

start a kezdő kulcsértékre mutat, *end* pedig a végértékre.

Visszatérés: SUCCESS — sikeres művelet
E_IO — írási vagy olvasási hiba
E_BADARG — hibás index hivatkozási név
E_MEMORY — nincs elég szabad memória

Lásd még: **isbuild()** , **isbegtrans()**

Példa: Lásd az **isbuild()** függvény leírásánál

3.4. ISENDTRANS

Funkció: Tranzakció vége

Használat: int **isendtrans()**

Leírás: Az **isendtrans()** függvény lezárja a korábban **isbegtrans()** függvénnyel elindított tranzakciót. A

tranzakció zárás során a köztes file-ból minden információ átkerül a megfelelő adatállományba, a köztes file törlődik.

Ha az **isendtrans()** függvény hibakóddal tér vissza, akkor valószínűleg csak az információ egy része került át az állományokba. Ilyen esetben az írási zárások feloldása nélkül kell kiszállni az alkalmazásból és minden más hozzáférés előtt le kell futtatni a TRANS segédprogramot, paraméterként a tranzakció köztes file-jának nevével. A TRANS segédprogram sikeres lefutása után az állományok tartalma helyes lesz és hozzáférhető további felhasználásra.

Ha a TRANS program sem tudja sikeresen befejezni a tranzakciót, akkor a kapott hibaüzenettől függően kell folytatni a munkát. Erről részletesebb információt a program leírásánál találhatunk.

Ha tranzakció közben súlyos hiba történt (pl. áramkimaradás), akkor lehet, hogy a tranzakció köztes file-ja nem törlődött. Ha a köztes file 0 hosszúságú, akkor töröljük a file-t és folytassuk a feldolgozást. Ellenkező esetben futassuk le a TRANS segédprogramot.

Visszatérés: SUCCESS — sikeres művelet
E_IO — írási vagy olvasási hiba
E_NOTRANS — nincs aktív tranzakció

Lásd még: **isbegtrans(), isbreaktrans(), iswlock(), iswunlock()**

Példa: Tranzakció befejezése konkurrens környezetben

```
#include <MISAM.H>
    .
    .
    .
struct FILEDESC     *fdp;
int                 rc;
    .
    .
    .
rc = isbreaktrans();
if ((rc!=SUCCESS) && (rc!=E_NOTRANS))
{
    printf("Hibas tranzakciozaras   !\n");
    printf("Futtassa a TRANS programot !\n");
    return(-1);
}
iswunlock(fdp);
```

3.5. ISFREEFDF

Funkció: MISAM leíró blokk felszabadítása

Használat:

int **isbuildrange**(*fdp*) ,
struct FILEDESC * *fdp* ; MISAM leíró blokk

Leírás: Az **isfreefdf()** függvény felszabadítja az **isgetfdf()** függvény által a MISAM leíró blokk

számára allokált memóriát. A MISAM leíró blokknak szbadnak kell lennie, vagyis:
- a file nem lehet nyitva,
- nem lehet eleme aktív tranzakciónak,
- az alkalmazás által nem lehet sem írásra, sem olvasásra foglalt.

Ha a leíró blokkot felszabadítottuk, akkor a filehoz történő minden hozzáférést meg kell, hogy előzzön egy újabb **isgetfdf()** függvényhívás.

Visszatérés: SUCCESS — sikeres művelet
Lásd még: **isgetfdf()**
Példa:

```
#include <MISAM.H>
struct FILEDESC        *fdp;

    .
    .

isgetfdf("PELDA.FDF",&fdp);

    .
    .

isfreefdf(fdp);
```

3.6. ISGETPOS

Funkció: Aktuális pozíció elmentése
Használat:

int **isgetpos(** *fdp, ixname, buff* **)**
struct FILEDESC **fdp* ; MISAM leíró blokk
char **ixname* ; index hiv. név
char **buff* ; output buffer

Leírás: Az **isgetpos()** függvénnyel elmenthetjük egy indexfile aktuális pozícióját. *fdp* egy nyitott MISAM file leíró blokkjának mutatója. ixname egy index hivatkozási neve. Lehet NULL, akkor a rendszer a kiválasztott (preferált) index aktuális pozícióját menti el.

buff a felhasználó által a pozíció elmentésére biztosított terület címe. Mérete a kulcs mérete + 9 byte. A függvény ebbe a bufferbe teszi az aktuális pozíciót leíró karaktersorozatot. A bufferben a pozíció a MISAM belső adminisztrációjának megfelelően tárolódik.

Visszatérés: SUCCESS — sikeres művelet

E_BADARG — hibás index hivatkozási név

E_TOPKEY — az aktuális pozíció a file eleje pozíció

E_ENDKEY — az aktuális pozíció a file vége pozíció

Lásd még: **issetpos() , isupdate() , isdelete() , isread()**

Példa: A kiválasztott index aktuáklis pozíciójának mentése. A kulcs hosszát a felhasználó által írt " keylen() " függvény számítja ki.

```
#include <MISAM.H>
struct FILEDESC     *fdp;
char                *buff;
int                 rc;
    .
    .
    .
buff = malloc(keylen(fdp,NULL)+9);
if (buff == NULL)
    return(E_MEMORY);
rc = isgetpos(fdp,NULL,buff)
```

3.7. ISSETCMP

Funkció: Hasonlító tábla megadása

Használat:

int **issetcmp**(*fdp, ixname, cmptab*)
struct FILEDESC **fdp* ; MISAM leíró blokk
char **ixname* ; index hiv. név
char **cmptab* ; a hasonlító tábla

Leírás: Az **issetcmp()** függvénnyel az indexek karakteres, byte és előjel nélküli byte (C,A,B,S) típusú darabjainak, az ASCII kódtáblától eltérő hasonlítási sorrendjét adhatjuk meg. A tábla egy 256 elemű karakter tömb. A tábla i-edik eleme megmondja, hogy az ASCII kódtábla i-edik eleme az új

hasonlítási sorrendben hányadik helyen szerepel. A hasonlító tábla megadásával helyesen kezelhetjük bármely ábécé szavait, figyelmen kívül hagyhatjuk a kis- és nagybetűk közötti különbséget stb.

A hasonlító táblát a file megnyitása után minden esetben be kell állítani. Ha az ASCII kódtáblától eltérő rendezettséget kívánunk elérni, akkor természetesen a file-ba írások során is használni kell a megfelelő hasonlító táblát.

A hasonlító tábla indexfile-onként készül és az index minden C,A,B vagy S típusú darabjára vonatkozik.

A különleges rendezettségű indexfile-okat a COMPRESS és REBUILD programok használata után a felhasználónak – a megfelelő hasonlító tábla szerint – újra kell építenie.

fdp egy nyitott MISAM file leíró blokkjának mutatója. *ixname* index hivatkozási név. Lehet NULL, akkor a függvényhívás a kiválasztott (preferált) indexre vonatkozik.

Visszatérés: SUCCESS — sikeres művelet
E_BADARG — hibás index hivatkozási név
E_MEMORY — nincs elég szabad memória

Lásd még: **isselind()**

Példa: A példát az előző fejezet vonatkozó pontja tartalmazza.

3.8. ISSETPOS

Funkció: Aktuális pozíció állítása

Használat:

int **issetpos**(*fdp, ixname, buff*)
struct FILEDESC **fdp* ; MISAM leíró blokk
char **ixname* ; index hiv. név
char **buff* ; az új pozíció

Leírás: Az **issetpos()** függvény egy korábban az **isgetpos()** függvénnyáltal elmentett pozíciót állíthatunk vissza. A pozíció beállítása után a file-ban szekvenciális mozgás ettől a pozíciótól értelmeződik, a törlés és felülírás az új pozíció által meghatározott rekordra vonatkozik.

Az **isgetpos()** , **issetpos()** párossal a file újranyitása után is be tudjuk állítani a zárás előtt elmentett pozíciót.

fdp egy nyitott MISAM file leíró blokkjának mutatója. *ixname* index hivatkozási név. Lehet NULL, akkor a függvényhívás a kiválasztott (preferált) indexre vonatkozik. *buff* az **isgetpos()** hívásnál használt buffer címe.

Visszatérés: SUCCESS — sikeres művelet
E_BADARG — hibás index hivatkozási név

Lásd még: **isgetpos(), isread(), isdelete(), isupdate(), isopen()**

Példa: Az **isgetpos()** függvénye elmentett pozíció visszaállítása.

```
#include <MISAM.H>
struct FILEDESC     *fdp;
char                *buff;
int                 rc;
    .
    .
    .
rc = issetpos(fdp,NULL,buff)
```

4. fejezet

A segédprogramok módosításai

4.1. FDFED: interaktív .FDF file editor

4.1.1. A program funkciója

A programmal interaktív módon tudjuk létrehozni, módosítani a MISAM állományokat leíró .FDF file-okat.

4.1.2. A program indítása és opciói

Indítása:

`FDFED [-h] [-p` *path* `] <.FDF` *file neve* `>`

Paraméterek:

A .FDF *file neve* paraméter kötelező, megadja a MISAM állományt leíró file-t. A kiterjesztést nem kell megadni, azt a program automatikusan ajánlja. Ha a leíró file már létezik, akkor a program módosítja, egyébként létrehozza azt.

A programot a `-h` opcióval indítva megkapjuk az indítási parancssort.

A `-p` *path* opcióval azt tudjuk megmondani a programnak, hogy a leíró file-ból generált .H header file-okat melyik könyvtárba tegye.

Ha az opciót elhagyjuk, akkor a .H file-ok az aktuális könyvtárba íródnak ki.

4.1.3. A kezelői párbeszéd

A program menüszerű képernyőkön ajánlja fel a választási lehetőségeket. A lehetőségek között a nyíl-billentyűkkel mozoghatunk, a kiválasztás a RETURN (ENTER) billentyű leütésével történik.

A programot az ESC billentyű leütésével hagyhatjuk el. Ha a munkát nem mentettük le, akkor kiszállás előtt a program felajánlja a lementést. Az ALT-W billentyűvel file-ba írhatjuk a leírást, az ALT-H billentyűvel pedig .H header file-t készíthetünk.

A program kezelését minden szinten az F1 billentyűvel hívható on-line help rendszer segíti.

4.2. TRANS: tranzakció-befejező program

4.2.1. A program funkciója

A program segítségével a félbemaradt tranzakciókat fejezhetjük be. Ha egy program tranzakció zárás közben hibát jelez, akkor a tranzakcióban részt vett á llományokhoz a TRANS program sikeres futtatása előtt mindenféle hozzáférés tilos.

Ha a TRANS program a tranzakció befejezése közben csak az indexfile-okra ad hibaüzenetet, akkor a futás után a REBUILD programmal újra kell építeni az indexfile-okat.

4.2.2. A program indítása és opciói

Indítása:

TRANS [-h] [-u] < *tranzakciós file neve* > **Paraméterek:**

A *tranzakciós file neve* paraméter az **isbegtrans()** függvény paraméteréül adott file-név, teljes ösvénnyel és kiterjesztéssel.

A -h opció hatására a program kiírja a képernyőre az indítási parancssort.

A program csak abban az esetben szabadítja fel az állományokon levő írási zárást, ha a futás sikeres volt. Az -u opcióval hiba esetén is ki tudjuk kényszeríteni az állományok felszabadítását.

4.2.3. A kezelői párbeszéd

A program nem vár adatokat az operátortól. A feldolgozás során a standard outputra írja a tranzakció befejezése közben előforduló hibákat. Ha a futás sikeres volt, akkor a program eltörli a tranzakció köztes file-ját és felszabadítja az állományokat. Ha hibaüzenetet kapunk, akkor a file-t a felhasználónak kell eltörölni.

4.3. QUERY: lekérdező és módosító

A QUERY programról itt csak a módosított indítási parancssort ismertetjük. Az operátori kapcsolat az előző változathoz képest nem változott, a program kezelését az F1 billentyűvel hívható on-line help rendszer segíti.

4.3.1. A program indítása és opciói

Indítása:

`QUERY [-h] [-m] [-d] <-q .QRF> [-f .FDF] [-w` *könyvtár*`]`

Paraméterek:

A programot a `-h` opcióval indítva a képernyőn megkapjuk az indítási parancssort az opciók rövid leírásával együtt.

A QUERY program alapértelmezésben csak lekérdezésre használható, rekord törlésre vagy módosításra nem. A `-m` opció hatására a program engedi a rekordok módosítását, a `-d` opcióra pedig a törlését. Ezáltal nagyobb alkalmazásokba szelektíven tudjuk beépíteni a QUERY-t, a hozzáférési jogoktól függően.

A `-q .QRF` paraméter egy ún. query leíró file-t ad meg. A kiterjesztés megadása nem kötelező. A query leíró file-t a függelék második szakasza ismerteti.

A `-q .QRF` paraméter helyett használhatjuk a `-f .FDF` paramétert is, a kettő egymást kizárja. A `-f .FDF` egyszerűbb lekérdezést valósít meg: a lekérdezés csak egy MISAM állományra vonatkozik és a lekérdezési szempontokat is mindig manuálisan meg kell adni.

A QUERY program köztes állományokat épít a lekérdezés során. Az állományok alapértelmezés szerint az aktuális könyvtárban képződnek, de a `-w` *könyvtár* opcióval megadhatjuk a temporális file-ok helyét tetszés szerint más könyvtárban is.

4.4. LIST: állomány listázó program

A LIST programról itt csak a módosított indítási parancssort ismertetjük. A query file használatát és a lista leíró file új formáját a függelék tartalmazza.

A LIST program új változata új szűrőfile-okat használ, ezért a korábbi változattal kapott szűrőfile-okat cseréljük le az új szűrőfile-okra.

4.4.1. A program indítása és opciói

Indítása:

`LIST [-h] <-r .REP> <-q .QRF> [-f .FDF] [-w`*könyvtár*`]`
`[-c .CNV] [-o` *output* `]`

Paraméterek:

A programot a `-h` opcióval indítva a képernyőn megkapjuk a indítási parancssort az opciók rövid leírásával együtt.

Az `-r .REP` paraméter a lista leíró file-t adja meg. A kiterjesztést elhagyhatjuk. Az új lista leíró file felépítését a függelék tartalmazza.

A `-q .QRF` paraméter egy query leíró file-t ad meg. (A query leíró file felépítését a függelék tartalmazza.) Használhatjuk helyette az `-f .FDF` paramétert, ami egy MISAM file leírót határoz meg. Az utóbbi esetben a listázás egy MISAM állományra korlátozódik. Amennyiben query leíró file-t használunk, úgy a query file-ban definiált rekordbuffernek meg kell egyeznie a lista leíró file-ban definiált rekordbufferrel.

A `-w` *könyvtár* opció a listázás során keletkező köztes állományok munkakönyvtárát adja meg. Alapértelmezés az aktuális könyvtár.

A `-c .CNV` opció egy szűrőfile-t határoz meg. (A kiterjesztést nem kötelező megadni.) Amennyiben megadjuk, felülbíráljuk a lista leíró file-ban megadott értéket. Ha nem adjuk meg, akkor a program az aktuális könyvtárban keresi a lista leíró file-ban megadott típusú nyomtatónak megfelelő szűrőfile-t. A szűrőfile-oknak tartalmaznia kell, hogy a képernyő ékezetes betűit milyen karaktersorozat kiírásával jeleníthetjük meg a nyomtatón, illetve milyen karaktersorozattal lehet a nyomtatási módot változtatni. A

szűrőfile-ok készítéséhez a disztributív anyagban adott szűrőfile-ok adnak segítséget.

Az -o *output* opcióval felülbírálhatjuk a lista leíró file-ban megadott *output* file-t. *output* értéke lehet: LOCALPRINTER (lokális nyomtató), GLOBALPRINTER (hálózati nyomtató), vagy egy file-név. (Hálózati nyomtató esetén a megfelelő file server és nyomtató kiválasztásáról a felhasználónak kell gondoskodni.)

4.5. FDFTOREP: lista leíró generátor

4.5.1. A program funkciója

A program a korábbi FDFTORPD és FDFTORPF programok funkcióját látja el, de a lista leíró file új formájának megfelelően: a MISAM állományt leíró file-ból lista leíró file-t generál.

4.5.2. A program indítása és opciói

Indítása:

`FDFTOREP [-h] <.FDF` *file* `> [-o` *header file* `]`

Paraméterek:

A programot a -h opcióval indítva a képernyőn megkapjuk az indítási parancssor formáját.

A .FDF file paraméter egy MISAM állományt leíró file, a kiterjesztést elhagyhatjuk.

A *header file* neve a .FDF file nevéből készül, az aktuális könyvtárba, .Rh kiterjesztéssel. Az -o *header file* opcióval megadhatjuk a header file nevét a teljes ösvénnyel és a kiterjesztéssel együtt.

5. fejezet

Függelék

5.1. A lista leíró file

A listákat .REP kiterjesztésű lista leíró file-okban írhatjuk le. A leíró file-ban kell megadni:

- az összes, potenciálisan kiírható változó nevét és jellemzőit,
- a fejlécek szövegét és változóit,
- a részletsorok szövegét és változóit,
- az összegfokozatok ismérveit,
- a listafile nevét,
- a nyomtató típusát,
- a nyomtató írásmódját,
- a dátum típusát,
- az egy lapra írandó sorok számát.

A lista leíró file-t tetszés szerinti szövegszerkesztő programmal létrehozhatjuk, vagy generálhatjuk az FDFTOREP programmal.

A lista leíró file a következő definíciós egységekből áll:

- elemi változók deklarálása,
- listázási paraméterek megadása,
- fejléc formátumának leírása,
- részletsor formátumának megadása,
- összegfokozatok definiálása.

A felsorolás egyben megadja a definíciós egységek előfordulásának sorrendjét is.

5.1.1. Az elemi változók megadása

Mindazokat a mezőket, amelyeket a listában ki akarunk íratni (az ún. report buffer), szerepeltetni kell az elemi mezőleírások között:

DETAILFIELDS=az elemi mezők száma

FIELD=név,offset,hossz,dimenzió,típus,mezőszélesség,dec, igazítás.

- A név egy max. 16 karakteres azonosító név.
- A relatív kezdőpozíció (offset) megadja, hogy az első mező kezdetétől hány byte-nyira kezdődik az aktuális mező (az előző mezők hosszának összege).
- A hossz értéke karakteres (C vagy A) mezőknél a karakterek számát, egyéb típusoknál pedig a belső ábrázolás hosszát adja meg. Ezek típusonként a következők:
  - int (I) és unsigned int (S) típusoknál 2 byte
  - long int (L), unsigned long int (U), date (M) és float (F) típusoknál 4 byte
  - double (D) esetén pedig 8 byte.
- A dimenzió jelenleg nem használt.
- A mező lehetséges típusait az előzőekben soroltuk fel.
- A mezőszélesség a kiírási pozíciók számát adja meg, tehát az adott változó értéke ennyi pozíción fog kiíródni.
- A következő számjegy float (F) vagy double (D) típusú mezőknél a decimális jegyek számát jelenti, egyéb típusoknál nincs jelentősége.
- Minden kiíratandó változóra megadhatunk egy mezőn belüli igazítást. Ez azt jelenti, hogy ha a mezőben balra akarjuk igazítani a kiírást, akkor L-t, ha jobbra, akkor R-t kell megadnunk.

A változókra a hivatkozás a változó névvel, ill. a név elé és után ilesztett „@" karakterrel lehet hivatkozni.

Az FDFTOREP programmal készített lista-leírásban a formátumokat és összegfokozatokat tetszés szerint változtathatjuk, de a report buffer megváltoztatása tilos.

Az elemi változókon kívül a formátumok megadásakor három belső változóra is hivatkozhatunk:

- pageno (aktuális lapszám),
- lineno (aktuális sorszám),
- date (aktuális dátum).

A rendszerváltozókra a névvel, valamint a név elé és után illesztett „#" karakterrel tudunk hivatkozni (# pageno#).

A listázási paraméterek megadása

LISTFILE=file

file lehet LOCALPRINTER, GLOBALPRINTER, vagy az MS_DOS szabályainak megfelelő file-név.
LOCALPRINTER a munkaállomáshoz, GLOBALPRINTER a file serverhez csatolt nyomtatót jelenti.

PRINTMODE=NORMAL,ELITE,CONDENSED

A nyomtatás sűrűségét szabályozhatjuk. Általában a NORMAL 80 karakter/sor, az ELITE 96 karakter/sor, a CONDENSED 120 karakter/sor sűrűségű kiírást eredményez, de ez nyomtató függő.

PAGELENGTH=laphossz

Az egy lapra kerülő sorok száma alapértelmezésben 65. Ha ez nem megfelelő, akkor ezzel a paranccsal meg lehet változtatni az értéket.

PRINTERTYPE=WALTERS,EPSON,...

A nyomtató típusát meg kell adni a szűrőfile azonosításához.

OPEN= A,W

Lehetőség van arra, hogy egy file-ba írt listát „folytassunk", azaz a következő listázás ne felülírja az előzőleg írtat, hanem mögé írjon. Ebben az esetben az OPEN=A (append) parancsot kell meg adnunk. Az alapértelmezés az OPEN=W (write), azaz a felülírás.

DATETYPE=HUN,ENG,GER

A dátum (M) típusú változók kiírási formátumát szabályozhatjuk. Ez lehet magyar formátumú (év /hónap /nap), angol formátumú (hónap /nap /év) vagy német formátumú (nap /hónap /év). Az alapértelmezett a magyar kiírási forma.

DETAILPRINT=YES,NO

Szükségünk lehet arra, hogy az összegző sorokat az elemi sorok vélkül írassuk ki. Az elemi sorok kiírását a DETAILPRINT=NO paranccsal tilthatjuk le. Az alpértelmezés szerint kiíródnak.

A formátumok megadásának szabályai

Minden kiírást (fejléc, elemi sor, összegsor) formátum ír le. A formátum megadásának szabályai mindhárom esetben azonosak. Ügyeljünk, hogy a formátum leírása (nem a nyomtatási kép) ne legyen 79 karakternél hosszabb. Ha nem fér el 79 karakteren, akkor használjunk folytatósort.

Egy formátumleírás tartalmazhat:

- Karakteres szövegeket. A karakteres szöveg tartalmazhat ékezetes karaktereket is. Ezeket a képernyő ékezetes karaktereivel kell megadni.
- Előzőleg definiált (FIELD=) változókat (ezek neveivel azonosítva) '@ ' jelek közé zárva (@ adatnev@).

- Rendszerváltozók hivatkozhatók az előbb ismertetett módon, '#' jelek közé zárva.
- A formátum megadásakor a ⟨ backslash ⟩ n új sort jelez, a ⟨ backslash⟩ folytatósort.
- A régi elemi sor adataira @old.adatnev@ formában lehet hivatkozni. (Minden elemi (detail) sorról készül egy mentés a kontrollváltások figyeléséhez. Ezek a mentett értékek érhetők így el.)
- A total (összegzett) adatok a @total(szint,adatnév)@ formájában hívhatók.

Az adathivatkozások a megadott sorpozícióban értelmeződnek.

5.1.2. Listafejléc megadása

Minden lista tartalmazhat 10 fejlécet, melyek közül a leírón belüli parancsokkal választhatunk. A fejléceket a következő két paranccsal kell megadni:

HEADER=⟨n⟩

HFORMAT=a fejléc formátumának a leírása

- ⟨n⟩ a fejléc sorszáma. Lehetséges értékei 1, 2, ... 10
- a formátum leírás meg kell, hogy feleljen a formátumleírás szabályainak.

5.1.3. Az elemi sorok megadása

Minden lista tartalmazhat 10 elemi sor leírást, melyeket a leírón belüli parancsokkal váltogathatunk. Az elemi sorok kiírási formájának megadása a következő:

DETAIL=⟨n⟩ DFORMAT=az elemi sor formátumának a leírása

ahol:

- ⟨n⟩ az elemi sorforma száma. Lehetséges értékei 1, 2, ... 10

- a formátum leírásában eleget kell tenni a fentiekben ismertetett szabályoknak.

5.1.4. Összegző sorok kiíratása

Az összegképzés leírása is a lista jellemzője. A listákban 9 szintű összegzés kérhető, a 10. a végösszeg, amely a megadott változókra a lista végéig összegez. Az összegfokozatok leírásának formája:

TOTAL=⟨n⟩

az n-edik összegző sor leírásának kezdete, n=1-től kezdődhet, folyamatosan felfelé, a végösszeg a 10-es. A végösszeghez nem kell folyamatosság.

CONTROL_ON=változónév

a kontrollváltozó neve, amelynek megváltozása esetén a kiírás megkezdődik. A végösszeg definiálásakor a CONTROL_ ON nem értelmes.

TOTAL_ON=változónév

az összegzendő adat neve. A név után vesszővel elválasztva megadhatók: a kiírási mezőszélesség, a dec. jegyek száma, igazítás (R: jobbra, L: balra). Akárhány TOTAL_ON lehet egy összegfokozaton belül. Minden megadott összegfokozat külön kummulálódik. Az összegsorok láncreakcióval íródnak ki, tehát pl. a 4-es szintű összegfokozat kiírási feltételének bekövetkezésekor az alsóbb szintű összegfokozatok (1,2,3) is kiíródnak. A kiírás után a megfelelő összegek nullázódnak.

FORMAT=formátumleírás

szabályai azonosak a többi formátumleíráséval.

Belső parancsok

A belső parancsok a leíró file-ban, valamelyik formátumleírás részeként adhatók meg.

A parancsot mindíg '#' jelek közé kell írni.

A következő parancsokat használhatjuk:

- # newpage#
  Feltétel nélküli lapemelés végrehajtása
- # totalpage#
  Csak az összegfokozatok leírásának formátumában szerepelhet. Hatására a láncreakciószerűen kiíródó összegző sorok közül az utolsóként kiírt után egy lapemelés hajtódik végre.
- #chdetform=⟨n⟩#
  Az aktuális kiírás után az ⟨n⟩-edik elemi sorformátum szerint íródnak ki a továbbiak.
- #chheadform=⟨n⟩#
  Az aktuális kiírás után az ⟨n⟩-edik fejléc formátum szerint íródnak ki a továbbiak.

5.2. A query leíró file

A leíró file segítségével több MISAM állományból álló adatbázis lekérdezése lehetséges.

A leíró file-ban kell megadni, hogy milyen adatokat kérünk az adatbázisból. Ez az adategyüttes a detail-sor, amelynek elemei az adatbázis által tartalmazott MISAM állományokból származnak. A detail-sorok együttesét nevezzük detail táblának. A detail-sorok táblán belüli rendezettségét is a query leíró file-ban kell megadni.

A detail-sor leírásánál minden egyes mezőnél meg kell adni, hogy melyik MISAM állományból származik és az állományon belül hol található.

A leíró file-ban le kell írni a detail-sor előállításához szükséges file-kapcsolatokat is. Mivel a MISAM rendszer indexfile orientált, a file-kapcsolatok a QUERY-ben is az indexállományokon keresztül kezelhetők. A detail-sor összeállítása mindig egy ún. master állományból indul ki. Ebben a master állományban jelölhetünk ki kulcs mezőket, amelyek alapján más állományokat kapcsolhatunk a master állományhoz. Az így csatolt állományok a link-állományok. A master file-hoz tetszőleges számú link-állomány kapcsolható. A link-állományokhoz további állományok csatolhatók. Íly módon egy több ágú fához hasonló link-struktúrát hozhatunk létre. A link-struktúra mélysége legfeljebb négy lehet. A detail-sor leírásánál csak olyan MISAM állományra lehet hivatkozni amelyik szerepel a link-struktúrában. A futtató rendszer mindig a master állományon veszi a következő rekordot és a link-leírás alapján olvassa a többi file-t. Előfordulhat, hogy egy master rekordhoz a link-állományban több rekord tartozik. Ezekre a rekordokra külön rendezettséget adhatunk meg. Ezáltal lehetőség van ún. kvázi set kapcsolatok kijelölésére, amit egyébként a MISAM rendszerben nem tudunk megtenni.

Mivel a detail-sor összeállítása mindig a master rekordból indul, a detail-tábla rendezettségét is a master állományon tudjuk kijelölni. Ez egyszerűen egy kulcs kijelölése a master file-ra. Ha a kijelölt kulcsnak megfelelő indexállomány nem létezik, akkor a futtató rendszer automatikusan épít egy köztes indexfile-t. A megadott kulcs alapján a detail-táblába kerülő detail-sorokra kijelölhetünk intervallumot is.

```
**************************************************
*                          MINTA.QRF
* Minta QUERY leiró file
*
**************************************************
* A QUERY-be bevont MISAM állományok száma
*
ISAM_NO=7
**************************************************
* MISAM állományokat leíró file-ok
*
MASTER=C:\qlib\_filedef\ CIKK.FDF
ISAM=C:\qlib\_filedef\ VEVO.FDF
ISAM=C:\qlib\_filedef\ SZALL.FDF
ISAM=C:\qlib\_filedef\ ESE.FDF
ISAM=C:\qlib\_filedef\ VERE.FDF
ISAM=C:\qlib\_filedef\ SZASZ.FDF
ISAM=C:\qlib\_filedef\ TEREL.FDF
**************************************************
* File kapcsolatok leírása
**************************************************
* CIKK állományhoz kapcsolt állományok
*
LINKED=CIKK
LINKED_NO=2
*
* VERE állomány csatolása
*
ISAM_NAME=VERE
```

```
*
* Master kulcs: a CIKK rekordban hol található az
* információ, amely alapján a VERE rekordra
* rátalálhatunk.
*
PARTNO=1
*
* Mező hivatkozási név (.FDF file), mezőn belüli
* offset és hossz
*
PARTS=cikk_kod 0 2
*
* Link kulcs: a VERE rekordban hol található az
* információ, amely alapján CIKK-ből hivatkozunk.
*
PARTNO=1
PARTS=cikk_kod 0 2
*
* Egyedi a kulcs vagy nem
* SET: egy cikk rekordhoz több VERE rekord tartozhat
KEY=SET
*
* SZASZ állomány csatolása
*
ISAM_NAME=SZASZ
* Master kulcs
PARTNO=1
PARTS=cikk_kod 0 2
* Link kulcs
```

```
PARTNO=1
PARTS=cikk_kod 0 2
* Egyedi a kulcs vagy nem
KEY=UNIQUE
************************************************
* VERE állományhoz kapcsolt állományok
*
LINKED=VERE
LINKED_NO=2
*
* TEREL állomány csatolása
*
ISAM_NAME=TEREL
* Master kulcs
PARTNO=2
PARTS=cikk_kod 0 2
PARTS=vevokod 0 2
* Link kulcs
PARTNO=2
PARTS=cikk_kod 0 2
PARTS=vevokod 0 2
* Egyedi a kulcs vagy nem
KEY=SET
*
* VEVO állomány csatolása
*
ISAM_NAME=VEVO
* Master kulcs
PARTNO=1
```

```
PARTS=vevokod 0 2
* Link kulcs
PARTNO=1
PARTS=vevokod 0 2
* Egyedi a kulcs vagy nem
KEY=UNIQUE
*************************************************
* TEREL állományhoz kapcsolt állományok
*
LINKED=TEREL
LINKED_NO=1
*
* ESE állomány csatolása
*
ISAM_NAME=ESE
* Master kulcs
PARTNO=1
PARTS=esekod 0 3
* Link kulcs
PARTNO=1
PARTS=esekod 0 3
* Egyedi a kulcs vagy nem
KEY=UNIQUE
*************************************************
* SZASZ állományhoz kapcsolt állományok
*
LINKED=SZASZ
LINKED_NO=1
*
```

```
* SZALL állomány csatolása
*
ISAM_NAME=SZALL
* Master kulcs
PARTNO=1
PARTS=szallkod 0 2
* Link kulcs
PARTNO=1
PARTS=szallkod 0 2
* Egyedi a kulcs vagy nem
KEY=UNIQUE
**************************************************
* Detail tábla egy sorának leirása
*
*       Oszlopok jelentése sorban:
* - soron belüli offset
* - soron belüli hossz
* - dimenzió
* - típus
* - kiírási szélesség
* - tizedesek száma
* - az adatot tartalmazó MISAM állomány neve
* - az adatot tartalmazó mező neve
* - az adat mezőn belüli kezdőpozíciója
**************************************************
DETAILFIELDS=8
FIELD= 0  2 1A  2 0 CIKK  cikk_kod 0
FIELD= 2 10 1A 10 0 CIKK  cikknev 0
FIELD=12 10 1A 10 0 VEVO  vevonev 0
```

```
FIELD=22  8 1D  8 2 VERE  rendeles 0
FIELD=30 10 1A 10 0 ESE   esenev 0
FIELD=40  4 1M  8 0 TEREL datum 0
FIELD=44 10 1A 10 0 SZALL szallnev 0
FIELD=54  8 1D  8 2 SZASZ rendeles 0
************************************************
* Master állomány kulcsának megadása
*
TMPLIB=c:\ tmp
PARTNO=1
PARTS=cikk_kod 0 2
************************************************
* Inetrvallum megadása, több darab esetén vesszővel
* kell elválasztani az egyes darabok értékeit.
*
*
FIRST_KEY=01
LAST_KEY=09
*
* MINTA.QRF vége
```

89.18471 Akadémiai Kiadó és Nyomda Vállalat, Budapest – Felelős vezető: Hazai György