```
 v_Values t_ValueTable;
BEGIN
 v_Values(1) := 'One';
 v_Values(3) := 'Three';
 v_Values(-2) := 'Minus Two';
 v_Values(0) := 'Zero';
 v_Values(100) := 'Hundred';

 DBMS_OUTPUT.PUT_LINE('Before DELETE, COUNT=' || v_Values.COUNT);
 v_Values.DELETE(100);
 DBMS_OUTPUT.PUT_LINE('After first DELETE, COUNT=' ||
v_Values.COUNT);
 v_Values.DELETE(1,3);  -- Removes 'One' and 'Three'
 DBMS_OUTPUT.PUT_LINE('After second DELETE, COUNT=' ||
v_Values.COUNT);
 v_Values.DELETE;       -- Removes all remaining values
 DBMS_OUTPUT.PUT_LINE('After last DELETE, COUNT=' ||
v_Values.COUNT);
END;
```

**EXISTS**

```
DECLARE
 TYPE t_FirstNameTable IS TABLE OF students.first_name%TYPE
  INDEX BY BINARY_INTEGER;
 FirstNames  t_FirstNameTable;
BEGIN
 FirstNames(1) := 'Scott';
 FirstNames(3) := 'Joanne';

IF FirstNames.EXISTS(1) THEN
   INSERT INTO temp_table (char_col) VALUES
    ('Row 1 exists!');
 ELSE
   INSERT INTO temp_table (char_col) VALUES
    ('Row 1 doesn''t exist!');
 END IF;
 IF FirstNames.EXISTS(2) THEN
   INSERT INTO temp_table (char_col) VALUES
    ('Row 2 exists!');
 ELSE
   INSERT INTO temp_table (char_col) VALUES
    ('Row 2 doesn''t exist!');
 END IF;
END;
```

**FIRST, LAST**

```
DECLARE
```



```
 TYPE t_LastNameTable IS TABLE OF students.last_name%TYPE
  INDEX BY BINARY_INTEGER;
 v_LastNames  t_LastNameTable;
 v_Index  BINARY_INTEGER;
BEGIN
 -- Insert rows in the table.
 v_LastNames(43) := 'Mason';
 v_LastNames(50) := 'Junebug';
 v_LastNames(47) := 'Taller';

  v_Index := v_LastNames.FIRST;
  v_Index := v_LastNames.LAST;
END;
```

```
DECLARE
 TYPE t_MajorTable IS TABLE OF students.major%TYPE
  INDEX BY BINARY_INTEGER;
 v_Majors t_MajorTable;
 v_Index  BINARY_INTEGER;
BEGIN
 v_Majors(-7) := 'Computer Science';
 v_Majors(4) := 'History';
 v_Majors(5) := 'Economics';

 v_Index := v_Majors.FIRST;
 LOOP
  INSERT INTO temp_table (num_col, char_col)
      VALUES (v_Index, v_Majors(v_Index));
  EXIT WHEN v_Index = v_Majors.LAST;
  v_Index := v_Majors.NEXT(v_Index);
 END LOOP;
END;
```

**Változók a parancsokban (Kapcsoló változók)**

```
DECLARE
 v_NumCredits  classes.num_credits%TYPE;
BEGIN
  v_NumCredits := 3;
 UPDATE CLASSES
  SET num_credits = v_NumCredits
  WHERE department = 'HIS'
  AND course = 101;
END;
```

```
DECLARE
 v_StudentRecord  students%ROWTYPE;
```

```
 v_Department    classes.department%TYPE;
 v_Course        classes.course%TYPE;
BEGIN
SELECT *
   INTO v_StudentRecord
   FROM students
   WHERE id = 10000;

SELECT department, course
   INTO v_Department, v_Course
   FROM classes
   WHERE room_id = 99997;
END;



DECLARE
 v_StudentID  students.id%TYPE;
BEGIN
 SELECT student_sequence.NEXTVAL
   INTO v_StudentID
   FROM dual;

 INSERT INTO students (id, first_name, last_name)
   VALUES (v_StudentID, 'Timothy', 'Taller');

 INSERT INTO students (id, first_name, last_name)
   VALUES (student_sequence.NEXTVAL, 'Patrick', 'Poll'); → sorozat
END;


DECLARE
 v_Major         students.major%TYPE;
 v_CreditIncrease  NUMBER := 3;
BEGIN
v_Major := 'History';
 UPDATE students
   SET current_credits = current_credits + v_CreditIncrease
   WHERE major = V_Major;
END;


DECLARE
 v_StudentCutoff  NUMBER;
BEGIN
 v_StudentCutoff := 10;
DELETE FROM classes
   WHERE current_students < v_StudentCutoff;

DELETE FROM students
   WHERE current_credits = 0
```



```
   AND   major = 'Economics';
END;
```

**Hivatkozás a táblára**

**[shema] tábla [@AB_kapcsolat]**

**Az SQL*Net-ben AB_kapcsolat létrehozása:**

```
CREATE DATABASE LINK  kapcsolat_név
   CONNECT TO  felhasználó_név  IDENTIFIED BY  jelszó
   USING  sqlnet_sor;

CREATE DATABASE LINK  example_backup
   CONNECT TO  example  IDENTIFIED BY  ****
   USING  'backup_database';
```

**Utána lehet, pl.:**

```
UPDATE  studets@ example_backup
 SET major = 'Music'
 WHERE  id = 10005;
```

**Privilégiumok**

GRANT és REVOKE
Object és System privilege

**Object privilégiumok:**

ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES,
SELECT, UPDATE

```
GRANT privilégium ON objektum TO tulajdonos
        [WITH GRANT OPTION];

GRANT SELECT ON classrs TO userA;
```

Rendszer (system) privilégiumok esetén:

```
GRANT privilégium TO tulajdonos
        [WITH ADMIN OPTION];
GRANT CREATE TABLE, ALTER ANY PROCEDURE TO userA;

REVOKE privilégium ON objektum FROM tulajdonos
        [CASCADE CONSTRAINTS];

REVOKE SELECT ON classses FROM userA;

REVOKE ALTER TABLE, EXECUTE ANY PROCEDURE FROM userA;
```

**SZEREPEK (Role)**

```
CREATE ROLE table_query;
GRANT SELECT ON students TO table_query;
GRANT SELECT ON classes TO table_query;
GRANT SELECT ON rooms TO table_query;

GRANT table_query TO userA;
GRANT table_query TO userB;


GRANT table_query TO PUBLIC;  -- mindenkinek
```

**Beépítet szerepek**

CONNECT → ALTER SESSION, CREATE CLUSTER, CREATE
DATABASE LINK, CREATE SEQUENCE, CREATESESSION, CREATE
SYNONIM, CREATE TABLE, CREATE VIEW

RESOURSE → CREATE CLUSTER, CREATE PROCEDURE, CREATE
SEQUENCE, CREATE TABLE

## 1. Alapvető fogalmak AB_6 (FOXPRO)

**Műveletek**
**Numerikus műveletek (a** prioritás sorrendjében)
** (vagy ^ ) (hatványozás)
* (szorzás), / ( osztás), % (maradékképzés)
+ (összeadás)
– (kivonás)

**Logikai műveletek (**prioritás sorrendjében)
**NOT** (vagy ! ) – negálás ( logikai **nem)**
**AND** – logikai **és**
**OR** - logikai **vagy**

**Karakteres műveletek**
+ kifejezések konkatenálása (összefűzése)
– kifejezések olyan konkatenálása, amely az első kifejezés végén levő szóközöket a második kifejezés végére helyezi el
X$Y – a $ művelet eredménye *True* (.T.), ha X Y része (vagy megegyezik vele) és – *False* (.F.), ha nem
== két kifejezés identikus egyenlősége (beleértve a kifejezések hosszát és a szóközök számát)

**Relációs műveletek**
Relációs műveletek numerikus-, karakter- és dátum kifejezésekkel alkalmazhatók.
< – kisebb, mint, > – nagyobb, mint, = – egyenlő,
# (<> vagy != ) – nem egyenlő,
<= – kisebb vagy egyenlő (nem nagyobb), >= – nagyobb vagy egyenlő (nem kisebb).

Karakter kifejezések összehasonlítása a második (jobb oldali) kifejezés hossza szerint történik.

**1. Példa**

'abc' = 'a' - True,
'a'='abc' - False.

**2. Parancsok felépítése**

Egy tipikus parancsnak, amely a műveletet az adatbázissal hatja végre, a következő struktúrája van:

**Parancs-név** [<hatókör>] [<kifejezéslista>] [**FOR** <feltétel-1>] [**WHILE** <feltétel-2>]

A <hatókör> azoknak a rekordoknak a halmazát adja meg, amely részt vesz a műveletben. A <hatókör> paraméternek a következő értékei lehetnek: **ALL / REST / NEXT** N / **RECORD** N ( / jel azt jelenti, hogy a felsorolt paraméterek (opciók) közül csak egy választható ki).
**ALL** – az összes rekord
**REST** – az aktuális rekordtól az utolsóig
NEXT N – a következő N rekord



**RECORD** N – az N sorszámú rekord

**Ha a <hatókör> a parancsban nincs megadva, akkor a parancs a <hatókör> alapértelmezését fogja alkalmazni.**

**FOR** <feltétel-1> – csak azok a rekordok vesznek részt a műveletben, amelyek megfelelnek a feltételnek,

**WHILE** <feltétel-2> – a művelet addig folytatódik, amíg a feltétel igaz (a WHILE feltételt csak rendezett adatbázison van értelme alkalmazni).

A FoxPro-ban a SET parancsokkal különböző beállításokat lehet végrehajtani:

**SET** <paraméter> **TO** <paraméter értéke>; vagy
**SET** <paraméter> **OFF/ON**

**ON –** a <paramétert> bekapcsolja, **OFF** – pedig kikapcsolja.

A karakter konstansokat különbözőképpen jelölhetjük: " ", ' ', [ ].
A dátum konstansok formátuma: {10. 02. 99}. Az üres dátumot így jelöljük: {}.

Logikai konstansok jelölése: **.t. (.T.), .f. (.F.)** .

**QUIT** parancs a FoxPro működését fejezi be és visszaadja a vezérlést az operációs rendszernek.

## 3. Adatbázis állományok létrehozása, megnyitása és bezárása

***Állomány struktúrájának létrehozása***

**CREATE** [<adatbázis-állomány>]

A mezők lehetséges típusai:

**Character (C) –** karakter típusú, maximális hossza – 254 karakter (byte),
**Numeric (N) –** numerikus típusú, maximális hossza 20, alapértéke – 8,
**Float (F) –** lebegőpontos numerikus mező, maximális hossza 20, alapértéke – 8,
**Date (D) –** dátum típusú, hossza – 8,
**Logical (L) –** logikai típusú, hossza – 1,
**Memo (M) –** memo típusú adatok határozatlan hosszúságú karaktersorozatot tartalmaznak és egy FPT kiterjesztésű állományban vannak elhelyezve. Ezt az állományt a CREATE parancs automatikusan hozza létre. A DBF állományban a memo-mező mutatója tárolódik, amely által a FPT- állomány megfelelő rekordjához hozzá lehet férni. A mutató hossza -10,
**General (G) –**ez az általános típus csak a FoxPro Windows változatában alkalmazható. Különböző objektumok (képek, hangok, táblázatok) tárolhatók. A DBF állomány az objektum mutatóját tárolja.

Az adatbázis állomány struktúráját a
**MODIFY STRUCTURE**

parancs segítségével meg lehet változtatni.

*Állomány megnyitása és bezárása.*

**USE** *f*
parancs megnyitja a *f.dbf* nevű állományt. Ha az *f* állomány memo-mezőket is tartalmaz, akkor az *f.fpt* állomány is automatikusan nyílik meg.
A **USE** parancs paraméterek nélkül bezárja az aktív állományt.

*Munkaterület*

A munkaterületek száma 225. Ha egyidejűleg több állományt szükséges megnyitni, akkor a munkaterületet a SELECT paranccsal kell kiválasztani és utána az USE paranccsal az állományt megnyitni. A munkaterületeket számokkal szokták megnevezni: 1, 2, 3,...,10 (vagy A, B, C,...,J) és W11, W12,.... A munkaterület kiválasztása:

**SELECT** <munkaterület>
parancs segítségével történik.

**2. Példa**

```
SELECT 1
USE varos
SELECT 2
USE elofizeto
```

Ha azt akarjuk, hogy a *varos* állomány legyen aktív, akkor a

```
SELECT 1
```

vagy

```
SELECT varos
```

paranccsal az első munkaterületre kell átkapcsolni. Egy állományt a következő szabad munkaterületen a

**USE** f **IN** 0

parancs segítségével lehet megnyitni.

**Összetett név**: *varos.telepnev, varos->telepnev, elofizeto.telszam, elofizeto->telszam.*

*Állományok bezárása*

**CLOSE DATABASE**
**CLOSE ALL**

**4. Állományok feltöltése adatokkal**
**APPEND [BLANK]**
parancs az új rekordot az állomány végére helyezi.
A memo-mezőkhöz a Ctrl+Home billentyűkkel lehet hozzáférni, vagy a mezőre az egérrel kétszer kell rákattintani. A memo-mezőből a Ctrl+W (Ctrl+End) vagy az Esc billentyűkkel lehet kilépni. Az Esc billentyű alkalmazása esetén a mezőbe új érték nem kerül be.

**INSERT [BEFORE] [BLANK],**

A dátum formátumát a

**SET DATE** <Date>



paranccsal lehet megváltoztatni.
A <Date> paraméter lehetséges értékei:

| | |
|---|---|
| **AMERICAN** | (hh/nn/éé) |
| **ANSI** | (éé.hh.nn) |
| **BRITISH** | (nn/hh/éé) |
| **FRENCH** | (nn/hh/éé) |
| **GERMAN** | (nn.hh.éé) |
| **ITALIAN** | (nn-hh-éé) |

**SET CENTURY OFF/ON**
ON - a dátum év négy számjegyű lesz,
OFF – csak az év két utolsó számjegye lesz látható.

A **SET BELL ON/OFF**
parancs a hangjelzést be- vagy kikapcsolja.

Az állományok különböző könyvtárakban tárolódhatnak. Az aktuális meghajtót és az elérési útvonalat a

**SET DEFAULT TO** [[<meghajtó>] <útvonal>]

paranccsal lehet megadni. Ha a SET DEFAULT parancs nem tartalmaz paramétereket (vagy nem hajtottuk végre), akkor az aktuális könyvtár az lesz, ahonnan a FoxPro-t elindítottunk.

**3. Példa**

```
SET DEFAULT TO A:\programok\foxprow
SET DEFAULT TO C:\munka
SET DEFAULT TO \
SET DEFAULT TO.
```

Ha az állományokat több könyvtárban tároljuk, akkor a

**SET PATH TO** [<útvonal-lista>]

parancs az állományok keresési útvonalát határozza meg.

**5. Rekordok módosítása**

A BROWSE, EDIT ( CHANGE )
Ha az adatok bevitelénél a mezők értékei rekordonként ismétlődnek, akkor célszerű a

**SET CARRY ON**

paranccsal beállítani az adatok automatikus másolását az előző rekordból. A parancs alapértelmezése - OFF (a másolás kikapcsolt állapotban van).

**SET CARRY TO** <mezőlista>

*BROWSE –ablak*

A **BROWSE** parancs struktúrája:

**BROWSE [FIELDS** <mezőlista>] [**FOR** <feltétel-1>]
[**FREEZE** <mező-1>] [**KEY** <kifejezés-1> [,<kifejezés-2>]]
[**LAST**] [**LEDIT/REDIT**] [**LOCK** N1] [**NOAPPEND**]
[**NOCLEAR**] [**NODELETE**] [**NOEDIT/NOMODIFY**]
[**NOLGRID/NORGRID**] [**NOLINK**] [**NOMENU**]
[**NOWAIT**] [**REST**] [**TIMEOUT** N2] [**TITLE** C]

[**WHEN** <feltétel-2>] [**WINDOW** <ablaknév>] [**VALID** [F:]
<feltétel-3> [**ERROR** C2]] [**COLOR SCHEME** N3/
**COLOR** ... ] [**FONT** C4[,N4]] [**STYLE** C5]

**4. Példa**

BROWSE FOR z> 200 AND z <=300

**5. Példa**

USE vevo
BROWSE FIELD lakhely = ALLTRIM(megye) + ', ' + ;
ALLTRIM(varos) && *lakhely* - szerkesztett mező

*CHANGE – ablak*

A **CHANGE** vagy **EDIT** parancsokat a CHANGE-ablakban hajtjuk végre. A parancs paraméterei és opciói azonosak a BROWSE-parancs paramétereivel és opcióival.

**6. Mezők módosítása a REPLACE paranccsal**

**REPLACE** [<hatókör>] <mezőnév> **WITH** <kifejezés>, <mezőnév> **WITH** <kifejezés>, ... ., [**FOR** <feltétel-1>] [**WHILE** <feltétel-2>] [**ADDITIVE**]

A **BLANK** parancs által a megnevezett (vagy az alapértelmezés szerint az összes) mezőt lehet törölni.

**BLANK** [<hatókör>] [**FIELDS** <mezőlista>] [**FOR** <feltétel-1>] [**WHILE** <feltétel-2>].

A **BLANK** parancs paraméterek nélkül törli az összes adatot az állományból, de a rekordok nem semmisülnek meg, csak üres mezőket tartalmaznak.

**7. Műveletek a rekordmutatókkal**

**GO TOP/ BOTTOM/** N [**IN** <munkaterület>] (vagy **GOTO**)

**SKIP** [N] [**IN** <munkaterület>]

Az állomány rekordjainak számát a

**RECCOUNT** ([<munkaterület>])

függvény állapítja meg.
**EOF** ([<munkaterület>]) , **BOF** ([<munkaterület >])

**8. Adatbázis tartalmának megjelenítése**

**DISPLAY** [<hatókör>] [**FIELDS** <mezőlista>] [**FOR** <feltétel-1>]
[**WHILE** <feltétel-2>] [**OFF**] [**TO PRINTER /**
**TO FILE** <állománynév>]

) - mező megjelenítésének szélességét a képernyőn

**SET MEMOWIDTH N**

állíthatja be. N lehetséges értékei: 8 ... 256. Az alapértelmezés - 50.

hatókör>] [**FIELDS** <mezőlista>] [**FOR** <feltétel-1>] [**WHILE**
eltétel-2>] [**OFF**] [**TO PRINTER / TO FILE** <állománynév>]



**Példa**

USE elofizeto
LIST FOR irszam=3300 TO FILE egriek

Az *elofizeto.dbf* állományból az *egriek.txt* állományba kerülnek az egri előfizető adatai.

**Példa**

GO 2
SET MEMOWIDTH 30
DISDPLAY nev, valtoz OFF && a *valtoz* memo-mező

Egy állományt a lemezről csak akkor lehet törölni, ha az nincs megnyitva.

**ERASE** <állománynév>

**ZAP**

A rekordokat **logikailag** és **fizikailag** lehet törölni. A logikai törlésnél a rekord megmarad az állományban, de meg lesz jelölve a későbbi fizikai törlésre. A logikai törlést a

**DELETE** [<hatókör>] [**FOR** <feltétel-1>] [**WHILE** <feltétel-2>]

parancs hajtja végre.
**SET DELETED ON/OFF**
ON – a logikailag törölt rekordokat a parancs figyelmen kívül hagyja (elrejti azokat),
OFF – a logikailag törölt rekordok megjelennek a képernyőn. Ezek a rekordok a * (csillag) karakterrel vannak megjelölve.
Az aktuális rekord állapotát a

**DELETED( )**

függvény állapíthatja meg. Ha a rekord törlésre van kijelölve, akkor a függvény .T. értéket ad vissza, másként - .F.-et.
A logikai törlésről a

**RECALL** [<hatókör>] [**FOR** feltétel-1] [**WHILE** feltétel-2]

parancs segítségével lehet lemondani. A <hatókör> alapértelmezése – az aktuális rekord.
A logikailag törölt rekordokat a

**PACK** [**MEMO**] [**DBF**]

paranccsal fizikailag lehet törölni. A fizikailag törölt rekordokat már nem lehet visszaállítani.

**9. Rekordok szűrése**

**SET FILTER TO** <feltétel>
**SET FILTER TO**
A szűrő tartalmát (feltételét) a megadott munkaterületben a

**FILTER** ([<munkaterület>])

függvénnyel lekérdezhetjük.

**Állományok rendezése, adatok keresése**
**Indexelés**

1) IDX kiterjesztésű **önálló (egyszerű) indexállományt,** amely csak *egy* indextáblát tartalmaz.
2) CDX *(compound* index) kiterjesztésű **multiindexes állományt,** amely *több* indextáblát tartalmazhat.

A multiindexes állományba tartozó táblát **tegnek** (TAG) nevezik. Minden tegnek neve van. A multiindexes indexállomány **strukturált** vagy **nem strukturált** típusú lehet. A strukturált CDX állomány neve megegyezik az adatbázis állomány nevével és mindig a DBF állománnyal együtt nyílik meg automatikusan. (Pl., ha az adatbázis állomány neve *f.dbf,* akkor a strukturált CDX állomány *f.cdx* nevet kap). A nem strukturált CDX állomány neve tetszőleges lehet, és a megnyitásáról külön kell gondoskodni. A multiindexes állomány alkalmazása akkor előnyösebb, ha egyszerre több indexállománnyal kell dolgoznunk.

A FoxPro az úgy nevezett **compact** indexeket tartalmazza, ami gyorsítja az indexelést és a rekordok elérését. A multiindexes állomány mindig compact tulajdonságú, az egyszerű indexállomány (IDX) csak akkor compact, ha megadjuk a COMPACT paramétert.
DBF állomány létrehozása az INDEX parancs által történik.

**INDEX ON** <kulcs-kifejezés> **TO** <IDX állomány>/ **TAG** <tegnév> **[OF** <CDX állomány>] **[FOR** <feltétel>] **[COMPACT] [ASCENDING/DESCENDING] [UNIQUE] [ADDITIVE]**

<kulcs-kifejezés> - kifejezés, amely szerint a DBF állományt kell rendezni. A kifejezés maximális hossza INX állomány esetén 100 karakter, a CDX állomány esetén pedig - 240. Kulcsként leggyakrabban egy mezőt szoktak alkalmazni. Általános esetben több mező is részt vehet a kulcs-kifejezésben;
<IDX állomány> - az egyszerű indexállomány neve;
<tegnév> - tegnek a neve;
<CDX állomány> - a nem strukturált indexállomány neve;
<feltétel> - az indexelésben csak azok a rekordok vesznek részt, amelyekre igaz a <feltétel> ( a FOR bejegyzés elhagyásával a műveletben az állomány összes rekordja részt vesz),
**ASCENDING/DESCENDING** – az indexelés a kulcs értékeinek növekvő (csökkenő) sorrendben történik;
**UNIQUE** – az indexállomány csak egyszer tartalmazhatja a kulcs egy bizonyos értéket. Ha több rekordnak azonos a kulcsértéke, akkor ezek közül csak az első rekordot kell indexelni;
**ADDITIVE** – az INDEX parancs a már eddig megnyitott indexállományokat nem zárja be. Alapértelmezésben – bezárja.

**Példa**
```
INDEX ON pol+fio TO p COMPACT
INDEX ON -irszám  TAG OF irrend
INDEX ON nem+nev TAG nem OF pf
```

**Indexállományok megnyitása és bezárása**

Az USE parancs a már létező indexállományokat megnyitja az adatbázis állománnyal egyidejűleg:

**USE** <adatbázisnév> **[IN** <munkaterület>] **[ALIAS** <aliasnév> **[AGAIN] [NOUPDATE] [INDEX** <indexállomány-lista>] **[ORDER** [N / <INX állomány> / **[TAG** <tag név> **[OF** <CDX állomány>]]]] **[ASCENDING/DESCENDING] [EXCLUSIVE]**



Ha az indexállomány-lista több állományt tartalmaz, akkor lényeges, hogy melyik állományt választjuk **fő indexállományként**. A USE utáni parancsok az adatbázis állomány rekordjait a fő index alapján érik el. A fő indexet meg lehet változtatni, vagy le lehet mondani az indexek alkalmazásáról (ebben az esetben érvényes a rekordok fizikai sorrendje).
**AGAIN** – a már megnyitott adatbázis-állományt egy másik munkaterületen is lehet megnyitni. Egy indexállomány csak egy munkaterületen nyitható meg;
**NOUPDATE** – az adatbázis nem módosítható;
<indexállomány-lista> - megadja az indexállományok sorrendjét. A strukturált indexállomány az adatbázissal együtt automatikusan kerül megnyitásra,
**ORDER** N – kiválasztja a fő indexállományt az indexállomány listából (N - az indexállomány sorszáma a listában). Ha N=0, akkor nem választja ki a fő indexállományt;
**ORDER** <INX állomány> – a fő index az INX- állomány neve által van kiválasztva;
**ORDER** TAG <tegnév> **[OF** <CDX állomány>] – a fő index a tegnek a neve által van megadva;
Alapértelmezés szerint (ha az ORDER paraméter hiányzik) - a lista első indexállománya lesz a főindex.
**ASCENDING/DESCENDING** – a fő indexben kijelöli a kulcsnak a növekedő (**ASCENDING**) vagy csökkenő (**DESCENDING**) sorrendjét; (alapértelmezés – **ASCENDING),**

Ha az adatbázis állomány már meg van nyitva és aktív, akkor a SET INDEX paranccsal a hozzátartozó indexállományokat utólag is meg lehet nyitni:

**SET INDEX TO** [<indexállomány-lista>] **[ORDER** N/ <.IDX állomány>/ **[TAG]** <tagnév> **[OF** <.CDX állomány>]] **[ASCENDING/DESCENDING] [ADDITIVE]**

**ADDITIVE** – az új állományok megnyitása nem zárja be a már megnyitott index állományokat.
Ha az ADDITIVE paraméter nincs megadva, akkor a megnyitott indexállományok be lesznek zárva.
Ha a lista több indexállományt tartalmaz, akkor a főindex az első állomány lesz (ha nincs megadva az ORDER paraméter) vagy az ORDER paraméterben kijelölt állomány.
Az összes indexállományt a

**SET INDEX TO**

vagy a

**CLOSE INDEX**

paranccsal lehet bezárni. A strukturált indexállomány csak az adatbázis-állománnyal egyidejűleg zárható be.

**Példa**
```
USE varos
INDEX ON irszam TO irrend
```
**Példa**
```
USE varos
INDEX ON SUBSTR (telepnev, 1, 5) + STR (irszam, 4) TO rend
```
**Példa**
```
USE ber
INDEX ON ber FOR ber > 150000 TO nagyber
```
**Példa**

```
USE emberek
INDEX ON cim TAG cim
INDEX ON valalat TAG valalat OF valrend.cdx
```

**Műveletek az indexekkel**

A fő index a

**SET ORDER TO** N/<IDX-állomány>/ [TAG] <tagnév> [**OF** <CDX-állomány>]
[**IN** <munkaterület>] [**ASCENDING/ DESCENDING**]

parancs által megváltoztató. A parancs paraméterei megegyeznek a SET INDEX TO parancs paramétereivel.
**SET ORDER TO** [0] – kikapcsolja a fő indexet. A további parancsok az adatbázis állomány rekordjaihoz a fizikai sorrendben férhetnek hozzá. A megnyitott indexállományok továbbra is az adatbázis állomány módosítása esetén megfelelően változnak.

**Példa**

```
USE video INDEX cim.idx, rend.cdx, kolcson.idx
```

A *video.dbf* állományhoz tartozik a strukturális indexállomány (*video.cdx*), amelynek 2 *teg*-je van: *szam* és *ev*. A *video.cdx* a *video.dbf* állománnyal automatikusan megnyílik. A *rend.cdx* állomány 2 teget tartalmaz: *vetel* és *adas*. Az indexállományok sorrendje alapján a SET ORDER TO N parancs a főindexet így választja ki -

```
SET ORDER TO 1 – cim.idx,
SET ORDER TO 2 – kolcson.idx,
SET ORDER TO 3 – video.cdx, Tag: szam,
SET ORDER TO 4 – video.cdx, Tag: ev,
SET ORDER TO 5 – rend.cdx Tag: vetel,
SET ORDER TO 6 – rend.cdx Tag: adas.
```

A fő index nevét az

**ORDER** ([<munkaterület>])

függvény segítségével lehet lekérdezni.
Az IDX- állományokat CDX- állományokká lehet átalakítani:

**COPY INDEXES** <IDX- állomány> /**ALL** [**TO** <CDX- állomány>]

Ha a <CDX- állomány> nincs megadva, akkor az IDX- állomány a strukturált indexállományba kerül. A

**COPY TAG** <tagnév> [**OF** <CDX- állomány>] **TO** <IDX- állomány>

parancs egy tegből egy IDX- állományt hoz létre.
Az indexállományok törlését a

**DELETE TAG** <tegnév> [**OF** <CDX- állomány>] vagy
**DELETE TAG ALL** [**OF** <CDX- állomány>]

parancsokkal lehet végrehajtani.

**Adatbázis rendezése**

**SORT TO** <f1> [**ASCENDING/ DESCENDING**] **ON** <mező1> [**/A**] [**/D**] [**/C**]
[,<mező2> [**/A**] [**/D**] [**/C**]…] [<hatókör>] [**FIELDS** <mezőlista>]



[**FOR** <feltétel1>] [**WHILE** <feltétel2>]

<f1> állomány a rendezett rekordokat fogja tartalmazni. A parancs az aktív állományt nem változtatja meg.
**ON** <mező> [**/A**] [**/D**] [**/C**] – a mező neve, amely szerint az állományt rendezzük,
**/A** – növekvő sorrendbe rendezi (alapértelmezés),
**/D** – csökkenő sorrendbe rendezi,
**/C** – a parancs nem különbözteti meg a kis- és a nagybetűket.
Ezeket az opciókat együtt is lehet alkalmazni. Pl., /DC.
A rendezésre összesen 10 mezőt lehet kijelölni. Több mező esetén a rendezés a megadott sorrendben történik. A rendezés elsősorban az első mező alapján történik. A második mező szerint a rendezés csak akkor történik, ha az első mező értékei több rekordban azonosak.
**ASCENDING** – növekvő sorrend (alapértelmezés),
**DESCENDING** – csökkenő sorrend.
Az **ASCENDING** és **DESCENDING** paraméterek az összes rendezésre kijelölt mezőre hatnak, az /A és /D opciók pedig csak a hozzá tartozó mezőre.
**FIELDS** <mezőlista> - az <f1> állományba csak a megadott mezők kerülnek (alapértelmezés – az összes mező),
**FOR** <feltétel1> - csak azok a rekordok kerülnek a <f1> állományba, amelyek megfelelnek a feltételnek,
**WHILE** <feltétel2> - a rendezés csak addig folytatódik, ameddig a feltétel igaz.

**Adatok keresése**

**Szekvenciális keresés**

Szekvenciális keresésre a következő két parancsot lehet alkalmazni – **LOCATE** és **CONTINUE.**

**LOCATE** [<hatókör>] [**FOR** <feltétel-1>] [**WHILE** <feltétel-2>]

A LOCATE parancs az adatbázis állományban olyan (első) rekordot keres, amely megfelel a FOR- feltételnek. Ha a <hatókör> és a WHILE- feltétel hiányzik, akkor a keresés az első rekordtól kezdődik. A parancs a rekordmutatónak új értéket ad. Ha a keresés sikeresen végződött, akkor a mutató a megfelelő rekord sorszámát kapja, ha nem – akkor a mutató eléri az állomány végét, és az értéke az állomány rekordjainak száma plusz egy lesz.
Ha a LOCATE parancs talált egy rekordot, amely megfelel az adott feltételnek, akkor a keresést tovább lehet folytatni (a LOCATE parancsban megadott FOR- feltétellel) a

**CONTINUE**

parancs segítségével. Ez a parancs paramétert nem tartalmaz. A LOCATE és CONTINUE parancsok eredményeinek értékelése a **RECNO**( ), **EOF**( ) és **FOUND**( ) függvényekkel történhet.
**FOUND**( ) függvény logikai értéket ad vissza: .T. – ha a keresés sikeres volt, .F. – ha nem.

**19. Példa**

```
SET TALK OFF
USE elofizeto
```

```
STORE 0 TO van
LOCATE FOR irszam=3300
DO WHILE FOUND( )
        van=van+1
        CONTINUE
ENDDO
? ' Egri előfizetők száma  : '+ LTRIM(STR(van))
USE
```

Hasznos lehet a következő függvény is:

**LOOKUP** (<mező-1>, <kifejezés>, <mező-2>)

LOOKUP keresi az első rekordot, amelyben a *<mező-2>* a *<kifejezés>*-t tartalmazza. Ha a keresés sikeres volt, akkor a függvény a *<mező-1>* értéket adja vissza, ha nem, akkor üres karaktersort.

**Példa**

? LOOKUP (kod, 'S', nev), nev

a képernyőre kiírja az azonosító számot és egy nevet, amely 'S' betűvel kezdődik.

**Adatok keresése indexelt állományokban**

**SEEK** <kifejezés>

parancs azt az első rekordot keresi, amelyben az index értéke egyenlő (vagy egyes esetekben közel van) a megadott kifejezés értékével. A kifejezés értékének keresése először az indexállományban bináris keresési eljárással történik és utána a rekordmutatót az adatbázis-állományban a megfelelő rekordra beállítja. A SEEK parancs csak az első rekordot keresi, amelyben a *<kulcs>* = *<kifejezés>* feltétel teljesül. Szükség esetén, a keresést lehet folytatni, mivel az állomány rendezett.

A SEEK parancs végrehajtása a

**SET EXACT ON/OFF**

parancs beállításától függ, amely a karakteres kifejezések összehasonlítását határozza meg.

ON - opció megköveteli az összehasonlítás pontos végrehajtását,

OFF - esetén az összehasonlítás eredménye akkor is igaz, ha a kifejezéseknek a hossza különböző.

A keresés eredménye függhet a

**SET NEAR ON/OFF**

parancs beállításától is. Ennek a parancsnak a hatása akkor érvényesül, ha a SEEK parancs nem talált olyan rekordot, amelyben az index értéke egyenlő a *<kifejezés>* értékével. Ha a NEAR parancsban az ON opció van beállítva, akkor a SEEK parancs kiválasztja azt a rekordot, amelynek a kulcsértéke elsőként haladja meg a *<kifejezés>* értékét. Ha az OFF opció van beállítva, akkor a mutató az utolsó rekord után fog megállni.

A SEEK parancs után (ugyanúgy, mint a LOCATE és CONTINUE parancsok) alkalmazzák a RECNO( ), FOUND( ) és EOF( ) függvényeket.

Hasznos lehet a

**SEEK** (<kifejezés> [,<munkaterület>])



függvény alkalmazása is, amely helyettesíti a SEEK parancsot és FOUND () függvényt. A SEEK függvény keresi a *<kifejezés>* értéket az aktív vagy a megadott *<munkaterületen>*, és ha a keresés sikeres volt, akkor .T. értéket ad vissza, ellenkező esetben - .F. értéket.

**Példa**

```
IF SEEK (2)
        DISPLAY
ENDIF
```

**Példa**

```
SET EXACT OFF
USE elofizeto
INDEX ON irszam TO rend
SEEK 3300
IF FOUND( )
        DISPLAY FIELDS irszam, nev, telepnev
ENDIF
```

Az indexállományokat a

**REINDEX**

parancs segítségével újból lehet rendezni.

**Adatbázis állományok összekapcsolása**

***Egy-egy* típusú összekapcsolás**

**SET RELATION TO** <kulcs> **INTO** <munkaterület> [,<kulcs> **INTO** <munkaterület> ... ] **[ADDITIVE]**

A parancs az aktív adatbázis-állományt egy kulcs által összekapcsolja egy másik állománnyal. Az állományokat, amelyeket hozzákapcsolunk az aktív állományhoz, a kulcs szerint kell indexelni. Ha azt akarjuk, hogy a már létező kapcsolatok megmaradjanak, akkor az ADDITIVE paramétert kell alkalmazni. A két állomány összekapcsolása után az állományok rekordmutatói összhangban fognak változni: az alárendelt állomány mutatója mindig arra a rekordra mutat, amelynek a kulcsértéke azonos az aktív állomány kulcsának az értékével. Ez a parancs azt követeli, hogy az alárendelt állományban a kulcsnak az értékei csak egyszer forduljanak elő.

Az állományok közti összekapcsolást a

**SET RELATION TO**

paranccsal (paraméter nélkül) lehet megszüntetni. A

**SET RELATION OFF INTO** <munkaterület>

megszünteti a kapcsolatot az adott munkaterületen.

***Egy-több* típusú összekapcsolás**

Az *egy-több* típusú összekapcsolás esetén az aktív állomány rekordjához az alárendelt állományban több rekord is kapcsolódhat. A FoxPro-ban ezt a

**SET SKIP TO** [<munkaterület-1>, ...]

paranccsal lehet létrehozni. Egy SET SKIP TO ... parancs több kapcsolatot hozhat létre. Ezt a parancsot a

**SET RELATION TO ...**

parancs után kell elhelyezni. Az *egy-több* kapcsolatot a

**SET SKIP TO**

paranccsal (paraméterek nélkül) lehet megszüntetni.

**Program állományok létrehozása és alkalmazása**

A programokat a PRG- kiterjesztésű állományokban tároljuk.
**MODIFY COMMAND** [<PRG - állomány neve>]
parancs segítségével lehet létrehozni.
A parancs egy szövegszerkesztőt nyit meg, amelyben megszerkeszthetjük a program parancsait. A parancsokat új sorban kell kezdeni. Ha a parancs nem fér el egy sorban, akkor a sor végén ; karaktert kell elhelyezni, és a parancsot a következő sorban lehet folytatni.
A program a **MODIFY COMMAND** paranccsal módosítható. A program mentése a Ctrl+W billentyűkkel történik.

A létrehozott program futtatása-

**DO** <rutin> **[WITH** <paraméterlista>]

<rutin> - a program (eljárás) neve,
**WITH** <paraméterlista> - paraméterek, amelyeket kell átadni a programnak. A WITH paraméter akkor alkalmazható, ha a program a PARAMETERS parancsot tartalmazza. A

**PARAMETERS** <változólista>

parancs azokat a memóriaváltozókat tartalmazza, amelyek megkapja a paraméterek értékeit.
A program futtatása akkor fejeződik be, amikor a programvezérlés végrehajtja a RETURN, CANCEL, QUIT parancsokat vagy eléri a program utolsó parancsát. A programban elhelyezett DO parancs félbeszakítja a program futtatását és elindítja az eljárást.
A program futtatását az **Esc** billentyű lenyomásával abban az esetben lehet befejezni, ha ezt a SET ESCAPE parancs beállítása engedélyezi

**SET ESCAPE ON/OFF**

Ha az **ON** opció van beállítva, akkor a program futtatását az **Esc** billentyűvel meg lehet állítani, ha **OFF** opció van beállítva, akkor nem.

A program **megjegyzéseket** is tartalmazhat. A FoxPro egy teljes sort megjegyzésként kezel, ha a sor

**NOTE** vagy *

karakterrel kezdődik.
Ha a megjegyzést a parancs után ugyanabban a sorban akarjuk elhelyezni, akkor a megjegyzés előtt a **&&** karaktereket kell elhelyezni.

Egy EXE - állományt

**!** <EXE – állomány> vagy **RUN** <EXE – állomány>

paranccsal lehet elindítani.



**Programok szerkezete**

Nagyobb méretű programokat célszerű részekre bontani, amelyeket a rendszer eljárásokként (PROCEDURE) fog kezelni. Eljárásokat egymástól függetlenül lehet létrehozni és tárolni. Eljárást függvény (FUNCTION) formában is lehet szerkeszteni. Azt az eljárást, amely különálló programállományban tárolódik, **külső eljárásnak** nevezzük. Ha egy eljárás a programállomány része, akkor ez **belső eljárás**. Eljárásokat és függvényeket tartalmazó állományokat Procedure- állománynak is szoktak nevezni.

**Külső eljárás**
A külső eljárás különálló program-állományban tárolódik. Az eljárás futtatása DO parancs segítségével történik:

**DO** p **[WITH** <paraméter-lista>] **[IN** <f>]

p – a programállomány neve,
<paraméter-lista> - az eljárásnak átadott paraméterek listája,
**IN** <f> - egy másik állomány neve, amelyben a *p* eljárás van elhelyezve,
A kiterjesztések szempontjából az eljárás keresésének a sorrendje a következő: **EXE, APP, FXP, PRG.**

A WITH parancsban a <paraméter-lista> az aktuális paramétereket tartalmazza, a PARAMETERS parancs pedig a formális paramétereket. Az aktuális paraméterek száma kisebb lehet, mint a formális paraméterek száma.
Az eljárás futtatása akkor ér véget, ha a vezérlés eléri a következő parancsot:

- az eljárás utolsó parancsát (visszatérés a hívó programba),
- a **RETURN** parancsot (visszatérés a hívó programba),
- a **CANCEL** parancsot (a vezérlés visszatér az interaktív-parancs szintre),
- vagy a **QUIT** parancsot (kilép a FoxPro rendszerből).

A RETURN parancs paramétereket tartalmazhat:

**RETURN [TO MASTER/** <eljárás> / <kifejezés>]

**TO MASTER –** a vezérlést a főprogramnak adja vissza,
<eljárás> - a vezérlést az <eljárásba> adja vissza,
<kifejezés> - csak függvényekben alkalmazható. A függvény a <kifejezés> értékét adja vissza.
A RETURN parancs a vezérlést a hívó program DO utáni parancsának adja vissza.

**RETRY**

parancs a vezérlést a DO parancsnak adja vissza (amely elindította az eljárást). A RETRY parancsot a hibakezelés esetén célszerű alkalmazni, amikor a hiba feldolgozása után egy eljárást (parancsot) újból meg kell ismételni.

**Belső eljárás**
A DO parancs a külső eljárást minden alkalommal újból visszatölti a memóriába. Ezért gyorsabb a belső eljárás alkalmazása. Nagyobb programok esetén külső eljárásokat szoktak alkalmazni.
Belső eljárás első parancsa a

**PROCEDURE p**

parancs. p – az eljárás neve.