SQL alapok - Adatok beolvasásának lehetőségei

Tanulási folyamatok során sokszor lesz szükségünk már kész adatbázisra hiszen így könnyen tudjuk tesztelni a lekérdezéseket, könnyebben ellenőrizhetjük tudásunkat. Ezt megtehetjük a már látott INSERT INTO segítségével, de érezhetően nehézkes és körülményes.

CSV állomány tartalmának beolvasása

Ebben az esetben első lépésben el kell készítenünk a tábla szerkezetét.

```
1.  create table tanulo(
2.      tanuloAz int primary key auto_increment,
3.      nev varchar(100) not null,
4.      kg float,
5.      cm int
6.  );
```

Hozzuk létre a CSV állományt a teszteléshez (tanulo.csv).

```
1       Kiss Anna       160     66
2       Szep Geza       180.2   81
3       Joo Attila      178.3   78
```

Majd olvassuk be az adatokat az üres adattáblába a CSV állományból.

```
1.  load data infile 'tanulo.csv'
2.  into table tanulo;
```

Ez azt feltételezi, hogy az adatbázis mappájában található az adatokat tartalmazó állomány. Más esetben az állomány elérési útját kell használnunk.

Lehetőség van arra, hogy megadjuk a CSV szeparátorár is.

```
4,Kiss Anna,160,66
5,Szep Geza,180.2,81
6,Joo Attila,178.3,78
```

Ekkor az adatbeolvasást a következő alakban tehetjük meg.

```
1.  load data infile 'tanulo2.csv'
2.  into table tanulo
3.  fields terminated by ',';
```

Nem csak az oszlopok közötti elválasztó jelet adhatjuk meg, hanem a sorvéget jelet is beállíthatjuk az importálásnál.

```
4,Kiss Anna,160,66|5,Szep Geza,180.2,81|6,Joo Attila,178.3,78
```

Ekkor az importálás a következőképpen oldható meg.

```
1.  load data infile 'tanulo4.csv'
2.  into table tanulo
3.  fields terminated by ','
4.  ignore 1 lines;
```

Az is gyakori, hogy az első sorban szerepel az oszlopnak a neve.

```
tanuloAz,nev,kg,cm
4,Kiss Anna,160,66
5,Szep Geza,180.2,81
6,Joo Attila,178.3,78
```

Erre MySQL esetén nincs szükségünk így ezt figyelmen kívül kell hagyni.

```
1. load data infile 'tanulo4.csv'
2. into table tanulo
3. fields terminated by ','
4. ignore 1 lines;
```

Amennyiben nem áll rendelkezésre az összes oszlop adata, akkor lehetőség van megmondani, hogy mely oszlopokba történjen meg a művelet. Példánk esetén az azonosítót a rendszerre adja, így ez „hiányzik”. Például.

```
Kiss Anna,160,66
Szep Geza,180.2,81
Joo Attila,178.3,78
```

Ekkor a parancs a következőképpen módosul.

```
1. load data infile 'tanulo5.csv'
2. into table tanulo
3. fields terminated by ','
4. (nev,kg,cm);
```

Az adatbeolvasás részletes alakja.

```
1  LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
2      [REPLACE | IGNORE]
3      INTO TABLE tbl_name
4      [CHARACTER SET charset_name]
5      [{FIELDS | COLUMNS}
6          [TERMINATED BY 'string']
7          [[OPTIONALLY] ENCLOSED BY 'char']
8          [ESCAPED BY 'char']
9      ]
10     [LINES
11         [STARTING BY 'string']
12         [TERMINATED BY 'string']
13     ]
14     [IGNORE number {LINES | ROWS}]
15     [(col_name_or_user_var
16         [, col_name_or_user_var] ...)]
17     [SET col_name={expr | DEFAULT},
18         [, col_name={expr | DEFAULT}] ...]
```

XML állomány tartalmának beolvasása

Az adatok gyakran XML formátumban áll rendelkezésünkre. Az adatok beolvasása hasonlóan működik mint a CSV állományok esetén. Az adatokat három különböző módon adhatjuk meg.

- Az oszlopneveket és az értékeket, mint attribútum adjuk meg.

```
<row column1="value1" column2="value2" .../>
```

- Az oszlopneveket mint tag-ek adjuk meg.

```
<row>
  <column1>value1</column1>
  <column2>value2</column2>
</row>
```

- Az oszlopokat field tag-gel adjuk meg a name attribútum használatával.

```
<row>
  <field name='column1'>value1</field>
  <field name='column2'>value2</field>
</row>
```

A XML állomány beolvasásának alakja.

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE [db_name.]tbl_name
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<tagname>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var
        [, field_name_or_user_var] ...)]
    [SET col_name={expr | DEFAULT},
        [, col_name={expr | DEFAULT}] ...]
```

Adatbeolvasás SQL script segítségével

Sokszor rendelkezésünkre áll egy más/mások által készített adatbázis, adatsor. Ezeket a legtöbb esetben exportálhatjuk, átalakíthatjuk a nekünk megfelelő formátumba. Így kötegelve vihetjük be a használni kívánt adatokat a megfelelő adatstruktúrába. Ebben az esetben az SQL nyelv eltéréseire kell figyelnünk, hiszen az egyes adatbáziskezelő rendszerek között kisebb-nagyobb eltérések vannak. A mintában láthatjuk, hogy elsőként az adatbázist hozza létre a script, majd sorra jönnek az adattáblát (CREATE TABLE), illetve a az adatbevitel sorai (INSERT INTO). Az egyes formátumok kezelésére/konvertálására sok segédprogram és webes online lehetőség is elérhető.

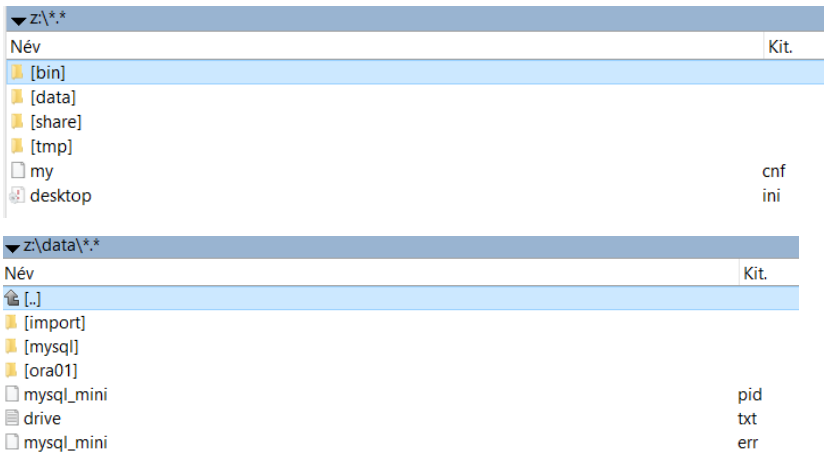
```
DROP DATABASE IF EXISTS `importScript`;
CREATE DATABASE IF NOT EXISTS `importScript`;
USE `importScript`;
#
# Table structure for table 'Alkalmazottak'
#
DROP TABLE IF EXISTS `Alkalmazottak`;

CREATE TABLE `Alkalmazottak` (
  `Törzsszám` INTEGER NOT NULL,
  `Név` VARCHAR(255),
  `Szület_idő` DATETIME,
  `Részleg` VARCHAR(255),
  `Beosztás` VARCHAR(255),
  `Belépés` INTEGER,
```

```
16.   `Alapbér` INTEGER,
17.   `Nyelvpótl` TINYINT(1) DEFAULT 0,
18.   PRIMARY KEY (`Törzsszám`)
19. ) ENGINE=myisam DEFAULT CHARSET=utf8;
20. SET autocommit=1;
21. #
22. # Dumping data for table 'Alkalmazottak'
23. #
24. INSERT INTO `Alkalmazottak` (`Törzsszám`, `Név`, `Szület_idő`, `Részleg`, `Beosztás`
    , `Belépés`, `Alapbér`, `Nyelvpótl`) VALUES (1, 'Kiss Éva', '1962-05-
    12 00:00:00', 'Munkaügy', 'Előadó', 1989, 68000, 0);
25. INSERT INTO `Alkalmazottak` (`Törzsszám`, `Név`, `Szület_idő`, `Részleg`, `Beosztás`
    , `Belépés`, `Alapbér`, `Nyelvpótl`) VALUES (2, 'Gál Péter', '1963-04-
    05 00:00:00', 'Igazgatás', 'Titkár', 1997, 110000, 1);
26. INSERT INTO `Alkalmazottak` (`Törzsszám`, `Név`, `Szület_idő`, `Részleg`, `Beosztás`
    , `Belépés`, `Alapbér`, `Nyelvpótl`) VALUES (3, 'Sas Gábor', '1965-11-
    05 00:00:00', 'Forgácsoló', 'Szakmunkás', 1979, 98500, 1);
```

Adatbázis másolása

Amennyiben lehetőségünk van az adatbázist tartalmazó/tároló szerver fájlaihoz hozzáférni, akkor MySQL esetén másolással is hordozhatjuk az adatbázisainkat. MySQL esetén minden adatbázis az data mappában egy külön mappában található. A mappa neve megegyezik az adatbázis nevével. Ezt a mappát átmásolva egy másik MySQL szerverre



használhatjuk a továbbiakban az „új” helyen is. A használt miniServer esetén az udrive mappában található a data mappa. Ez a mappa elérhető a map-pelt „Z” meghajtón is.

Az adatok esetén lehetőségünk van exportálásra is az előzőekben ismertetett alakhoz hasonlatosan. Elsőként készítsük el a lekérdezésünket, amit exportálni szeretnénk CSV állományba. Majd illesszük be azokat a kapcsolókat, amikkel megadhatjuk az állomány felépítéséhez szükséget paramétereket.

```
SELECT
    orderNumber, status, orderDate, requiredDate, comments
FROM
    orders
WHERE
    status = 'Cancelled'
INTO OUTFILE 'C:/tmp/cancelled_orders.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```

Az adatbázis szerkezetének módosítása - ALTER TABLE

Az adatbázisunk szerkezetét sokszor finomítanunk, bővítenünk, módosítanunk kell a megjelenő igények szerint. Az utasítás alakja:

```
ALTER TABLE táblanév
ADD …     -- pl. új oszlop, új megszorítások megadása
MODIFY … -- meglévő definíciós elemek megváltoztatása,
             megszorítások állapotának változtatása
DROP …    -- definíciós elemek eltávolítása
…;        -- egyéb változtatások
```

Példák

Új oszlop

```
ALTER TABLE hallgato
ADD telefon VARCHAR(10);
```

Oszlop módosítása

```
ALTER TABLE hallgato
MODIFY cim CHAR(20) DEFAULT 'Székesfehérvár' NOT NULL;
```

Külső kulcs törlése

```
ALTER TABLE hallgato
  DROP FOREIGN KEY szakAz;
```

Nézettábla létrehozása, törlése

A nézettábla más táblá(k)ból vagy nézettáblákból (SELECT utasítással) származtatott virtuális tábla. Segítségével növelhetjük az adatvédelmet, szabályozhatjuk az adathozzáférést. Összetett, bonyolultabb lekérdezések eredményét egyszerűbben kezelhetjük.

Általános alak.

```
CREATE [OR REPLACE] VIEW nézettáblanév [(oszlopnév,...)]
AS szelekció
[WITH CHECK OPTION] [CONSTRAINT megszorítás]
[WITH READ ONLY];
```

```
1.  DROP VIEW nézettábla;
```

Például.

```
1.  CREATE OR REPLACE VIEW bejarok
2.  AS SELECT * FROM hallgato
3.  WHERE cim != 'Székesfehérvár';
```

Indexek kezelése

Az indexek használatával a táblákon való keresések felgyorsíthatók. Ezek adminisztrációja viszont időigényes, így a változtatások végrehajtásához szükséges idő megnő. Egy relációhoz több index is megadható. Az indexes használata nem befolyásolja az SQL utasításainkat. Az indexeket a rendszer automatikusan kezeli és használja az adott esetben.

Index létrehozása

```
1.  CREATE INDEX indexNév
2.  ON tábla (oszlop1, oszlop2, ...);
```

Példa

```
1.  CREATE INDEX idxHallgatoNeve
2.  ON Hallgato (nev);
```

Több oszlopra vonatkozó index létrehozása.

```
1.  CREATE INDEX idx_Hallgato
2.  ON Hallgato (Vezeteknev, Keresztnev);
```

Amennyiben azt akarjuk, hogy az index értékek egyediek legyenek, akkor az UNIQUE kell használnunk.

Index törlése MySQL esetén.

```
1.  ALTER TABLE Hallgato
2.  DROP INDEX idxHallgatoNeve;
```

Álnevek/Aliases használata

Az álneveket a relációk, illetve a relációk oszlopainak ideiglenes elnevezése esetén használjuk. Azaz az álnevek csak a lekérdezés futása alatt léteznek. Az álnevek segítik az oszlopnevek „beszédesebb” hivatkozását.

Oszlop esetén

```
1.  SELECT oszlopNeve AS alnev
2.  FROM tablaNeve;
```

Tábla esetén

```
1.  SELECT oszlopNeve(k)
2.  FROM tablaNev AS alnev;
```

Példa

```
1. SELECT szulNev AS "Születési név"
2. FROM hallgato;
```

```
1. SELECT CONCAT(vezNev,' ',kerNev) AS "Név"
2. FROM hallgato;
```

```
1. SELECT nev
2. FROM hallgato AS h
3. WHERE h.nev LIKE "A%"
```