

# SQL haladó

Külső összekapcsolások,  
Csoportosítás/Összesítés,  
Beszúrás/Törlés/Módosítás,  
Táblák létrehozása/Kulcs  
megszorítások

# Külső összekapcsolás

- ◆ Összekapcsoljuk  $R$  és  $S$  relációkat:  $R \bowtie_c S$ .
- ◆  $R$  azon sorait, melyeknek nincs  $S$ -beli párja *lógó* soroknak nevezzük.
  - ◆  $S$ -nek is lehetnek lógó sorai.
- ◆ A külső összekapcsolás megőrzi a lógó sorokat NULL értékkel helyettesítve a hiányzó értékeket.

# Példa: külső összekapcsolás

R = (

A	B
1	2
4	5

S = (

B	C
2	3
6	7

(1,2) és (2,3) összekapcsolható, a másik két sor azonban „lóg”.

R OUTERJOIN S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7


# Külső összekapcsolás (SQL)

◆ R OUTER JOIN S: a külső összekapcsolásoknál mindig ez szerepel.

1. Opcionális NATURAL az OUTER előtt.
2. Opcionális ON <feltétel> JOIN után.
3. Opcionális LEFT, RIGHT, vagy FULL az OUTER előtt.

- ◆ LEFT = csak R lógó sorait őrzi meg.
- ◆ RIGHT = csak S lógó sorait őrzi meg.
- ◆ FULL = az összes lógó sort megőrzi.

Csak az egyik szerepelhet.



# Összesítések (aggregáció)

- ◆ SUM, AVG, COUNT, MIN, és MAX összesítő függvényeket a SELECT záradékban alkalmazhatjuk egy-egy oszlopra.
- ◆ COUNT(\*) az eredmény sorainak számát adja meg.

# Példa: Összesítés

- ◆ A Felszolgál(kocsmá, sör, ár) tábla segítségével adjuk meg a Bud átlagos árát:

```
SELECT AVG (ár)
FROM Felszolgál
WHERE sör = 'Bud' ;
```

# Ismétlődések kiküszöbölése összesítésben

- ◆ Az összesítő függvényen belül DISTINCT.
- ◆ **Példa:** hány *különféle* áron árulják a Bud sört?

```
SELECT COUNT(DISTINCT ár)  
FROM Felszolgál  
WHERE sör = 'Bud';
```

# NULL értékek nem számítanak az összesítésben

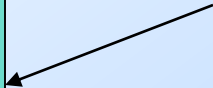
- ◆ NULL soha nem számít a SUM, AVG, COUNT, MIN, MAX függvények kiértékelésekor.
- ◆ De ha nincs NULL értéktől különböző érték az oszlopban, akkor az összesítés eredménye NULL.
  - ◆ **Kivétel:** COUNT az üreshalmazon 0-t ad vissza.



# Példa: NULL értékek összesítésben


```
SELECT count(*)  
FROM Felszolgál  
WHERE sör = 'Bud';
```

A Bud sört árusító  
kocsmák száma.



```
SELECT count(ár)  
FROM Felszolgál  
WHERE sör = 'Bud';
```

Azon kocsmák száma,  
ahol ismerjük a Bud sör  
árát.



# Csoportosítás

- ◆ Egy SELECT-FROM-WHERE kifejezést GROUP BY záradékkal folytathatunk, melyet attribútumok listája követ.
- ◆ A SELECT-FROM-WHERE eredménye a megadott attribútumok értékei szerint csoportosítódik, az összesítéseket ekkor minden csoportra külön alkalmazzuk.

# Példa: Csoportosítás

- ◆ A Felszolgál(kocsmá, sör, ár) tábla segítségével adjuk meg a sörök átlagos árát.

```
SELECT sör, AVG(ár)
FROM Felszolgál
GROUP BY sör;
```

beer	AVG(price)
Bud	2.33
Miller	2.45

# Példa: Csoportosítás



```
SELECT alkesz, AVG(ár)
FROM Látogat, Felszolgál
WHERE sör = 'Bud' AND
      Látogat.koccsma =
      Felszolgál.koccsma
```

```
GROUP BY alkesz;
```

Alkesz-  
koccsma-ár  
hármask a  
Bud sörre.

Alkeszek  
szerinti  
csoportosítás.

# A SELECT lista és az összesítések

- ◆ Ha összesítés is szerepel a lekérdezésben, a SELECT-ben felsorolt attribútumokra a következő érvényes
  1. Összesítések, amelyekben egy összesítési operátort alkalmazunk egy attribútumra vagy egy attribútumot tartalmazó kifejezésre. Ezek a kifejezések csoportonként kerülnek kiértékelésre.
  2. Attribútumok, amelyek a GROUP BY záradékban szerepelnek, mint a példában *al/kesz*. Egy összesítéseket tartalmazó SELECT záradékban csak a GROUP BY záradékban is megtalálható attribútumok jelenhetnek meg összesítési operátor nélkül.

# Helytelen lekérdezés

- ◆ Elsőre sokan gondolhatják azt, hogy az alábbi lekérdezés a Bud sört legolcsóbban áruló kocsmát adja vissza:  
*SELECT kocsmá, MIN(ár)*  
*FROM Felsőzolgál*  
*WHERE sör= 'Bud';*
- ◆ Valójában ez egy helytelen SQL lekérdezés.

# HAVING záradék

- ◆ A GROUP BY záradékot egy HAVING <feltétel> záradék követheti.
- ◆ Ebben az esetben a feltétel az egyes csoportokra vonatkozik, ha egy csoport nem teljesíti a feltételt, nem lesz benne az eredményben.

# Példa: HAVING


- ◆ A Felszolgál(kocsmá, sör, ár) és Sörök(név, gyártó) táblák felhasználásával adjuk meg az átlagos árát azon söröknek, melyeket legalább három kocsmában felszolgálnak, vagy Pete a gyártójuk.



# Megoldás

```
SELECT sör, AVG(ár)
FROM Felszolgál
GROUP BY sör
```


Sör csoportok, melyeket  
legalább 3 nem-NULL  
kocsmában árulnak, vagy  
Pete a gyártójuk.



```
HAVING COUNT(kocsmas) >= 3 OR
```

```
sör IN (SELECT név
FROM Sörök
WHERE gyártó = 'Pete');
```

Sörök,  
melyeket  
Pete gyárt.



# A HAVING feltételére vonatkozó megszorítások

- ◆ Az alkérdésre nincs megszorítás.
- ◆ Az alkérdésen kívül csak olyan attribútumok szerepelhetnek, amelyek:
  1. vagy csoportosító attribútumok,
  2. vagy összesített attribútumok.(Azaz ugyanazok a szabályok érvényesek, mint a SELECT záradéknál).
- ◆ A HAVING záradékban hivatkozott összesítés csak az éppen feldolgozott csoport soraira vonatkozik.
- ◆
  - A FROM záradékban megadott relációk bármely attribútumára képezhetünk a HAVING záradékban összesítést, összesítés nélkül a HAVING záradékban csak azok az attribútumok fordulhatnak elő, amelyek a GROUP BY listában is szerepeltek. (Ugyanaz a szabály, mint ami a SELECT záradékra is vonatkozott.)

# Adatbázis módosítások

- ◆ A *módosító* utasítások nem adnak vissza eredményt, mint a lekérdezések, hanem az adatbázis tartalmát változtatják meg.
- ◆ 3-féle módosító utasítás létezik:
  1. *Insert* (beszúrás).
  2. *Delete* (törlés).
  3. *Update* (létező sorok értékeinek módosítása).
- ◆ Data Manipulation Language (DML).

# Beszúrás

- ◆ Ha egyetlen sort szúrunk be:

```
INSERT INTO <reláció>
```

```
VALUES ( <attribútum lista> );
```

- ◆ **Példa:** a Szeret(alkesz, sör) táblában rögzítjük, hogy Zsuzska szereti a Bud sört.

```
INSERT INTO Szeret
```

```
VALUES ( 'Zsuzska' , 'Bud' );
```

# Az INSERT-nél megadhatjuk az attribútumokat

- ◆ A reláció neve után megadhatjuk az attribútumait.
- ◆ Ennek két oka lehet:
  1. elfelejtettük, hogy a reláció definíciójában, milyen sorrendben szerepeltek az attribútumok.
  2. Nincs minden attribútumnak értéke, és azt szeretnénk, ha a hiányzó értékeket NULL vagy default értékkel helyettesítenék.

# Példa: Attribútumok megadása

```
INSERT INTO Szeret (sör, alkesz)  
VALUES ('Bud', 'Zsuzska');
```

# Default (alapértelmezett) értékek megadása

- ◆ A CREATE TABLE utasításban az oszlopnevet DEFAULT kulcsszó követheti és egy érték.
- ◆ Ha egy beszúrt sorban hiányzik az adott attribútum értéke, akkor a default értéket kapja.

# Példa: Default (alapértelmezett) érték

```
CREATE TABLE Alkeszek (  
    név CHAR(30) PRIMARY KEY,  
    cím CHAR(50)  
        DEFAULT '123 Sesame St.',  
    telefon CHAR(16)  
);
```



## Példa: Default (alapértelmezett) értékek

```
INSERT INTO Alkeszek (név)  
VALUES ( ' Zsuzska' ) ;
```

Az eredmény sor:

név	cím	telefon
Zsuzska	123 Sesame St	NULL

# Több sor beszúrása

- ◆ Egy lekérdezés eredményét is beszúrhatjuk a következő módon:

```
INSERT INTO <reláció>  
( <alkérdés> );
```

# Példa: Beszúrás alkérdéssel

- ◆ A Látogat(alkesz, kocsmá) tábla felhasználásával adjuk hozzá a Szesztestvérek(név) táblához Zsuzska szesztestvéreit, vagyis azokat, akikkel legalább egy közös kocsmát látogatnak.

A szesztestvér.

# Megoldás

Alkesz párok: az első Zsuzska, a második nem Zsuzska, de van olyan kocsmá, amit mindketten látogatnak.

```
INSERT INTO Szesztestvérek
```

```
(SELECT I2.alkesz
```

```
FROM Látogat I1, Látogat I2  
WHERE I1.alkesz = 'Zsuzska' AND  
      I2.alkesz <> 'Zsuzska' AND  
      I1.kocsmá = I2.kocsmá
```

```
);
```

# Törlés

- ◆ A törlendő sorokat egy feltétel segítségével adjuk meg:

```
DELETE FROM <reláció>  
WHERE <feltétel>;
```

# Példa: Törlés

```
DELETE FROM Szeret  
WHERE alkesz = 'Zsuzska' AND  
      sör = 'Bud';
```

# Példa: Az összes sor törlése

```
DELETE FROM Likes;
```

# Példa: Több sor törlése

- ◆ A Sörök(név, gyártó) táblából töröljük azokat a söroket, amelyekhez létezik olyan sör, amit ugyanaz a cég gyártott.

```
DELETE FROM Sörök s
WHERE EXISTS (
  SELECT név FROM Sörök
  WHERE gyártó = s.gyártó
    AND név <> s.név);
```

Azok a sörök,  
amelyeknek ugyanaz  
a gyártója, mint az *s*  
éppen aktuális  
sorának, a nevük  
viszont különböző.



# A törlés szemantikája --- (1)

- ◆ Tegyük fel, hogy az Anheuser-Busch csak Bud és Bud Lite söröket gyárt.
- ◆ Tegyük fel még, hogy  $s$  sorai közt a Bud fordul elő először.
- ◆ Az alkérdés nem üres, a későbbi Bud Lite sor miatt, így a Bud törlődik.
- ◆ Kérdés, hogy a Bud Lite sor törlődik-e?

# A törlés szemantikája --- (2)

- ◆ **Válasz:** igen, a Bud Lite sora is törlődik.
- ◆ A törlés ugyanis két lépésben hajtódik végre.
  1. Kijelöljük azokat a sorokat, amelyekre a WHERE feltétele teljesül.
  2. Majd töröljük a kijelölt sorokat.

# Módosítás

- ◆ Bizonyos sorok bizonyos attribútumainak módosítása.

UPDATE <reláció>

SET <attribútum értékadások listája>

WHERE <sorokra vonatkozó feltétel>;

# Példa: Módosítás

- ◆ Fecó telefonszámát 555-1212-re változtatjuk (Fecó itt egy alkesz):

```
UPDATE Alkeszek  
SET telefon = '555-1212'  
WHERE név = 'Fecó';
```

# Példa: Több sor módosítása

- ◆ Legfeljebb 4 dollárba kerülhessenek a sörök:

```
UPDATE Felszolgal  
SET ár = 4.00  
WHERE ár > 4.00;
```

# Adatbázis sémák SQL-ben

- ◆ Data Definition Language (DDL), az SQL nyelv része, ennek segítségével hozhatunk létre adatobjektumokat, deklarálhatunk megszorításokat stb.

# Relációk létrehozása

- ◆ A legegyszerűbb forma:

```
CREATE TABLE <név> (  
    <elemek listája>  
);
```

- ◆ Relációk törlése:

```
DROP TABLE <név>;
```

# Tábla definíciók elemei

- ◆ A legegyszerűbb elem: az attribútum és annak típusa.
- ◆ A legfontosabb típusok a következők:
  - ◆ INT vagy INTEGER (szinonimák).
  - ◆ REAL vagy FLOAT (szinonimák).
  - ◆ CHAR( $n$ ) = rögzített hosszúságú sztring  $n$  karakter hosszú.
  - ◆ VARCHAR( $n$ ) = változó hosszúságú sztring legfeljebb  $n$  karakter hosszú.



## Példa : Create Table

```
CREATE TABLE Felszolgal (
    kocsmas CHAR(20),
    sor VARCHAR(20),
    ar REAL
);
```

# SQL értékek

- ◆ Az egészek és lebegőpontos típusú konstansokat csak „szimplán le kell írni”. (Pont jelöli a tizedesvesszőt.)
- ◆ A sztringek esetében aposztrófok közé kell tennünk a konstansokat.
  - ◆ Két aposztróf = egyetlen aposztróf,  
például: ' Joe' ' s Bar' .
- ◆ Minden érték lehet NULL is.

# Dátum és idő

- ◆ A DATE és TIME külön típusok az SQL-ben.

- ◆ A dátum típus formátuma:

DATE 'yyyy-mm-dd'

- ◆ **Példa:** DATE '2007-09-30' (2007. szeptember 30.)

# Idő típus

- ◆ Az TIME típus formátuma:

TIME 'hh:mm:ss'

Opcionálisan tizedespont is következhet a másodpercek után.

- ◆ **Példa:** TIME '15:30:02.5' = fél négy múlt két és fél másodperccel.

# Kulcsok megadása (deklarálása)

- ◆ Egy attribútumot vagy attribútum listát kulcsként deklarálhatunk (PRIMARY KEY vagy UNIQUE).
- ◆ Mindkét formája a megszorításnak azt követeli meg, hogy relációnak ne legyen két olyan sora, melyek megegyeznek a kulcs attribútumokon.
- ◆ A különbségekről később lesz szó.

# Egy attribútumos kulcs deklarációja

- ◆ PRIMARY KEY vagy UNIQUE kulcsszót írhatjuk közvetlenül az attribútum mögé.

- ◆ Példa:

```
CREATE TABLE Sörök (  
    név          CHAR(20)  UNIQUE,  
    gyártó       CHAR(20)  
);
```

# Több attribútumú kulcsok megadása

- ◆ A kulcs deklaráció a CREATE TABLE utasítás egy eleme is lehet az attribútum-típus elemek után.
- ◆ Több attribútumú kulcsokat csak ebben a formában deklarálhatunk.
  - ◆ Ugyanakkor az egyetlen attribútumból álló kulcsokat is megadhatjuk ily módon.

# Példa: Több attribútumú kulcs

```
CREATE TABLE Felszolgal (
    kocsmasma CHAR(20),
    sör        VARCHAR(20),
    ár         REAL,
    PRIMARY KEY (kocsmasma, sör)
);
```



# PRIMARY KEY vs. UNIQUE

1. Egy relációhoz egyetlen PRIMARY KEY tartozhat és több UNIQUE megszorítás.
2. A PRIMARY KEY egyetlen attribútuma sem kaphat NULL értéket. A UNIQUE megszorításnál szerepelhetnek NULL értékek egy soron belül akár több is.