

Adatbázisok 1.

SQL haladó – 1. rész

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

Összekapcsolás típusok

- Descartes-szorzat (CROSS JOIN)
- Belső összekapcsolás (NATURAL JOIN, INNER JOIN...)
- Külső összekapcsolás (OUTER JOIN)

Belső összekapcsolások (*inner join*)

- Természetes összekapcsolás
- Théta összekapcsolás
- Félig összekapcsolás (*semi join*, jelölés: \bowtie):
 - $R(A_1, \dots, A_n), S(B_1, \dots, B_m)$ sémájú táblák esetén: $R \bowtie S \equiv \Pi_{A_1, A_2, \dots, A_n}(R \mid X \mid S)$
- Anti összekapcsolás (*anti join*, jelölés: \triangleright):
 - $R \triangleright S \equiv R - R \bowtie S$

Relációs algebra és az SQL

R

| A | B |
|----|---|
| a1 | 1 |
| a2 | 2 |

$R \bowtie S$

| R.A | B |
|-----|---|
| a1 | 1 |

$R \triangleright S$

| R.A | B |
|-----|---|
| a2 | 2 |

S

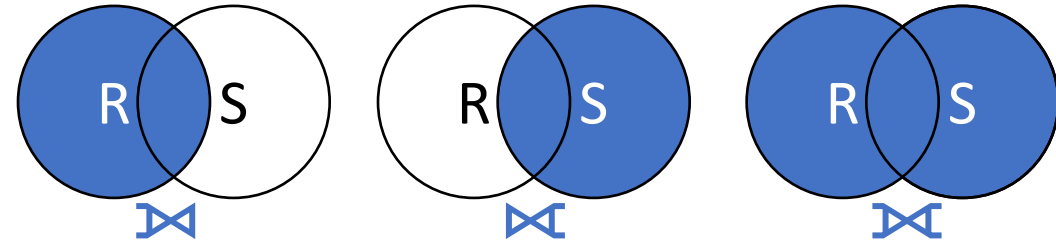
| A | C |
|----|---|
| a1 | 2 |
| a3 | 3 |

```
SELECT R.* FROM R
WHERE EXISTS(
  SELECT * FROM S
  WHERE R.A = S.A);
```

```
SELECT R.* FROM R
WHERE NOT EXISTS(
  SELECT * FROM S
  WHERE R.A = S.A);
```

Külső összekapcsolás (*outer join*)

Könyvbeli jelölés:



- Kiterjesztett relációs algebra
- Összekapcsoljuk **R** és **S** relációkat: $R \bowtie_c S$, $R \bowtie_c S$, $R \bowtie_c S$
 - A „természetes” változatnál nincs az alsó indexben „C”.
- Három különböző művelet
- **R** azon sorait, melyeknek nincs **S**-beli párja *lógó* soroknak (*dangling tuples*) nevezzük.
 - **S**-nek is lehetnek lógó sorai.
- A külső összekapcsolás megőrzi a lógó sorokat NULL értékkel helyettesítve a hiányzó értékeket.

Külső összekapcsolás (SQL)

- R <típus> OUTER JOIN S: a külső összekapcsolásoknál mindig ez szerepel.
 1. Opcionális NATURAL az OUTER előtt. ← Csak az egyik szerepelhet.
 2. Opcionális ON <feltétel> az S után. ←
 3. 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.

Példa: Külső összekapcsolások

(Kiterjesztett relációs algebra és az SQL)

R

| A | B |
|----|---|
| a1 | 1 |
| a2 | 2 |

R ⋈_{R.A=S.A} **S**

| R.A | B | S.A | C |
|-----|---|------|------|
| a1 | 1 | a1 | 2 |
| a2 | 2 | NULL | NULL |

R ⋈_{R.A=S.A} **S**

| R.A | B | S.A | C |
|------|------|-----|---|
| a1 | 1 | a1 | 2 |
| NULL | NULL | a3 | 3 |

R ⋈_{R.A=S.A} **S**

| R.A | B | S.A | C |
|------|------|------|------|
| a1 | 1 | a1 | 2 |
| a2 | 2 | NULL | NULL |
| NULL | NULL | a3 | 3 |

S

| A | C |
|----|---|
| a1 | 2 |
| a3 | 3 |

```
SELECT * FROM R  
LEFT OUTER JOIN S  
ON R.A = S.A;
```

```
SELECT * FROM R  
RIGHT OUTER JOIN S  
ON R.A = S.A;
```

```
SELECT * FROM R  
FULL OUTER JOIN S  
ON R.A = S.A;
```

Példa: Külső összekapcsolások

(Kiterjesztett relációs algebra és az SQL)

R

| A | B |
|----|---|
| a1 | 1 |
| a2 | 2 |

R ⋈ S

| A | B | C |
|----|---|------|
| a1 | 1 | 2 |
| a2 | 2 | NULL |

```
SELECT * FROM R  
NATURAL LEFT OUTER JOIN S;
```

R ⋈ S

| A | B | C |
|----|------|---|
| a1 | 1 | 2 |
| a3 | NULL | 3 |

```
SELECT * FROM R  
NATURAL RIGHT OUTER JOIN S;
```

S

| A | C |
|----|---|
| a1 | 2 |
| a3 | 3 |

R ⋈ S

| A | B | C |
|----|------|------|
| a1 | 1 | 2 |
| a2 | 2 | NULL |
| a3 | NULL | 3 |

```
SELECT * FROM R  
NATURAL FULL OUTER JOIN S;
```


Összesítések, azaz aggregációk (*aggregations*)

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

Példa: Összesítés

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgal(teázó, tea, ár)

Látogat(vendég, teázó)

- A Felszolgal(teázó, tea, ár) tábla segítségével adjuk meg a Brisk átlagos árát:

```
SELECT AVG(ár)
FROM Felszolgal
WHERE tea = 'Brisk';
```

Ismétlődések kiküszöbölése összesítésben

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- Az összesítő függvényen belül DISTINCT.
- **Példa:** hány *különféle* áron árulják a Brisk teát?

```
SELECT COUNT(DISTINCT ár)
FROM Felszolgál
WHERE tea = 'Brisk';
```

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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)


Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

```
SELECT count(*)  
FROM Felszolgál  
WHERE tea = 'Brisk';
```

A Brisk teát árusító
teázók száma.



```
SELECT count(ár)  
FROM Felszolgál  
WHERE tea = 'Brisk';
```

Példa: NULL értékek összesítésben

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)


```
SELECT count(*)  
FROM Felszolgál  
WHERE tea = 'Brisk';
```

A Brisk teát árusító
teázók száma.



```
SELECT count(ár)  
FROM Felszolgál  
WHERE tea = 'Brisk';
```

Azon teázók száma,
ahol ismerjük a Brisk tea
árát.



Csoportosítás (*grouping*)

- 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(teázó, tea, ár)** tábla segítségével adjuk meg a teák átlagos árát.

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

| tea | AVG(ár) |
|---------|---------|
| Brisk | 2.33 |
| Pyramid | 2.45 |

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

Példa: Csoportosítás

- ```
SELECT vendég, AVG(ár)
FROM Látogat, Felszolgál
WHERE tea = 'Brisk' AND
 Látogat.teázó =
 Felszolgál.teázó
GROUP BY vendég;
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

# Példa: Csoportosítás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- SELECT vendég, AVG(ár)

```
FROM Látogat, Felszolgál
WHERE tea = 'Brisk' AND
 Látogat.teázó =
 Felszolgál.teázó
```

GROUP BY vendég;

Vendég-  
teázó-ár  
hármask a  
Brisk teára.

# Példa: Csoportosítás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- SELECT vendég, AVG(ár)

```
FROM Látogat, Felszolgál
WHERE tea = 'Brisk' AND
 Látogat.teázó =
 Felszolgál.teázó
```

```
GROUP BY vendég;
```

Vendég-  
teázó-ár  
hármask a  
Brisk teára.

Vendégek  
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 a *vendég*. Egy összesítéseket tartalmazó SELECT záradékban csak a GROUP BY záradékban is megtalálható attribútumok jelenhetnek meg összesítési operátor nélkül.

# Helytelen lekérdezés (*illegal query*)

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- Elsőre sokan gondolhatják azt, hogy az alábbi lekérdezés a Brisk teát legolcsóbban áruló teázót adja vissza:
- ```
SELECT teázó, MIN(ár)  
FROM Felhasználó  
WHERE tea = 'Brisk';
```
- 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

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

- A Felszolgál(teázó, tea, ár) és Teák(név, gyártó) táblák felhasználásával adjuk meg az átlagos árát azon teáknak, melyeket legalább három teázóban felszolgálnak, vagy Pete a gyártójuk.

Megoldás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

SELECT tea, AVG(ár)

FROM Felszolgál

GROUP BY tea

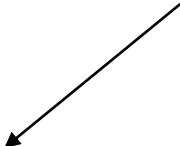
HAVING COUNT(teázó) >= 3 OR

tea IN (SELECT név

FROM Teák

WHERE gyártó = 'Pete');

Tea csoportok, melyeket
legalább 3 nem-NULL
teázóban árulnak



Megoldás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

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

Tea csoportok, melyeket
legalább 3 nem-NULL
teázóban árulnak

```
HAVING COUNT(teázó) >= 3 OR
```

```
tea IN (SELECT név
        FROM Teák
        WHERE gyártó = 'Pete');
```

Teák,
melyeket
Pete gyárt.

A HAVING feltételére vonatkozó megszorítások

- Az alkérdésre nincs megszorítás.
- Az alkérdésen kívül ugyanazok a szabályok érvényesek, mint a SELECT záradéknál
- 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 záradéokra is vonatkozott.)
- A HAVING záradékban hivatkozott összesítés csak az éppen feldolgozott csoport soraira vonatkozik.