

Adatbázisok 1.

SQL bevezetés – 2. rész

Select-From-Where záradékok

Több relációt tartalmazó lekérdezések

Alkérdezések

Többrelációs lekérdezések (*multirelation queries*)

- Általában több táblából kell kinyernünk az adatokat.
- Ekkor a relációkat a FROM záradékban kell felsorolnunk.
- Az azonos attribútum neveket az alábbi módon különböztetjük meg egymástól: “<reláció>.<attribútum>” .

Példa: két reláció összekapcsolása

```
SELECT tea
FROM Szeret, Látogat
WHERE teázó = 'Joe teázója' AND
      Látogat.vendég = Szeret.vendég;
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

Formális szemantika (*formal semantics*)

- Majdnem ugyanaz, mint korábban:
 1. Vegyük a FROM záradékban szereplő relációk Descartes-szorzatát.
 2. Alkalmazzuk a WHERE záradék feltételét.
 3. Vetítsünk a SELECT záradék oszlopaira.

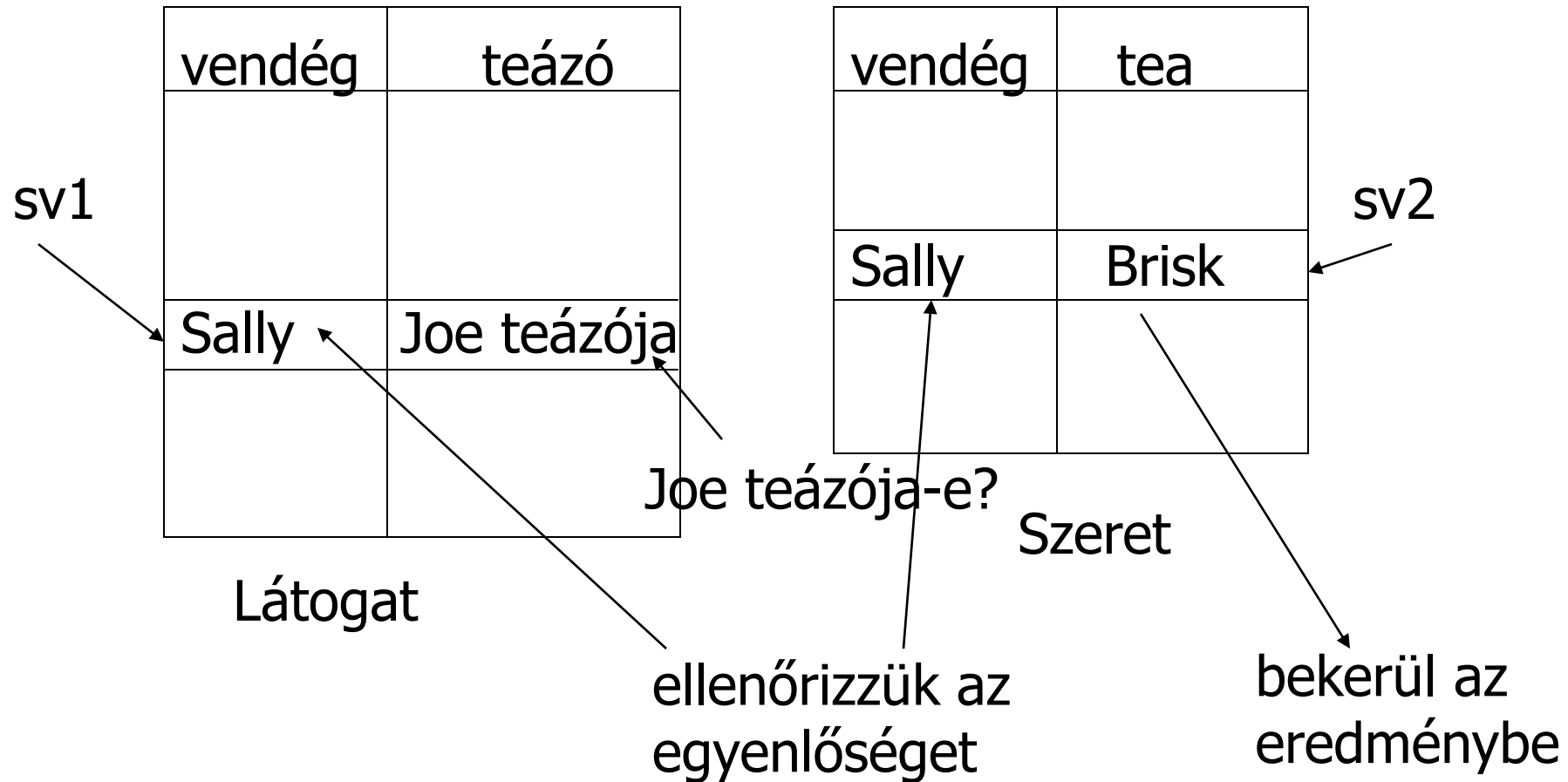
Működési szemantika (*operational semantics*)

- Képzeli úgy, mintha minden FROM záradékbeli táblához tartozna egy sorváltozó.
 - Ezekkel a sorok összes lehetséges kombinációját vesszük.
- Ha a sorváltozók a WHERE záradékot kielégítő sorra mutatnak, küldjük el ezeket a sorokat a SELECT záradékba.

Példa

```
SELECT tea  
FROM Szeret, Látogat  
WHERE teázó = 'Joe teázója' AND  
Látogat.vendég = Szeret.vendég;
```

Teák(név, gyártó)
Teázók(név, cím, engedélySzám)
Vendégek(név, cím, telefon)
Szeret(vendég, tea)
Felszolgál(teázó, tea, ár)
Látogat(vendég, teázó)



Explicit sorváltóók

- Esetenként egy tábla több példányára is szükségünk van.
- A FROM záradékban a relációk neve után adjuk meg a hozzájuk tartozó sorváltóók nevét.
- Egy relációt mindig átnevezhetünk ily módon, akkor is, ha egyébként nincs rá szükség.

Példa: önmagával vett összekapcsolás (*self-join*)

```
SELECT b1.név, b2.név  
FROM Teák b1, Teák b2  
WHERE b1.gyártó = b2.gyártó AND  
       b1.név < b2.név;
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

Alkérdeések (*subqueries*)

- A FROM és WHERE záradékban zárójelezett SELECT-FROM-WHERE utasításokat (*alkérdés*) is használhatunk.
- **Példa:** a FROM záradékban a létező relációk mellett, alkérdéssel létrehozott ideiglenes táblát is megadhatunk.
 - Ilyenkor a legtöbb esetben explicite meg kell adnunk a sorváltozó nevét.

Példa: alkérdés FROM-ban

- Keressük meg a Joe teázójának vendégei által kedvelt teákat!

Vendégek, akik látogatják
Joe teázóját.

SELECT tea

FROM Szeret, (SELECT vendég

FROM Látogat

WHERE teázó = 'Joe teázója') JD

WHERE Szeret.vendég = JD.vendég;

Egy sort visszaadó alkérdések

- Ha egy alkérdés biztosan egy sort ad vissza eredményként, akkor úgy használható, mint egy konstans érték.
 - Általában az eredmény sornak egyetlen oszlopa van.
 - Futásidejű hiba keletkezik, ha az eredmény nem tartalmaz sort, vagy több sort tartalmaz.

Példa: egysoros alkérdés (*single-tuple subquery*)

- A **Felhasználó(teázó, tea, ár)** táblában keressük meg azokat a teázókat, ahol a Pyramid ugyanannyiba kerül, mint Joe teázójában a Brisk.
- Két lekérdezésre biztos szükségünk lesz:
 1. Mennyit kér Joe a Briskért?
 2. Melyik teázókban adják ugyanennyiért a Pyramidot?

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

Kérdés + alkérdés

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

```
SELECT teázó  
FROM Felszolgál  
WHERE tea = 'Pyramid' AND
```

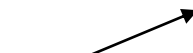
```
ár = (SELECT ár
```

```
FROM Felszolgál
```

```
WHERE teázó = 'Joe teázója'
```

```
AND tea = 'Brisk');
```

Ennyit kér
Joe a Briskért.



Az IN művelet

- < sor > IN (< alkérdés >) igaz, akkor és csak akkor, ha a sor eleme az alkérdés eredményének.
 - Tagadás: < sor > NOT IN (< alkérdés >).
- Az IN-kifejezések a WHERE záradékban jelenhetnek meg.

Példa: IN

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

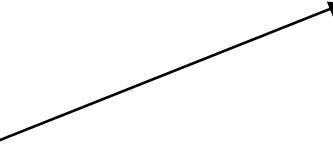
Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

```
SELECT *  
FROM Teák  
WHERE név IN (SELECT tea  
FROM Szeret  
WHERE vendég = 'Fred');
```

A teák,
melyeket Fred
kedvel.



Mi a különbség?

```
SELECT a  
FROM R, S  
WHERE R.b = S.b;
```

```
SELECT a  
FROM R  
WHERE b IN (SELECT b FROM S);
```


IN az R soraira vonatkozó predikátum (*predicate*)

```
SELECT a  
FROM R  
WHERE b IN
```

Két 2 érték.

```
(SELECT b FROM S);
```

Egy ciklus R sorai
fölött.

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) kielégíti a
feltételt;
1 egyszer jelenik
meg az
eredményben.

Itt R és S sorait párosítjuk

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```

Dupla ciklus R és S
sorai fölött

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) és (2,5)
(1,2) és (2,6)
is kielégíti a
feltételt;
1 kétszer kerül
be az eredménybe.

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

Az EXISTS művelet

- EXISTS(<alkérdés>) akkor és csak akkor igaz, ha az alkérdés eredménye nem üres.
- **Példa:** A **Teák(név, gyártó)** táblában keressük meg azokat a teákat, amelyeken kívül a gyártójuk nem gyárt másikat.

Példa: EXISTS

```
SELECT név  
FROM Teák b1  
WHERE NOT EXISTS (
```

Azon b1
teáktól
különböző
teák,
melyeknek
ugyanaz
a gyártója.

```
SELECT *  
FROM Teák  
WHERE gyártó = b1.gyártó AND  
név <> b1.név);
```

Változók láthatósága: itt a
a gyártó a legközelebbi
beágyazott FROM-beli táblából
való, aminek van ilyen
attribútuma.

Korrelált alkérdés

A „nem
egyenlő”
művelet
SQL-ben.

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

Az ANY művelet

- $x = \text{ANY}(\langle \text{alkérdés} \rangle)$ akkor és csak akkor igaz, ha x egyenlő az alkérdés legalább egy sorával.
 - = helyett bármilyen aritmetikai összehasonlítás szerepelhet.
- **Példa:** $x > \text{ANY}(\langle \text{alkérdés} \rangle)$ akkor igaz, ha x az alkérdés legkisebb eleménél nagyobb.
 - Itt az alkérdés sorai egy mezőből állnak.

Az ALL művelet

- $x \neq \text{ALL}(\langle \text{alkérdés} \rangle)$ akkor és csak akkor igaz, ha x az alkérdés egyetlen sorával sem egyezik meg.
- \neq helyett tetszőleges összehasonlítás szerepelhet.
- **Példa:** $x \geq \text{ALL}(\langle \text{alkérdés} \rangle)$ x az alkérdés eredményének maximum értékével azonos, vagy nagyobb nála.

Példa: ALL

```
SELECT tea  
FROM Felszolgal
```

```
WHERE ár >= ALL(  
    SELECT ár  
    FROM Felszolgal);
```

A külső lekérdezés
Felszolgaljának teája
egyetlen alkérdésbeli
teánál sem lehet
olcsóbb.

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgal(teázó, tea, ár)

Látogat(vendég, teázó)

Unió, metszet, különbség

- A szintaxis:
 - (<alkérdés>) UNION (<alkérdés>)
 - (<alkérdés>) INTERSECT (<alkérdés>)
 - (<alkérdés>) MINUS (<alkérdés>)
- MINUS helyett EXCEPT is szerepelhet.

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

Példa: metszet

- A Szeret(vendég, tea), Felszolgál(teázó, tea, ár) és Látogat(vendég, teázó) táblák segítségével keressük meg azon vendégeket és teákat:
 1. ahol a vendég szereti az adott teát,
 2. a vendég legalább egy olyan teázót látogat, ahol felszolgálják a szóban forgó teát.

Megoldás

Az alkérdés
egy tárolt
táblát ad
vissza.

```
(SELECT * FROM Szeret)
```

INTERSECT

A vendég látogatja
azt a teázót, ahol
felszolgálják azt a
teát.

```
(SELECT vendég, tea  
FROM Felszolgál, Látogat  
WHERE Felszolgál.teázó =  
Látogat.teázó);
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgál(teázó, tea, ár)

Látogat(vendég, teázó)

Multihalmaz szemantika (*bag semantics*)

- A SELECT-FROM-WHERE állítások multihalmaz szemantikát használnak, a halmazműveleteknél mégis a halmaz szemantika (*set semantics*) az érvényes.
 - Azaz sorok nem ismétlődnek az eredményben.

Motiváció: hatékonyság (*efficiency*)

- Ha projektálunk, akkor egyszerűbb, ha nem töröljük az ismétlődéseket (*duplicates*).
 - Csak szépen végigmegyünk a sorokon.
- A metszet, különbség számításakor általában az első lépésben lerendezik a táblákat.
 - Ez után az ismétlődések kiküszöbölése már nem jelent extra számításigényt.

Ismétlődések kiküszöbölése (*duplicate elimination*)

- Mindenképpen törlődjenek az ismétlődések: `SELECT DISTINCT . . .`
- Ne törlődjenek az ismétlődések:
pl: `SELECT ALL . . .` vagy
`. . . UNION ALL . . .`

Példa: DISTINCT

```
SELECT DISTINCT ár  
FROM Felszolgal;
```

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felszolgal(teázó, tea, ár)

Látogat(vendég, teázó)

Példa: ALL

- A **Látogat(vendég, teázó)** és **Szeret(vendég, tea)** táblák felhasználásával:

```
(SELECT vendég FROM Látogat)
```

```
EXCEPT ALL
```

```
(SELECT vendég FROM Szeret);
```

- Kilistázza azokat a vendégeket, akik több teázót látogatnak, mint amennyi teát szeretnek, és a két leszámlálás különbsége azt mutatja, hogy mennyivel több teázót látogatnak mint amennyi teát kedvelnek.

Összekapcsolás *(join)* kifejezések

- Az SQL-ben számos változata megtalálható az összekapcsolásoknak.
- Ezek a kifejezések önmagukban is állhatnak lekérdezésként (*stand-alone queries*), vagy a FROM záradékban is megjelenhetnek.

Descartes szorzat és természetes összekapcsolás

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasználó(teázó, tea, ár)

Látogat(vendég, teázó)

- Természetes összekapcsolás:

`R NATURAL JOIN S;`

- Szorzat:

`R CROSS JOIN S;`

- Példa:

`Szeret NATURAL JOIN Felhasználó;`

- A relációk helyén zárójellezett alkérdések is szerepelhetnek.

Théta-összekapcsolás

- R JOIN S ON <feltétel>
- **Példa:** az **Vendégek(név, cím)** és **Látogat(vendég, teázó)** táblákból:

```
Vendégek JOIN Látogat ON  
    név = vendég;
```

azokat (v, c, v, t) négyeseket adja vissza, ahol a v vendég c címen lakik és a t teázót látogatja.

Teák(név, gyártó)

Teázók(név, cím, engedélySzám)

Vendégek(név, cím, telefon)

Szeret(vendég, tea)

Felhasznál(teázó, tea, ár)

Látogat(vendég, teázó)

Relációs algebra és az SQL

R

A	B
a1	1
a2	2
a3	3
a3	2

$\sigma_{A='a3' \wedge B>2}(R)$

A	B
a3	3

```
SELECT * FROM R
WHERE A='a3' AND B>2;
```

$\Pi_A(\sigma_{A='a1'}(R))$

A
a1

```
SELECT A FROM R
WHERE A='a1';
```

RUS

A	B
a1	1
a2	2
a3	3
a3	2
a2	3
a3	1

S

A	B
a1	1
a2	3
a3	1

$R \cap S$

A	B
a1	1

```
(SELECT * FROM R)
INTERSECT
(SELECT * FROM S);
```

$R - S$

A	B
a2	2
a3	3
a3	2

```
(SELECT * FROM R)
EXCEPT
(SELECT * FROM S);
```

```
(SELECT * FROM R)
UNION
(SELECT * FROM S);
```

Relációs algebra és az SQL

R

A	B
a1	1
a2	2
a3	3
a3	2

S

A	B
a1	1
a2	3
a3	1

```
SELECT * FROM R, S;
```

vagy

```
SELECT * FROM R CROSS JOIN S;
```

R × S

R.A	R.B	S.A	S.B
a1	1	a1	1
a1	1	a2	3
a1	1	a3	1
a2	2	a1	1
a2	2	a2	3
a2	2	a3	1
a3	3	a1	1
a3	3	a2	3
a3	3	a3	1
a3	2	a1	1
a3	2	a2	3
a3	2	a3	1

Relációs algebra és az SQL

R

A	B
a1	1
a2	2
a3	3
a3	2

S

A	B
a1	1
a2	3
a3	1

R|X|S

A	B
a1	1

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

R|X|_{R.B=S.B}S

R.A	R.B	S.A	S.B
a1	1	a1	1
a1	1	a3	1
a3	3	a2	3

```
SELECT * FROM R JOIN S  
ON R.B = S.B;
```

vagy

```
SELECT * FROM R INNER JOIN S  
ON R.B = S.B;
```

vagy

```
SELECT * FROM R, S  
WHERE R.B = S.B;
```

$\sigma_{R.B=S.B} (R \times S)$

Relációs algebra és az SQL

- Relációs algebra: érdekes a „hogyan” kérdés is
 - bár csak magas szinten
- Kapcsolat a relációs algebra műveletek és SQL záradékok, kulcsszavak... között
- SQL-nél mondtuk: „hogyan” helyett „mit”
- Fontos: DBMS kitalálja a leggyorsabb végrehajtási módot, háttérben optimalizál
- Van a relációs kalkulus, amely deklaratív nyelv (*declarative language*)