

# Adatbázisok 1.

## SQL bevezetés – 2. rész

Select-From-Where záradékok

Több relációt tartalmazó lekérdezések

Alkérdezések

# Többrelációs lekérdezések

- Általában több tábla kell

# Többrelációs lekérdezések

- Általában több tábla kell
- Ekkor a relációkat a FROM záradékban kell felsorolnunk.

# Többrelációs lekérdezések

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

# Példa: két reláció összekapcsolása

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

```
SELECT sör  
FROM Szeret, Látogat  
WHERE kocsmasma = 'Joe bárja' AND  
    Látogat.alkesz = Szeret.alkesz;
```

# Formális szemantika

- Majdnem ugyanaz, mint korábban:

# Formális szemantika

- Majdnem ugyanaz, mint korábban:
  1. Vegyük a FROM záradékban szereplő relációk Descartes-szorzatát.

# Formális szemantika

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



# Formális szemantika

- 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.

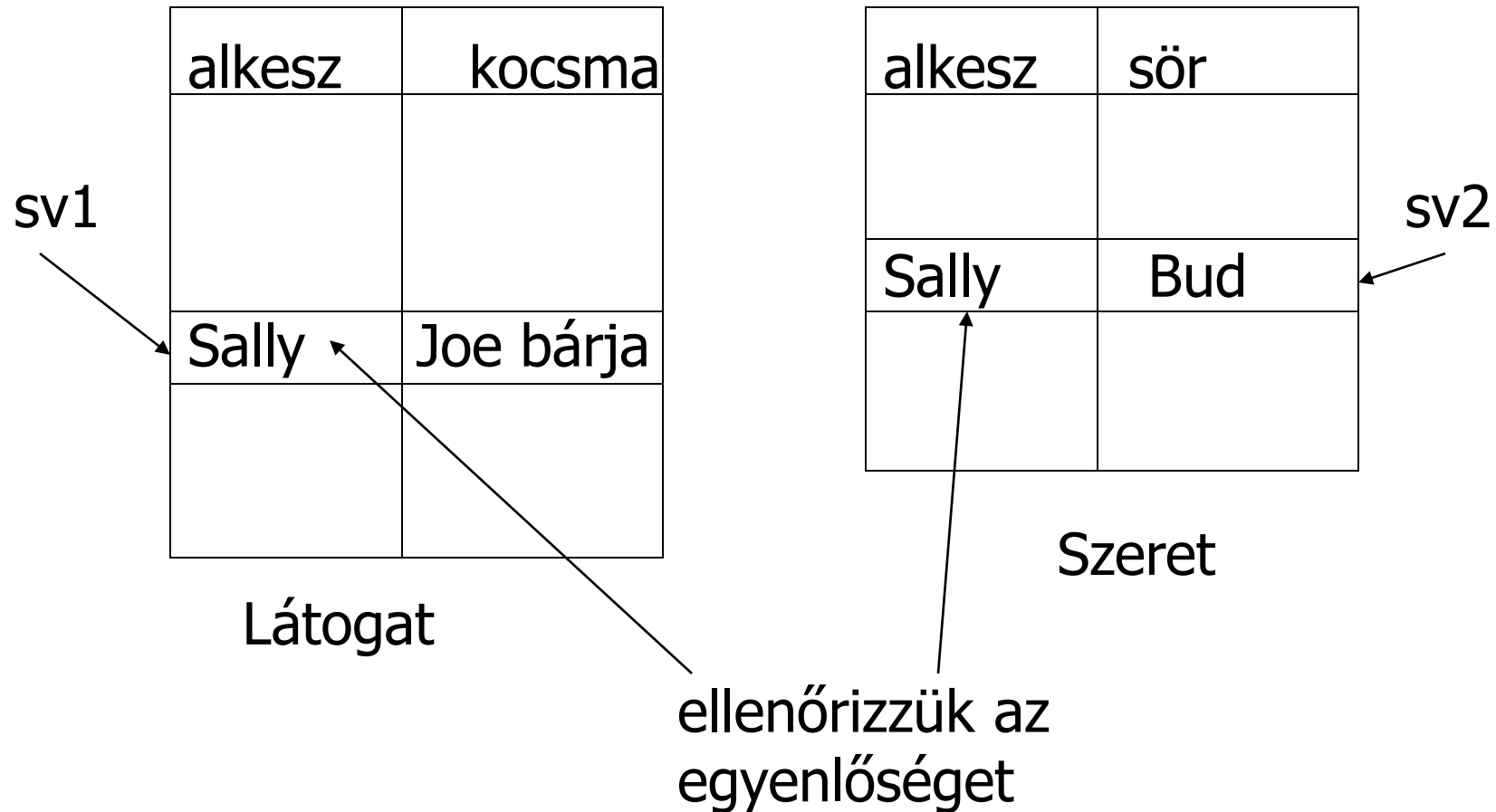
# Működési (operációs) szemantika

- Képzeld 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 sör
FROM Szeret, Látogat
WHERE kocsmas = 'Joe bárja' AND
  Látogat.alkeasz = Szeret.alkeasz;
```

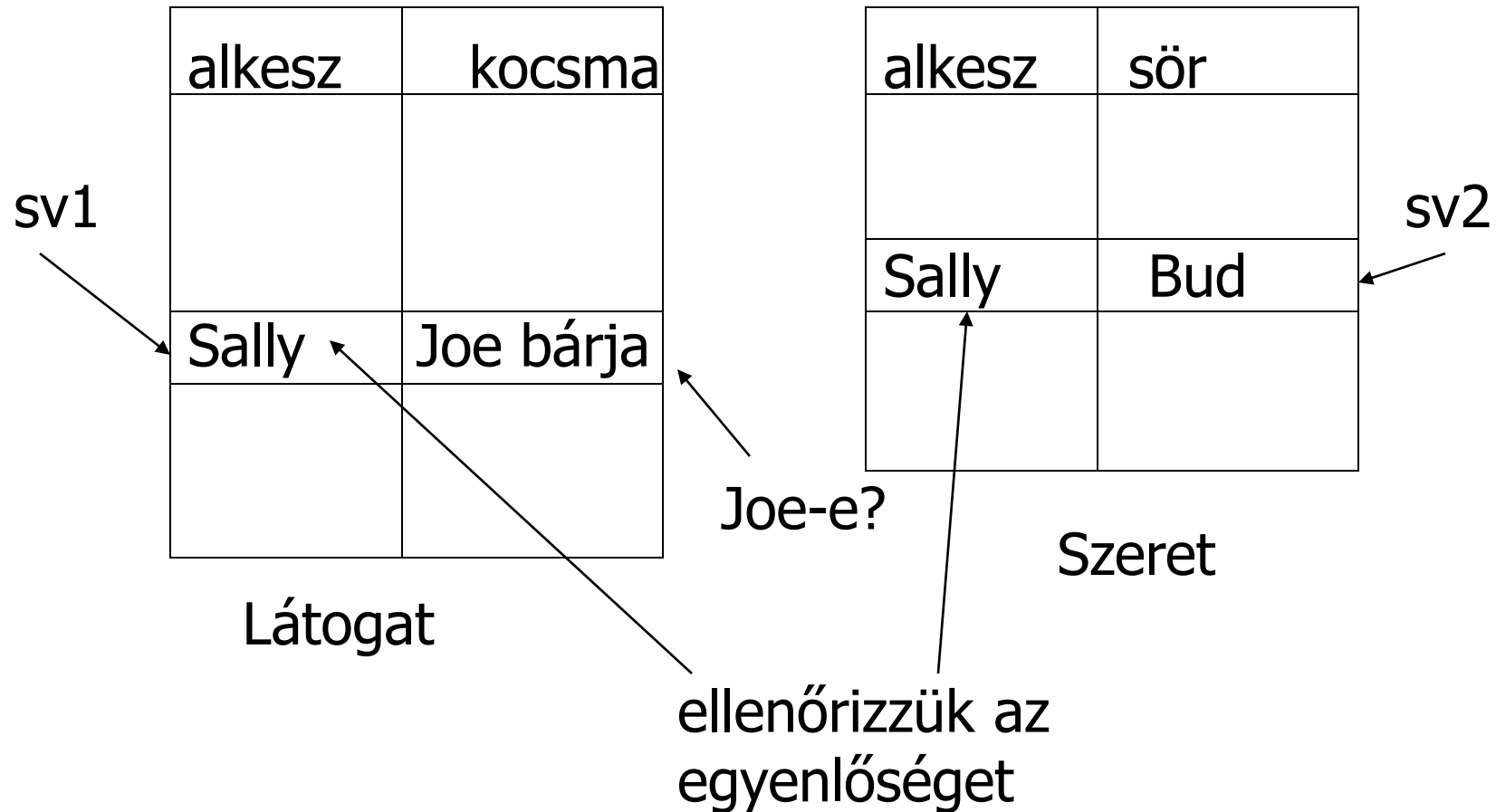
Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkeasz, sör)  
Felszolgál(kocsmas, sör, ár)  
Látogat(alkeasz, kocsmas)



# Példa

```
SELECT sör
FROM Szeret, Látogat
WHERE kocsmas = 'Joe bárja' AND
    Látogat.alkeasz = Szeret.alkeasz;
```

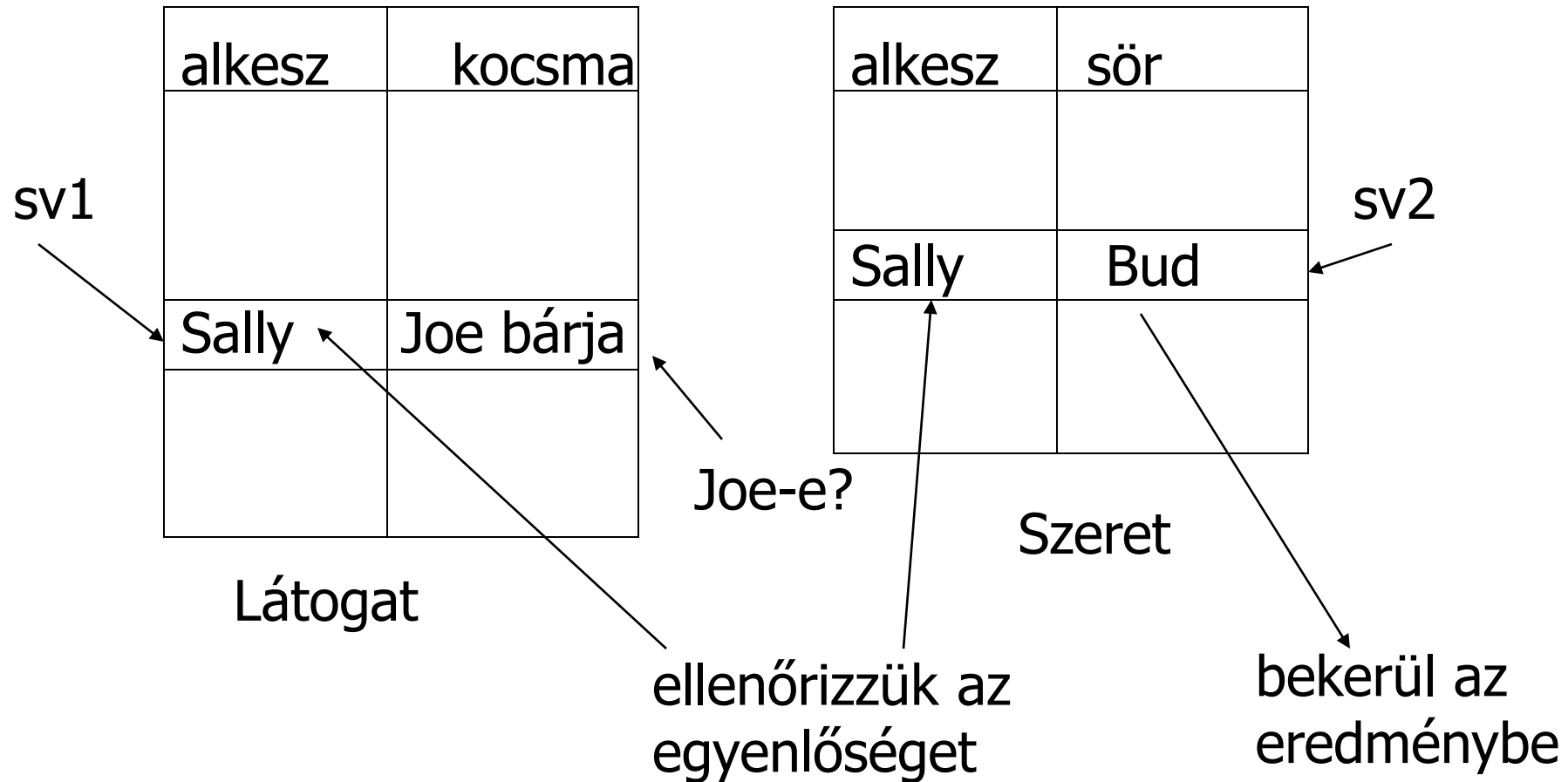
Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkeasz, sör)  
Felszolgál(kocsmas, sör, ár)  
Látogat(alkeasz, kocsmas)



# Példa

```
SELECT sör
FROM Szeret, Látogat
WHERE kocsmá = 'Joe bárja' AND
      Látogat.alkeasz = Szeret.alkeasz;
```

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkeasz, sör)  
Felszolgál(kocsmá, sör, ár)  
Látogat(alkeasz, kocsmá)



# 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

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

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Alkérdeések

- A FROM és WHERE záradékban zárójelezett SELECT-FROM-WHERE utasításokat (*alkérdés*) is használhatunk.



# Alkérdeések

- 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.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: alkérdés FROM-ban

- Keressük meg a Joe bárja vendégei által kedvelt söröket.

Alkeszek, akik látogatják  
Joe bárját.

```
SELECT sör
FROM Szeret, (SELECT alkesz
FROM Látogat
WHERE kocsm = 'Joe bárja') JD
WHERE Szeret.alkesz = JD.alkesz;
```

# 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.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: egysoros alkérdés

- A Felszolgál(kocsma, sör, ár) táblában keressük meg azokat a kocsmákat, ahol a Miller ugyanannyiba kerül, mint Joe bárjában a Bud.
- Két lekérdezésre biztos szükségünk lesz:
  1. Mennyit kér Joe a Budért?
  2. Melyik kocsmákban adják ugyanennyiért a Millert?

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Kérdés + alkérdés

```
SELECT kocsm  
FROM Felszolgál  
WHERE sör = 'Miller' AND  
      ár = (SELECT ár  
            FROM Felszolgál  
            WHERE kocsm = 'Joe bárja'  
            AND sör = 'Bud');
```

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Kérdés + alkérdés

```
SELECT kocsm  
FROM Felszolgál  
WHERE sör = 'Miller' AND
```

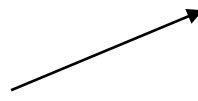
```
ár = (SELECT ár
```

```
FROM Felszolgál
```

```
WHERE kocsm = 'Joe bárja'
```

```
AND sör = 'Bud');
```

Ennyit kér  
Joe a Budért.



# 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 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.




Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: IN

```
SELECT *  
FROM Sörök  
WHERE név IN (SELECT sör  
               FROM Szeret  
               WHERE alkesz = 'Fred');
```

A sörök,  
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

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

Egy ciklus R sorai  
fölött.



# IN az R soraira vonatkozó predikátum

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

Egy ciklus R sorai  
fölött.

a	b
1	2
3	4

R

b	c
2	5
2	6

S

# IN az R soraira vonatkozó predikátum

```
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

# IN az R soraira vonatkozó predikátum

```
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



# 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



# 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.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Az EXISTS művelet

- EXISTS(<alkérdés>) akkor és csak akkor igaz, ha az alkérdés eredménye nem üres.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Az EXISTS művelet

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

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: EXISTS

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

Azon b1  
sörtől  
különböző  
sörök, →  
melyeknek  
ugyanaz  
a gyártója.

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

Korrelált alkérdés

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: EXISTS

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

Azon b1  
sörtől  
különböző  
sörök,  
melyeknek  
ugyanaz  
a gyártója.

```
SELECT *  
FROM Sörök  
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.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: EXISTS

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

Azon b1  
sörtől  
különböző  
sörök,  
melyeknek  
ugyanaz  
a gyártója.

```
SELECT *  
FROM Sörök  
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.

A „nem  
egyenlő”  
művelet  
SQL-ben.

# 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.



Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: ALL

```
SELECT sör  
FROM Felszolgál
```

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

A külső lekérdezés  
Felszolgáljának söre  
egyetlen alkérdésbeli  
sörnél sem lehet  
olcsóbb.

# Unió, metszet, különbség

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

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: metszet

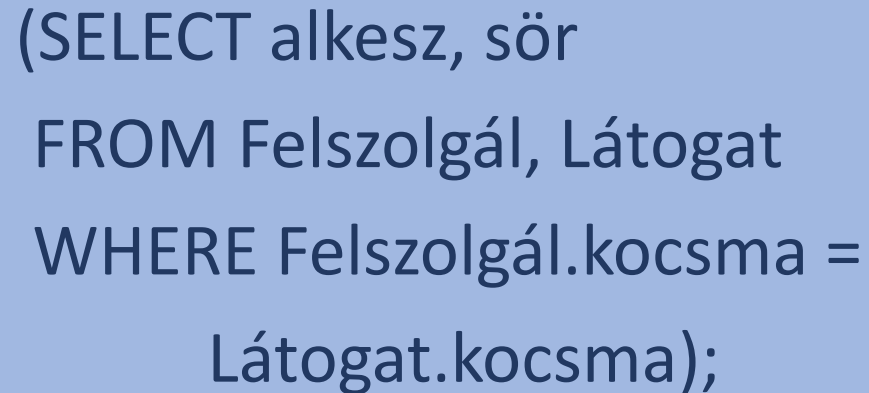
- A Szeret(alkesz, sör), Felszolgál(kocsm, sör, ár) és Látogat(alkesz, kocsm) táblák segítségével keressük meg azon alkeszeket és söröket:
  1. ahol az alkesz szereti az adott sört,
  2. az alkesz legalább egy olyan kocsmát látogat, ahol felszolgálják a szóban forgó sört.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Megoldás

(SELECT \* FROM Szeret)  
INTERSECT

Az alkesz látogatja  
azt a kocsmát, ahol  
felszolgálják azt a  
sört.



```
(SELECT alkesz, sör  
FROM Felszolgál, Látogat  
WHERE Felszolgál.kocsm =  
Látogat.kocsm);
```

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Megoldás

Az alkérdés  
egy tárolt  
táblát ad  
vissza.

(SELECT \* FROM Szeret)

INTERSECT

Az alkesz látogatja  
azt a kocsmát, ahol  
felszolgálják azt a  
sört.

(SELECT alkesz, sör  
FROM Felszolgál, Látogat  
WHERE Felszolgál.kocsm =  
Látogat.kocsm);

# Multihalmaz szemantika

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

# Motiváció: hatékonyság

- Ha projektálunk, akkor egyszerűbb, ha nem töröljük az ismétlődéseket.
  - Csak szépen végigmegyünk a sorokon.

# Motiváció: hatékonyság

- Ha projektálunk, akkor egyszerűbb, ha nem töröljük az ismétlődéseket.
  - 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

- Mindenképpen törölődjenek az ismétlődések: `SELECT DISTINCT . . .`

# Ismétlődések kiküszöbölése

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

# Ismétlődések kiküszöbölése

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

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: DISTINCT

```
SELECT DISTINCT ár  
FROM Felszolgál;
```

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

## Példa: ALL

- A Látogat(alkesz, kocsma) and Szeret(alkesz, sör) táblák felhasználásával:

```
(SELECT alkesz FROM Látogat)
EXCEPT ALL
(SELECT alkesz FROM Szeret);
```

- Kilistázza azokat az alkeszeket, akik több kocsmát látogatnak, mint amennyi sört szeretnek, és a leszámlálás különbsége azt mutatná, hogy mennyivel több kocsmát látogatnak mint amennyi sört kedvelnek.

# Összekapcsolás (join) kifejezések

- Az SQL-ben számos változata megtalálható az összekapcsolásoknak.
- Ezek a kifejezések önmagukban is állhatnak lekérdezésként, vagy a FROM záradékban is megjelenhetnek.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Descartes szorzat és természetes összekapcsolás

- Természetes összekapcsolás:

`R NATURAL JOIN S;`

- Szorzat:

`R CROSS JOIN S;`

- Példa:

`Szeret NATURAL JOIN Felszolgál;`

- A relációk helyén zárójelezett alkérdések is szerepelhetnek.

Sörök(név, gyártó)  
Kocsmák(név, cím, engedélySzám)  
Alkeszek(név, cím, telefon)  
Szeret(alkesz, sör)  
Felszolgál(kocsma, sör, ár)  
Látogat(alkesz, kocsma)

# Théta-összekapcsolás

- R JOIN S ON <feltétel>
- **Példa:** az Alkesz(név, cím) és Látogat(alkesz, kocsm) táblákból:

```
Alkesz JOIN Látogat ON  
    név = alkesz;
```

azokat  $(d, a, d, b)$  négyeseket adja vissza, ahol a  $d$  alkesz  $a$  címen lakik és a  $b$  kocsmát látogatja.



# 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;
```

**S**

A	B
a1	1
a2	3
a3	1

# 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';
```

**S**

A	B
a1	1
a2	3
a3	1

# 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';
```

**S**

A	B
a1	1
a2	3
a3	1

**RUS**

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

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

# 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 ∩ S**

A	B
a1	1

```
(SELECT * FROM R)
INTERSECT
(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

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

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 ∩ 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

**R|X|S**

A	B
a1	1

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

**S**

A	B
a1	1
a2	3
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|<sub>R.B=S.B</sub>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