

SQL bevezetés

Select-From-Where záradékok

Több relációt tartalmazó
lekérdezések

Alkérdezések

Miért az SQL?

- ◆ Az SQL magas szintű programozási nyelv.
 - ◆ A "hogyan" helyett azt mondjuk meg, hogy "mit" szeretnénk.
 - ◆ Így elkerülünk egy csomó macerát a procedurális nyelvekhez képest, mint pl C++ vagy Java.
- ◆ Az abkezelő rendszer kitalálja a leggyorsabb végrehajtási módot.
 - ◆ Ezt nevezik "lekérdezés optimalizációnak."

Select-From-Where záradékok

SELECT az érdekes attribútumok

FROM egy vagy több tábla

WHERE a táblák soraira vonatkozó
feltételek

A példa, amit használunk

- ◆ Minden SQL lekérdezést a következő adatbázisséma fölött hajtunk végre.

- ◆ Az aláhúzás a kulcsattribútumokat jelöli.

Sörök(név, gyártó)

Kocsmák(név, cím, engedélySzám)

Alkeszek(név, cím, telefon)

Szeret(alkesz, sör)

Felszolgál(kocsma, sör, ár)

Látogat(alkesz, kocsma)

Példa

- ◆ A Sörök(név, gyártó) táblában mely söröket gyártotta az Anheuser-Busch?

```
SELECT név
```

```
FROM Sörök
```

```
WHERE gyártó = 'Anheuser-Busch';
```

A lekérdezés eredménye

név
Bud
Bud Lite
Michelob
. . .

Az eredmény egyetlen attribútumot (név) tartalmaz a sorok Anheuser-Busch által gyártott söröket adják.

A lekérdezés jelentése

- ◆ Kezdjük a FROM záradékban megadott relációval.
- ◆ Alkalmazzuk a WHERE záradékban megadott kiválasztási feltételt.
- ◆ Vetítsük le az eredményt a SELECT záradékban megadott oszlopokra.

Szemantika (a példában)

név	gyártó
Bud	Anheuser-Busch

A t sorváltóval
a sorokat vesszük
egymás után.

t.név bekerül az
eredménybe, ha igen.
Ellenőrizzük, hogy
Anheuser-Busch-e.

* a SELECT záradékban

- ◆ Ha egy reláció szerepel a FROM záradékban, a * SELECT záradékban a reláció összes attribútumát helyettesíti.

- ◆ Példa: Sörök(név, gyártó):

```
SELECT *
```

```
FROM Sörök
```

```
WHERE gyártó = 'Anheuser-Busch';
```

A válasz:

név	gyártó
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
.

Azaz a **Sörök** reláció összes attribútuma szerepel.

Attribútumok átnevezése

◆ Az attribútumok átnevezéséhez "AS <new name>" utasítást használhatjuk.

◆ Példa: Sörök(név, gyártó):

```
SELECT név AS sör, gyártó
```

```
FROM Sörök
```

```
WHERE gyártó = 'Anheuser-Busch';
```

Az eredmény:

sör	gyártó
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
.

A SELECT záradék kifejezései

◆ Minden kifejezés, ami „értelmesnek tűnik” megjelenhet a SELECT záradékban.

◆ **Példa:** Felszolgál(kocsmá, sör, ár):

```
SELECT kocsmá, sör,  
        ár*114 AS árJenben  
FROM Felszolgál;
```

Az eredmény

kocsma	sör	árJenben
Joe's	Bud	285
Sue's	Miller	342
...

Konstansok

◆ Szeret(alkesz, sör):

```
SELECT alkesz,  
       'szereti a Budot' AS  
       BudIvó  
FROM Szeret  
WHERE sör = 'Bud';
```

Result of Query

alkesz	BudIvó
Sally	szereti a Budot
Fred	szereti a Budot
...	...

Információ integráció

- ◆ Sokszor az adatbázisokat sok forrásból építik fel (adattárházak).
- ◆ Tegyük fel, hogy minden kocsmának van egy saját **Menü(sör, ár)** táblája.
- ◆ A **Felhasználó(kocsmasor, sör, ár)** tábla elkészítéséhez minden ilyen táblát fel kell dolgoznunk és a kocsmasor nevét konstansként kell beszúrnunk.

Információ integráció --- (2)

- ◆ Például Joe bárja esetében ezzel a lekérdezéssel dolgozhatunk:

```
SELECT 'Joe bárja', sör, ár  
FROM Menü;
```

Összetett feltételek a WHERE záradékban

- ◆ Logikai műveletek: AND, OR, NOT.
- ◆ Összehasonlítások =, <>, <, >, <=, >=.

Példa összetett feltételre

- ◆ A Felszolgál(kocsma, sör, ár) táblában keressük meg Joe bárjában mennyit kérnek a Bud sörért:

```
SELECT ár
```

```
FROM Felszolgál
```

```
WHERE kocsmá = 'Joe bárja' AND  
       sör = 'Bud';
```

Minták

- ◆ A feltételekben a szavakat mintákra illeszthetjük
 - ◆ <Attribútum> LIKE <minta> vagy
<Attribútum> NOT LIKE <minta>
- ◆ *Minta* aposztrófok közötti szöveg az alábbi jelekkel: % = "akármennyi karakter"; _ = "tetszőleges karakter, pontosan egy."

Példa: LIKE

- ◆ Az Alkeszek(név, cím, telefon) keressük a budapestieket.

```
SELECT név  
FROM Alkeszek  
WHERE cím LIKE '%Budapest%';
```

NULL értékek

- ◆ A sorok mezői az SQL relációkban NULL értékeket is tartalmazhatnak.
- ◆ A jelentés a kontextustól függően változhat. Általában:
 - ◆ *hiányzó érték* : pl. nem ismerjük Joe bárja címét.
 - ◆ *értelmetlen* : egy szingli esetében a házastárs neve.

NULL összehasonlítás

- ◆ Az SQL valójában 3-értékű logikát használ: TRUE, FALSE, UNKNOWN.
- ◆ Ha egy értéket (NULL értéket is beleértve) NULL-lal hasonlítunk, az eredmény UNKNOWN.
- ◆ Egy sor akkor és csak akkor kerül be az eredménybe, ha a WHERE záradék TRUE értéket ad.

3-értékű logika

- ◆ Tegyük fel a következőt: TRUE = 1, FALSE = 0, and UNKNOWN = $\frac{1}{2}$.
- ◆ Ekkor: AND = MIN; OR = MAX, NOT(x) = $1-x$.

◆ Példa:

$$\begin{aligned} &\text{TRUE AND (FALSE OR NOT(UNKNOWN))} \\ &= \text{MIN}(1, \text{MAX}(0, (1 - \frac{1}{2}))) = \\ &\text{MIN}(1, \text{MAX}(0, \frac{1}{2})) = \text{MIN}(1, \frac{1}{2}) = \frac{1}{2}. \end{aligned}$$

Meglepetés!

◆ Az alábbi Felszolgál tábla esetén:

kocsmá	sör	ár
Joe bárja	Bud	NULL

SELECT kocsmá

FROM Felszolgál

WHERE ár < 2.00 OR ár >= 2.00;

← UNKNOWN →

← UNKNOWN →

← UNKNOWN →

Többrelációs lekérdezések

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

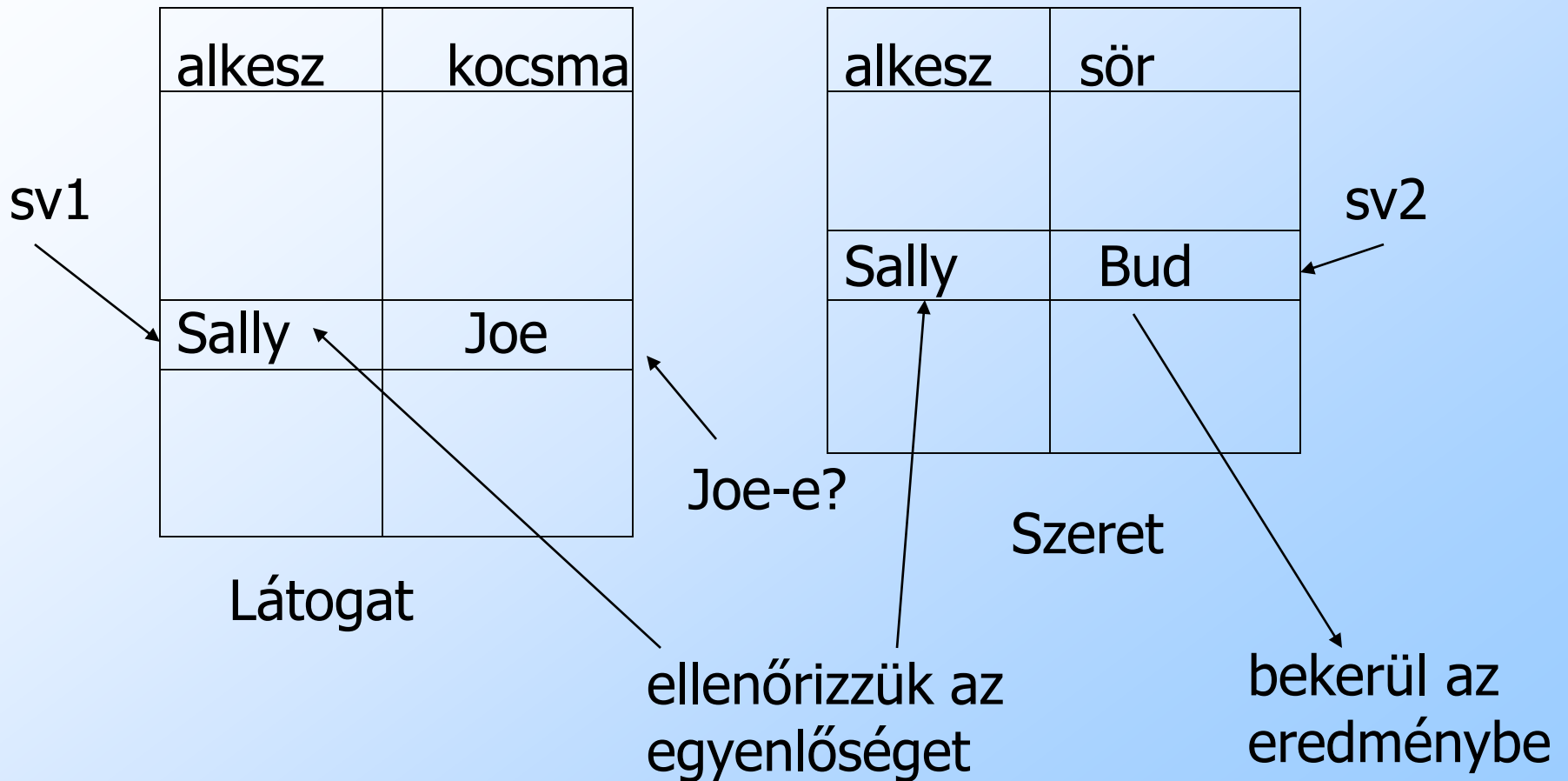
Formális szemantika

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

Működési (operációs) szemantika

- ◆ Képzeljük 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



Explicit sorváltozók

- ◆ Esetenként egy tábla több példányára is szükségünk van.
- ◆ A FROM záradékban a relációk neve után adjuk meg a hozzájuk tartozó sorváltozók 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;
```

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.

Példa: alkérdés FROM-ban

- ◆ Keressük meg a Joe bárja vendégei által kedvelt söröket.

```
SELECT sör
```

Alkeszek, akik látogatják
Joe bárját.



```
FROM Szeret, (SELECT alkesz
```

```
FROM Látogat
```

```
WHERE kocsmasma = '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.

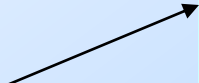
Példa: egysoros alkérdés

- ◆ 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?

Kérdés + alkérdés

```
SELECT kocsmas  
FROM Felszolgal  
WHERE sor = 'Miller' AND  
ar = (SELECT ar
```

Ennyit ker
Joe a Budert.



```
FROM Felszolgal  
WHERE kocsmas = 'Joe barja'  
AND sor = 'Bud');
```

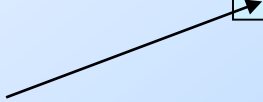
Az IN művelet

- ◆ `< 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

```
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);
```

Két 2 érték.

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.

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.

Példa: EXISTS

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

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

Azon b1
sörtől
különböző
sörök,
melyeknek
ugyanaz
a gyártója.

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

A „nem
egyenlő”
művelet
SQL-ben.

Az ANY művelet

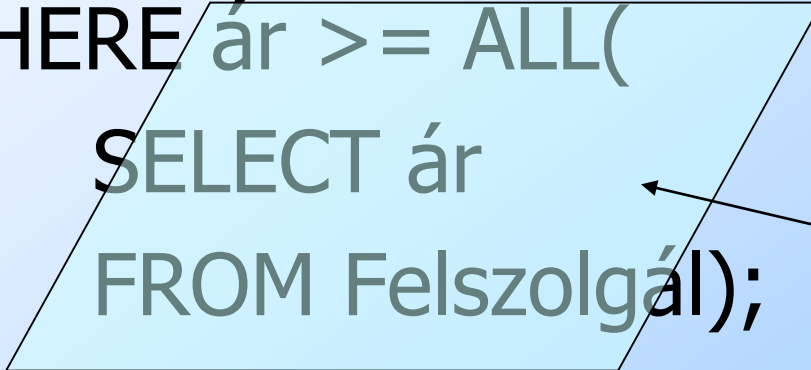
- ◆ $x = \text{ANY}(<\text{alkérdés}>)$ akkor és csak akkor igaz, ha x egyenlő az alkérdés legalább egy sorával.
 - ◆ = helyett bármilyen aritmetikai összehasonlítás szerepelhet.
- ◆ **Példa:** $x > \text{ANY}(<\text{alkérdés}>)$ akkor igaz, ha x az alkérdés legkisebb eleménél nagyobb.
 - ◆ Itt az alkérdés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

Az ALL művelet

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

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:

- ◆ ($\langle \text{alkérdés} \rangle$) UNION ($\langle \text{alkérdés} \rangle$)
- ◆ ($\langle \text{alkérdés} \rangle$) INTERSECT ($\langle \text{alkérdés} \rangle$)
- ◆ ($\langle \text{alkérdés} \rangle$) MINUS ($\langle \text{alkérdés} \rangle$)

◆ MINUS helyett EXCEPT is szerepelhet.

Példa: metszet

- ◆ A Szeret(alkesz, sör), Felszolgál(kocsmá, sör, ár) és Látogat(alkesz, kocsmá) 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.

Az alkérdés
egy tárolt
táblát ad
vissza.

Megoldás

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

(SELECT * FROM Szeret)

INTERSECT

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

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

Példa: DISTINCT

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

Példa: ALL

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

```
(SELECT alkész FROM Látogat)
```

```
EXCEPT ALL
```

```
(SELECT alkész FROM Szeret);
```

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

Join kifejezések

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

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.

Théta-összekapcsolás

◆ R JOIN S ON <feltétel>

◆ Példa: az Alkesz(név, cím) és
Látogat(alkesz, kocsmá) 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.