

# Lekérdezések az SQL-ben 1.rész

Tankönyv: Ullman-Widom:  
Adatbázisrendszerek Alapvetés  
Második, átdolgozott kiadás,  
Panem, 2009

---

## 6.1. Egyszerű (egy-relációs) lekérdezések az SQL-ben

- Select-From-Where utasítás
- Műveletek nullértékekkel
- Ismeretlen igazságérték
- Az eredmény rendezése



# Select-From-Where (SFW) utasítás

## Egyetlen reláció lekérdezése

- Melyek az unér műveletek? A relációs algebra mely műveletei egy argumentumosak?
  - Ezek a vetítés  $\Pi_{Lista}$
  - a kiválasztás  $\sigma_{Felt}$
  - és az átnevezés  $\rho_{S(B_1, \dots, B_k)}(R)$
- Egyetlen reláció lekérdezése:  $\Pi_{Lista} ( \sigma_{Felt} ( \rho_{S(B_1, \dots, B_k)}(R) ) )$
- **SELECT utasítás** több részből áll, a részeket **záradékok**nak nevezzük. Három alapvető záradék van:
- **SELECT lista** -- milyen típusú sort szeretnénk az eredményben látni?  
**FROM Relációnév** -- a tábla neve  
**WHERE feltétel** -- milyen feltételeknek eleget tevő sorokat kiválasztani?

# Példa – Sörivók adatbázisséma

- Az előadások SQL lekérdezései az alábbi Sörivók adatbázissémán alapulnak (aláhúzás jelöli a kulcs attribútumokat)

**Sörök(név, gyártó)**

**Bárok(név, város, tulaj, engedély)**

**Ivó(név, város, tel)**

**Kedvel(név, sör)**

**Felszolgál(bár, sör, ár)**

**Látogat(név, bár)**

# Egyszerű példa Select-From-Where-re

- Használjuk **Sörök(név, gyártó)** relációsémát, mely söröket gyártja a Dreher?

```
SELECT név  
FROM Sörök  
WHERE gyártó = 'Dreher' ;
```

# A lekérdezés eredménye

| név            |
|----------------|
| Arany Ászok    |
| Dreher Classic |
| . . .          |

A lekérdezés eredménye egy reláció, amelynek egy attribútuma van (név) és a sorai az összes olyan sör neve, amelyet a Dreher gyárt.

**Eltérés a relációs algebrától: Az SQL alapértelmezésben nem szűri ki a duplikátumokat, az eredmény multihalmaz.**

# Egy-relációs lekérdezés formális kiértékelése

- Kiindulunk a **FROM záradékból**, mely táblára vonatkozik a lekérdezés?
- Elvégezzük a **WHERE záradékban** szereplő feltételnek eleget tevő sorok kiválasztását
- Alkalmazzuk a **SELECT záradékban** jelölt kiterjesztett projekciót. Lényeges különbség a relációs algebra és SQL között, hogy **az SQL-ben az eredmény** alapértelmezés szerint **nem halmaz**, hanem **multihalmaz** (egy sor az eredményben többször is előfordulhat, ennek az oka, hogy az olcsóbb és hatékonyabb kiértékelést tekintjük az SQL-ben alapértelmezésnek).
- Ahhoz, hogy halmazt kapjunk, azt külön kérni kell  
SELECT **DISTINCT** Lista

# A műveletek szemantikája

| név         | gyártó |
|-------------|--------|
|             |        |
| Arany Ászok | Dreher |
|             |        |

1) Ellenőrizzük a feltételt, hogy a gyártó Dreher-e

2) Ha a feltétel teljesült, akkor képezünk egy t eredménysort

3) Ebből a t sorból a SELECT listának megfelelő típusú sort képezzük, példa: t.név

# Egy táblát használó SFW alapértelmezése

```
SELECT [DISTINCT] kif1 [[AS] onév1], ... , kifn [onévn]  
FROM táblanév [sorváltozó]  
[WHERE feltétel]
```

**Alapértelmezés** (a műveletek szemantikája -- általában)

- A FROM záradékban levő relációhoz tekintünk egy **sorváltozót**, amely a reláció minden sorát bejárja
- Minden egyes „aktuális” sorhoz **kiértékeljük** a WHERE záradékot
- Ha helyes (vagyis **igaz**) választ kaptunk, akkor képezünk egy sort a SELECT záradékban szereplő kifejezéseknek megfelelően.



# SELECT záradékban \* jelentése

- Amikor egy reláció van a FROM záradékban, akkor a SELECT záradékban levő \* jelentése: „a reláció minden attribútuma”
- **Példa:** Keressük a Sörök(név, gyártó) tábla alapján a Dreher-sörök adatait.
- A lekérdezés eredménye

**SELECT \***

**FROM Sörök**

**WHERE gyártó = 'Dreher' ;**

- A lekérdezés eredménye a Sörök tábla összes attribútumát tartalmazza.

# Attribútumok átnevezése

- Ha az eredményben más attribútumnevet szeretnénk használni, akkor “[AS] új\_oszlopnév” segítségével tudunk más oszlopnevet kiírni (Oracle-ben nem kell ‘AS’)

- Példa: Sörök(név, gyártó)

```
SELECT név sör, gyártó
```

```
FROM Sörök
```

```
WHERE gyártó = 'Dreher' ;
```

- A lekérdezés eredményében az oszlopnevek sör és gyártó lesznek.

# SELECT záradékban levő kifejezések

- Tetszőleges kifejezés, amely értelmezhető előfordulhat a SELECT záradék elemeként  
(lásd bővebben majd a gyakorlatok példáit)

- Példa: Felszolgál(bár, sör, ár)

```
SELECT bár, sör, ár*114 árYenben  
FROM Felszolgál;
```

- Konstansok a kifejezésekben Kedvel(név, sör):

```
SELECT név AS DABkedvelő  
FROM Kedvel  
WHERE sör = 'DAB' ;
```

# WHERE záradék (összetett feltételek)

- Hasonlóan, mint a relációs algebra kiválasztás ( $\sigma$ ) feltételében **elemi feltételekből építkezünk**, ahol elemi feltételen két kifejezés  $=$ ,  $<>$ ,  $<$ ,  $>$ ,  $<=$ ,  $>=$  aritmetikai összehasonlítását, a theta műveletet értjük.
- **Logikai műveletek** AND, OR, NOT és zárójel ( ) segítségével kapjuk **az összetett feltételeket**.
- **Példa: Felszolgál (bár, sör, ár)** relációséma esetén keressük a „Joe's Bar” bárban árult „DAB” sörök árát:

```
SELECT ár
FROM Felszolgál
WHERE bár = 'Joe' 's Bar' AND
       sör = 'DAB' ;
```

# WHERE záradék (további lehetőségek)

SQL specialitások, amelyek könnyen átírhatóak relációs algebrai kifejezésre (összetett kiválasztási feltételre)

- **BETWEEN .. AND ..** intervallumba tartozás
- **IN (értékhalma)** egyszerű értékek halmaza

SQL specialitások, nem írhatók át relációs algebrába:

(--- ezek jönnek a köv.lapon...)

- Karakterláncok **LIKE** összehasonlítása mintákkal
- **IS NULL** összehasonlítás

# LIKE

- **Karakterláncok összehasonlítása mintákkal:**
  - <attribútum> LIKE <minta> vagy
  - <attribútum> NOT LIKE <minta>
- **Minta** egy olyan karakterlánc, amelyben használhatjuk a speciális % és \_ karaktereket. A mintában % megfelel bármilyen karakterláncnak és \_ bármilyen karakternek.
- **Példa:** Azokat a bárokat keressük, amelynek a nevében van 's (mint például *Joe's Bar*)

```
SELECT név
FROM Bárók
WHERE név LIKE ' % ' 's % ' ;
```

# NULL értékek

- Az SQL lehetővé teszi, hogy a relációk soraiban az attribútum értéke egy speciális NULL nullérték legyen.
- **A nullérték értelmezésére** több lehetőségünk is van:
  - **Ismeretlen érték:** például tudom, „Joe’s Bár”-jának van valamilyen címe, de nem tudom, hogy mi az.
  - **Nem-definiált érték:** például a házastárs attribútumnak egyedülálló embereknél nincs olyan értéke, aminek itt értelme lenne, nincs házastársa, ezért nullérték.
- **Where záradékban a nullérték** vizsgálata:
  - IS NULL
  - IS NOT NULL

# NULL értékek használata

- **Where záradékban a nullérték** használata:
  - Amikor egy aritmetikai műveletben az egyik tag **NULL**, akkor az eredmény is **NULL**.
  - Amikor egy **NULL** értéket hasonlítunk össze bármely más értékkel (beleértve a NULL-t is) az összehasonlítási operátorok (=, <>, <, <=, >, >=) segítségével, akkor az eredmény **UNKNOWN** (ismeretlen).



# Az ismeretlen (unknown) igazságérték

- Az SQL-ben szereplő logikai feltételek valójában **háromértékű logika**: TRUE, FALSE, UNKNOWN (magyarban igaz, hamis, ismeretlen rövidítése miatt inkább meghagyjuk az angol T, F, U rövidítéseket).
- A WHERE záradékban szereplő logikai feltételt a rendszer minden egyes sorra ellenőrzi és a logikai érték TRUE, FALSE vagy UNKNOWN valamelyike lehet, de az eredménybe csak azok a sorok kerülnek, amelyeknek a feltétel kiértékelése TRUE értéket adott.

# A 3-értékű logika

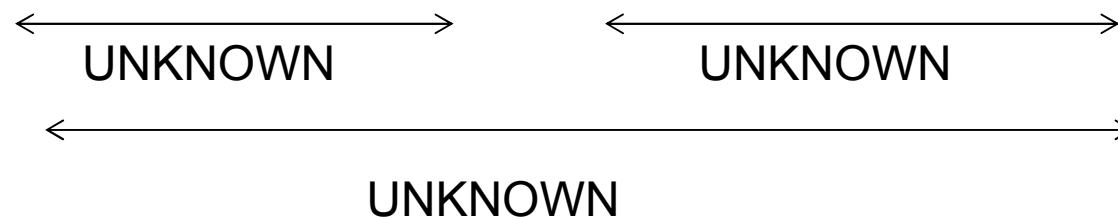
- **Hogyan működnek** az AND, OR, és NOT logikai műveletek a 3-értékű logikában?
- A szabályt könnyű megjegyezni, ha úgy tekintjük, hogy TRUE = 1, FALSE = 0, és UNKNOWN =  $\frac{1}{2}$ .
- Ekkor AND = MIN, OR = MAX, NOT(x) = 1-x.
- **Példa:**  
TRUE AND (FALSE OR NOT(UNKNOWN)) =  
MIN(1, MAX(0, (1 -  $\frac{1}{2}$ ))) =  
MIN(1, MAX(0,  $\frac{1}{2}$ )) =  
MIN(1,  $\frac{1}{2}$ ) =  $\frac{1}{2}$  = UNKNOWN
- A 3-értékű logika AND, OR és NOT igazságtáblázatát lásd a **Tk. 6.2.ábráját** (vagy kitöltése a fenti szabállyal)

# Egy meglepő példa

- **Példa:** Felszolgál reláció legyen az alábbi:

| bár       | sör | ár   |
|-----------|-----|------|
| Joe's Bar | Bud | NULL |

```
SELECT bar
FROM Felszolgál
WHERE ár < 2.00 OR ár >= 2.00;
```



# Oka: a 2-értékű $\neq$ 3-értékű szabályok

- Bizonyos általános szabályok, mint például, hogy az AND kommutatív érvényes a 3-értékű logikában is.
- Ellenben nem igaz, például a **kizáró szabály**, vagyis  $p \text{ OR } \text{NOT } p = \text{TRUE}$  nem teljesül, ha  $p = \text{UNKNOWN}$ , mert ekkor a baloldal:  $\text{MAX}( \frac{1}{2}, (1 - \frac{1}{2}) ) = \frac{1}{2} \neq 1$  vagyis a 3-értékű logikában baloldal értéke nem TRUE.
- Ezért az előző példában nem az eredeti egy soros táblát, hanem az üres táblát (amelynek egy sora sincs) kaptuk meg az eredménytáblaként.

# Az eredmény rendezése

- SQL SELECT utasításban a záradékok
- Az SQL lehetővé teszi, hogy a lekérdezés eredménye bizonyos sorrendben legyen rendezve. Az első attribútum egyenlősége esetén a 2.attribútum szerint rendezve, stb, minden attribútumra lehet növekvő vagy csökkenő sorrend.
- Select-From-Where utasításhoz a következő záradékot adjuk, a WHERE záradék és minden más záradék (mint például GROUP BY és HAVING) után következik:

**SELECT ... FROM ... [WHERE ...] [...]**

**ORDER BY {attribútum [DESC], ...}**

- **Példa: SELECT \* FROM Felszolgál  
ORDER BY ár DESC, sör**