

Lekérdezések az SQL-ben 2.rész

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



- ## 6.2. Több relációra vonatkozó lekérdezések az SQL-ben
- Szorzat és összekapcsolás
 - Sorváltozók használata
 - Lekérdezések alapértelmezése
 - Halmazműveletek az SQL-ben

Select-From-Where (SFW) utasítás

Több reláció lekérdezése

- Gyakran előforduló relációs algebrai kifejezés

$\Pi_{\text{Lista}} (\sigma_{\text{Felt}} (R_1 \times \dots \times R_n))$ típusú kifejezések

- Szorzat és összekapcsolás az SQL-ben

- SELECT s-lista -- milyen típusú sort szeretnénk az eredményben látni?

FROM f-lista -- relációk (táblák) összekapcsolása, illetve szorzata

WHERE felt -- milyen feltételeknek eleget tevő sorokat kell kiválasztani?

- FROM f-lista elemei (ezek ismétlődhetnek)

táblanév [[AS] sorváltozó, ...]

Itt: táblák direkt szorzata SQL-ben is bevezethetünk további lehetőségeket a különböző összekapcsolásokra, ezt később a köv.héten tárgyaljuk. Ma: a lekérdezések alapértelmezése

Attribútumok megkülönböztetése ---1

- **Milyen problémák merülnek fel?**
- (1) Ha egy attribútumnév több sémában is előfordul, akkor nem elég az attribútumnév használata, mert ekkor nem tudjuk, hogy melyik sémához tartozik.
- Ezt a problémát az SQL úgy oldja meg, hogy megengedi egy relációnévnek és egy pontnak a használatát egy attribútum előtt: **R.A** (az R reláció A attribútumát jelenti).
- **Természetes összekapcsolás** legyen R(A, B), S(B,C)

```
SELECT A, R.B B, C
FROM R, S
WHERE R.B=S.B;
```

Attribútumok megkülönböztetése ---2

- Milyen problémák merülnek még fel?
- (2) Semmi nem tiltja, hogy ugyanaz a reláció többször is szerepeljen, szükség lehet arra, hogy ugyanaz a relációnév többször is előforduljon a FROM listában.
- Ekkor a FROM listában másodnevet kell megadni, erre **sorváltzóként** is szoktak hivatkozni, megadjuk azt is, hogy melyik sorváltzó melyik relációt képviseli:

FROM $R_1 [t_1], \dots, R_n [t_n]$

Ekkor a SELECT és WHERE záradékok kifejezésekben a hivatkozás: **$t_i.A$** (vagyis sorváltzó.attribútumnév)

SFW szabvány alapértelmezése ---1

- Kiindulunk a **FROM záradékból**: a FROM lista minden eleméhez **egy beágyazott ciklus**, végigfut az adott tábla sorain a ciklus minden lépésénél az n darab sorváltozónak lesz egy-egy értéke
- ehhez kiértékeljük a WHERE feltételt, vagyis elvégezzük a **WHERE záradékban** szereplő feltételnek eleget tevő sorok kiválasztását (csak a helyesek, ahol TRUE=igaz választ kapunk), azok a sorok kerülnek az eredménybe.
- Alkalmazzuk a **SELECT záradékban** jelölt kiterjesztett projekciót. Az **SQL-ben az eredmény alapértelmezés szerint** itt sem halmaz, hanem **multihalmaz**.

Ahhoz, hogy halmazt kapjunk, azt külön kérni kell:
SELECT DISTINCT Lista

SFW szabvány alapértelmezése ---2

FOR t_1 sorra az R_1 relációban DO

FOR t_2 sorra az R_2 relációban DO

...

FOR t_n sorra az R_n relációban DO

IF a where záradék igaz, amikor az attribútumokban
 t_1, t_2, \dots, t_n megfelelő értékei találhatóak

THEN

t_1, t_2, \dots, t_n -nek megfelelően kiértékeljük a
select záradék attribútumait
és az értékekből alkotott sort
az eredményhez adjuk

SFW szabvány alapértelmezése ---3

```
SELECT [DISTINCT] kif1 [[AS] onév1], ..., kifn [[AS] onévn]  
FROM R1 [t1], ..., Rn [tn]  
WHERE feltétel (vagyis logikai kifejezés)
```

Alapértelmezés (a műveletek szemantikája -- általában)

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

Megjegyzés: konverzió relációs algebrába

SELECT [DISTINCT] kif₁ [[AS] onév₁], ..., kif_n [[AS] onév_n]

FROM R₁ [t₁], ..., R_n [t_n]

WHERE feltétel (vagyis logikai kifejezés)

- 1.) A FROM záradék sorváltozóiból indulunk ki, és tekintjük a hozzájuk tartozó relációk Descartes-szorzatát. Átnevezéssel valamint R.A jelöléssel elérjük, hogy minden minden attribútumnak egyedi neve legyen.
- 2.) A WHERE záradékot átalakítjuk egy kiválasztási feltétellé, melyet alkalmazunk az elkészített szorzatra.
- 3.) Végül a SELECT záradék alapján létrehozuk a kifejezések listáját, a (kiterjesztett) vetítési művelethez.

$$\Pi_{\text{onév}_1, \dots, \text{onév}_n} (\sigma_{\text{feltétel}} (R_1 \times \dots \times R_n))$$

Példa: Két tábla összekapcsolása ---1

- Mely söröket szeretik a Joe's Bárba járó sörivók?

```
SELECT sör
```

```
FROM Kedvel, Látogat
```

```
WHERE bár = 'Joe' 's Bar'
```

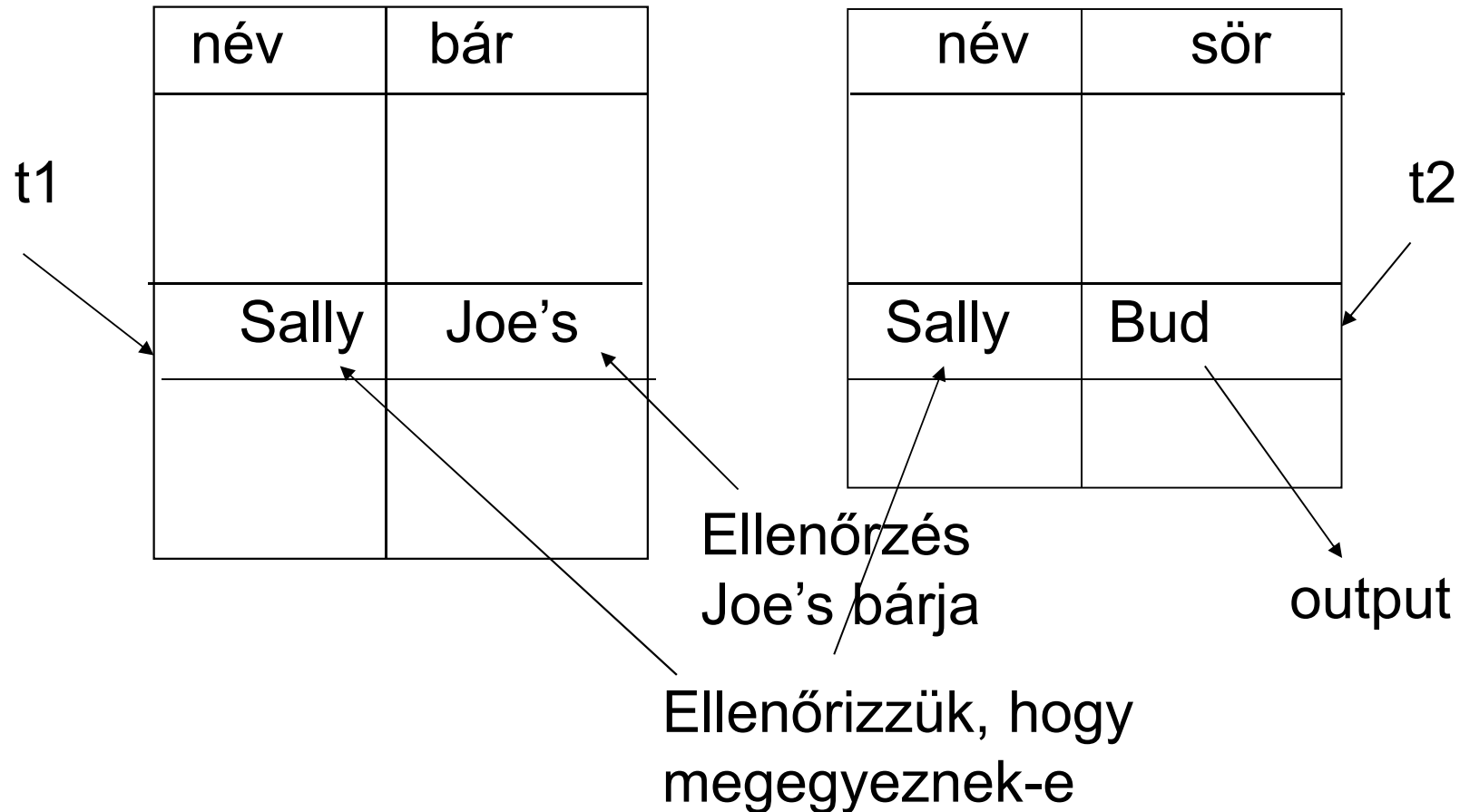
```
AND Látogat.név = Kedvel.név;
```

- Kiválasztási feltétel: **bár = 'Joe' 's Bar'**
- Összekapcsolási feltétel: **Látogat.név = Kedvel.név**
- Alapértelmezését lásd a következő oldalon
- Összekapcsolások SQL:1999-es szintaxisa a köv.órán.

Példa: Két tábla összekapcsolása ---2

Látogat

Kedvel



Tábla önmagával való szorzata ---1

- Bizonyos lekérdezéseknél arra van szükségünk, hogy ugyanannak a relációnak több példányát vegyük.
- Ahhoz, hogy meg tudjuk különböztetni a példányokat a relációkat átnevezzük, másodnevet adunk, vagyis **sorváltozókat** írunk mellé a FROM záradékban.
- A relációkat mindig átnevezhetjük ily módon, akkor is, ha egyébként nincs rá szükség (csak kényelmesebb).
- **Példa: R(Szülő, Gyerek)** séma feletti relációban adott szülő-gyerek adatpárokból állítsuk elő a megállapítható Nagyszülő-Unoka párokat!

```
SELECT t1.Szülő NagySzülő, t2.Gyerek Unoka
FROM R t1, R t2
WHERE t1.Gyerek = t2.Szülő;
```

Tábla önmagával való szorzata ---2

- **Példa: Sörök(név, gyártó)** tábla felhasználásával keressük meg az összes olyan sörpárt, amelyeknek ugyanaz a gyártója.
 - Ne állítsunk elő (Bud, Bud) sörpárokat.
 - A sörpárokat ábécé sorrendben képezzük, például ha (Bud, Miller) szerepel az eredményben, akkor (Miller, Bud) ne szerepeljen.

```
SELECT s1.név, s2.név  
FROM Sörök s1, Sörök s2  
WHERE s1.gyártó = s2.gyártó  
AND s1.név < s2.név;
```

Halmazműveletek

- Mi hiányzik még, hogy a relációs algebra alpműveleteit mindet az SQL-ben vissza tudjuk adni?
- A relációs algebrai halmazműveletek: **unió, különbség** mellett az **SQL-ben ide soroljuk a metszetet is** (ugyanis SQL-ben megvan a metszetet implementációja is).
- Az SQL-ben a halmazműveleteket úgy vezették be, hogy azt mindig két lekérdezés között lehet értelmezni, vagyis nem relációk között, mint $R \cup S$, hanem lekérdezem az egyiket is és a másikat is, majd a lekérdezések unióját veszem.

(lekérdezés1)

[**UNION** | **INTERSECT** | {**EXCEPT** | **MINUS**}]

(lekérdezés2);

Példa: Intersect (metszet)

- Kedvel(név, sör), Felszolgál(bár, sör, ár) és Látogat(név, bár) táblák felhasználásával keressük

Trükk: itt ez az alkérdés valójában az adatbázisban tárolt tábla azokat a sörivókat és söroket, amelyekre a sörivó szereti az adott sört **és** a sörivó látogat olyan bárt, ahol felszolgálják a sört.

(**SELECT * FROM Kedvel**)

INTERSECT

(**SELECT név, sör**

FROM Felszolgál, Látogat

WHERE Látogat.bár = Felszolgál.bár);

(név, sör) párok, ahol a sörivó látogat olyan bárt, ahol ezt a sört felszolgálják

Halmaz-multihalmaz szemantika

- A **SELECT-FROM-WHERE** állítások **multihalmaz** szemantikát használnak, a **halmazműveleteknél** mégis a **halmaz szemantika** az érvényes.
 - Azaz sorok nem ismétlődnek az eredményben.
- 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.
- **Motiváció:** hatékonyság, minimális költségek

Példa: ALL (multihalmaz szemantika)

- **Látogat(név, bár)** és **Kedvel(név, sör)** táblák felhasználásával kilistázzuk azokat a sörivókat, akik több bárt látogatnak, mint amennyi sört szeretnek, és annyival többet, mint ahányszor megjelennek majd az eredményben

(SELECT név FROM Látogat)

EXCEPT ALL

(SELECT név FROM Kedvel) ;