

6.előadás: Adatbázisok-I.

dr. Hajas Csilla (ELTE IK)
<http://sila.hajas.elte.hu/>

Többléptáblás lekérdezések az SQL-ben

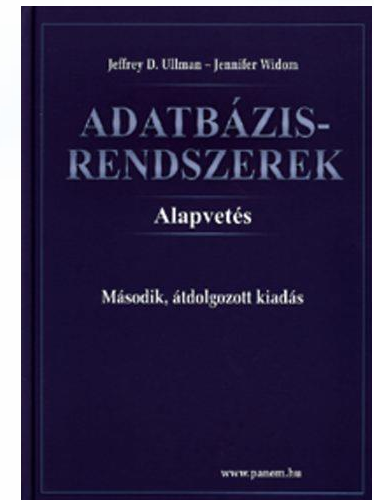
Tankönyv:

6.2. Több táblás lekérdezések SQL-ben

6.3. Alkérdezések FROM-ban, WHERE-ben

Összekapcsolások az SQL-ben

http://sila.hajas.elte.hu/AB1ea/REL4_adatb_hcs.pdf

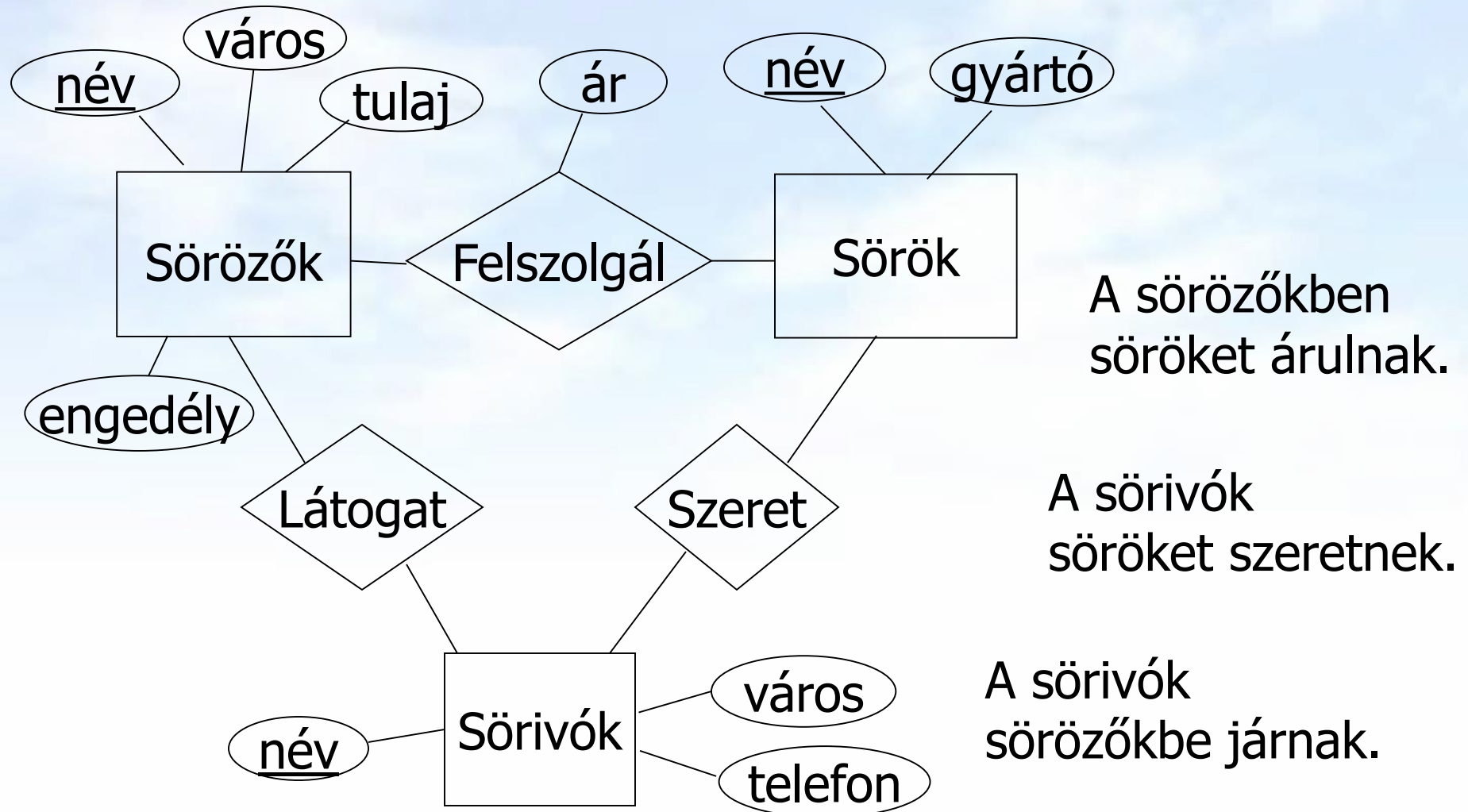


Előzmények [REL4.pdf]

- Az első 5 előadást normál tantermi előadásként tartottuk
 - Egy táblára vonatkozó ismeretek
- [01] [TERV1.pdf](#) (Relációs modell és az E/K modell bev)
[SQL1.pdf](#) (SQL bev, create table/1.tipusok, kulcsok)
- [02] [REL1.pdf](#) (Egytáblás lekérdezések, vetítés, szűrés)
- [03] [REL2.pdf](#) (Egytáblás lekérdezések, csoportosítás)
 - Több táblára vonatkozó ismeretek
- [04] [TERV2.pdf](#) (E/K haladó, megszorítások, alosztályok)
[SQL2.pdf](#) (create table/2., constraints, hivatk.épség)
- [05] [REL3.pdf](#) (Több táblás lekérd. relációs algebrában)
- [06] Mai előadáson: Több táblás lekérdezések SQL-ben

[Emlékeztető]

Az előadás példája (E/K diagram)



[Emlékeztető]

Az előadás példája (relációs sémák)

- Az egyedhalmazok átírása
(aláhúzás jelöli a kulcs attribútumokat)

Sörözők(név, város, tulaj, engedély)

Sörök(név, gyártó)

Sörivők(név, város, telefon)

- A sok-sok kapcsolatok átírása (és átnevezéssel:
sör hivatkozási épség megszorítás a Sörök.név-re,
söröző hivatk.épség pedig a Sörözők.név-re szól)

Felszolgál(söröző, sör, ár)

Látogat(név, söröző)

Szeret(név, sör)

6.2. Több táblára vonatkozó lekérdezések az SQL-ben

Select-From-Where (SFW) utasítás

- 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 from-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: a from lista elemei a táblák direkt szorzatát jelenti, az összekapcsolási feltételt where-ben adjuk meg, később bevezetünk majd további lehetőségeket a különböző összekapcsolásokra az SQL from záradékában.

Példa: Két tábla összekapcsolása ---1

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

```
SELECT sör
```

```
FROM Szeret, Látogat
```

```
WHERE söröző = 'Joe' 's Bar'
```

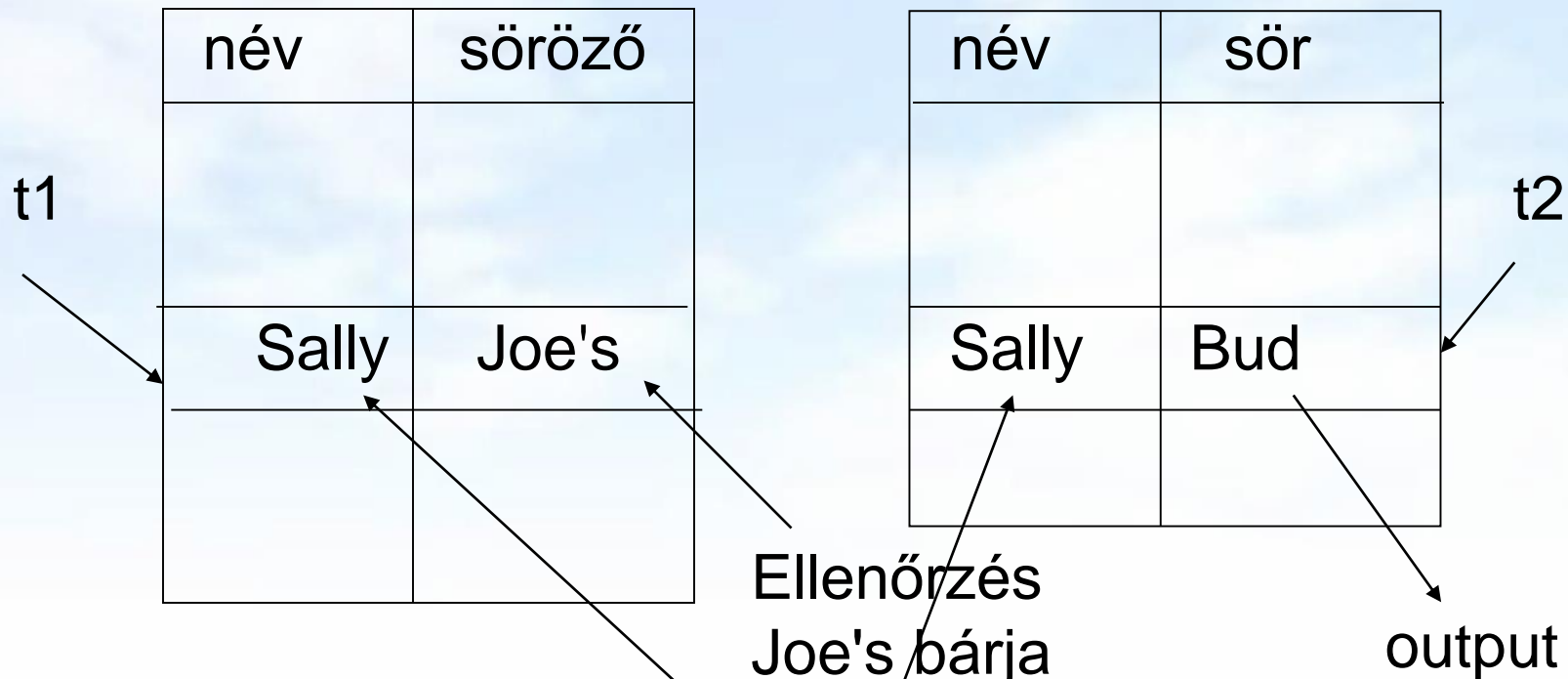
```
AND Látogat.név = Szeret.név;
```

- Kiválasztási feltétel: **söröző = 'Joe' 's Bar'**
- Összekapcsolási feltétel: **Látogat.név = Szeret.név**

Példa: Két tábla összekapcsolása ---2

Látogat

Szeret



Attribútumok megkülönböztetése

- **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 (hasonlóan, mint a rekord.mező jelölés), **R.A** (az R reláció aktuális sorának azt a komponensét jelenti, amely az A attribútum alatt szerepel)
- **Természetes összekapcsolás** legyen R(A, B), S(B,C)
SELECT A, R.B B, C
FROM R, S
WHERE R.B=S.B;

Tábla önmagával való szorzata

- Milyen problémák merülnek még fel?
- (2) Bizonyos lekérdezéseknél arra van szükségünk, hogy ugyanannak a relációnak több példányát vegyük.
- Ahhoz, hogy meg tudjuk különböztetni a példányokat a relációkat átnevezzük. Ekkor a FROM listában a táblához másodnevet kell megadni, erre **sorváltzóként** is szoktak hivatkozni, megadjuk, 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)

- A relációkat mindig átnevezhetjük ily módon, akkor is, ha egyébként nincs rá szükség (csak kényelmesebb).

Tábla önmagával való szorzata

- **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;
```

SFW szabvány alapértelmezése

```
SELECT [DISTINCT] kif1 [[AS] onév1], ..., kifn [[AS] onévn]  
FROM R1 [ [AS] t1 ], ..., Rn [ [AS] 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.

Konverzió relációs algebrába

SELECT [DISTINCT] kif₁ [[AS] onév₁], ..., kif_n [[AS] onév_n]
FROM R₁ [[AS] t₁], ..., R_n [[AS] 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 attribútumnak egyedi neve legyen.
- 2.) A WHERE záradékot átalakítjuk egy kiválasztási feltétellé, melyet alkalmazunk az elkészített szorzatra.
- 3.) Végül a SELECT záradék alapján létrehozzuk a kifejezések listáját, a (kiterjesztett) vetítési művelethez.

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

Halmazműveletek az SQL-ben

- Mi hiányzik még, hogy a relációs algebra alpműveleteit az SQL-ben vissza tudjuk adni az összes műveletet?
- Halmazműveletek: **unió**, **különbség** és **metszet**, ebből az unió és különbség alpművelet, mindhárom használható, SQL-ben is implementálva van mind a három).
- 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.

(SFW-lekérdezés1)

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

(SFW-lekérdezés2);

Példa: Intersect (metszet)

- Szeret(név, sör), Felszolgál(söröző, sör, ár) és Látogat(név, söröző) táblák felhasználásával keressük

Trükk: itt ez az alkérdés valójában az adatbázisban tárolt tábla azokat (név,sör) párokat, ahol a név = sörivó látogat olyan sörözőt, ahol felszolgálnak olyan sört, amelyet szeret (a „boldog” sörivók).

(SELECT * FROM Szeret) (név, sör) párok, ahol a sörivó látogat olyan sörözőt, ahol ezt a sört felszolgálják

INTERSECT

**(SELECT név, sör
FROM Látogat L, Felszolgál F
WHERE L.söröző = F.söröző);**

Halmazműveletek multihalmaz értelemben

- **Unió:** $R \cup S$ -ben egy t sor annyiszor fordul elő ahányszor előfordul R -ben, plusz ahányszor előfordul S -ben: $n+m$
- **Metszet:** $R \cap S$ -ben egy t sor annyiszor fordul elő, amennyi az R -ben és S -ben lévő előfordulások minimuma: $\min[n, m]$
- **Különbség:** $R - S$ -ben egy t sor annyiszor fordul elő, mint az R -beli előfordulások mínusz az S -beli előfordulások száma, ha ez pozitív, egyébként pedig 0, vagyis $\max[0, n-m]$
- $(R \cup S) - T =? (R - T) \cup (S - T)$ (Ez Hz: igen, multihz:nem)

R		S		A	B
A	B	A	B	1	3
1	3	1	3	1	2
1	2	2	5	1	3
				2	5

\cup \Rightarrow

Halmaz-multihalmaz szintaxis

- Halmazműveletek: **unió**, **különbség** és **metszet**, ebből az unió és különbség alapművelet, mindhárom használható.
- A **SELECT-FROM-WHERE** utasítások általában **multihalmaz** szemantikát használnak, külön kell kérni **DISTINCT**-tel ha halmazt szeretnénk kapni
- A **halmazműveleteknél** viszont a **halmaz szemantika** az érvényes, (a **multihalmaz szemantikát** kell kérni: **ALL**)
- A halmazműveleteket az SQL-ben lekérdezés között:

(SFW-lekérdezés1)

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

(SFW-lekérdezés2);

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)

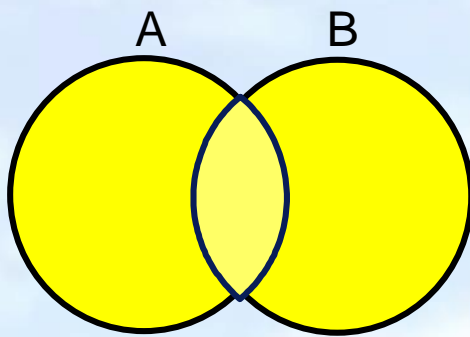
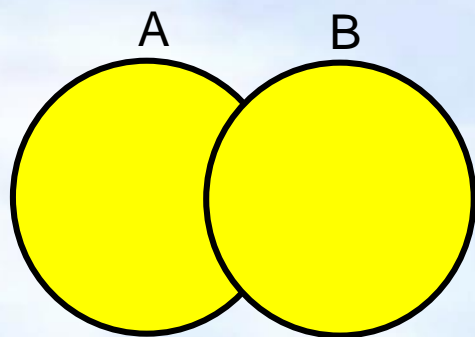
- Látogat(név, söröző) és Szeret(név, sör) táblák felhasználásával kilistázzuk azokat a sörivókat, akik több sörözőt látogatnak, mint amennyi sört szeretnek, és annyival többet, mint ahányszor megjelennek majd az eredményben

```
(SELECT név FROM Látogat)
```

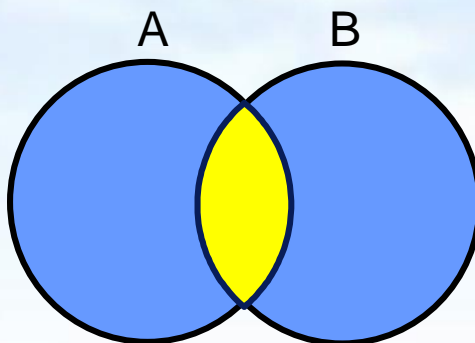
```
EXCEPT ALL
```

```
(SELECT név FROM Szeret);
```

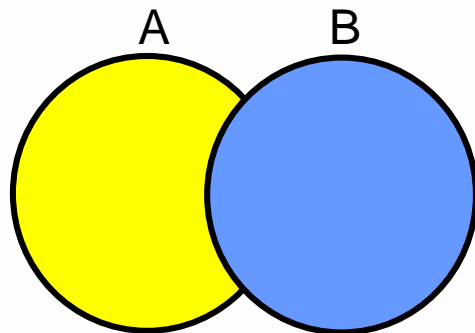
Halmazműveletek az Oracle-ben



UNION/UNION ALL



INTERSECT



MINUS

ORACLE-ben EXCEPT helyett a MINUS szót használjuk, és

UNION - UNION ALL lehet csak.

6.3.1.-6.3.5. és 6.4.7. Alkérdések

Alkérdeések

- Zárójelezett SFW SELECT-FROM-WHERE utasításokat **(alkérdeéseket)** is használhatunk a **FROM listán** a **WHERE záradékban**, és a **HAVING záradékban**.
- **Szintaktikus alakja**: zárójelbe kell tenni a lekérdezést
- **Hol használható?** Ott, ahol relációnevet használunk:
 - (1) **FROM listában**: új listaelem (rel.név változó SQL-ben)
(lekérdezés) [AS] sorváltozó
Ez felel meg annak, ahogyan a relációs algebrában tetsz.helyen használhattuk a lekérdezés eredményét.
 - (2) **WHERE záradékban**: sorokra vonatkozó feltételekben
 - (3) **HAVING záradékban**: csoportokra vonatk.feltételekben

Alkérdeések használata FROM listán

- **FROM záradékban** a tényleges relációk helyett alkérdeéssel létrehozott ideiglenes táblát is megadhatunk. Minthogy az alkérdeés eredményrelációjának nincs neve, meg kell adnunk egy másodnevet (a sorváltozó nevét), ezzel tudunk az alkérdeés által létrehozott reláció soraira hivatkozni.
- **Szintaktikus alakja:**
 - (lekérdezés) [AS] sorváltozó
- Ez felel meg annak, ahogyan a relációs algebrában tetsz.helyen használhattuk a lekérdezés eredményét.
- **Szemantikája:** A FROM záradékban kiértékelődik az alkérdeés, utána a sorváltozót ugyanúgy használjuk, mint a FROM lista többi közösleges relációk soraira.

Alkérdések használata FROM listán

- **Példa:** Keressük meg a Joe's bár vendégei által kedvelt söroket (a feladatnak sok megoldása van)

```
SELECT sör
FROM Szeret SZ,
      (SELECT név FROM Látogat
       WHERE söröző = 'Joe' 's bar') JD
WHERE SZ.név = JD.név;
```

Sörivők, akik látogatják Joe's bárja sörözőt.

- **H.F.:** Írjuk át egyszerű SELECT FROM utasításra, ahol csak táblák vannak, nincs alkérdés a FROM-ban

Alkérdeések a WHERE záradékban

WHERE záradékban:

- (i) Az alkérdés eredménye egyetlen **skalárérték**, vagyis az alkérdés olyan, mint a konstans, ami egy új elemi kifejezésként tetszőleges kifejezésben használható.
- (ii) **Skalár értékekből álló multihalmaz** logikai kifejezésekben használható:
 - [NOT] EXISTS (lekérdezés)
 - kifejezés [NOT] IN (lekérdezés)
 - kifejezés Θ [ANY | ALL] (lekérdezés)
- (iii) **Teljes, többdimenziós tábla** a visszatérő érték:
 - [NOT] EXISTS (lekérdezés)
 - (kif₁, ... kif_n) [NOT] IN (lekérdezés)

Alkérdések a WHERE záradékban

- Milyen változók szerepelhetnek egy alkérdésben?
 - Lokális saját változói a saját FROM listáról
 - Külső kérdés változói: ekkor az alkérdés korrelált.
- **Definíció:** Ha az alkérdés a külső kérdés FROM listában szereplő tábláiból származó attribútumot is tartalmaz, akkor azt **korrelált alkérdés**nek nevezzük.

Szemantikája

- Ha az alkérdés **nem korrelált**, önállóan kiértékelhető és ez az eredmény a külső kérdés közben nem változik, a külső kérdés szempontjából ez egy konstanstábla, akkor a kiértékelés mindig a legbelsőből halad kifelé.
- **Korrelált alkérdés**, lásd később az EXISTS példánál

Skalár értéket visszaadó alkérdések

- Ha egy alkérdés biztosan egy attribútumon egy sort ad vissza eredményként (egyelemű), akkor úgy használható, mint egy konstans érték.
 - az eredmény sornak egyetlen oszlopa van.
 - Futásidejű hiba keletkezik, ha az eredmény nem tartalmaz sort, vagy több sort tartalmaz.
- **Példa:** Felszolgál(söröző, sör, ár) táblában keressük meg azokat a sörözőket, 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?

Skalár értéket visszaadó alkérdések

Példa: Felszolgál(söröző, sör, ár) táblában keressük meg azokat a sörözőket, ahol a Miller ugyanannyiba kerül, mint Joe bárjában a Bud.

```
SELECT söröző
```

```
FROM Felszolgál
```

```
WHERE sör = 'Miller' AND
```

```
    ár = (SELECT ár
```

```
        FROM Felszolgál
```

```
        WHERE söröző = 'Joe' 's bar'
```

```
        AND sör = 'Bud' );
```


Ennyit kér
Joe a Budért.

H.F: Adjunk meg ugyanezt alkérdés használata nélkül is!

Példa: Skalár értékű alkérdés

```
SELECT sör
FROM Felszolgal
WHERE ár = (
    SELECT MAX(ár)
    FROM Felszolgal);
```

A külső lekérdezés
Felszolgaljának söre
a legdrágább sör



Skalár értékekből álló multihalmazt visszaadó alkérdések: 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 nem az alkérdés legkisebb elemével azonos.
 - Itt az alkérdés sorai egy mezőből állnak.


Skalár értékekből álló multihalmazt visszaadó alkérdések: ALL művelet

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

Példa: ALL

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

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



Az IN művelet a WHERE záradékban

- sor IN (alkérdés) akkor és csak akkor **igaz**, ha a sor eleme az alkérdés eredményének (itt a sor egy sor/tuple, nem sör)
- Tagadás: sor NOT IN (alkérdés).
- Az IN-kifejezések a WHERE záradékban jelenhetnek meg

➤ Példa:

```
SELECT *
```

```
FROM Sörök
```

```
WHERE név IN (SELECT sör
```

```
FROM Szeret
```

```
WHERE név = 'Fred' );
```

A sörök,
melyeket
Fred szeret.

Több mélységig beágyazhatóak

- Tankönyv Filmek példájában sorokat tartalmazó feltételek: Keressük Harrison Ford filmjeinek gyártásirányítóját

```
SELECT név
FROM GyártásIrányító
WHERE azonosító IN
    (SELECT producerAzon
     FROM Filmek
     WHERE (cím, év) IN
        (SELECT filmCím, filmév
         FROM SzerepelBenne
         WHERE színész = 'Harrison Ford')
    );
```

Van-e különbség a kiértékelés között?

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

a	b
1	2
3	4

R

b	c
2	5
2	6

S

```
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.

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.

Korrelált alkérdés

- **Definíció:** Ha egy alkérdés nem csak lokális változókat használ a saját FROM listájáról, hanem a külső kérdés FROM listában szereplő tábláiból származó attribútumot is tartalmaz, akkor azt **korrelált alkérdés**nek nevezzük.

Szemantikája

- Ha az alkérdés **nem korrelált**, önállóan kiértékelhető és ez az eredmény a külső kérdés közben nem változik, a külső kérdés szempontjából ez egy konstanstábla, akkor a kiértékelés mindig a legbelsőből halad kifelé.
- **Korrelált alkérdés**, többször kerül kiértékelésre, minden egyes kiértékelés megfelel egy olyan értékadásnak, amely az alkérdésen kívüli sorváltozóból származik.

Az EXISTS művelet a WHERE-ben

- EXISTS (alkérdés) akkor és csak akkor igaz, ha az alkérdés eredménye nem üres.
 - Tagadása: NOT EXISTS (alkérdés)
- **Példa: A Sörök(név, gyártó)** táblában keressük meg azokat a söröket, amelyeken kívül a gyártójuk nem gyárt másikat.
- Ez korrelált alkérdés, többször kerül kiértékelésre, a külső tábla minden sorára kiértékeljük az alkérdést.
- A korrelált lekérdezések használata közben figyelembe kell vennünk a nevek érvényességi körére vonatkozó szabályokat.

Példa: EXISTS

```
SELECT név
FROM Sörök s1
WHERE NOT EXISTS
  (SELECT *
   FROM Sörök
   WHERE gyártó = s1.gyártó
   AND név <> s1.név) ;
```

Azon s1
sörtől
különböző
sörök,
melyeknek
ugyanaz
a gyártója.

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

A „nem egyenlő”
művelet SQL-ben.

Tk.példa: Korrelált alkérdés

- A több, mint egyszer előforduló filmcímek megkeresése:

```
SELECT DISTINCT cím
FROM Filmek Régi
WHERE év < ANY
  (SELECT év
   FROM Filmek
   WHERE cím = Régi.cím
  );
```

Csoportosítás (GROUP BY) esetén alkérdések a HAVING záradékban

- A GROUP BY záradékot egy **HAVING <feltétel>** záradék követheti.
- HAVING feltétel az egyes csoportokra vonatkozik, ha egy csoport nem teljesíti a feltételt, nem lesz benne az eredményben.
- csak olyan attribútumok szerepelhetnek, amelyek:
 - vagy csoportosító attribútumok,
 - vagy összesített attribútumok.(vagyis ugyanazok a szabályok érvényesek, mint a SELECT záradéknál).

Példa alkérdésre a HAVING-ben --1

- Felszolgál(söröző, sör, ár) és Sörök(név, gyártó) táblák felhasználásával adjuk meg azon sörök árainak az összegét, amelyeket
 - legalább három sörözőben felszolgálnak,
 - vagy Pete a gyártójuk!

Példa alkérdésre a HAVING-ben --2

SELECT sör, SUM(ár)

FROM Felszolgál

GROUP BY sör

HAVING COUNT(söröző) >= 3 OR

sör IN (SELECT név

FROM Sörök

WHERE gyártó = 'Pete');

(HAVING...) Sör csoportok,
Melyeket legalább három
nem-NULL bárban árulnak,
Vagy Pete a gyártójuk.

(SELECT...)
Sörök, melyeket
Pete gyárt

6.3.6.-6.3.8. Összekapcsolások az SQL-ben

Összekapcsolások a FROM listán

- Az SQL-ben összekapcsolások számos változata megtalálható: Természetes összekapcsolás
- JOIN a USING (oszlopnév) illetve ON feltétel segítségével
- Théta-összekapcsolás (ON feltételben =, !=, <, <=, >, >=)
- Bal-, vagy jobboldali-, vagy teljes külső összekapcsolások
- Descartes-szorzat (direktszorzat, keresztszorzat)

```
SELECT tábla1.oszlop, tábla2.oszlop FROM tábla1  
[NATURAL JOIN tábla2] |  
[JOIN tábla2 USING (oszlopnév)] |  
[JOIN tábla2 ON (tábla1.oszlopnév = tábla2.oszlopnév)]  
[ {LEFT | RIGHT | FULL} OUTER JOIN tábla2  
  ON (tábla1.oszlopnév = tábla2.oszlopnév)]  
[CROSS JOIN tábla2]
```

Példa: Többtáblás lekérdezések

EMPLOYEES (DOLGOZÓK)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS (OSZTÁLYOK)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

Természetes összekapcsolás

- A NATURAL JOIN utasítás a benne szereplő két tábla azonos nevű oszlopain alapul.
- A két tábla azon sorait eredményezi, ahol az azonos nevű oszlopokban szereplő értékek megegyeznek.
- Ha az azonos nevű oszlopok adattípusa eltérő, akkor hibával tér vissza az utasítás.

```
SELECT employee_id, department_id, department_name  
FROM employees  
NATURAL JOIN departments ;
```


Összekapcsolás USING kulcsszóval

- Ha több oszlopnak azonos a neve, de az adattípusa eltérő, akkor a USING segítségével megadható, hogy mely oszlopokat lehet használni az egyenlőségen alapuló összekapcsoláshoz.
- Használjunk USING-ot, ha csak egy oszlop egyezik meg.
- Ne használjuk a tábla eredeti vagy alias nevét a kiválasztott oszlopok megadásánál.
- A NATURAL JOIN és a USING kulcsszavak együttes használata nem megengedett.

Oszlopnevek összekapcsolása

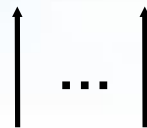
EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales



Foreign key

Primary key

- Az osztályok dolgozóinak meghatározásához a Departments tábla és az Employees tábla DEPARTMENT_ID oszlopaikban szereplő értékeinek összehasonlítása kell. Ez egy egyenlőségen alapuló összekapcsolás lesz. Az ilyen típusú összekapcsolásban általában az elsődleges- és az idegen kulcs komponensei szerepelnek.

A USING kulcsszó használata lekérdezésben

```
SELECT employees.employee_id, employees.last_name,  
       departments.location_id, department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
124	Mourgos	1500	50
141	Rajs	1500	50
142	Davies	1500	50
144	Vargas	1500	50
143	Matos	1500	50

...

19 rows selected.

Azonos nevű oszlopok megkülönböztetése

- Használjuk a táblaneveket előtagként az azonos nevű oszlopok megkülönböztetésére
- A előtagok használata javítja a hatékonyságot is.
- Használhatunk alias neveket az olyan oszlopokra, amelyeket megkülönböztetünk a többi táblában lévő azonos nevű társaiktól.
- Ne használjunk alias nevet azon oszlopokra, amelyeket a USING kulcsszó után adtunk meg és az SQL utasításban még más helyen is szerepelnek.

Sorváltók használata tábláknál

- A lekérdezések átláthatósága miatt használhatunk sorváltót (tábla alias neveket).
- A sorváltók használata javítja a lekérdezés teljesítményét.
- A sorváltók maximum 30 karakter hosszúak lehetnek (minél rövidebb annál jobb)
- A sorváltók csak az aktuális SELECT utasítás során lesznek használhatóak!

Összekapcsolások az ON kulcsszó segítségével

- A természetes összekapcsolás alapvetően az azonos nevű oszlopok egyenlőségvizsgálatán alapuló összekapcsolása volt.
- Az ON kulcsszót használhatjuk az összekapcsolás tetszőleges feltételének vagy oszlopainak megadására.
- Az összekapcsolási feltétel független a többi keresési feltételtől.
- Az ON használata áttekinthetőbbé teszi a kódot

Lekérdezés az ON kulcsszó használatával

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

...

19 rows selected.

- Az ON segítségével különböző nevű oszlopok is összekapcsolhatóak

Önmagával való összekapcsolás (self-join) az ON kulcsszóval 1.

EMPLOYEES (WORKER)

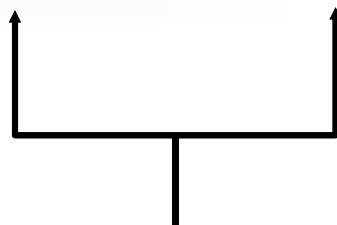
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

...



**A WORKER tábla Manager_ID mezője megfelel
a MANAGER tábla EMPLOYEE_ID mezőjével**

Önmagával való összekapcsolás (self-join) az ON kulcsszóval 2

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

...

Három-utas összekapcsolás ON segítségével

```
SELECT employee_id, city, department_name
FROM employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

- **Három tábla összekapcsolását nevezük három-utas összekapcsolásnak**
Az SQL 1999-es szintaxis szerint az ilyen összekapcsolások balról jobbra haladva hajtódnak végre (DEPARTMENTS – EMPLOYEES) – LOCATION

Nem egyenlőségvizsgálaton alapuló összekapcsolás

EMPLOYEES

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

...

20 rows selected.

JOB_GRADES

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

Az EMPLOYEES tábla fizetés mezőjének értéke a JOBS_GRADE tábla legmagasabb illetve legalacsonyabb fizetés közötti kell legyen.

Példa a nem egyenlőségvizsgálaton alapuló összekapcsolás

```
SELECT e.last_name, e.salary, j.grade_level
FROM employees e JOIN job_grades j
ON e.salary
   BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

Külső összekapcsolás

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...
20 rows selected.

**A 190-es számú osztályon
nincs alkalmazott**

Belső vagy külső összekapcsolás?

- SQL-1999: Belső összekapcsolásnak nevezzük azokat az összekapcsolásokat, amelyek két tábla megegyező soraival térnek vissza.
- Két tábla olyan összekapcsolását, amely a belső összekapcsolás eredményéhez hozzáveszi a bal (vagy jobboldali) tábla összes sorát, baloldali (vagy jobboldali) külső összekapcsolásnak nevezzük.
- Teljes külső összekapcsolásnak hívjuk azt az esetet, amikor a külső összekapcsolás egyszerre bal- és jobboldali.

Baloldali külső összekapcsolás

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

Jobboldali külső összekapcsolás

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id) ;
```

...

Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

20 rows selected.

Teljes külső összekapcsolás

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing

...

A direkt szorzat

- A direkt-szorzat a következőként kapható:
 - az összekapcsolási feltétel elhagyásával,
 - nem megengedett összekapcsolási feltétellel,
 - az első tábla összes sorának összekapcsolása a másik tábla összes sorával.
- A direkt szorzatok elkerülése érdekében, mindig kell legalább egy megengedett összekapcsolási feltétel legyen.

A direkt szorzat

EMPLOYEES (20 rows)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8 rows)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

Direkt-szorzat :
20 x 8 = 160 sor

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

A direkt szorzat

- A `CROSS JOIN` kulcsszó előállítja két tábla kereszt-szorzatát (vagyis a direkt szorzatát)

```
SELECT last_name, department_name  
FROM employees CROSS JOIN departments ;
```

Az előző 5.előadás relációs algebrai példáját Termékek (Tk.2.4.1.feladat) nézzük meg SQL-ben!

Legyen adott az alábbi **relációs sémák** feletti relációk:

Termék (gyártó, modell, típus)

PC (modell, sebesség, memória, merevlemez, ár)

Laptop (modell, sebesség, memória, merevlemez, képernyő, ár)

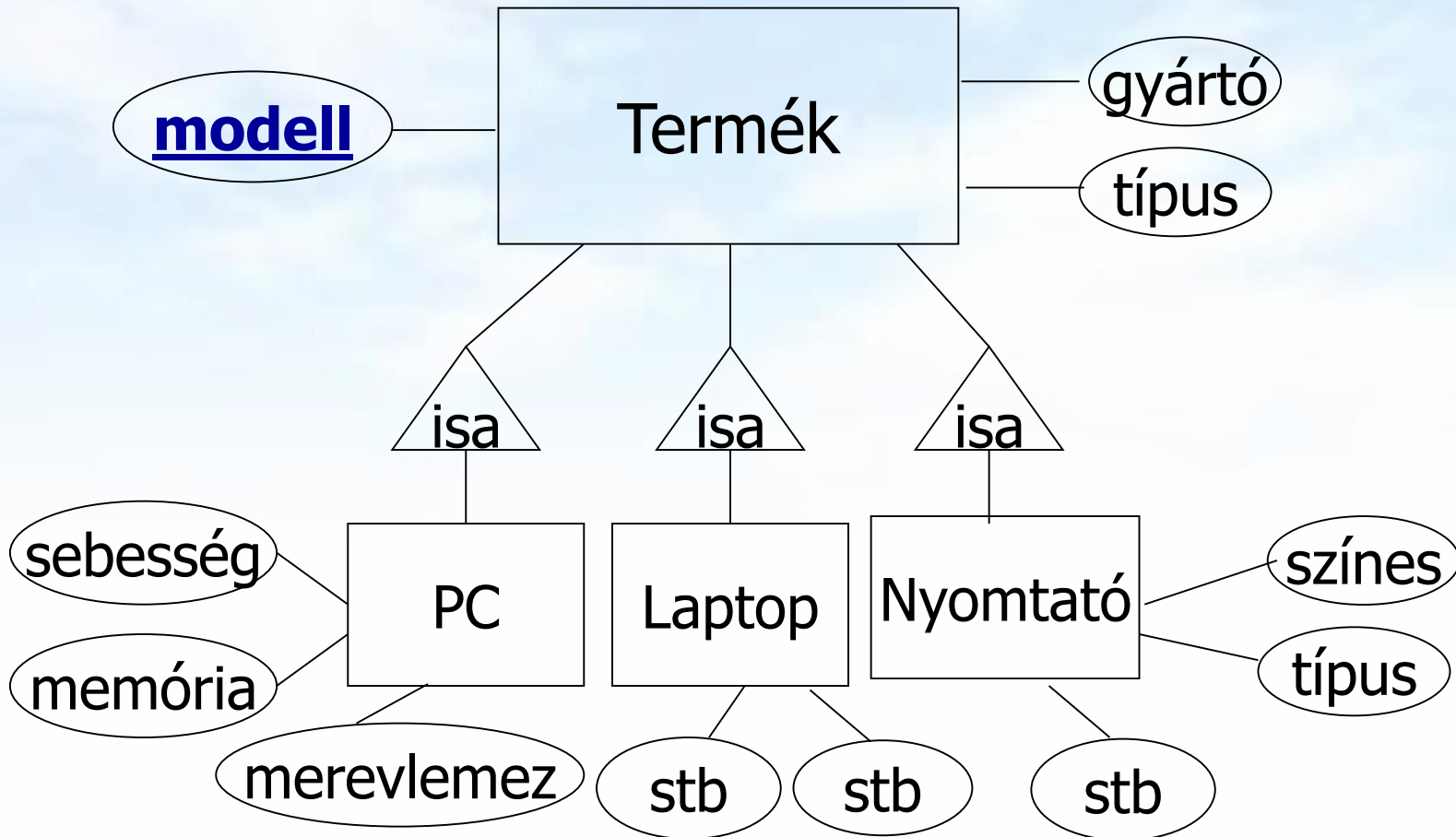
Nyomtató (modell, színes, típus, ár)

Feladatok Tk.2.4.1.feladat (ezeket a kérdéseket konkrét táblák alapján természetes módon meg lehet válaszolni, majd felírjuk relációs algebraiban)

- b) Mely gyártók készítenek legalább száz gigabájt méretű merevlemezrel rendelkező laptopot?
- c) Adjuk meg a B gyártó által gyártott összes termék modellszámát és árát!
- e) Melyek azok a gyártók, amelyek laptopot árulnak, PC-t viszont nem?
- h) Melyek azok a gyártók, amelyek gyártanak legalább két, egymástól különböző legalább 2.80 gigahertzen működő számítógépet?
- i) Melyik gyártó gyártja a leggyorsabb számítógépet (laptopot vagy PC-t)?
- k) Melyek azok a gyártók, akik pontosan három típusú PC-t forgalmaznak? (relációs algebraiban táblák közötti műveletekkel)

Emlékeztetőül a Termékek példához

E/K modell: Erre megnéztünk háromféle stratégiát is hogyan alakítjuk át relációsémákra (1 v. 3 v. 4 táblára)



Példa: Termékek (Tk.2.4.1.feladat)

- Relációs algebra kifejezések ilyen bevezetése valóban használható a lekérdezések megadására?
 - Tk.2.4.1.feladat
 - **Példa:** Adottak az alábbi **relációs sémák** feletti relációk
Termék (gyártó, modell, típus)
PC (modell, sebesség, memória, merevlemez, cd, ár)
Laptop (modell, sebesség, memória, merevlemez, képernyő, ár)
Nyomtató (modell, színes, típus, ár)
 - Jelölje: T(gy, m, t)
PC(m, s, me, ml, ár)
L(m, s, me, ml, k, ár)
Ny(m, sz, t, ár)
- Megj.: a két típus attr.név nem ugyanazt fejezi ki és így $T \bowtie Ny$ természetes összekapcsolásnál „zűr”

Probléma: természetes összekapcsolás

Termek táblának modell elsődleges kulcsára hivatkozik a Nyomtato táblában modell a külső kulcs (hivatkozás)

Természetes összekapcsolás: Itt hibás eredményt kapunk!

```
SELECT modell, gyarto, tipus
```

```
FROM Termek NATURAL JOIN Nyomtato;
```

-- Hiba: modell, tipus (két oszlopnak is megegyezik a neve)

-- mivel a tipus mást jelent a két táblában, ezért ÜRES lesz

A natural join művelet új szintaxisával megadhatjuk a kapcsolómezőket is, és ez jó megoldást ad:

```
SELECT modell, gyarto, T.tipus, N.tipus
```

```
FROM Termek T JOIN Nyomtato N
```

```
USING ( modell );
```


Összekapcsolási feltétel megadásával

Descartes szorzattal (FROM listán a táblákat megadjuk, és a WHERE-be írjuk be a kapcsolást, itt a DBMS joint végez)

```
SELECT T.modell, gyarto, T.tipus, N.tipus  
FROM Termek T, Nyomtato N  
WHERE T.modell = N.modell;
```

Ugyanebben a szemléletben az új szintaxissal:

```
SELECT T.modell, gyarto, T.tipus, N.tipus  
FROM Termek T JOIN Nyomtato N  
ON T.modell = N.modell;
```

Példák SELECT és algebra átírásokra

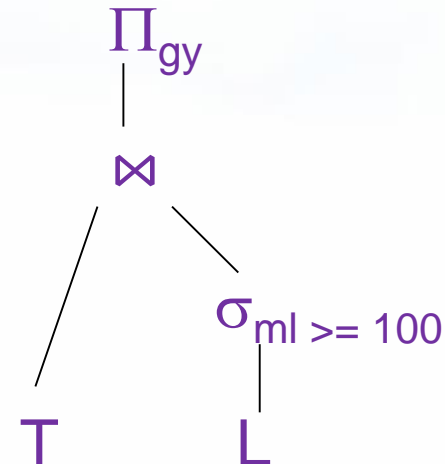
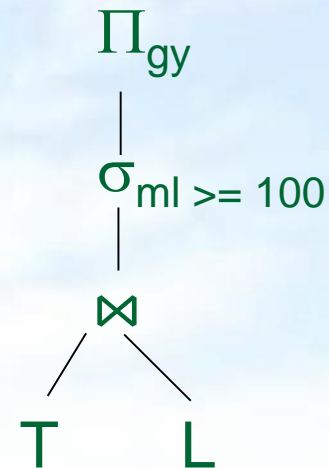
b.) Mely gyártók készítenek legalább száz gigabájt méretű merevlemezzel rendelkező laptopot?

$\Pi_{gy} (\sigma_{ml \geq 100} (T \bowtie L))$

SELECT gyarto
FROM Termek NATURAL JOIN Laptop
WHERE merevlemez >= 100;

SELECT gyarto
FROM Termek T JOIN Laptop L
ON T.modell=L.modell
WHERE merevlemez >= 100;

-- és alkérdéssel:
SELECT gyarto FROM Termek
WHERE modell IN
(SELECT modell FROM Laptop
WHERE merevlemez >= 100);



Félig-összekapcsolások és az alkérdések

- Az R és S relációk **félig-összekapcsolása** $R \bowtie S$ (semijoin) az R azon sorainak halmaza, amely sorok megegyeznek az S legalább egy sorával az R és S összes közös attribútumán.
- **H.F:** Adjunk meg olyan három különböző relációs algebrai kifejezést, amelyek ekvivalensek az $R \bowtie S$ kifejezéssel!
- Nézzük meg azokat az összekapcsolásokat, ahol csak az első reláció attribútumait kérdezzük le:

Félig-összekapcsolások (semijoin)

- Legyen $R(X, Y)$ és $S(Y, Z)$, ahol X, Y, Z attr.halmazok
Az R és S relációk **félig-összekapcsolása** $R \bowtie S$ (semijoin)
 - (1a) `SELECT X, Y FROM R NATURAL JOIN S;`
 - (1b) `SELECT X, Y FROM R JOIN S USING (Y);`
 - (2a) `SELECT X, R.Y FROM R, S WHERE R.Y=S.Y;`
 - (2b) `SELECT X, R.Y FROM R JOIN S ON R.Y=S.Y;`
 - (3a) `SELECT X,Y FROM R --- spec. ha Y egy attribútum:
WHERE Y = ANY (SELECT Y FROM S);`
 - (3b) `SELECT X,Y FROM R --- általában is Y attr.halmaz:
WHERE Y IN (SELECT Y FROM S);`
 - (4) `SELECT X,Y FROM R
WHERE EXISTS (SELECT * FROM S WHERE Y=R.Y);`

Félig-összekapcsolás (alkérdésekkel)

b.) Mely gyártók készítenek legalább száz gigabájt méretű merevlemezzel rendelkező laptopot?

IN (alkérdéssel)

```
SELECT gyarto FROM Termek
WHERE modell IN
  (SELECT modell FROM Laptop
   WHERE merevlemez >= 100);
```

EXISTS (alkérdéssel)

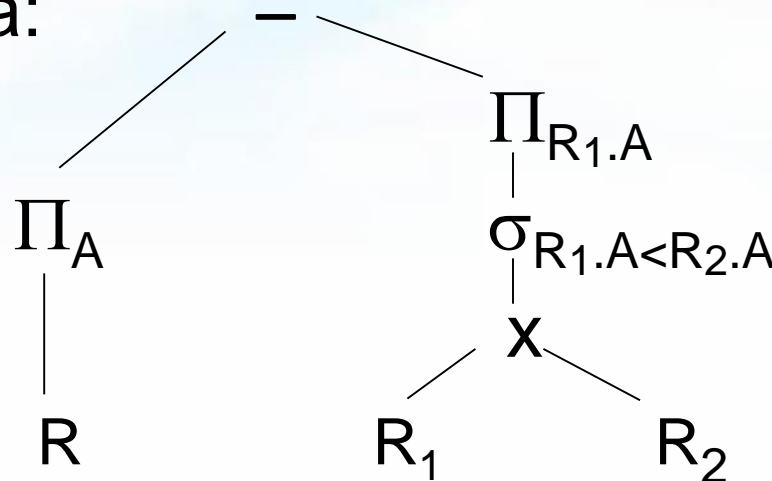
```
SELECT gyarto FROM Termek T
WHERE EXISTS (SELECT * FROM Laptop
             WHERE T.modell=modell
             AND merevlemez >= 100);
```

i.) feladat MAX előállítása relációs algebrában

- Nézzük meg a maximum előállításának a kérdését! Legyen $R(A,B)$. **Feladat:** Adjuk meg $MAX(A)$ értékét! (Alap relációs algebrában nincs MAX összesítő függvény, sem csoportosítás).

- $\pi_A(R) - \pi_{R1.A}(\sigma_{R1.A < R2.A}(\rho_{R1}(R) \times \rho_{R2}(R)))$

- Kiértékelő fa:



[folyt.] Rel.algebrai kifejezés átírása SQL-re

- Előző oldal folyt.max előállítás átírása SQL-re:
- Kiértékelő fa szerinti átírás SQL-be:

```
(SELECT A FROM R)
  MINUS
(SELECT R1.A AS A
 FROM R R1, R R2
 WHERE R1.A<R2.A);
```

- Egy másik megoldása alkérdéssel (antijoin)

```
SELECT A FROM R MAXA
  WHERE NOT EXISTS
    (SELECT A FROM R
     WHERE A > MAXA.A);
```

Példa: külső join és csoportosítás ---1

Adott relációs sémák feletti relációk (gyakorlat példája)

Dolgozo (dkod, dnev, foglalkozas, fonoke, belepes,
fizetes, jutalek, oazon)

Osztaly (oazon, onev, telephely)

Adjuk meg osztályonként a dolgozók összfizetését az osztály nevét megjelenítve ONEV, SUM(FIZETES) formában, és azok az osztályok is jelenjenek meg ahol nem dolgozik senki, ott az összfizetés 0 legyen.

Ha van olyan dolgozó, akinek nincs megadva, hogy mely osztályon dolgozik, azokat a dolgozókat egy 'FIKTIV' nevű osztályon gyűjtsük össze. Minden osztályt a nevével plusz ezt a 'FIKTIV' osztályt is jelenítsük meg az itt dolgozók összfizetésével.

Példa: külső join és csoportosítás ---2

```
SELECT NVL(onev, 'Fiktív') osztály,  
       NVL(AVG(fizetes), 0) + 100 emelt  
FROM dolgozo d FULL OUTER JOIN osztaly o  
   ON d.oazon=o.oazon  
GROUP BY o.oazon, onev  
ORDER BY emelt;
```

$$\tau_{\text{emelt}} \left(\pi_{\text{onev} \rightarrow \text{osztaly}, \text{avg}(\text{fizetes})+100 \rightarrow \text{emelt}} \left(\gamma_{\text{o.oazon}, \text{onev}, \text{avg}(\text{fizetes})} (d \bowtie_o o) \right) \right)$$

Kérdés / Válasz

- **Köszönöm a figyelmet! Kérdés/Válasz?**
- **Összefoglalva:** Több relációra vonatkozó lekérdezések és az alkérdések használata az SQL SELECT utasítás FROM, WHERE és HAVING záradékaiban, továbbá összekapcsolások az SQL-ben (Tankönyv 6.2.-6.3.)

Feladatok

- **Házi feladat: Gyakorlás az Oracle Példatár feladatai:**
Példatár 3. fejezetek feladatai SQL-lekérdezésekben
- **Keressünk új megoldásokat!** „Amikor azt gondolod, hogy már minden lehetőséget kimerítettél, még mindig van legalább egy.” (Thomas Alva Edison)