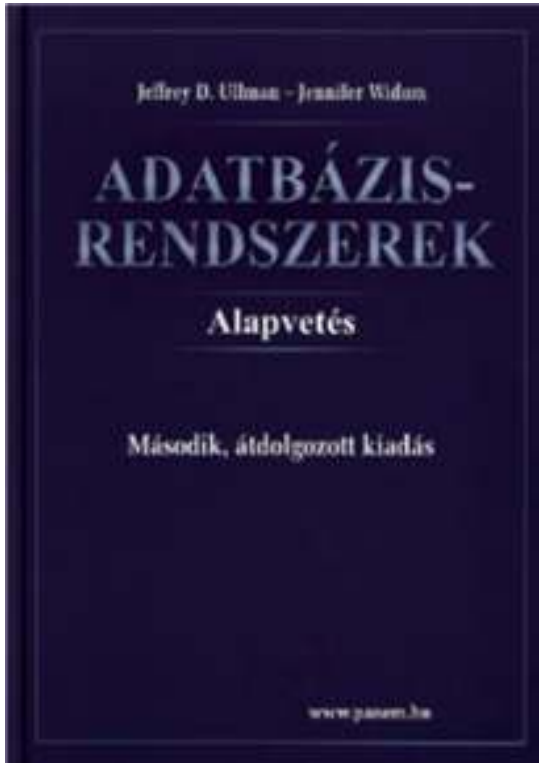
Relációs algebra kiterjesztése (SQL) csoportosítás és összesítések

Tankönyv: Ullman-Widom:

Adatbázisrendszerek Alapvetés

Második, átdolgozott kiadás,

Panem, 2009



5.1.- 5.2. Relációs műveletek multihalmazokon és kiterjesztett műveletek a relációs algebrában

6.4.3.-6.4.7. Csoportosítás és összesítések az SQL-ben

6.2. Több relációra vonatkozó lekérdezések az SQL-ben

6.4.1.-6.4.2. Ismétlődések kezelése, Halmazműveletek SQL-ben

Ismétlés: Relációs algebra

Relációs algebrai kifejezések formális felépítése

- 1.) Elemi kifejezések
  - (i) $R_i \in \mathbb{R}$ (az adatbázis-sémában levő relációnevek)
  - (ii) konstans reláció (véges sok, konstansból álló sor)
- 2.) Összetett kifejezések
- Ha E_1, E_2 kifejezések, akkor a következő E is kifejezés
  - $E := \Pi_{lista} (E_1)$ vetítés (típus a lista szerint)
  - $E := \sigma_{Feltétel} (E_1)$ kiválasztás (típus nem változik)
  - $E := E_1 \cup E_2$ unió, ha azonos típusúak (és ez a típusa)
  - $E := E_1 - E_2$ különbség, ha E_1, E_2 azonos típusúak (típus)
  - $E := E_1 \bowtie E_2$ term. összekapcsolás (típus attr-ok uniója)
  - $E := \rho_{S(B1, \ldots, Bk)} (E_1 (A_1, \ldots A_k))$ átnevezés (típ.új attr.nevek)
  - $E := (E_1)$ kifejezést zárójelezve is kifejezést kapunk
- Ezek és csak ezek a kifejezések, amit így meg tudunk adni

Ismétlés: Egy táblás lekérdezések

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], … , kif_n [$onév_n$]

FROM táblanév [sorváltozó]

[WHERE feltétel]

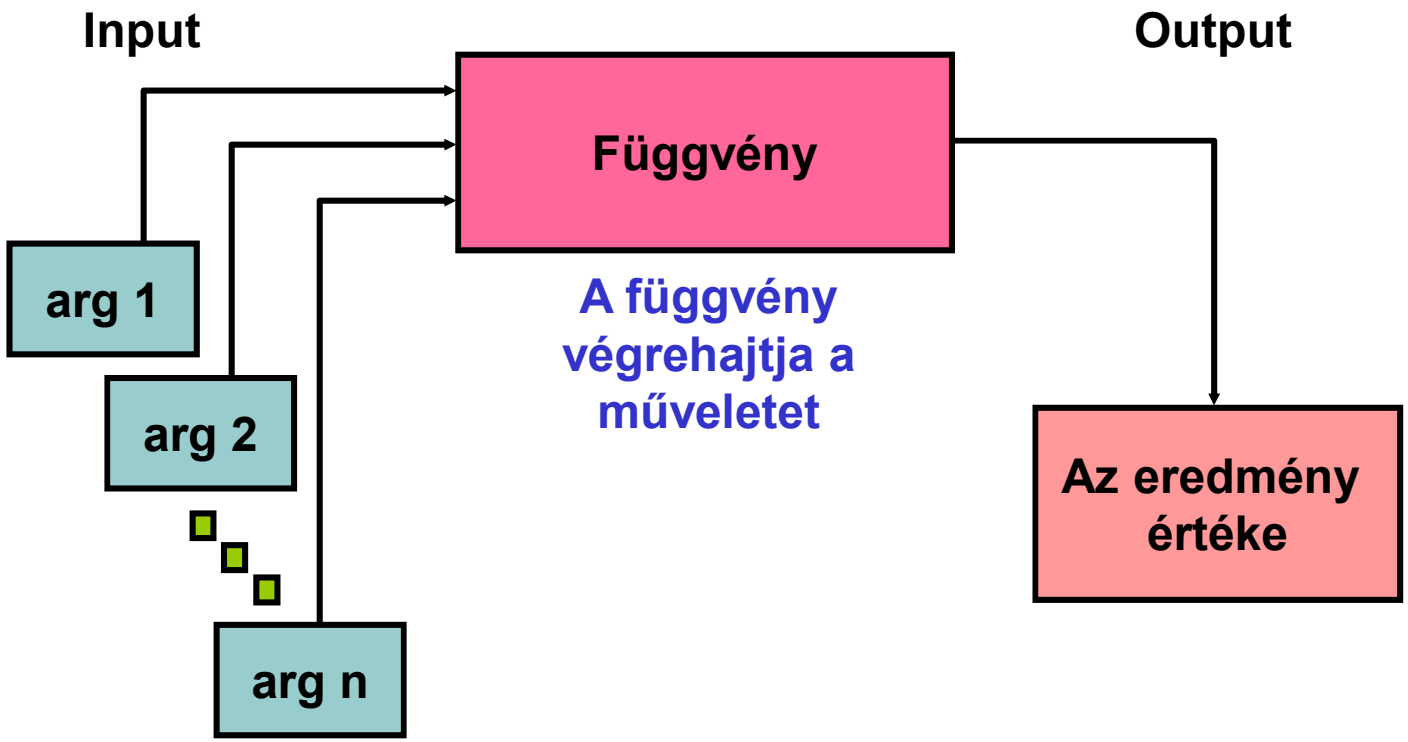
Alapértelmezés (a műveletek szemantikája)

- A FROM záradékban levő relációhoz tekintünk egy **sorváltozót**, amely a reláció minden sorát bejárja
- Minden egyes „aktuális” sorhoz **kiértékeljük** a WHERE záradékot. Ha **igaz** választ kaptunk (vigyázat: 3 értékű logika, T: igaz, F: hamis és U: ismeretlen), akkor erre:
- A SELECT záradékban szereplő **kifejezéseknek** megfelelően képezzük egy sort (output tábla egy sora)
- Ha van DISTINCT, akkor az ismétlődő sorokat elhagyjuk

Ismétlés: vetítés és kiválasztás SQL-ben

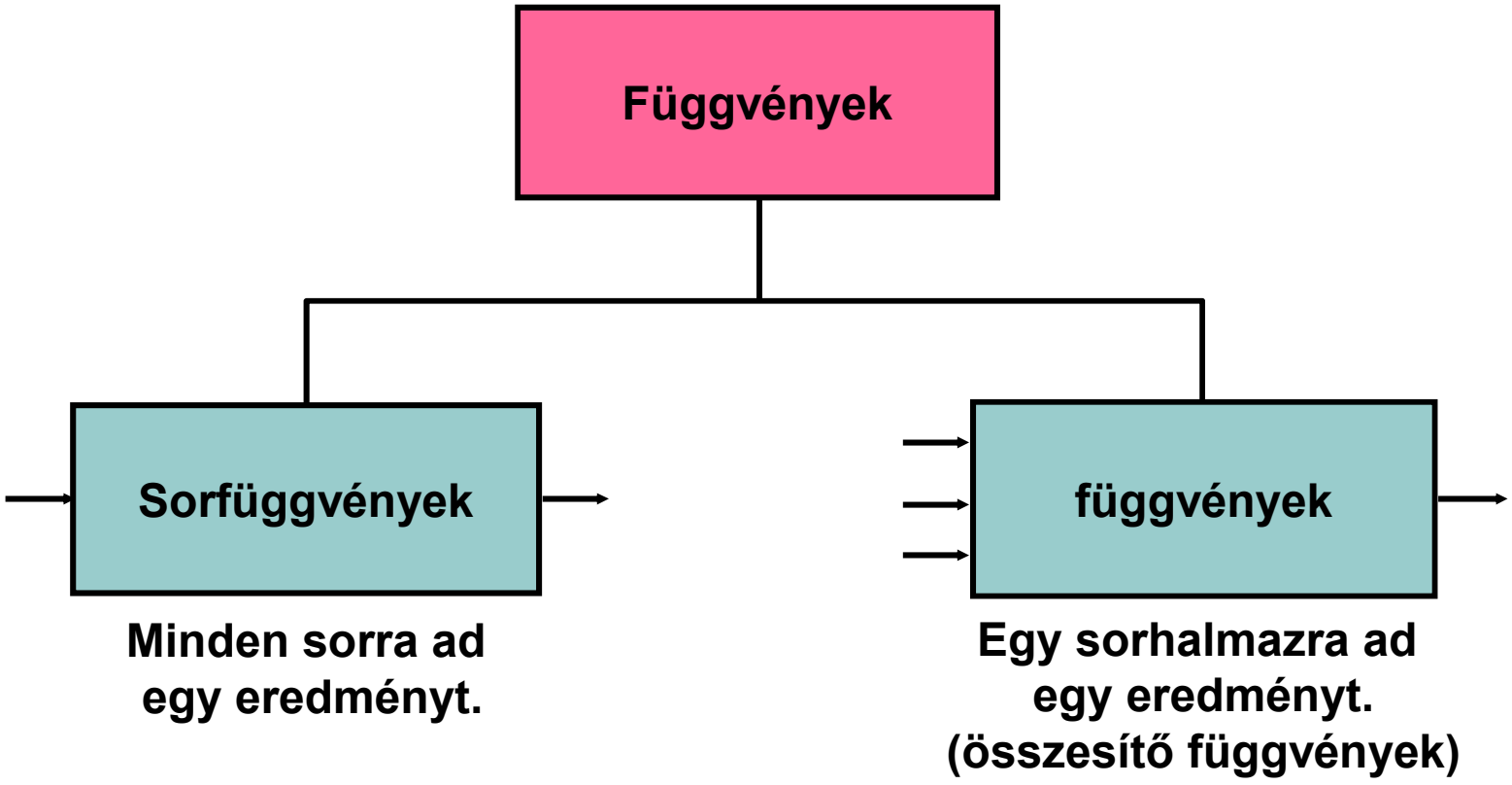
- Az SQL-ben halmazok helyett **multihalmazokat** használunk (egy sor többször is előfordulhat)
- $\Pi_{\text{Lista}}\ \sigma_{\text{Feltétel}}(\mathbf{R})$ kifejezésben a vetítés L-listája és a kiválasztás F-feltétele az attribútumnevek helyén **kifejezések** állnak, amely függvényeket és műveleti jeleket is tartalmazhat
- SQL leggyakrabban használt sorfüggvények:
- Numerikus, karakteres, dátum, konverziós, általános, például NULL értéket megadott értékkel helyettesítő NVL és COALESCE sorfüggvényeket a 2.gyakorlaton néztük meg Oracle SQL sorfüggvényekre a 3.leckét
- 2.gyak.volt: Oracle Példatár 1.fejezet 1.1-1.20 feladatai
- jön 3.gyak: Oracle Példatár 2.fejezet 2.1-2.24 feladatai

Oracle gyak: SQL-függvények

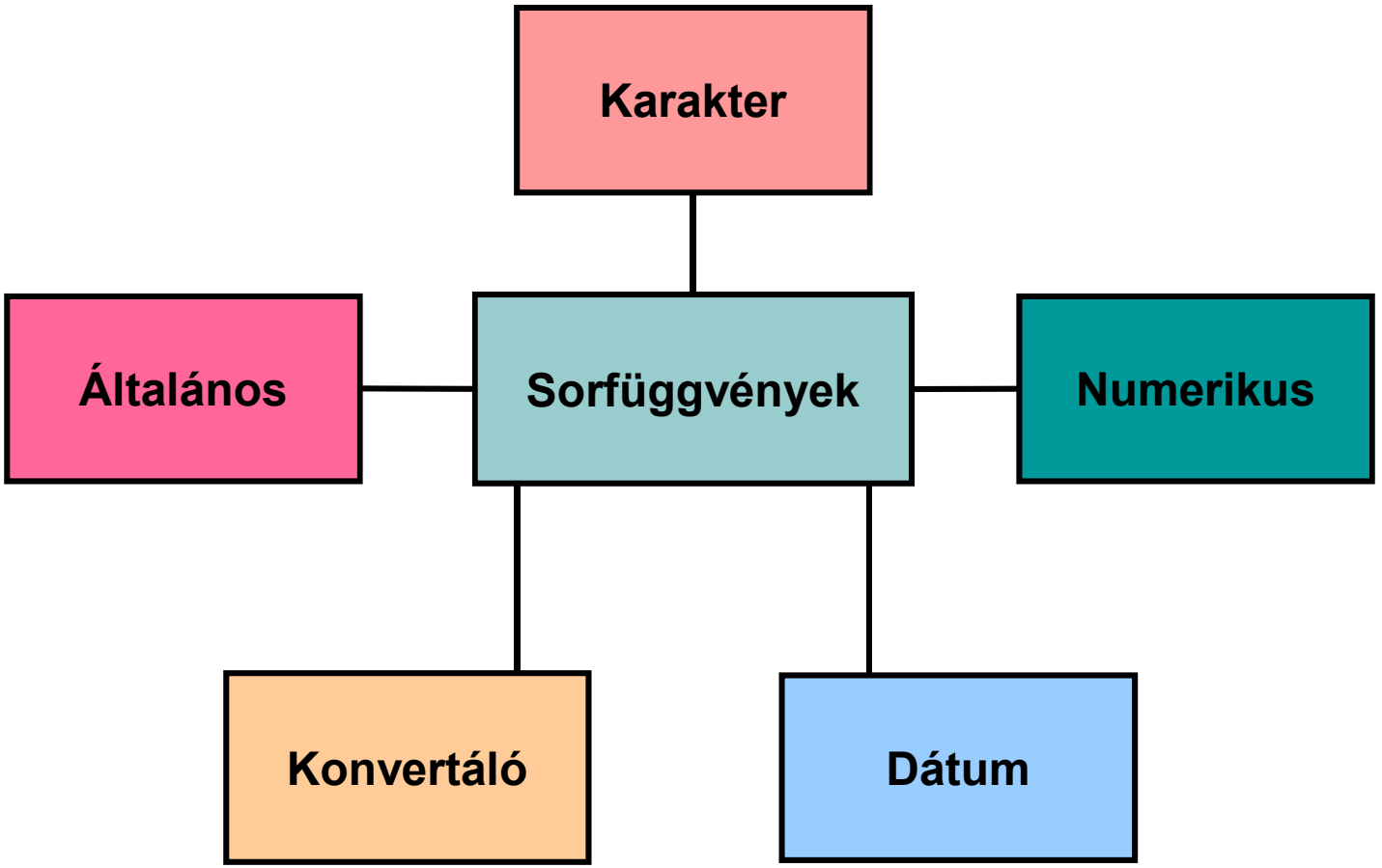


A gyakorlaton bemutatott függvények többsége Oracle-specifikus.

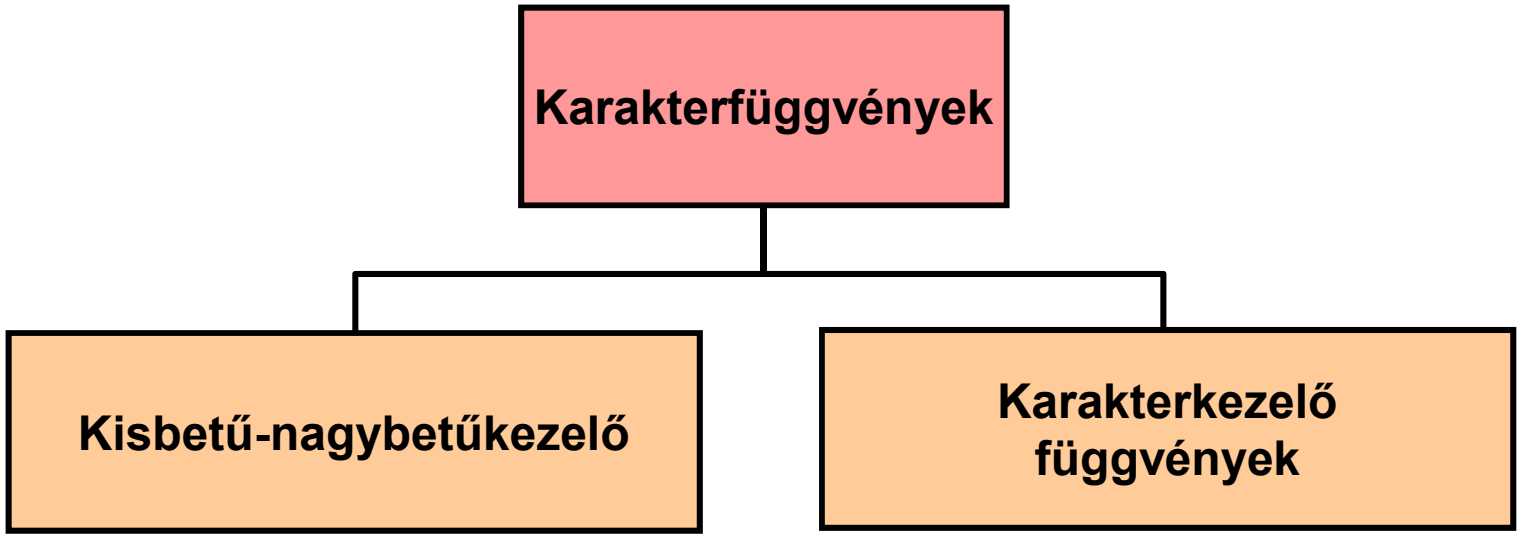
Az SQL-függvények két típusa



Sorfüggvények



Karakterfüggvények



Karakterfüggvények

Függvény	Leírás
LOWER(*column*|*expression*)	**Kisbetűre konvertál**
UPPER(*column*|*expression*)	**Nagybetűre konvertál**
INITCAP(*column*|*expression*)	**A szavak első betűjét nagybetűre, a többit kisbetűre konvertálja**
CONCAT(*column1*|*expression1*, *column2*|*expression2*)	**A két karakterértéket összefűzi; ugyanaz mint a || művelet.**
SUBSTR(*column*|*expression*,*m*[,*n*])	**Az m-ik karaktertől kezdődően n karaktert ad vissza. (Ha *m* negatív, akkor a végétől vett m-ik karaktert jelenti.) Ha *n* hiányzik, akkor az összes karatert megkapjuk a karakterlánc végéig.**

Karakterfüggvények

Függvény	Leírás
LENGTH(*column|expression*)	**A karakterlánc hossza.**
INSTR(*column|expression, 'string', [,m], [n]*)	**A karaktereslánc a kifejezésben balról az m-ik betűtől számítva hanyadik helyen fordul elő először. Kereshetjük az n-ik előfordulás kezdő helyét is. Alapértelmezésben m=n=1.**
LPAD(*column|expression, n, 'string'*) **RPAD(*column|expression, n, 'string'*)**	**A szöveget kiegészíti balról a megadott karakterekkel az adott hosszig,** **A szöveget kiegészíti jobbról a megadott karakterekkel az adott hosszig,** **Karaktereket nem kötelező megadni, ekkor szóközt használ.**
TRIM(*leading|trailing|both, trim_character FROM trim_source*)	**A karakterlánc elejéről és/vagy végéről eltávolítja a szóközöket, illetve a megadott karaktert.**
REPLACE(*text, search_string, replacement_string*)	**A szövegben lecseréli egy szövegrész összes előfordulását a megadott szövegre. Ha az utóbbit nem adjuk meg, akkor törli a keresett szöveget.**

Numerikus függvények

- ROUND: Adott tízedesjegyre kerekít (ha n negatív,
  akkor a tízedesvesszőtől balra kerekít).
- TRUNC: Adott tízedesjegy utáni részt levágja
- MOD: A maradékos osztást maradékát adja vissza

Függvény	Eredmény
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

Dátumfüggvények

Függvény	Eredmény
`MONTHS_BETWEEN(date1, date2)`	A dátumok közti hónapok száma
`ADD_MONTHS(date, n)`	n hónappal növeli a dátumot
`NEXT_DAY(date, 'char')`	A következő adott nevű nap dátuma.
`LAST_DAY(date)`	A dátum hónapjának utolsó napja.
`ROUND(date[,'fmt'])`	A dátum kerekítése
`TRUNC(date[, 'fmt'])`	A dátum levágása

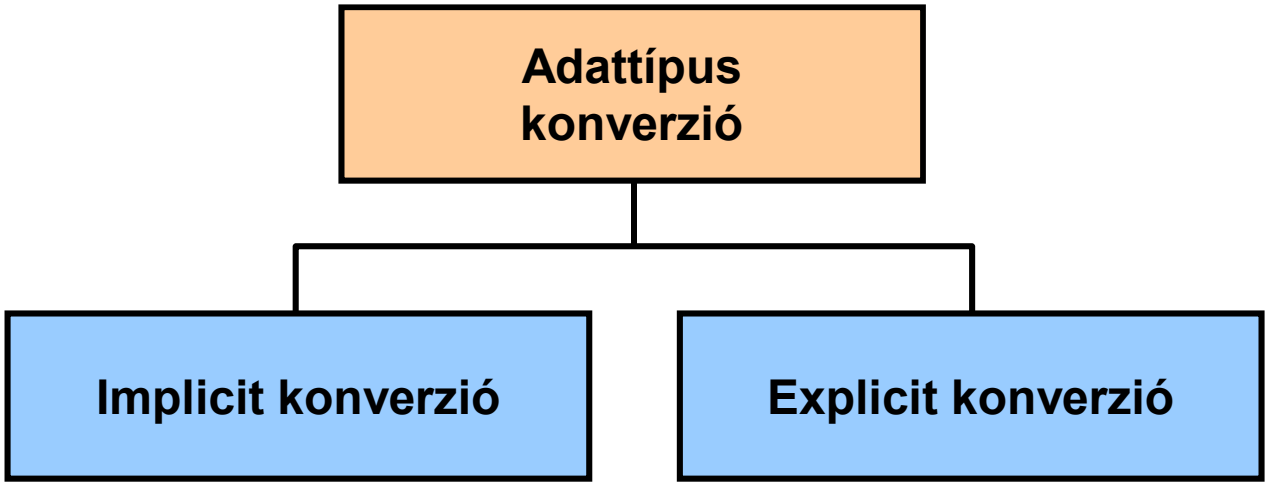
A dátumfüggvények

Függvény	Eredmény
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

```
SELECT employee_id, hire_date,
   MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
   ADD_MONTHS (hire_date, 6) REVIEW,
   NEXT_DAY (hire_date, 'FRIDAY'),
   LAST_DAY(hire_date)
FROM   employeesSYSDATE
WHERE  MONTHS_BETWEEN (, hire_date) < 70;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(	LAST_DAY(
107	07-FEB-99	31.6982407	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	22.4079182	16-MAY-00	19-NOV-99	30-NOV-99
149	29-JAN-00	19.9885633	29-JUL-00	04-FEB-00	31-JAN-00
178	24-MAY-99	28.1498536	24-NOV-99	28-MAY-99	31-MAY-99

Konvertáló függvények



A hasonló adattípusok konverzióját az Oracle szerverre is bízhatjuk (implicit), de ajánlott inkább konvertáló függvényeket használni (explicit).

Implicit adattípus-konverzió

- A következő típusok konverzióját az Oracle szerver automatikusan elvégzi:

Miről	Mire
VARCHAR2 vagy CHAR	NUMBER
VARCHAR2 vagy CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

```
select hire_date from hr.employees where hire_date > '1990-01-01'
```

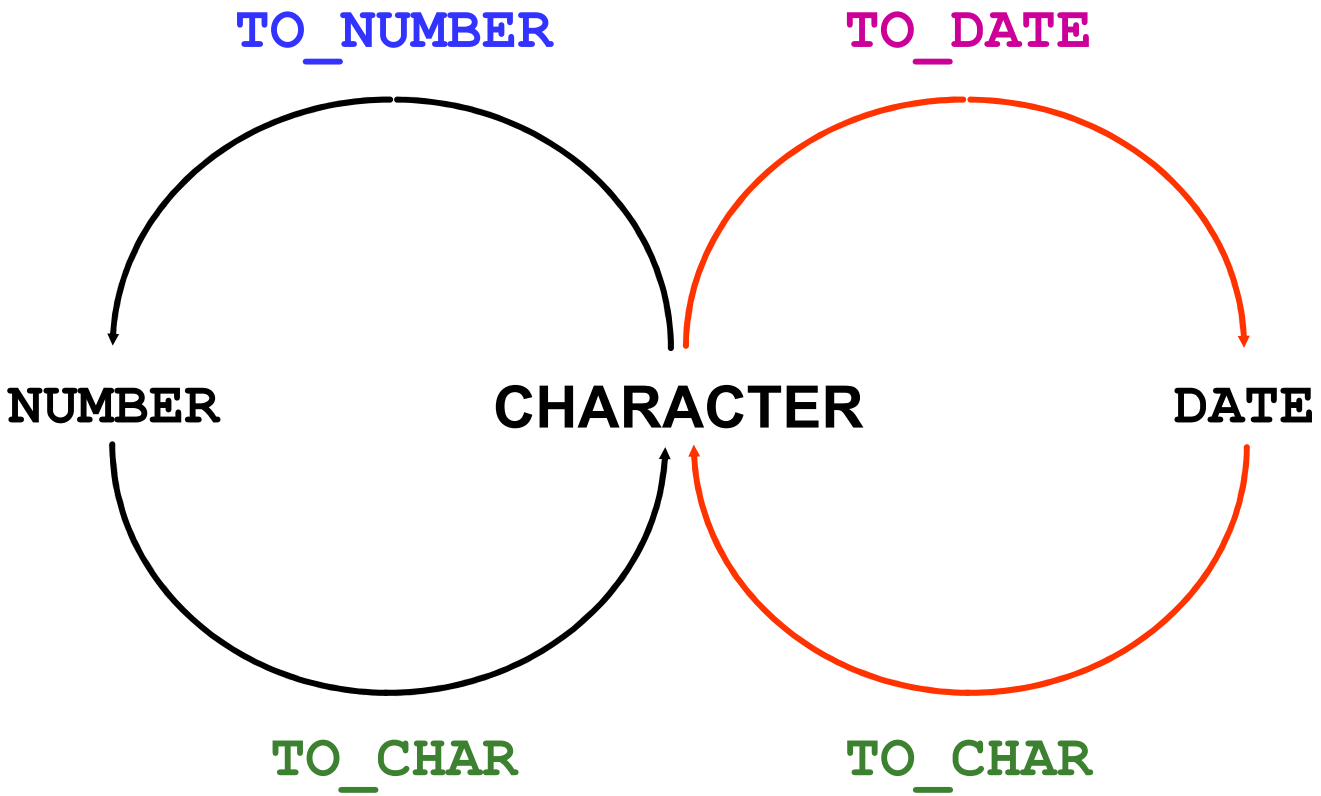
A jobb oldal karakteres, a bal oldal dátum, mégis érvényes az összehasonlítás.

Implicit adattípus-konverzió

- A következő típusú kifejezések konverzióját az Oracle szerver

Miről	Mire
`VARCHAR2 vagy CHAR`	`NUMBER`
`VARCHAR2 vagy CHAR`	`DATE`

Explicit adattípus-konverzió

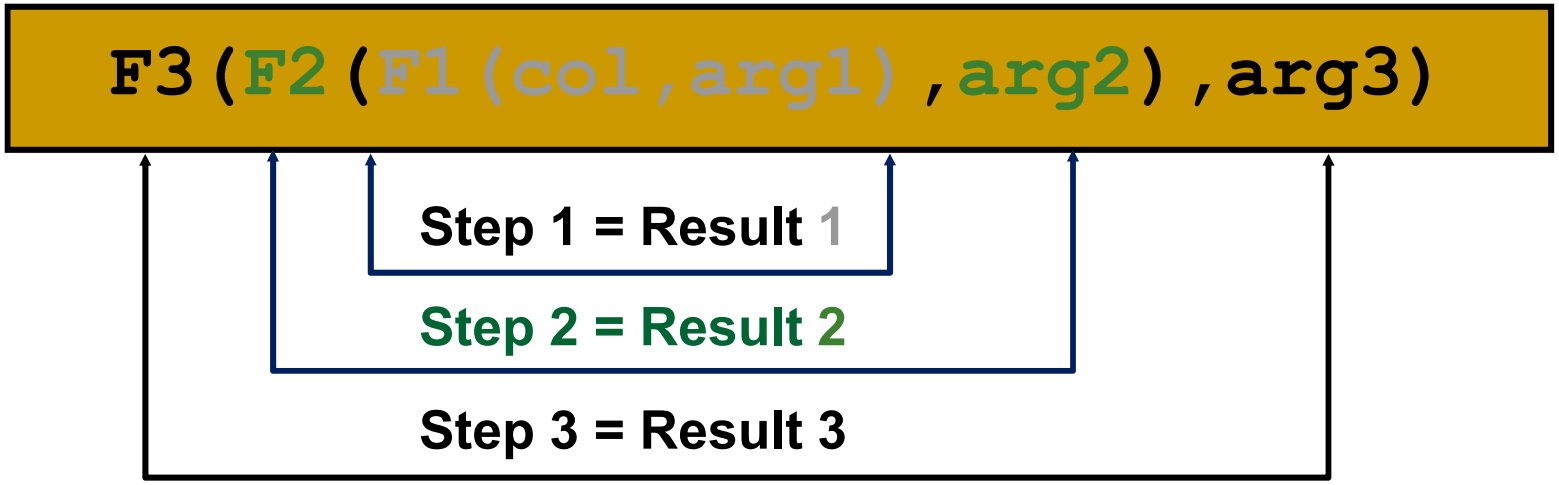


Explicit adattípus-konverzió

Függvény	Leírás
TO_CHAR(number|date,[fmt], [nlsparams])	A VARCHAR2 karakter formátumát az *fmt* modellel lehet megadni. Az nlsparams paraméter mondja meg, hogy milyen tízedesvesszőt, ezres csoportosítót, pénznemeket használunk.
TO_CHAR(number|date,[fmt], [nlsparams])	Dátumkonverzió esetén az nlsparams paraméter mondja meg, hogy milyen nyelven adtuk meg a napok, hónapok nevét, vagy miként rövidítettük a neveket.
TO_NUMBER(char,[fmt], [nlsparams])	Az fmt és nlsparams opcionális paraméterek értelme a fentiek szerint.
TO_DATE(char,[fmt],[nlsparams])	Az fmt és nlsparams opcionális paraméterek értelme a fentiek szerint.

Függvények egymásba ágyazása

- A sorfüggvények tetszőleges mélységig egymásba ágyazhatók.
- A kiértékelés belülről kifele történik.



Általános függvények

- Ezek a függvények tetszőleges adattípussal és nullértékek esetén is működnek.
  - `NVL (expr1, expr2)`
  - `NVL2 (expr1, expr2, expr3)`
  - `NULLIF (expr1, expr2)`
  - `COALESCE (expr1, expr2, ..., exprn)`

Az NVL függvény

- A nullértéket a megadott értékkel helyettesíti:
  - Az adattípus lehet dátum, karakter, szám.
  - Az argumentumok adattípusának egyezőknek kell lenniük:
    - `NVL(commission_pct,0)`
    - `NVL(hire_date,'01-JAN-97')`
    - `NVL(job_id,'No Job Yet')`

A COALESCE függvény használata

- A COALESCE függvény esetében - az NVL függvénnyel szemben - több helyettesítő értéket is megadhatunk.
- Ha az első kifejezés nem nullértéket ad vissza, akkor ez a függvény értéke, különben a COALESCE függvényt alkalmazza a maradék kifejezésekre.

ÚJ ANYAG: Összesítő függvények

- az összesítő függvény csoportosított sorok halmazain működik, és egyetlen eredményt ad vissza csoportonként.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

. . .

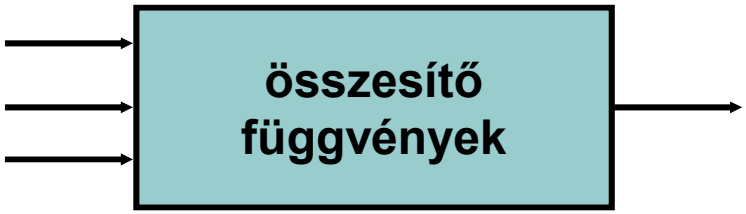
20 rows selected.

A legmagasabb fizetés az EMPLOYEES táblában

MAX(SALARY)
24000

Az aggregáló függvények típusai

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Az aggregáló függvények típusai (folyt.)

Függvény	Leírása
AVG([DISTINCT|ALL]*n*)	n átlagértéke (nullértékeket kihagyva)
COUNT({*|[DISTINCT|ALL]*expr*})	Azon sorok száma, amelyekre *expr* kiértékelése nem null (DE: * esetén az összes sorok száma, beleértve az ismétlődőket és a null értéket tartalmazókat is)
MAX([DISTINCT|ALL]*expr*)	*Expr* legnagyobb értéke (nullértékeket kihagyva)
MIN([DISTINCT|ALL]*expr*)	*Expr* legkisebb értéke (nullértékeket kihagyva)
STDDEV([DISTINCT|ALL]*x*)	*n* szórása (nullértékeket kihagyva)
SUM([DISTINCT|ALL]*n*)	*n* értékeinek összege (nullértékeket kihagyva)
VARIANCE([DISTINCT|ALL]*x*)	*n* szórásnégyzete (nullértékeket kihagyva)

Az aggregáló függvények használata

```
SELECT      [column,] oszlop_függvény(column), . . .
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```

Az oszlopfüggvények a nullértéket tartalmazó sorokat kihagyják, a nullérték helyettesítésére használhatók az NVL, NVL2 és COALESCE függvények.

AVG és SUM összesítő függvény

- Az AVG és SUM csak numerikus adatokra használható (a VARIANCE és STDDEV szintén).

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

MIN és MAX összesítő függvény

- A MIN és MAX numerikus, karakteres és dátum típusú adatokra használható (LOB és LONG típusokra nem).

```
SELECT MIN(hire_date), MAX(hire_date)
FROM       employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

COUNT összesítő függvény

- COUNT(*) visszaadja a sorok számát a

1

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

COUNT(*)
5

2

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(COMMISSION_PCT)
3

A DISTINCT kulcsszó használata

- COUNT(DISTINCT expr) azoknak a soroknak a számát adja vissza, amelyekben expr értéke különböző és nem nullérték
- Pl. a különböző (nem null) osztályazonosítók száma az EMPLOYEES táblában:

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

összesítő függvények és a nullértékek

- Az összesítő függvények általában ignorálják a nullértéket tartalmazó sorokat:

1

```
SELECT AVG(commission_pct)
FROM   employees;
```

AVG(COMMISSION_PCT)
.2125

2

```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```

AVG(NVL(COMMISSION_PCT,0))
.0425

Adatcsoportok létrehozása

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...

20 rows selected.

4400

9500

3500

6400

10033

Az EMPLOYEES tábla osztályai és azokon az átlagfizetések

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

Adatcsoportok létrehozása: a GROUP BY rész szintaxisa

```
SELECT     column, group_function(column)
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

- Egy tábla sorai csoportosíthatóak a GROUP BY rész használatával.
- A GROUP BY részben oszlop másodnevek nem szerepelhetnek.

A GROUP BY rész használata

- A SELECT lista minden olyan oszlopnevének, amely nem összesítő függvényekben fordul elő, szerepelnie kell a GROUP BY részben.

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

A GROUP BY rész használata

- A GROUP BY oszlopneveknek nem kötelező szerepelni a SELECT listában.

```
SELECT   AVG(salary)
FROM     employees
GROUP BY department_id ;
```

AVG(SALARY)
4400
9500
3500
6400
10033.3333
19333.3333
10150
7000

összesítő függvény szerepelhet az ORDER BY részben is.

Csoportosítás több oszlopnév alapján

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600

...

20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Az EMPLOYEES tábla osztályain az egyes beosztások átlagfizetései

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

A GROUP BY rész használata több oszlopnév esetén

```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY department_id, job_id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

összesítő függvényt tartalmazó szabálytalan lekérdezések

- Bármely oszlopnévnek vagy kifejezésnek a SELECT listában, ha az nem aggregáló

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Az oszlopnév nem szerepel a `GROUP BY` részben!

összesítő függvényt tartalmazó szabálytalan lekérdezések

- A csoportok korlátozására a WHERE feltétel nem használható.
- A HAVING rész szolgál a csoportok korlátozására.

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE  AVG(salary) > 8000
       *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

A csoportok korlátozására a `WHERE` feltétel nem használható!

Csoportok korlátozása

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	
20	6000
110	12000
110	8300

20 rows selected.

A legmagasabb fizetés osztályonként, ha az nagyobb mint $10,000

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

HAVING

- A HAVING rész használata esetén az Oracle szerver az alábbiak szerint korlátozza a csoportokat:
  1. Csoportosítja a sorokat.
  2. Alkalmazza Az összesítő függvényeket a csoportokra.
  3. A HAVING résznek megfelelő csoportokat megjeleníti.

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

A HAVING rész használata

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY department_id
HAVING    MAX(salary)>10000 ;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

A HAVING rész használata

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING    SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

összesítő függvények egymásba ágyazása

- Az osztályonkénti legmagasabb átlagfizetés megjelenítése:

```
SELECT    MAX(AVG(salary))
FROM      employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

Az összesítő függvények csak kétszeres mélységig ágyazhatóak egymásba!

Oracle GYAK: Összefoglalás

- Ebben a részben megtanultuk:
  - a COUNT, MAX, MIN, és AVG összesítő függvények használatát,
  - hogyan írjunk GROUP BY részt tartalmazó lekérdezéseket,
  - hogyan írjunk HAVING részt tartalmazó lekérdezéseket.

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

Új anyag: Relációs algebra kibővítése

- Az eddig tanult műveleteket: vetítés (Π), kiválasztás (σ), halmazműveletek: unió ($\cup$), különbség ($-$), metszet ($\cap$), szorzás: természetes összekapcsolás($\bowtie$), direkt-szorzat (x), stb. multihalmazok fölött értelmezzük, mint az SQL-ben, egy reláció nem sorok halmazából, hanem multihalmazából áll, vagyis megengedett a sorok ismétlődése.
- Ezeken kívül a SELECT kiegészítéseinek és záradékainak megfeleltetett új műveletekkel is kibővítjük a rel. algebrát:
  - Ismétlődések megszüntetése (δ) - select distinct ..
  - Vetítési művelet kiterjesztése (Π_{lista}) - select kif [as onev]..
  - Rendezési művelet (τ_{lista}) - order by..
  - Összesítő műveletek és csoportosítás (γ_{lista}) - group by..
  - Külső összekapcsolások ($\bowtie^{o}$)- [left | right | full] outer join

Multihalmazok egyesítése, különbsége

- Unió: R ∪ S-ben egy t sor annyiszor fordul elő ahányszor előfordul R-ben, plusz ahányszor előfordul S-ben: n+m
- Metszet: R ∩ 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 U S) – T =? (R – S) U (S – T) (Ez hz: igen, multihz:nem)

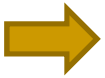
R

A	B
1	3
1	2

∪

S

A	B
1	3
2	5



A	B
1	3
1	2
1	3
2	5

A „többi művelet” multihalmazok fölött

- A projekció, szelekció, Descartes-szorzat, természetes összekapcsolás és Théta-összekapcsolás végrehajtása során nem küszöböljük ki az ismétlődéseket.

R

A	B
1	2
1	5
2	3

$\Pi_A(R)$

A
1
1
2

Új műveletek: Ismétlődések megszüntetése (duplikátumok kiszűrése)

- Ismétlődések megszüntetése: R1:= δ(R2)
- A művelet jelentése: R2 multihalmazból R1 halmazt állít elő, vagyis az R2-ben egyszer vagy többször előforduló sorok csak egyszer szerepelnek az R1-ben.
- A DISTINCT reprezentálására szolgál (jele: δ kis-delta)
- A δ speciális esete lesz az általánosabb γ műveletnek

R =

A	B
1	2
3	4
1	2

δ(R) =

A	B
1	2
3	4

Kiválasztott sorok rendezése

- Rendezés: $\tau_{A_1, \ldots, A_n}(R)$.
- Először A_1 attribútum szerint rendezzük R sorait. Majd azokat a sorokat, amelyek értéke megegyezik az A_1 attribútumon, A_2 szerint, és így tovább.
- Az ORDER BY reprezentálására szolgál (jele: τ tau)
- Ez az egyetlen olyan művelet, amelynek az eredménye nem halmaz és nem multihalmaz, hanem rendezett lista.

R =

(A	B)
1	2
3	4
5	2

$\tau_B (R) = [(5,2), (1,2), (3,4)]$

Összesítő (aggregáló) függvények

- Miért hívják aggregáló függvényeknek? Ha kiszámoltuk az összeget a tábla bizonyos soraira, akkor újabb sorok figyelembe vételével felhasználhatjuk az eddigi összeget.

R =

A	B
1	3
3	4
3	2

SUM(A) = 7
COUNT(A) = 3
MIN(B) = 2
MAX(B) = 4
AVG(B) = 3

Csoportosítás és összesítés $\gamma_L(R)$

- A csoportosítást (GROUP BY), a csoportokon végezhető összesítő függvényeket (AVG, SUM, COUNT, MIN, MAX, stb…) reprezentálja.
- A művelet jele: γ_L gamma
- Itt az **L** lista valamennyi eleme a következők egyike:
  - R olyan attribútuma, amely szerepel a GROUP BY záradékban, egyike a csoportosító attribútumoknak.
  - R egyik attribútumára (ez az összesítő attribútum) alkalmazott összesítő operátor.
    - Ha az összesítés eredményére névvel szeretnénk hivatkozni, akkor nyilat és új nevet használunk.

Csoportosítás és összesítés $\gamma_L(R)$

- Osszuk *R* sorait csoportokba. Egy csoport azokat a sorokat tartalmazza, amelyek az *L* listán szereplő csoportosítási attribútumokhoz tartozó értékei megegyeznek
  - Vagyis ezen attribútumok minden egyes különböző értéke egy csoportot alkot.
- Minden egyes csoporthoz számoljuk ki az *L* lista összesítési attribútumaira vonatkozó összesítéseket
- Az eredmény minden egyes csoportra egy sor:
  - Eredmény: a csoportosítási attribútumok és
  - az összesítési attribútumra vonatkozó összesítések (az adott csoport összes sorára)

Példa: Csoportosításra és összesítésre

R =

A	B	C
1	2	3
4	5	6
1	2	5

$\gamma_{A,B,AVG(C)->X}(R) = ??$

Először csoportosítunk

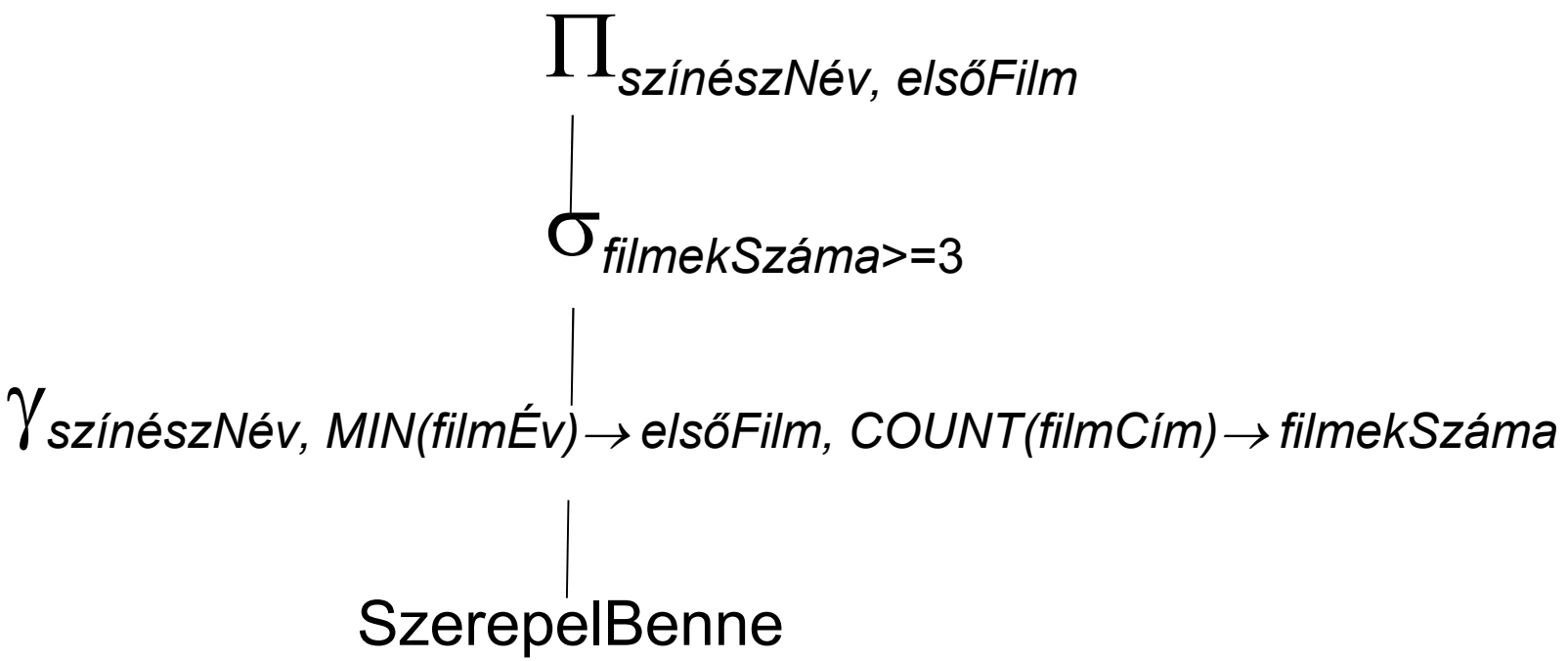
A	B	C
1	2	3
1	2	5
4	5	6

majd csoportonként összesítünk:

A	B	X
1	2	4
4	5	6

Tankönyv példája

- Adjuk meg azokat a színészeket, akik már szerepeltek legalább három filmben illetve adjuk meg az első olyan filmet is, amiben szerepelt:



Külső összekapcsolások

- Ez nem relációs algebrai művelet, uis kilép a modellből.
- Lehet baloldali, jobboldali, teljes külső összekapcsolás.
- R, S sémái $R(A_1,\ldots,A_n,B_1,\ldots,B_k)$, ill. $S(B_1,\ldots,B_k,C_1,\ldots,C_m)$
- $R \overset{o}{\bowtie} S = R \bowtie S$ relációt kiegészítjük az R és S soraival, a hiányzó helyekre NULL értéket írva megőrzi a „lógó sorokat”
- Van teljes, baloldali és jobboldali külső összekapcsolás attól függően, hogy melyik oldalon szereplő reláció sorait adjuk hozzá az eredményhez (a lógó sorokat kiegészítve NULL értékkel) $\perp$ szimbólummal.

Példák külső összekapcsolásokra

A	B	C
1	2	3
4	5	6
7	8	9

R reláció

B	C	D
2	3	10
2	3	11
6	7	12

S reláció

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	⊥
7	8	9	⊥
⊥	6	7	12

$R \overset{o}{\bowtie} S$ eredmény

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	⊥
7	8	9	⊥

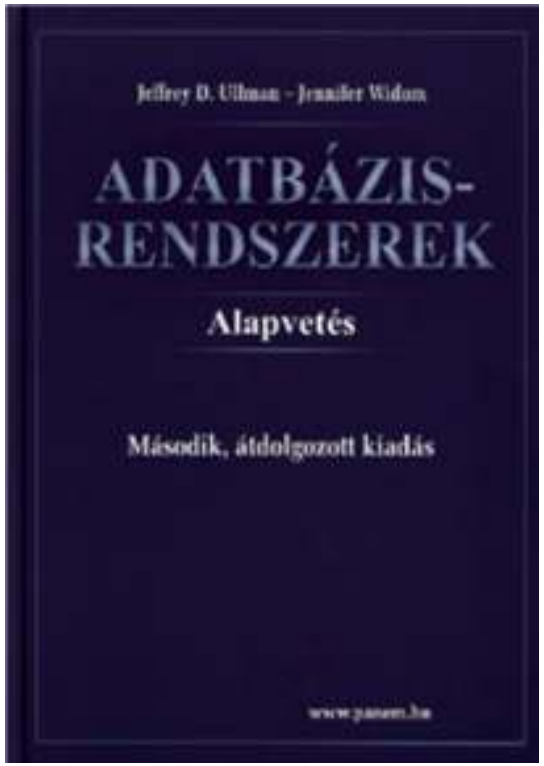
$R \overset{o}{\bowtie}_{L} S$ eredmény

A	B	C	D
1	2	3	10
1	2	3	11
⊥	6	7	12

$R \overset{o}{\bowtie}_{R} S$ eredmény

SQL lekérdezések

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



6.4. Csoportosítás és összesítések
az SQL-ben, group by, having

Nézzük meg SQL SELECT utasítással! Emlékeztető az előadás példa: Sörivók

- Az előadások SQL lekérdezései az alábbi Sörivók adatbázissémán alapulnak

  (aláhúzás jelöli a kulcs attribútumokat)

  Sörök(<u>név</u>, gyártó)
  Sörözők(<u>név</u>, város, tulaj, engedély)
  Sörivók(<u>név</u>, város, tel)

  Szeret(<u>név</u>, <u>sör</u>)
  Felszolgál(<u>söröző</u>, <u>sör</u>, ár)
  Látogat(<u>név</u>, <u>söröző</u>)

SQL: Ismétlődések megszüntetése

- SELECT DISTINCT … FROM …
- A δ művelet SQL-beli megfelelője, amellyel az eredményben kiszűrjük a duplikátumokat, vagyis multihalmazból halmazt állítunk elő.

SQL: Ismétlődések kezelése halmazművelet során

- (SELECT … FROM …)

  {UNION | INTERSECT | EXCEPT} [ALL]

  (SELECT … FROM …)

- Alapértelmezésben a halmaz-szemantika (duplikátumok szűrése)
- Az ALL kulcsszóval ezek a műveletek multihalmaz-szemantika szerint működnek.

SQL: Összesítések (aggregálás)

- SELECT listán:

  <Aggregáló művelet>(kifejezés) [[AS] onév], …

  SUM, COUNT, MIN, MAX aggregáló műveleteket, AVG (bevezették ezt is, mivel gyakran kell AVG) a SELECT záradékban alkalmazhatjuk egy oszlopra.

- COUNT(*) az eredmény sorainak számát adja meg.
- Itt is fontos a halmaz, multihalmaz megkülönböztetés.

  pl. SUM(DISTINCT R.A) csak a különböző értékűeket veszi figyelembe.
- NULL értékek használata, pl. SUM nem veszi figyelembe (implementáció függő, ellenőrizzük le a COUNT-ra - gyak.)

Példa: Összesítő függvények

- A Felszolgál(bár, sör, ár) tábla segítségével adjuk meg a Bud átlagos árát:

```
SELECT AVG(ár)
FROM Felszolgál
WHERE sör = 'Bud';
```

Ismétlődések kiküszöbölése összesítésben

- Az összesítő függvényen belül DISTINCT.
- Példa: hány *különféle* áron árulják a Bud sört?

```
SELECT COUNT(DISTINCT ár)
FROM Felszolgál
WHERE sör = 'Bud';
```

NULL értékek nem számítanak az összesítésben

- NULL nem számít a SUM, AVG, COUNT, MIN, MAX függvények kiértékelésekor.
- De ha nincs NULL értéktől különböző érték az oszlopban, akkor az összesítés eredménye NULL.
  - Kivétel: COUNT az üres halmazon 0-t ad vissza.

Példa: NULL értékek összesítésben

```
SELECT count(*)
FROM Felszolgál
WHERE sör = 'Bud';
```

A Bud sört árusító kocsmák száma.

```
SELECT count(ár)
FROM Felszolgál
WHERE sör = 'Bud';
```

A Bud sört ismert áron árusító kocsmák száma.

Csoportosítás

- SELECT …
  FROM …
  [WHERE …]
  [GROUP BY kif_1, … kif_k]
- Egy SELECT-FROM-WHERE kifejezést GROUP BY záradékkal folytathatunk, melyet attribútumok listája követ.
- A SELECT-FROM-WHERE eredménye a megadott attribútumok értékei szerint csoportosítódik, az összesítéseket ekkor minden csoportra külön alkalmazzuk.

Példa: Csoportosítás

- A Felszolgál(bár, sör, ár) tábla segítségével adjuk meg a sörök átlagos árát.

```
SELECT sör, AVG(ár)
FROM Felszolgál
GROUP BY sör;
```

sör	AVG(ár)
Bud	2.33
Miller	2.45

A SELECT lista és az összesítések

- Ha összesítés is szerepel a lekérdezésben, a SELECT-ben felsorolt attribútumok
  1. vagy egy összesítő függvény paramétereként szerepelnek,
  2. vagy a GROUP BY attribútumlistájában is megjelennek.

Csoportok szűrése: HAVING záradék

- 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:
  1. vagy csoportosító attribútumok,
  2. vagy összesített attribútomok.

  (vagyis ugyanazok a szabályok érvényesek, mint a SELECT záradéknál).

Egy másik példa csoportosításra

Példa: hallgató (azon, név, város, tantárgy, jegy)

```
SELECT név, AVG(jegy) AS átlag
FROM hallgató
WHERE város = 'Bp'
GROUP BY azon, név
HAVING COUNT(tantárgy) > 2;
```

$\Pi_{\text{név, átlag}}$

|

$\sigma_{\text{db} > 2}$

|

$\gamma_{\text{azon, név, AVG(jegy)}\rightarrow\text{átlag,COUNT(tantárgy)}\rightarrow\text{db}}$

|

$\sigma_{\text{város = 'Bp'}}$

|

hallgató

(Megjegyzés: a relációs algebra kibővítése a csoportosításra is)

ORDER BY: Az eredmény rendezése

- SQL SELECT utasítás utolsó záradéka
- 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

Összefoglalva: záradékok

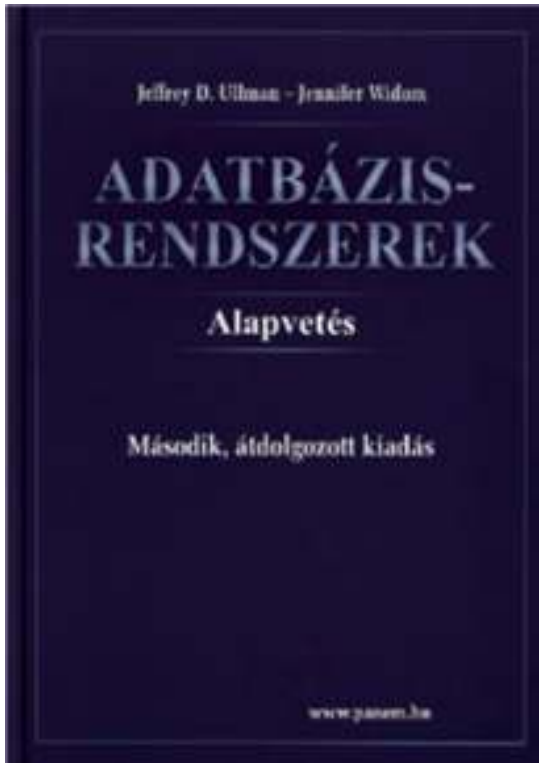
- Teljes SELECT utasítás

  (záradékok sorrendje nem cserélhető fel)

```
SELECT  [DISTINCT] …
FROM …
   [WHERE   … ]
   [GROUP BY …
      [HAVING … ] ]
   [ORDER BY …]
```

SQL lekérdezések

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



6.2. Több relációra vonatkozó
lekérdezések az SQL-ben

Select-From-Where (SFW) utasítás

- Gyakran előforduló relációs algebrai kifejezés

  Π_{Lista} (σ_{Felt} (R_1 x… x 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 tovább 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) Ugyanaz a reláció többször is szerepelhet, vagyis szükség lehet arra, hogy ugyanaz a relációnév többször is előforduljon a FROM listában.
- Ekkor a FROM listában a táblához másodnevet kell megadni, erre sorváltozóként is szoktak hivatkozni, megadjuk, h. melyik sorváltozó melyik relációt képviseli:

  FROM R_1 [t_1], …, R_n [t_n]

  Ekkor a SELECT és WHERE záradékok kifejezésekben a hivatkozás: **t_i.A** (vagyis sorváltozó.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 *tn* sorra az *Rn* relációban DO

IF a where záradék igaz, amikor az attribútumokban *t*1, *t*2, ..., *tn* megfelelő értékei találhatóak

THEN

*t*1, *t*2, ..., *tn* -nek megfelelően kiértékeljük a select záradék attribútumait

és az értékekből alkotott sort

az eredményhez adjuk

SFW szabvány alapértelmezése ---3

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], …, kif_n [[AS] $onév_n$]

FROM R_1 [t_1], …, R_n [t_n]

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.

Megj.: konverzió relációs algebrába

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], …, kif_n [[AS] $onév_n$]

FROM R_1 [t_1], …, 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 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.

$$\Pi_{onév1,\ldots,\ onévn}\ (\ \sigma_{feltétel}\ (R_1 \times \ldots \times R_n\ ))$$

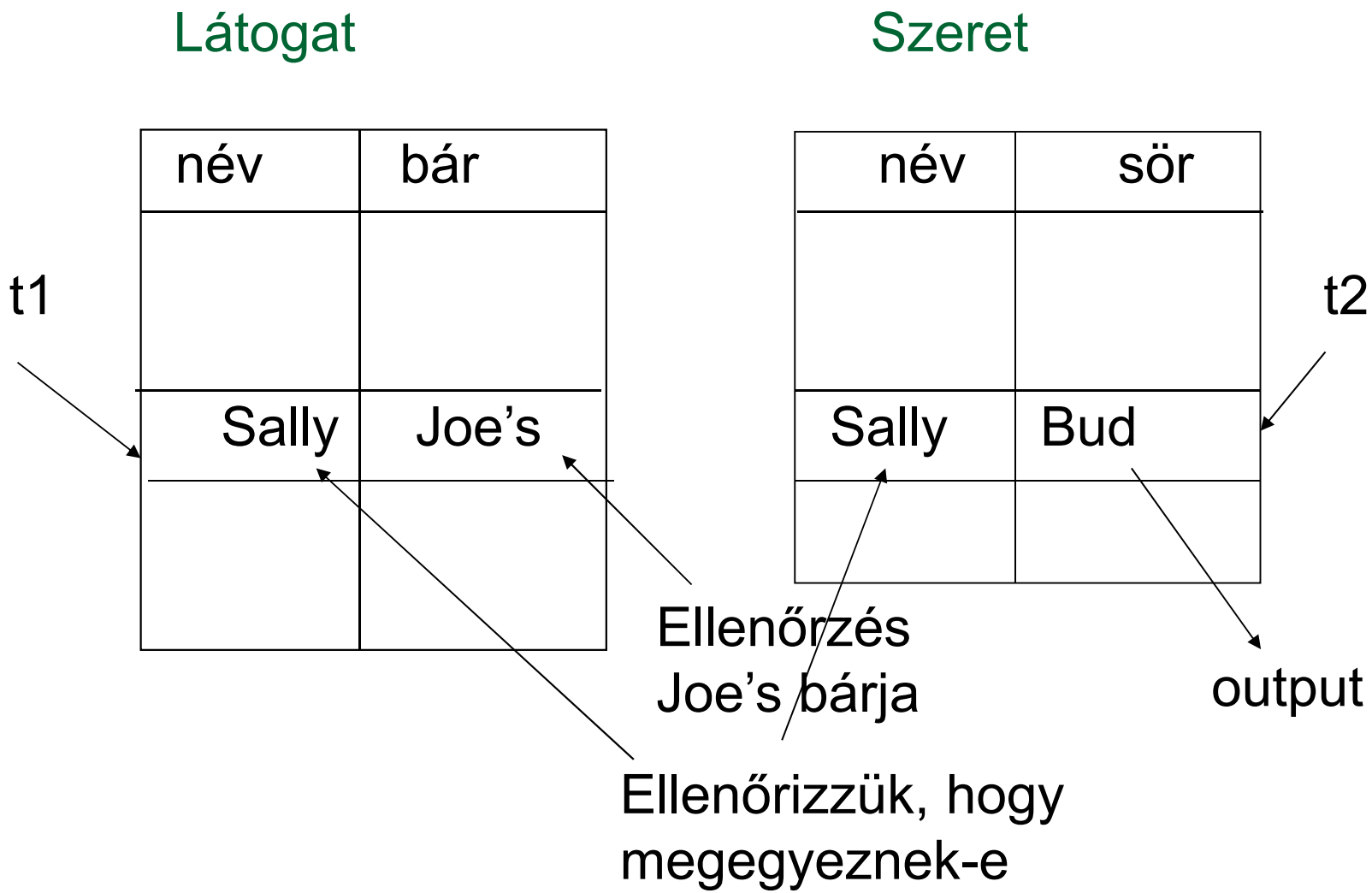
Példa: Két tábla összekapcsolása ---1

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

```
SELECT sör
   FROM Szeret, Látogat
   WHERE bár = 'Joe''s Bar'
      AND Látogat.név = Szeret.név;
```

- Kiválasztási feltétel: `bár = 'Joe''s Bar'`
- Összekapcsolási feltétel: `Látogat.név = Szeret.név`
- Alapértelezését lásd a következő oldalon
- Összekapcsolások SQL:1999-es szintaxisát is nézzük majd

Példa: Két tábla összekapcsolása ---2



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 az SQL-ben

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

  (lekérdezés1)
  [**UNION** | **INTERSECT** | {**EXCEPT** | **MINUS**}]
  (lekérdezés2);

Példa: Intersect (metszet)

- Szeret(név, sör), Felszolgál(bár, sör, ár) és Látogat(név, bár) táblák felhasználásával keressük azokat a sörivókat és söröket, amelyekre a sörivó szereti az adott sört **és** a sörivó látogat olyan bárt, ahol felszolgálják a sört.

Trükk: itt ez az alkérdés valójában az adatbázisban tárolt tábla

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

```
(SELECT * FROM Szeret)
  INTERSECT
(SELECT név, sör
 FROM Felszolgál, Látogat
 WHERE Látogat.bár = Felszolgál.bár);
```

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 Szeret(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 Szeret);
```

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- Gyakorlás az 1EA-hoz: Példák táblák létrehozására, kulcs- és hivatkozási épség megadására, adattípusok.
- Gyakorlás a 2EA-hoz: Egy táblára vonatkozó (EMP tábla) lekérdezésekre példák, vetítés és kiválasztás művelete, oszlopnevek helyett kifejezések, függvények használata
- Házi feladat: Oracle Példatár 1.fejezet feladatai:
- Gyakorlás a 3EA-hoz: Egy táblára vonatkozó (EMP tábla) lekérdezésekre példák, csoportosítás és összesítés.
- Házi feladat: Oracle Példatár 2.fejezet feladatai:

  http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf