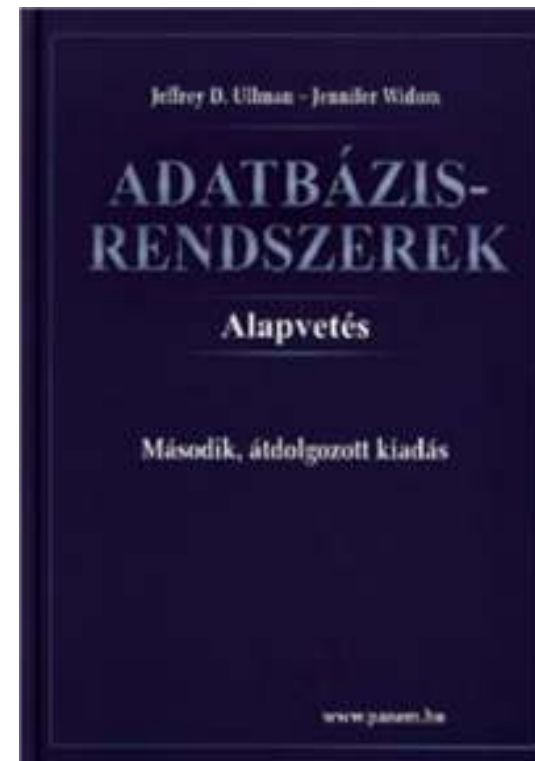


# Kiterjesztett relációs algebra

## SQL: csoportosítás és összesítések

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



---

[Oracle gyak.] gépes lekérdezések:  
SQL függvények használata

5.1.- 5.2. Kiterjesztett műveletek

6.4.1. Ismétlődések kezelése

6.4.2. Halmazműveletek SQL-ben

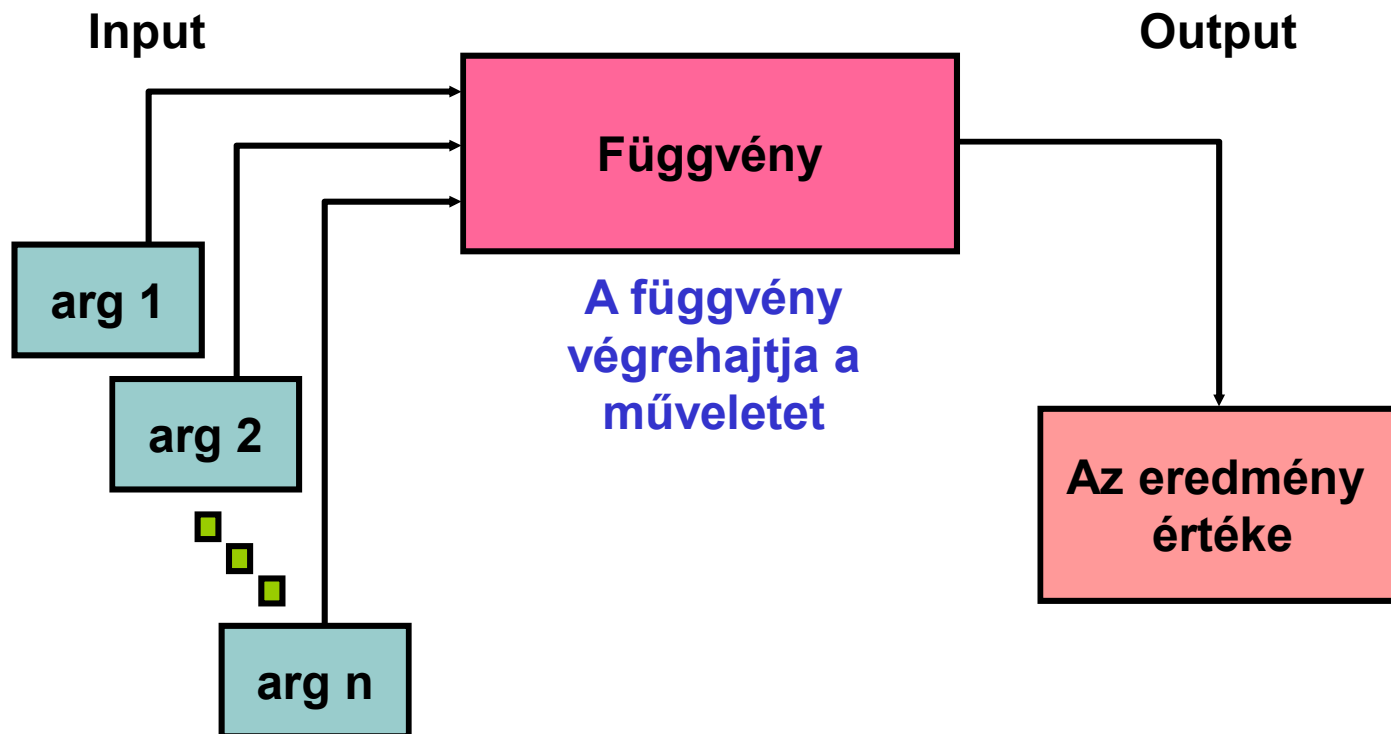
6.4.3.-6.4.8. Csoportosítás és összesítések az SQL-ben

[Oracle gyak.] 6.3.6.-6.3.9. Összekapcsolások az SQL-ben

# Egytáblás lekérdezések az SQL-ben

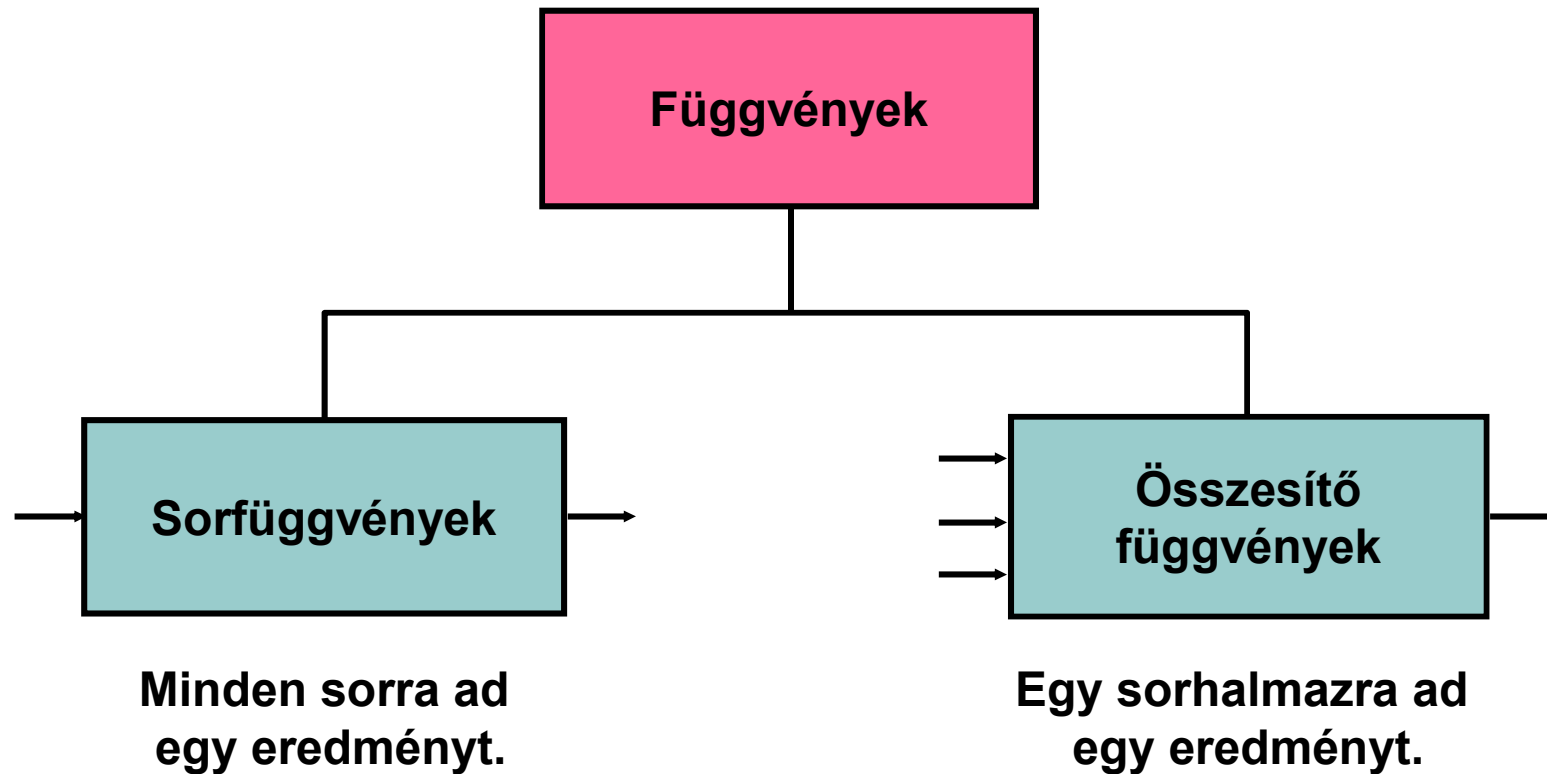
- Az SQL-ben halmazok helyett **multihalmazokat** használunk (egy sor többször is előfordulhat)
- $\Pi_{\text{Lista}} \sigma_{\text{Feltétel}}(\mathbf{R})$  kifejezésben a vetítés L-listája és a kiválasztás F-feltétele az attribútumnevek helyén **kifejezések** állnak, amely függvényeket és műveleti jeleket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

# Oracle GYAK: SQL-függvények

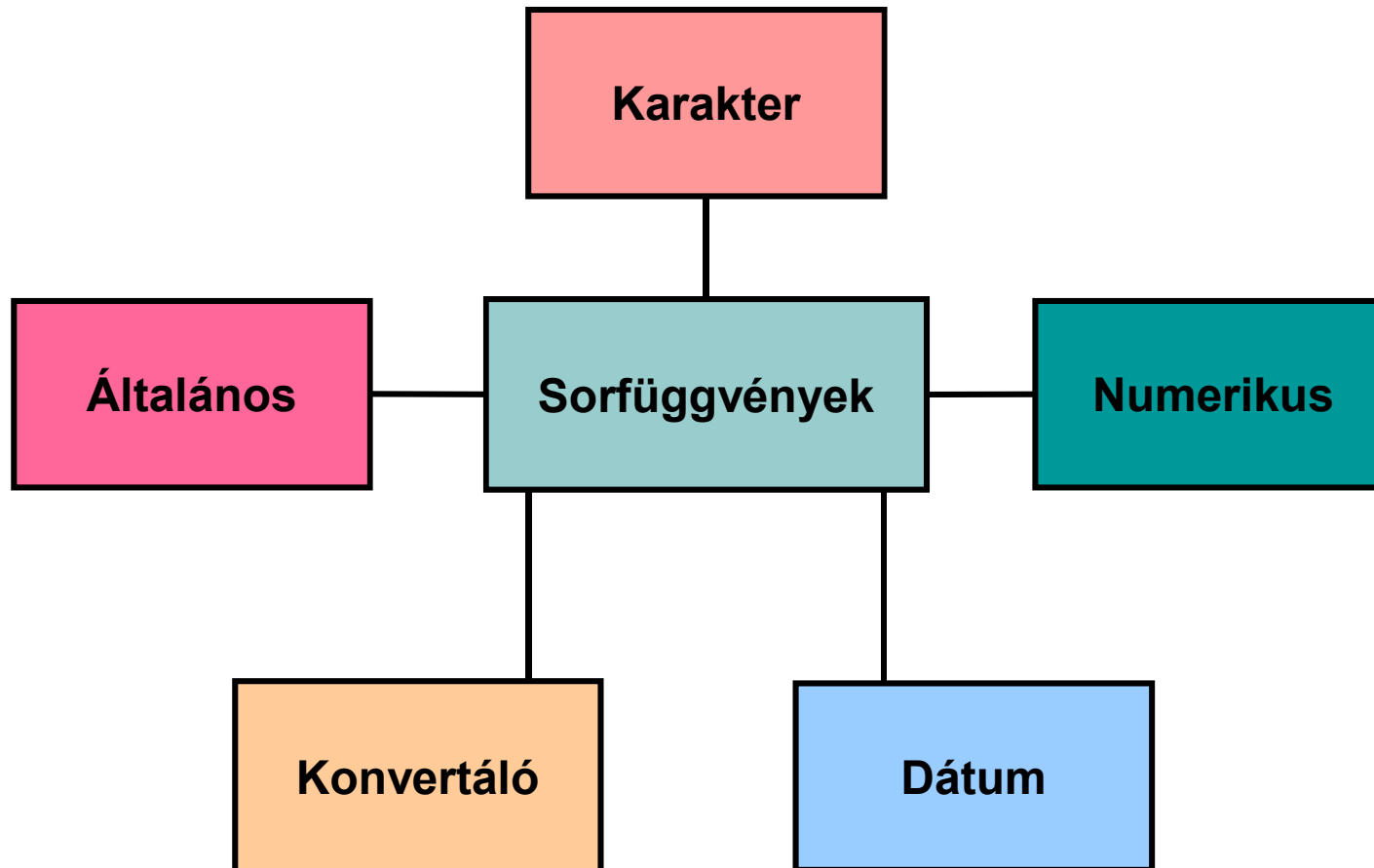


**A gyakorlaton bemutatott függvények többsége Oracle-specifikus.**

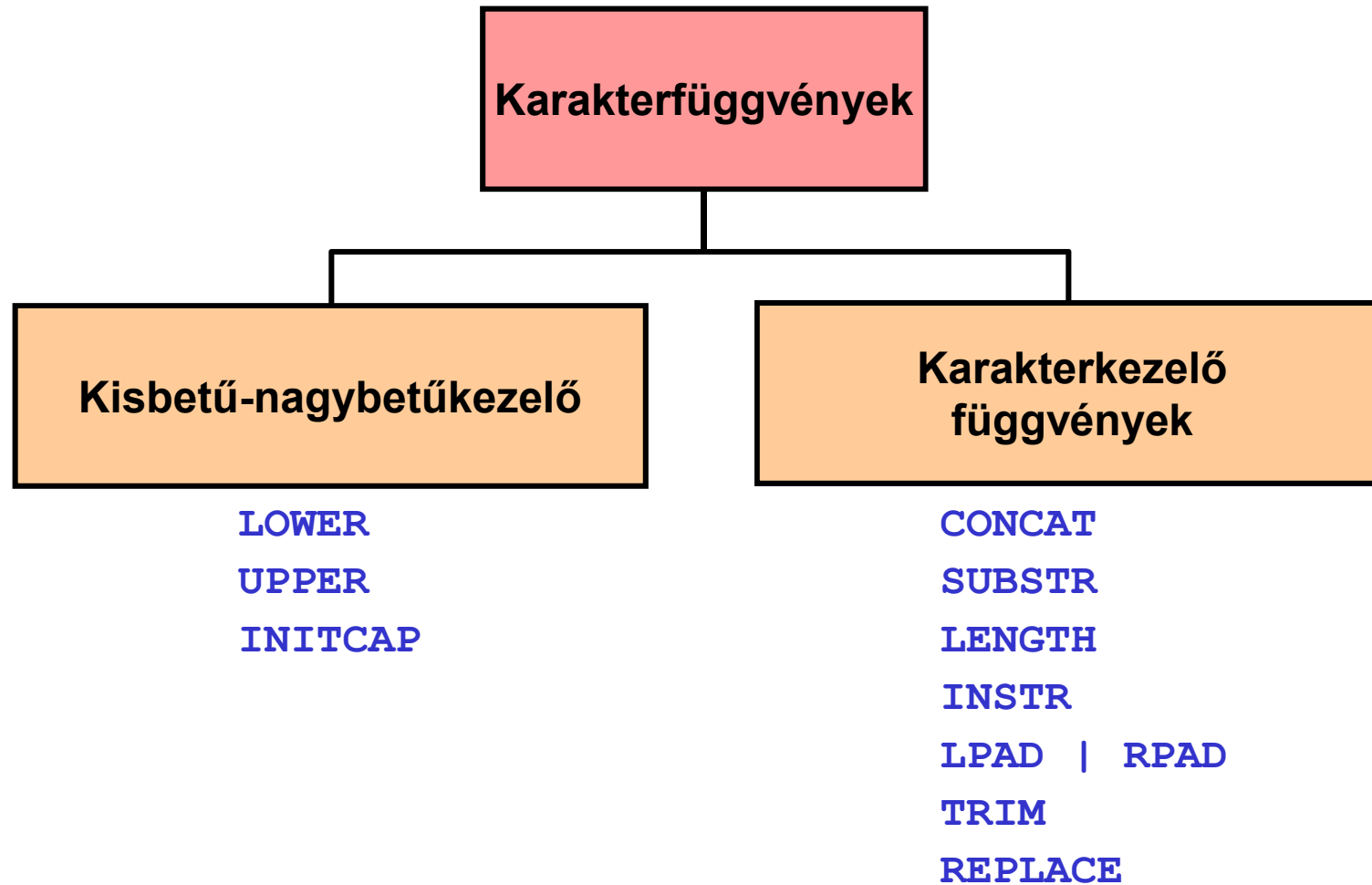
# Az SQL-függvények két típusa



# Sorfüggvények



# Karakterfüggvények



# Dátumfüggvények

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

```

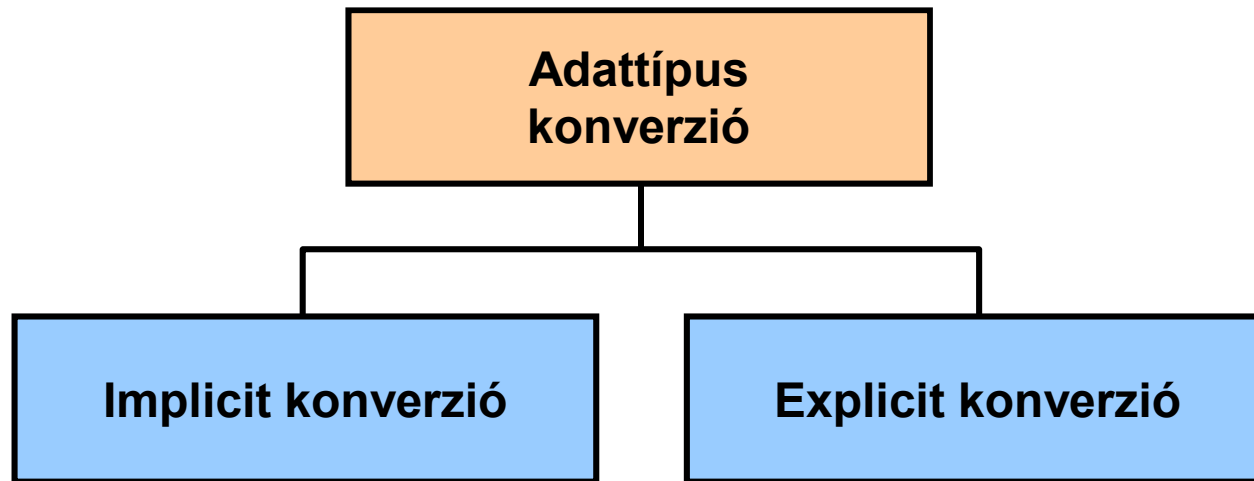
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'),
       LAST_DAY(hire_date)
FROM employees
WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 70;

```

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

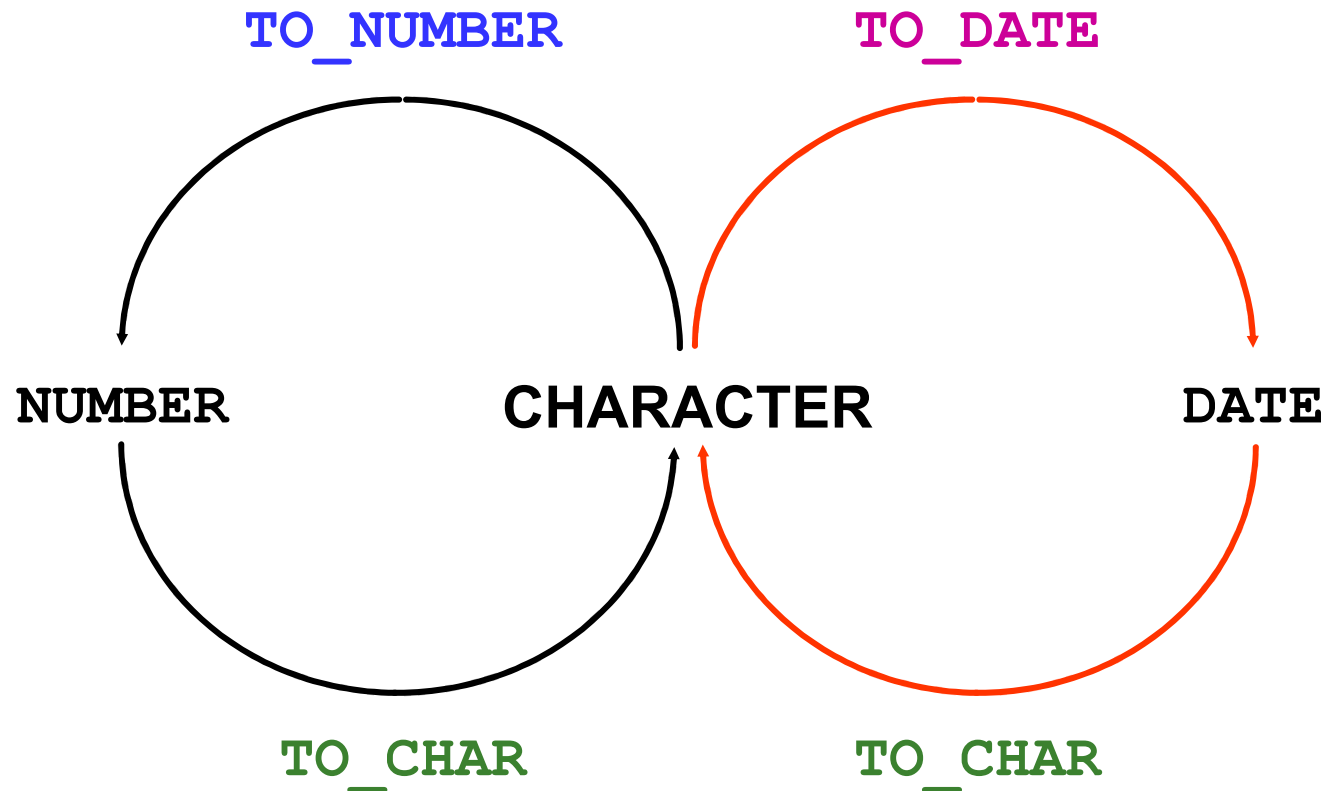


# Konvertáló függvények



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

# Explicit adattípus-konverzió

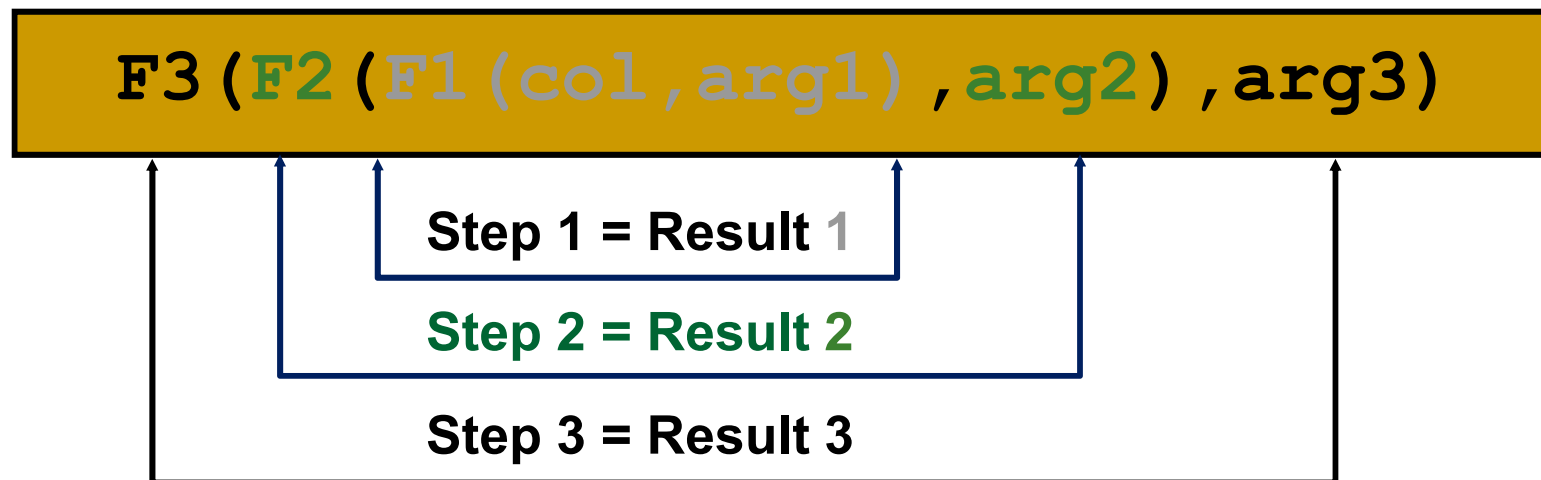


# Explicit adattípus-konverzió

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

# Függvények egymásba ágyazása

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



# Az NVL függvény

A nullértéket a megadott értékkel helyettesíti:

- Az adattípus lehet dátum, karakter, szám.
- Az argumentumok adattípusának egyezőnek kell lenniük:

```
NVL(fizetés, 0)
```

```
NVL(dátum, to_date('1997.01.07', 'YYYY.MM.DD'))
```

```
NVL(foglalkozás, 'Tanár')
```

# 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.

# Példa: COALESCE függvény

```
SELECT product_id, list_price, min_price,  
       COALESCE(0.9*list_price, min_price, 5) "Sale"  
FROM oe.product_information  
WHERE supplier_id = 102050  
ORDER BY product_id;
```

<u>PRODUCT ID</u>	<u>LIST PRICE</u>	<u>MIN PRICE</u>	<u>Sale</u>
1769	48		43.2
1770		73	73
2378	305	247	274.5
2382	850	731	765
3355			5

# Oracle GYAK: Összesítő függvények

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

**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

vissza csoportonként.

**A legmagasabb  
fizetés az  
EMPLOYEES  
táblában**

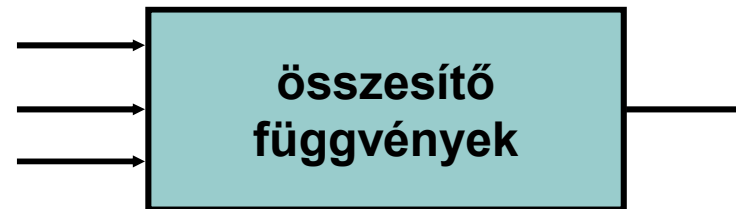
MAX(SALARY)
24000

■■■  
20 rows selected.



# Az aggregáló függvények típusai

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



# Az aggregáló függvények típusai

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

# Az aggregáló függvények használata

```
SELECT      [column,] oszlop_fuggvény(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 táblában:

1

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

COUNT(*)
5

- COUNT(expr) azoknak a soroknak a számát adja vissza, amelyekben expr nem nullérték:

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

- Az NVL függvénnyel kikényszeríthető a nullértéket tartalmazó sorok figyelembe vétele:

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

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

...

20 rows selected.

# 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  
átlagfizetési

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
60	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.

# 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, SUM 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];
```

# Relációs algebra kiterjesztése

## SQL: csoportosítás, összesítések, külső összekapcsolások

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

---

- 5.1. Relációs műveletek  
multihalmazokon
- 5.2. Kiterjesztett műveletek a  
relációs algebrában



# Új anyag: Relációs algebra kibővítése

- Az eddig tanult műveleteket: **vetítés** ( $\Pi$ ), **kiválasztás** ( $\sigma$ ), **halmazműveletek: unió** ( $\cup$ ), **különbség** ( $-$ ), **metszet** ( $\cap$ ), **szorzás: természetes összekapcsolás** ( $\bowtie$ ), **direkt-szorzat** ( $\times$ ), 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** megfelelően **új műveletekkel is kibővítjük** a rel. algebrát:
  - **Ismétlődések megszüntetése** ( $\delta$ ) - **select distinct ..**
  - **Összesítő műveletek és csoportosítás** ( $\gamma_{\text{lista}}$ ) - **group by..**
  - **Vetítési művelet kiterjesztése** ( $\Pi_{\text{lista}}$ ) - **select kif [as onev]..**
  - **Rendezési művelet** ( $\tau_{\text{lista}}$ ) - **order by..**
  - **Külső összekapcsolások** ( $\overset{0}{\bowtie}$ ) - **[left | right | full] outer join**

# Multihalmazok egyesítése, különbsége

- **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		<table><thead><tr><th>A</th><th>B</th></tr></thead><tbody><tr><td>1</td><td>3</td></tr><tr><td>1</td><td>2</td></tr></tbody></table>	A	B	1	3	1	2																
A	B																									
1	3																									
1	2																									
<table><thead><tr><th>A</th><th>B</th></tr></thead><tbody><tr><td>1</td><td>3</td></tr><tr><td>1</td><td>2</td></tr></tbody></table>	A	B	1	3	1	2	$\cup$	<table><thead><tr><th>A</th><th>B</th></tr></thead><tbody><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>5</td></tr></tbody></table>	A	B	1	3	2	5	$\rightarrow$	<table><thead><tr><th>A</th><th>B</th></tr></thead><tbody><tr><td>1</td><td>3</td></tr><tr><td>1</td><td>2</td></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>5</td></tr></tbody></table>	A	B	1	3	1	2	1	3	2	5
A	B																									
1	3																									
1	2																									
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			$\Pi_A(R)$
A	B		A
1	2	➔	1
1	5		1
2	3		2



# Új műveletek: Ismétlődések megszüntetése (duplikátumok kiszűrése)

- **Ismétlődések megszüntetése**:  $R1 := \delta(R2)$
- 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:  $\delta$  kis-delta)
- A  $\delta$  speciális esete lesz az általánosabb  $\gamma$  műveletnek

$$R = \left( \begin{array}{|c|c|} \hline A & B \\ \hline 1 & 2 \\ 3 & 4 \\ 1 & 2 \\ \hline \end{array} \right)$$
$$\delta(R) = \begin{array}{|c|c|} \hline A & B \\ \hline 1 & 2 \\ 3 & 4 \\ \hline \end{array}$$

# 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

$$\text{SUM}(A) = 7$$

$$\text{COUNT}(A) = 3$$

$$\text{MIN}(B) = 2$$

$$\text{MAX}(B) = 4$$

$$\text{AVG}(B) = 3$$

# 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:  $\gamma_L$  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

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

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

# A vetítési művelet kiterjesztése


- $\Pi_L(R)$  kiterjesztett vetítés L listájában szerepelhetnek:
- Az R reláció attribútuma
- $E \rightarrow z$  kifejezés, ahol E az R reláció attribútumaira vonatkozó (konstansokat, aritmetikai műveleteket, függvényeket tartalmazó kifejezés), z pedig az

R

A	B
1	2
1	5
2	3

$\Pi_{A+B \rightarrow z}(R)$

Z
3
6
5



E kifejezés által számolt, az eredményekhez tartozó új attribútum nevét jelöli

# Kiválasztott sorok rendezése

- **Rendezés:**  $\tau_{A_1, \dots, A_n}(R)$
- 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:  $\tau$  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

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

# Külső összekapcsolások

- Ez **nem relációs algebrai művelet**, uis kilép a modellből.
- Lehet baloldali, jobboldali, teljes külső összekapcsolás.
- R, S sémái  $R(A_1, \dots, A_n, B_1, \dots, B_k)$ , ill.  $S(B_1, \dots, B_k, C_1, \dots, C_m)$
- $R \overset{\circ}{\bowtie} S = R \bowtie S$  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

$R \bowtie S$  eredmény

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

$R \bowtie_L S$  eredmény

A	B	C	D
1	2	3	10
1	2	3	11
⊥	6	7	12

$R \bowtie_R S$  eredmény

# SQL lekérdezések

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

---



Ismétlés: 6.2.5. Halmazműveletek  
(egyesítés, metszet és különbség)

6.4.1. Ismétlődések megszüntetése

6.4.2. Ismétlődések kezelése a  
halmazműveletek során

6.4.3.-6.4.7. Csoportosítás és összesítések az SQL-ben  
group by és having záradékok

6.3.6.-6.3.8. Összekapcsolások az SQL-ben

---

# Emlékeztető az előadás példa: Sörivők

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

**Sörök(név, gyártó)**

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

**Sörivők(név, város, tel)**

**Szeret(név, sör)**

**Felhasználó(söröző, sör, ár)**

**Látogat(név, söröző)**

# Halmazműveletek az SQL-ben

- A relációs algebrai halmazműveletek: **unió**, **különbség** és **metszet**, ebből csak az unió és különbség alapművelet, az SQL-ben mindhárom használható, implementálva van
- A **SELECT-FROM-WHERE** utasítások általában **multihalmaz** szemantikát használnak, külön kell kérni **DISTINCT**-tel ha halmazt szeretnénk kapni, viszont a **halmazműveleteknél** mégis a **halmaz szemantika** az érvényes, itt a **multihalmaz szemantikát** kell kérni: **ALL**
- Az SQL-ben a halmazműveleteket úgy vezették be, hogy azt mindig két lekérdezés között lehet értelmezni:

(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

# SQL: Ismétlődések megszüntetése

- SELECT **DISTINCT** ... FROM ...
- A  $\delta$  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.

# Példa: Intersect (metszet)

- Szeret(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 söröket, amelyekre a sörivó szereti az adott sört **és** a sörivó látogat olyan bárt, ahol felszolgálják a sört.

(**SELECT \* FROM Szeret**)

**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



# 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) ;
```

- Megj.: ORACLE-ben EXCEPT helyett MINUS-t használunk, illetve UNION és UNION ALL lehet

# SQL: Az eredmény rendezése

- SQL SELECT utasítás utolsó záradéka: **ORDER BY**
- 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**

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

# 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 *különbéle* áron árulják a Bud sört?

```
SELECT COUNT(DISTINCT ár)
FROM Felszolgal
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.

# SQL: Csoportosítás

- SELECT ...  
FROM ...  
[WHERE ... ]  
[GROUP BY kif<sub>1</sub>, ... kif<sub>k</sub> ]
- 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 paramétereiként szerepelnek,
  2. vagy a GROUP BY attribútumlistájában is megjelennek.

# Csoportok szűrése: HAVING záradék

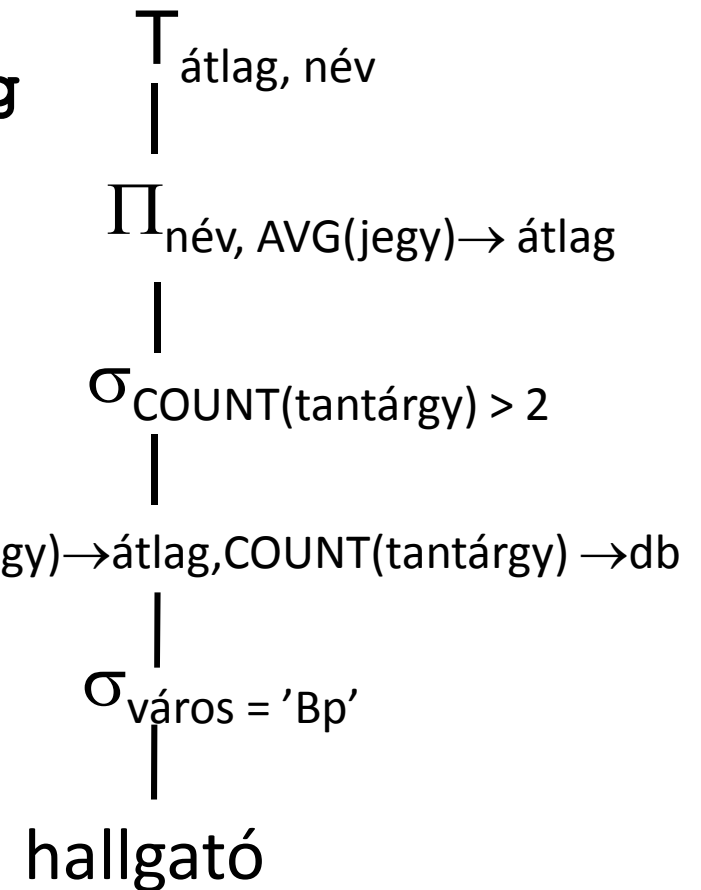
- A GROUP BY záradékot egy HAVING <feltétel> záradék követheti.
- HAVING feltétel az egyes csoportokra vonatkozik, ha egy csoport nem teljesíti a feltételt, nem lesz benne az eredményben.
- csak olyan attribútumok szerepelhetnek, amelyek:
  1. vagy csoportosító attribútumok,
  2. vagy összesített attribútumok.(vagyis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

# Példa: group by, having és order by

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
ORDER BY átlag, név;
```

(Kiterjesztett relációs algebra)



# Összefoglalás: SELECT utasítás záradékai

- Teljes SELECT utasítás(a záradékok sorrendje adott)

```
SELECT [DISTINCT] Lista1      -- 5 és 6
FROM R t                      -- 1
  [WHERE Felt1 ]              -- 2
  [GROUP BY csopkif          -- 3
   [HAVING Felt2 ] ]         -- 4
  [ORDER BY Lista2]          -- 7
```

$$\tau_{\text{Lista2}} \delta \Pi_{\text{Lista1}} \sigma_{\text{Felt2}} (\gamma_{\text{csopkif}, \dots, \text{AGGR}(\text{kif}) \rightarrow \text{onev}} \sigma_{\text{Felt1}} (R))$$

# Oracle GYAK: Összekapcsolások

- Az SQL-ben összekapcsolások számos változata megtalálható: Természetes összekapcsolás
- USING utasítással történő összekapcsolás
- Teljes (vagy két oldali) külső összekapcsolás
- Tetszőleges feltételen alapuló külső összekapcsolás
- Direktszorzat (kereszt-összekapcsolás).

```
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]
```

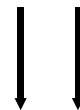
# Példa: Többtáblás lekérdezések

**EMPLOYEES**

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

**DEPARTMENTS**

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.



# Példa a természetes összekapcsolásra

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford

8 rows selected.

- A WHERE használható további megszorítások megfogalmazására. Például, ha csak a 20-as illetve 50-es department\_id-kra vagyunk kíváncsiak, akkor:

```
SELECT department_id department_name,  
       location_id city  
FROM   departments  
NATURAL JOIN locations  
WHERE  department_id IN (20, 50);
```

# 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.

# Sorváltozók használata tábláknál

- A lekérdezések átláthatósága miatt használhatunk sorváltozót (tábla alias neveket).
- A sorváltozók használata javítja a lekérdezés teljesítményét.
- A sorváltozók maximum 30 karakter hosszúak lehetnek (minél rövidebb annál jobb)
- A sorváltozók 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);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

---  
19 rows selected.

# További feltételek megadása egy összekapcsoláshoz

```
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)  
AND   e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

- Ugyanezt érhetjük el a WHERE feltétellel is, azaz:

```
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)  
WHERE  e.manager_id = 149;
```

# 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 nevezzük 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) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
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
...		
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
	190	Contracting

21 rows selected.

# 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 keresztszorzatát (vagyis a direkt szorzatát)

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

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

■■■  
160 rows selected.

# Kérdés/Válasz

- **Köszönöm a figyelmet! Kérdés/Válasz?**
- Több táblára (DEPT és EMP tábla) vonatkozó lekérdezésekre példák, összekapcsolások.
- **Házi feladat:** Oracle Példatár 2.fejezet feladatai, összesítések és csoportosítás, sorok rendezése.
- Oracle Példatár 3.fejezet feladatai, természetes összekapcsolások és alkérdések használata, de a hierarchikus és rekurzív lekérdezések még nem:  
<http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf>