

4.előadás: Adatbázisok-I.

dr. Hajas Csilla (ELTE IK)

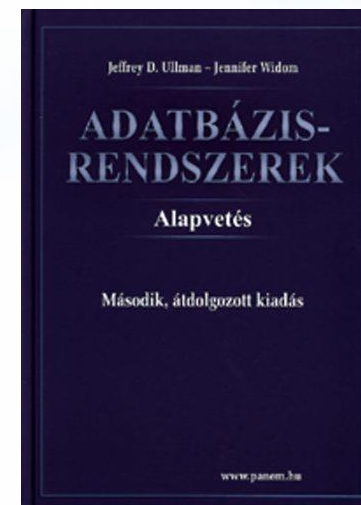
<http://sila.hajas.elte.hu/>

SQL gyakorlatban: SELECT záradékai és a kiterjesztett relációs algebra

Tankönyv:

5.1.- 5.2. Kiterjesztett relációs algebra

6.4. Ismétlődések kezelése, összesítések
csoportosítás, az SQL-ben
group by, having záradékok



Lekérdezések az SQL-ben

- 1.) Az SQL-ben halmazok helyett **multihalmazokat** használunk (vagyis egy sor többször is előfordulhat)
- 2.) **SELECT ... FROM ... WHERE ...** lekérdezésekben vagyis $\Pi_{\text{select-lista}} \sigma_{\text{where-feltétel}}$ (from-lista táblák szorzata) a select-listán és where-feltételben az attribútumnevek helyén olyan **kifejezések** állhatnak az SQL-ben, amely függvényeket és műveleti jeleket is tartalmazhat
- Az attribútumnevek helyén álló kifejezésekben használt **legfontosabb sorfüggvényeket, lásd az SQL gyakorlaton:**
 - Numerikus, karakteres, dátum, konverziós függvények
 - NULL hiányzó értéket megadott értékkel helyettesítő függvények, például NVL, COALESCE használata, stb.
 - ... részletesen, lásd az SQL gyakorlatok példáit ...

A kiterjesztett relációs algebra

- Az eddig tanult műveleteket: **vetítés** (Π), **kiválasztás** (σ), **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** megfeleltetett **új műveletekkel is kibővítjük** a rel. algebrát:
 - **Ismétlődések megszüntetése** (δ) - select distinct ..
 - **Összesítő műveletek és csoportosítás** (γ_{lista}) - group by..
 - **Vetítési művelet kiterjesztése** (Π_{lista}) - select kif [as onev]..
 - **Rendezési művelet** (τ_{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

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 relációs algebrai művelet értelmezése multihalmazok fölött

- A projekció, szelekció, Descartes-szorzat (direkt szorzat), természetes összekapcsolás, théta-összekapcsolás, stb. 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 a kiterjesztett algebrában:

1.) Ismétlődések megszüntetése

- **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: δ kis-delta)
- A δ speciális esete lesz az általánosabb γ műveletnek

$R = ($

A	B
1	2
3	4
1	2

$\delta(R) =$

A	B
1	2
3	4

2.) 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

3.) Összesítő (aggregáló) 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

Összesítő (aggregáló) függvények

- Miért hívják **aggregáló** függvényeknek?
- Ha kiszámoltuk az összeget a tábla bizonyos soraira, akkor újabb sorok figyelembe vételével (aggregálva) felhasználhatjuk a korábban kapott eddigi összeget
- Kivéve például az AVG esetén a fenti nem igaz, viszont az AVG érték hányadosa a SUM és COUNT értékeknek, amelyeket aggregálva tudunk megkapni.

R =

A	B
1	3
3	4
3	2

SUM(A) = 7

COUNT(A) = 3

MIN(B) = 2

MAX(B) = 4

AVG(B) = 3

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

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
beosztások
szerint
a fizetések
összege

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.

Összesítések és csoportosítás --- 2

- 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.

Összesítések és csoportosítás --- 3

- **Értelmezése, kiértékelése:** Osszuk az R tábla 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: Összesítés és csoportosításra

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

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.

$$\sigma_{csop.kiv.felt} \gamma_{csop.attr.} \sigma_{felt} (R)$$

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

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

4.) 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: 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

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

5.) 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. $S(B_1, \dots, B_k, C_1, \dots, C_m)$
- $R \overset{0}{\bowtie} S = R \bowtie S$ 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

$R \overset{0}{\bowtie} S$ eredmény

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

$R \overset{0}{\bowtie}_L S$ eredmény

A	B	C	D
1	2	3	10
1	2	3	11
⊥	6	7	12

$R \overset{0}{\bowtie}_R S$ eredmény

Nézzük meg a kiterjesztett algebra műveleteit az SQL SELECT-ben

Az előadások példája (emlékeztető)

- A lekérdezések **Sörivők adatbázissémán** alapulnak

Sörök(név, gyártó)

Sörözők(név, város, tulaj, engedély)

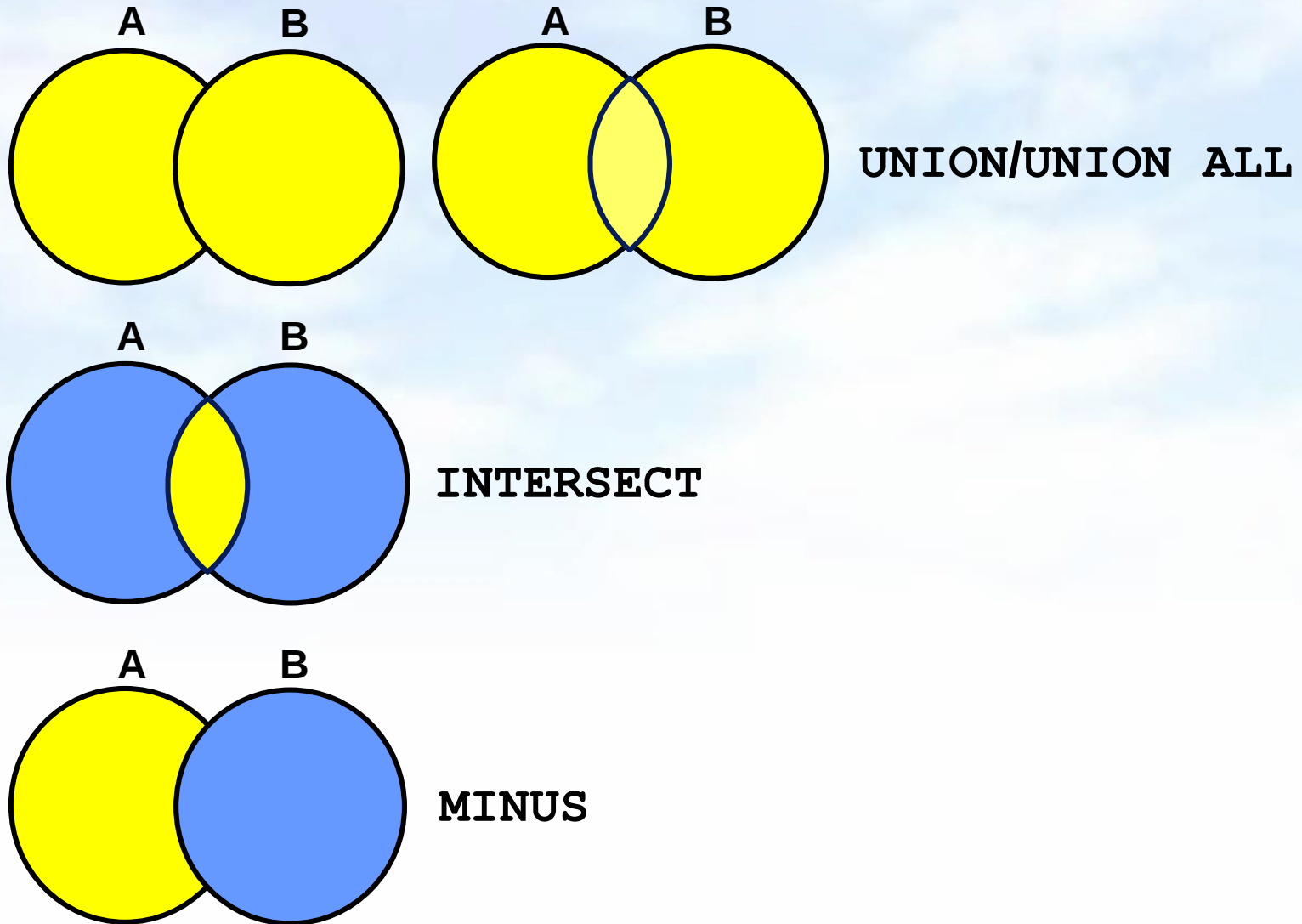
Sörivők(név, város, tel)

Szeret(név, sör)

Felszolgál(söröző, sör, ár)

Látogat(név, söröző)

Halmazműveletek



Multihalmaz szemantika az SQL-ben

- A SELECT-FROM-WHERE utasítások általában multihalmaz szemantikát használnak, külön kell kérni DISTINCT-tel ha a halmazt szeretnénk eredményül, kapni, a DISTINCT kiküszöböli az ismétlődéseket.
- A halmazműveleteknél viszont a halmaz szemantika az érvényes, mert ott az az egyszerűbb, és a multihalmaz szemantikát (ezt fogjuk a mai előadáson megbeszélni) külön kell kérni az ALL szócskával
- Az SQL-ben a halmazműveletek:

(SFW-lekérdezés1)

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

(SFW-lekérdezés2);

Halmazműveletek irányelvei

- A SELECT-listákban szereplő kifejezéseknek számban és adattípusban meg kell egyezniük.
- Használhatunk zárójeleket a végrehajtási sorrend megváltoztatására.
- Az ORDER BY záradék:
 - Csak az utasítás legvégén szerepelhet
 - Az első SELECT utasításban szereplő oszlopneveket vagy másodneveket jeleníti meg.

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: 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 a sörivókat és söröket, amelyekre a sörivó szereti az adott sört **és** a sörivó látogat olyan sörözőt, ahol felszolgálják a sört.

(SELECT * FROM Szeret)

INTERSECT

(SELECT név, sör

FROM Látogat, Felszolgál

WHERE Látogat.söröző = Felszolgál.söröző) ;

(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

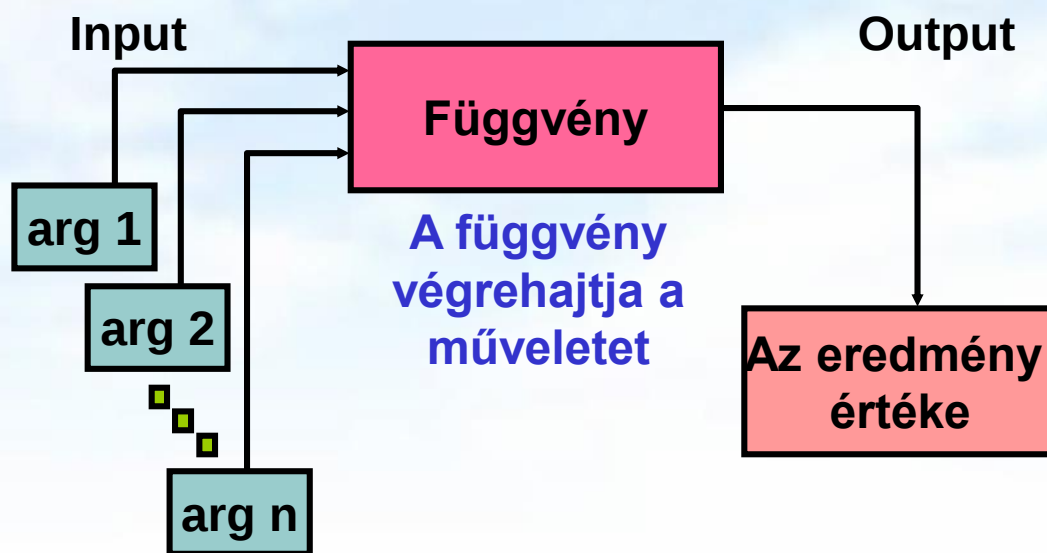
SQL: Ismétlődések megszüntetése

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

SQL sorfüggvények (Oracle példák)

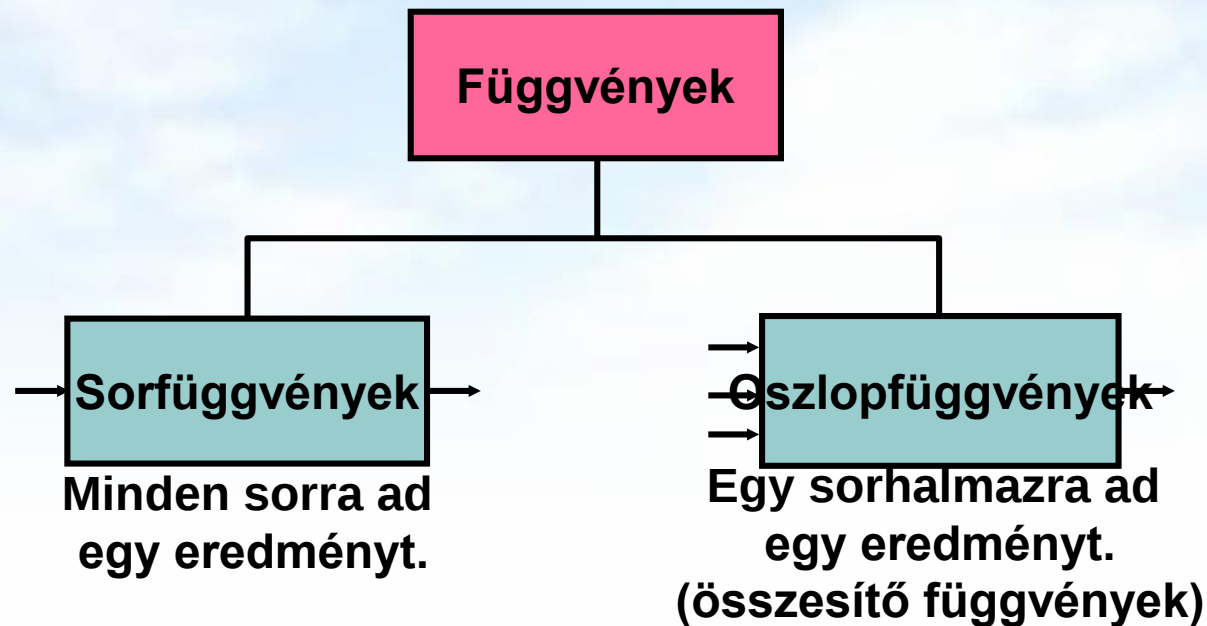
- Különböző típusú SQL sorfüggvények megismerése
- A karaktertípusú, numerikus, illetve dátumtípusú sorfüggvények használata a `SELECT` utasításokban
- Típus-átalakító sorfüggvények megismerése

SQL-függvények



A bemutatott függvények többsége Oracle-specifikus.

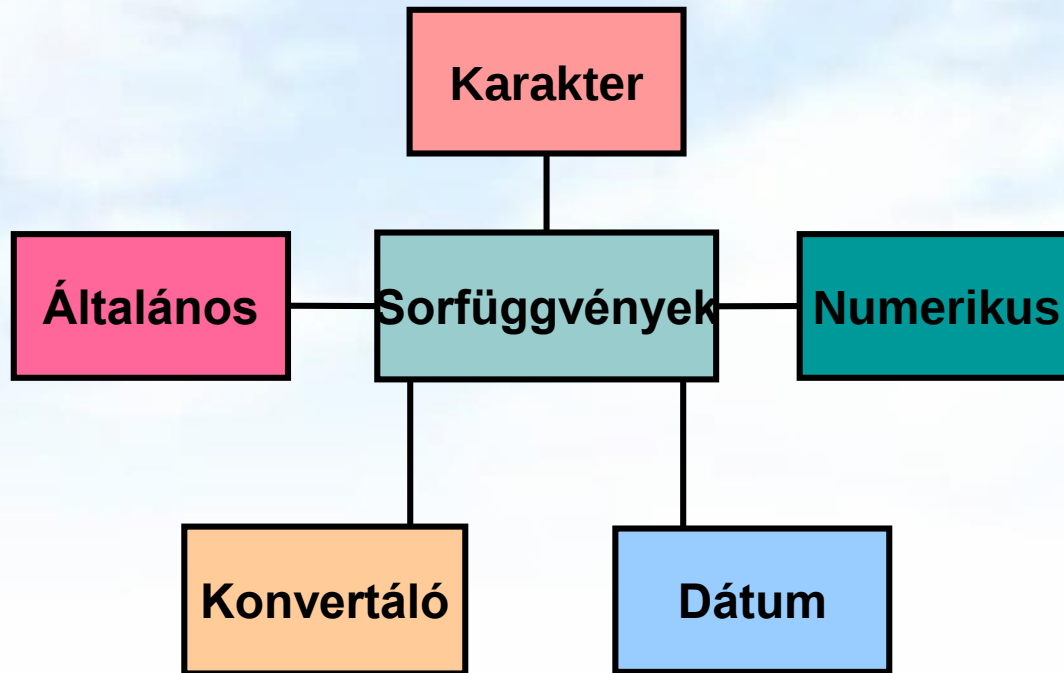
Az SQL-függvények két típusa



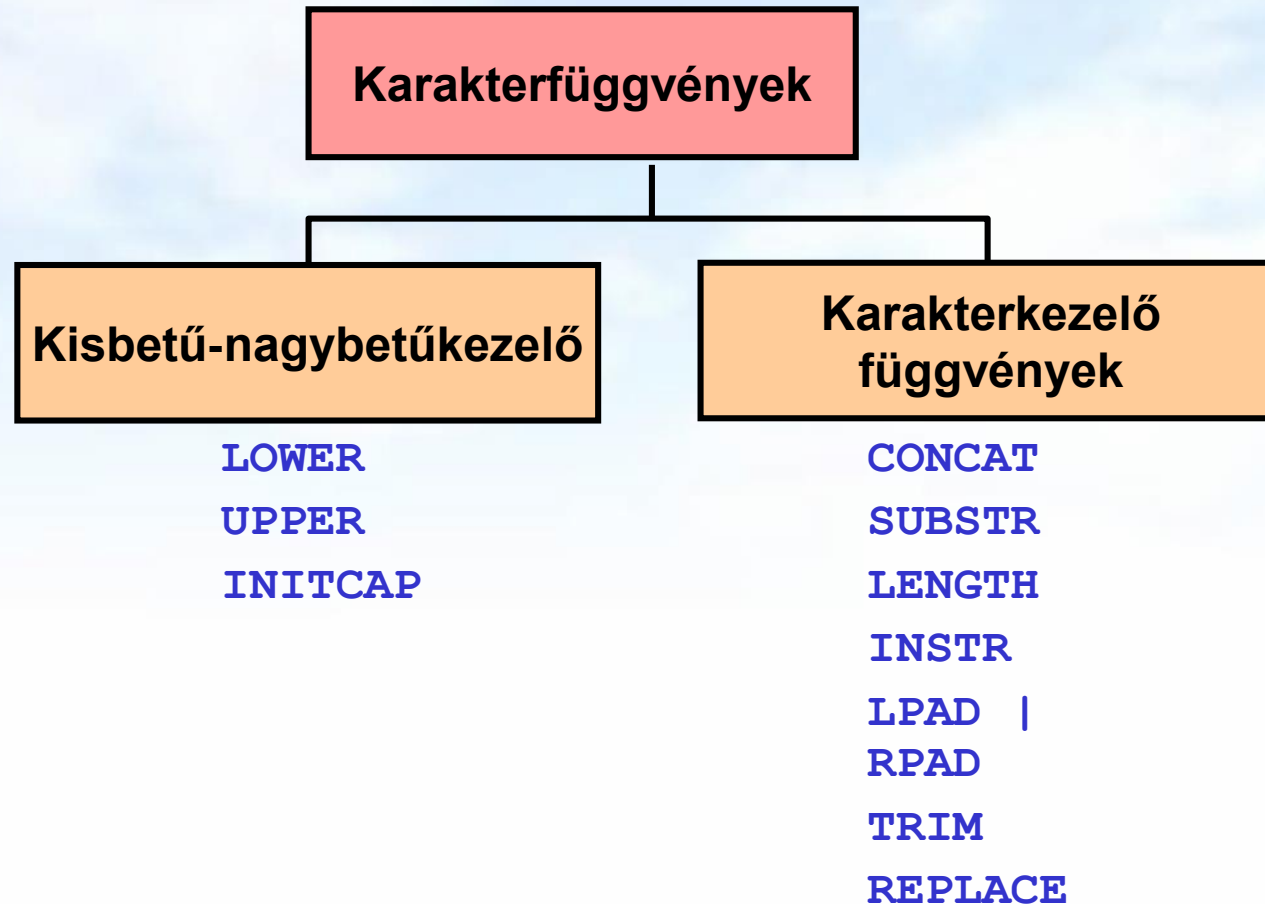
Sorfüggvények

- A sorfüggvények: `function_name [(arg1, arg2, ...)]`
 - Az adattételek átalakítására, feldolgozására használhatók.
 - Több argumentumból egy értéket eredményez.
 - Az argumentum lehet felhasználói konstans, változó, oszlopnév, kifejezés.
 - A lekérdezés eredményének minden sorára meghívódik.
 - Minden sorra egy értéket ad vissza.
 - Az eredménye más adattípusú is lehet mint az argumentum.
 - Egymásba lehet ágyazni.
 - Használható a SELECT, WHERE és ORDER BY stb. részekben.

Sorfüggvények



Karakterkezelő függvények



Karakterkezelő függvények

Függvény	Leírás
LOWER(<i>column expression</i>)	Kisbetűre konvertál
UPPER(<i>column expression</i>)	Nagybetűre konvertál
INITCAP(<i>column expression</i>)	A szavak első betűjét nagybetűre, a többi kisbetűre konvertálja
CONCAT(<i>column1 expression1, column2 expression2</i>)	A két karakterértéket összefűzi; ugyanaz mint a művelet.
SUBSTR(<i>column expression,m[,n]</i>)	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 karaktert megkapjuk a karakterlánc végéig.

Karakterkezelő függvények

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

Kisbetű-nagybetű kezelő függvények

➤ Például:

Függvény	Eredmény
LOWER('SQL Nyelv')	sql nyelv
UPPER('SQL Nyelv')	SQL NYELV
INITCAP('SQL Nyelv')	Sql Nyelv

```
SELECT 'The job id for' || UPPER(last_name) || ' is  
,  
|| LOWER(job_id) AS "EMPLOYEE DETAILS"  
FROM employees;
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp

Karakterkezelő függvények használata

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE('JACK and JUE', 'J', 'BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

A függvényekbe helyettesítő változókat is tehetünk:

select upper('&valami') from dual

Numerikus függvények

- **ROUND:** Adott tizedes jegyre kerekít (ha n negatív, akkor a tizedesvesszőtől balra kerekít).
- **TRUNC:** Adott tizedesjegy 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

A dátumok használata

- A SYSDATE függvény segítségével megkaphatjuk:
 - az adatbázis-kezelő rendszerdátumát és
 - az adatbázis-kezelő rendszeridejét.

▪ Ha a hónapok nevét magyarul akarjuk látni:

```
ALTER SESSION set NLS_LANGUAGE = "HUNGARIAN"  
select to_char(sysdate,'yyyy-Month-dd') from dual
```

TO_CHAR(SYSDATE,'YYYY-MONTH-DD')
2008-Február -16

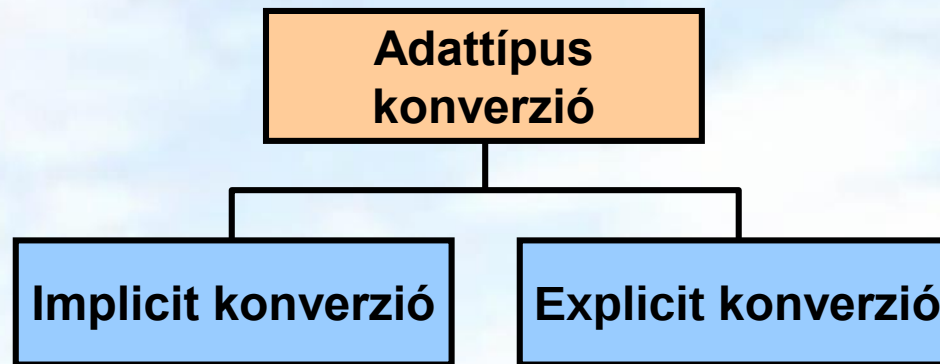
Dátumaritmetika

- Egy dátumhoz hozzá lehet adni vagy ki lehet vonni egy számot. A számnak megfelelő nappal növeli vagy csökkenti a dátum értékét.
- Két dátum kivonása a köztük eltelt napok számát adja vissza.
- Mivel egy óra a nap 24-ed része, így órákat is hozzá tudunk adni egy dátumhoz.

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

Konvertáló függvények



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

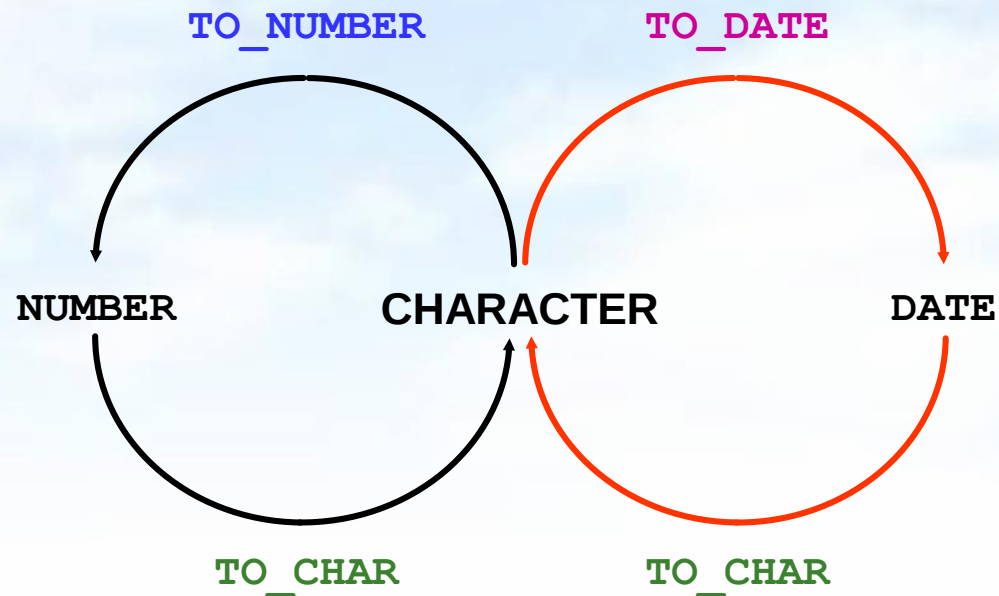
Implicit adattípus-konverzió

- A következő **típusok** konverzióját az Oracle szerver automatikusan elvégzi:

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

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 <i>fmt</i> 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.

A dátumformátum leggyakoribb elemei

Elem	Értelme
YYYY	Évszám (számokkal)
YEAR	Évszám (szövegesen)
MM	Hónap sorszáma
MONTH	Hónap teljes neve
MON	Hónap 3 betűvel rövidítve
DY	A hét napja 3 betűvel rövidítve
DAY	A hét napjának teljes neve
DD	A nap sorszáma a hónapban

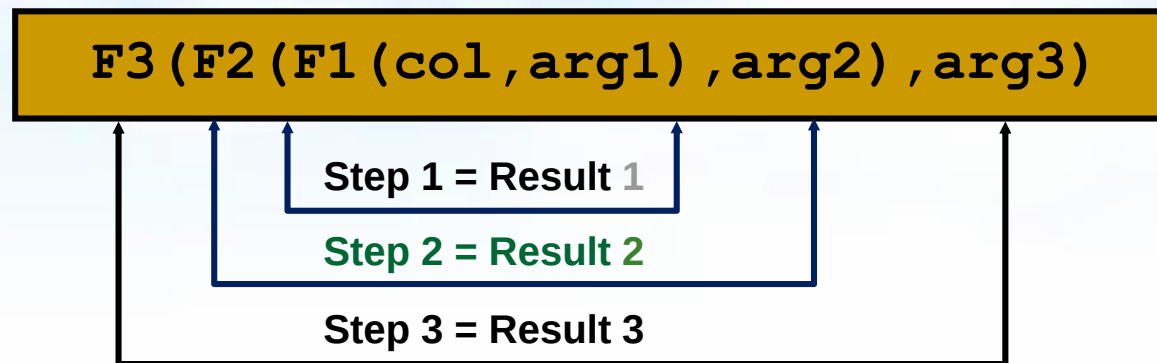
select to_char(sysdate,'Year') from dual

TO_CHAR(SYSDATE,'YEAR')

Two Thousand Eight

Függvények egymásba ágyazása

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

Feltételes kifejezések

- Segítségükkel IF-THEN-ELSE típusú logikát lehet használni az SQL utasításban
- Kétféle módszert használhatunk:
 - CASE expression
 - DECODE function

A CASE kifejezés

- Feltételes lekérdezéseket lehet megfogalmazni vele az IF-THEN-ELSE utasításhoz hasonlóan:

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
      [WHEN comparison_expr2 THEN return_expr2  
      WHEN comparison_exprn THEN return_exprn  
      ELSE else_expr]  
END
```

Sorfüggvények összefoglalása

- Ebben a részben megtanultuk:
 - hogyan kell a dátumokkal műveleteket végezni, függvényekben dátumokat használni
 - hogyan lehet módosítani az adatokat függvények segítségével
 - hogyan lehet a lekérdezés eredményét adó sorokat formázni
 - hogyan lehet különböző dátumformátumokat használni a megjelenítésben
 - hogyan lehet az adattípusokat konvertálni
 - hogyan kell használni az NVL függvényt
 - hogyan működnek a feltételes - IF-THEN-ELSE – logikájú kifejezések

SQL: Összesítések (aggregálás)

- **Összesítések** (aggregáló műveletek) a **SELECT** listán alkalmazhatjuk egy oszlopra (kifejezésre).
<Aggregáló művelet>(kifejezés) [[AS] onév], ...
- **Az 5 legfontosabb összesítő függvény:**
SUM, **COUNT**, **MIN**, **MAX** (aggregálással számolható),
AVG (bevezették ezt is, mivel gyakran kell AVG és aggregálással számolható függvényekből számolható)
- **Példa:** A **Felszolgál(söröző, sör, ár)** tábla segítségével adjuk meg a Bud átlagos árát:

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

Összesítések (aggregálás)

- Itt is fontos a **halmaz, multihalmaz** megkülönböztetés.
- Aggregáló_művelet(ALL|DISTINCT R.A)
ALL (ez az alapértelmezés, ha nincs jelezve, akkor ALL),
ha DISTINCT szerepel, akkor csak a különböző
értékűeket veszi figyelembe az összesítéseknél.
- NULL nem számít a SUM, AVG, COUNT, MIN, MAX
függvények kiértékelésekor. (implementáció függő,
ellenőrizzük le)
- De ha nincs NULL értéktől különböző érték az
oszlopban, akkor az összesítés eredménye NULL.
- Kivétel: COUNT az üres halmazon 0-t ad vissza.
- COUNT(*) az eredmény sorainak számát adja meg.

NULL értékek nem számítanak az összesítésben, kivéve COUNT(*)

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

A Bud sört árusító
kocsmák száma,
üres halmazon
0-t ad

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

A Bud sört ismert
áron árusító
kocsmák száma,
üres halmazon
NULL-t ad.

Ismétlődések kiküszöbölése összesítésben (DISTINCT)

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

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

SQL: Csoportosítás

- **SELECT ...**
FROM ...
[WHERE ...]
[GROUP BY kif_1 , ... kif_k]
- Egy SELECT-FROM-WHERE kifejezést **GROUP BY záradékkal** folytathatunk, melyet attribútumok (kifejezések) 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
 - vagy egy összesítő függvény paramétereiként szerepelnek,
 - vagy a GROUP BY attribútumlistájában is megjelennek.

Az összesítő függvények csak két mélységig ágyazhatóak egymásba

- A **Felszolgál(bár, sör, ár)** tábla segítségével fejezzük ki szavakkal mit jelent az alábbi két lekérdezés? Melyik adhat nagyobb eredményt?

a.) **SELECT MAX (AVG (ár))**
FROM Felszolgál
GROUP BY sör;

b.) **SELECT AVG (MAX (ár))**
FROM Felszolgál
GROUP BY sör;

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:
 - vagy csoportosító attribútumok,
 - vagy összesített attribútumok.(vagyis ugyanazok a szabályok érvényesek, mint a SELECT záradéknál).

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 Felszolgal
ORDER BY ár DESC, sör**

Összefoglalás: SELECT utasítás záradékai

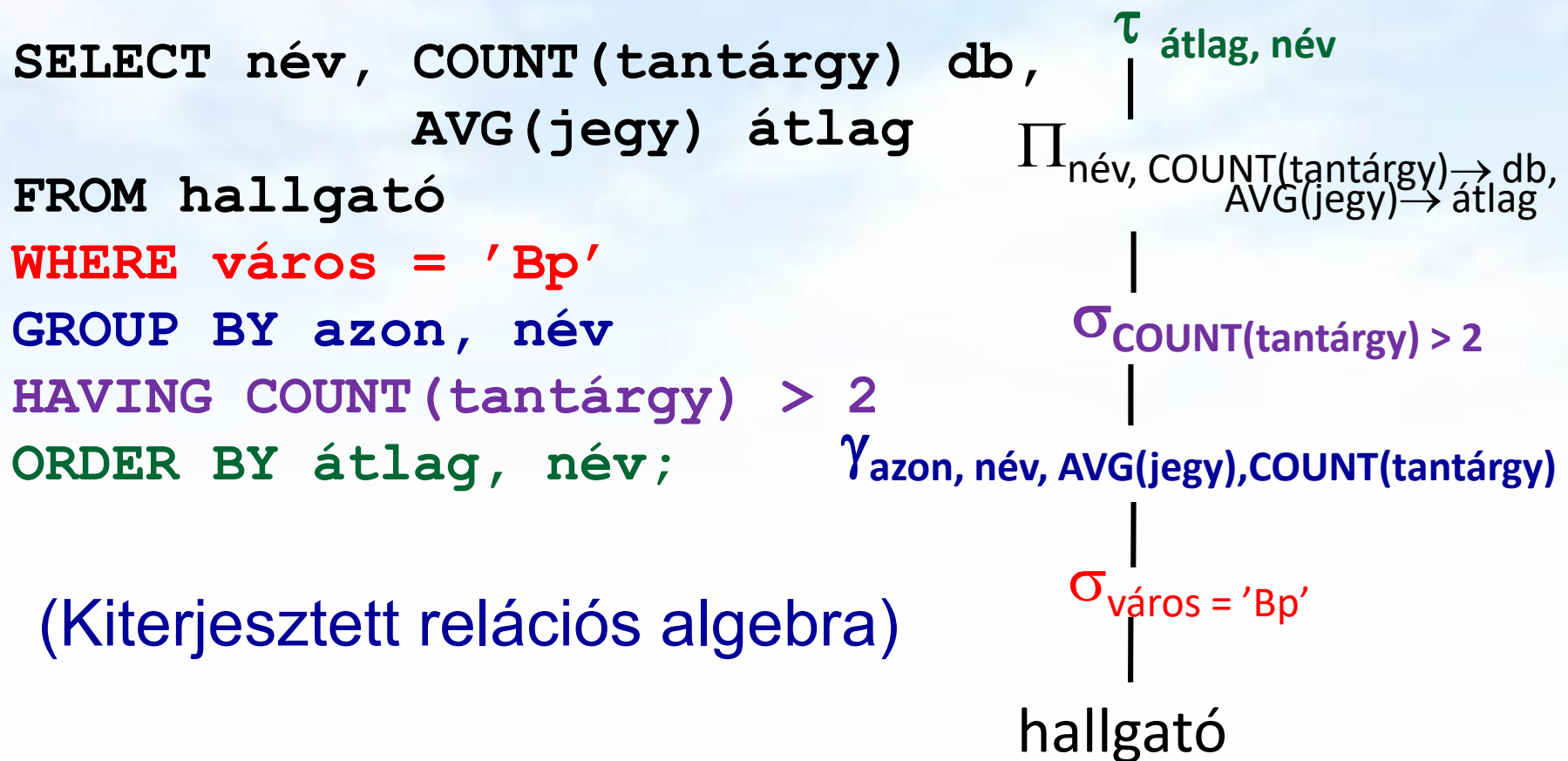
- Teljes SELECT utasítás(a záradékok sorrendje adott)

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

$$\tau_{\text{Lista2}} \left(\delta \left(\Pi_{\text{Lista1}} \sigma_{\text{Felt2}} \left(\gamma_{\text{csopkif, AGGR(kif)}} \sigma_{\text{Felt1}} (\mathbf{R}) \right) \right) \right)$$

Példa: group by, having és order by

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



Példa: külső összekapcsolás+csoportosítás

```
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;
```

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

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?

Feladatok

- Házi feladat: Gyakorlás az Oracle Példatár feladatai:
- Példatár 1.-2. fejezetek feladatai SQL-lekérdezésekben kifejezések, függvények, összesítések és csoportosítás
- **Keressünk új megoldásokat!** „Amikor azt gondolod, hogy már minden lehetőséget kimerítettél, még mindig van legalább egy.” (Thomas Alva Edison)