Kerepes Tamás – Czinkóczki László

Adatbázisok

Hatékony elemzések az adatbázisban: Analitikus függvények

Az Oracle SQL függvényei

Kategória	Példa
Analitikus	RANK,SUM,AVG,LAG,LEAD,FIRST_VALUE
Konverziós	TO_NUMBER,TO_CHAR,TO_DATE,TO_CLOB, TO_YMINTERVAL,TO_DSINTERVAL
Dátum	ADD_MONTHS,TRUNC,ROUND
Csoport	SUM,AVG,COUNT,MIN,MAX,CORR,LISTAGG
Numerikus	ABS,TRUNC,ROUND,SQRT,LN,POWER, CEIL,FLOOR,SIN,SINH,COS,COSH
Egyéb	DECODE,CASE,NVL,NVL2,COALESCE, NLSSORT,EMPTY_BLOB(),CHR
Karakteres	SUBSTR,INSTR,REPLACE,LPAD, RPAD

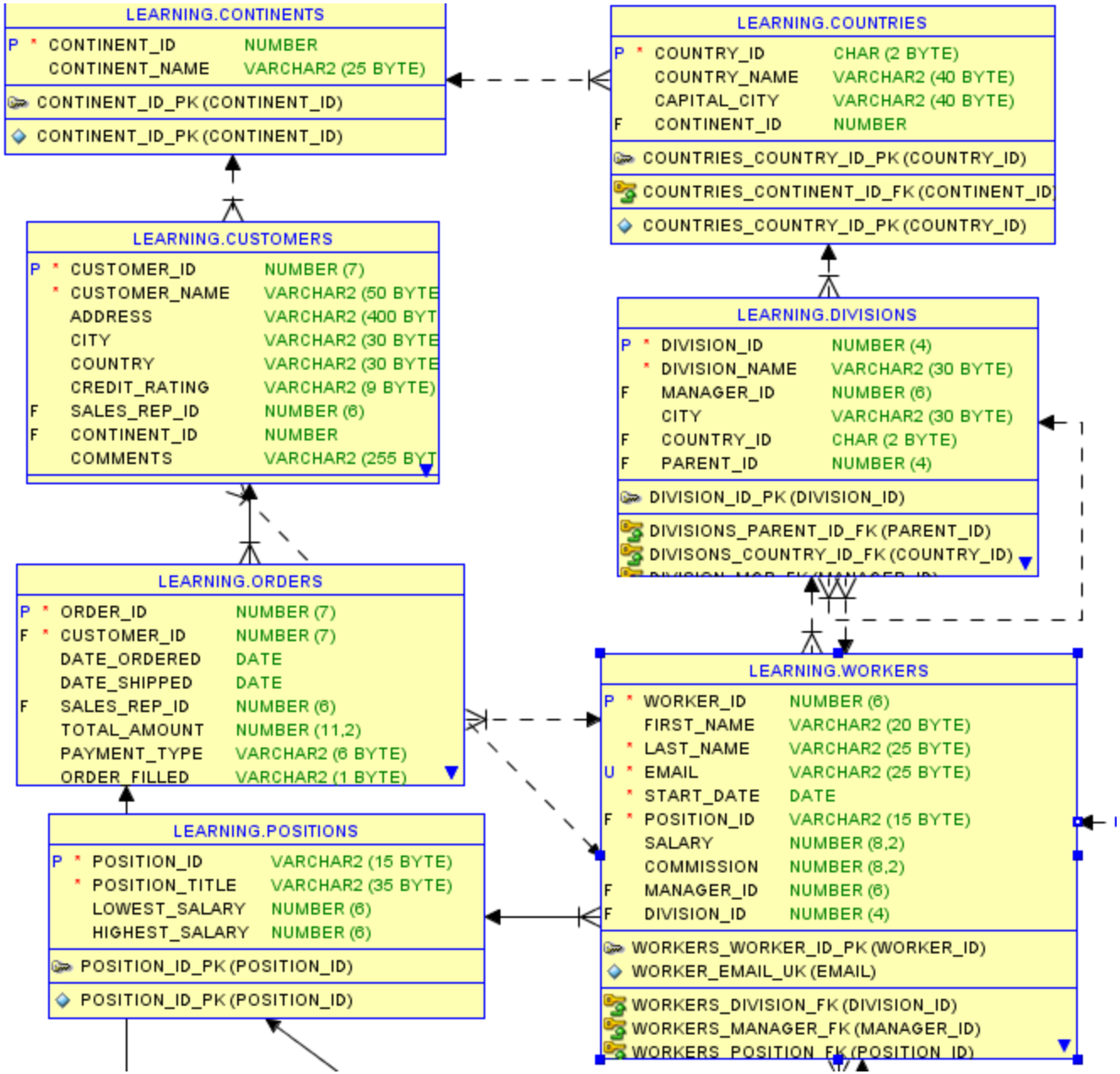
Bevezetés

- Az első analitikus függvények az Oracle 8.1.6-os verziójában kerültek bevezetésre.
- Számos feladat csak analitikus függvények segítségével oldható meg SQL-ben. (halmozott összesítések, mozgó MIN, MAX, AVG)
- Bizonyos feladatok analitikus függvényekkel jóval hatékonyabban oldhatók meg mint hagyományos eszközökkel (több időszakokhoz tartozó adatok együttes kezelése)
- ANSI/ISO szabványnak megfelelő konstrukciók (SQL 1999)
- A legfontosabb célkitűzés: a hatékony elemzés!

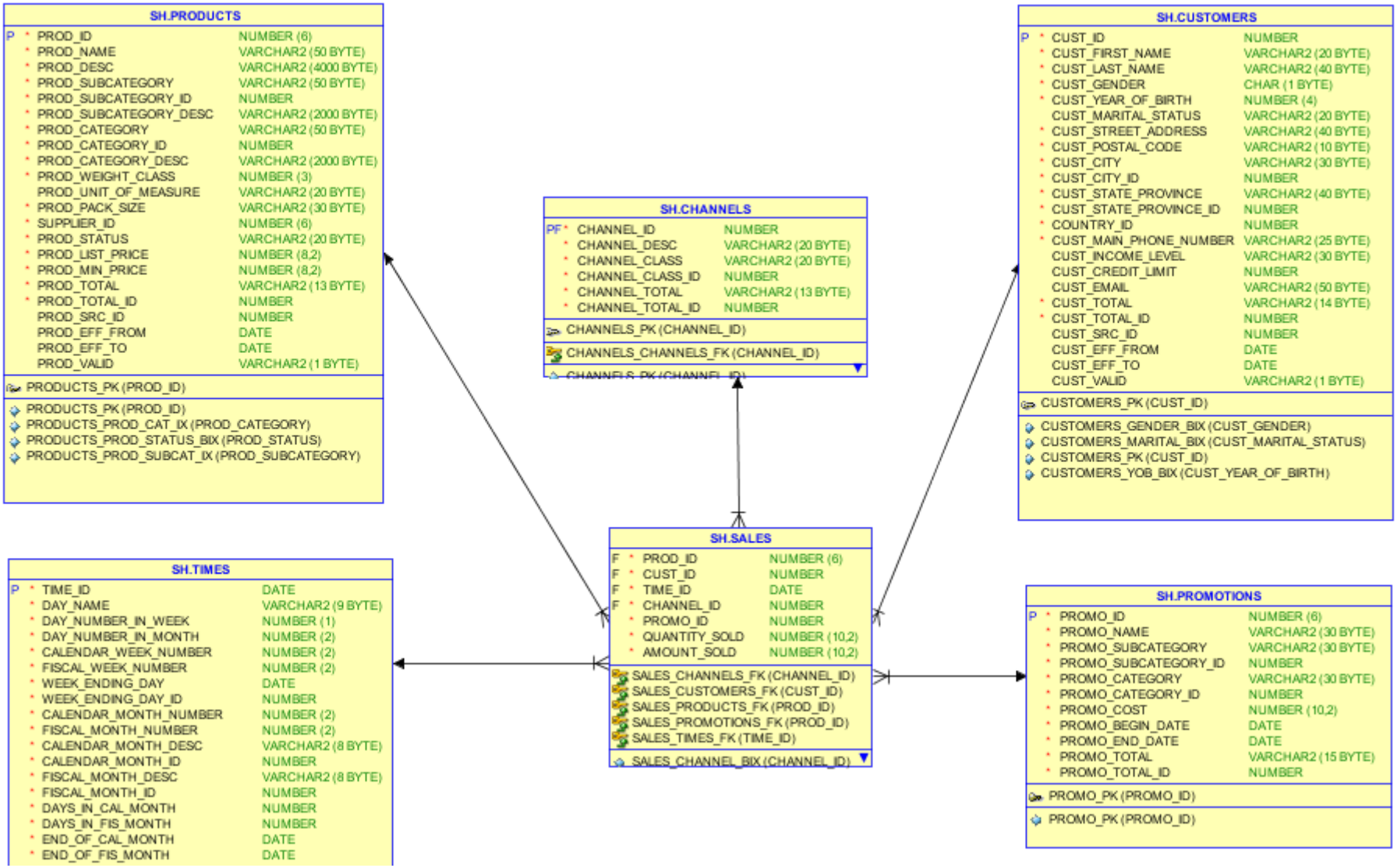
Koncepció

- Az analitikus függvények logikai csoportok (partition) sorain operálnak.
- Ellentétben a hagyományos csoport függvényekkel, az analitikus függvények a partíció minden sorára generálnak értéket.
- A partíción belül ablak (window) definiálható az analitikus utasításrésszel (analytic clause).
- Az ablakot mindig az aktuális sorhoz (current row) relatívan specifikáljuk, intervallumként:
  - Fizikailag (a sorok számával)
  - Logikailag (pl idő adatok alapján)

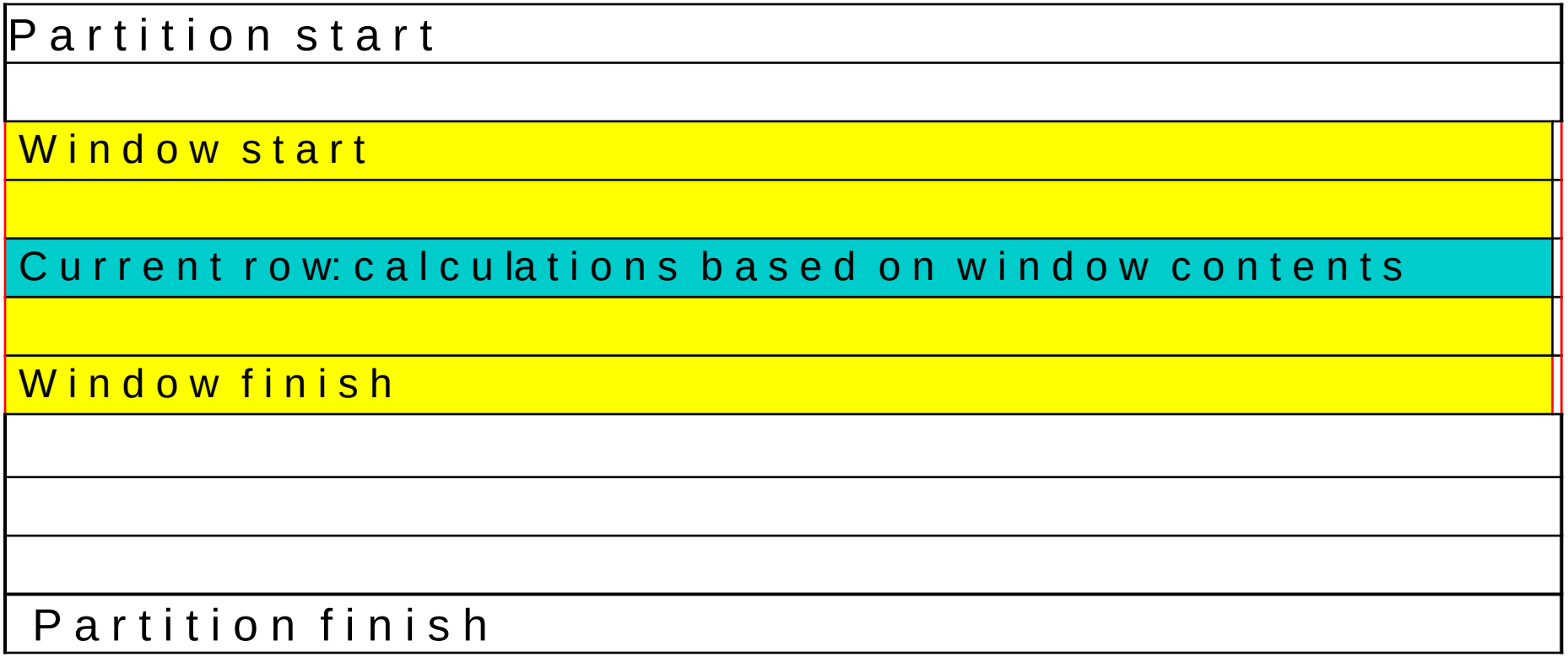
A LEARNING séma részhalmaza



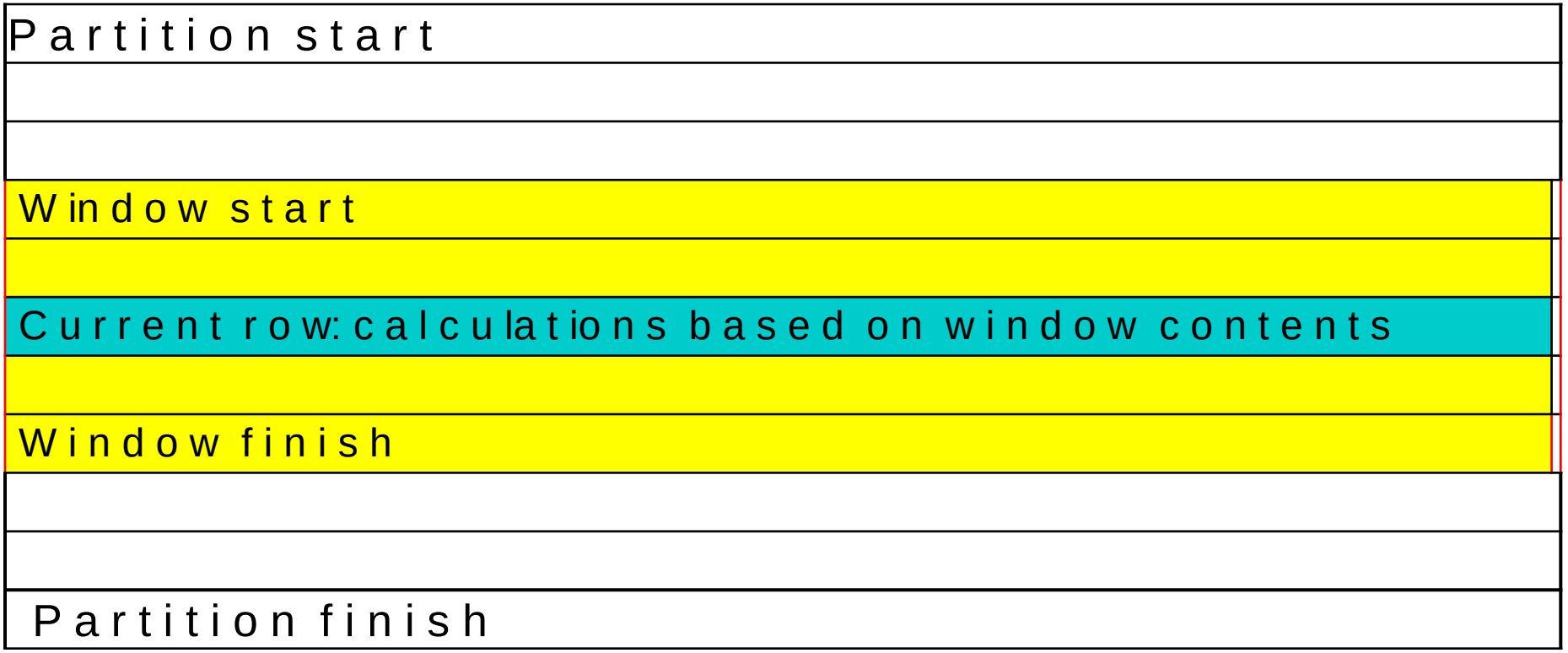
Az SH séma részhalmaza (Star schema)



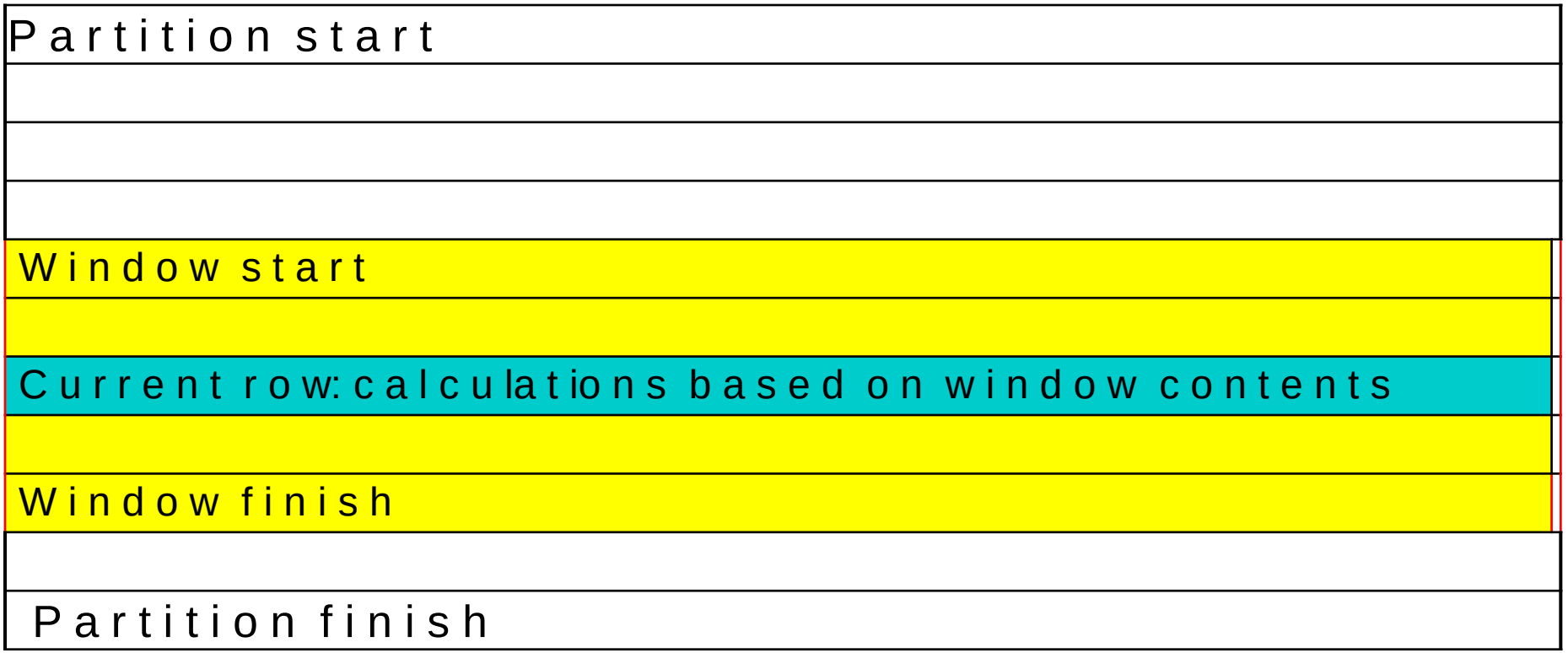
Az analitikus függvények anatómiája



Az analitikus függvények anatómiája



Az analitikus függvények anatómiája



A feldolgozás sorrendje

1. lépés

joins,

WHERE,

GROUP BY,

& HAVING

klauzulák

2. lépés

Partíciók létrehozása

Az analitikus függvények alkalmazása a partíció minden sorára

3. lépés

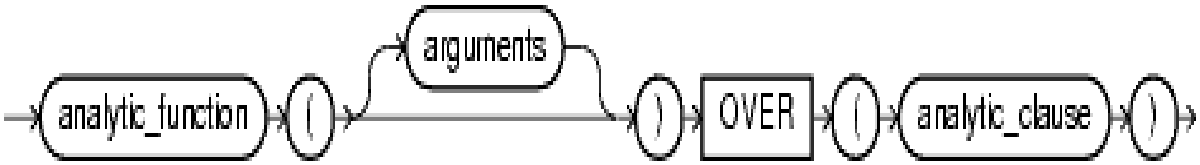
ORDER BY

Az analitikus függvények (egy lehetséges) csoportosítása

- Rangsoroló függvények:
  Normál:RANK, sűrű: DENSE_RANK, % -os értékek
- Fix méretű ablak technikát alkalmazó függvények:
  Mozgó MIN, MAX, AVG, SUM
- Riport készítő függvények:
  Halmozott összesítések (MIN, MAX, AVG, SUM)
- Más időszakhoz tartozó adatok kezelése:
  LAG (“előző”), LEAD (“követő”)
- Statisztikai függvények:
  Hisztogram: WITH_BUCKET, Kvartilis: NTILE
  Korrelációs együttható (CORR)

Az analitikus függvények szintakszisa I.

analytic function::=



analytic_clause::=

6 Rangsorolási technikák

A hagyományos rangsorolás nem oldja meg a problémát!

```
SELECT ROWNUM,last_name,salary FROM
(SELECT * FROM workers ORDER BY salary DESC) t
WHERE ROWNUM<=&db;
```

```
DB=2 esetén
1 GAUSS          24000
2 EULER          17000

DB=3 esetén
1 GAUSS          24000
2 EULER          17000
3 BERNOULLI      17000
```

Rangsoroló függvények

RANK

Sorszámot ad minden, az OVER utasításrészben szereplő kifejezésre

DENSE_RANK

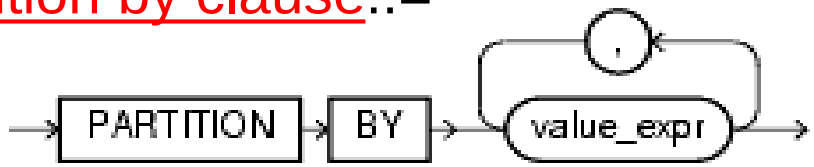
Sorszámot ad minden, az OVER utasításrészben szereplő különböző kifejezésre

```
SELECT last_name,salary,
RANK() OVER( ORDER BY salary DESC) normal_rank,
DENSE_RANK() OVER( ORDER BY salary desc) dense_rank
FROM workers;
```

	LAST_NAME	SALARY	NORMAL_RANK	DENSE_RANK
1	GAUSS	24000	1	1
2	EULER	17000	2	2
3	BERNOULLI	17000	2	2
4	RUSSELL	14000	4	3
5	COANDA	13000	5	4
6	BERING	12000	6	5
7	NERUDA	12000	6	5
8	GRIEG	12000	6	5
9	FEUERSTEIN	11500	9	6

A lekérdezés eredmény halmazának particionálása

query partition by clause::=




```
SELECT division_id,last_name,salary,
RANK()OVER( PARTITION BY division_id ORDER BY salary
desc) normal_rank,
DENSE_RANK()OVER( PARTITION BY division_id ORDER BY
salary desc) dense_rank
FROM workers;
```

	DIVISION_ID	LAST_NAME	SALARY	NORMAL_RANK	DENSE_RANK
1	10	JOPLIN	4400	1	1
2	20	COANDA	13000	1	1
3	30	GURION	11000	1	1
4	30	KHAN	3100	2	2
5	30	PELE	3100	2	2
6	30	EUSEBIO	2800	4	3
7	40	BARTÓK	6500	1	1

A lekérdezés eredmény halmazának particionálása sorszámozással

A ROW_NUMBER analitikus függvény<> ROWNUM !

```
SELECT  row_number() OVER(ORDER BY division_id)
sorszam ,ROWNUM,
ROW_NUMBER() OVER( PARTITION BY division_id order by
worker_id)
divizio_sorszam,division_id,last_name,salary,
RANK()OVER( PARTITION BY division_id ORDER BY salary
desc) normal_rank,
DENSE_RANK()OVER( PARTITION BY division_id ORDER BY
salary desc) dense_rank
FROM workers;
```

	SORSZAM	ROWNUM	DIVIZIO_SORSZAM	DIVISION_ID	LAST_NAME	SALARY	NORMAL_RANK	DENSE_RANK
1	1	47	1	10	JOPLIN	4400	1	1
2	2	48	1	20	COANDA	13000	1	1
3	3	14	1	30	GURION	11000	1	1
4	4	15	2	30	KHAN	3100	2	2
5	5	16	3	30	PELE	2900	3	3
6	6	17	4	30	EUSEBIO	2800	4	4
7	7	50	1	40	BARTÓK	6500	1	1
8	8	28	1	50	REMBRANDT	2500	6	6

Top N analízis 1. példa
Listázzuk ki az évi 5 legjobb hónapot

```
SELECT T.* FROM (
SELECT t.calendar_year,t.calendar_month_name MONTH,
       sum(s.amount_sold) amount,
       SUM(s.QUANTITY_sold) quantity,
RANK()  OVER (PARTITION BY t.CALENDAR_YEAR ORDER BY
       SUM(s.AMOUNT_sold) DESC ) RANG
 FROM sh.sales s, sh.times t ,sh.products p where
   s.time_id=t.time_id AND s.prod_id=p.prod_id
 GROUP BY t.calendar_year,t.calendar_month_name) t
       WHERE t.rang<=5;
```

	CALENDAR_YEAR	MONTH	AMOUNT	QUANTITY	RANG
1	1998	February	2372690.87	14307	1
2	1998	January	2277420.49	15132	2
3	1998	October	2236464.53	19878	3
4	1998	September	2167008.19	17059	4
5	1998	April	1975978.3	11818	5
6	1999	February	2357629.26	24122	1
7	1999	January	2077439.76	20437	2
8	1999	September	2030917.97	23024	3
9	1999	December	1931931.01	18873	4
10	1999	August	1904916.61	22225	5

Termék kategóriánként az 5 legjobb hónap

```
SELECT t.* FROM (SELECT p.prod_category,
t.calendar_year||' '||t.calendar_month_name MONTH,
TO_CHAR(SUM(s.amount_sold),'999,999,999.99') amount,
SUM(s.quantity_sold) quantity,
RANK()  OVER (PARTITION BY p.prod_category
ORDER BY SUM(s.amount_sold) DESC ) RANG
FROM sh.sales s, sh.times t , sh.products p
WHERE s.time_id=t.time_id AND s.prod_id=p.prod_id
GROUP BY p.prod_category,t.calendar_year||'
'||t.calendar_month_name) t
WHERE t.rang<=5;
```

	PROD_CATEGORY	MONTH	AMOUNT	QUANTITY	RANG
1	Electronics	2000 September	691,448.94	4080	1
2	Electronics	2000 November	661,146.75	4246	2
3	Electronics	2000 October	652,224.76	3790	3
4	Electronics	2000 December	553,534.39	2735	4
5	Electronics	1999 December	519,524.79	2538	5
6	Hardware	1998 February	803,488.51	547	1
7	Hardware	1998 November	732,740.04	556	2
8	Hardware	1998 April	702,028.45	441	3
9	Hardware	1998 September	687,276.80	516	4
10	Hardware	1998 January	641,850.31	452	5

„Mi lenne ha?” típusú elemzések a RANK függvénnyel

```
SELECT division_id,SUM(salary),COUNT(*)
RANK(10000)
WITHIN GROUP(ORDER BY salary DESC)rank_normal,
DENSE_RANK(10000)
WITHIN GROUP(ORDER BY salary DESC) rank_dense
FROM workers GROUP BY division_id;
```

	DIVISION_ID	SUM(SALARY)	COUNT(*)	RANK_NORMAL	RANK_DENSE
1	10	4400	1	1	1
2	20	13000	1	2	2
3	30	19800	4	2	2
4	40	6500	1	1	1
5	50	18800	6	1	1
6	60	24000	4	1	1
7	70	10000	1	1	1
8	80	86300	9	5	5
9	90	58000	3	4	3
10	100	52800	8	2	2
11	110	20300	2	2	2
12	120	2500	1	1	1
13	130	2600	1	1	1
14	160	20600	4	1	1
15	200	6000	1	1	1
16	210	34100	4	2	2
17	230	6500	1	1	1

PERCENT_RANK és a CUME_DIST

(rang a partícióban - 1) / (a partíció sorainak száma - 1)

(rang a partícióban) / (a partíció sorainak száma)

```
SELECT division_id, last_name, salary,rank() OVER
(PARTITION BY division_id ORDER BY salary DESC) AS rank,
ROUND(percent_rank() OVER (PARTITION BY division_id ORDER
BY salary DESC),2) AS percent_rank,
ROUND(CUME_DIST()OVER(PARTITION BY division_id
ORDER BY  Salary DESC),2) cum_dist
FROM workers;
```

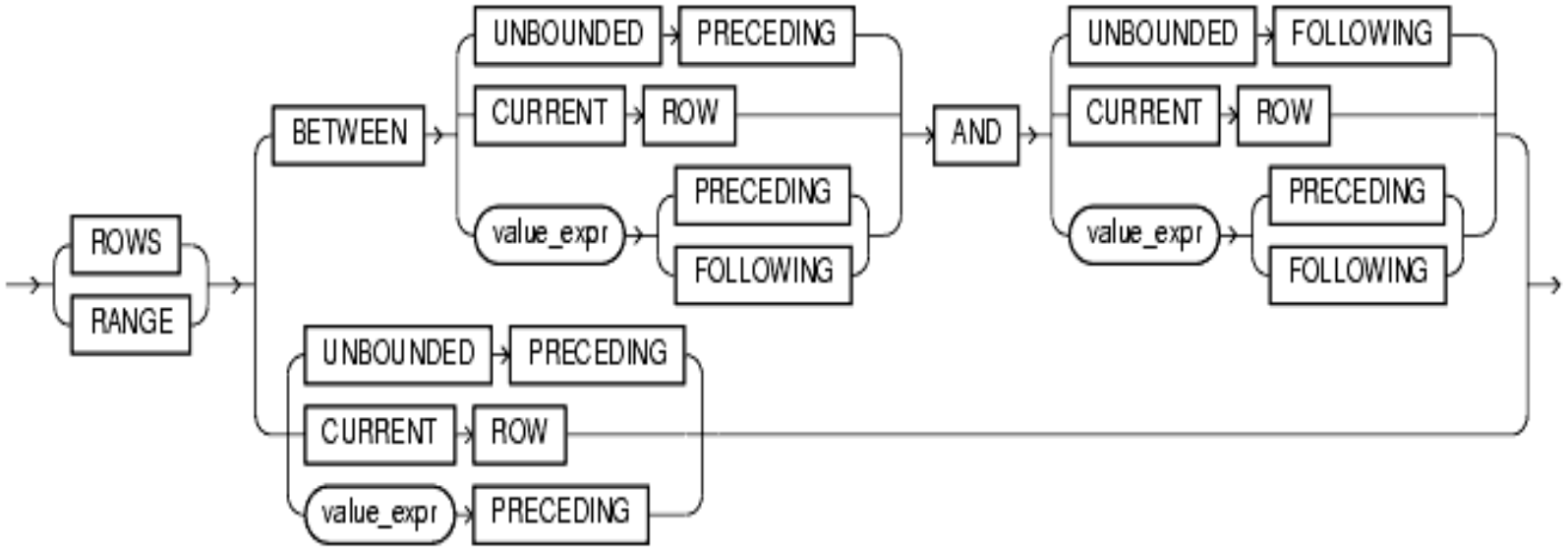
	DIVISION_ID	LAST_NAME	SALARY	RANK	PERCENT_RANK	CUM_DIST
1	10	JOPLIN	4400	1	0	1
2	20	COANDA	13000	1	0	1
3	30	GURION	11000	1	0	0.25
4	30	KHAN	3100	2	0.33	0.5
5	30	PELE	2900	3	0.67	0.75
6	30	EUSEBIO	2800	4	1	1
7	40	BARTÓK	6500	1	0	1
8	50	BELL	4000	1	0	0.17
9	50	NEWTON	3900	2	0.2	0.33
10	50	PUSKIN	3000	3	0.4	0.5
11	50	FERMI	2800	4	0.6	0.67
12	50	MICHELANGELO	2600	5	0.8	0.83
13	50	REMBRANDT	2500	6	1	1

6

Az ablak technikát alkalmazó függvények

Az analitikus függvények szintakszisa II.

windowing_clause::=



6

Változó méretű ablakok képzése és használata

A halmozott összesítések logikája (a partíció első 3 sora)

A halmozott összesítések logikája (a partíció első 4 sora)

A halmozott összesítések logikája (a partíció első 5 sora)

Növekvő méretű ablak
Halmozott összesítés (SUM)

Alapértelmezés:

Az aktuális sor(CURRENT ROW)

```
SELECT worker_id, last_name, salary,
SUM(salary) OVER (ORDER BY salary desc
 RANGE UNBOUNDED PRECEDING) cum_sum,
SUM(salary) OVER (ORDER BY salary desc
RANGE BETWEEN UNBOUNDED PRECEDING AND   CURRENT ROW)  same
FROM workers;
```

	WORKER_ID	LAST_NAME	SALARY	CUM_SUM	SAME
1	100	GAUSS	24000	24000	24000
2	101	EULER	17000	58000	58000
3	102	BERNOULLI	17000	58000	58000
4	145	RUSSELL	14000	72000	72000
5	201	COANDA	13000	85000	85000
6	205	BERING	12000	121000	121000
7	147	NERUDA	12000	121000	121000

RANGE vagy ROWS?
Az ablak definiálásának módjai

ROWS fizikailag

RANGE logikailag

```
SELECT employee_id, last_name, salary,
SUM(salary) OVER (ORDER BY salary desc
 RANGE UNBOUNDED PRECEDING) cum_range,
SUM(salary) OVER (ORDER BY salary desc
  ROWS UNBOUNDED PRECEDING)  CUM_ROWS
          FROM employees;
```

	WORKER_ID	LAST_NAME	SALARY	CUM_RANGE	CUM_ROWS
1	100	GAUSS	24000	24000	24000
2	101	EULER	17000	58000	41000
3	102	BERNOULLI	17000	58000	58000
4	145	RUSSELL	14000	72000	72000
5	201	COANDA	13000	85000	85000
6	205	BERING	12000	121000	97000
7	147	NERUDA	12000	121000	109000
8	108	GRIEG	12000	121000	121000
9	168	FEUERSTEIN	11500	132500	132500

Halmozott összesítés particiónként

A példában a divízió kódja a partíció képzés alapja

```
SELECT division_id, last_name, salary,
SUM(salary) OVER(PARTITION BY division_id ORDER BY salary desc
RANGE UNBOUNDED PRECEDING) CUM_RANGE,
SUM(salary) OVER(PARTITION BY division_id ORDER BY salary desc
ROWS UNBOUNDED PRECEDING)  CUM_ROWS
FROM workers;
```

	DIVISION_ID	LAST_NAME	SALARY	CUM_RANGE	CUM_ROWS
1	10	JOPLIN	4400	4400	4400
2	20	COANDA	13000	13000	13000
3	30	GURION	11000	11000	11000
4	30	KHAN	3100	14100	14100
5	30	PELE	2900	17000	17000
6	30	EUSEBIO	2800	19800	19800
7	40	BARTÓK	6500	6500	6500

Százalékos adatok képzése (RATIO_TO_REPORT)

```
SELECT worker_id, last_name, salary,
TO_CHAR(RATIO_TO_REPORT(salary) OVER() ,'0.999') AS rate
FROM workers ORDER BY salary DESC;
```

	WORKER_ID	LAST_NAME	SALARY	RATE
1	100	GAUSS	24000	0.061
2	101	EULER	17000	0.043
3	102	BERNOULLI	17000	0.043
4	145	RUSSELL	14000	0.036
5	201	COANDA	13000	0.033

```
SELECT division_id, last_name, salary,
TO_CHAR(RATIO_TO_REPORT(salary) OVER (PARTITION BY
division_id),'0.999') AS rr
FROM workers GROUP BY division_id, last_name,salary;
```

	DIVISION_ID	LAST_NAME	SALARY	RR
1	10	JOPLIN	4400	1.000
2	20	COANDA	13000	1.000
3	30	EUSEBIO	2800	0.141
4	30	GURION	11000	0.556
5	30	KHAN	3100	0.157
6	30	PELE	2900	0.146
7	40	BARTÓK	6500	1.000

Százalékos adatok halmozott összesítésekre (előkészület)

```
SELECT worker_id, last_name, salary,
TO_CHAR(RATIO_TO_REPORT(salary) OVER (),'0.999') AS rr,
SUM(salary) OVER (ORDER BY salary desc
RANGE UNBOUNDED PRECEDING) cum_sum,
sum(salary) OVER
( ORDER BY salary DESC  ROWS  BETWEEN  UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS total_sum
FROM workers;
```

	WORKER_ID	LAST_NAME	SALARY	RR	CUM_SUM	TOTAL_SUM
1	100	GAUSS	24000	0.061	24000	393200
2	101	EULER	17000	0.043	58000	393200
3	102	BERNOULLI	17000	0.043	58000	393200
4	145	RUSSELL	14000	0.036	72000	393200
5	201	COANDA	13000	0.033	85000	393200
6	205	BERING	12000	0.031	121000	393200
7	147	NERUDA	12000	0.031	121000	393200
8	108	GRIEG	12000	0.031	121000	393200
9	168	FEUERSTEIN	11500	0.029	132500	393200

Százalékos adatok halmozott összesítésekre inline nézet segítségével

```
SELECT worker_id,last_name,salary,rr,cum_sum,
TO_CHAR(cum_sum /total_sum,'0.999') rate_for_cum_sum FROM
(SELECT worker_id, last_name, salary,
to_char(RATIO_TO_REPORT(salary) OVER (),'0.999') AS rr,
SUM(salary) OVER (ORDER BY salary desc
ROWS UNBOUNDED PRECEDING) cum_sum,
sum(salary) OVER
( ORDER BY salary DESC  ROWS  BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED   FOLLOWING) AS total_sum FROM workers) t;
```

	WORKER_ID	LAST_NAME	SALARY	RR	CUM_SUM	RATE_FOR_CUM_SUM
1	100	GAUSS	24000	0.061	24000	0.061
2	101	EULER	17000	0.043	41000	0.104
3	102	BERNOULLI	17000	0.043	58000	0.148
4	145	RUSSELL	14000	0.036	72000	0.183
5	201	COANDA	13000	0.033	85000	0.216
6	205	BERING	12000	0.031	97000	0.247
7	147	NERUDA	12000	0.031	109000	0.277
8	108	GRIEG	12000	0.031	121000	0.308
9	168	FEUERSTEIN	11500	0.029	132500	0.337
10	174	ABEL	11000	0.028	143500	0.365
11	114	GURION	11000	0.028	154500	0.393

6

Fix méretű ablakok képzése és használata

Mozgó összegzés
Ablak méret=3

Az aktuális sor,az előtte és utána lévő 3 sor, és az aktuális belépési dátum 6 hónapos ablakkal.

```
SELECT last_name, start_date, salary,
SUM(salary)  OVER (ORDER BY start_date
RANGE BETWEEN NUMTOYMINTERVAL(3,'MONTH')
PRECEDING AND numtoyminterval(3,'MONTH') FOLLOWING) AS
mov_sum_3_months,
SUM(salary)  OVER (ORDER BY start_date
ROWS BETWEEN 3 PRECEDING AND 3 FOLLOWING) AS
mov_sum_3_rows FROM workers;
```

	LAST_NAME	START_DATE	SALARY	MOV_SUM_3_MONTHS	MOV_SUM_3_ROWS
1	GAUSS	17-JUN-87	24000	28400	54400
2	JOPLIN	17-JUN-87	4400	28400	60400
3	EULER	21-SEP-89	17000	17000	77400
4	BERNOULLI	03-JAN-90	9000	9000	88400
5	WILLIS	21-MAY-91	6000	6000	67500
6	BERNOULLI	13-JAN-93	17000	17000	66700
7	GURION	07-DEC-94	11000	11000	53700
8	KHAN	18-MAY-95	3100	6700	57700
9	CRUYFF	14-JUL-95	3600	6700	62700
10	BELL	04-FEB-96	4000	17000	53700

Mozgó átlag és mozgó maximum értékesítési adatokra

```
SELECT t.calendar_quarter_desc,p.prod_category,
       sum(s.amount_sold) sold_amount,
ROUND(AVG(SUM(s.amount_sold)) OVER( ORDER BY
      t.calendar_quarter_desc,p.prod_category
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING),2)  mov_avg,
       MAX(SUM(s.amount_sold)) OVER(ORDER BY
      t.calendar_quarter_desc,p.prod_category
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS  mov_max
      FROM sh.sales s, sh.times t ,sh.products p
   WHERE s.time_id=t.time_id AND s.prod_id=p.prod_id
   GROUP BY t.calendar_quarter_desc,p.prod_category;
```

	CALENDAR_QUARTER_DESC	PROD_CATEGORY	SOLD_AMOUNT	MOV_AVG	MOV_MAX
1	1998-01	Electronics	466055.42	1195819.13	1925582.84
2	1998-01	Hardware	1925582.84	1564859.55	2302940.38
3	1998-01	Peripherals and Accessories	2302940.38	1735652.09	2302940.38
4	1998-01	Photo	978433.04	1363015.25	2302940.38
5	1998-01	Software/Other	807672.32	754970.86	978433.04
6	1998-02	Electronics	478807.21	1044227.53	1846203.07
7	1998-02	Hardware	1846203.07	1477612.2	2107826.31
8	1998-02	Peripherals and Accessories	2107826.31	1551089.56	2107826.31
9	1998-02	Photo	699239.29	1089661.29	2107826.31
10	1998-02	Software/Other	461918.26	534957.88	699239.29
11	1998-03	Electronics	443716.09	776056.24	1422534.37

FIRST_VALUE és a LAST_VALUE (Ablak = Particó)

```
SELECT division_id, last_name, salary,
FIRST_VALUE(salary) OVER(
PARTITION BY division_id ORDER BY salary
ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lowest_sal,
LAST_VALUE(salary)OVER(
PARTITION BY division_id ORDER BY salary
ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED   FOLLOWING) AS
highest_sal
FROM workers ORDER BY division_id;
```

	DIVISION_ID	LAST_NAME	SALARY	LOWEST_SAL	HIGHEST_SAL
1	10	JOPLIN	4400	4400	4400
2	20	COANDA	13000	13000	13000
3	30	EUSEBIO	2800	2800	11000
4	30	PELE	2900	2800	11000
5	30	KHAN	3100	2800	11000
6	30	GURION	11000	2800	11000
7	40	BARTÓK	6500	6500	6500

A LAG / LEAD család
Számolás előző és követő sorokkal belépési dátumok alapján

```
SELECT t.*,round(t.elozo_fiz/t.salary,2) "Előző/aktuális",
ROUND(t.koveto_fiz/t.salary,2) "Kovetkezo/aktuális " FROM
(SELECT e.worker_id,e.start_date,e.salary,
LAG(e.SALARY,1) OVER (ORDER BY E.start_date) AS  elozo_fiz,
LEAD(e.SALARY,1) OVER (ORDER BY e.start_date) AS
koveto_fiz
FROM workers e) t;
```

	WORKER_ID	START_DATE	SALARY	ELOZO_FIZ	KOVETO_FIZ	Előző/aktuális	Kovetkezo/aktuális
1	100	17-JUN-87	24000		4400		0.18
2	200	17-JUN-87	4400	24000	17000	5.45	3.86
3	101	21-SEP-89	17000	4400	9000	0.26	0.53
4	103	03-JAN-90	9000	17000	6000	1.89	0.67
5	104	21-MAY-91	6000	9000	17000	1.5	2.83
6	102	13-JAN-93	17000	6000	11000	0.35	0.65
7	114	07-DEC-94	11000	17000	3100	1.55	0.28
8	115	18-MAY-95	3100	11000	3600	3.55	1.16
9	137	14-JUL-95	3600	3100	4000	0.86	1.11
10	192	04-FEB-96	4000	3600	13000	0.9	3.25
11	201	17-FEB-96	13000	4000	11000	0.31	0.85
12	174	11-MAY-96	11000	13000	8000	1.18	0.73

A LAG / LEAD család
Aktuális, 1 évvel korábbi és 1 évvel későbbi értékesítési adatok

```
SELECT t.calendar_month_desc
months,SUM(s.quantity_sold),LAG(SUM(s.quantity_sold),12)
  OVER (ORDER BY t.calendar_month_DESC )  year_before,
          lead(sum(s.quantity_sold),12)
  OVER ( ORDER BY t.calendar_month_desc )  year_after
          FROM sh.sales s, sh.times t
WHERE s.time_id=t.time_id GROUP BY t.calendar_month_desc;
```

	MONTHS	SUM(S.QUANTITY_SOLD)	YEAR_BEFORE	YEAR_AFTER
1	1998-01	15132		20437
2	1998-02	14307		24122
3	1998-03	14248		19627
4	1998-04	11818		17004
5	1998-05	12309		19213
6	1998-06	11631		18016
7	1998-07	16257		21889
8	1998-08	17199		22225
9	1998-09	17059		23024
10	1998-10	19878		22256
11	1998-11	14612		21259
12	1998-12	14384		18873
13	1999-01	20437	15132	22135
14	1999-02	24122	14307	20609

LAG, LEAD + halmozás

```
SELECT t.calendar_month_desc,
MAX(t.calendar_month_NAME) MONTH_NAME ,sum(s.quantity_sold)
quantity_sum,
LAG(SUM(S.QUANTITY_SOLD),12) OVER (ORDER BY t.calendar_month_DESC )
AS  YEAR_BEFORE,
LeaD(SUM(S.QUANTITY_SOLD),12) OVER (ORDER BY t.calendar_month_DESC )
AS  YEAR_AFTER,
SUM( SUM (s.quantity_sold)) OVER (PARTITION BY t.calendar_year
ORDER BY t.calendar_month_DESC RANGE UNBOUNDED PRECEDING ) cum_sum
FROM sh.sales s, sh.times t
WHERE s.time_id=t.time_id
GROUP BY t.calendar_year,t.calendar_month_desc;
```

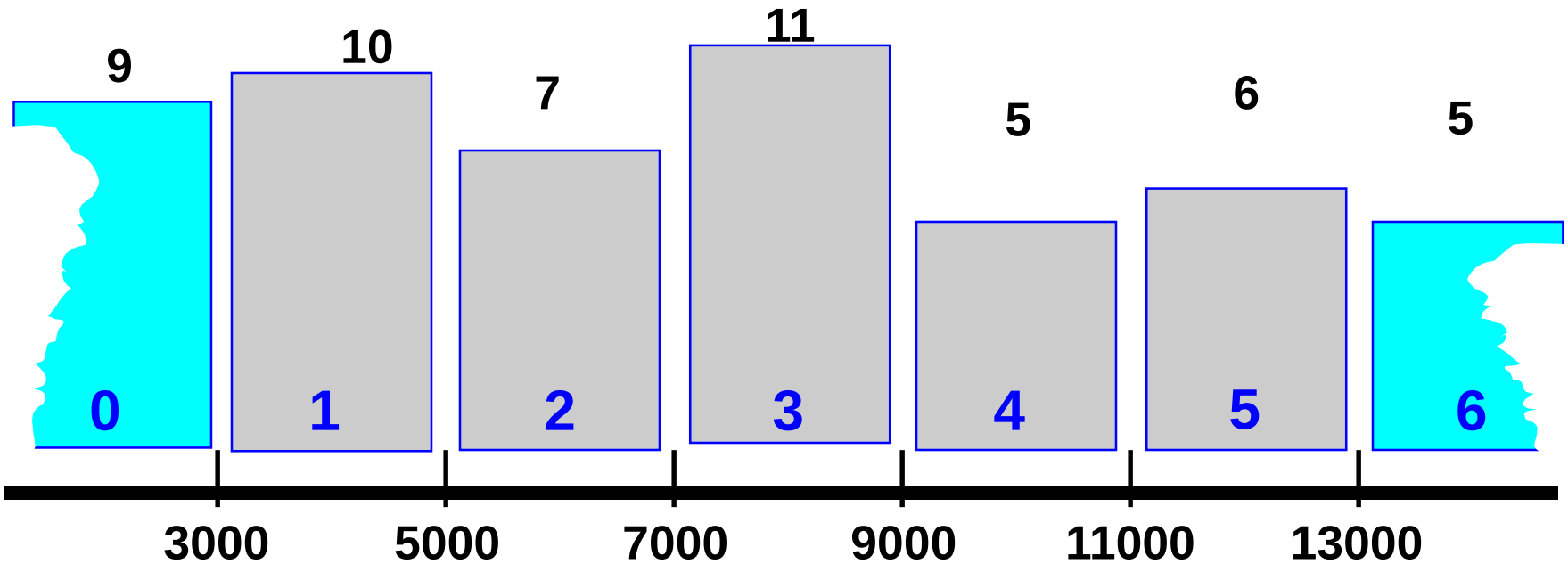
	CALENDAR_MONTH_DESC	MONTH_NAME	QUANTITY_SUM	YEAR_BEFORE	YEAR_AFTER	CUM_SUM
1	1998-01	January	15132		20437	15132
2	1998-02	February	14307		24122	29439
3	1998-03	March	14248		19627	43687
4	1998-04	April	11818		17004	55505
5	1998-05	May	12309		19213	67814
6	1998-06	June	11631		18016	79445
7	1998-07	July	16257		21889	95702
8	1998-08	August	17199		22225	112901
9	1998-09	September	17059		23024	129960
10	1998-10	October	19878		22256	149838
11	1998-11	November	14612		21259	164450
12	1998-12	December	14384		18873	178834
13	1999-01	January	20437	15132	22135	20437

6

Hisztogramok, Kvartilisek

Hisztogram készítése
A WIDTH_BUCKET függvény

```
SELECT last_name, salary,
width_bucket(salary,3000,13000,5)
FROM  workers ORDER BY 3;
```



Kvartilis
Az NTILE függvény

Az eredmény halmaz sorait N (közel) egyenlő részre osztja statisztikai rangjuk alapján, melyeket sorszámmal lát el.


```
SELECT last_name, salary,
       NTILE(7)
OVER (ORDER BY salary ) quartile
FROM workers;
```

	LAST_NAME	SALARY	QUARTILE
1	SEAGAL	2200	1
2	REMBRANDT	2500	1
3	NKOMO	2500	1
4	MICHELANGELO	2600	1
5	GANDHI	2600	1
6	TOLSZTOJ	2700	1
7	EUSEBIO	2800	1
8	FERMI	2800	1
9	PELE	2900	2
10	PUSKIN	3000	2
11	KHAN	3100	2
12	ROBERTS	3200	2
13	STILES	3200	2
14	CRUYFF	3600	2
15	NEWTON	3900	2
16	BELL	4000	2

Kvartilisek a gyakorlatban

```
SELECT quartile,sum(salary),round(avg(salary)) avg,
       MIN(salary),MAX(salary),count(*)
  FROM(SELECT last_name, salary, ntile(7)
OVER (ORDER BY salary ) quartile FROM workers)
     GROUP BY quartile ORDER BY quartile;
```

	QUARTILE	SUM(SALARY)	AVG	MIN(SALARY)	MAX(SALARY)	COUNT(*)
1	1	20700	2588	2200	2800	8
2	2	26900	3363	2900	4000	8
3	3	43600	5450	4200	6200	8
4	4	58100	7263	6500	8000	8
5	5	59400	8486	8000	9000	7
6	6	75500	10786	9500	12000	7
7	7	109000	15571	12000	24000	7

Hisztogramok a gyakorlatban

```
SELECT buckets,sum(salary),round(avg(salary)) avg,
       min(salary),max(salary), count(*)
  FROM (SELECT last_name, salary,
WIDTH_BUCKET(salary,3000,13000,5) AS buckets FROM
                                    workers)
 GROUP BY buckets
 ORDER BY buckets;
```

	BUCKETS	SUM(SALARY)	AVG	MIN(SALARY)	MAX(SALARY)	COUNT(*)
1	0	23600	2622	2200	2900	9
2	1	37400	3740	3000	4800	10
3	2	43200	6171	5800	6500	7
4	3	86500	7864	7000	8600	11
5	4	48000	9600	9000	10500	5
6	5	69500	11583	11000	12000	6
7	6	85000	17000	13000	24000	5

Kvartilisek az eladási adatokra

```
SELECT sorszam,round(avg(amount)) avg,sum(amount)
        FROM(SELECT NTILE(9) OVER(ORDER BY
            SUM(s.AMOUNT_sold)desc)sorszam,
        P.PROD_NAME,round(SUM(s.AMOUNT_sold))amount
         FROM sh.sales s, sh.times t,sh.PRODUCTS P
WHERE s.time_id=t.time_id AND s.prod_id=p.prod_id GROUP BY
                                p.prod_name)
                        GROUP BY sorszam
                        ORDER BY sorszam;
```

SORSZAM	AVG	SUM(AMOUNT)
1	7312497	58499973
2	2077070	16616556
3	887169	7097352
4	597760	4782083
5	515256	4122046
6	364988	2919907
7	278280	2226238
8	180395	1443163
9	71216	498515

A LISTAGG függvény analitikus változata

```
SELECT division_id "Dept", start_date "Date", last_name
                                "Name",
            LISTAGG(last_name, '; ')
      WITHIN GROUP (ORDER By division_id, last_name)
            OVER (PARTITION BY division_id) emp
      FROM workers WHERE division_id NOT IN (50,80);
```

	Dept	Date	Name	EMP
1	10	17-JUN-87	JOPLIN	JOPLIN
2	20	17-FEB-96	COANDA	COANDA
3	30	24-JUL-97	EUSEBIO	EUSEBIO; GURION; KHAN; PELE
4	30	07-DEC-94	GURION	EUSEBIO; GURION; KHAN; PELE
5	30	18-MAY-95	KHAN	EUSEBIO; GURION; KHAN; PELE
6	30	24-DEC-97	PELE	EUSEBIO; GURION; KHAN; PELE
7	40	07-JUN-99	BARTÓK	BARTÓK
8	60	03-JAN-90	BERNOULLI	BERNOULLI; LORENTZ; VERDI; WILLIS
9	60	07-FEB-99	LORENTZ	BERNOULLI; LORENTZ; VERDI; WILLIS
10	60	05-FEB-98	VERDI	BERNOULLI; LORENTZ; VERDI; WILLIS
11	60	21-MAY-91	WILLIS	BERNOULLI; LORENTZ; VERDI; WILLIS