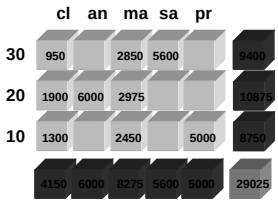
Hurtownie danych - przegląd technologii

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Funkcje analityczne SQL – CUBE (1)

JOB	DEPTNO	SUM(SAL)
		29025
	10	8750
	20	10875
	30	9400
CLERK		4150
CLERK	10	1300
CLERK	20	1900
CLERK	30	950
ANALYST		6000
ANALYST	20	6000
MANAGER		8275
MANAGER	10	2450
MANAGER	20	2975
MANAGER	30	2850
SALESMAN		5600
SALESMAN	30	5600
PRESIDENT		5000
PRESIDENT	10	5000

```
select job, deptno, sum(sal)
from emp
group by cube(job, deptno);
```



Funkcje analityczne SQL – CUBE (2)

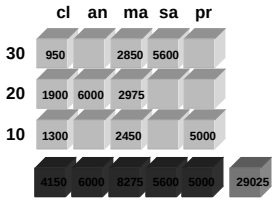
➲ **CUBE (job, deptno, mgr) jest równoważne:**

```
select ... group by job
union all
select ... group by deptno
union all
select ... group by mgr
union all
select ... group by job, deptno
union all
select ... group by job, mgr
union all
select ... group by deptno, mgr
union all
select ... group by job, deptno, mgr
union all
select ..."total";
```


Funkcje analityczne SQL – ROLLUP (1)

JOB	DEPTNO	SUM(SAL)
CLERK	10	1300
CLERK	20	1900
CLERK	30	950
CLERK		4150
ANALYST	20	6000
ANALYST		6000
MANAGER	10	2450
MANAGER	20	2975
MANAGER	30	2850
MANAGER		8275
SALESMAN	30	5600
SALESMAN		5600
PRESIDENT	10	5000
PRESIDENT		5000
		29025

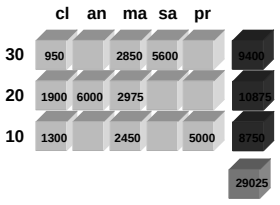
```
select job, deptno, sum(sal)
from emp
group by ROLLUP(job, deptno);
```



Funkcje analityczne SQL – ROLLUP (2)

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10		8750
20	CLERK	1900
20	ANALYST	6000
20	MANAGER	2975
20		10875
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
30		9400
		290251

```
select deptno, job, sum(sal)
from emp
group by ROLLUP(deptno, job);
```



Funkcje analityczne SQL – ROLLUP (3)

➲ **ROLLUP (job, deptno, mgr) jest równoważne:**

```
select ... group by job
union all
select ... group by job, deptno
union all
select ... group by job, deptno, mgr
union all
select ..."total";
```

Funkcje analityczne SQL – GROUPING SETS (1)

```
 JOB       DEPTNO    MGR   SUM(SAL)
 --------- ------ ----- ----------
CLERK          10              1300
CLERK          20              1900
CLERK          30               950
ANALYST        20              6000
MANAGER        10              2450
MANAGER        20              2975
MANAGER        30              2850
SALESMAN       30              5600
PRESIDENT      10              5000
CLERK                7698       950
CLERK                7782      1300
CLERK                7788      1100
CLERK                7902       800
ANALYST              7566      6000
MANAGER              7839      8275
SALESMAN             7698      5600
PRESIDENT                      5000
                              29025
```

```
select job, deptno, mgr, sum(sal)
from emp
group by GROUPING SETS
((job, deptno), (job, mgr), ());
```

group by job, deptno

group by job, mgr

Funkcje analityczne SQL – GROUPING SETS (2)

➲ **GROUPING SETS (job, deptno, mgr) jest równoważne:**

```
select ... group by job
union all
select ... group by deptno
union all
select ... group by mgr;
```

➲ **GROUPING SETS (job, deptno, (deptno, mgr)) jest równoważne:**

```
select ... group by job
union all
select ... group by deptno
union all
select ... group by deptno, mgr;
```


Funkcje analityczne SQL – GROUPING SETS (3)

➲ **CUBE (job, deptno, mgr) jest równoważne:**

```
GROUPING SETS ( (job, deptno, mgr),
                (job, deptno),
                (job, mgr),
                (deptno, mgr),
                (job),
                (deptno),
                (mgr),
                () )
```

➲ **ROLLUP (job, deptno, mgr) jest równoważne:**

```
GROUPING SETS ( (job, deptno, mgr),
                (job, deptno),
                (job),
                () )
```


Funkcje analityczne SQL – GROUPING

```
select job, deptno, sum(sal), GROUPING(job), GROUPING(deptno)
from emp
group by ROLLUP(job, deptno);
```

```
 JOB       DEPTNO   SUM(SAL) GROUPING(JOB) GROUPING(DEPTNO)
 --------- ------ ---------- ------------- ----------------
CLERK          10       1300             0                0
CLERK          20       1900             0                0
CLERK          30        950             0                0
CLERK                   4150             0                1
ANALYST        20       6000             0                0
ANALYST                 6000             0                1
MANAGER        10       2450             0                0
MANAGER        20       2975             0                0
MANAGER        30       2850             0                0
MANAGER                 8275             0                1
SALESMAN       30       5600             0                0
SALESMAN                5600             0                1
PRESIDENT      10       5000             0                0
PRESIDENT               5000             0                1
                       29025             1                1
```


Funkcje analityczne SQL – RANK

➲ **wyznaczanie rankingu**

```
select *
from
    (select d.dname, sum(e.sal+nvl(comm, 0)) as total_sal,
     RANK() over (order by sum(e.sal+nvl(comm, 0)) desc nulls last) as rank,
     DENSE_RANK() over (order by sum(e.sal+nvl(comm, 0)) desc nulls last) as drank
     from dept d, emp e
     where d.deptno=e.deptno
     group by d.dname)
where rank<=4;
```

```
DNAME          TOTAL_SAL       RANK      DRANK
-------------- ---------- ---------- ----------
HUMAN RES.         11600          1          1
SALES              11600          1          1
RESEARCH           10875          3          2
ACCOUNTING          8750          4          3
```

kolejność sortowania wartości NULL
-domyślnie NULLS LAST
-możliwe NULLS FIRST

Funkcje analityczne SQL – RANK PARTITION BY (1)

➲ **wyznaczanie rankingu**

```
select *
from
    (select ename, deptno, sal,
     RANK() over (PARTITION BY deptno
                  order by sal desc nulls last) as rank
     from emp)
where rank<=2;
```

```
ENAME      DEPTNO        SAL       RANK
---------- ------ ---------- ----------
KING           10       5000          1
CLARK          10       2450          2
SCOTT          20       3000          1
FORD           20       3000          1
BLAKE          30       2850          1
ALLEN          30       1600          2
```

Funkcje analityczne SQL – RANK PARTITION BY (2)

- wyznaczanie rankingu podzbiorach

```
select *
from
    (select deptno, job, sum(sal),
     RANK() over (PARTITION BY deptno
                  order by sum(sal) desc nulls last) as rank
     from emp
     group by deptno, job)
where rank<=2;
```

DEPTNO	JOB	SUM(SAL)	RANK
10	PRESIDENT	5000	1
10	MANAGER	2450	2
20	ANALYST	6000	1
20	MANAGER	2975	2
30	SALESMAN	5600	1
30	MANAGER	2850	2

Funkcje analityczne SQL – RANK what if analysis

- wyznaczanie hipotetycznego rankingu pracownika o zarobkach 4000

```
select RANK(4000) WITHIN GROUP (order by sal desc) as hrank
from emp;
```

- pensja 4000 byłaby drugą co do wielkości

HRANK
2

Funkcje analityczne SQL – NTILE

- dzieli uporządkowany zbiór wynikowy na n podzbiorów
  - każdy podzbiór otrzymuje numer
  - liczba rekordów w podzbiorach różni się maksymalnie o 1
  - zakres wartości minimalnej i maksymalnej w tym samym podzbiorze może być szeroki

```
select ename, sal,
       NTILE(2) over(order by sal desc) as "ntile(2)"
from emp
where deptno=20;
```

ENAME	SAL	ntile(2)
SCOTT	3000	1
FORD	3000	1
JONES	2975	1
ADAMS	1100	2
SMITH	800	2

Funkcje analityczne SQL – WIDTH_BUCKET

- dzieli uporządkowany zbiór wynikowy na n podzbiorów;
  - zbiór wynikowy zawiera rekordy z zadanego przedziału
  - liczba rekordów w podzbiorach może się różnić znacząco
  - zakres wartości minimalnej i maksymalnej w tym samym podzbiorze powinien być niewielki

ENAME	SAL	WB
SMITH	800	0
JAMES	950	0
ALLEN	1600	1
WARD	1250	1
ADAMS	1100	1
TURNER	1500	1
MARTIN	1250	1
MILLER	1300	1
JONES	2975	2
BLAKE	2850	2
CLARK	2450	2
SCOTT	3000	3
FORD	3000	3
KING	5000	5

wartości < 1000

wartości >= 5000

- zakres dziedziny <1000, 5000) jest dzielony na 4 równej szerokości przedziały: <1000, 2000), <2000, 3000), <3000, 4000), <4000, 5000)
- przedział <4000, 5000) nie ma rekordów

```
select ename, sal,
       WIDTH_BUCKET(sal, 1000, 5000, 4) as WB
from emp
order by WB;
```

zakres wartości <1000, 5000)

Funkcje analityczne SQL – suma kumulacyjna (1)

```
select deptno, sum(sal),
       sum(sum(sal))
         over (order by deptno rows unbounded preceding)
                                              as cum_sum
from emp
group by deptno;
```

DEPTNO	SUM(SAL)	CUM_SUM
10	8750	8750
20	10875	19625
30	9400	29025
60	11600	40625

suma kumulacyjna jest obliczana z wykorzystaniem bieżącego rekordu i wszystkich rekordów go poprzedzających

Funkcje analityczne SQL – suma kumulacyjna (2)

```
select deptno, sum(sal),
       sum(sum(sal))
         over (order by deptno rows 1 preceding) as cum_sum
from emp
group by deptno;
```

DEPTNO	SUM(SAL)	CUM_SUM
10	8750	8750
20	10875	19625
30	9400	20275
60	11600	21000

suma kumulacyjna jest obliczana z wykorzystaniem bieżącego rekordu i 1 rekordu go poprzedzającego

Funkcje analityczne SQL – suma kumulacyjna (3)

```
select deptno, sum(sal),
       sum(sum(sal))
         over (order by deptno
               rows between 1 preceding and 1 following)
                                              as cum_sum
from emp
group by deptno;
```

DEPTNO	SUM(SAL)	CUM_SUM
10	8750	19625
20	10875	29025
30	9400	31875
60	11600	21000

19635=8750+10875

29025=8750+10875+9400

Funkcje analityczne SQL – suma kumulacyjna (4)

```
SELECT t.time_key, SUM(f.purchase_price) as sales,
       SUM(SUM(f.purchase_price))
            OVER (ORDER BY t.time_key
                  RANGE BETWEEN INTERVAL '2' DAY PRECEDING AND
                  INTERVAL '2' DAY FOLLOWING) as sales_5_day
FROM purchases f, time t
WHERE f.time_key = t.time_key
GROUP BY t.time_key;
```

jest możliwe określanie interwału z wykorzystaniem słów kluczowych MONTH lub YEAR

TIME_KEY	SALES	SALES_5_DAY
01-JAN-1999	56,02	239,04
02-JAN-1999	183,02	239,04
01-FEB-1999	122	186
02-FEB-1999	42	198
03-FEB-1999	22	224
04-FEB-1999	12	102
05-FEB-1999	26	60
01-MAR-1999	42	84
02-MAR-1999	42	84
01-APR-1999	42	42

239,04=56,02+183,02

84=42+42

Funkcje analityczne SQL – max (1)

```
select job, deptno, sum(sal) as job_sal,
       max(sum(sal)) over (partition by job) as max_job_sal
from emp
group by job, deptno;
```

JOB	DEPTNO	JOB_SAL	MAX_JOB_SAL
ANALYST	20	6000	6000
CLERK	10	1300	1900
CLERK	20	1900	1900
CLERK	30	950	1900
MANAGER	10	2450	2975
MANAGER	20	2975	2975
MANAGER	30	2850	2975
PRESIDENT	10	5000	5000
SALESMAN	30	5600	5600

pensje na poszczególnych etatach w poszczególnych zespołach

maksymalne pensje na etatach

Funkcje analityczne SQL – max (2)

```
select job, deptno, job_sal
from (select job, deptno, sum(sal) as job_sal,
             max(sum(sal)) over (partition by job)
                                        as max_job_sal
      from emp
      group by job, deptno)
where job_sal=max_job_sal;
```

JOB	DEPTNO	JOB_SAL
ANALYST	20	6000
CLERK	20	1900
MANAGER	20	2975
PRESIDENT	10	5000
SALESMAN	30	5600

Funkcje analityczne SQL – RATIO_TO_REPORT

```
select job, deptno, sum(sal) as job_sal,
       sum(sum(sal)) over () as total_sal,
       RATIO_TO_REPORT(sum(sal)) OVER() as ratio_to_report
from emp
group by job, deptno;
```

suma pensji dla wszystkich rekordów pracowników

JOB	DEPTNO	JOB_SAL	TOTAL_SAL	RATIO_TO_REPORT
ANALYST	20	6000	29025	,206718346
CLERK	10	1300	29025	,044788975
CLERK	20	1900	29025	,06546081
CLERK	30	950	29025	,032730405
MANAGER	10	2450	29025	,084409991
MANAGER	20	2975	29025	,102497847
MANAGER	30	2850	29025	,098191214
PRESIDENT	10	5000	29025	,172265289
SALESMAN	30	5600	29025	,192937123

RATIO_TO_REPORT = total_sal/sum_sal

Funkcje analityczne SQL – LAG i LEAD

- umożliwiają dostęp do wartości atrybutów rekordów poprzedzających dany rekord (LAG) lub następujących po danym rekordzie (LEAD)
- argument wywołania funkcji określa przesunięcie w tył/przód względem bieżącego rekordu

```
SELECT t.month, SUM(f.purchase_price) as sales,
       LAG(SUM(f.purchase_price),1)
           OVER (ORDER BY t.month) as sales_last_month,
       LEAD(SUM(f.purchase_price),1)
           OVER (ORDER BY t.month) as sales_next_month
FROM purchases f, time t
WHERE f.time_key = t.time_key AND t.year = 1999
GROUP BY t.month;
```

MONTH	SALES	SALES_LAST_MONTH	SALES_NEXT_MONTH
1	239,04		200
2	200	239,04	84
3	84	200	42
4	42	84	

NULL – brak rekordu poprzedzającego

NULL – brak rekordu następującego

Schemat magazynu danych do ćwiczeń