

Implementacja przetwarzania analizy przy wykorzystaniu funkcji SQL Oracle9i OLAP

Maciej Zakrzewicz
PLOUG

Plan prezentacji

- Wprowadzenie do Oracle9i OLAP – przetwarzanie analityczne danych relacyjnych
- Rozszerzone operatory agregujące SQL
- Nowe funkcje analityczne SQL
- Podsumowanie

Oracle9i OLAP

- Funkcjonalność OLAP wprowadzona do systemu zarządzania bazą danych
- Główne cechy Oracle9i OLAP
 - Usługa OLAP Services, integrująca serwer Express z serwerem Oracle
 - Interfejs programistyczny Java OLAP API
 - Centralne repozytorium metadanych OLAP
 - Analityczne rozszerzenia języka zapytań SQL

Operatory grupujące

Tabela Sprzedaż

Region	Produkt	Kwota
...



Wynik zapytania

	<i>Oleje</i>	<i>Opony</i>	<i>Reflektory</i>	<i>Tłumiki</i>	razem
<i>Mazowsze</i>	23000	97000	97000	1150000	1367000
<i>Wielkopolska</i>	230000	152000	48000	230000	660000
razem	253000	249000	145000	1380000	2027000

```
select region, produkt, sum(kwota)
from sprzedaz
group by region, produkt
```

```
select sum(kwota)
from sprzedaz
```

```
select produkt, sum(kwota)
from sprzedaz
group by produkt
```

```
select region, sum(kwota)
from sprzedaz
group by region
```

Rozszerzenia SQL w Oracle 8i/9i

- Wprowadzono nowe operatory grupujące, umożliwiające wykonanie wielopoziomowego grupowania hierarchicznego w pojedynczym przebiegu zapytania:
 - GROUP BY ROLLUP
 - GROUP BY CUBE
 - GROUP BY GROUPING SETS
- Nowe operatory pozwalają usprawnić wydajność aplikacji analitycznych, budujących tabelaryczne wycinki wielowymiarowej kostki danych

Operator ROLLUP

Tabela Sprzedaz

Region	Produkt	Kwota
...



Wynik zapytania

	<i>Oleje</i>	<i>Opony</i>	<i>Reflektory</i>	<i>Tłumiki</i>	razem
<i>Mazowsze</i>	<i>23000</i>	<i>97000</i>	<i>97000</i>	<i>1150000</i>	1367000
<i>Wielkopolska</i>	<i>230000</i>	<i>152000</i>	<i>48000</i>	<i>230000</i>	660000

```
select region, produkt, sum(kwota)
from sprzedaz
group by rollup(region, produkt)
```

Operator CUBE

Tabela Sprzedaż

Region	Produkt	Kwota
...



Wynik zapytania

	<i>Oleje</i>	<i>Opony</i>	<i>Reflektory</i>	<i>Tłumiki</i>	razem
<i>Mazowsze</i>	<i>23000</i>	<i>97000</i>	<i>97000</i>	<i>1150000</i>	1367000
<i>Wielkopolska</i>	<i>230000</i>	<i>152000</i>	<i>48000</i>	<i>230000</i>	660000
razem	253000	249000	145000	1380000	2027000

```
select region, produkt, sum(kwota)
from sprzedaz
group by cube(region, produkt)
```

Operator GROUPING SETS

Tabela Sprzedaż

Region	Produkt	Kwota
...



Wynik zapytania

	<i>Oleje</i>	<i>Opony</i>	<i>Reflektory</i>	<i>Tłumiki</i>	razem
<i>Mazowsze</i>	<i>23000</i>	<i>97000</i>	<i>97000</i>	<i>1150000</i>	1367000
<i>Wielkopolska</i>	<i>230000</i>	<i>152000</i>	<i>48000</i>	<i>230000</i>	660000
razem	253000	249000	145000	1380000	

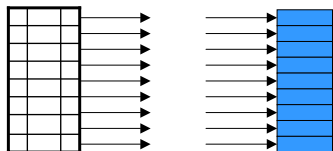
```
select region, produkt, sum(kwota)
from sprzedaz
group by grouping sets
((region, produkt),(region),(produkt))
```

Nowe funkcje analityczne SQL w Oracle8i/9i

- Tradycyjne funkcje grupowe mogą operować na partycjach obliczeniowych lub na ruchomym oknie obliczeniowym
- Nowe funkcje analityczne:
 - RATIO_TO_REPORT()
 - FIRST_VALUE()
 - LAST_VALUE()
 - LAG()
 - LEAD()
 - RANK()
 - DENSE_RANK()
 - CUME_DIST()
 - PERCENT_RANK()
 - NTILE()
 - ROW_NUMBER()

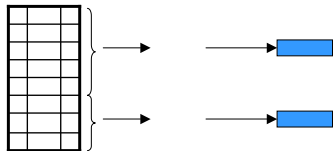
Partycje obliczeniowe

funkcja jednorekordowa



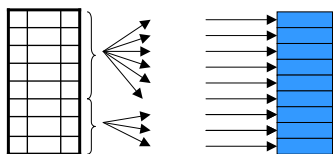
```
select upper(nazwisko)
from pracownicy
```

funkcja grupowa z GROUP BY



```
select avg(pensja)
from pracownicy
group by stanowisko
```

funkcja grupowa z PARTITION



```
select avg(pensja) over
(partition by stanowisko)
from pracownicy
```

Partycje obliczeniowe - przykład

```
select region, produkt, kwota,
       sum(kwota) over (partition by region) sum_kwota,
       round(100*kwota/(sum(kwota) over (partition by region))) "UDZIAŁ%"
from sprzedaz
order by region, produkt
```

REGION	PRODUKT	KWOTA	SUM_KWOTA	UDZIAŁ%
Mazowsze	Oleje	23000	1367000	2
Mazowsze	Opony	97000	1367000	7
Mazowsze	Reflektory	97000	1367000	7
Mazowsze	Tłumiki	1150000	1367000	84
Wielkopolska	Oleje	230000	660000	35
Wielkopolska	Opony	152000	660000	23
Wielkopolska	Reflektory	48000	660000	7
Wielkopolska	Tłumiki	230000	660000	35

Funkcja RATIO_TO_REPORT

```
select region, produkt, kwota,
       sum(kwota) over (partition by region) sum_kwota,
       100*(ratio_to_report(sum(kwota)) over (partition by region)) "UDZIAŁ%"
from sprzedaz
order by region, produkt
```

REGION	PRODUKT	KWOTA	SUM_KWOTA	UDZIAŁ%
Mazowsze	Oleje	23000	1367000	2
Mazowsze	Opony	97000	1367000	7
Mazowsze	Reflektory	97000	1367000	7
Mazowsze	Tłumiki	1150000	1367000	84
Wielkopolska	Oleje	230000	660000	35
Wielkopolska	Opony	152000	660000	23
Wielkopolska	Reflektory	48000	660000	7
Wielkopolska	Tłumiki	230000	660000	35

Ruchome okno obliczeniowe

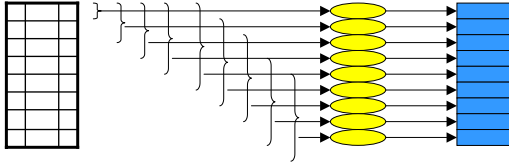


Tabela Sprzedaż

DZIEŃ	KWOTA
09-MAY-02	150
10-MAY-02	130
11-MAY-02	100
12-MAY-02	180
13-MAY-02	200
14-MAY-02	220
15-MAY-02	230
16-MAY-02	210

```
select dzien, avg(kwota) over
  (order by dzien range
    between 3 preceding and current row)
  as avg3
from sprzedaz
```

Wynik zapytania

DZIEŃ	AVG3
09-MAY-02	150.00
10-MAY-02	140.00
11-MAY-02	126.67
12-MAY-02	136.67
13-MAY-02	160.00
14-MAY-02	200.00
15-MAY-02	216.67
16-MAY-02	220.00

Nowe funkcje: FIRST_VALUE() i LAST_VALUE()

Funkcje LAG() i LEAD()

```
select dzien,
  kwota,
  lag(kwota,1) over (order by dzien) as poprz,
  kwota - (lag(kwota,1) over (order by dzien)) as wzrost
from sprzedaz
```

Tabela Sprzedaż

DZIEŃ	KWOTA
09-MAY-02	150
10-MAY-02	130
11-MAY-02	100
12-MAY-02	180
13-MAY-02	200
14-MAY-02	220
15-MAY-02	230
16-MAY-02	210

Wynik zapytania

DZIEŃ	KWOTA	POPRZ	WZROST
09-MAY-02	150		
10-MAY-02	130	150	-20
11-MAY-02	100	130	-30
12-MAY-02	180	100	80
13-MAY-02	200	180	20
14-MAY-02	220	200	20
15-MAY-02	230	220	10
16-MAY-02	210	230	-20

Funkcje rankingowe SQL

- **RANK()** - numer pozycji rankingowej rekordu w ramach grupy rekordów, przy czym wystąpienie jednakowej pozycji rankingowej dla wielu rekordów powoduje powstanie przerw w numeracji
- **DENSE_RANK()** - podobnie jak RANK(), jednak nie powoduje powstawania przerw w numeracji w powyższej sytuacji
- **CUME_DIST()** - wartość pozycji rankingowej wyrażona liczbą z przedziału (0;1> - liczba ta rozumiana jest jako procent rekordów poprzedzających dany rekord w rankingu z uwzględnieniem bieżącego rekordu
- **PERCENT_RANK()** – podobnie jak CUME_DIST() lecz nie uwzględnia bieżącego rekordu w określaniu procentu rekordów poprzedzających
- **NTILE()** - numer kolejnej n-tki (trójki, piątki, dziesiątki, itp.) rankingu, do której należy bieżący rekord; funkcja niedeterministyczna
- **ROW_NUMBER()** - bazuje na rankingu w celu wyznaczenia liczby porządkowej dla każdego rekordu; funkcja niedeterministyczna

Funkcje RANK() i DENSE_RANK()

```
select region, produkt, kwota,
       rank() over (partition by region order by kwota desc) as rank,
       dense_rank() over (partition by region order by kwota desc) as dense_rank
from sprzedaz
order by region, produkt
```

REGION	PRODUKT	KWOTA	RANK	DENSE_RANK
Mazowsze	Oleje	23000	4	3
Mazowsze	Opony	97000	2	2
Mazowsze	Reflektory	97000	2	2
Mazowsze	Tłumiki	1150000	1	1
Wielkopolska	Oleje	230000	1	1
Wielkopolska	Opony	152000	3	2
Wielkopolska	Reflektory	48000	4	3
Wielkopolska	Tłumiki	230000	1	1

Pozostałe funkcje rankingowe

```
select kwota,  
       rank() over (order by kwota desc) as rank,  
       cume_dist() over (order by kwota desc) as cume_dist,  
       percent_rank() over (order by kwota desc) as percent_rank,  
       ntile(3) over (order by kwota desc) as ntile_3,  
       row_number() over (order by kwota desc) as row_number  
from sprzedaz order by kwota
```

KWOTA	RANK	CUME_DIST	PERCENT_RANK	NTILE_3	ROW_NUMBER
23000	8	1	1	3	8
48000	7	,875	,857142857	3	7
97000	5	,75	,571428571	2	5
97000	5	,75	,571428571	2	6
152000	4	,5	,428571429	2	4
230000	2	,375	,142857143	1	2
230000	2	,375	,142857143	1	3
1150000	1	,125	0	1	1

Odwrotne funkcje rankingowe

- Zwracają wartość rekordu, który w rankingu reprezentowany jest przez zadany wynik funkcji CUME_DIST()
 - PERCENTILE_DISC()
 - PERCENTILE_CONT()

Przykład: znajdowanie wartości środkowej

```
select percentile_disc(0.5) within group  
       (order by kwota desc) as percentile  
from sprzedaz  
order by kwota  
  
PERCENTILE  
-----  
152000
```

Funkcje regresji liniowej

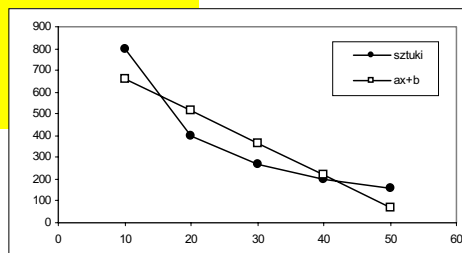
```
select * from sprzedaz_wg_cen;
```

PRODUKT	CENA	SZTUKI
HDD1525SX	50	160
HDD1525SX	40	200
HDD1525SX	30	270
HDD1525SX	20	400
HDD1525SX	10	800

```
select regr_slope(sztuki,cena) as a,
       regr_intercept(sztuki,cena) as b
from sprzedaz_wg_cen
```

A	B
-14,8	810

REGR_COUNT()
 REGR_AVGX()
 REGR_AVGY()
 REGR_SLOPE()
 REGR_INTERCEPT()
 REGR_R2()
 REGR_SXX()
 REGR_SYY()
 REGR_SXY()



Grupowe funkcje prognostyczne

KWOTA	RANK
23000	8
48000	7
97000	5
97000	5
152000	4
230000	2
230000	2
1150000	1

Na której pozycji w obecnym rankingu znalazłaby się dana wartość, gdyby została dołączona do tabeli?

- RANK(x)
- DENSE_RANK(x)
- PERCENT_RANK(x)
- CUME_DIST(x)

```
select rank(100000) within group (order by kwota desc) as rank
from sprzedaz
order by kwota;
```

RANK
5

Podsumowanie

- Rozszerzenia analityczne języka SQL umożliwiają programistom zwiększenie czytelności i wydajności kodu analitycznego aplikacji operujących na magazynie (hurtowni) danych opartym na relacyjnej bazie danych
- Przedstawione rozwiązania są kolejnym krokiem w kierunku integracji funkcjonalności OLAP z systemami zarządzania relacyjnymi bazami danych