

Rozszerzenia grupowania

Plan rozdziału

- Wprowadzenie
- ROLLUP
- CUBE
- GROUPING SETS
- GROUPING

Rozszerzenia grupowania danych

- W złożonych magazynach danych oprócz tabel faktów i wymiarów istnieje dodatkowo wiele perspektyw materializowanych mających na celu zwiększenie wydajności systemu
- W większości przypadków przechowują one dane zagregowane, każda na innym poziomie, przy wykorzystaniu grupowania w oparciu o różne zbiory atrybutów
- Odświeżanie takiego zestawu perspektyw materializowanych jest poważnym obciążeniem dla systemu
- Pomocą mogą w tym służyć rozszerzenia grupowania danych, dzięki którym jedno zapytanie może wyznaczać wiele zbiorów agregacji na różnych poziomach grupowania
- Rozszerzenia te stanowią rozbudowę klauzuli GROUP BY

ROLLUP

- Polecenie ROLLUP jest rozszerzeniem klauzuli GROUP BY, które pozwala wyliczać dodatkowe podsumowania częściowe i ogólne. Polecenie ROLLUP służy do konstruowania pół-kostek danych. ROLLUP jest wyrażeniem wyjątkowo wydajnym. Dla n kolumn grupujących ROLLUP tworzy $n + 1$ podsumowań.
- Dodatkowe podsumowania wyznaczone przez ROLLUP wyliczane są przez eliminowanie kolejno kolumn grupujących począwszy od ostatniej a skończywszy na pierwszej
- Przykładowo klauzula
GROUP BY ROLLUP(kontynent, kraj, miejscowość, dzielnica)
Wyznaczy następujące podsumowania
 - GROUP BY kontynent, kraj, miejscowość, dzielnica
 - GROUP BY kontynent, kraj, miejscowość
 - GROUP BY kontynent, kraj
 - GROUP BY kontynent
 - bez GROUP BY – podsumowanie całkowite

Przykład użycia ROLLUP

```
SELECT nr_konta, typ, kategoria, sum(kwota)
FROM   transakcje
GROUP BY nr_konta, typ, kategoria
```



```
SELECT nr_konta, typ, kategoria, sum(kwota)
FROM   transakcje
GROUP BY rollup(nr_konta, typ, kategoria);
```

NR_KONTA	TYP	KATEGORIA	SUM(KWOTA)
11-11111111	WPŁATA	PENSJA	8150
11-11111111	WPŁATA	UMOWA O DZIEŁO	1650
11-11111111	WPŁATA		9800
11-11111111	WYPŁATA	RACHUNEK ZA PRĄD	-720,5
11-11111111	WYPŁATA	RACHUNEK ZA TELEFON	-550
11-11111111	WYPŁATA	WYPŁATA W BANKOMACIE	-1050
11-11111111	WYPŁATA		-2320,5
11-11111111			7479,5
22-22222222	WPŁATA	PENSJA	10080
22-22222222	WPŁATA	UMOWA O DZIEŁO	1820
22-22222222	WPŁATA		11900
22-22222222	WYPŁATA	RACHUNEK ZA PRĄD	-790,5
22-22222222	WYPŁATA	RACHUNEK ZA TELEFON	-560
22-22222222	WYPŁATA	WYPŁATA W BANKOMACIE	-1050
22-22222222	WYPŁATA		-2400,5
22-22222222			9499,5
33-33333333	WPŁATA	PENSJA	7420
33-33333333	WPŁATA	UMOWA O DZIEŁO	1480
33-33333333	WPŁATA		8900
33-33333333	WYPŁATA	RACHUNEK ZA PRĄD	-650,5
33-33333333	WYPŁATA	RACHUNEK ZA TELEFON	-440
33-33333333	WYPŁATA	WYPŁATA W BANKOMACIE	-940
33-33333333	WYPŁATA		-2030,5
33-33333333			6869,5
			23848,5

GROUP BY nr_konta, typ, kategoria

GROUP BY nr_konta, typ

GROUP BY nr_konta

bez GROUP BY

Wartości kolumn które uległy "zwinięciu" w wyniku rozszerzenia klauzuli GROUP BY przyjmują wartości puste

Częściowa operacja ROLLUP

- Operacja ROLLUP może być operacją częściową – czyli obejmować tylko część kolumn występujących w klauzuli GROUP BY
- Dla przykładu:
 GROUP BY nr_konta, typ, ROLLUP(kategoria)
 Wyznaczy następujące posumowania
 - GROUP BY nr_konta, typ, kategoria
 - GROUP BY nr_konta, typ

```
SELECT nr_konta, typ, sum(kwota)
FROM   transakcje
GROUP BY nr_konta, rollup(typ);
```

```
GROUP BY nr_konta, typ
```

```
GROUP BY nr_konta
```

NR_KONTA	TYP	SUM(KWOTA)
11-11111111	WPLATA	9800
11-11111111	WYPŁATA	-2320,5
11-11111111		7479,5
22-22222222	WPLATA	11900
22-22222222	WYPŁATA	-2400,5
22-22222222		9499,5
33-33333333	WPLATA	8900
33-33333333	WYPŁATA	-2030,5
33-33333333		6869,5

CUBE

- Operator CUBE tworzy podsumowania dla wszystkich możliwych kombinacji grupowanych kolumn. W terminologii analiz wielowymiarowych, CUBE generuje podsumowania częściowe i ogólne tabeli faktów dla wszystkich możliwych wymiarów. Dla n kolumn grupujących CUBE tworzy 2^n podsumowań.
- Przykładowo klauzula
 GROUP BY CUBE(kontynent, kraj, miejscowość, dzielnica)
 Wyznaczy następujące posumowania
 - GROUP BY kontynent, kraj, miejscowość, dzielnica
 - GROUP BY kontynent, kraj, miejscowość
 - GROUP BY kontynent, kraj, dzielnica
 - GROUP BY kontynent, dzielnica, miejscowość
 - GROUP BY dzielnica, miejscowość, kraj
 - GROUP BY kontynent, kraj
 - GROUP BY kontynent, miejscowość
 - GROUP BY kontynent, dzielnica
 - GROUP BY kraj, dzielnica
 - GROUP BY kraj, miejscowość
 - GROUP BY dzielnica, miejscowość
 - GROUP BY kontynent
 - GROUP BY kraj
 - GROUP BY miejscowość
 - GROUP BY dzielnica
 - bez GROUP BY – podsumowanie całkowite

Przykład użycia CUBE

```
SELECT nr_konta, typ, sum(kwota)
FROM   transakcje
GROUP BY cube(nr_konta, typ)
ORDER BY nr_konta, typ
```

NR_KONTA	TYP	SUM(KWOTA)
11-11111111	WPŁATA	9800
11-11111111	WYPŁATA	-2320,5
11-11111111		7479,5
22-22222222	WPŁATA	11900
22-22222222	WYPŁATA	-2400,5
22-22222222		9499,5
33-33333333	WPŁATA	8900
33-33333333	WYPŁATA	-2030,5
33-33333333		6869,5
	WPŁATA	30600
	WYPŁATA	-6751,5
		23848,5

GROUP BY nr_konta, typ

GROUP BY nr_konta

GROUP BY typ

bez GROUP BY

Częściowa operacja CUBE

- Analogicznie jak w przypadku częściowej operacji ROLLUP tak samo istnieje częściowa operacja CUBE
- Pozwala ona objąć operacją CUBE tylko część wierszy występujących w klauzuli GROUP BY

- Dla przykładu:

GROUP BY nr_konta, CUBE(typ, kategoria)

Wyznaczy następujące posumowania

- GROUP BY nr_konta, typ, kategoria
- GROUP BY nr_konta, typ
- GROUP BY nr_konta, kategoria
- GROUP BY nr_konta

GROUP BY nr_konta, typ

GROUP BY nr_konta

```
SELECT nr_konta, typ, sum(kwota)
FROM   transakcje
GROUP BY nr_konta, cube(typ)
ORDER BY nr_konta, typ
```

NR_KONTA	TYP	SUM(KWOTA)
11-11111111	WPLATA	9800
11-11111111	WYPŁATA	-2320,5
11-11111111		7479,5
22-22222222	WPLATA	11900
22-22222222	WYPŁATA	-2400,5
22-22222222		9499,5
33-33333333	WPLATA	8900
33-33333333	WYPŁATA	-2030,5
33-33333333		6869,5

GROUPING SETS

- Zarówno operacja CUBE jak i ROLLUP może wyznaczać oprócz pożądaných podsumowań, także te, które są zbędne
- Operacja GROUPING SETS pozwala na jednoznaczne wskazanie tych podsumowań, które chcemy uzyskać

- Dla przykładu

```
GROUPING SETS(      (nr_konta, typ, kategoria),  
                   (nr_konta, typ),  
                   (nr_konta, kategoria) )
```

Wyznaczy następujące posumowania

- GROUP BY nr_konta, typ, kategoria
- GROUP BY nr_konta, typ
- GROUP BY nr_konta, kategoria

Przykład użycia GROUPING SETS

```
SELECT nr_konta, typ, sum(kwota)
FROM   transakcje
GROUP BY grouping sets((nr_konta, typ), (typ), (nr_konta), ())
ORDER BY nr_konta, typ
```

NR_KONTA	TYP	SUM(KWOTA)
11-11111111	WPŁATA	9800
11-11111111	WYPŁATA	-2320,5
11-11111111		7479,5
22-22222222	WPŁATA	11900
22-22222222	WYPŁATA	-2400,5
22-22222222		9499,5
33-33333333	WPŁATA	8900
33-33333333	WYPŁATA	-2030,5
33-33333333		6869,5
	WPŁATA	30600
	WYPŁATA	-6751,5
		23848,5

GROUP BY nr_konta, typ

GROUP BY nr_konta

GROUP BY typ

bez GROUP BY

Funkcja GROUPING

- Aby właściwie zinterpretować wiersze uzyskane w wyniku operacji rozszerzających grupowanie musimy potrafić rozróżniać wiersze reprezentujące określone podsumowania
- Oparcie się na wartościach pustych występujących w kolumnach grupowania może prowadzić do błędów
- Wartości puste w kolumnach grupowania mogą bowiem wynikać ze zawartości bazy danych
- Funkcja GROUPING posiadająca jako argument wyrażenie występujące w klauzuli GROUP BY jednoznacznie wskazuje na "zwinięcie" danego wymiaru. Wartość funkcji 1 wskazuje na "zwinięcie", wartość 0 na "normalny" wiersz jaki pojawiłby się bez wykorzystania rozszerzeń grupowania

Użycie funkcji GROUPING (1/2)

```

SELECT  nr_konta, typ, grouping(nr_konta) gnr, grouping(typ) gt, sum(kwota)
FROM    transakcje
GROUP BY grouping sets((nr_konta, typ), (typ), (nr_konta), ())
ORDER BY nr_konta, typ

```

NR_KONTA	TYP	GNR	GT	SUM(KWOTA)
11-11111111	WPLATA	0	0	9800
11-11111111	WYPŁATA	0	0	-2320,5
11-11111111		0	1	7479,5
22-22222222	WPLATA	0	0	11900
22-22222222	WYPŁATA	0	0	-2400,5
22-22222222		0	1	9499,5
33-33333333	WPLATA	0	0	8900
33-33333333	WYPŁATA	0	0	-2030,5
33-33333333		0	1	6869,5
	WPLATA	1	0	30600
	WYPŁATA	1	0	-6751,5
		1	1	23848,5

Użycie funkcji GROUPING (2/2)

- Funkcję GROUPING bardzo często wykorzystuje się także do zastąpienia pustej wartości bardziej znaczącą informacją

```
SELECT decode(grouping(nr_konta),0,nr_konta,'Wszystkie konta') nr_konta,
       decode(grouping(typ),0,typ,'Wszystkie typy') typ,
       sum(kwota)
FROM   TRANSAKCJE
GROUP BY grouping sets((nr_konta, typ), (typ), (nr_konta), ( ))
```

NR_KONTA	TYP	SUM(KWOTA)
-----	-----	-----
11-11111111	WPLATA	9800
22-22222222	WPLATA	11900
33-33333333	WPLATA	8900
11-11111111	WYPŁATA	-2320,5
22-22222222	WYPŁATA	-2400,5
33-33333333	WYPŁATA	-2030,5
Wszystkie konta	WPLATA	30600
Wszystkie konta	WYPŁATA	-6751,5
11-11111111	Wszystkie typy	7479,5
22-22222222	Wszystkie typy	9499,5
33-33333333	Wszystkie typy	6869,5
Wszystkie konta	Wszystkie typy	23848,5

Wykorzystanie rozszerzenia grupowania

- W aplikacji rozszerzenie grupowania można wykorzystać do wypełnienia za pomocą jednego zapytania szablonu macierzowego.
- Przez dodanie innych funkcji grupujących można za pomocą jednego zapytania wyliczyć wiele takich szablonów

```
SELECT nr_konta, typ, avg(kwota)
FROM   transakcje
GROUP BY cube(nr_konta, typ)
ORDER BY nr_konta, typ
```

NR_KONTA	TYP	AVG (KWOTA)
11-11111111	WPŁATA	1400,00
11-11111111	WYPŁATA	-257,83
11-11111111		467,47
22-22222222	WPŁATA	1487,50
22-22222222	WYPŁATA	-300,06
22-22222222		593,72
33-33333333	WPŁATA	1271,43
33-33333333	WYPŁATA	-225,61
33-33333333		429,34
	WPŁATA	1390,91
	WYPŁATA	-259,67
		496,84

NR_KONTA	TYP		ŚREDNIA:
	WPŁATA	WYPŁATA	
11-11111111	1400,00	-257,83	467,47
22-22222222	1487,50	-300,06	593,72
33-33333333	1271,43	-225,61	429,34
ŚREDNIA:	1390,91	-259,67	496,84

Wykorzystanie rozszerzenia grupowania

- Innym przykładowym sposobem wykorzystania rozszerzonego grupowania jest jednoczesne odświeżenie wielu tabel pełniących następnie rolę perspektyw materializowanych

```

INSERT FIRST
WHEN (gk=1 AND gt=0) THEN
INTO mv_typ_sum(typ ,kwota) VALUES (typ, suma)
WHEN (gk=0 AND gt=1) THEN
INTO mv_nr_konta_sum(nr_konta,kwota) VALUES (nr_konta,suma)
WHEN (gk=0 AND gt=0) THEN
INTO mv_nr_konta_typ_sum(nr_konta, typ, kwota) VALUES (nr_konta, typ, suma)
SELECT nr_konta, typ, sum(kwota) suma,
       grouping(nr_konta) gk, grouping(typ) gt
FROM   transakcje
GROUP BY cube(nr_konta, typ)

```

```
SELECT * FROM mv_typ_sum;
```

TYP	KWOTA
-----	-----
WPLATA	30600
WYPŁATA	-6751,5

```
SELECT * FROM mv_nr_konta_sum;
```

NR_KONTA	KWOTA
-----	-----
11-11111111	7479,5
22-22222222	9499,5
33-33333333	6869,5

```
SELECT * FROM mv_nr_konta_typ_sum;
```

NR_KONTA	TYP	KWOTA
-----	-----	-----
11-11111111	WPLATA	9800
11-11111111	WYPŁATA	-2320,5
22-22222222	WPLATA	11900
22-22222222	WYPŁATA	-2400,5
33-33333333	WPLATA	8900
33-33333333	WYPŁATA	-2030,5

Rozszerzenia grupowania a wydajność

- Wszystkie rozszerzenia grupowania są operacjami bardzo wydajnymi znacząco przewyższającymi alternatywne rozwiązania

```
SELECT nr_konta, typ, avg(kwota)
FROM   transakcje
GROUP BY cube(nr_konta, typ);
```

Plan wykonywania

```
-----
0  SELECT STATEMENT Optimizer=CHOOSE (Cost=4 ...)
1 0  SORT (GROUP BY) (Cost=4 Card=5 Bytes=110)
2 1  GENERATE (CUBE)
3 2  SORT (GROUP BY) (Cost=4 ...)
4 3  TABLE ACCESS (FULL) OF ...
```

Plan wykonywania

```
-----
0  SELECT STATEMENT Optimizer=CHOOSE (Cost=14 ...)
1 0  UNION-ALL
2 1  SORT (GROUP BY) (Cost=4 Card=5 Bytes=110)
3 2  TABLE ACCESS (FULL) OF 'TRANSAKCJE' (Cost=2 Card=48 Bytes=1056)
4 1  SORT (GROUP BY) (Cost=4 Card=3 Bytes=45)
5 4  TABLE ACCESS (FULL) OF 'TRANSAKCJE' (Cost=2 Card=48 Bytes=720)
6 1  SORT (GROUP BY) (Cost=4 Card=2 Bytes=22)
7 6  TABLE ACCESS (FULL) OF 'TRANSAKCJE' (Cost=2 Card=48 Bytes=528)
. . .
```

```
SELECT nr_konta, typ, avg(kwota)
FROM   transakcje
GROUP BY nr_konta, typ
UNION ALL
SELECT nr_konta, NULL, avg(kwota)
FROM   transakcje
GROUP BY nr_konta
UNION ALL
SELECT NULL, typ, avg(kwota)
FROM   transakcje
GROUP BY typ
UNION ALL
SELECT NULL, NULL, avg(kwota)
FROM   transakcje
```