helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

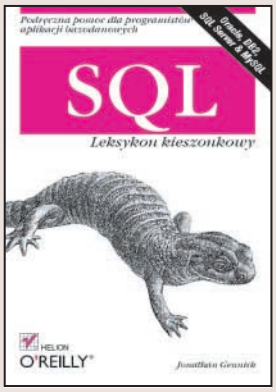
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

SQL. Leksykon kieszonkowy



Autor: Jonathan Gennick
Tłumaczenie: Tomasz Pędziwiatr
ISBN: 83-7361-596-2
Tytuł oryginału: SQL Pocket Guide
Format: B5, stron: 192

Podręczna pomoc dla programistów aplikacji bazodanowych

Język SQL jest podstawowym narzędziem programistów i operatorów baz danych. Posiada stosunkowo niewiele instrukcji, a jednak za jego pomocą można wykonywać wszystkie operacje na danych, tabelach i bazach. Często jednak podczas pracy trzeba przypomnieć sobie składnię instrukcji, znaczenie jej parametrów lub sposób korzystania z niej. Sprawę dodatkowo komplikuje to, że implementacje języka SQL w różnych systemach zarządzania bazami danych różnią się nieznacznie od siebie. Przetrząsanie kilkusetstronicowej dokumentacji zwykle zajmuje zbyt wiele czasu. Programistom, pracującym najczęściej pod presją czasu, potrzebne jest podręczne źródło podstawowych informacji.

Taką właśnie rolę pełni książka „SQL. Leksykon kieszonkowy". Zgromadzono w niej opisy poleceń języka w implementacjach dla najpopularniejszych systemów baz danych — Oracle, DB2, MS SQL Server oraz MySQL. Opis każdego z poleceń jest zilustrowany przykładami, co dodatkowo ułatwia zrozumienie jego zastosowania. W książce opisano:

- Funkcje grupowania i sumowania
- Funkcje przetwarzające dane
- Polecenie SELECT wraz z podzapytaniami i funkcjami agregującymi
- Sposoby uaktualniania i usuwania danych
- Metody wprowadzania danych
- Zarządzanie transakcjami
- Złączenia tabel

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wprowadzenie 7
Struktura książki 8
Informacje zwrotne 9
Konwencje typograficzne 9
Podziękowania 10
Przykłady kodu 11

Funkcje 13
Funkcje przetwarzania daty 15
Funkcje liczbowe i matematyczne 22
Funkcje trygonometryczne 25
Funkcje tekstowe 26
Pozostałe funkcje 33

Funkcje grupowania i sumowania 34
Funkcje agregujące 34
Klauzula GROUP BY 35
Użyteczne techniki stosowania klauzuli GROUP BY 38
Klauzula HAVING 39
Rozszerzenia klauzuli GROUP BY (Oracle) 41
Rozszerzenia klauzuli GROUP BY (SQL Server) 44

Konwersja typów danych 46
Funkcja ANSI/ISO CAST 47
Funkcja ANSI/ISO EXTRACT 48
Konwersja daty i czasu (Oracle) 49
Konwersja wartości liczbowych (Oracle) 54
Pozostałe funkcje konwersji (Oracle) 57
Konwersja daty i czasu (DB2) 58
Konwersja wartości liczbowych (DB2) 62
Inne funkcje konwersji (DB2) 64
Konwersja daty i czasu (SQL Server) 64
Konwersja wartości liczbowych (SQL Server) 69

Inne funkcje konwersji (SQL Server) ... 71
Konwersja daty i czasu (MySQL) ... 72
Konwersja wartości liczbowych (MySQL) ... 77

Literały ... *79*
Literały tekstowe ... 79
Literały liczbowe ... 81
Literały daty i czasu ... 81

Pobieranie danych ... *83*
Klauzula SELECT ... 83
Słowa kluczowe ALL i DISTINCT ... 93
Klauzula FROM ... 95
Klauzula WHERE ... 99
Klauzula GROUP BY ... 99
Klauzula HAVING ... 100
Klauzula ORDER BY ... 100

Podzapytania ... *101*
Klauzula WITH ... 102
Klauzula WITH i podzapytania skorelowane ... 104

Predykaty ... *106*
Predykaty porównań grupowych ... 107
Większa liczba wartości po lewej stronie porównania (Oracle) ... 109
Predykaty EXISTS ... 110
Predykaty IN ... 110
Predykaty BETWEEN ... 111
Predykaty LIKE ... 112

Scalanie danych ... *113*

Uaktualnianie danych ... *115*
Proste uaktualnianie ... 115
Nowe wartości pozyskiwane z podzapytań ... 116
Uaktualnianie danych za pośrednictwem kursora ... 117
Uaktualnianie danych za pomocą widoków i podzapytań ... 117
Uaktualnianie partycji (Oracle) ... 118
Zwracanie uaktualnionych danych (Oracle) ... 118
Klauzula FROM instrukcji UPDATE (SQL Server) ... 119

Unie *120*
Operacja UNION i UNION ALL 120
Kolejność przetwarzania instrukcji 122
Operacja EXCEPT (lub MINUS) 123
Operacja INTERSECT 125

Usuwanie danych *127*
Usuwanie wszystkich wierszy 129
Usuwanie danych z widoków i podzapytań 130
Usuwanie danych z partycji (Oracle) 130
Zwracanie usuwanych danych (Oracle) 131
Podwójna klauzula FROM (SQL Server) 132

Wartości NULL *133*
Predykaty dla wartości NULL 133
Wartości NULL w wyrażeniach CASE 135
Funkcje operujące wartościami NULL (Oracle) 135
Funkcje operujące wartościami NULL (DB2) 136
Funkcje operujące wartościami NULL (SQL Server) 137
Funkcje operujące wartościami NULL (MySQL) 137

Wprowadzanie danych *138*
Wprowadzanie pojedynczych wierszy 138
Obiekty docelowe podczas wprowadzania danych 140
Wprowadzanie danych do podzapytań 140
Wprowadzanie danych
za pomocą ścieżek bezpośrednich (Oracle) 141
Zwracanie wprowadzonych wartości (Oracle) 142
Wprowadzenie danych do wielu tabel (Oracle) 143

Wyrażenia CASE *145*
Proste wyrażenia CASE 145
Przeszukiwane wyrażenia CASE 146

Wyrażenia regularne *148*
Wyrażenia regularne (Oracle) 148
Wyrażenia regularne (SQL Server) 151
Wyrażenia regularne (MySQL) 151

Zapytania hierarchiczne *153*
Klauzula WITH rekurencyjnych zapytań ANSI/ISO (DB2) 153
Składnia klauzuli CONNECT BY (Oracle).....155

Zapytania rekurencyjne..... *161*

Zapytania retrospektywne (Oracle)..... *161*

Zarządzanie transakcjami *162*
Tryb automatycznego zatwierdzania 163
Rozpoczynanie transakcji 164
Kończenie transakcji 167
Przerwanie transakcji 169
Przerwanie transakcji i powrót do wyznaczonego punktu.....170

Złączanie tabel *171*
Koncepcja złączenia 171
Złączenia bezwarunkowe 173
Złączenia wewnętrzne.....174
Złączenia wyznaczane za pomocą nierówności 178
Złączenia zewnętrzne 179

Skorowidz *185*

Zmiana wielkości liter ciągu tekstowego

Aby zamienić litery ciągu tekstowego na wielkie lub małe, należy zastosować odpowiednio funkcje `UPPER` lub `LOWER`.

```
UPPER(ciąg_tekstowy)
LOWER(ciąg_tekstowy)
```

Serwer Oracle udostępnia również funkcję `INITCAP(ciąg_tekstowy)`, której zadaniem jest ustanowienie wielkiej litery na początku każdego słowa ciągu tekstowego i zapewnienie, że pozostałe litery każdego słowa będą małymi literami.

Baza danych DB2 obsługuje również inne nazwy dla funkcji `UPPER` i `LOWER` — `UCASE` i `LCASE`.

Pozostałe funkcje

Wśród funkcji bazy danych Oracle są dostępne dwie, które nie spełniają kryteriów żadnej z wymienionych kategorii. Nie oznacza to jednak, że są mniej użyteczne.

`GREATEST(wartość[, wartość...])`
: Funkcja ta zwraca największą wartość z listy wprowadzonych wartości. Danymi wejściowymi mogą tu być zarówno liczby, jak i daty bądź ciągi tekstowe.

`LEAST(wartość[, wartość...])`
: Funkcja ta zwraca najmniejszą wartość z listy wprowadzonych wartości. Danymi wejściowymi mogą tu być zarówno liczby, jak i daty bądź ciągi tekstowe.

Funkcje grupowania i sumowania

Język SQL umożliwia grupowanie wierszy w zbiory, a następnie zestawianie uzyskanych wyników na wiele sposobów. Ostatecznie zwracany jest pojedynczy wiersz utworzony na bazie takiego zbioru. Uzyskanie wspomnianego rezultatu wymaga zastosowania klauzul `GROUP BY` lub `HAVING` oraz funkcji agregujących.

Funkcje agregujące

Funkcje agregujące pobierają jako dane wejściowe zbiór wartości, po jednej z każdego wiersza, i zwracają jedną wartość wyniku. Jedną z najczęściej wykorzystywanych funkcji agregujących jest `COUNT`. Jej zadanie polega na zliczaniu wszystkich niepustych wartości kolumny. Przedstawiona poniżej instrukcja zlicza wszystkie odsyłacze zapisane w tabeli `atrakcje`.

```
SELECT COUNT(url_atrakcji)
FROM atrakcje;
```

Dołączenie do polecenia słów kluczowych `ALL` i `DISTINCT` pozwala na określenie, czy wszystkie niepuste (różne od `NULL`) wartości zostaną wykorzystane jako dane wejściowe oraz czy odrzucone zostaną wartości powtarzające się.

```
SELECT COUNT(DISTINCT id_miasta),
       COUNT(ALL id_miasta)
FROM atrakcje;
```

Słowo kluczowe `ALL` jest dołączane w sposób domyślny — instrukcja `COUNT(`*`wyrażenie`*`)` jest tożsama z instrukcją `COUNT(ALL` *`wyrażenie`*`)`.

Funkcja `COUNT` ma szczególny charakter, gdyż umożliwia przekazanie jako argumentu symbolu gwiazdki (*).

```
SELECT COUNT(*) FROM atrakcje;
```

W przypadku zastosowania instrukcji `COUNT(*)` zliczaniu podlegają wiersze, a nie wartości kolumn. Fakt występowania wartości `NULL` w kolumnie nie ma tu żadnego znaczenia, ponieważ odrzucanie wartości pustych występuje jedynie podczas zliczania wartości kolumn, a nie wierszy jako całości.

W tabeli 3. zostały zestawione najczęściej wykorzystywane funkcje agregujące. Większość producentów baz danych implementuje wszystkie z wymienionych funkcji. Na szczególną uwagę zasługuje w tym przypadku firma Oracle, która wyposaża swoje produkty w bardzo wiele funkcji agregujących, choć składnia ich wywołania często jest dość skomplikowana. Więcej informacji na ten temat znajduje się w dokumentacji serwera.

Klauzula GROUP BY

Użyteczność funkcji agregujących jest zauważalna dopiero wówczas, gdy zastosuje się je do grupy wierszy, a nie do wszystkich wierszy tabeli. W tym celu trzeba zastosować klauzulę `GROUP BY`. Kolejna z prezentowanych instrukcji umożliwia wyznaczenie liczby atrakcji turystycznych w każdym mieście.

```
SELECT m.nazwa_miasta, COUNT(*)
FROM miasta m INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY m.nazwa_miasta;
```

Tabela 3. Najczęściej wykorzystywane funkcje agregujące

Funkcja	Opis
`AVG(`*x*`)`	Zwraca wartość średnią ze zbioru liczb
`COUNT(`*x*`)`	Zlicza wartości różne od `NULL` danego zbioru
`MAX(`*x*`)`	Zwraca największą wartość ze zbioru
`MEDIAN(`*x*`)`	Zwraca wartość mediany (wartość środkową) danego zbioru liczb. Zwracana wartość może być wynikiem interpolacji. Funkcja jest dostępna jedynie w bazie danych Oracle

MIN(*x*)	Zwraca najmniejszą wartość ze zbioru
STDDEV(*x*)	Zwraca wartość odchylenia standardowego dla zbioru wartości. W serwerze SQL Server w nazwie funkcji STDEV występuje tylko jedna litera D
SUM(*x*)	Zwraca wartość sumy liczb zbioru
VARIANCE(*x*)	Zwraca wartość wariancji dla zbioru wartości. W serwerze SQL Server funkcja ta ma nazwę VAR. Nie występuje natomiast w bazach danych DB2 i MySQL

Podczas wykonywania tego typu zapytania baza danych w pierwszej kolejności sortuje wiersze, a następnie grupuje je zgodnie z wyrażeniem określonym w klauzuli GROUP BY.

```
Munising        Pictured Rocks
Munising        Valley Spur
Munising        Shipwreck Tours

Gladstone       Hoegh Pet Casket Company

Hancock         Quincy Steam Hoist
Hancock         Temple Jacob
Hancock         Finlandia University

Germfask        Seney National Wildlife Refuge

...
```

Grupy mogą się niekiedy składać tylko z jednego wiersza. Grupowanie wymaga zazwyczaj przeprowadzenia pewnej formy sortowania danych. Niemniej, jak można zauważyć na powyższym przykładzie, sortowanie musi być wykonywane jedynie do pewnego etapu — do chwili pogrupowania stosownych wierszy.

Po utworzeniu grup każda funkcja agregująca jest wykonywana tylko raz na danej grupie. Wartość zastosowanej w tym przypadku funkcji COUNT(*) jest wyznaczana oddzielnie dla każdej grupy.

```
Munising        3
```

```
Munising
Munising

Gladstone       1

Hancock         3
Hancock
Hancock

Germfask        1

...
```

Wszystkie kolumny, którym nie odpowiada żadna wartość funkcji agregującej, zostały zastąpione jednym wpisem.

```
Munising        3
Gladstone       1
Hancock         3
Germfask        1
...
```

W praktyce sprowadzenie wielu wierszy do jednego wiersza wartości zagregowanych oznacza, że funkcja agregująca *musi* zostać zastosowana w odniesieniu do dowolnej kolumny nie objętej klauzulą `GROUP BY`.

Użyteczne techniki stosowania klauzuli GROUP BY

W dalszej części tego podrozdziału zostały opisane niektóre, częściej stosowane techniki budowania zapytań z klauzulą `GROUP BY`.

Zmniejszenie liczby kolumn w klauzuli GROUP BY

Niekiedy zachodzi konieczność umieszczenia danej kolumny w sekcji SELECT zapytania z klauzulą GROUP BY, ale bez umieszczania tej kolumny w klauzuli GROUP BY. Jako przykład można rozważyć przedstawione poniżej zapytanie, w którym nazwa miasta wynika z identyfikatora miasta.

```
SELECT m.id_miasta, n.nazwa_miasta, COUNT(*)
FROM miasta m INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY m.id_miasta, m.nazwa_miasta;
```

W takim przypadku, zamiast grupowania względem dwóch kolumn, znacznie korzystniejsze będzie przeprowadzenie grupowania jedynie względem kolumny *id_miasta*. Skróceniu ulega w ten sposób klucz sortowania. Sortowanie grupujące będzie prawdopodobnie wykonane znacznie szybciej i zużyje mniej przestrzeni dyskowej. Jednym ze sposobów osiągnięcia zamierzonego efektu jest zastosowanie następującego zapytania.

```
SELECT m.id_miasta, MAX(m.nazwa_miasta), COUNT(*)
FROM miasta m INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY m.id_miasta;
```

Kolumna *nazwa_miasta* została usunięta z klauzuli GROUP BY. Jednocześnie do sekcji SELECT zostało dodane wywołanie funkcji MAX. Dzięki temu spełnione jest założenie, że wszystkie wyrażenia nie objęte klauzulą GROUP BY muszą być uwzględnione w funkcji agregującej. W rezultacie otrzymujemy tak samo użyteczną wartość nazwy miasta. Wynika to z faktu, że wszystkie nazwy miast należące do grupy miast o jednakowej wartości *id_miasta* są takie same. Funkcja MAX może więc zwrócić tylko tę jedną nazwę.

Grupowanie przed złączeniem

W prezentowanych wcześniej przykładach zastosowania klauzuli `GROUP BY` definiowane były złączenia wykonywane przed operacją grupowania. Wykorzystanie podzapytań umożliwia wykonanie zapytania w inny sposób, tak by złączenie zostało wykonane po zakończeniu działania funkcji agregującej.

```
SELECT m.nazwa_miasta, agr.liczba_atr
FROM miasta m INNER JOIN(
   SELECT id_miasta, COUNT(*) liczba_atr
   FROM atrakcje
   GROUP BY id_miasta) agr
ON m.id_miasta = agr.id_miasta;
```

Zaletą zastosowania tej wersji zapytania jest to, że złączenie jest wykonywane na znacznie mniejszej liczbie wierszy, niż gdyby było przeprowadzone przed agregacją danych. Dodatkową korzyścią jest zmniejszenie ilości potrzebnej przestrzeni dyskowej oraz pamięci operacyjnej. Wiersze poddawane grupowaniu nie zawierają bowiem żadnych informacji pochodzących z tabeli `miasta`.

Klauzula HAVING

Klauzula `HAVING` nakłada ograniczenia na wiersze zwracane przez zapytanie zawierające klauzulę `GROUP BY`. Na przykład w celu pobrania nazw tylko tych miast, w których występuje więcej niż jedna atrakcja turystyczna, można zastosować instrukcję:

```
SELECT m.nazwa_miasta, COUNT(*)
FROM miasta m INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY m.nazwa_miasta
HAVING COUNT(*) > 1;
```

Nigdy nie należy umieszczać w klauzuli `HAVING` warunku, który nie obejmuje funkcji agregacji. Rozważmy kolejne zapytanie, które

udostępnia informacje na temat liczby atrakcji turystycznych Keweenaw Peninsula.

```
SELECT m.nazwa_miasta, COUNT(*)
FROM m miasta INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY m.nazwa_miasta
HAVING m.nazwa_miasta IN
  ('Copper Harbor', 'Hancock');
```

Znacznie efektywniejsze jest zastosowane zapytania, w którym ograniczenia nakładane na nazwy miast są wyrażone za pomocą klauzuli `WHERE`.

```
SELECT m.nazwa_miasta, COUNT(*)
FROM m miasta INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
WHERE m.nazwa_miasta IN
  ('Copper Harbor', 'Hancock')
GROUP BY m.nazwa_miasta
```

Klauzula `WHERE` zmniejsza liczbę wierszy podawanych działaniu klauzuli `GROUP BY`. Konieczne jest więc zapisanie mniejszej liczby wierszy i wykonanie mniejszej liczby operacji funkcji agregującej. Klauzula `HAVING` znajduje zastosowanie w przypadku filtrowania wierszy będących wynikiem operacji `GROUP BY`, jeśli filtrowaniu podlegają agregowane wartości.

Rozszerzenia klauzuli GROUP BY (Oracle)

W bazach danych Oracle zostało zaimplementowanych kilka użytecznych rozszerzeń klauzuli `GROUP BY`. Należą do nich: `ROLLUP`, `CUBE` oraz `GROUPING SETS`. Funkcje poszczególnych opcji zostały omówione w kolejnych podrozdziałach. Dodatkowo opisane zostały również pewne funkcje skalarne, które ułatwiają przetwa-

rzanie danych pozyskanych w wyniku zastosowania wspomnianych rozszerzeń.

Rozszerzenie ROLLUP (Oracle)

Rozszerzenie klauzuli GROUP BY o słowo kluczowe ROLLUP powoduje dodanie wiersza podsumowania do każdej grupy danych.

```
SELECT h.nazwa_hrabstwa, m.nazwa_miasta,
       COUNT(a.id_atrakcji) liczba
FROM (hrabstwa h INNER JOIN miasta m
     ON h.id_hrabstwa = m.id_hrabstwa)
     INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY ROLLUP(h.nazwa_hrabstwa, m.nazwa_miasta)
HAVING COUNT(a.id_atrakcji) > 1;
```

Poniżej został zamieszczony wynik wykonania instrukcji. Wiersze pogrubione zostały wygenerowane w wyniku użycia opcji ROLLUP.

```
NAZWA_HRABSTWA  NAZWA_MIASTA        LICZBA
--------------- --------------- ----------
Alger           Munising                 3
Alger                                    3
Houghton        Hancock                  3
Houghton                                 3
Marquette       Ishpeming                2
Marquette       Marquette                3
Marquette                                5
                                        20
```

Dołączenie klauzuli GROUP BY spowodowało wygenerowanie standardowego podsumowania względem miast. Słowo kluczowe ROLLUP odpowiada za dodanie podsumowań na wszystkich pozostałych poziomach — wygenerowanych dla poszczególnych hrabstw i dla całego zestawienia. Hrabstwo Marquette ma pięć atrakcji turystycznych, natomiast całe zestawienie informuje o dwudziestu atrakcjach.

Opcja ROLLUP nie musi być stosowana w odniesieniu do wszystkich kolumn klauzuli GROUP BY. Przykładowy, przedstawiony poniżej, fragment zapytania umożliwia zliczenie atrakcji wszystkich miast i wszystkich hrabstw, ale nie powoduje wygenerowania ogólnej wartości sumy, widocznej w poprzednim przykładzie.

```
GROUP BY h.nazwa_hrabstwa, ROLLUP(m.nazwa_miasta)
```

Rozszerzenie CUBE (Oracle)

Dodanie do instrukcji słowa kluczowego CUBE powoduje wygenerowanie podsumowań dla wszystkich możliwych kombinacji kolumn wymienionych w zapytaniu oraz dołączenie ogólnej wartości sumy.

```
SELECT h.nazwa_hrabstwa, m.nazwa_miasta,
       COUNT(a.id_atrakcji) liczba
FROM (hrabstwa h INNER JOIN miasta m
     ON h.id_hrabstwa = m.id_hrabstwa)
     INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
WHERE h.nazwa_hrabstwa = 'Houghton'
GROUP BY CUBE(h.nazwa_hrabstwa, m.nazwa_miasta);

NAZWA_HRABSTWA  NAZWA_MIASTA        LICZBA
--------------- --------------- ----------
                                         3
                Hancock                  3
Houghton                                 3
Houghton        Hancock                  3
```

Rozszerzenie GROUPING SETS (Oracle)

Wprowadzona w bazie danych Oracle 9*i* opcja GROUPING SETS umożliwia wyznaczanie dowolnych obszarów grupowania.

```
SELECT h.nazwa_hrabstwa, m.nazwa_miasta,
       COUNT(a.id_atrakcji) liczba
```

```
FROM (hrabstwa h INNER JOIN miasta m
     ON h.id_hrabstwa = m.id_hrabstwa)
     INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY
  GROUPING SETS(h.nazwa_hrabstwa, m.nazwa_miasta)
HAVING COUNT(a.id_atrakcji) > 1;

NAZWA_HRABSTWA  NAZWA_MIASTA        LICZBA
--------------- --------------- ----------
Alger                                    3
Houghton                                 3
Marquette                                5
                Hancock                  3
                Ishpeming                2
                Marquette                3
                Munising                 3
```

Zastosowane w tym przykładzie rozszerzenie `GROUPING SETS` powoduje wykonanie funkcji agregującej dla hrabstw i miast. Uzyskanie takiego samego wyniku bez wykorzystania opcji `GROUPING SETS` wymagałoby zrealizowania dwóch zapytań.

Funkcje związane z klauzulą GROUP BY (Oracle)

Jako kolejne zostaną przedstawione funkcje, których stosowanie wraz z dostępnymi w bazie Oracle rozszerzeniami klauzuli `GROUP BY` bywa bardzo użyteczne.

`GROUPING(`*`kolumna`*`)`
: Funkcja ta zwraca wartość 1, jeśli wynikiem operacji `CUBE`, `ROLLUP` lub `GROUPING SETS` była wartość `NULL`. W przeciwnym przypadku wynikiem jest wartość 0.

`GROUPING_ID(`*`kolumna`*`, `*`kolumna`*`, ...)`
: Działanie tej funkcji jest podobne do funkcji `GROUPING`. Różnica polega na tym, że w jej wyniku zwracany jest wektor

bitów o wartościach 1 lub 0, zależnych od tego, czy odpowiadające im kolumny zawierają wartości NULL, wygenerowane przez rozszerzenia klauzuli GROUP BY. Funkcja ta jest dostępna w bazie danych Oracle w wersji 9*i* i kolejnych.

GROUP_ID()
: Funkcja ta umożliwia wyznaczenie duplikowanych wierszy, generowanych przez opcje CUBE, ROLLUP i GROUPING SETS. Wynikiem jej działania jest wartość z przedziału od 0 do *n*–1, odpowiadająca każdemu wierszowi zbioru o *n* duplikatach. Poprzez wykorzystanie zwracanej przez funkcję wartości można zdecydować o tym, ile zduplikowanych wierszy powinno pozostać w wyniku. Aby wyeliminować wszystkie duplikaty, wystarczy zastosować klauzulę HAVING GROUP_ID()=0.

Rozszerzenia klauzuli GROUP BY (SQL Server)

W serwerze SQL, podobnie jak w bazie danych Oracle, również zaimplementowane zostały rozszerzenia CUBE i ROLLUP. Niestety sposób ich wykorzystania jest nieco odmienny i mniej elastyczny.

Rozszerzenie ROLLUP (SQL Server)

Opcja ROLLUP dodaje wiersz podsumowania dla każdej grupy danych.

```
SELECT h.nazwa_hrabstwa, m.nazwa_miasta,
       COUNT(a.id_atrakcji) liczba
FROM (hrabstwa h INNER JOIN miasta m
     ON h.id_hrabstwa = m.id_hrabstwa)
     INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
GROUP BY h.nazwa_hrabstwa, m.nazwa_miasta WITH ROLLUP
HAVING COUNT(a.id_atrakcji) > 1;
```

Uzyskany rezultat jest identyczny z prezentowanym podczas omawiania rozszerzenia ROLLUP bazy danych Oracle. Serwer SQL Server nie pozwala jednak na stosowanie opcji ROLLUP jedynie w odniesieniu do wybranych kolumn klauzuli GROUP BY. Rozszerzenie obejmuje wszystkie kolumny lub nie obejmuje żadnej z nich.

Rozszerzenie CUBE (SQL Server)

Opcja CUBE generuje podsumowania dla wszystkich kombinacji kolumn ujętych w klauzuli GROUP BY. Dodaje także ogólne podsumowanie.

```
SELECT h.nazwa_hrabstwa, m.nazwa_miasta,
       COUNT(a.id_atrakcji) liczba
FROM (hrabstwa h INNER JOIN miasta m
     ON h.id_hrabstwa = m.id_hrabstwa)
     INNER JOIN atrakcje a
     ON m.id_miasta = a.id_miasta
WHERE h.nazwa_hrabstwa = 'Houghton'
GROUP BY h.nazwa_hrabstwa, m.nazwa_miasta WITH CUBE;
```

Uzyskany wynik nie odbiega w żaden sposób od rezultatu wykonania zapytania z opcją CUBE w bazie danych Oracle (opisanego wcześniej). Podobnie jak w przypadku rozszerzenia ROLLUP, nie można stosować opcji CUBE jedynie w odniesieniu do wybranych kolumn klauzuli GROUP BY.