

Rozwiązania wspomagające przetwarzanie wielkich zbiorów danych (VLDB) we współczesnych systemach zarządzania bazami danych

Tomasz Traczyk
ttraczyk@ia.pw.edu.pl



Wydział Elektroniki i Technik Informatycznych
Politechnika Warszawska

Rozwiązania wspomagające VLDB...

- ❖ Wprowadzenie
- ❖ Sposoby ucieczki od problemu
- ❖ Rozwiązania dla zastosowań transakcyjnych i analitycznych
- ❖ Rozwiązania dla baz analitycznych
- ❖ Podsumowanie



T. Traczyk

DataCenter GigaCon'07

2

Wprowadzenie

VLDB – *Very Large Databases*

- Co to znaczy VLDB?
 - granica nieostra i przesuwa się
 - zależy od sprzętu i sposobu wykorzystania
 - pojedyncze GB – jeszcze raczej nie
 - TB – już raczej tak
- Problemy z VLDB
 - wydajność zapytań
 - kłopotliwe operacje administracyjne
 - » np. backup
 - duże przyrosty danych
 - » pogorszenie zachowania systemów
 - koszty sprzętu
 - » pamięci dyskowej (?)
 - » szybkich serwerów
 - » urządzeń do backupów

O czym będzie ten wykład?

- Wybrane środki techniczne
 - wspomagające obsługę danych w VLDB
 - we współczesnych RDBMS
- Zakres
 - tylko zagadnienia dotyczące struktur danych
 - rozwiązania dla systemów
 - » transakcyjnych
 - » analitycznych
 - bez procesów ETL
 - bez wielowymiarowego składowania danych
 - bez zagadnień związanych ze
 - » sprzętem
 - » architekturą systemów
 - » administrowaniem DBMS



T. Traczyk

DataCenter GigaCon'07

3

Rozwiązania wspomagające VLDB...

- ❖ Wprowadzenie
- ❖ Sposoby ucieczki od problemu
- ❖ Rozwiązania dla zastosowań transakcyjnych i analitycznych
- ❖ Rozwiązania dla baz analitycznych
- ❖ Podsumowanie



T. Traczyk

DataCenter GigaCon'07

4

Podział bazy

Podział bazy danych

- Sposób
 - dane dzielimy np. wg głównych użytkowników
 - powstaje kilka niezależnych mniejszych baz
- Zalety
 - prostota koncepcyjna
 - wystarczy słabszy sprzęt
 - niepotrzebne dodatkowe umiejętności
- Wady
 - niejasne kryteria podziału
 - brak możliwości łącznego przetwarzania danych
 - większe koszty oprogramowania
 - większe koszty administrowania

Rozproszenie danych

- Sposób
 - dane dzielimy np. wg miejsca używania
 - powstaje kilka baz, tworzących łącznie bazę rozproszoną
- Zalety
 - można łącznie przetwarzać dane
 - wystarczy słabszy sprzęt
- Wady
 - niejasne kryteria rozproszenia
 - większe koszty oprogramowania
 - znacznie większe koszty administrowania
 - potrzebne nowe znaczące umiejętności
 - problemy z replikacją
 - » opóźnienia
 - » konflikty przy replikacji *multi-master*
 - problemy z transakcjami rozproszonymi
 - » transakcje wątpliwe
 - koszty przewyższają korzyści

Kompresja

Kompresja tabel

- Idea
 - a) „odzysk” miejsca po wartościach pustych (NULL, 0 itp.)
 - b) prawdziwa kompresja (całych tabel lub wybranych partycji)
- Zastosowanie
 - głównie w bazach analitycznych
 - może być połączona z partycjonowaniem
 - kompresja wskazana, gdy
 - a) w tabeli jest dużo wartości pustych
 - b) w tabeli jest dużo powtórzeń (np. liczne powtórzenia długich kluczy obcych)
 - niewskazana, gdy przewiduje się dużo operacji DML
- Zalety
 - zmniejsza użycie dysku i pamięci operacyjnej
- Wady
 - zwiększa użycie CPU
 - wyraźnie spowalnia operacje DML

Kompresja indeksów

- Idea
 - zastąpienie wielokrotnych powtórzeń początkowych kolumn klucza jednym zapisem i odpowiednimi powiązaniem
- Zastosowanie
 - stosowana dla indeksów drzewiastych
 - » złożonych, tj. o kluczu wielokolumnowym
 - » gdy wartości początkowych kolumn indeksu powtarzają się wielokrotnie
 - takiej kompresji można też poddać tabele zorganizowane indeksowo
- Zalety
 - poważnie zmniejsza wielkość indeksu
 - zmniejsza liczbę operacji dyskowych
- Wady
 - zwiększa użycie CPU
 - nie zawsze prowadzi do wzrostu wydajności zapytań, czasem powoduje nawet niewielki spadek

Tabele zewnętrzne

Idea

- Dane w zewnętrznym pliku (poza bazą)
- Ten plik danych jest „podłączany” do bazy
 - specjalne pliki konfiguracyjne definiują format danych i sposób odczytu
- Plik jest w bazie widoczny tak jak tabela
 - dane z pliku są dostępne przez zwykłe zapytania SQL
 - wydajność zapytań jest oczywiście znacznie gorsza

Zastosowanie

- Sporadyczny dostęp do wielkich mało zmiennych zbiorów danych
 - np. z bibliotek CD-ROM
- Ładowanie danych zewnętrznych
 - np. procesy ETL w bazach analitycznych
- Zalety
 - dane nie „obciążają” bazy
 - » np. operacji administracyjnych
 - możliwy odczyt zewnętrznych formatów bez konwersji
 - możliwe wykorzystanie nośników *read-only*
- Wady
 - tylko odczyt
 - mała wydajność zapytań

Struktury relacyjno-obiektowe

Idea struktur relacyjno-obiektowych

- Struktury relacyjne są podstawą budowy b.d.
- W tabelach relacyjnych można umieszczać złożone obiekty
 - np. zagnieżdżone tabele
- SQL jest rozszerzony o możliwość działania na takich obiektach
- Zalety relacyjno-obiektowej b.d.
 - kompatybilność z rozwiązaniami relacyjnymi
 - możliwość wykorzystania części zalet obiektowości w połączeniu z zaletami bazy relacyjnej
 - ewolucyjne przejście do obiektowości

Zastosowanie struktur r-o w VLDB

- Upakowanie złożonych danych w mniejszej liczbie tabel, z zastosowaniem
 - złożonych struktur
 - zagnieżdżonych tabel
- Zalety
 - zmniejszenie liczby złączeń
 - przyspieszenie niektórych zapytań
 - znaczna redukcja użycia przestrzeni dyskowej
- Wady
 - spowolnienie niektórych zapytań
 - spowolnienie DML
 - » dla niektórych struktur, które są odczytywane/zapisywane w całości
 - znacznie trudniejsze programowanie
 - » konieczność stosowania specjalnych konstrukcji spoza ogólnie znanej części SQL
 - słaba kompatybilność z narzędziami do tworzenia aplikacji

- ❖ Wprowadzenie
- ❖ Sposoby ucieczki od problemu
- ❖ Rozwiązania dla zastosowań transakcyjnych i analitycznych
- ❖ Rozwiązania dla baz analitycznych
- ❖ Podsumowanie

Idea *index organized table*

- Wszystkie dane przechowywane są w strukturze indeksu klucza głównego
 - nie ma osobnego segmentu tabeli
 - dane spoza klucza głównego są „doklejone” do liści drzewa indeksu
- Dzięki temu przy wyszukiwaniu wg klucza głównego zmniejsza się liczba odczytów

Zastosowanie

- Tabele, w których składniki klucza głównego zajmują duży procent wiersza
- Typowe zastosowania:
 - GIS
 - hurtownie (wymiar)
 - struktury generyczne (metadane)
- Zalety
 - zmniejszenie zużycia dysku
 - przyspieszenie zapytań wg klucza głównego
- Wady
 - pewne spowolnienie zapytań z użyciem innych indeksów

Partycjonowanie

Co można partycjonować?

- Poszczególne tabele i ich indeksy
- Całą bazę danych
 - partycje tabel przypisuje się do partycji bazy
 - partycje mogą się znajdować na różnych serwerach

Wykorzystanie partycjonowania

- Podział na partycje zwykle zależy od wartości wybranych kolumn
 - np. zakresy dat
- Optymalizator zapytań może umieść
 - zidentyfikować potrzebne partycje
 - » np. na podstawie warunków w zapytaniu
 - wykonać eliminację zbędnych partycji (*partition pruning*)
 - » pozwala to wykonać efektywnie nawet *full scan*
 - wykorzystywać partycje w realizacji złączeń (*partition-wise joins*)

Zastosowanie

- Zalecane już dla tabel powyżej 2GB
- Bardzo wskazane dla danych intensywnie przyrastających
 - np. w hurtowniach danych
- Zalety
 - przezroczyste dla aplikacji
 - zapewnia znakomitą skalowalność
 - » wydajność wielu zapytań niemal niezależna od przyrostu danych
 - » operacje administracyjne wykonywane na każdej partycji osobno
 - » możliwe zrównoleżenie operacji na partycjach tej samej tabeli
- Wady
 - wad technicznych właściwie brak
 - dość trudne poprawne zaprojektowanie
 - » partycjonowanie powinno być uwzględnione w projektowaniu struktur danych
 - spory dodatkowy koszt licencji

Partycjonowanie, c.d.

Partycjonowanie tabel

- Sposoby partycjonowania
 - wg zakresów wartości klucza partycjonowania
 - » najczęściej wg zakresu dat
 - wg wartości wskazanych enumeratywnie
 - » umożliwia grupowanie wartości niekolejnych
 - » odpowiednie gdy nie można znaleźć kryterium równomiernego podziału
 - złożone
 - » dwupoziomowe
- Celem jest uzyskanie partycji
 - o rozsądnych rozmiarach
 - o podobnej wielkości
 - zawierających dane, które zapewne będą łącznie używane

Partycjonowanie indeksów

- Indeksy partycjonowane lokalnie
 - partycje indeksu zgodne z partycjami tabeli
 - łatwe do zarządzania i utrzymania przez DBMS
 - klucz partycjonowania nie musi być początkową częścią klucza indeksu
 - pożądany sposób partycjonowania
 - » zwłaszcza w bazach analitycznych
- Indeksy partycjonowane globalnie
 - partycje indeksu niezgodne z partycjami tabeli
 - klucz partycjonowania zwykle musi być początkową częścią klucza indeksu
 - rozwiązanie bardziej uniwersalne
 - » indeksy mogą być partycjonowane inaczej od tabel
 - stosowane raczej w systemach transakcyjnych

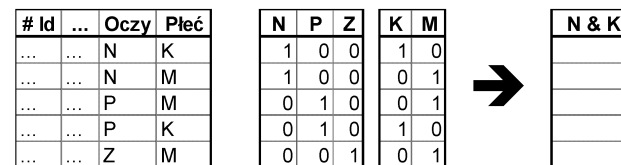
- ❖ Wprowadzenie
- ❖ Sposoby ucieczki od problemu
- ❖ Rozwiązania dla zastosowań transakcyjnych i analitycznych
- ❖ Rozwiązania dla baz analitycznych
- ❖ Podsumowanie

Idea

- Klucz indeksowania zawiera niewiele wartości o dowolnej częstotliwości
 - nie jest potrzebna wysoka selektywność
- W zapytaniach wiele warunków równościowych
- Dla każdego klucza indeksowania tworzy się binarną macierz rzadką
 - wiersze odpowiadają wierszom w tabeli
 - kolumny odpowiadają wartościom klucza
- Macierz tę przechowuje się na dysku w postaci skompresowanej
- Operatory logiczne na warunkach równościowych realizuje się jako operacje logiczne na bitach odpowiednich kolumn macierzy indeksów bitowych

Zastosowanie

- W zasadzie wyłącznie w bazach analitycznych
 - indeksowanie tabel faktów po wymiarach
- Zalety
 - niewielkie rozmiary indeksów na dysku – szybki odczyt
 - wykorzystanie wielu indeksów do tej samej tabeli w jednym zapytaniu
- Wady
 - bardzo spowalniają DML



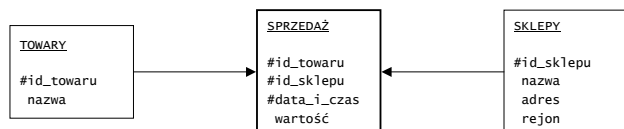
Idea

- Indeksuje tabelę faktów wartościami z tabeli wymiarów

```
CREATE BITMAP INDEX sprzedaz_towar_idx
ON sprzedaz(towary.nazwa)
FROM sprzedaz, towary
WHERE sprzedaz.id_towaru =
towary.id_towaru
```

Zastosowanie

- Przeznaczone do użycia w bazach analitycznych (struktury gwiazdowe)
- Stosowane, gdy warunki zapytania dotyczą deskryptora, a nie identyfikatora wymiaru
- Zalety
 - pozwalają uniknąć złączenia
 - » znacznie przyspieszają zapytania
- Wady
 - mogą ograniczać współbieżność

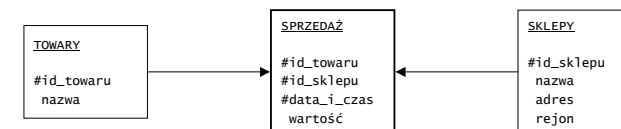


Idea

- Działanie
 - stosowane do zapytań
 - » do tabeli faktów
 - » ze złączeniami do wymiarów
 - automatyczne przepisywanie zapytań
 - » uniknięcie „prawdziwego” złączenia
- Fazy
 - podzapytanie do tabeli faktów z użyciem indeksów bitowych (bez złączenia!)
 - złączenie (*semi-join*) wyniku tego podzapytania z tabelami faktów (małymi)

Zastosowanie

- Warunki
 - hurtownia danych – struktura gwiazdy
 - zapytanie do tabeli faktów ze złączeniami do tabeli wymiarów
 - klucze obce zaindeksowane indeksami bitowymi
- Zalety
 - znaczny wzrost wydajności



- ❖ Wprowadzenie
 - ❖ Sposoby ucieczki od problemu
 - ❖ Rozwiązania uniwersalne
 - ❖ Rozwiązania dla baz analitycznych
 - ❖ Podsumowanie
-

- ✓ Dla VLDB istnieją liczne rozwiązania techniczne, dostępne w czołowych DBMS
- ✓ Rozwiązania wspomagające wykorzystanie VLDB mogą być
 - uniwersalne – do różnych zastosowań
 - specjalizowane do baz analitycznych (hurtowni danych)
- ✓ Rozwiązania te
 - są najczęściej przezroczyste dla aplikacji
 - ale zwykle powinny być projektowane już na etapie projektu struktur danych
- ✓ Dobrze zaprojektowany system
 - dobry DBMS
 - właściwie skonfigurowany
 - na dobrym sprzęcie
 - odpowiednio wykorzystujący specjalne rozwiązania dla VLDBpozwole efektywnie przetwarzać bardzo wielkie zbiory danych

Rozwiązania wspomagające
przetwarzanie wielkich zbiorów danych (VLDB)
we współczesnych systemach
zarządzania bazami danych