POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

WSPOMAGANIE INFORMATYCZNE ROZWOJU SPOŁECZNO-GOSPODARCZEGO I OCHRONY ŚRODOWISKA

Redakcja:
Jan Studziński
Ludosław Drelichowski
Olgierd Hryniewicz

WSPOMAGANIE INFORMATYCZNE ROZWOJU SPOŁECZNO-GOSPODARCZEGO I OCHRONY ŚRODOWISKA

Polska Akademia Nauk Instytut Badań Systemowych

Seria: BADANIA SYSTEMOWE

tom 36

Redaktor naukowy:

Prof. dr hab. Jakub Gutenbaum

Warszawa 2004

WSPOMAGANIE INFORMATYCZNE ROZWOJU SPOŁECZNO-GOSPODARCZEGO I OCHRONY ŚRODOWISKA

Redakcja:

Jan Studziński
Ludosław Drelichowski
Olgierd Hryniewicz

Książka wydana dzięki dotacji KOMITETU BADAŃ NAUKOWYCH

Książka zawiera wybór artykułów poświęconych omówieniu aktualnego stanu badań w kraju w zakresie rozwoju modeli, technik i systemów zarządzania oraz ich zastosowań w różnych dziedzinach gospodarki narodowej. Wyodrębnioną grupę stanowią artykuły omawiające aplikacyjne wyniki projektów badawczych i celowych KBN.

Recenzenci artykułów:

Dr Lucyna Bogdan
Prof. dr hab. inż. Olgierd Hryniewicz
Dr Grażyna Petriczek
Prof. dr hab. inż. Andrzej Straszak
Dr inż. Jan Studziński

Komputerowa edycja tekstu: Anna Gostyńska

Wydawca: Instytut Badań Systemowych PAN
ul. Newelska 6, 01-447 Warszawa

Sekcja Informacji Naukowej i Wydawnictw IBS PAN
tel. 836-68-22

Druk: Zakład Poligraficzny Urzędu Statystycznego w Bydgoszczy
Nakład 110 egz.

ISBN 83-85847-92-8
ISSN 0208-8028

GWIAŹDZISTE ZAPYTANIA W RELACYJNYCH BAZACH DANYCH I INDEKSOWANIE BITMAPOWE

Piotr Kuczyński
Instytut Systemów Informatycznych Zarządzania
Politechnika Szczecińska
<piotr.kuczynski@wi.ps.pl>

The area of this article is characteristic of bitmap indexing used as an optimization technique for information searching in complex data structures, focusing on star queries commonly used in data warehouses.

Keywords: Databases, star queries, data indexing, bitmap index, data warehouses.

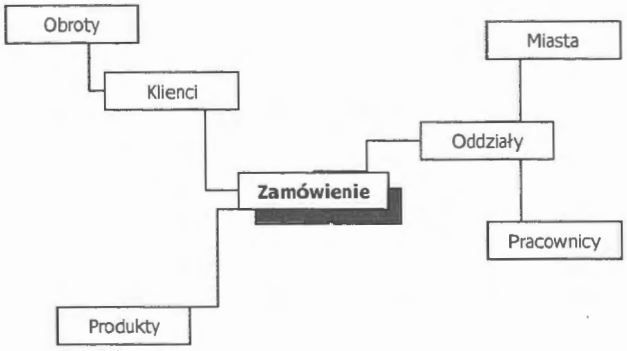
1. Wstęp

Hurtownie danych (HD) stanowią bardzo duże zbiory ustrukturyzowanych i uporządkowanych informacji obsługiwanych w większości przypadków przez systemy zarządzania relacyjnymi bazami danych (RDBMS), do których dostęp zapewnia oprogramowanie klasy OLAP (Online Analytical Processing) (Chaudhuri, Dayal, 1997).

Oprogramowanie to dostarcza narzędzi do interaktywnego wykonywania zapytań na HD w celu podejmowania skuteczniejszych i szybszych decyzji biznesowych. Informacje zawarte w HD pobierane są z różnych zewnętrznych źródeł, zaś na etapie zasilania HD zostają poddane procesowi oczyszczania i integracji i przekształcane są do statycznej, ujednoliconej postaci (Inmon, 1993). Do źródeł tych można zaliczyć zarówno oprogramowanie OLTP (Online Transaction Processing), jak również archaiczne systemy funkcjonujące od długiego czasu (ang. legacy systems) oraz inne nowoczesne systemy informatyczne wykorzystywane do obsługi operacyjnej działalności organizacji.

Żądania informacji z HD to zazwyczaj dość skomplikowane i powtarzalne zapytania dotyczące prowadzonej przez organizację działalności, jak na przykład: "*Wyszukaj wszystkie zamówienia złożone w oddziale w Szczecinie, na produkty kosztujące powyżej 50,00 zł, od klientów, których roczne obroty przekraczają 100 000,00zł*". Wymagają one wykonania znacznej ilości operacji połączenia informacji umieszczonych w różnych tablicach relacyjnej bazy danych, przechowujących bardzo duże ilości rekordów. Zapytania takie opierają się na tzw. ***złączeniach gwiaździstych***, ponieważ ich głównym trzonem jest tablica detalu (*detail table*), do której dołączanych jest wiele tabel opisowych (*descriptive*

tables), co pokazuje rys. 1. Tablica detalu jest zazwyczaj bardzo duża i dla przykładowej sytuacji podanej powyżej będzie zawierała informacje o elementach każdego zamówienia, dla dowolnego produktu i dowolnego klienta, obsługiwanego przez dowolny oddział firmy. Natomiast tabele opisowe są zazwyczaj dużo mniejsze, np. jeden rekord dla klienta. Niejednokrotnie występuje wiele poziomów tabel opisowych, które jak na przykładowym rys. 1. prowadzą od oddziałów firmy do miast, w których się te oddziały znajdują.



Rysunek 1. Schemat złączenia gwieździstego

Podczas pobierania informacji z HD na równi często z skomplikowanymi złączeniami wykorzystywane są techniki zagnieżdżania zapytań oraz agregacji danych, bazujące na grupowaniu rekordów w oparciu o złożone warunki logiczne, w celu wyznaczenia ich syntetycznych wartości. Wykonanie tak złożonych operacji może trwać godzinami, a nawet dniami, ze względu na ogromne ilości danych, jakie podlegają przetworzeniu.

Przeważająca część zapytań kierowanych do HD wiąże się również z wykonaniem dynamicznych zapytań *ad hoc*. Użytkownicy potrzebują zadawać dowolne pytania w dowolnej chwili, wyznaczając dowolne warunki odnośnie tabel bazowych umieszczonych w HD. Umiejętność szybkiej odpowiedzi na takie pytania stanowi krytyczny problem w środowisku hurtowni danych. Stąd też proponowane są różne rozwiązania zmierzające do przyspieszenia przetwarzania skomplikowanych zapytań do bazy danych. Jednym z nich jest indeksacja danych.

2. Indeksacja danych

Indeksy są obiektami bazy danych związanymi z tablicami tejże bazy danych, stworzonymi w celu przyspieszenia dostępu do informacji w nich umieszczonych. Różnorodne techniki indeksowania są znane od wielu lat dla relacyjnych baz danych i systemów klasy OLTP. Jednak ich podstawową wadą jest

pogarszająca się skuteczność wraz ze wzrostem ilości danych oraz podczas wykonywania skomplikowanych i iteracyjnych zapytań typowych dla aplikacji OLAP. Podstawowe różnice pomiędzy OLTP i OLAP pokazują, iż istnieje konieczność modyfikacji istniejących metod indeksacji danych, które wykorzystane do budowy systemów OLAP, nie spełniają pokładanych w nich oczekiwań (Colliat, 1996).

2.1 Czynniki wykorzystywane do określania techniki indeksowania dla wybranej kolumny tabeli

Istnieje wiele czynników, które należy rozpatrzyć na etapie projektowania tabel bazy danych, podejmując decyzję, które kolumny i w jaki sposób należy indeksować. Wśród nich należy wymienić najważniejsze:

a) Charakterystyka kolumny

Każda kolumna cechuje się różnymi atrybutami, których zestawienie pozwala dobrać najlepszy sposób indeksowania:

- Moc zbioru – moc zbioru dla wybranej kolumny określa liczbę rozróżnialnych wartości danych umieszczonych w tej kolumnie. Informacja o tym, czy moc jest duża, czy mała może być bardzo ważna podczas wyboru metody indeksowania, gdyż niektóre z nich pracują wydajnie tylko z określoną wartością mocy.
- Dystrybucja – jest to wartość, która mówi jak często występuje każda z rozróżnialnych wartości kolumny.
- Zakres wartości - zakres wartości dla informacji przechowywanych w indeksowanej kolumnie.

b) Zakres operacji języka SQL na wybranej kolumnie

Wiedza o tym, w jaki sposób będziemy odwoływali się do kolumny oraz jak wiele zapytań będzie posiadało warunki ograniczające oparte na wartościach kolumny również pozwala nam na lepsze dopasowanie metody indeksowania. Szczególną uwagę należy zwrócić na kolumny służące do operacji złączania tabel (zarówno równościowego, jak i zewnętrznego) oraz wykorzystywane w klauzulach ORDER BY i GROUP BY.

2.2 Cechy decydujące o jakości metody indeksowania i jej przydatności dla potrzeb HD

Podczas rozważania nad jakością wybranej metody indeksowania danych w tablicach relacyjnej bazy danych i jej przydatnością dla hurtowni danych, należy rozpatrzyć następujące aspekty:

- Ze względu na ogromną ilość danych przechowywanych w HD, indeks powinien mieć niewielki rozmiar i w sposób efektywny gospodarować przestrzenią dyskową.
- Indeks powinien umożliwiać wykorzystanie go w połączeniu z innymi indeksami w celu filtracji rekordów przed uzyskaniem bezpośredniego dostępu do danych.
- Indeks powinien wspierać wykonywanie zapytań *ad hoc*, ze szczególnym naciskiem na optymalizację operacji złączenia.
- Jednocześnie indeks powinien być łatwy do zbudowania (oraz dynamicznej aktualizacji), implementacji i zarządzania.

W systemach hurtowni danych stosuje się wiele technik indeksowania. Każda z nich jest bardziej lub mniej przydatna w konkretnych sytuacjach. Poniżej opisana została bitmapowa metoda indeksacji, zaproponowana po raz pierwszy (O'Neil, Graefe, 1995).

3. Bitmapowa metoda indeksacji

Tradycyjny indeks kolumny i tablicy T przyporządkowuje każdemu elementowi ze zbioru wartości kluczowych K listę identyfikatorów wierszy, o wartości indeksowanej kolumny równej wartości klucza (RID), tak jak zapisano we wzorze

$$T_{RID} \in T^j \Leftrightarrow T_{RID,i} = K_j, \text{gdzie } j \in \langle 0,k \rangle \tag{1.1}$$

Wiadomym jest, iż lista ta może być reprezentowana za pomocą bitmapy, zbudowanej z k wektorów o długości n-bitów, gdzie k odpowiada ilości elementów zbioru K, zaś n stanowi ogólną liczbę wierszy tej tabeli T. Ustawienie wartości bitu na 1 oznacza, iż wiersz przechowywany pod wskazanym indeksem zawiera kluczową wartość, zaś 0 przeciwnie (O'Neil, Graefe, 1995).

Rozważmy dla przykładu tabelę `Pracownicy`, która składa się z kilku kolumn, a w tym: `NumerPrac`, `Plec` i `StanCywilny`.

Tablica 1. Przykładowa tablica bazy danych

NumerPrac	...	Plec	StanCywilny
...	...	...	...
101		mężczyzna	żonaty
112		kobieta	panna
134		kobieta	panna
158		mężczyzna	wdowiec
160		mężczyzna	kawaler
...	...	...	...

Zbiory wartości w kolumnach `Plec` i `StanCywilny` charakteryzują się niedużą mocą (odpowiednio 2 możliwe wartości dla `Plec` i 6 dla `StanCywilny`), przez co uzasadnione wydaje się zastosowanie indeksu bitmpowego. Nie sprawdzi się on natomiast w przypadku kolumny `NumerPrac`, której wartości są unikalne, a więc posiada moc równą ilości rekordów w tablicy. W takim przypadku pozostajemy przy tradycyjnym indeksie opartym na *B-drzewie*.

Zbudujmy zatem indeks bitmapowy dla kolumny `Plec`. Ponieważ moc tego zbioru wynosi 2, otrzymujemy 2 wektory n-elementowe, których elementy odpowiadają kolejnym rekordom tabeli, co zostało pokazane w tab. 2 Ustawienie wartości 1 dla i-tego bitu wektora K_0 oznacza, że pole `Plec` i-tego rekordu tablicy posiada wartość `mężczyzna`.

Tablica 2. Indeks bitmapowy dla kolumny `Plec`

NumerPrac	K_0 <Plec='mężczyzna'>	K_1 <Plec='kobieta'>
...	...	...
101	1	0
112	0	1
134	0	1
158	1	0
160	1	0
...	...	...

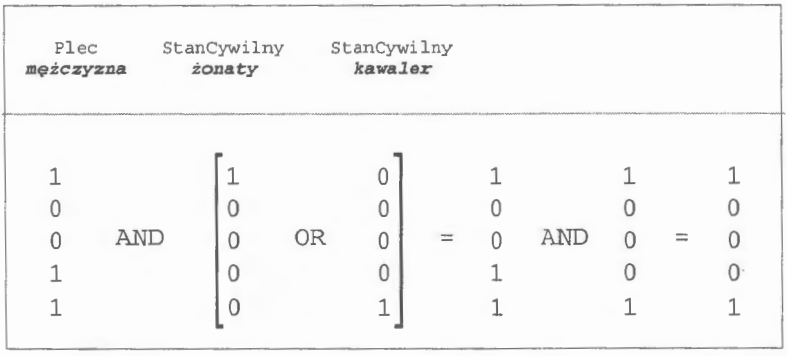
Ostatecznie dla omawianej przykładowej tabeli możemy zbudować dwa osobne indeksy, których finalna postać została zaprezentowana w tab. 3.

Tablica 3. Indeksy bitmapowe dla kolumn `Plec` i `StanCywilny`

NumerPrac	K^{Plec}	$K^{StanCywilny}$
...	...	...
101	1 0	1 0 0 0
112	0 1	0 1 0 0
134	0 1	0 1 0 0
158	1 0	0 0 1 0
160	1 0	0 0 0 1
...	...	...

Jeśli teraz zażądamy wyszukania *"wszystkich mężczyzn żonatych i kawalerów"*, indeks bitmapowy okaże się bardzo efektywny, gdyż będzie wymagał wykonania jedynie kilku prostych operacji bitowych, tak jak zaprezentowano to na rys. 2. I tutaj właśnie objawia się największa siła indeksów bitmapowych, gdzie wiersze, spełniające tylko część warunków zapytania, ale nie spełniające wszystkich, są odfiltrowywane zanim jeszcze nastąpi bezpośredni dostęp do tablicy. Taki mechanizm działania potrafi wręcz dramatycznie skrócić czas wykonania

zapytania. Dzięki niemu również, w stosunku do indeksów bitmapowych nie stosuje się reguły 80/20, która mówi, iż indeks powinien zostać użyty jedynie wtedy, gdy pobierane jest 20% lub mniej wierszy tabeli, zaś pełne skanowanie, jeżeli pobiera się ich więcej. Niebagatelne znaczenie ma także fakt, iż wykonanie warunków z klauzuli WHERE opiera się jedynie na prostych operacjach AND i OR które mogą z powodzeniem być przetwarzane w sposób równoległy za pomocą wielopotokowych jednostek operujących na 32 bądź 64-bitowych słowach, występujących we wszystkich współczesnych procesorach.



Rysunek 2. Wykonanie zapytania z wykorzystaniem indeksu bitmapowego

Jeśli chodzi o zapotrzebowanie na zasoby, należy zwrócić uwagę, że indeksy zbudowane w oparciu o drzewa binarne (ang. *B-tree*) są zazwyczaj dość kosztowne, a indeks taki niekiedy potrafi być kilkukrotnie większy niż dane w tabeli. W tym samym czasie indeks bitmapowy stanowi zazwyczaj ułamek tego rozmiaru, o wielkości zależnej od mocy indeksowanego zbioru. Dla kolumny o 10 mln wierszy i 4 wariantach wartości indeks bitmapowy będzie wymagał użycia $4*10000000/8/1024/1024=4,77$ *MB* podczas gdy zapamiętanie wszystkich identyfikatorów wierszy w indeksie *b-tree* pochłonie co najmniej 8 razy więcej przestrzeni (przy wykorzystaniu 32-bitowych RID). Tak niewielki rozmiar indeksu bitmapowego pozwala na zachowanie go w pamięci podręcznej, co znacząco wpływa na prędkość jego przetwarzania, poprzez zmniejszoną ilość operacji I/O. Na tę ostatnią cechę wpływa również fakt, iż indeks bitmapowy zwraca identyfikatory wierszy zgodnie z ich kolejnością dodawania do tabeli, a więc (z dużym prawdopodobieństwem) również ich fizycznym uporządkowaniem na dysku, co oczywiście owocuje zwiększoną prędkością odczytu.

Warto wspomnieć również o tym, że indeksy bitmapowe mogą zostać użyte także dla warunków opartych o wartości NULL (które również podlegają tutaj indeksowaniu), a także złączeń różnowartościowych.

4. Podsumowanie i wnioski

Indeksy bitmapowe charakteryzują się dużą przydatnością w relacyjnych systemach baz danych ze szczególnym uwzględnieniem HD, które cechują się bardzo dużymi ilościami informacji i zapytaniami wydawanymi z marszu, a jednocześnie bardzo niewielką ilością operacji modyfikacji danych (DML).

W takich zastosowaniach indeksy bitmapowe zapewniają:

- skrócony czas odpowiedzi dla większości zapytań *ad hoc*,
- zmniejszone zapotrzebowania na przestrzeń dyskową i pamięć w stosunku do innych metod indeksowania,
- bardzo znaczące zwiększenie wydajności nawet dla serwerów o stosunkowo niedużej ilości procesorów i pamięci wykorzystywanych podczas równoległego przetwarzania zapytań,
- wydajną obsługę indeksu podczas operacji modyfikacji danych.

Najlepsze rezultaty stosowania indeksów bitmapowych osiąga się, wykorzystując je dla kolumn, których udział wartości unikalnych w stosunku do całkowitej ilości wierszy w tabeli wynosi 1%.

Indeksy bitmapowe tracą znacząco na wydajności w środowiskach, w których wykonywana jest znaczna ilość operacji modyfikujących dane (`INSERT`, `UPDATE`, `DELETE`), a zatem nie nadają się do zastosowania w systemach OLTP. Należy być również ostrożnym podczas używania optymalizatora bazującego na regułach, gdyż w takich przypadkach indeksy bitmapowe nie są brane pod uwagę. Użycie ograniczenia `NOT NULL` oraz typów danych o stałej długości, zmniejsza rozmiar indeksu bitmapowego, więc warto to rozważyć podczas projektowania bazy.

Literatura

Chaudhuri S., Dayal U. (1997) An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, **26** (1), 65-74.

Inmon W.H. (1993) *Building the Data Warehouse*. John Wiley & Sons.

Colliat G. (1996) OLAP, Relational and multimedimensional database system. *ACM SIGMOD Record*, **25** (3), 64-69.

O'Neil P., Graefe G. (1995) Multi-table joins through bitmapped join indices. *ACM SIGMOD Record*, **24** (3), 8-11.

ISSN 0208-8028
ISBN 83-85847-92-8

W celu uzyskania bliższych informacji i zakupu dodatkowych egzemplarzy
prosimy o kontakt z Instytutem Badań Systemowych PAN
ul. Newelska 6, 01-447 Warszawa
tel. 837-35-78 w. 241 e-mail: biblioteka@ibspan.waw.pl