STUDIA INFORMATICA 2004
Volume 25 Number 1 (57)

Jacek FRĄCZEK
Politechnika Śląska, Instytut Informatyki

OPTYMALIZACJA OPERACJI ZŁĄCZENIA W REALIZACJI SYSTEMU BEZPIECZEŃSTWA HURTOWNI DANYCH

Streszczenie. Realizacja mechanizmów kontroli dostępu na poziomie wierszy w hurtowniach danych typu ROLAP, z wykorzystaniem perspektyw i tablicy praw, wymaga wykonania operacji złączenia pomiędzy tablicą faktów i tablicą praw. Wykonanie operacji złączenia można usprawnić tworząc dodatkowe struktury indeksowe. Zaproponowany w pracy zestaw indeksów *MultiColumn* pozwala zwiększyć wydajność realizacji zapytań kierowanych do zabezpieczonej hurtowni danych.

Słowa kluczowe: hurtownia danych, system bezpieczeństwa, kontrola dostępu na poziomie wierszy, operacja złączenia, indeksy, *MultiColumn Join Index*

OPTIMIZATION OF A JOIN OPERATION IN IMPLEMENTATION OF A DATA WAREHOUSE SECURITY SYSTEM

Summary. Implementation of row-level access control mechanisms, in ROLAP data warehouses, with the use of database views and a security table, requires a join operation between fact and security tables. The efficiency of this operation may be improved through the creation of additional index structures. The hereby-proposed set of *MultiColumn* indices allows to increase the performance of queries sent to a secured data warehouse.

Keywords: data warehouse, security system, row-level access control, join operation, indices, *MultiColumn Join Index*

1. Wstęp

Realizacja wydajnie działającego systemu zabezpieczającego dostęp do informacji w hurtowni danych jest zagadnieniem złożonym [1,2]. Projektanci systemu muszą wypracować

kompromis pomiędzy wymaganiami założonej polityki bezpieczeństwa a koniecznością zapewnienia odpowiedniej funkcjonalności, wydajności pracy i minimalizacji kosztów utrzymania hurtowni. Problem wydajności działania systemu zabezpieczonego ma szczególne znaczenie w przypadku dużych, centralnych hurtowni danych, które zarządzają znacznymi ilościami informacji, udostępnianymi licznej społeczności użytkowników.

W przypadku hurtowni danych wykorzystujących relacyjną bazę danych (hurtownie typu ROLAP, ang. *Relational OnLine Analytical Processing*, opisane szerzej w [1,3,21]) najbardziej odpowiednia jest realizacja procedur bezpieczeństwa na poziomie bazy danych [2]. Systemy zarządzania bazami danych oferują różnorodne mechanizmy kontroli dostępu do informacji. Podstawową techniką jest mechanizm uprawnień [4] oparty na wykorzystaniu poleceń GRANT/REVOKE [33]. W przypadku realizacji złożonej polityki bezpieczeństwa, zakładającej kontrolę dostępu na poziomie wierszy (pojedynczych rekordów), można wykorzystać m.in.: perspektywy (m.in.: [5,11]), prywatne, wirtualne bazy danych [6] lub etykiety bezpieczeństwa [7]. Mechanizmy te pozwalają na implementację podstawowych dla systemu bazodanowego typów kontroli: uznaniowego (ang. *discretionary*) [4,9] oraz obowiązkowego (ang. *mandatory*) [9,10]. Przykładową realizację systemu zabezpieczeń w oparciu o perspektywy i wirtualne, prywatne bazy danych przedstawiono w [11].

Jeżeli do zapisu uprawnień użytkowników wykorzystano tablicę praw, to w celu wyznaczenia podzbioru informacji, do którego ma dostęp pojedynczy użytkownik, należy dokonać złączenia tej tablicy z tablicą przechowującą dane (w przypadku hurtowni danych – z tablicą faktów lub wymiarów)[2]. W celu efektywnego wykonania operacji złączenia można wykorzystać wydajne algorytmy złączeń (np. *hybrid-hash join*), indeksowanie (szczególnie indeksy złączeniowe) lub też zastosować mechanizm zmaterializowanych perspektyw, pozwalający na wcześniejsze wyznaczenie i zapamiętanie wyniku zapytania. Analizę wydajności działania tych metod przy występowaniu modyfikacji relacji bazowych zawiera [23].

Indeksy są jednym z podstawowych mechanizmów wykorzystywanych w procesie optymalizacji wykonywania zapytań przez bazę danych [12,13,14,15]. Szczególny rozwój technik indeksujących związany jest z rozpowszechnieniem systemów hurtowni danych i potrzebą wydajnej realizacji charakterystycznych dla tych systemów zapytań wielowymiarowych. Wśród indeksów wspierających operacje tego typu można wyróżnić m.in.: indeks bitmapowy (ang. *bitmap*), projekcyjny (ang. *projection*), bitowo-segmentowy (ang. *bit-sliced*), złączenia gwiaździstego (ang. *star join*), bitmapowy złączeniowy (ang. *bitmap join*), opisane m.in. w [15,16,17,18,19,20,29]. W dalszej części pracy przedstawiono propozycję nowego zestawu indeksów *MultiColumn*, który pozwala zwiększyć wydajność

realizacji zapytań kierowanych do hurtowni danych zabezpieczonej z wykorzystaniem perspektyw.

Indeksy *MultiColumn* są indeksami złączeniowymi o szczególnym typie funkcjonalności. Indeksy tego typu można deklarować dla tablic, dla których operacja złączenia wiąże wiele kolumn jednej tablicy z pojedynczą kolumną tablicy drugiej. W rozwiązaniu tym kolumna pojedyncza zbiorczo przechowuje wartości należące do różnych klas atrybutów. Dane poszczególnych klas są rozróżniane wartością dodatkowego atrybutu. Atrybut ten wykorzystywany jest we wszystkich operacjach odwołujących się do danych określonej klasy. Indeks wspomagający wykonanie operacji złączenia może uwzględniać również dodatkowe warunki związane z przechowywanymi w tablicach danymi.

Uzyskana funkcjonalność jest szczególną cechą rodziny indeksów *MultiColumn* i stanowi oryginalny dorobek prezentowanej pracy. Zastosowanie indeksu w praktyce wymaga specyficznego sposobu deklaracji indeksu oraz podania algorytmu jego budowy (w sensie procesu tworzenia). Indeks nie ogranicza natomiast możliwości zastosowania różnych, fizycznych struktur przechowywania jego danych. Wybór konkretnego sposobu implementacji (b-drzewo, bitmapa, ew. inne) wpływa na sposób utworzenia indeksu, jego utrzymania (aktualizacji) oraz możliwości jego późniejszego wykorzystania.

Prezentowany w pracy praktyczny przykład zastosowania indeksu typu *MultiColumn* odnosi się do tablicy praw wykorzystywanej przez obiekty systemu zabezpieczeń. Tworzony indeks ma charakter indeksu złączeniowego z możliwością wydajnego wyszukiwania danych według zadanych kryteriów (bezpieczeństwa). Pracę uzupełniają badania doświadczalne.

2. Zabezpieczenie informacji z użyciem perspektyw

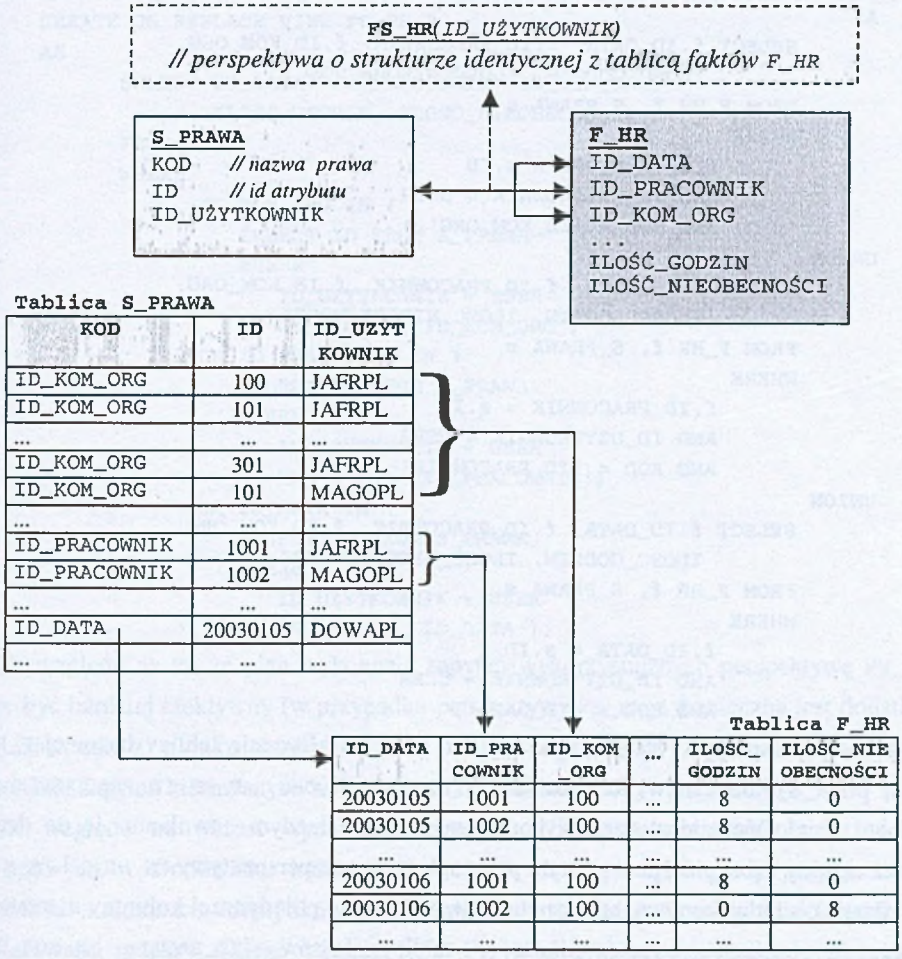
Fundamentalnym wzorcem projektowym wykorzystywanym przy tworzeniu hurtowni danych typu ROLAP jest schemat gwiazdy (ang. *star*) i pochodny schemat płatka śniegu (ang. *snowflake*) [3,22]. Jedną z podstawowych metod kontroli dostępu do danych przechowywanych w obiektach tych schematów (tablicach faktów i wymiarów) jest wykorzystanie perspektyw [2]. Perspektywy zbudowane na bazie podstawowych obiektów hurtowni umożliwiają stworzenie systemu udostępniającego tylko autoryzowane podzbiory danych. Mechanizm ten pozwala uzyskać bardziej szczegółowy poziom kontroli niż w przypadku przywilejów nadawanych z użyciem instrukcji GRANT.

W przypadku bardziej złożonych polityk bezpieczeństwa, wymagających kompleksowej kontroli dostępu na poziomie wierszy danych, prawa nadane użytkownikom można zapisać w formie tablicy praw [2]. Wykorzystanie tablicy praw w definicjach perspektyw zabezpieczających pozwala na budowę jednego, uniwersalnego zestawu perspektyw,

dostępnego dla wszystkich użytkowników. Szczególną zaletą rozwiązania korzystającego z tablicy praw jest możliwość modyfikacji posiadanych przez użytkownika uprawnień bez konieczności zmian definicji obiektów bazodanowych odpowiedzialnych za zabezpieczenie danych (perspektyw, procedur). W istniejących systemach zarządzania bazami danych zmiany definicji obiektów bazodanowych powodują konieczność rekompilacji tych obiektów oraz obiektów opartych na obiektach zmienianych (korzystających z modyfikowanej perspektywy lub procedury). Proces ten może powodować kaskadowe unieważnienie (tzw. inwalidację), a następnie rekompilację całego szeregu obiektów. Rozwiązanie wykorzystujące tablicę praw pozwala uniknąć wystąpienia takiej sytuacji kosztem ograniczenia możliwości definiowania pewnych bardziej złożonych polityk bezpieczeństwa. W przypadku tablicy praw uprawnienia definiuje się przez jawną specyfikację identyfikatorów elementów dostępnych dla danego użytkownika. Ze względu na możliwość posiadania praw do elementów należących do różnych wymiarów hurtowni danych użytkownik może uzyskać dostęp do różnorodnych (zawierających się, pokrywających się częściowo lub rozłącznych) podzbiorów danych. Wynikowy zbiór danych powstaje poprzez wykonanie pewnej operacji na uzyskanych podzbiorach. Rodzaj wykonywanej operacji określa się w definicjach widoków zabezpieczających. W praktycznej realizacji systemu bezpieczeństwa najczęściej wybiera się operator sumy zbiorów (w odniesieniu do zbiorów uzyskiwanych jako wynik polecenia SELECT używa się operatora UNION). Inne z możliwych rozwiązań, np. zastosowanie operacji iloczynu zbiorów (INTERSECT), czy nawet wykonanie różnych operacji na uzyskiwanych podzbiorach danych nie są w zasadzie wykorzystywane w praktyce.

Rysunek 1 przedstawia fragment hurtowni danych: tablicę faktów F_HR, tablicę praw S_PRAWA oraz zbudowaną na ich podstawie perspektywę FS_HR (przykład zaczerpnięty z [11]). Użytkownik ma dostęp do danych tablicy F_HR wyłącznie poprzez perspektywę FS_HR. W tablicy praw znajdują się informacje pozwalające użytkownikowi (ID_UŻYTKOWNIK) na dostęp do zestawu danych określonego nazwą (KOD). W zależności od potrzeb można definiować różne zestawy danych, identyfikowane odpowiednio dobranymi nazwami. Określając możliwości dostępu do danych na podstawie wartości elementów wymiarów hurtowni danych (ograniczenia typu *rok=2003*, *dzień=20030501*, *pion=Produkcja*, *pracownik=Kowalski*), można przyjąć regułę, że nazwa zestawu danych odpowiada nazwie zabezpieczanego wymiaru (np. *Czas*, *Struktura organizacyjna*, *Pracownik*) lub nazwie identyfikatora tego wymiaru w tablicy faktów (np. *id_data*, *id_kom_org*, *id_pracownik* – tę właśnie regułę zastosowano w dalszej części pracy). Nazwa określa grupę identyfikatorów (kolumna ID) elementów wymiarów, do których użytkownik ma przypisane prawa. Podobnie jak w przypadku wymiarów, identyfikatory ID można zastosować do zabezpieczenia tablicy faktów, gdzie określają one dostępny dla użytkownika podzbiór danych (według określonego

KODem wymiaru). Przy odwołaniu się do perspektywy FS_HR przez użytkownika o identyfikatorze ID_UŻYTKOWNIK, z tablicy faktów F_HR wybierane są tylko te rekordy, których odpowiednie identyfikatory wymiarów znajdują się w tablicy praw S_PRAWA (kolumna ID). Identyfikatory wymiaru *czasu* – *id_data* są wyszukiwane wśród identyfikatorów ID oznaczonych KODem ID_DATA, identyfikatory wymiaru *struktury organizacyjnej* – *id_kom_org* są wyszukiwane wśród identyfikatorów ID oznaczonych KODem ID_KOM_ORG, a identyfikatory wymiaru *pracownik* – *id_pracownik* są wyszukiwane wśród identyfikatorów ID oznaczonych KODem ID_PRACOWNIK. Tablicę praw S_PRAWA wypełnia, zazwyczaj z pomocą odpowiedniej aplikacji, pracownik odpowiedzialny za zarządzanie systemem bezpieczeństwa.



Rys. 1. Zabezpieczenie tablicy faktów F_HR z użyciem tablicy praw S_PRAWA i perspektywy FS_HR

Fig. 1. Securing a fact table F_HR by the use of a security table S_PRAWA and a database view FS_HR

Przy definiowaniu założeń implementacyjnych dla systemu bezpieczeństwa przyjęto również, że w tablicy praw znajdują się identyfikatory elementów wyłącznie najniższych poziomów hierarchii wymiarów (czyli np. prawa nadane na poziomie identyfikatorów *lat* schematu płatka śniegu są transformowane do odpowiadającej im postaci praw na poziomie identyfikatorów dat dziennych).

Dla opisanych struktur tablic faktów i praw w pracy [11] podano dwie definicje perspektyw zabezpieczających dostęp do tablicy faktów F_HR. Perspektywa FS_HR_1 wykorzystuje zapytania połączone operatorem UNION, a perspektywa FS_HR_2 (definicja podana dalej) stosuje konstrukcję z zapytaniem zagnieżdżonym:

```
CREATE OR REPLACE VIEW FS_HR_1
AS
      SELECT f.ID_DATA, f.ID_PRACOWNIK, f.ID_KOM_ORG,
             ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
      FROM F_HR f, S_PRAWA s
      WHERE
            f.ID_KOM_ORG = s.ID
            AND ID_UZYTKOWNIK = USER
            AND KOD = 'ID_KOM_ORG'
UNION
      SELECT f.ID_DATA, f.ID_PRACOWNIK, f.ID_KOM_ORG,
             ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
      FROM F_HR f, S_PRAWA s
      WHERE
            f.ID_PRACOWNIK = s.ID
            AND ID_UZYTKOWNIK = USER
            AND KOD = 'ID_PRACOWNIK'
UNION
      SELECT f.ID_DATA, f.ID_PRACOWNIK, f.ID_KOM_ORG,
             ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
      FROM F_HR f, S_PRAWA s
      WHERE
            f.ID_DATA = s.ID
            AND ID_UZYTKOWNIK = USER
            AND KOD = 'ID_DATA';
```

Definicja perspektywy FS_HR_1 w zawiera potrójne złączenie tablicy bazowej F_HR z tablicą praw S_PRAWA – według kolumn F_HR.ID_KOM_ORG, F_HR.ID_PRACOWNIK oraz F_HR.ID_DATA. Operacje złączeń wykonywane są przy każdym odwołaniu się do danych poprzez obiekty zabezpieczone – w tym przypadku poprzez perspektywy FS_HR_1 i FS_HR_2. Specyficzną cechą wykonywanego zapytania jest złączenie pojedynczej kolumny S_PRAWA.ID z różnymi kolumnami identyfikatorów tablicy faktów (ID_DATA, ID_KOM_ORG, ID_PRACOWNIK). Występujące operacje złączenia reprezentują tzw. półzłączenia (ang. *semi-join*), gdyż użytkownik końcowy uzyskuje dostęp wyłącznie do danych z tablicy F_HR.

Tablica S_PRAWA wykorzystywana jest jedynie jako źródło dodatkowych warunków selekcji danych dla operacji wykonywanych na tablicy F_HR.

Definicja perspektywy FS_HR_2 ma format zapytania powstałego w wyniku zastosowania – używanej w hurtowniach danych – techniki optymalizacyjnej, zwanej transformacją gwiaździstą (ang. *star transformation*) [24]. Transformacja ta jest charakterystyczna dla zapytań wielowymiarowych i pozwala na efektywne wykorzystanie indeksów bitmapowych zdefiniowanych na kolumnach atrybutów biorących udział w zapytaniu. Transformację można również wykonać w obecności indeksów *b-tree*, które w trakcie wykonywania zapytania są przekształcane na reprezentację bitmapową – technika ta, zwana *dynamic bitmap indexes*, jest dostępna w niektórych systemach zarządzania bazami danych [30].

```
CREATE OR REPLACE VIEW FS_HR_2
AS
      SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
             ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
      FROM F_HR
      WHERE
            ID_KOM_ORG IN (
               SELECT ID FROM S_PRAWA
               WHERE
                  ID_UZYTKOWNIK = USER
                  AND KOD = 'ID_KOM_ORG')
            OR ID_PRACOWNIK IN (
               SELECT ID FROM S_PRAWA
               WHERE
                  ID_UZYTKOWNIK = USER
                  AND KOD = 'ID_PRACOWNIK')
            OR ID_DATA IN (
               SELECT ID FROM S_PRAWA
               WHERE
                  ID_UZYTKOWNIK = USER
                  AND KOD = 'ID_DATA');
```

Ze względu na to, że plan wykonania zapytań wykorzystujących perspektywę FS_HR_2 może być bardziej efektywny (w przypadku perspektywy FS_HR_1 konieczna jest dodatkowa faza usuwania z wyniku duplikatów), w dalszej części pracy zajęto się optymalizacją wykonania zapytań kierowanych wyłącznie do tej struktury, nazywanej dalej FS_HR.

Ze względu na wzrost wydajności, płynący z zastosowania technologii indeksów bitmapowych w środowisku hurtowni danych, w dalszej części pracy przyjęto, że na wszystkich kolumnach identyfikatorów wymiarów występujących w tablicy faktów F_HR zdefiniowano indeksy takiego właśnie typu. Przykład praktycznej analizy porównawczej wykonania zapytań opartych na indeksach bitmapowych, przy wykorzystaniu transformacji gwiaździstej oraz wykonania zapytań z użyciem indeksów *b-tree*, można znaleźć w [25].

3. Struktury indeksowe *MultiColumn*

W instrukcji selekcji danych mogą się pojawić różnego typu warunki ograniczające wynikowy zbiór rekordów. Na potrzeby prezentowanej pracy ogólny schemat zapytania kierowanego do zabezpieczonej hurtowni danych zapisano następująco:

```
SELECT <identyfikatory wymiarów>, <fakty>
FROM <tablica faktów>
WHERE <warunki bezpieczeństwa>
[AND <warunki OLAP> ];
```

Uwzględniając definicję perspektywy FS_HR, powyższe zapytanie można przekształcić i przedstawić symbolicznie w następującej formie:

```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE
      (ID_KOM_ORG IN (
         SELECT ID FROM S_PRAWA
         WHERE
            ID_UZYTKOWNIK = USER
            AND KOD = 'ID_KOM_ORG')
      OR ID_PRACOWNIK IN (
         SELECT ID FROM S_PRAWA
         WHERE
            ID_UZYTKOWNIK = USER
            AND KOD = 'ID_PRACOWNIK')
      OR ID_DATA IN (
         SELECT ID FROM S_PRAWA
         WHERE
            ID_UZYTKOWNIK = USER
            AND KOD = 'ID_DATA'))
      [AND <warunki OLAP> ];
```

Fraza warunków bezpieczeństwa jest stała i występuje w każdym zapytaniu kierowanym do tablicy bazowej F_HR, natomiast fraza warunków OLAP (ang. *OnLine Analytical Processing*) jest opcjonalnie specyfikowana w trakcie pracy przez użytkownika. Ze względu na stałość frazy bezpieczeństwa można zbudować struktury wspomagające wykonanie tej części zapytania. Struktury te w naturalny sposób wspomagałyby wykonywanie zapytań nie zawierających dodatkowych warunków (zapytania typu 1), np.:

```
SELECT MAX(KWOTA) FROM FS_HR;
SELECT COUNT (DISTINCT(ID_PRACOWNIK)) FROM FS_HR;
SELECT STDDEV(ILOSC_GODZIN) FROM FS_HR;
```

a nawet

```
SELECT * FROM FS_HR;
```

w przypadku, gdy użytkownik ma prawo do silnie ograniczonej liczby rekordów. Pytania tego typu występują jednak rzadko w hurtowniach danych. Bardziej złożony charakter zapytań z dodatkową frazą warunków OLAP (zapytania typu 2) wymaga utworzenia nieco innych struktur optymalizujących niż w przypadku zapytania typu 1. Realizację obu opisanych typów zapytań wspomaga zaproponowany poniżej zestaw indeksów *MultiColumn*.

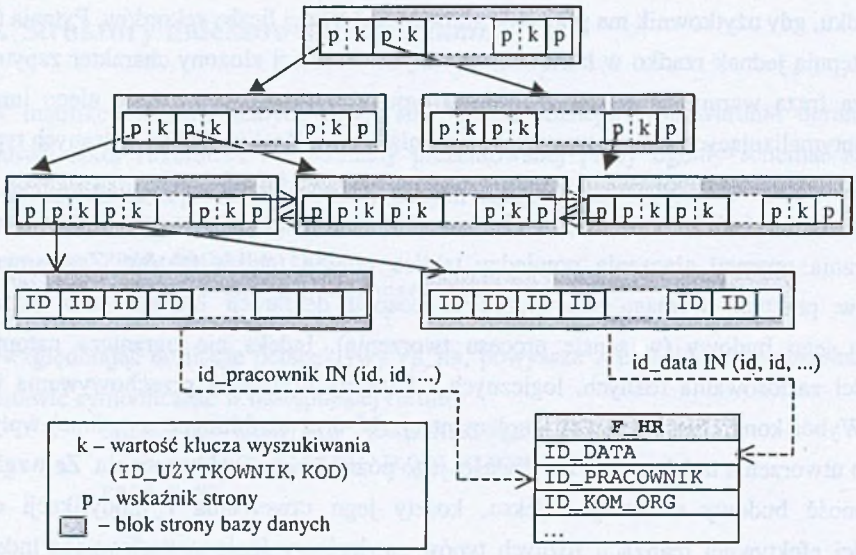
Grupa struktur indeksowych *MultiColumn* została zaprojektowana w celu przyspieszenia wykonywania operacji złączenia pomiędzy tablicą praw i tablicą faktów. Zastosowanie indeksu w praktyce wymaga specyficznego sposobu deklaracji indeksu oraz podania algorytmu jego budowy (w sensie procesu tworzenia). Indeks nie ogranicza natomiast możliwości zastosowania różnych, logicznych i fizycznych struktur przechowywania jego danych. Wybór konkretnego sposobu implementacji (b^+-drzewo, bitmapa, ew. inne) wpływa na sposób utworzenia indeksu oraz możliwości jego późniejszego wykorzystania. Ze względu na złożoność budowy struktury indeksu, koszty jego utworzenia i modyfikacji oraz możliwości efektywnej realizacji różnych typów zapytań wyróżniono następujące indeksy grupy *MultiColumn*:

- indeks podstawowy – *Basic MultiColumn Index* – o najprostszej strukturze,
- indeksy typu złączeniowego:
  - złączeniowy typu I – *MultiColumn Join Index I*,
  - złączeniowy typu II –*MultiColumn Join Index II* – o potencjalnie najwyższej wydajności przy przetwarzaniu zapytań bez dodatkowych warunków selekcji,
  - złączeniowy, bitmapowy typu I – *MultiColumn Bitmap Join Index I*,
  - złączeniowy, bitmapowy typu II – *MultiColumn Bitmap Join Index II* – o potencjalnie najwyższej wydajności przy przetwarzaniu zapytań z dodatkowymi warunkami selekcji.

3.1. Basic MultiColumn Index

Indeks *Basic MultiColumn* jest zakładany na tablicy praw. Zadaniem indeksu jest minimalizacja czasu wyszukiwania praw (identyfikatorów `ID`) danego typu (`KOD`), przypisanych do danego użytkownika (`ID_UŻYTKOWNIK`). W strukturze indeksu, pokazanej schematycznie na rysunku 2, można wyróżnić:

- zrównoważoną strukturę drzewiastą typu *b^+-tree*, służącą do wyszukiwania kluczy (`ID_UŻYTKOWNIK`, `KOD`),
- listy identyfikatorów, dołączone do poziomu liści struktury drzewiastej i przechowujące wartości `ID` przypisane do odpowiednich kluczy wyszukiwania.



Rys. 2. Struktura indeksu *Basic MultiColumn*
Fig. 2. Structure of the *Basic MultiColumn* index

Struktura drzewa wyszukiwania ma budowę zgodną ze schematem implementacji jednowymiarowgo, wielokolumnowego indeksu typu b^+-*tree* [12,31]. Informacje znajdujące się na kolejnych poziomach drzewa przechowywane są w listach bloków bazodanowych, zawierających dwuelementowe struktury o budowie: klucz wyszukiwania k i wskaźnik następnego, niższego poziomu bloków informacji p (dla zachowania czytelności rysunku, zrezygnowano z przypisania indeksów poszczególnym kluczom k_{ij} oraz wskaźnikom p_i). Liczba poziomów drzewa wyszukiwania zdeterminowana jest liczbą indeksowanych atrybutów oraz krotnością ich wartości, a także rozmiarami strony bazodanowej, klucza k i wskaźnika p. Ze względu na różne możliwości logicznej i fizycznej budowy struktur indeksu, w szczególności możliwa jest implementacja indeksu w postaci indeksu jednowymiarowego dwukolumnowego (tak jak na rys.1) lub w formie różnych odmian indeksu wielowymiarowego (np. wielokrotnego klucza, ang. *multiple-key*, lub *kd*-drzewa, ang. *kd-tree*, [15]).

W przypadku realizacji zapytania związanego z tablicą faktów indeks *Basic MultiColumn* wspomaga wykonanie podzapytań zawartych w definicji perspektywy zabezpieczającej. Podany poniżej przykład podzapytania dotyczy wymiaru struktury organizacyjnej:

```
SELECT ID FROM S_PRAWA
WHERE
      ID_UZYTKOWNIK = USER
      AND KOD = 'ID_KOM_ORG';
```

Dla każdej pary kluczy (`ID_UZYTKOWNIK, KOD`), indeks *Basic MultiColumn* zwraca listę wszystkich identyfikatorów `ID` przypisanych do tej pary. Odczytane z indeksu listy

identyfikatorów wykorzystuje się do wyznaczenia rozwiązania dla warunków fraz "WHERE ID_<nazwa_wymiaru> IN (<lista ID>)", tworzących definicję perspektywy FS_HR. Na podstawie uzyskanych identyfikatorów ID dokonywana jest selekcja danych z tablicy faktów. Przypisując podzapytaniu związanemu z wymiarem *struktury organizacyjnej*, uzyskaną z indeksu, listę identyfikatorów ID_KOM_ORG_LIST, podzapytaniu związanemu z wymiarem *czasu* – listę identyfikatorów ID_DATA_LIST, a podzapytaniu związanemu z wymiarem *pracownika* – listę identyfikatorów ID_PRACOWNIK_LIST, możemy zapytanie typu 1 do tablicy faktów zapisać symbolicznie w następujący sposób:

```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE
      ID_KOM_ORG IN ( ID_KOM_ORG_LIST )
      OR ID_PRACOWNIK IN ( ID_PRACOWNIK_LIST )
      OR ID_DATA IN ( ID_DATA_LIST );
```

Zastosowanie podstawowego indeksu *MultiColumn* pozwala uniknąć wykonywania klasycznego złączenia pomiędzy tablicami F_HR i S_PRAWA.

Analizując przedstawiony sposób implementacji indeksu *Basic MultiColumn* (jak również i pozostałych indeksów tej grupy) jako struktury drzewiastej (rys.2) należy zauważyć, że jest on jednym z wielu możliwych. Część struktury indeksu związaną z wyszukiwaniem klucza można bowiem zbudować w oparciu o inne rozwiązania, na przykład wykorzystując funkcję mieszającą [15].

3.2. MultiColumn Join Index I

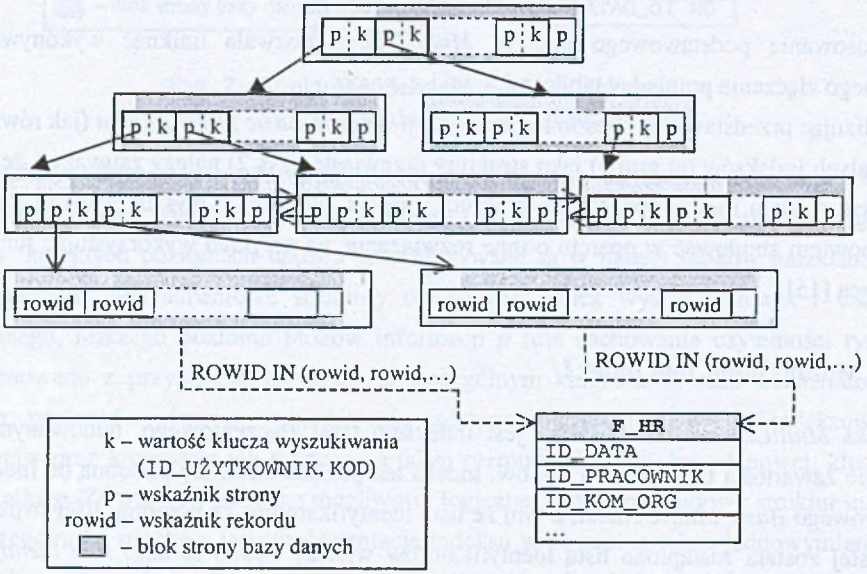
Indeks *MultiColumn Join Index I* jest indeksem typu złączeniowego, budowanym na podstawie zawartości tablic praw i faktów. Indeks ten posiada strukturę podobną do indeksu podstawowego *Basic MultiColumn*, z tym że lista identyfikatorów ID poziomu liści struktury drzewiastej została zastąpiona listą identyfikatorów wierszy rowid (z ang. *row identifier*) indeksowanej tablicy faktów (rys.3). Ponieważ identyfikatory rowid odpowiadają fizycznym adresom danych, to pozwalają one na uzyskanie najszybszego z możliwych dostępu do danych pojedynczego rekordu [27].

Dla każdej pary kluczy (ID_UZYTKOWNIK, KOD) indeks *MultiColumn Join Index I* zwraca listę wszystkich identyfikatorów rekordów rowid tablicy faktów, udostępnionych danemu użytkownikowi w ramach danego prawa. Na podstawie uzyskanych identyfikatorów rowid dokonywana jest selekcja danych z tablicy faktów. Przypisując podzapytaniu związanemu z wymiarem *struktury organizacyjnej*, uzyskaną z indeksu, listę identyfikatorów ROWID_ID_KOM_ORG_LIST, podzapytaniu związanemu z wymiarem *czasu* – listę identyfikatorów ROWID_ID_DATA_LIST, a podzapytaniu związanemu z wymiarem *pracownika*

– listę identyfikatorów ROWID_ID_PRACOWNIK_LIST, możemy zapytanie typu 1 do tablicy faktów zapisać symbolicznie w następujący sposób:

```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE
      ROWID IN ( ROWID_ID_KOM_ORG_LIST )
      OR ROWID IN ( ROWID_ID_PRACOWNIK_LIST )
      OR ROWID IN ( ROWID_ID_DATA_LIST );

//lub prościej //(+) jest symbolicznym operatorem łączenia list
//WHERE ROWID IN (
//     ROWID_ID_KOM_ORG_LIST (+)
//     ROWID_ID_PRACOWNIK_LIST (+)
//     ROWID_ID_DATA_LIST );
```

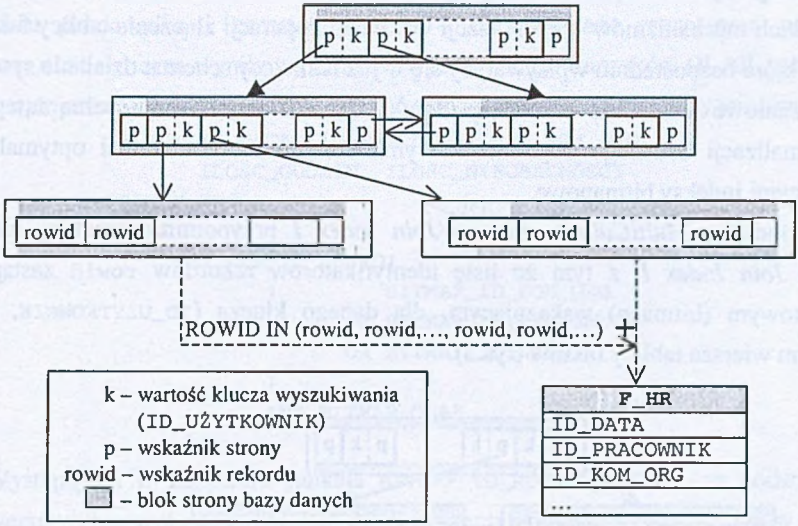


Rys. 3. Struktura indeksu *MultiColumn Join I*
Fig. 3. Structure of the *MultiColumn Join I* index

3.3. MultiColumn Join Index II

Wykorzystanie indeksu *MultiColumn Join* można usprawnić przez wcześniejsze połączenie list identyfikatorów rowid przypisanych do poszczególnych użytkowników, posortowanie listy wynikowej i usunięcie z niej powtarzających się wskaźników. Rezygnacja z rozróżnienia list praw według ich nazw (kolumna KOD) prowadzi do uzyskania pojedynczej listy identyfikatorów rowid dla każdego z użytkowników. Dodatkowo, ze względu na

zmniejszenie struktury drzewa wyszukiwania (teraz zawiera ono wyłącznie klucze ID_UŻYTKOWNIK), uzyskuje się przyspieszenie wyszukiwania żądanej listy. Tak zmodyfikowany indeks nazwano *MultiColumn Join Index II* (rys.4).



Rys. 4. Struktura indeksu *MultiColumn Join II*
Fig. 4. Structure of the *MultiColumn Join II* index

Przyjmując, że wynikowa lista identyfikatorów rowid, uzyskana z indeksu *MultiColumn Join Index II*, nosi nazwę ROWID_LIST, to zapytanie typu 1 można zapisać symbolicznie w następujący sposób:

```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE
      ROWID IN ( ROWID_LIST );
```

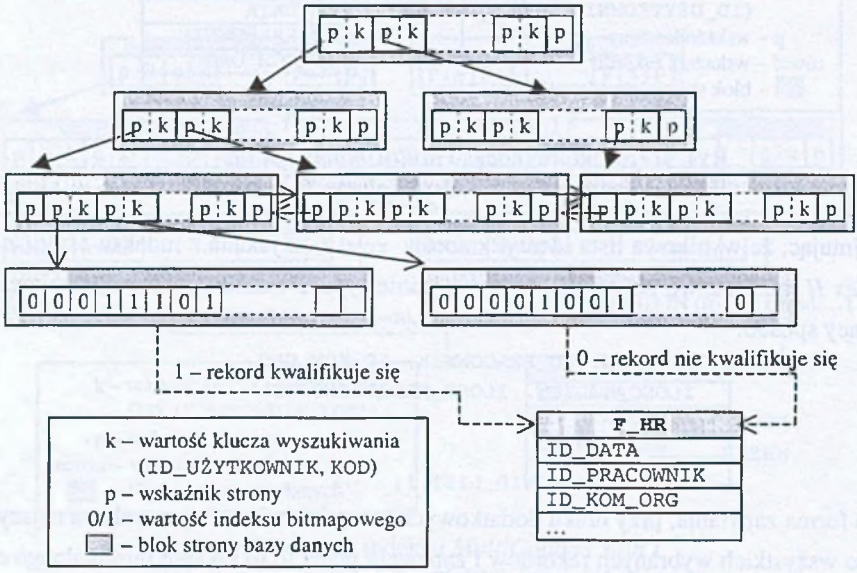
Taka forma zapytania, przy braku dodatkowych warunków OLAP, pozwala na najszybszy dostęp do wszystkich wybranych rekordów i zapewnia przez to uzyskanie minimalnego czasu wykonania zapytania.

3.4. MultiColumn Bitmap Join Index I

Indeksy bitmapowe są strukturami zoptymalizowanymi pod kątem wykonywania operacji zorientowanych na przetwarzanie zbiorów danych. Indeksy tego typu bardzo dobrze sprawdzają się w środowisku hurtowni danych, gdzie operuje się dużymi ilościami danych, przy występowaniu wielu kryteriów selekcji (tu: warunków OLAP). Wykorzystanie indeksów bitmapowych leży również u podstaw techniki optymalizacyjnej, nazywanej transformacją

gwiaździstą (ang. *star transformation*), która polega na modyfikacji formatu zapytania tak, aby można było w prosty sposób wyznaczyć, zgodny z zadanymi warunkami, wynikowy zbiór rekordów [19,24]. Ze względu na bardzo dobre wyniki uzyskiwane poprzez zastosowanie indeksów bitmapowych oraz transformacji gwiaździstej (test m.in w [28]), pożądane jest stworzenie takich mechanizmów optymalizacji wykonania operacji złączenia tablicy faktów i tablicy praw, które bezpośrednio wpisywałyby się w już istniejący schemat działania systemu. Indeksy złączeniowe *MultiColumn Bitmap Join Index I/II* umożliwiają pełną integrację technik optymalizacji badanego złączenia z wymienionymi mechanizmami optymalizacji wykorzystującymi indeksy bitmapowe.

Struktura indeksu *MultiColumn Bitmap Join Index I* przypomina strukturę indeksu *MultiColumn Join Index I*, z tym że listę identyfikatorów rekordów `rowid` zastąpiono wektorem bitowym (bitmapą) wskazującym, dla danego klucza (`ID_UŻYTKOWNIK`, `KOD`), związane z nim wiersze tablicy faktów (rys.5).



Rys. 5. Struktura indeksu *MultiColumn Bitmap Join I*
Fig. 5. Structure of the *MultiColumn Bitmap Join I* index

W trakcie realizacji zapytania wielowymiarowego na tablicy faktów (poprzez perspektywę zabezpieczającą), dla danego użytkownika, ze struktur indeksu *MultiColumn Bitmap Join Index I* pobierane są bitmapy praw przypisanych do poszczególnych wymiarów. Bitmapy te są łączone ze sobą za pomocą operatora bitowego `OR`, a następnie, w przypadku występowania warunków OLAP, są łączone z bitmapami tych warunków operatorem `AND`. Na

podstawie bitmapy wynikowej, poprzez jej konwersję do listy identyfikatorów rowid, uzyskuje się dostęp do końcowego zestawu rekordów.

Zakładając, że z indeksu *MultiColumn Bitmap Join Index I* uzyskano bitmapy: BITMAP_ID_KOM_ORG – dla wymiaru *struktury organizacyjnej*, BITMAP_ID_PRACOWNIK – dla wymiaru *pracownika* i BITMAP_ID_DATA – dla wymiaru *czasu* oraz przyjmując, że BITMAP_OLAP oznacza wynikową bitmapę wyznaczoną z warunków OLAP, można zapytanie wielowymiarowe do tablicy faktów zapisać symbolicznie w następujący sposób:

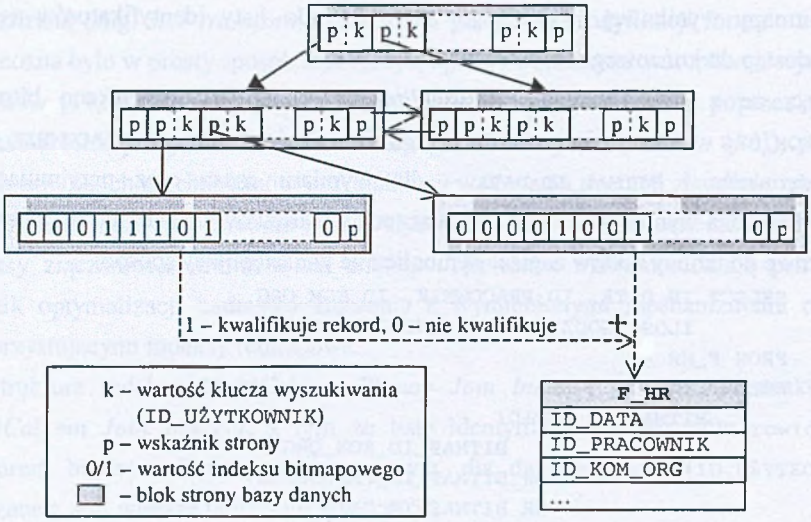
```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE
     BITMAP_TO_ROWID(
          (        BITMAP_ID_KOM_ORG
                OR BITMAP_ID_PRACOWNIK
                OR BITMAP_ID_DATA
          )
          AND BITMAP_OLAP
     );
```

Występująca w zapytaniu funkcja BITMAP_TO_ROWID pozwala, na podstawie wektora bitowego, wyznaczyć odpowiadającą mu listę identyfikatorów rowid rekordów indeksowanej tablicy (operację tę wykonuje jądro systemu zarządzania bazą danych).

Podobnie jak w przypadku indeksu *MultiColumn Join Index I*, również i tu system zarządzania bazą danych musi każdorazowo wyliczać wynikową bitmapę warunków bezpieczeństwa. W odróżnieniu od indeksów *MultiColumn Join Index I/II* indeksy złączeniowe oparte na bitmapach potrafią efektywnie współdziałać z pozostałymi indeksami bitmapowymi zdefiniowanymi na tablicy faktów. Dodatkowo, w przypadku dużych ilości danych o małym zróżnicowaniu, rozmiary indeksów bitmapowych są znacznie mniejsze od rozmiarów indeksów stosujących identyfikatory rowid. Mechanizmy kompresji bitmap czynią występującą różnicę rozmiarów jeszcze większą, a algorytmy pozwalające na przeprowadzanie operacji bitowych na bitmapach skompresowanych pozwalają na dalsze zwiększenie efektywności wykorzystania tego typu indeksów [32].

3.5. MultiColumn Bitmap Join Index II

Indeks *MultiColumn Bitmap Join Index II* jest bitmapowym odpowiednikiem indeksu *MultiColumn Join Index II*. Podobnie jak poprzednik, pomija on rozróżnienie grup praw według kolumny KOD i udostępnia, dla każdego z użytkowników, wynikową bitmapę wierszy tablicy faktów, do których ma on dostęp (rys.6).



Rys. 6. Struktura indeksu *MultiColumn Bitmap Join II*
Fig. 6. Structure of the *MultiColumn Bitmap Join II* index

Zakładając, że z indeksu *MultiColumn Bitmap Join Index II* uzyskano bitmapę BITMAP_BEZP i przyjmując, że BITMAP_OLAP oznacza wynikową bitmapę wyznaczoną z warunków OLAP, można zapytanie wielowymiarowe do tablicy faktów zapisać symbolicznie w następujący sposób:

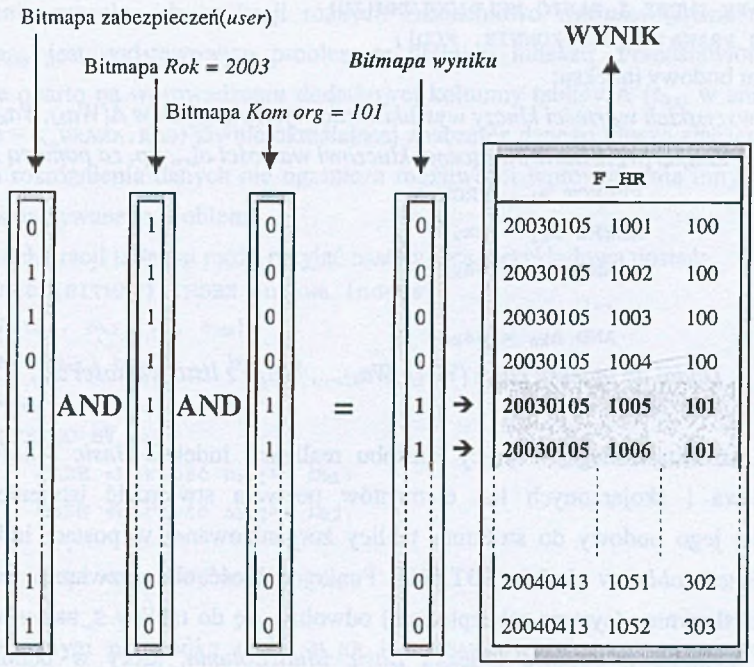
```
SELECT ID_DATA, ID_PRACOWNIK, ID_KOM_ORG,
       ILOSC_GODZIN, ILOSC_NIEOBECNOSCI
FROM F_HR
WHERE BITMAP_TO_ROWID(
                BITMAP_BEZP
            AND BITMAP_OLAP
      );
```

Taka forma zapytania, wykorzystująca indeks *MultiColumn Bitmap Join Index II*, może stanowić najszybszy sposób wykonania zapytania do zabezpieczonej tablicy faktów, przy występowaniu warunków OLAP. Przykładową realizację takiego zapytania z narzuconymi dodatkowymi warunkami na wymiar czasu i struktury organizacyjnej pokazano na rysunku 7.

4. Zagadnienia związane z implementacją indeksów *MultiColumn*

Rodzinę indeksów *MultiColumn* można podzielić, ze względu na sposób definicji i tworzenia indeksów, na dwie grupy:

- indeks podstawowy *Basic MultiColumn*,
- indeksy typu złączeniowego.



Rys. 7. Wyznaczenie wyniku zapytania wielowymiarowego za pomocą indeksu *MultiColumn Bitmap Join II*

Fig. 7. Calculation of the result of a multidimensional query with the use of the *MultiColumn Bitmap Join II* index

Indeksy typu *Basic MultiColumn* można zakładać na tablicach posiadających co najmniej dwie kolumny. Jedna z kolumn indeksowanej tablicy stanowi źródło elementów, które zostają logicznie pogrupowane (i zapisane w postaci list) według klucza wyszukiwania, określonego przez co najmniej jedną z pozostałych kolumn tablicy. W ramach definicji indeksu budowanego na tablicy A należy określić:

1) kolumnę (kolumny) tablicy (a_{k1}, a_{k2},..., a_{kn}) stanowiącą klucz wyszukiwania indeksu,
2) kolumnę tablicy a_{key}, (różną od a_{k1}, a_{k2},..., a_{kn}), której elementy zostaną przypisane do odpowiednich kluczy wyszukiwania.

W procesie tworzenia indeksu każdemu z kluczy wyszukiwania (a_{k1}, a_{k2},..., a_{kn}) przypisana zostaje lista odpowiadających mu elementów kolumny a_{key}.

Format deklaracji indeksu może przyjąć następującą, przykładową postać:

```
CREATE INDEX <nazwa indeksu>(akey)
ON A(ak1, ak2, . . . , akn);
```

W rozważanym przypadku tablicy S_PRAWA polecenie utworzenia indeksu typu *Basic MultiColumn* przybiera następującą postać:

```
CREATE INDEX I_BASIC_MULTICOLUMN(ID)
ON S_PRAWA(ID_UZYTKOWNIK, KOD);
```

Algorytm budowy indeksu:

Dla wszystkich wartości kluczy wyszukiwania występujących w $A(Wa_{k1}, Wa_{k2}, ..., Wa_{kn})${

Znajdź przypisane bieżącemu kluczowi wartości a_{key}*, np. za pomocą zapytania:*

```
SELECT akey FROM A
WHERE ak1 = Wak1
  AND ak2 = Wak2
  ...
  AND akn = Wakn
```

Dodaj do indeksu klucz $(Wa_{k1}, Wa_{k2}, ..., Wa_{kn})$ *z listą wartości* a_{key}

}

Analiza przedstawionego w pracy sposobu realizacji indeksu *Basic MultiColumn* w postaci drzewa i skojarzonych list elementów pozwala stwierdzić istnienie pewnego podobieństwa jego budowy do struktury tablicy zorganizowanej w postaci indeksu (ang. *index organized table*, w skrócie IOT)[26]. Funkcjonalność obu rozwiązań jest również podobna. Użytkownicy (system zabezpieczeń) odwołują się do tablicy S_PRAWA według pól kluczowych zgodnie z budową indeksu *Basic MultiColumn*, który w pełni duplikuje wszystkie dane znajdujące się w tablicy S_PRAWA. Analogia ta wskazuje na korzyści, jakie można uzyskać w tradycyjnych systemach bazodanowych poprzez implementację tablicy S_PRAWA jako tablicy zorganizowanej w postaci indeksu. Deklarując tablicę S_PRAWA jako tablicę typu IOT, uporządkowaną według klucza ID_UZYTKOWNIK, KOD (z ewentualną kompresją tych kolumn), uzyskujemy podobne grupowanie identyfikatorów jak w indeksie *Basic MultiColumn*, z tym że każdy identyfikator ID w tablicy typu IOT przynależy do odrębnego rekordu danych, natomiast indeks *Basic MultiColumn* zwraca pełną listę identyfikatorów w postaci pojedynczego obiektu.

Sposób definicji i tworzenia indeksów złączeniowych *MultiColumn* jest inny niż w przypadku indeksu podstawowego. Wymaganiem warunkującym budowę indeksu złączeniowego jest wskazanie sposobu łączenia indeksowanych tablic. W ramach definicji indeksu budowanego na tablicach A (odpowiednik S_PRAWA) i B (odpowiednik F_HR) należy określić:

- kolumnę (kolumny) tablicy B biorącą udział w złączeniu (b_{k1}, b_{k2},..., b_{kn}),
- kolumnę (kolumny) tablicy A (a_{k1}, a_{k2},..., a_{km}) klucza wyszukiwania indeksu,
- kolumnę tablicy A przechowującą klucze złączenia (a_{key}),
- sposób rozróżnienia wartości występujących w kolumnie a_{key} – odpowiednio, według biorących udział w złączeniu kolumn tablicy B.

Określenie sposobu identyfikacji różnych znaczeniowo wartości (elementów) klucza złączenia a_{key} jest podstawowym problemem definicji indeksu. Przedstawione w pracy rozwiązanie oparto na wprowadzeniu dodatkowej kolumny tablicy A (a_{kx}, w analizowanym przykładzie – S_PRAWA.KOD) jawnie określającej znaczenie danego klucza złączenia. Przyjęty mechanizm rozróżnienia danych nie ogranicza możliwości wprowadzenia innych sposobów rozwiązania opisywanego problemu.

Format deklaracji indeksu może przyjąć następującą, przykładową postać:

```
CREATE [BITMAP] INDEX <nazwa indeksu>
ON A(ak1, ak2,..., akm)
JOIN B(bk1, bk2,..., bkn)
ON akey
CLUSTERED BY akx(
      CASE <Wartość akx1>: bk1,
      CASE <Wartość akx2>: bk2,
      ...
      CASE <Wartość akxn>: bkn
);
```

W rozważanym przypadku tablic F_HR i S_PRAWA wyżej zaproponowane polecenie utworzenia indeksu typu *MultiColumn Bitmap Join Index I* przybiera następującą postać:

```
CREATE BITMAP INDEX I_F_HR_S_PRAWA1
ON S_PRAWA(ID_UZYTKOWNIK, KOD)
JOIN F_HR(ID_DATA, ID_KOM_ORG, ID_PRACOWNIK)
ON ID CLUSTERED BY KOD(
      CASE 'ID_DATA': ID_DATA,
      CASE 'ID_KOM_ORG': ID_KOM_ORG,
      CASE 'ID_PRACOWNIK': ID_PRACOWNIK
);
```

a polecenie utworzenia indeksu typu *MultiColumn Bitmap Join Index II* przybiera postać:

```
CREATE BITMAP INDEX I_F_HR_S_PRAWA2
ON S_PRAWA(ID_UZYTKOWNIK)
JOIN F_HR(ID_DATA, ID_KOM_ORG, ID_PRACOWNIK)
ON ID CLUSTERED BY KOD(
      CASE 'ID_DATA': ID_DATA,
      CASE 'ID_KOM_ORG': ID_KOM_ORG,
      CASE 'ID_PRACOWNIK': ID_PRACOWNIK
);
```

Algorytm budowy indeksu *MultiColumn Join Index I/II*:

Dla wszystkich wartości kluczy wyszukiwania występujących w $A(Wa_{k1}, Wa_{k2}, \ldots Wa_{km})${

W przypadku gdy a_{kx} *wchodzi w skład klucza wyszukiwania*{

/ MultiColumn Join Index I */*

Na podstawie definicji indeksu wyznacz kolumnę b_k *związaną z bieżącym kluczem wyszukiwania.*

Wyznacz odpowiednie adresy rekordów z B – powstaje lista L_{bk}.

```
SELECT rowid FROM B
WHERE bk in (
      SELECT akey FROM A
      WHERE (
            AND ak1 = Wak1 AND ak2 = Wak2
            ... AND akm = Wakm
      )
)
```

Dodaj do indeksu klucz $(Wa_{k1}, Wa_{k2}, \ldots, Wa_{km})$ *z listą adresów* L_{bk}.

}

w przeciwnym wypadku { */** a_{kx} *nie wchodzi w skład klucza wyszukiwania */*

/ MultiColumn Join Index II */*

Wyznacz odpowiednie adresy rekordów z B – powstaje lista adresów L.

```
SELECT rowid FROM B
WHERE bk1 in (
      SELECT akey FROM A
      WHERE (
            akx = <Wartość akx1>
            AND ak1 = Wak1 AND ak2 = Wak2
            ... AND akm = Wakm
      )
)
...
OR bkn in (
      SELECT akey FROM A
      WHERE (
            akx = <Wartość akxn>
            AND ak1 = Wak1 AND ak2 = Wak2
            ... AND akm = Wakm
      )
)
```

Dodaj do indeksu klucz $(Wa_{k1}, Wa_{k2}, \ldots, Wa_{km})$ *z listą adresów L.*

}

}

Podstawowym krokiem procedury fizycznej budowy indeksu *MultiColumn* jest wyznaczenie i zapamiętanie zestawu rekordów odpowiadających danemu kluczowi wyszukiwania. Proces tworzenia indeksu obejmuje wykonanie, dla każdego z kluczy wyszukiwania (w prezentowanym przykładzie, dla każdej wartości ID_USER), zapytania określającego udostępniony podzbiór danych. Zależnie od typu tworzonego indeksu wynikowy zestaw rekordów zapisywany jest w formie listy identyfikatorów ROWID lub wektora bitowego. W pracy [17] wskazano na równoważność reprezentacji indeksów w postaci list ROWID i wektorów bitowych oraz zasugerowano możliwość zmiany sposobu zapisu zależnie od stopnia gęstości indeksu (dla indeksów gęstych, wskazujących wiele rekordów – postać bitmapy, dla indeksów rzadkich – postać listy ROWID). Z tego też względu można wyobrazić sobie unifikację indeksów *MultiColumn Join* i *MultiColumn Bitmap Join* do postaci indeksu o strukturze mieszanej, zależnej od rodzaju aktualnie indeksowanych danych. Jak już wspomniano wcześniej, również i struktura odpowiedzialna za wyszukiwanie kluczy indeksu nie musi mieć ustalonej formy – może przybrać postać drzewa zrównoważonego, tablicy z funkcją mieszającą lub inną. Podobnym modyfikacjom może podlegać sposób definicji klucza wyszukiwania indeksu. Klucz ten może mieć formę uzależnioną nie tylko od nazwy użytkownika ale i innych, bardziej złożonych warunków (np. odnoszących się do czasu wykonania zapytania, rodzaju używanej aplikacji itp.).

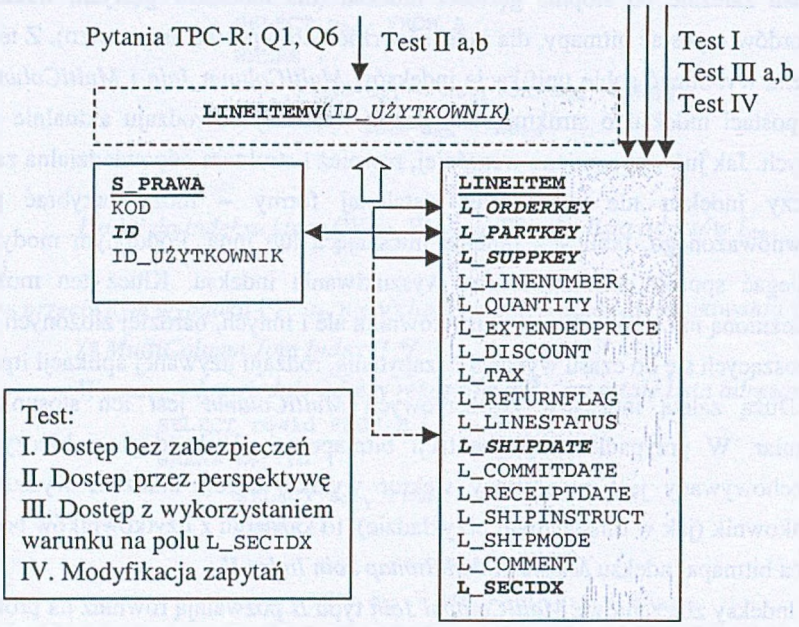
Dużą zaletą indeksów złączeniowych *MultiColumn* jest ich stosunkowo niewielki rozmiar. W przypadku implementacji bitmapowej dla każdego z kluczy wyszukiwania przechowywany jest pojedynczy wektor wyniku. Jeżeli kluczem wyszukiwania będzie użytkownik (jak w omawianym przykładzie), to każdemu z użytkowników będzie przypisana jedna bitmapa indeksu *MultiColumn Bitmap Join Index II*.

Indeksy złączeniowe *MultiColumn Join* typu *II* pozwalają również na proste rozróżnienie sytuacji, w których indeks wskazuje na pełny lub pusty zbiór rekordów. W przypadku omawianego systemu zabezpieczeń pierwsza z sytuacji ma miejsce, gdy użytkownik posiada prawa do wszystkich informacji, a druga, gdy użytkownik nie posiada żadnych praw. Reprezentację bitmapową pierwszego przypadku stanowi wektor samych jedynek, a drugiego – wektor zer. Rozpoznanie przez optymalizator sytuacji pojawienia się wektorów szczególnych powinno prowadzić do zmiany sposobu działania systemu. W przypadku wektora jedynek zapytanie powinno dotyczyć bezpośrednio niezabezpieczonej struktury podstawowej, a w przypadku wektora zer system powinien od razu zwrócić komunikat o braku danych (lub w bardziej zaawansowanej wersji – o braku dostępu).

Indeksy złączeniowe *MultiColumn* cechują się znacznymi kosztami budowy i modyfikacji ich struktury fizycznej. Koszty ewentualnej przebudowy indeksu, związane ze zmianami wartości indeksowanych danych, mogą być na tyle duże, że najszybszą metodą aktualizacji

indeksu może się okazać jego usunięcie i ponowna budowa (szczególnie dotyczy to indeksów typu II). W praktyce koszty przebudowy indeksu *MultiColumn* można ograniczyć:

- Zmiany uprawnień dotyczące pojedynczych użytkowników powodują unieważnienie tylko pojedynczego wektora/listy związanego z tym właśnie użytkownikiem.
- W przypadku partycjonowania poziomego tablicy faktów możliwa jest budowa indeksów złączeniowych *MultiColumn* jako indeksów lokalnych dla danej partycji, co przy modyfikacji danych w pojedynczej partycji prowadzi tylko do częściowego unieważnienia indeksu (w przypadku indeksu *MultiColumn* typu I).



Rys. 8. Schemat testowy
Fig. 8. Test schema

5. Badania doświadczalne

W przeprowadzonych doświadczeniach wykorzystano zmodyfikowaną tablicę LINEITEM pochodzącą ze standardowego schematu testowego TPC-R [34,35]. Badana tablica przechowywała około 60 milionów rekordów o łącznym rozmiarze 7.6GB (wskaźnik rozmiaru danych generatora *scale factor*=10). Rozszerzenie tablicy LINEITEM o dodatkową kolumnę (L_SECIDX, rys. 8), wskazującą, że wybrany użytkownik posiada dostęp do danego wiersza wraz z utworzeniem indeksu na tej kolumnie, pozwala zasymulować istnienie indeksu typu *MultiJoin* w środowiskach, które nie posiadają tego typu indeksów. Kolumna

L_SECIDX przechowuje wartości 1 lub 0, oznaczające posiadanie prawa dostępu do danego rekordu lub jego brak. Zawartość kolumny jest uzależniona od nazwy użytkownika, na rzecz którego jest wykonywana bieżąca procedura testowa i jest ona wyznaczana (przeliczana) indywidualnie przed każdą serią testów. Budowa indeksu bitmapowego na kolumnie L_SECIDX pozwala uzyskać indeks zawierający dwie mapy bitowe, oznaczające rekordy dostępne (bitmapa jedynek) i niedostępne (bitmapa zer) dla testowanego w danej chwili użytkownika. Uzyskane mapy bitowe odpowiadają sobie – mapa zer jest negacją mapy jedynek – i na potrzeby budowy indeksu wystarczyłaby tylko jedna z nich.

Zwiększenie rozmiaru tablicy LINEITEM związane z dodaniem nowej kolumny nie wpływa na jakość osiąganych rezultatów, gdyż wszystkie doświadczenia (w tym i oryginalne zapytania testu TPC-R) przeprowadzono na tablicy zmodyfikowanej. Minimalizując wpływ istnienia dodatkowej kolumny na uzyskiwane wyniki w stosunku do oryginalnego schematu TPC-R należy przypisać jej typ o jak najmniejszym rozmiarze fizycznym, np. typ bitowy.

Z listy pytań testu TPC-R wybrano dwa pytania związane z tablicą LINEITEM: pytanie Q1 – sumaryczny raport wyników oraz Q6 – raport przewidywanych zmian przychodów [34]. Pytania były wykonywane przez użytkowników U0-U6 mających następujące prawa:

- U0 – dostęp bez ograniczeń, użytkownik zadaje pytania z pominięciem struktur zabezpieczających,
- U1 – ograniczenie dostępu do 15% danych, wg warunku: rok daty dostawy (L_SHIPDATE) jest równy 1994,
- U2 – ograniczenie dostępu do 10% danych, wg warunków na kolumny L_ORDERKEY, L_PARTKEY, L_SUPPKEY,
- U3 – użytkownik posiada prawa do wszystkich danych – dostęp bez ograniczeń, ale za pośrednictwem mechanizmów bezpieczeństwa,
- U4 – ograniczenie dostępu do 1% danych, wg warunków na kolumnę L_PARTKEY,
- U5 – ograniczenie dostępu do 1% danych, wg warunków na kolumny L_ORDERKEY, L_PARTKEY, L_SUPPKEY.

Użytkownikom U1-U5 nadano odpowiednie prawa, zapisane w tablicy praw S_PRAWA w postaci par (kod, identyfikator). Sumarycznie tablica praw przechowywała ponad 17 milionów rekordów (tabela 1), a jej rozmiar wynosił 464MB.

W zależności od użytego w doświadczeniach schematu bazy danych (oraz mechanizmów zabezpieczających i optymalizujących) wyróżniono 4 grupy testów:

- test I – schemat bez zabezpieczeń:
  - pytania kierowane są bezpośrednio do tablicy LINEITEM,
  - pytania wykonuje użytkownik U0,

Tabela 1

Liczby rekordów tablicy praw S_PRAWA przypisanych użytkownikom testowym

Użytkownik	Prawo	Liczba rekordów
U1	L_SHIPDATE	365
	Liczba rekordów U1	***365***
U2	L_ORDERKEY	10
	L_PARTKEY	12 999
	L_SUPPKEY	8 349
	Liczba rekordów U2	***21 358***
U3	L_ORDERKEY	15 000 000
	L_PARTKEY	2 000 000
	L_SUPPKEY	100 000
	Liczba rekordów U3	***17 100 000***
U4	L_PARTKEY	12 999
	Liczba rekordów U4	***12 999***
U5	L_ORDERKEY	10
	L_PARTKEY	6 299
	L_SUPPKEY	449
	Liczba rekordów U5	***6 758***
U1-U5	Sumaryczna liczba rekordów	**17 141 480**

- test II – schemat zabezpieczony perspektywą:
  - pytania wykonują użytkownicy U1-U5,
  - test IIa dotyczy perspektywy LINEITEMV zbudowanej na podstawie tablic LINEITEM i S_PRAWA połączonych według kolumn L_SHIPDATE, L_ORDERKEY, L_PARTKEY, L_SUPPKEY (4 kolumny łączące),
  - test IIb dotyczy perspektywy LINEITEMV zbudowanej na podstawie tablic LINEITEM i S_PRAWA połączonych według kolumn L_ORDERKEY, L_PARTKEY, L_SUPPKEY (3 kolumny łączące),
- test III – schemat zabezpieczony dodatkowym warunkiem na kolumnie L_SECIDX:
  - pytania wykonują użytkownicy U1-U5,
  - symulacja indeksu *MultiJoin Bitmap Index II* – na kolumnie L_SECIDX założono indeks bitmapowy,
  - test IIIa – pytania wykonywane są według domyślnego planu wygenerowanego przez optymalizator kosztowy,
  - test IIIb – w celu wymuszenia korzystania z indeksu i uniknięcia bezpośredniego testowania zawartości kolumny L_SECIDX pytania wykonywane są według alternatywnej ścieżki dostępu do danych; wygenerowany w teście plan

wykonania zapytania zakłada wstępne łączenie (ang. *merge*) indeksów bitmapowych zbudowanych na kolumnach, na które narzucono warunki frazy `WHERE`,

- test IV – schemat z zabezpieczeniem dostępu poprzez modyfikację zapytań:
  - pytania kierowane są bezpośrednio do tablicy `LINEITEM`,
  - dodatkowy warunek zabezpieczający jest zapisany we frazie `WHERE`,
  - pytania wykonuje użytkownik U1 – ze względu na prosty warunek dostępu: rok daty dostawy (`L_SHIPDATE`) jest równy `1994`.

W przypadku występującej w teście IIa perspektywy `LINEITEMV` wymagana jest konwersja typu przy porównywaniu wartości `L_SHIPDATE` (typ data) i `S_PRAWA.ID` (typ całkowity). Perspektywa z testu IIb nie wymaga tego typu działań, gdyż kolumny `L_ORDERKEY`, `L_PARTKEY`, `L_SUPPKEY` – podobnie jak `S_PRAWA.ID` – są typu całkowitego.

Przy wykonywaniu doświadczeń korzystano z optymalizatora kosztowego i zadbano o aktualność statystyk bazodanowych związanych z tablicami `LINEITEM` i `S_PRAWA`. Wyniki przeprowadzonych testów (średnie arytmetyczne z min. 3 prób) przedstawia tabela 2.

Tabela 2

Czasy wykonania zapytań Q1, Q6 dla różnych schematów zabezpieczeń bazy danych (wytłuszczono najlepsze wyniki dla danego użytkownika)

Schemat \ Użytkownik	U0	U1	U2	U3	U4	U5
I. Schemat TPC-H						
TPC-H Q1 [hh:mm:ss]	**00:04:48**	–	–	–	–	–
TPC-H Q6 [hh:mm:ss]	**00:02:07**	–	–	–	–	–
IIa. Perspektywa zabezp. 4 kol						
TPC-H Q1 [hh:mm:ss]	–	08:17:28	00:28:40	00:22:52	00:25:39	00:25:55
TPC-H Q6 [hh:mm:ss]	–	00:04:43	00:02:08	00:06:56	00:02:08	00:02:08
IIb. Perspektywa zabezp. 3 kol						
TPC-H Q1 [hh:mm:ss]	–	00:16:15*	00:20:46	00:22:55	00:18:00	00:18:04
TPC-H Q6 [hh:mm:ss]	–	**00:02:06***	00:02:08	00:06:56	00:02:08	00:02:08
IIIa. Symulacja MJ Bitmap Ind.II						
TPC-H Q1 [hh:mm:ss]	–	**00:02:08**	**00:02:08**	**00:04:56**	**00:02:07**	**00:02:08**
TPC-H Q6 [hh:mm:ss]	–	00:02:07	**00:02:07**	**00:02:07**	00:02:08	00:02:08
IIIb. Symulacja MJ Bitmap Ind.II						
TPC-H Q1 [hh:mm:ss]	–	00:04:50	00:03:01	00:08:38	00:02:40	00:02:43
TPC-H Q6 [hh:mm:ss]	–	00:03:31	00:02:20	00:03:40	**00:00:52**	**00:00:55**
IV. Modyfikacja zapytań						
TPC-H Q1 [hh:mm:ss]	–	**00:02:08**	–	–	–	–
TPC-H Q6 [hh:mm:ss]	–	00:02:07	–	–	–	–

*Ze względu na brak odpowiednich praw dostępu otrzymano wynik pusty.

Zróżnicowanie jakościowe i ilościowe zestawu posiadanych praw wpływa na końcowy wynik i szybkość jego uzyskania. Ze względu na różnice w rodzaju posiadanych praw wyniki zgodne z rezultatami uzyskiwanymi przez użytkownika U0 otrzymano jedynie w przypadku użytkownika U3, który posiadał pełne prawa oraz w przypadku pytania Q6 wykonanego przez użytkownika U1 (występujący w pytaniu warunek nie kolidował z zakresem posiadanych przez użytkownika praw). W szczególności użytkownik U1 uzyskał wynik pusty w teście IIb (gdyż posiadał wyłącznie prawa do kolumny L_SHIPDATE, która nie była uwzględniana w definicji perspektywy zabezpieczającej).

Mimo zredukowania obszaru badań doświadczalnych do zagadnienia analizy czasów i planów wykonania tylko dwóch zapytań, uzyskane wyniki pozwalają sformułować pewne praktyczne wnioski. Analiza wyników z tabeli 2 pozwala stwierdzić, że ograniczenie zestawu dostępnych danych i oznaczenie go strukturą indeksującą (test III) pozwala na uzyskanie odpowiedzi w znacząco krótszym czasie aniżeli użycie samej perspektywy zabezpieczającej (test II). W stosunku do schematu nie stosującego mechanizmów zabezpieczeń (test I, użytkownik U0) rozwiązanie z użyciem samej tylko perspektywy zabezpieczającej może w znaczący sposób obniżyć wydajność przetwarzania zapytań. Efekt ten jest szczególnie widoczny w przypadku zapytania Q1. W przypadku zapytania Q6 większość czasów testu IIa,b odpowiada czasom uzyskanym w teście I, choć w szczególnym przypadku użytkownika posiadającego wszystkie prawa (U3) są one znacząco większe.

Ze względu na analizę możliwości zastosowania indeksów *MultiColumn* najważniejszy jest test IIIb. W teście tym wymuszono wykonanie zapytania z wykorzystaniem indeksu zbudowanego na kolumnie L_SECIDX. W systemach zarządzania bazami danych struktury indeksujące stanowią jeden z wielu mechanizmów wspomagających proces wykonywania zapytań. Wykorzystanie konkretnego indeksu uzależnione jest od decyzji modułu optymalizatora zapytań. Test IIIa, korzystający z domyślnego planu wykonania zapytania, nie gwarantuje wykorzystania indeksu zbudowanego na kolumnie L_SECIDX. Rzeczywiście, w przypadku prowadzonych doświadczeń, domyślny plan wykonania zapytania Q1 w teście IIIa zakładał pełne przeglądanie tablicy (ang. *full scan*) z testem zawartości kolumny L_SECIDX. Kolumna ta jest tylko kolumną pomocniczą, wprowadzoną na potrzeby doświadczeń i w rzeczywistym przypadku zastosowania indeksu *MultiColumn* kolumna ta nie występuje. Dlatego też w ocenie działania indeksu należy uwzględnić przede wszystkim wyniki testu IIIb. Dodatkowo uwzględniając, że plan wykonania zapytania Q6 w teście IIIa był odmienny niż w przypadku testu IIIb, ale również wykorzystywał indeks zbudowany na kolumnie L_SECIDX, można uznać, że uzyskane w tym teście czasy są możliwe do osiągnięcia poprzez zastosowanie indeksu *MultiColumn*.

Dla małych ilości dostępnych danych (użytkownicy U4, U5) zastosowanie indeksu może potencjalnie przynieść największe zyski czasowe. W takich przypadkach (test IIIb, Q6) indeksy z grupy *MultiColumn Bitmap Join* dobrze współpracują z innymi indeksami bitmapowymi, efektywnie uczestnicząc w wyznaczaniu wyników złożonych warunków logicznych.

Zastosowanie techniki modyfikacji zapytań (test IV) przez użytkownika U1 pozwoliło na uzyskanie dobrego rezultatu w przypadku zapytania Q1 (w porównaniu z IIIb, Q1) i podobnego wyniku w przypadku pytania Q6 (w porównaniu z IIIa, Q6; w przypadku IIIb, Q6 osiągnięty tam czas jest lepszy). Dobre wyniki uzyskiwane przy zastosowaniu tego mechanizmu zachęcają do jego szerszego stosowania [36]. Należy jednak pamiętać, że modyfikację zapytań można stosować w przypadku prostych, łatwych do zdefiniowania warunków bezpieczeństwa (przypadek użytkownika U1). W sytuacji występowania wielu indywidualnie określanych warunków dostępu (praw) – tak jak w przypadku użytkowników U2-U5 – modyfikacja jest trudna do przeprowadzenia lub też wymaga zastosowania tablicy praw (co wydajnościowo sprowadza to rozwiązanie do poziomu uzyskiwanego przy zastosowaniu perspektywy zabezpieczającej).

Budowa systemu zabezpieczeń w oparciu o perspektywy zabezpieczające wymaga szczegółowej analizy w obszarze zakresu wprowadzanych zabezpieczeń oraz wpływu rozwiązania na wydajność systemu. Uzyskane w pracy wyniki doświadczalne (test II) potwierdzają znaczny, negatywny wpływ użycia perspektyw opartych na tablicy praw na szybkość uzyskiwania odpowiedzi. Wpływ ten jest szczególnie widoczny w przypadku złożonej perspektywy z testu IIa, której definicja zawiera operacje konwersji typów kolumn tablic. W przypadku stosowania perspektyw zabezpieczających należy dążyć do maksymalnego uproszczenia ich definicji, np. poprzez rezygnację z zabezpieczenia według pewnych kryteriów, tak jak w definicji perspektywy z testu IIb. Jeżeli jest to niemożliwe, to w przypadku sytuacji pojawiającej się w teście IIa należy dążyć do ujednolicenia typów wszystkich identyfikatorów, według których następuje określenie warunków zabezpieczających lub też należy zbudować perspektywę opartą na wielu tablicach praw – związanych z identyfikatorami różnych typów. Zastosowanie indeksu *MultiColumn* (test IIIb) pozwala w pewien sposób uniezależnić się od stopnia złożoności definicji perspektywy zabezpieczającej. Złożoność struktury perspektywy wpływa na czas budowy indeksu, ale nie ogranicza możliwości uzyskiwania krótszych czasów odpowiedzi na zadawane zapytania.

6. Podsumowanie

Przedstawiona grupa indeksów *MultiColumn* w skuteczny sposób wspomaga wykonywanie operacji złączenia pomiędzy tablicami faktów i tablicą praw. Wśród opisanych wariantów indeksów *MultiColumn* należy wyróżnić:

- indeks podstawowy *Basic MultiColumn* – ze względu na potencjalnie najniższe koszty budowy i aktualizacji indeksu,
- indeks *MultiColumn Join Index II* – ze względu na potencjalnie najwyższą wydajność przy przetwarzaniu zapytań nie zawierających dodatkowych warunków narzucanych przez użytkownika, przy małej ilości udostępnionych mu danych,
- indeks *MultiColumn Bitmap Join Index II* – ze względu na potencjalnie najwyższą wydajność przy przetwarzaniu zapytań z dodatkową frazą warunków OLAP, w obecności innych indeksów bitmapowych.

Prowadzanie analiz wielowymiarowych i generowanie raportów jest nieodłącznie związane ze specyfikacją parametrów określających podzbiór przetwarzanych danych. Ze względu na powszechność występowania kryteriów selekcji w wykonywanych zapytaniach, zastosowanie indeksu typu *MultiColumn Bitmap Join Index II* wydaje się być najlepszym rozwiązaniem dla systemu zabezpieczonego z wykorzystaniem perspektyw. Wykorzystanie indeksu w znaczący sposób potrafi podnieść wydajność pracy zabezpieczonej hurtowni danych.

LITERATURA

1. Kimball R., Reeves L., Margy R., Thornthwaite W.: The Data Warehouse Lifecycle Toolkit. John Wiley & Sons, 1998.
2. Gorawski M., Frączek J.: Kontrola dostępu do informacji w hurtowniach danych. ZN Pol. Śl. Studia Informatica Vol. 24, No 2B(54), Gliwice 2003.
3. Kimball R.: The Data Warehouse Toolkit. John Wiley & Sons, Inc, 1996.
4. Castano S., Fugini M., Matrella G., Samarati P.: Database Security. ACM Press, 1995.
5. Rosenthal A, Sciore E.: View Security as the Basis for Data Warehouse Security. Proc. of the Int. Workshop on Design and Management of Data Warehouses. Stockholm, 2000.
6. Oracle9*i* Security Overview Release 2 (9.2). Oracle Corporation 2002.
7. Oracle Label Security Administrator's Guide, Release 2 (9.2), Oracle Corporation, 2002.
8. Sandhu R., Samarati P.: Access Control: Principles and Practice. IEEE Communications, vol.32, no.9, 1994.

9. Bell E., La Padula L.: Secure Computer Systems: Mathematical Foundations. MITRE Technical Report 2547, vol. I, The MITRE Corporation, 1973.
10. Bell E., La Padula L.: Secure Computer Systems: A Mathematical Model. Secure Computer Systems: Mathematical Foundations. MITRE Technical Report 2547, vol. II, The MITRE Corporation, 1973.
11. Gorawski M., Frączek J.: Implementacja procedur bezpieczeństwa w hurtowniach danych. ZN Pol. Śl. Studia Informatica – w recenzji.
12. Elmasri R., Navathe Sh.: Foundamentals of Database Systems. Third Edition. Addison-Wesley, 2000.
13. Date C. J.: An Introduction to Database Systems. Seventh Edition. Addison-Wesley Longman, 2000.
14. O'Neil P., O'Neil E.: Database principles, programming, and performance. Second Edition. Morgan Kaufmann Publishers, 2001.
15. Garcia-Molina H., Ullman J.D., Widom J.: Database Systems: The Complete Book. Aprentice Hall, 2002.
16. Chan Ch-Y., Ioannidis Y.: Bitmap Index Design and Evaluation. ACM SIGMOD. 1998.
17. O'Neil P., Quass D.: Improved query performance with variant indexes. Proc. ACM SIGMOD Intl. Conf. on Management of Data. 1997, pp. 38-49.
18. Datta A., Viguier I.: The DataIndex: A Structure for Smaller, Faster Data Warehouses. The DATA BASE for Advances in Information Systems. Vol. 29, No. 4, 1998.
19. O'Neil P., Graefe G.: Multi-Table Joins Through Bitmapped Join Indices. ACM SIGMOD Record. 1995, pp. 8-11.
20. Bellatreche L, Karlapalem K, Mohania M.: Some Issues in Design of Data Warehousing Systems. http://citeseer.nj.nec.com/453918.html.
21. Ponniah P.: Data Warehousing Fundamentals. A Comprehensive Guide for IT Professionals. John Wiley and Sons, 2001.
22. Frączek J., Gorawski M., Kozielski S.: Modelowanie struktur wielowymiarowych w hurtowniach danych. Archiwum Informatyki Teoretycznej i Stosowanej. Tom 12 (2000). z.3. pp 173-201.
23. Blakeley J., Martin N.: Join Index, Materialized View, and Hybrid-Hash Join: A Performance Analysis. Proc. of the Sixth International Conference on Data Engineering. IEEE Computer Society, 1990.
24. Oracle9i Data Warehousing Guide, Release 2 (9.2). Oracle Corporation, 2002.
25. Thombre N., Ozbutun C., Jiang L.: Key Data Warehousing Features in Oracle9i: A Comparative Performance Analysis. An Oracle White Paper. Oracle Corporation, 2001.

26. Srinivasan J., et al.: Oracle8i Index-Organized Table and its Application to New Domains. Proc. of the 26th Int. Conf. on Very Large Databases, 2000.
27. Oracle9i Database Performance Tuning Guide and Reference, Release 2 (9.2). Oracle Corporation, 2002.
28. Atzenberger B., Bender M. i inni.: Data Warehouse Performance Enhancements with Oracle9i. An Oracle White Paper. Oracle Corporation 2001.
29. Valduriez P.: Join indices. ACM Transactions on Database Systems. Vol. 12(2), pp. 218-246. 1987.
30. Miszczyk J., Harris N., Kocinski P., Stice J., Unger K.: DB2/400: Mastering Data Warehousing Functions. IBM Corporation. 1998. http://www.redbooks.ibm.com.
31. Viguier I., Datta A.: Sizing Access Structures for Data Warehouses. Technical report, Dept. of MIS, University of Arizona, Tucson, AZ. 1997. http://citeseer.nj.nec.com/viguier97sizing.html.
32. Amer-Yahua S., Johnson Th.: Optimizing Queries On Compressed Bitmaps. Proc. of the 26th VLDB Conf. 2000.
33. Date C. J., Darwen H.: SQL. Omówienie standardu języka. Wydawnictwa Naukowo-Techniczne, Warszawa 2000.
34. TPC Benchmark R (Decision Support). Standard Specification. Revision 2.1.0. Transaction Processing Performance Council (TPC), 2003. http://www.tpc.org.
35. Poess M., Floyd Ch.: New TPC Benchmarks for Decision Support and Web Commerce. ACM Press. ACM SIGMOD Record Vol. 29, Issue 4, New York 2000.
36. Oracle9i Security Overview Release 2 (9.2). Oracle Corporation, 2002.

Recenzent: Prof. dr hab. inż. Tadeusz Morzy

Wpłynęło do Redakcji 3 października 2003 r.

Abstract

This paper describes a new mechanism that optimizes the efficiency of a row-level access control system, based on database views. A description of complex security polices in a database system, often requires implementation of an additional security table. The definition of a security view build over the security table and a fact table specifies a multi-join operation between these two tables (Fig.1). To improve the effectiveness of this operation, additional indexing structures may be created. The proposed set of five *MultiColumn* indices is intended

to increase the performance of queries sent to a secured data warehouse. The *Basic MultiColumn* index (Fig.2) has the lowest complexity, minimal creation and update costs. The following *MultiColumn Join* indices (Fig.3, 4) use ROWIDs of records from the fact table, and are more effective in performing the join operation. In the case of a simple *select* query that has no other OLAP conditions specified, the *MultiColumn Join Index II* is the most effective index. *MultiColumn Bitmap Join Indices I/II* (Fig.5, 6) use bitmap representations of users' rights that can be efficiently combined with other bitmaps representing OLAP conditions (Fig.7). The star transformation-like rewrite of multidimensional queries and the use of *Bitmap Join Index II*, ensures faster query execution times. Fig.8 shows the database test schema used for experiments. The results of simulation of *Bitmap Join Index II* are shown in table 2.

Adres

Jacek FRĄCZEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-101 Gliwice, Polska, jacekf@zeus.polsl.gliwice.pl .