XVI Konferencja PLOUG
Kościelisko
Październik 2010

Materializowane perspektywy eksploracyjne w praktyce: Wnioski z implementacji w Oracle 11g

Damian Mierzwiński, Marek Wojciechowski
Politechnika Poznańska

Marek.Wojciechowski@cs.put.poznan.pl

Abstrakt. Materializowane perspektywy są powszechnie wykorzystywane we współczesnych systemach zarządzania bazami danych, w szczególności do skrócenia czasów wykonywania zapytań analitycznych w hurtowniach danych. W ostatnich latach w literaturze naukowej pojawiły się propozycje rozszerzenia zastosowań perspektyw materializowanych o wybrane techniki eksploracji danych. Artykuł poświęcony jest prototypowej implementacji w systemie Oracle 11g jednego z zaproponowanych w literaturze podejść do wykorzystania zmaterializowanych wyników poprzednich zapytań eksploracyjnych w kontekście odkrywania zbiorów częstych.

Informacja o autorze. Dr inż. Marek Wojciechowski jest adiunktem w Instytucie Informatyki Politechniki Poznańskiej. Specjalizuje się w eksploracji danych, technologiach internetowych i obsłudze danych multimedialnych w bazach danych. Jest autorem ok. 70 publikacji o charakterze naukowym i technicznym. Od kilku lat prowadzi szkolenia z technologii internetowych i produktów Oracle w Polsce i za granicą w ramach Oracle University. W trakcie konferencji, szkół i seminariów PLOUG wielokrotnie prowadził warsztaty, tutoriale i wykłady na temat tworzenia aplikacji internetowych i obsługi zaawansowanych typów danych przez system zarządzania bazą danych Oracle.

1. Wprowadzenie

Eksploracja danych stanowi jeden z etapów pojmowanego szerzej procesu odkrywania wiedzy (ang. knowledge discovery in databases, KDD). Definiowana jest jako proces automatycznego odkrywania nietrywialnych, dotychczas nieznanych, potencjalnie użytecznych reguł, zależności, wzorców, schematów, podobieństw lub trendów w dużych repozytoriach danych [FPS96]. Pomimo iż proces ten jest w znacznej mierze zautomatyzowany, centralne miejsce zajmuje w nim człowiek, który może ukierunkowywać proces eksploracji poprzez specyfikowanie warunków i ograniczeń w ramach tzw. zapytań eksploracyjnych [IM96].

Dobór ograniczeń prowadzących do wyboru możliwych do ogarnięcia interesujących wzorców nie jest zadaniem łatwym, dlatego też interaktywna eksploracja danych jest procesem iteracyjnym, w którym użytkownik modyfikuje kryteria zapytania aż do uzyskania satysfakcjonujących rezultatów. Z punktu widzenia systemu eksploracji danych taki scenariusz oznacza sekwencję podobnych zapytań do wykonania. Materializacja wyników zapytań może pozwolić na zwiększenie efektywności przetwarzania kolejnych zapytań, poprzez ich modyfikację (tzw. przepisanie – ang. query rewrite), tak aby wykorzystały zmaterializowane wyniki zapytań poprzednich [ZMW04, WZ05]. Oczywiście z materializowanych wyników zapytań jednego użytkownika mogą w przyszłości skorzystać również inni użytkownicy, pod warunkiem opracowania architektury umożliwiającej współdzielenie materializowanych wyników zapytań eksploracyjnych w bazie danych.

Niniejszy artykuł stanowi sprawozdanie z budowy biblioteki FI_WRAPPER stanowiącej nakładkę na dostarczaną przez firmę Oracle funkcję odkrywania zbiorów częstych, udostępnianą w ramach systemu zarządzania bazą danych Oracle 10g oraz 11g w postaci pakietu DBMS_FREQUENT_ITEMSET. Biblioteka ta miała być praktyczną implementacją metod przepisywania zapytań eksploracyjnych, zaproponowanych w [ZMW04]. Bazując na przedstawionych w tym opracowaniu rozważaniach teoretycznych oraz wynikach testów wydajnościowych, a także dokumentacji udostępnianej przez firmę Oracle, określony miał zostać algorytm przepisywania zapytań odkrywania zbiorów częstych. Powstała na jego bazie implementacja miała dalej posłużyć zweryfikowaniu adekwatności wykorzystania wspomnianych metod, w stosunku do komercyjnego rozwiązania firmy Oracle. Podstawą dla takiej weryfikacji miały być testy wydajnościowe, porównujące powstałą implementację z bazową funkcją w kontekście całego scenariusza zapytań eksploracyjnych. W ten sposób zweryfikowane miały zostać przyjęte założenia oraz jakość wykonanej implementacji.

2. Odkrywanie zbiorów częstych

2.1. Opis problemu

Odkrywanie zbiorów częstych (ang. frequent itemset mining) jest jedną z podstawowych technik eksploracji danych (ang. data mining), stosowaną zarówno samodzielnie jak i jako pierwszy etap w odkrywaniu reguł asocjacyjnych (ang. association rules) [AIS93]. Problem ten został pierwotnie sformułowany w kontekście analizy koszyka zakupów, ale szybko znalazł liczne inne zastosowania, do których należą m.in. analiza danych medycznych, społecznych i telekomunikacyjnych oraz analiza zachowań użytkowników WWW.

Odkrywanie zbiorów częstych polega na odkrywaniu najczęściej występujących podzbiorów w bazie danych zawierającej zbiory nazywane transakcjami[1]. Miarą częstości występowania zbio-

[1] Nazewnictwo nawiązuje do transakcji dokonywanych przez klientów w sklepach, a nie do pojęcia transakcji w systemach baz danych.

ru w bazie danych jest współczynnik wsparcia (ang. support) definiowany jako liczba transakcji zawierających dany zbiór. W rzeczywistych systemach, dla wygody użytkowników, wsparcie wyrażane jest procentowo względem liczby wszystkich transakcji. Podstawowe sformułowanie problemu zakłada, że użytkownik zlecając zadanie odkrywania zbiorów częstych oprócz wskazania danych źródłowych specyfikuje jedynie próg minimalnego wsparcia, określający które zbiory z punktu widzenia użytkownika są częste. W praktyce, w celu zawężenia zbioru zbiorów częstych stanowiących wynik eksploracji, interfejsy do specyfikowania zadań odkrywania zbiorów częstych umożliwiają specyfikację dodatkowych ograniczeń, które muszą być spełnione przez odkrywane zbiory, dotyczących zawierania bądź niezawierania konkretnych elementów lub rozmiaru (rozumianego jako liczba elementów w zbiorze). Ograniczenia te mogą być zweryfikowane po odkryciu zbiorów spełniających kryterium częstości lub też ich weryfikacja może być zintegrowana z algorytmem odkrywania zbiorów częstych. Zależy to od natury ograniczenia, wybranego algorytmu bazowego i jego implementacji [PH00].

W celu zilustrowania problemu rozważmy przykładową bazę danych $D = \{\langle 100,\{a,b,c\}\rangle, \langle 200,\{a,c,d\}\rangle, \langle 300,\{a,b,c,d\}\rangle, \langle 400,\{a,c,d\}\rangle, \langle 500,\{a,b,d\}\rangle\}$, w której każda transakcja jest zbiorem zawierającym elementy ze zbioru $I = \{a,b,c,d\}$, opatrzonym liczbowym identyfikatorem. Tabela 1 prezentuje wszystkie zbiory częste w bazie danych D dla podanego przez użytkownika progu minimalnego wsparcia równego 50%. W kontekście bazy danych D taki próg wsparcia oznacza, że zbiór aby był uznany za częsty, musi być zawarty w co najmniej 3 transakcjach.

Tabela 1. Zbiory częste w przykładowej bazie danych dla progu minimalnego wsparcia równego 50%

Zbiór	Liczba transakcji wspierających	Wsparcie
{a}	5	100%
{b}	3	60%
{c}	4	80%
{d}	4	80%
{a,b}	3	60%
{a,c}	4	80%
{a,d}	4	80%
{c,d}	3	60%
{a,c,d}	3	60%

Sformułowanie problemu odkrywania zbiorów częstych jest bardzo proste, niemniej dla dużych baz danych i dziedzin możliwych elementów o dużej liczności, stanowi on wyzwanie od strony obliczeniowej. W szczególności, w praktyce nie jest możliwe zastosowanie trywialnego podejścia polegającego na zliczeniu wystąpień wszystkich możliwych podzbiorów w czasie jednego odczytu bazy danych, ze względu to, że ich liczba zależy wykładniczo od liczności dziedziny elementów. Z tego względu zaproponowano szereg mniej lub bardziej wyrafinowanych algorytmów odkrywania zbiorów częstych, z których zdecydowanie najpowszechniej implementowanym jest Apriori, wykorzystywany m.in. w Oracle 10g i 11g.

2.2. Algorytm Apriori

Zaproponowany w [AS94] algorytm Apriori opiera się na spostrzeżeniu, iż każdy podzbiór zbioru częstego jest także zbiorem częstym. Własność ta pozwala uniknąć zliczania wystąpień zbiorów, których przynajmniej jeden podzbiór został wcześniej zidentyfikowany jako nieczęsty. Apriori działa iteracyjnie, w kolejnych iteracjach odkrywając zbiory częste o coraz większym rozmiarze. Podczas pierwszej iteracji zliczana jest liczba wystąpień każdego pojedynczego elementu i na jej podstawie wyznaczane są 1-elementowe zbiory częste. Kolejne iteracje składają się z dwóch faz: fazy generowania zbiorów kandydackich na podstawie zbiorów częstych odkrytych

w poprzedniej iteracji oraz fazy ich weryfikacji, podczas której zliczane są wystąpienia i odrzucane są te zbiory, które nie posiadają wymaganego minimalnego wsparcia. Proces ten powtarzany jest do momentu, aż w którejś iteracji nie zostaną odkryte żadne nowe zbiory częste.

Procedura generacji kandydatów, stanowiąca pierwszą fazę każdej iteracji algorytmu Apriori, sama składa się z dwóch etapów. W pierwszym etapie znajdowane są wszystkie pary zbiorów częstych z poprzedniej iteracji, które po posortowaniu elementów w porządku leksykograficznym różnią się jedynie na ostatniej pozycji. Z każdej takiej pary powstaje jeden zbiór kandydacki zawierający wszystkie elementy obu dopasowanych zbiorów. W drugim etapie procedury generacji kandydatów, ze zbioru kandydatów odrzucane są zbiory, których nie wszystkie podzbiory o rozmiarze o 1 mniejszym zostały znalezione jako zbiory częste w poprzedniej iteracji.

Zliczanie wystąpień kandydatów, realizowane w każdej iteracji po ich wygenerowaniu, każdorazowo wymaga odczytu danych źródłowych poddawanych eksploracji. Podczas takiego odczytu dla każdej transakcji odczytanej z bazy danych sprawdzane jest, które zbiory kandydackie się w niej zawierają. W celu usprawnienia procesu zliczania kandydatów, autorzy algorytmu zaproponowali specjalną strukturę danych do przechowywania kandydatów w pamięci, nazwaną drzewem haszowym. Struktura ta pozwala na wstępne „dopasowanie" transakcji do zbioru kandydatów, eliminując niektóre zbiory kandydackie bez przeprowadzania kompletnego testu zawierania.

3. Pakiet DBMS_FREQUENT_ITEMSET

3.1. Zawartość pakietu

Począwszy od wersji 10g w systemie zarządzania bazą danych Oracle dostępny jest pakiet DBMS_FREQUENT_ITEMSET zawierający funkcje do odkrywania zbiorów częstych w danych przechowywanych w tabelach bazy danych [Ora08]. Dzięki obecności tego pakietu odkrywanie zbiorów częstych, w przeciwieństwie do innych technik eksploracji danych, nie wymaga dodatkowo płatnej opcji Oracle Data Mining.

Pakiet DBMS_FREQUENT_ITEMSET obejmuje dwie funkcje: FI_HORIZONTAL oraz FI_TRANSACTIONAL. Różnią się one jedynie formatem danych wejściowych, które wskazywane są przez pierwszy parametr funkcji. Pozostałe parametry ich wywołania są identyczne. FI_HORIZONTAL działa na danych podanych w tzw. postaci „horyzontalnej", gdzie każdy wiersz stanowi jedną transakcję, z jej elementami umieszczonymi w poszczególnych kolumnach. Liczba kolumn w tym formacie może być dowolna, niemniej stanowi ona ograniczenie rozmiaru transakcji. FI_TRANSACTIONAL przyjmuje jako parametr dane w tzw. postaci „transakcyjnej", którą wg terminologii bazodanowej należy określić mianem znormalizowanej. Na każdy wiersz składają się dwa atrybuty, z których pierwszy to identyfikator transakcji, a drugi zawiera element. Pojedyncza transakcja w tym formacie może więc być reprezentowana za pomocą jednego lub więcej wierszy, przy czym format nie ogranicza rozmiaru transakcji.

Za podstawowy format danych wejściowych należy uznać format transakcyjny jako bardziej poprawny koncepcyjnie z dwóch następujących powodów. Po pierwsze, nie ogranicza on rozmiaru źródłowych transakcji. Po drugie, nie występują w nim wartości puste, których duża liczba jest nieunikniona w formacie horyzontalnym ze względu na różne rozmiary transakcji stanowiących dane źródłowe. Dlatego też w dalszych rozważaniach uwzględniona będzie tylko funkcja FI_TRANSACTIONAL, choć odnoszą się one również do funkcji FI_HORIZONTAL. Należy w tym miejscu jeszcze dodać, że dokumentacja i eksperymenty z dwoma metodami pakietu DBMS_FREQUENT_ITEMSET nie pozostawiają wątpliwości co do roli funkcji FI_HORIZONTAL i związanego z nią formatu danych. Pierwszym krokiem działania funkcji FI_TRANSACTIONAL jest transformacja danych źródłowych do formatu horyzontalnego, który stanowi „wewnętrzny" format danych wejściowych dla firmowej implementacji algorytmu od-

krywania zbiorów częstych. Objawia się to dłuższymi czasami działania FI_TRANSACTIONAL w porównaniu z FI_HORIZONTAL. Wg dokumentacji, funkcja FI_HORIZONTAL została udostępniona, aby umożliwić uniknięcie czasochłonnej transformacji formatu danych źródłowych, jeśli użytkownik przygotuje je już w formacie horyzontalnym.

3.2. Wykorzystywany algorytm odkrywania zbiorów częstych

Jeśli chodzi o sam algorytm odkrywania zbiorów częstych w systemie Oracle, to jest nim implementacja algorytmu Apriori wykorzystująca rozwiązania mające na celu zwiększenie wydajności, opatentowane przez firmę Oracle. Dokumentacja nie dostarcza informacji na temat szczegółów implementacyjnych. Pewne informacje w tym zakresie można jednak znaleźć w dostępnym w Internecie wniosku patentowym [LHM+05]. Kluczową innowacją zaproponowaną przez Oracle jest dynamiczny sposób wyboru techniki zliczania zbiorów. Dla każdej iteracji algorytmu estymowane są koszty zastosowania jednej z dwóch struktur danych:

- drzewo prefiksowe – struktura podobna do drzewa haszowego z [AS94], przyspieszająca weryfikację zawierania się kandydatów w poszczególnych transakcjach, niestety cechująca się dużą zajętością pamięciową;
- indeksy bitmapowe – bitmapy opisujące poszczególne zbiory, zawierające informacje o transakcjach zawierających dany zbiór.

Na podstawie przeprowadzonej estymacji wybierana jest ta struktura, dla której koszt wykonania danej iteracji jest niższy. Należy zwrócić uwagę na fakt, że w przypadku wykorzystania indeksów bitmapowych znacząco ograniczone są odczyty danych z oryginalnego źródła. Jest to możliwe dzięki temu, że bitmapy z poprzednich iteracji są utrzymywane w pamięci operacyjnej i wykorzystywane do tworzenia bitmap potrzebnych w kolejnej iteracji.

3.3. Sposób użycia

Funkcje pakietu DBMS_FREQUENT_ITEMSET przyjmują następujący zestaw parametrów: kursor oparty o zapytanie SQL zwracające dane źródłowe do eksploracji (w różnym formacie dla FI_HORIZONTAL i FI_TRANSACTIONAL), próg minimalnego wsparcia, minimalny rozmiar zbioru częstego, maksymalny rozmiar zbioru częstego, kursor zwracający listę elementów, z których przynajmniej jeden musi być zawarty w zbiorze częstym oraz kursor zwracający listę elementów, z których żaden nie może być zawarty w zbiorze częstym. Dwa pierwsze parametry są wymagane przez podstawowe sformułowanie problemu odkrywania zbiorów częstych. Pozostałe parametry umożliwiają użytkownikowi dodatkowe zawężenie zbioru odkrytych zbiorów częstych zgodnie z jej/jego preferencjami.

Wynikiem działania funkcji pakietu DBMS_FREQUENT_ITEMSET jest tabela wierszy (TABLE OF ROW), z których każdy zawiera zbiór częsty, jego wsparcie, jego rozmiar (liczbę elementów) i całkowitą liczbę transakcji w źródłowym zbiorze danych (taką samą dla wszystkich odkrytych zbiorów częstych). W celu wywołania funkcji z pakietu DBMS_FREQUENT_ITEMSET należy najpierw utworzyć typ tabeli zagnieżdżonej (ang. nested table) elementów i rzutować zwrócone przez funkcję zbiory do tego typu.

Aby zilustrować sposób korzystania z pakietu DBMS_FREQUENT_ITEMSET, a konkretnie jego funkcji FI_TRANSACTIONAL, rozważmy następujący przykład. Niech dana będzie przedstawiona poniżej tabela KOSZYKI z danymi źródłowymi w formacie transakcyjnym.

Tabela 2. Przykładowa tabela z danymi źródłowymi (format transakcyjny)

id_koszyka	id_produktu
1	banan
1	jabłko

id_koszyka	id_produktu
1	gruszka
2	jabłko
2	gruszka
3	banan
3	śliwka
3	gruszka

Jak wspomniano wcześniej, zanim zostanie wydane zapytanie SQL odkrywające zbiory częste, należy najpierw zdefiniować typ tabeli zagnieżdżonej dla zbiorów wynikowych:

```
CREATE TYPE fi_nested_char IS TABLE OF VARCHAR2(30);
/
```

Poniższe zapytanie zleca odkrycie zbiorów częstych w danych przedstawionych w Tabeli 2 przy progu minimalnego wsparcia równym 60% (0,6), przy czym zwrócone mają być tylko zbiory częste o rozmiarze 2 (2 jako jednocześnie minimalny i maksymalny wymagany rozmiar), zawierające element „jabłko". Zapytanie nie specyfikuje żadnych elementów, które nie mogą wystąpić w zbiorze wynikowym.

```
SELECT CAST(itemset AS fi_nested_char) itemset,
support, length, total_tranx
FROM TABLE(
          DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
          CURSOR(SELECT id_koszyka, id_produktu FROM koszyki),
          0.6, 2, 2,
          CURSOR(SELECT * FROM TABLE(fi_nested_char('jabłko'))),
          NULL));
```

W składni powyższego zapytania należy zwrócić uwagę na wykorzystanie zdefiniowanego wcześniej pomocniczego typu tabeli zagnieżdżonej zarówno do rzutowania wynikowych zbiorów, jak i stworzenia zbioru elementów wymaganych. Ponadto, charakterystyczne dla tego rodzaju zapytań jest wykorzystanie funkcji TABLE do udostępnienia zawartości kolekcji w postaci relacyjnej tabeli (zarówno dla kolekcji wynikowej jak i kolekcji specyfikujących ograniczenia na odkrywane zbiory).

4. Teoretyczne podstawy wykorzystywania zmaterializowanych wyników zapytań eksploracyjnych

Problematyka wykorzystania wyników poprzednich zadań odkrywania zbiorów częstych i reguł asocjacyjnych jest od lat obecna w literaturze naukowej. Motywację dla rozwiązań w tym zakresie stanowiła wizja interaktywnych systemów eksploracji danych, w których użytkownik będzie iteracyjnie modyfikował warunki zapytania eksploracyjnego aż do uzyskania satysfakcjonujących wyników. Taki scenariusz interakcji z systemem powinien prowadzić do sekwencji podobnych zapytań wysyłanych do systemu, co z kolei nasuwa pytanie o możliwość wykorzystania wyników poprzednich zapytań w celu uzyskania większej wydajności niż w przypadku przetwarzania każdego z zapytań „od zera", w oderwaniu od zapytań historycznych.

Spośród wielu zaproponowanych w literaturze modeli i technik wykorzystywania wyników poprzednich zapytań eksploracyjnych jako najodpowiedniejsze do implementacji w systemie Oracle zostało wybrane podejście przedstawione w [ZMW04, WZ05] ze względu na spełnienie poniższych trzech kluczowych z punktu widzenia planowanej implementacji warunków:

- operowanie na poziomie zbiorów częstych, a nie jak w wypadku wielu podejść konkurencyjnych już na poziomie reguł asocjacyjnych,
- możliwość wykorzystania wyników kilku poprzednich zapytań, a nie tylko jednego,
- niezależność od konkretnego algorytmu odkrywania zbiorów częstych dzięki operowaniu na poziomie zapytań eksploracyjnych i ich wyników.

Podstawowy pomysł zaproponowany w [ZMW04] to materializacja wyników zapytań (zbiory częste wraz z ich wsparciem) w formie tzw. materializowanych perspektyw eksploracyjnych z myślą o ich późniejszym wykorzystaniu do wydajnego wygenerowania odpowiedzi na kolejne zapytania. Pomysł ten oczywiście stanowi rozszerzenie o eksplorację danych obszaru zastosowań perspektyw materializowanych, w tym samych charakterze co w hurtowniach danych [Rou97].

Kluczowym problemem w wykorzystywaniu wyników poprzednich zapytań jest oczywiście nie sama materializacja, ale opracowanie algorytmów wykorzystywania wyników jednego zapytania w celu wygenerowania odpowiedzi na drugie oraz identyfikacja zależności składniowych między zapytaniami pozwalających na zaaplikowanie tych algorytmów. W [ZMW04] zidentyfikowano sześć klas zależności składniowych między zapytaniami odkrywającymi zbiory częste i zaproponowano dla nich stosowne algorytmy. Eksperymenty przeprowadzone na danych składowanych w plikach (bez użycia systemu zarządzania bazą danych) wykazały jednak, że tylko dla czterech z rozważanych sześciu klas zależności wykorzystanie poprzedniego zapytania skróciło czas wykonania nowego zapytania. Z oczywistych względów tylko te cztery pozytywnie zweryfikowane klasy zależności są istotne z punktu widzenia ewentualnej praktycznej implementacji mechanizmów wykorzystywania poprzednich zapytań zachowanych w formie materializowanych perspektyw eksploracyjnych. Te cztery klasy zależności i odpowiadające im algorytmy to:

1. Nowe zapytanie identyczne z zapytaniem perspektywy materializowanej.

   W tym trywialnym przypadku, nazwanym View Ready, zawartość perspektywy materializowanej jest bez żadnego przetwarzania zwracana jako wynik nowego zapytania.

2. Nowe zapytanie operujące na identycznym zbiorze danych co zapytanie perspektywy materializowanej, próg wsparcia i ograniczenia dla odkrywanych zbiorów częstych w nowym zapytaniu nie mniej restrykcyjne niż w zapytaniu perspektywy.

   Sposób postępowania w tym przypadku, nazwany Verifying Mining, sprowadza się do przejrzenia zawartości perspektywy materializowanej i zaaplikowania ograniczeń i progu wsparcia nowego zapytania do zawartych w niej zbiorów.

3. Nowe zapytanie operujące na nadzbiorze źródłowego zbioru danych perspektywy materializowanej, próg wsparcia i ograniczenia dla odkrywanych zbiorów częstych identyczne z zapytaniem perspektywy.

   Schemat postępowania w tym przypadku to Incremental Mining, polegający na odkryciu zbiorów częstych w zbiorze danych będącym różnicą źródłowego zbioru danych dla nowego zapytania i źródłowego zbioru danych zapytania perspektywy, a następnie zliczenia wystąpień odkrytych w ten sposób zbiorów i zbiorów z perspektywy. Poprawność tej metody wynika z faktu, że aby zbiór był częsty w zbiorze danych będących wynikiem połączenia dwóch rozłącznych zbiorów danych, musi on być częsty w przynajmniej jednym z tych zbiorów.

4. Nowe zapytanie operujące na nadzbiorze źródłowego zbioru danych perspektywy materializowanej, próg wsparcia i ograniczenia dla odkrywanych zbiorów częstych w nowym zapytaniu nie mniej restrykcyjne niż w zapytaniu perspektywy.

   W tym przypadku należy wykorzystać kombinację strategii Verifying Mining i Incremental Mining, tzn. najpierw wybrać z perspektywy zbiory spełniające próg wsparcia i ograniczenia dla zbiorów częstych nowego zapytania, następnie znaleźć zbiory częste w różnicy między danymi źródłowymi nowego zapytania a zapytania perspektywy spełniające ograniczenia

nia dla zbiorów częstych nowego zapytania i w końcu zliczyć wystąpienia wszystkich zebranych w dwóch pierwszych krokach zbiorów.

Powyższe strategie pochodzące z [ZMW04] dotyczyły wykorzystania pojedynczej materializowanej perspektywy eksploracyjnej. W [WZ05] strategia Incremental Mining została rozszerzona o możliwość wykorzystania kilku perspektyw materializowanych. W tym wypadku każda ze zbioru wykorzystywanych perspektyw w dalszym ciągu musi być oparta o zapytanie eksploracyjne na podzbiorze danych źródłowych nowego zapytania, a dodatkowo dane źródłowe zapytań o które oparte są te perspektywy nie mogą się nakładać. Możliwość wykorzystania zbioru perspektyw, a nie tylko jednej, w kontekście strategii Incremental Mining daje szansę większego pokrycia danych źródłowych nowego zapytania przez perspektywy, a przez to zmniejszenie całkowitej liczby odczytów bloków dyskowych.

5. Implementacja mechanizmów wykorzystywania wyników poprzednich zapytań eksploracyjnych w Oracle

5.1. Kluczowe problemy implementacyjne

Nie ulega wątpliwości, że system zarządzania bazą danych Oracle dzięki pakietowi DBMS_FREQUENT_ITEMSET stanowi przykład systemu umożliwiającego interaktywne i iteracyjne odkrywanie zbiorów częstych, którego wizja stanowiła punkt wyjścia dla teoretycznych rozważań na temat wykorzystywania poprzednich wyników eksploracji, opisanych w poprzedniej sekcji. Co więcej, ze względu na wielodostęp i możliwość współbieżnej pracy wielu użytkowników daje on możliwość wykorzystania nie tylko wyników poprzednich zapytań danego użytkownika, ale również innych użytkowników.

Implementacja w systemie Oracle opisanego w poprzedniej sekcji podejścia do wykorzystywania wyników poprzednich zapytań odkrywających zbiory częste wymagała rozwiązania trzech problemów:

- sposobu przechowywania wyników poprzednich zapytań i informacji o nich;
- implementacji technik wykorzystywania zmaterializowanych wyników i algorytmu wyboru odpowiednich spośród wszystkich dostępnych;
- automatycznej materializacji wyników zapytań eksploracyjnych wykonywanych w systemie.

Pierwszy z problemów został rozwiązany poprzez zaprojektowanie repozytorium w formie zestawu tabel bazy danych. Pozostałe dwa problemy ze względu na niemożność modyfikacji procedury dostarczanej przez Oracle zostały rozwiązane na zasadzie nakładki na procedurę Oracle w formie pakietu PL/SQL o nazwie FI_WRAPPER.

5.2. Repozytorium materializowanych wyników zapytań

Podstawowa koncepcja repozytorium sprowadza się do zapisywania wyników zapytań eksploracyjnych w bazie danych w formie tabel o schemacie identycznym z typem kolekcji zwracanej przez funkcję FI_TRANSACTIONAL. Tabele te stanowią implementację materializowanych perspektyw eksploracyjnych. Dodatkowo parametry zapytań i nazwy tabel z ich wynikami zapisywane są w dodatkowej tabeli pełniącej rolę słownika repozytorium. Schemat tabeli będącej słownikiem repozytorium przedstawiono w Tabeli 3.

Tabela 3. Schemat tabeli pełniącej rolę słownika repozytorium

Kolumna	Opis	Typ danych
table_name	nazwa tabeli wskazanej w zapytaniu	VARCHAR2(30) NOT NULL
tid_coll_name	nazwa kolumny zawierającej identyfikatory transakcji	VARCHAR2(30) NOT NULL
iid_coll_name	nazwa kolumny zawierającej identyfikatory elementów	VARCHAR2(30) NOT NULL
filter_coll_name	nazwa kolumny filtrującej	VARCHAR2(30)
filter_coll_type	typ kolumny filtrującej	VARCHAR2(30)
l_closure	dolne ograniczenie	VARCHAR2(30)
r_closure	górne ograniczenie	VARCHAR2(30)
min_sup	próg wsparcia	NUMBER NOT NULL
cache_table	nazwa tabeli zawierającej wyniki zapytania	VARCHAR2(30) NOT NULL
total_tranx	liczba transakcji	NUMBER NOT NULL
last_used	data ostatniego wykorzystania	DATE DEFAULT SYSDATE NOT NULL

Podczas projektowania repozytorium, ze względu na prototypowy charakter całej implementacji, przyjęto cztery istotne uproszczenia, które znalazły odzwierciedlenie w schemacie tabeli słownika repozytorium. Po pierwsze, przyjęto, że dane do eksploracji znajdują się w pojedynczej tabeli (ograniczenie to można obejść posługując się tradycyjną perspektywą). Po drugie, założono, że ograniczenie danych źródłowych do podzbioru wierszy tabeli źródłowej będzie możliwe tylko w oparciu o warunki przedziałowe dla jednej kolumny, przy czym przedział będzie lewostronnie domknięty, a prawostronnie otwarty. Po, trzecie zrezygnowano z ograniczeń dotyczących rozmiaru odkrywanych zbiorów i zawierania lub niezawierania przez nie konkretnych elementów. Należy tu podkreślić, że rezygnacja z tych ograniczeń nie skutkuje niemożnością weryfikacji techniki Verifying Mining, gdyż dotyczy ona również różnic miedzy progami wsparcia. Co więcej, nawet w rzeczywistej implementacji sensowne byłoby aplikowanie tych ograniczeń już po materializacji wyników uzyskanych bez ich uwzględnienia, w celu zwiększenia szansy wykorzystania wyników jednego zapytania przez drugie. Po czwarte, nie zaimplementowano kompletnego algorytmu zarządzania rozmiarem repozytorium, ograniczając się jedynie do pamiętania dla każdego zmaterializowanego wyniku zapytania czasu ostatniego jego użycia przez mechanizm przepisywania zapytań, co może stanowić podstawę do implementacji np. algorytmu LRU.

5.3. Pakiet FI_WRAPPER

Ze względu na brak dostępu do kodów źródłowych pakietu DBMS_FREQUENT_ITEMSET przyjęto rozwiązanie, w którym funkcje materializacji wyników zapytań i ich wykorzystywania są realizowane poprzez stworzony pakiet FI_WRAPPER, a konkretnie jego procedurę FI_TRANSACTIONAL_QR. Należy tu zwrócić uwagę, że takie podejście mogłoby być zastosowane nawet w produkcyjnej implementacji, gdyż daje ono użytkownikom swobodę wyboru czy chcą wykonać konkretne zapytanie eksploracyjne z wykorzystaniem mechanizmów przepisywania zapytań czy bez nich.

Pakiet FI_WRAPPER został zaimplementowany w całości w języku PL/SQL, ze względu na łatwość zagnieżdżania poleceń SQL i mały narzut czasowy związany z ich wykonywaniem oraz wydajną kompilację natywną podprogramów w języku PL/SQL dostępną w Oracle 11g. Poniżej zostaną ogólnie przedstawione kluczowe rozwiązania przyjęte podczas implementacji procedury

FI_TRANSACTIONAL_QR tego pakietu. Szczegółowe informacje na temat algorytmów przez nią implementowanych można znaleźć w [MW09].

Podstawowa idea procedury FI_TRANSACTIONAL_QR była taka, aby zestaw jej parametrów był identyczny z procedurą FI_TRANSACTIONAL, w celu uczynienia korzystania z pakietu FI_WRAPPER możliwie naturalnym dla użytkowników obeznanych z pakietem DBMS_FREQUENT_ITEMSET. Zdecydowano się jednak na jedno drobne odstępstwo polegające na zastąpieniu kursora zwracającego dane źródłowe do eksploracji zapytaniem SQL w formie tekstowej. Zmiana ta była podyktowana koniecznością dostępu do treści zapytania wybierającego dane źródłowe w celu zapamiętania go w repozytorium.

Działanie procedury FI_TRANSACTIONAL_QR rozpoczyna się od wybrania najkorzystniejszej strategii wykonania zapytania eksploracyjnego. W pierwszej kolejności szukana w repozytorium jest oczywiście perspektywa materializowana z wynikiem zapytania identycznego z bieżącym (strategia View Ready). Jeśli taka nie jest dostępna, szukana jest materializowana perspektywa pozwalająca na zastosowanie strategii Verifying Mining jako znacznie wydajniejszej od Incremental Mining w świetle wyników eksperymentów przedstawionych w [ZMW04]. Jeśli to poszukiwanie również nie zakończy się powodzeniem, poszukiwana jest materializowana perspektywa lub zbiór perspektyw dla strategii Incremental Mining. W przypadku wielu możliwości użycia metody Incremental Mining preferowany jest zestaw perspektyw w jak największym stopniu pokrywający dane źródłowe bieżącego zapytania, a gdy kilka zestawów perspektyw oferuje identyczny stopień pokrycia, wybierany jest spośród nich zestaw składający się z najmniejszej liczby perspektyw. W ostateczności, gdy nie jest możliwe wykorzystanie żadnej perspektywy materializowanej, odkrywanie zbiorów częstych przeprowadzane jest „od zera".

Zarówno w celu odkrywania zbiorów częstych „od zera" jak i w celu odkrycia zbiorów częstych w podzbiorze danych źródłowych w ramach strategii Incremental Mining wywoływana jest funkcja FI_TRANSACTIONAL pakietu DBMS_FREQUENT_ITEMSET. Oczywiście funkcja ta nie jest w ogóle wywoływana w wypadku zastosowania strategii View Ready lub Verifying Mining.

Wynik wykonywanego zapytania, niezależnie od strategii, która do niego doprowadziła, jest zapisywany w repozytorium materializowanych perspektyw eksploracyjnych w ostatnim kroku działania funkcji FI_TRANSACTIONAL_QR.

6. Wyniki eksperymentów

W celu ewaluacji prototypowej implementacji mechanizmów przepisywania zapytań odkrywających zbiory częste w oparciu o materializowane perspektywy eksploracyjne, opisanej w poprzedniej sekcji, przeprowadzono szereg eksperymentów na syntetycznych zbiorach danych. Poniżej przedstawione będą wyniki podstawowego eksperymentu na jednym testowym zbiorze danych. Zbiór ten został wygenerowany generatorem GEN [AMS+96] przy następujących ustawieniach: liczba transakcji = 100000, średnia liczba elementów w transakcji = 9, liczba różnych elementów = 1000, liczba wzorców = 1000, średnia długość wzorca = 4. Opis wyników pozostałych eksperymentów można znaleźć w [MW09].

Podstawowy eksperyment polegał na wykonaniu sekwencji sześciu zapytań eksploracyjnych funkcją FI_TRANSACTIONAL pakietu DBMS_FREQUENT_ITEMSET i funkcją FI_TRANSACTIONAL_QR pakietu FI_WRAPPER w celu porównania czasów wykonania. Sekwencja zapytań została dobrana w taki sposób, aby wystąpiły w trakcie jej wykonywania przypadki obsługiwane technikami Verifying Mining i Incremental Mining, a także konieczność odkrywania zbiorów częstych „od zera". Zapytania tworzące sekwencję różniły się zestawem danych źródłowych i progiem wsparcia. W zapytaniach nie były wykorzystywane ograniczenia dla odkrywanych zbiorów częstych, ze względu na wspomniane wcześniej przyjęte uproszczenia.

Testową sekwencję zapytań eksploracyjnych ilustruje Tabela 4. Wszystkie zapytania operowały na podzbiorze wygenerowanego testowego zbioru danych, zawierającego 100000 transakcji. Dolne i górne ograniczenie zbioru danych pokazane w tabeli oznaczają przedział transakcji z testowego zbioru stanowiący dane źródłowe danego zapytania.

Tabela 4. Sekwencja zapytań wykorzystana w eksperymencie

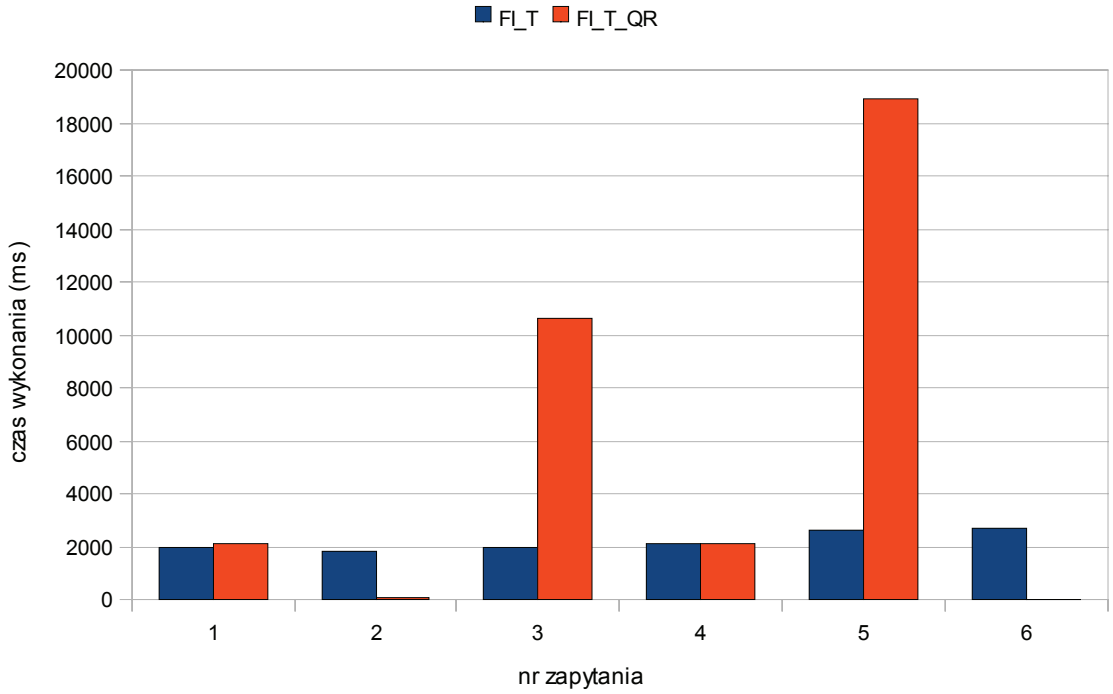
Lp.	Dolne ograniczenie zbioru danych	Górne ograniczenie zbioru danych	Próg wsparcia
1	80001	100000	1%
2	80001	100000	2%
3	60001	100000	2%
4	0	50000	2%
5	0	100000	2%
6	0	100000	2,5%

Zapytanie nr 1 wymagało oczywiście pełnego odkrywania zbiorów częstych, gdyż w chwili jego wykonywania repozytorium zmaterializowanych wyników poprzednich zapytań było puste. Zapytanie nr 2 było wykonane metodą Verifying Mining w oparciu o wyniki zapytania nr 1. Zapytanie nr 3 było wykonane metodą Incremental Mining z wykorzystaniem wyników zapytania nr 2. Zapytanie nr 4 wymagało pełnego odkrywania zbiorów częstych, gdyż żaden z wyników trzech poprzednich zapytań nie kwalifikował się do wykorzystania w jego przypadku. Zapytanie nr 5 było wykonane metodą Incremental Mining z wykorzystaniem wyników zapytań nr 3 i 4. Na koniec, zapytanie nr 6 było wykonane metodą Verifying Mining w oparciu o wyniki zapytania nr 5.

Czasy wykonania poszczególnych zapytań funkcjami FI_TRANSACTIONAL i FI_TRANSACTIONAL_QR przedstawia Rys. 1. Przede wszystkim, widoczne jest zredukowanie prawie do zera czasu wykonania zapytań w przypadkach gdy dostępna była metoda Verifying Mining (zapytania nr 2 i nr 6). Z kolei metoda Incremental Mining (zapytania nr 3 i nr 5) nie tylko nie skróciła czasu wykonania zapytań, ale nawet wydłużyła go kilkukrotnie, co zostanie dokładniej przeanalizowane poniżej. W końcu w sytuacji gdy konieczne było odkrywanie zbiorów częstych „od zera", dało się zaobserwować nieznaczne wydłużenie czasu wykonania dla FI_TRANSACTIONAL_QR, ze względu na konieczność materializacji wyników z myślą o kolejnych zapytaniach. Narzut ten jest jednak stosunkowo niewielki i może być uznany za akceptowalny.

Zaobserwowane zachowanie Incremental Mining było pewnym zaskoczeniem, ale można je wyjaśnić w oparciu o analizę implementacji Apriori w Oracle. Metoda Incremental Mining była projektowana z myślą o algorytmach eksploracji dokonujących wielokrotnych odczytów danych źródłowych, gdyż wymaga dodatkowego odczytu danych, który musi być skompensowany redukcją ilości danych odczytywanych przez algorytm bazowy. Ponieważ we wszystkich eksperymentach implementacja Oracle algorytmu Apriori wykonywała tylko jeden odczyt danych, następnie przechodząc na poziom bitmap w pamięci, dodatkowy odczyt wykonywany na zakończenie Incremental Mining nie miał szans być skompensowany. Drugim i bardziej istotnym problemem metody Incremental Mining był sposób implementacji testów zawierania odkrywanych zbiorów częstych w transakcjach w trakcie wspomnianego dodatkowego końcowego odczytu danych. Implementacja Incremental Mining realizowała test zawierania w sposób tradycyjny, podczas gdy implementacja Apriori w Oracle wykorzystywała do tego celu bitmapy. Znaczenie kosztu realizacji testów zawierania staje się jasne po porównaniu czasów dla zapytań nr 3 i nr 5. W przypadku zapytania nr 5 pokrycie jego danych źródłowych przez perspektywy materializowane było większe niż dla zapytania nr 3, a mimo to metoda Incremental Mining wypadła względnie gorzej dla zapytania nr 5. Powodem jest fakt, że zapytanie nr 5 wykorzystywało dwie perspektywy materializowane, co wymagało zliczenia większej liczby zbiorów w czasie dodatkowego odczytu bazy

danych niż w przypadku zapytania nr 3 wykorzystującego tylko jedną perspektywę materializowaną.



Rys. 1. Porównanie czasów wykonania zapytań przez funkcje FI_TRANSACTIONAL (FI_T) i FI_TRANSACTIONAL_QR (FI_T_QR) dla testowego zbioru danych

7. Podsumowanie

W niniejszym artykule przedstawiono implementację w systemie Oracle technik wykorzystania wyników poprzednich zapytań odkrywających zbiory częste opartych o koncepcję materializowanych perspektyw eksploracyjnych. Implementacja ma postać pakietu FI_WRAPPER z funkcją FI_TRANSACTIONAL_QR „opakowującą" wywołanie funkcji FI_TRANSACTIONAL pakietu DBMS_FREQUENT_ITEMSET dostępnego w bazie danych Oracle. Pakiet FI_WRAPPER w sposób niewidoczny dla użytkownika wyszukuje i wykorzystuje zmaterializowane w specjalnie zaprojektowanym repozytorium wyniki poprzednich zapytań oraz materializuje w repozytorium wyniki bieżącego zapytania po jego wykonaniu.

Opisana w artykule implementacja ma zdecydowanie charakter prototypowy ze względu na przyjęte uproszczenia w zakresie warunków selekcji danych źródłowych. Niemniej, pozwala ona sformułować kilka wniosków odnośnie praktycznej użyteczności koncepcji materializowanych perspektyw eksploracyjnych oraz wydajności implementacji algorytmu Apriori w Oracle. Po pierwsze, prototypowa implementacja stanowi potwierdzenie, że rozszerzenie systemu zarządzania bazą danych umożliwiającego odkrywanie zbiorów częstych z poziomu zapytań SQL o materializowane perspektywy eksploracyjne może być zrealizowane w nieinwazyjny sposób, nawet w formie nakładki na istniejącą funkcjonalność. Po drugie, o ile skuteczność metody Verifying Mining nie podlega dyskusji nawet w kontekście wysoce zoptymalizowanego bazowego algorytmu eksploracji, to praktyczna przydatność techniki Incremental Mining w tym wypadku w świetle eksperymentów z biblioteką FI_WRAPPER wydaje się wątpliwa. Po trzecie, implementację algorytmu Apriori dostarczaną przez Oracle należy ocenić jako wysoce wydajną. Apriori w wersji Oracle, dzięki wykorzystaniu pamięciowych indeksów bitmapowych, do minimum ogranicza

odczyty danych źródłowych z dysku. Jest to najprawdopodobniej powód zaobserwowanej nieskuteczności nałożonej na bibliotekę Oracle metody Incremental Mining, która była projektowana dla algorytmów dokonujących wielokrotnych odczytów danych źródłowych z dysku.

Bibliografia

[AIS93] Agrawal R., Imielinski T., Swami A.: Mining Association Rules Between Sets of Items in Large Databases, Proc. of the 1993 ACM SIGMOD Conference on Management of Data, 1993.

[AMS+96] Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T.: The Quest Data Mining System, Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

[AS94] Agrawal R., Srikant R.: Fast Algorithms for Mining Association Rules, Proc. of the 20th International Conference on Very Large Data Bases, 1994.

[FPS96] Fayyad U., Piatetsky-Shapiro G., Smyth P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data, Communication of the ACM, 39(11), 1996.

[IM96] Imielinski T.,, Mannila H..: A Database Perspective on Knowledge Discovery, Communication of the ACM, 39(11), 1996.

[LHM+05] Li W., Huang J., Mozes A., Thomas S., Callaghan M.: Dynamic selection of frequent itemset counting technique, United States Patent Application 20050044087, 2005.

[MW09] Mierzwiński D., Wojciechowski M.: Biblioteka do obsługi przepisywania zapytań eksploracyjnych dla systemu Oracle 11g, Raport Instytutu Informatyki Politechniki Poznanskiej RB-03/09, 2009.

[Ora08] Oracle Corporation, Oracle Database PL/SQL Packages and Types Reference 11g Release 1 (11.1), 2008.

[PH00] Pei J., Han J.: Can We Push More Constraints into Frequent Pattern Mining?, Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[Rou97] Roussopoulos N.: Materialized Views and Data Warehouses, SIGMOD Record 27(1), 1997.

[WZ05] Wojciechowski M., Zakrzewicz M.: Efficient Processing of Frequent Itemset Queries Using a Collection of Materialized Views, Proc. of the International IIS: IIPWM'05 Conference, 2005.

[ZMW04] Zakrzewicz M., Morzy M., Wojciechowski M.: A Study on Answering a Data Mining Query Using a Materialized View, Proc. of the 19th International Symposium on Computer and Information Sciences, 2004.