STUDIA INFORMATICA 2007
Volume 28 Number 3A (72)

Marcin GORAWSKI, Szymon PANFIL
Politechnika Śląska, Instytut Informatyki

DEKOMPOZYCJA RELACJI JAKO METODA OCHRONY PRYWATNOŚCI DANYCH W ROZPROSZONYCH PRZESTRZENNYCH HURTOWNIACH DANYCH

Streszczenie. Artykuł przedstawia dekompozycję relacji jako metodę ochrony poufności informacji w nowej architekturze rozproszonej przestrzennej hurtowni danych (DSDW). Każdy z węzłów w nowej architekturze DSDW posiada własny moduł ochrony prywatności, mogący się komunikować z innymi modułami. Umożliwia to przetwarzanie zapytań równolegle przez wszystkie moduły, a w rezultacie daje znaczne skrócenie czasu realizacji zapytania w DSDW.

Słowa kluczowe: hurtownie danych, ochrona prywatności, dekompozycja relacji, kodowanie atrybutu, bezpieczeństwo, poziome rozproszenie danych, pionowe rozproszenie.

RELATION DECOMPOSITION AS A METHOD OF PRESERVING DATA PRIVACY IN DISTRIBUTED SPATIAL DATA WAREHOUSE

Summary. The article presents an implementation of the relation decomposition as a method of preserving data confidentiality in a new distributed spatial data warehouse architecture that we proposed. Each node in this new architecture has its own privacy preserving module which is able to communicate with other modules. It enables parallel query processing through all privacy preserving modules and as a result it shortens the time of query implementation in DSDW.

Keywords: data warehouse, privacy preserving, relation decomposition, attribute coding, security, horizontally distributed data, vertically distributed data.

1. Wprowadzenie

Rosnąca popularność zaawansowanych systemów hurtowni danych (DW) idzie w parze z coraz większą dbałością o ich bezpieczeństwo. Systemy DW przechowują dane, których

udostępnienie osobom niepowołanym może powodować naruszenie ich prywatności. Zatem projektując takie systemy, trzeba uwzględnić coraz bardziej restrykcyjne przepisy ochrony danych osobowych i informacji prywatnych.

Zwykle poufna informacja złożona jest ze zbioru powiązanych ze sobą danych, co powoduje, że informacja ta nie powinna być dostępna w całości dla użytkownika bez uprawnień autoryzacji. Mogą być jednak dostępne pojedyncze dane lub fragmenty informacji. Posiadając tylko dane o zarobkach czy dacie urodzenia, nie jesteśmy w stanie uzyskać wiedzy, która naruszałaby prywatność informacji. Dopiero gdy dane te powiążemy np. z nazwiskiem osoby, może mieć miejsce naruszenie prywatności.

Istnieje kilka sposobów ochrony prywatności danych. Jednym z nich jest dodawanie do tabel krotek zaszumiających, w taki sposób, że prawdopodobieństwo, by którykolwiek zbiór wartości atrybutów był częścią „rzeczywistej" krotki, było mniejsze od ustalonej wartości. Inną metodą jest tzw. *k*-anonimowość, wykorzystywana na przykład przy zapewnieniu anonimowości lokalizacji [1]. Stosując takie rozwiązanie, uzyskujemy sytuację, że wrażliwe pod względem prywatności dane dotyczą co najmniej *k* różnych podmiotów (np. osób).

Obiektami w systemach DW, które muszą być chronione, nie są (jak w przypadku transakcyjnych systemów) pliki lub tabele, lecz wymiary w wielowymiarowych kostkach danych, hierarchie wewnątrz wymiarów oraz fakty [2, 3]. W pracy [4] przedstawiono podstawowy model autoryzacji dla systemu DW, bazujący na modelu podmiotowym i modelu obiektowym. Model podmiotowy odwołuje się tylko do użytkowników w kontekście związku między hierarchiami ról i wymiarów. Nakreślono nieformalny język autoryzacji pozwalający formułować ograniczenia dostępu do DW na podstawie modelu obiektowego.

Jako alternatywę przedstawiono w pracy [5] system rozproszonej przestrzennej hurtowni danych (ang. *Distributed Spatial Data Warehouse*, DSDW), bazujący na metodzie takiego rozdzielenia danych pomiędzy jego węzłami, że wyjawienie danych tylko z jednego węzła nie powoduje naruszenia prywatności. W przypadku gdy dane stanowiące informację prywatną nie mogą zostać rozdzielone, można je zakodować, wykorzystując istnienie kilku rozproszonych węzłów, bez potrzeby korzystania z dodatkowego serwera kluczy szyfrowania.

Prezentowana niżej nowa architektura systemu DSDW z ochroną prywatności danych jest kontynuacją badań podjętych w pracy [5].

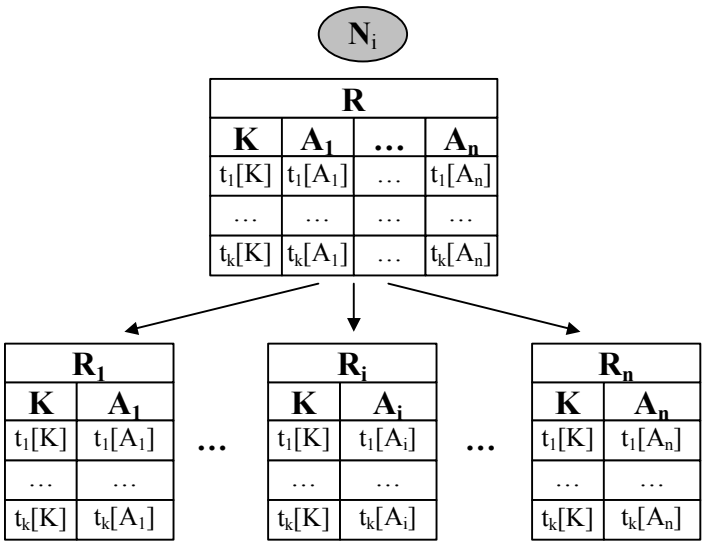
2. Ochrona prywatności bazująca na rozdzieleniu danych

Systemy rozproszonej przestrzennej hurtowni danych DSDW opisano w pracach [5, 6, 7, 8]. W pracy [5] zaproponowano dwie techniki ochrony prywatności danych w DSDW: dekompozycja relacji oraz kodowanie atrybutów.

2.1. Dekompozycja relacji

Dekompozycja relacji pozwala na rozdzielenie *n* atrybutów pomiędzy *m* węzłów DSDW. Konieczne dla zachowania prywatności jest spełnienie warunku $n \leq m$, co pozwoli nam na umieszczenie każdego z poufnych atrybutów w osobnym węźle DSDW. Innym sposobem ochrony prywatności przez dekompozycję jest usunięcie z każdego węzła jednego atrybutu ze zbioru atrybutów chronionych. W takim przypadku istnieje ryzyko, że jeśli włamującemu uda się dostać do dwóch węzłów przechowujących chronione dane, to jest on w stanie uzyskać pełną poufną informację. Niemniej jednak przy udanym ataku tylko na jeden węzeł takiej informacji nie jest w stanie uzyskać.

Rozważmy relację *R* zawierającą zbiór atrybutów $A_1,\ldots,A_n$. Zgodnie z powyższymi założeniami, dla zapewnienia ochrony prywatności danych rozważanej relacji konieczne jest jej rozdzielenie na *n* podrelacji. Relacja R_i zawiera wartości atrybutu A_i oraz wartości atrybutu K, będącego kluczem głównym. Na rys. 1 [5] przedstawiony został schemat dekompozycji relacji.



Rys. 1. Schemat dekompozycji relacji
Fig. 1. Schema of decomposition relation

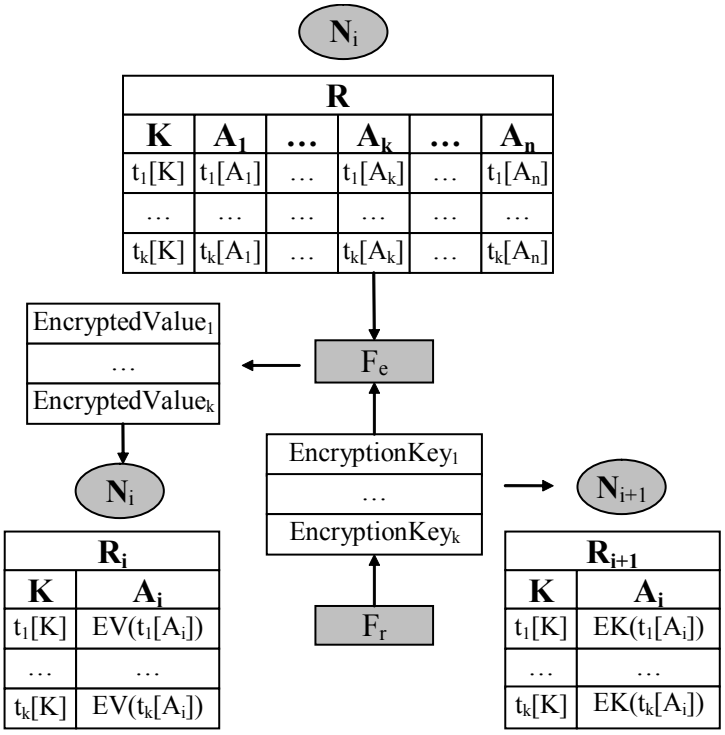
Jeżeli zaatakowany zostanie węzeł N_i, ujawnione zostaną tylko wartości atrybutu A_i, zachowując nadal pełną ochronę prywatności informacji. Tylko w przypadku uzyskania dostępu do wszystkich *n* węzłów DSDW miałoby miejsce naruszenie prywatności. Dla przykładu, jeżeli poufna informacja jest składowana na dwóch węzłach, to atakujący musiałby uzyskać dostęp do tych dwóch węzłów, na których składowane są atrybuty. Jednakże gdyby prywatna informacja została rozdzielona pomiędzy pięć węzłów, to jej ujawnienie wymagałoby od atakującego uzyskania dostępu do wszystkich pięciu węzłów. Jak widać, wraz ze wzrostem liczby węzłów, pomiędzy które rozdzielona została prywatna informacja, zmniejsza się prawdopodobieństwo jej ujawnienia.

2.2. Kodowanie atrybutu

Dekompozycja relacji rozdziela pomiędzy węzłami hurtowni danych zbiór atrybutów, które umieszczone wspólnie w jednej bazie danych mogą być źródłem naruszenia prywatności. Możliwa jest jednak sytuacja, w której prywatną informację stanowią wartości pojedynczego atrybutu. W takim przypadku dekompozycja relacji traci sens, a jednym z rozwiązań może być kodowanie atrybutu [5].

Idea kodowania atrybutu w architekturze DSDW polega na rozdzieleniu wartości wybranego atrybutu pomiędzy dwa lub więcej węzłów rozproszonej przestrzennej hurtowni danych. Węzły DSDW należy podzielić na dwie grupy: pierwsza, w bazach danych której przechowywane będą wartości zakodowane atrybutu, oraz druga, w bazach danych której przechowywane będą wartości kluczy szyfrowania. Takie podejście zapewnia nam w pełni ochronę prywatności wartości wybranego atrybutu, gdyż na żadnym z węzłów nie znajdują się jego rzeczywiste wartości. Alternatywnym rozwiązaniem dla zwiększenia wydajności wykonywania zapytań w architekturze DSDW jest umieszczenie w pewnej liczbie węzłów wartości niezakodowanych tego atrybutu. Pociąga to jednak za sobą zmniejszenie stopnia ochrony prywatności danego atrybutu.

Na rys. 2 [5] zaprezentowany został schemat kodowania atrybutu A_i w węźle N_i. W miejsce oryginalnych wartości wstawione zostały w węźle N_i wartości zakodowane (np. dla komórki t_1 jest to $EV(t_1[A_1])$) oraz w węźle N_{i+1} wartości kluczy szyfrowania (gdzie np. $EK(t_1[A_1])$ oznacza wartość klucza szyfrowania odpowiadającego wartości atrybutu A_i w krotce t_1). Proces kodowania atrybutu A_i przebiega następująco: losowa funkcja F_r generuje klucze szyfrowania, które następnie wraz z oryginalnymi wartościami atrybutu trafiają na wejście funkcji szyfrującej F_e. Funkcja szyfrująca zwraca zbiór wartości zakodowanych, gdzie każdej z tych wartości odpowiada wartość jej klucza szyfrowania. Zatem, dodatkowy, zaufany serwer generujący i przechowujący klucze szyfrowania nie jest potrzebny, gdyż wszystkie potrzebne dane w procesie dekodowania znajdują się w węzłach DSDW.



Rys. 2. Schemat kodowania atrybutu
Fig. 2. Schema of attribute coding

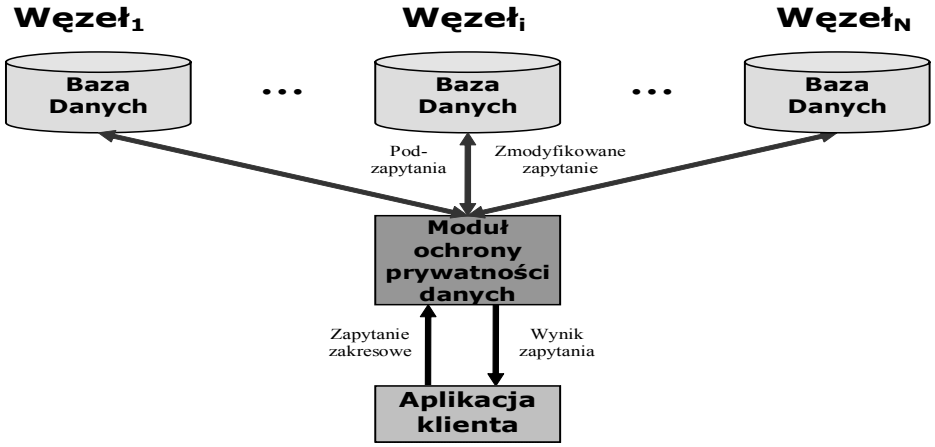
Funkcja szyfrująca F_e może być zrealizowana wg jednego z popularnych algorytmów (np. DES, AES, IDEA), a także prostą funkcją matematyczną (np. suma logiczna, dodawanie). W przeprowadzonych testach użyty został ostatni z zaproponowanych typów funkcji F_e. Wykorzystywane przez tę funkcję w procesie kodowania losowe wartości kluczy szyfrowania przechowywane są w jednym węźle, a w drugim wynik dodawania tych wartości do oryginalnych wartości kodowanego atrybutu.

Podobnie jak w przypadku ochrony prywatności danych przez dekompozycję relacji również dla kodowania atrybutu musi istnieć co najmniej jeden atrybut kluczowy o unikalnych wartościach, który pozwoli połączyć wartości zaszyfrowane znajdujące się w relacji R_i z odpowiadającymi im wartościami kluczy szyfrowania znajdującymi się w relacji R_{i+1}.

3. Bazowa architektura DSDW

Niniejsza praca jest kontynuacją badań nad ideą ochrony prywatności danych poprzez ich rozdzielenie w rozproszonych przestrzennych hurtowniach danych [5]. Głównym jej celem było wdrożenie **modułu ochrony prywatności danych** (*Data Privacy Preserving Module,* **DPPM**), który pośredniczyłby między bazami danych rozproszonego systemu a aplikacją klienta. Moduł ten może znajdować się w jednym z węzłów DSDW lub poza rozproszoną

hurtownią danych na odległej maszynie. Poniżej (rys. 3 [5]) przedstawiona została architektura rozproszonego systemu.



Rys. 3. Bazowa architektura DSDW
Fig. 3. Basic DSDW architecture

3.1. Moduł ochrony prywatności danych

W ramach modułu ochrony prywatności danych (DPPM) wyodrębnione zostały dwa podmoduły. Za wdrożenie określonej przez użytkownika polityki prywatności danych w węzłach hurtowni danych odpowiedzialny jest **moduł zarządzania ochroną prywatności DPPMU**. Definiując politykę prywatności, należy określić, które z tabel wymiarów (tabel) mogą zawierać poufne informacje, a następnie dla nich wybrać atrybuty, które poddane zostaną dekompozycji relacji, oraz te, których wartości należy zakodować. Ostatecznie w wyniku wdrożenia polityki prywatności hurtownię danych cechuje fragmentacja pionowa tabel wymiarów oraz zreplikowana na każdym z węzłów tabela faktów. Założenie replikowania tabeli faktów wprowadzało znaczne uproszczenie procedury przetwarzania zapytań i jednocześnie nie modyfikowało idei działania rozważanych metod ochrony prywatności danych.

Kolejnym modułem wchodzącym w skład DPPM jest **moduł przetwarzania zapytań DPPMQ** – odpowiedzialny za wykonywanie zapytań w DSDW chronionej pod względem prywatności określonej w DDPMU. Każde zapytanie wykonywane jest jako wywołanie funkcji DPPMQ najpierw sprawdzającej, czy wartości atrybutów, do których odwołuje się zapytanie, są zdekomponowane lub zakodowane. Jeżeli nie – zapytanie może zostać wykonywane na bazie danych dowolnego węzła, a jeżeli moduł ochrony prywatności danych został uruchomiony w ramach jednego z węzłów, to na jego lokalnej bazie danych. W przeciwnym razie realizowana jest odpowiednia procedura wykonania zapytania na danych chronionych w rozproszonej architekturze, której algorytm został przedstawiony poniżej.

1. Pobierz tekst zapytania od aplikacji klienta.
2. Dla każdego warunku selekcji odwołującego się do atrybutu chronionego we frazie ‘WHERE’ zapytania stwórz *podzapytanie*, pozwalające pobrać wartości klucza głównego z tej samej relacji, spełniające dany warunek. Pobrane wartości wstaw jako nowy warunek IN w miejsce analizowanego warunku.
3. Odwołania do atrybutów chronionych we frazie ‘SELECT’ zapytania zastąp odwołaniami do kluczy głównych z tej samej relacji.
4. Wykonaj *zmodyfikowane zapytanie* (pozbawione wystąpień atrybutów chronionych) w dowolnym węźle hurtowni danych.
5. Jeżeli bazowe zapytanie nie posiadało odwołań do atrybutów chronionych w frazie ‘SELECT’, to przejdź do kroku 8. W przeciwnym przypadku przejdź do kroku 6.
6. Dla każdego atrybutu kluczowego (zastępującego atrybut chroniony) we frazie ‘SELECT’ zmodyfikowanego zapytania stwórz *podzapytanie*, pozwalające pobrać wartości chronione odpowiadające pobranym w punkcie 4 wartościom analizowanego atrybutu.
7. Pobrane wartości wstaw w miejsce wartości kluczy głównych.
8. Wynik odeślij aplikacji klienta.

Powyższy algorytm w uproszczeniu opisuje procedurę wykonywania zapytań na danych chronionych. Kroki 2 i 3 algorytmu są jednymi z ważniejszych, w ramach których wystąpienia atrybutów chronionych w bazowym tekście zapytania zastępowane są wystąpieniami odwołującymi się do atrybutów kluczowych z tej samej relacji. Stworzone w ten sposób zmodyfikowane zapytanie pozbawione jest wystąpień atrybutów chronionych, a więc może zostać zrealizowane w bazie danych dowolnego węzła DSDW. Liczba kroków modyfikacji zapytania w dużej mierze zależy od tego, które z fraz zapytania posiadają wystąpienia atrybutów chronionych. W omawianym rozwiązaniu moduł DPPMQ umożliwiał przetwarzanie zapytań zarówno pobierających wartości atrybutów chronionych, jak i wykonujących na nich warunki selekcji. Dodatkowo realizowane były zapytania sortujące wynik. Szczegółowa analiza tego algorytmu zawarta została w [5].

4. Nowa architektura DSDW

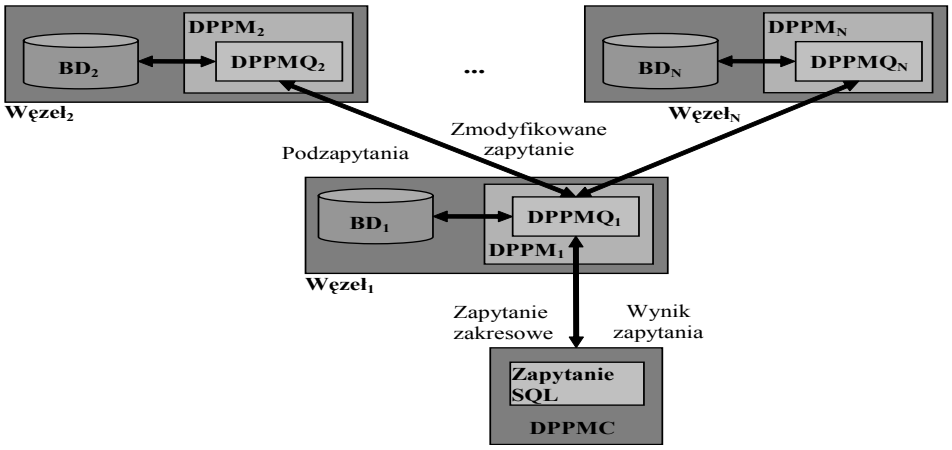
Głównym komponentem opisanej wyżej architektury DSDW był DPPM, który umożliwiał wdrożenie określonej przez użytkownika polityki prywatności, wykorzystując dekompozycję relacji i kodowanie atrybutu. Dalsze prace skoncentrowały się na przetwarzaniu podstawowych typów zapytań OLAP, pobierających wartości chronione oraz wykonujących na wartościach chronionych warunki *selekcji*. Wykonywanie tego typu zapytań przez pojedyn-

czy moduł, jak miało to miejsce w [5], nie wymagało długiego czasu realizacji, a jednocześnie zaimplementowanie dla nich procedur przetwarzania stanowiło podstawę budowy parsera SQL.

W przypadku standardowych zapytań OLAP poza pobieraniem wartości jest również wykonywane np. sortowanie czy grupowanie wyniku, dla którego czas realizacji w omówionej architekturze DSDW jest znaczący. Wynika to ze złożoności obliczeniowej, jaką cechują się tego typu operacje – pokazały to testy zapytań z klauzulą 'ORDER BY', których obsługę zaimplementowano w DPPMQ.

Problem ten pozwolił nakreślić dalszy kierunek rozwoju, którym było rozproszenie procesu wykonywania zapytań pomiędzy węzły systemu, tak by proces ten realizowany był w sposób bardziej efektywny. Aby móc to osiągnąć, każdy z węzłów DSDW powinien zawierać własny DPPM, którego moduł wykonywania zapytań, posiadając dostęp do lokalnej bazy danych i jeśli to konieczne do innych baz danych, uczestniczyłby wraz innymi DPPMQ we współbieżnej realizacji zapytania. Dodatkowo, dla realizacji tej idei konieczne jest uwzględnienie w DSDW zaniechanej we wcześniejszym rozwiązaniu poziomej fragmentacji tabeli faktów pomiędzy wszystkie jej węzły. Ostatecznie po wdrożeniu nowej architektury DSDW zgodnej z powyższymi założeniami oraz przeniesieniu do niej możliwości wykonywania dotychczasowych typów zapytań dalsze prace skoncentrowały się nad poszerzaniem parsera SQL o inne typowe zapytania OLAP. W rezultacie uzyskano nową architekturę o większej funkcjonalności wykonywania analiz przy zachowaniu prywatności danych.

W nowej architekturze systemu DSDW zmodyfikowano moduł wykonywania zapytań (DPPMQ) oraz schemat połączeń pomiędzy poszczególnymi komponentami systemu. Każdy z węzłów w takiej architekturze posiada własną bazę danych oraz moduł ochrony prywatności, który ma możliwość komunikacji z pozostałymi modułami oraz za ich pośrednictwem z ich lokalnymi bazami danych. Dodatkowo aplikacja klienta może zostać podłączona do DPPM dowolnego węzła, po czym jego DPPMQ staje się nadzorcą całego procesu realizacji zapytań. Na rys. 4 przedstawiona została nowa architektura systemu DSDW oraz algorytm wykonywania zapytań na danych chronionych w systemie o takiej architekturze.



Rys. 4. Nowa architektura DSDW
Fig. 4. New DSDW architecture

1. Pobierz tekst zapytania od aplikacji klienta.
2. Dla każdego warunku selekcji we frazie ‘WHERE’ zapytania, odwołującego się do atrybutu chronionego, stwórz *podzapytanie,* pozwalające pobrać wartości klucza głównego z tej samej relacji, spełniające dany warunek. Pobrane wartości wstaw jako nowy warunek IN w miejsce analizowanego warunku.
3. Odwołania do atrybutów chronionych we frazie ‘SELECT’ zapytania zastąp odwołaniami do kluczy głównych z tej samej relacji.
4. Dla każdego atrybutu chronionego, którego wartości są pobierane przez zapytanie, stwórz *podzapytanie,* pozwalające pobrać wszystkie jego wartości wraz z odpowiadającymi wartościami klucza głównego z tej samej relacji.
5. *Zmodyfikowane zapytanie* - pozbawione wystąpień atrybutów chronionych – wraz z pobranymi w kroku 4 danymi prześlij pozostałym modułom wykonywania zapytań.
6. Każdy z modułów wykonuje zapytanie na lokalnej bazie danych, a następnie zastępuje wartości kluczy głównych odpowiadającymi im wartościami chronionymi.
7. Moduł zarządzający odbiera od pozostałych modułów wykonywania zapytań wynikowe zbiory wierszy, a następnie wszystkie zbiory łączy w jeden zbiór wierszy, będący wynikiem zapytania.
8. Odeślij wynik zapytania do aplikacji klienta.

Dla ułatwienia zrozumienia idei przetwarzania zapytań w powyższej architekturze rozproszonego systemu zaprezentowany został algorytm dla zapytań, które posiadają odwołania do atrybutów chronionych we frazie ‘SELECT’ oraz ‘WHERE’. Procedury przetwarzania danych w DPPMQ zostały tak zaimplementowane, aby współbieżnie wykonywana (przez wszystkie moduły) była jak największa liczba etapów danego typu zapytań. W powyższym przykładzie równolegle wykonywany jest tylko krok 6. Niemniej jednak dla liczby wierszy, jaka zwykle jest zwracana jako wynik tego rodzaju zapytań w hurtowniach danych, różnica czasu wykonania w porównaniu z poprzednim rozwiązaniem jest znacząca. Wynika to z fak-

tu, iż w kroku 6 każdy DPPMQ poza wykonaniem zapytania pozbawionego atrybutów chronionych wstawia jeszcze do otrzymanego wyniku w miejsce wartości kluczy głównych odpowiednie wartości chronione. Zatem, dysponując pojedynczym modułem przetwarzania zapytań (jak miało to miejsce w [5]) etap ten dla dużej liczby wierszy zwracanych w wyniku zapytania odbywałby się znacznie dłużej. Na uwagę zasługuje również krok 4 procedury, w którym konstruowane są podzapytania, pozwalające pobrać wszystkie wartości każdego z chronionych atrybutów frazy 'SELECT'. Nie jest to jednak kosztowna operacja, gdyż jest ona realizowana na wartościach atrybutów z tabel wymiarów, których rozmiar jest niewielki w hurtownia danych.

Zatem, efektywność nowej architektury uwydatnia się w przypadku typowych dla OLAP zapytań, na których realizację składa się większa liczba etapów, mogących być wykonanych współbieżnie przez każdy z węzłów rozproszonego systemu. Aktualnie moduł ochrony prywatności pozwala na wykonywanie zapytań z atrybutami chronionymi we frazach *select*, *where*, *order by* oraz *group by*. Dodatkowo, możliwe jest definiowanie funkcji agregacji, takich jak: *min*, *max*, *count*, *sum*, *avg*. W poprzednim rozwiązaniu [5] możliwe było przetwarzanie zapytania z atrybutami chronionymi w pierwszych trzech frazach spośród wymienionych. Szczegółowa analiza wykonywania zapytań na danych chronionych w nowej architekturze przeprowadzona została w punkcie 5.

5. Wykonywanie zapytań na danych chronionych

Poniżej przedstawione jest przykładowe zapytanie. Spośród tabel, do których się ono odwołuje, na tabeli LOCATION została uprzednio wykonana dekompozycja relacji. Atrybuty LOCX oraz LOCY tej tabeli znajdują się odpowiednio w węzłach N1, N2, natomiast atrybut LOCZ został poddany kodowaniu, w wyniku czego wartości zakodowane znajdują się w węźle N1, a wartości kluczy szyfrowania w węźle N2. Dodatkowo, przykładowe zapytanie zawiera warunek selekcji na atrybucie LOCX oraz pobiera wartości atrybutu VALUE z tabeli faktów. W dalszej części rozdziału zaprezentowane zostało przetwarzanie zapytania, obrazujące przedstawiony wcześniej algorytm wykonywania zapytań. Zatem, otrzymane od aplikacji klienta zapytanie przedstawia się następująco:

```
select L.LOCX, L.LOCY, L.LOCZ, M.VALUE
from COUNTER C, LOCATION L, MEASURE M
where M.COUNTERUD = C.COUNTERID
and C.LOCATIONID = L.LOCATIONID
and L.LOCX > 100
```

Zarządzający moduł dla każdego warunku selekcji z atrybutem chronionym pobiera wartości kluczowego atrybutu, spełniające zadany warunek:

```
select LOCATIONID
from LOCATION
where LOCX > 100
```

Utworzone *podzapytanie* jest wysyłane do węzła, na którym jest składowany atrybut *LOCX* (N_1). Wynikowy zbiór identyfikatorów (*RES_ID*) zostaje wykorzystany do stworzenia nowego warunku selekcji w bazowym zapytaniu:

```
select L.LOCX, L.LOCY, L.LOCZ, M.VALUE
from COUNTER C, LOCATION L, MEASURE M
where M.COUNTERUD = C.COUNTERID
and C.LOCATIONID = L.LOCATIONID
and L.LOCATIONID IN (RES_ID)
```

Każde wystąpienie atrybutu chronionego we frazie select jest zastępowane wystąpieniem atrybutu kluczowego z tej samej relacji. Wszystkie rozważane poufne atrybuty należą do tabeli LOCATION. Zatem, aby zoptymalizować zapytanie, usuwane są duplikaty atrybutów kluczowych z frazy select:

```
select L.LOCATIONID, M.VALUE
from COUNTER C, LOCATION L, MEASURE M
where M.COUNTERUD = C.COUNTERID
and C.LOCATIONID = L.LOCATIONID
and L.LOCATIONID IN (RES_ID)
```

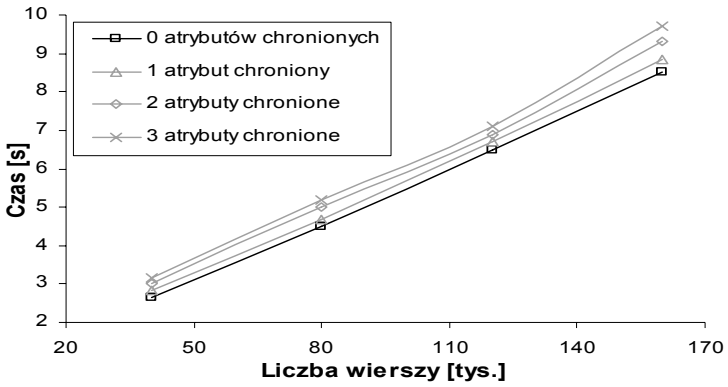
Dla każdego atrybutu chronionego we frazie select pobierane są jego wartości wraz z odpowiadającymi im wartościami atrybutu kluczowego. Pierwsze dwa podzapytania wykonywane są odpowiednio w węzłach N1 i N2, a ostatnie w obu węzłach, skąd pobieranie są wartości zakodowane oraz klucze szyfrowania:

```
select LOCATIONID, LOCX
from LOCATION //LOCX

select LOCATIONID, LOCY
from LOCATION //LOCY

select LOCATIONID, LOCZ
from LOCATION //LOCZ
```

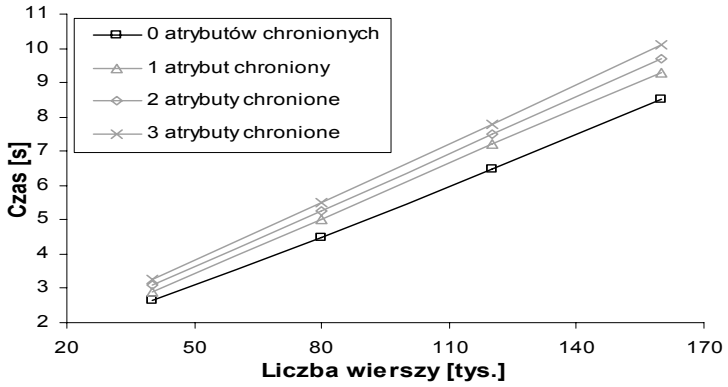
Zmodyfikowane zapytanie nie zawiera wystąpień atrybutów chronionych, więc może zostać wykonane przez każdy z modułów ochrony prywatności w lokalnej bazie danych. Pojedynczy moduł poza tekstem zapytania otrzymuje wartości atrybutów chronionych wraz z odpowiadającymi im wartościami atrybutów kluczowych. Ponadto, po wykonaniu zmodyfikowanego zapytania jego wynik jest uzupełniany o duplikaty kolumn atrybutów kluczowych, a następnie wartości chronione. Zarządzający moduł ochrony prywatności łączy wszystkie wyniki zmodyfikowanego, głównego zapytania w jeden zbiór wynikowy, który odsyła do aplikacji klienta.

6. Testy

Przeprowadzone testy miały na celu wyznaczenie oraz analizę czasu wykonywania zapytań w nowej architekturze DSDW. Do testów wykorzystano dwie maszyny o podobnej konfiguracji: Windows XP, P4 2,8 GHz / 2,4 GHz, 512 MB RAM. Pojedyncza maszyna opowiadała węzłowi nowej architektury, na którym uruchomiona była baza danych Oracle oraz moduł wykonywania zapytań – DPPMQ. Aplikacja klienta łączyła się z jednym z DPPMQ, który rozpoczynał procedurę przetwarzania otrzymanego zapytania. Dla każdego z przedstawionych poniżej testów uprzednio wdrażana była w bazach danych węzłów odpowiednia polityka prywatności.

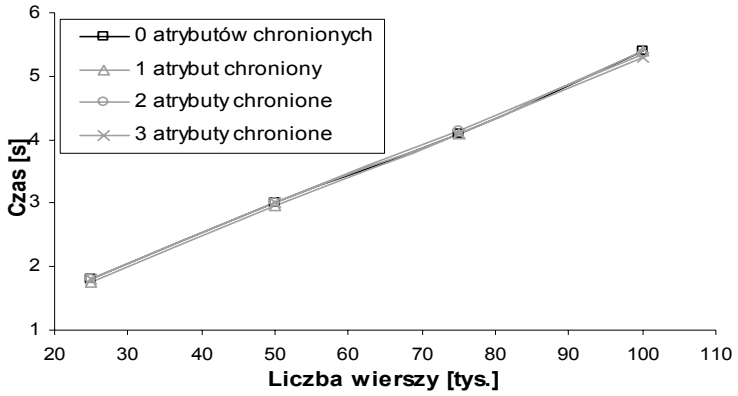


Rys. 5. Pobieranie wartości atrybutów zdekomponowanych
Fig. 5. Receiving values of decomposed attributes

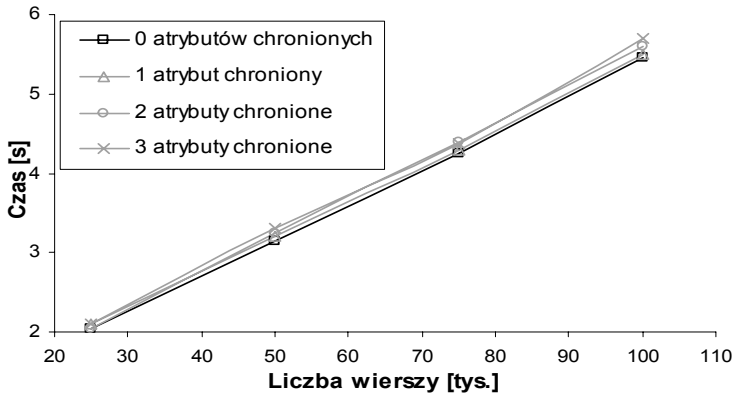


Rys. 6. Pobieranie wartości atrybutów zakodowanych
Fig. 6. Receiving values of coded attributes

Powyższe wykresy (rys. 5 i 6) obrazują wzrost czasu wykonania zapytań w zależności od liczby pobieranych wierszy, a także liczby atrybutów dekomponowanych oraz kodowanych, których wartości są pobierane. Można zauważyć, że większy wpływ na czas wykonania zapytania ma pobieranie wartości atrybutów zakodowanych. Wynika to z dwukrotnie większej liczby danych chronionych pobieranych z węzłów DSDW oraz z konieczności przeprowadzenia operacji dekodowania wartości atrybutów.

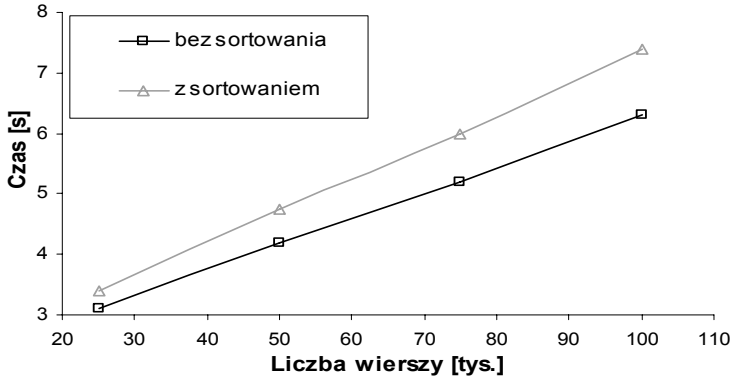


Rys. 7. Realizacja warunków selekcji na atrybutach zdekomponowanych
Fig. 7. Realization of selection conditions for decomposed attributes

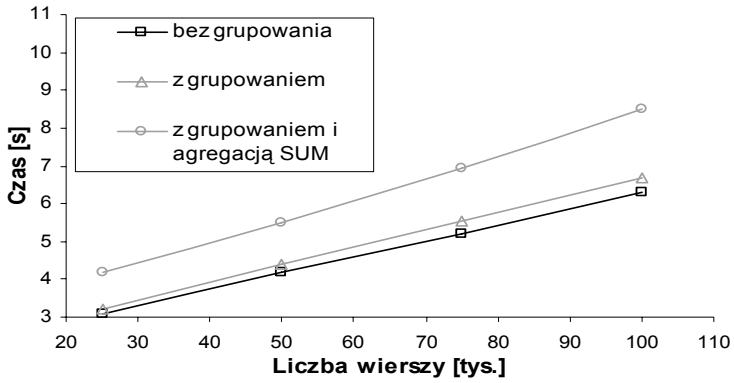


Rys. 8. Realizacja warunków selekcji na atrybutach zakodowanych
Fig. 8. Realization of selection conditions for coded attributes

Wzrost zarówno czasu wykonania zapytań w zależności od liczby pobieranych wierszy, jak również liczby wykorzystywanych w warunkach selekcji atrybutów dekomponowanych oraz zakodowanych jest przedstawiony na rys. 7 i 8. Jak widać w obu przypadkach, wzrost liczby atrybutów chronionych w warunkach selekcji nie prowadzi do zauważalnego wydłużenia czasu wykonania zapytania.



Rys. 9. Realizacja sortowania danych
Fig. 9. Realization of data sorting



Rys. 10. Realizacja grupowania danych
Fig. 10. Realization of data grouping

Na rys. 9 przedstawiono wpływ sortowania na czas wykonania zapytania. Rysunek 10 przedstawia porównanie czasów wykonania trzech rodzajów zapytania: bez grupowania, z grupowaniem oraz z grupowaniem i agregacją, przy czym agregowano wartość jednej z miar tabeli faktów. Jak widać, różnica między czasami przetwarzania zapytania z sortowaniem oraz zapytania bez sortowania zwiększa się średnio tylko o 0,25 s wraz ze wzrostem liczby wierszy. Również grupowanie pociąga za sobą nieznaczny wzrost czasu wykonania zapytania. Dopiero konieczność wykonania agregacji wnosi zauważalny narzut czasowy. W obydwu testach pobierane były wartości dwóch atrybutów zdekomponowanych oraz jednego atrybutu zakodowanego.

7. Wnioski

W pracy [5] przedstawiona została nowa metoda ochrony prywatności danych z przetwarzaniem zapytania, realizowanym przez pojedynczy moduł ochrony prywatności. Bazując na zaproponowanej metodzie ochrony prywatności, przedstawiliśmy nową architekturę systemu rozproszonej przestrzennej hurtowni danych. Każdy z węzłów w tej architekturze posiada własny moduł ochrony prywatności, która ma możliwość komunikacji z innymi modułami oraz z lokalną bazą danych, a w razie konieczności pobrania danych chronionych z innych baz danych. Powyższe założenia umożliwiły równoległe realizowanie procedury wykonywania zapytań przez wszystkie moduły ochrony prywatności. Dało to w efekcie znaczne skrócenie czasu realizacji zapytania w DSDW.

Wykonywanie zapytań na danych chronionych wprowadza dodatkowy narzut czasowy, wynikający z konieczności wykonywania podzapytań odwołujących się do danych chronionych. Przeprowadzone testy wykazały, że zarówno zapytania pobierające wartości atrybutów chronionych, jak i dokonujące selekcji na atrybutach chronionych powodują nieznaczne wydłużenie czasu wykonania. W przeciwieństwie do rozwiązania opisanego w [5], w niniej-

szej pracy narzut czasowy dla obu typów zapytań tylko w niewielkim stopniu zależy od liczby wykorzystywanych w nich atrybutów chronionych.

Aby poszerzyć zakres możliwych do wykonania analiz w DSDW, wdrożona została obsługa zapytań sortujących, grupujących oraz grupujących wraz z agregacją. Dalsze testy potwierdziły niewielki wpływ, jaki na czas wykonania zapytania OLAP pobierającego wartości atrybutów chronionych ma realizacja nowych operacji.

Przyszłe prace skupiać się będą na poszerzaniu funkcjonalności DSDW poprzez możliwość wykonywania szerszego zakresu typów zapytań, a zarazem analiz. Dodatkową drogą rozwoju może być rozszerzenie systemu o funkcje eksploracji danych.

LITERATURA

1. Gedik B., Liu L.: A Customizable k-Anonymity Model for Protecting Location Privacy. 25th International Conference on Distributed Computing Systems (IEEE ICDCS 2005).
2. Kirkgöze R., Katic N., Stolba M., Tjoa A. M.: A Security Concept for OLAP. Proceedings of the 8th. Intenational Workshop on Database and Expert Systems Applications (DEXA'97), IEEE Computer Society, 1997.
3. Rosenthal A., Sciore E.: View Security as Basis for Data Warehouse Security. International Workshop on Design and Management of Data Warehouse (DMDW'2000), Sweden, June 2000.
4. Weippl E., Mangisengi O., Essmayr W., Lichtenberger F., Winiwarter W.: An Autorization Model for Data Warehouses and OLAP. Software Competence Center Hagenberg, Austria 2001.
5. Gorawski M., Bularz J.: Protecting Private Information by Data Separation in Distributed Spatial Data Warehouse. International Workshop "Dependability Aspects on Data Warehousing and Mining applications" DAWAM 2007 *in conjunction with* ARES 2007, Austria, Wiedeń, 2007, s. 837÷843.
6. Gorawski M., Malczok R.: Materialized aR-tree in Distributed Spatial Data Warehouse. International Journal Intelligent Data Analysis, (IDA), ISSN: 1088-467X, IOS Press, 2006, Vol.10, nr. 4, s. 361÷377.
7. Gorawski M., Stachurski K.: On Association Rules Mining Algorithms with Data Privacy Preserving. Advances in Web Intelligence Third International Atlantic Web Intelligence Conference, AWIC05, LNAI3528 Springer, ISBN3-540-26219-9, 2005, s. 170÷175.

8. Gorawski M., Słabiński Ł.: Implementation of Homomorphic Encryption in Privacy-Preserving. 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD06), Tenth East-European Conference on Advances in Databases and Information Systems, ADBIS06, 2006, s. 37÷47.

Recenzent: Dr hab. inż. Krzysztof Goczyła, prof. Pol. Gdańskiej

Wpłynęło do Redakcji 17 sierpnia 2007 r.

Abstract

Dramatically increasing amount of data stored in a database systems and the facts that the data are strategic for enterprises and organizations which possess them and are desired source of information for competitive companies enforces more restrictive requirements for privacy policies. We implemented the relation decomposition as a method of preserving data confidentiality in a new distributed spatial data warehouse architecture that we proposed. Separating data between nodes of distributed system allows protecting data privacy and eliminates the need of sensitive data encryption usage. Each node in a new architecture has its own privacy preserving module which can communicate with other modules. The solution enables parallel query processing through all privacy preserving modules and as a result it shortens the time of query execution in DSDW. We also discussed the way of processing data. Tests of our data privacy preserving solution are also presented.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, M.Gorawski@polsl.pl.
Szymon PANFIL: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, S.Panfil@polsl.pl