STUDIA INFORMATICA 2010
Volume 31 Number 2A (89)

Łukasz MOSIEJ
Politechnika Warszawska, Wydział Elektroniki
i Technik Informacyjnych, Instytut Informatyki

WYKRYWANIE NIESPÓJNOŚCI DANYCH W ROZPROSZONYCH SYSTEMACH TRANSAKCYJNYCH Z WYKORZYSTANIEM REGUŁ ASOCJACYJNYCH (ALGORYTM A PRIORI)

Streszczenie: Ostatnie 15 lat było okresem popularyzacji rozproszonych systemów transakcyjnych w wielu firmach. Stosowanie rozproszonych systemów transakcyjnych poza zaletami ma jedną dużą wadę związaną stricte z redundancją danych: podczas wykonywania modyfikacji danych mogą powstać niespójności danych. Niniejszy artykuł przedstawia nową metodę wykrywania niespójności danych oraz porównuje ją do obecnie stosowanych metod.

Słowa kluczowe: niespójności danych, systemy rozproszone, transakcyjność, reguły asocjacyjne, algorytm A priori

DETECTION DATA INCONSISTENCY IN DISTRIBUTED SYSTEMS USING ASSOCIATION RULES (ALGORITHM A PRIORI)

Summary: Last 15 years was a period of popularization of distributed transactional systems in many companies. The use of distributed transactional systems beyond many advantages has one disadvantage, which is connecting with redundancy of data: during modification data, data inconsistency may arise. This article describes new detecting data inconsistency method and compares it with others available methods.

Keywords: data inconsistency, distributed systems, transaction, association rules, algorithm A priori

1. Wstęp

1.1. Rozproszone systemy transakcyjne

Rozproszony system transakcyjny jest to system komputerowy oparty na systemie współpracujących ze sobą podsystemów [2]. Z uwagi na częste oddalenie geograficzne podsystemy połączone są ze sobą za pomocą sieci komputerowej i wykorzystują do komunikacji dedykowany protokół komunikacyjny. Podsystemy scalane są za pomocą dedykowanego systemu zwanego Integratorem[1]. Każdy z podsystemów udostępnia interfejsy, za pomocą których Integrator wykonuje odpowiednie akcje, takie jak: dodawanie, modyfikacja, usuwanie danych. Nie każdy podsystem wykonuje operacje w sposób transakcyjny.

Każdy z podsystemów trzyma tylko interesujące go dane w odpowiedniej formie (np. system A trzyma kwotę do zapłaty w postaci kwoty brutto, a system B trzyma tę samą daną w postaci dwóch danych: kwoty netto i kwoty VAT). Podsystemy systemów rozproszonych z uwagi na redundancje danych dzielą się na dwa typy systemów:

1. Master – oznacza, iż dany system jest masterem danych, jakie przechowuje. W przypadku modyfikacji danych przez Integratora pierwszym systemem, w których dane zostają zmodyfikowane, jest system master.
2. Slave – oznacza, iż dany system nie jest masterem danych, jakie przechowuje. W przypadku stwierdzenia niespójności danych dane z systemu slave są nadpisywane przez odpowiednie dane z systemu master.

1.2. Cel stosowania rozproszonych systemów transakcyjnych

Ostatnie 15 lat było okresem dużego rozwoju firm z dziedziny telekomunikacji i informatyki. Z uwagi na bardzo szybki rozwój firmy IT musiały wprowadzać nowe systemy do obsługi coraz większej liczby klientów. Stosowanie jednego, scentralizowanego systemu, trzymającego wszystkie dane, z uwagi na duży wolumen danych było niemożliwe. Z drugiej strony na potrzeby odpowiednich akcji biznesowych powstawały dedykowane systemy (podsystemy systemu rozproszonego), które z uwagi na oferowane funkcjonalności nie miały potrzeby trzymania wszystkich danych. W sieciach telekomunikacyjnych powstały następujące podsystemy:

1. CRM2 – podsystem do zarządzania relacjami z klientem.
2. Billing – podsystem do zarządzania rozliczeniami między operatorem a klientami.
3. Sieć – podsystem do zarządzania katalogiem usług, jaki udostępnia operator klientom.

[1] Integrator – system odpowiadający za wykonanie zmiany biznesowej na systemach dziedzinowych.
[2] CRM – (ang. Customer Relationship Management) system do zarządzania relacjami z klientem.

Dzięki tym podsystemom odpowiednie działy w firmie telekomunikacyjnej mogły funkcjonować. W przypadku zmian danych klienta (np. zmiana planu taryfowego) dane z jednego systemu były propagowane za pośrednictwem Integratora do innych podsystemów.

Następnym celem stosowania rozproszonych systemów transakcyjnych jest zwiększenie dostępności danych w przypadku niedostępności jednego z podsystemów. W takich przypadkach z uwagi na redundancję danych dane można pobrać z innych dostępnych podsystemów.

1.3. Główne problemy architektoniczne

Z uwagi na brak możliwości zapewnienia transakcyjności we wszystkich podsystemach istnieje duże ryzyko powstania niespójności danych [2]. Często z uwagi na architekturę niektórych podsystemów (m.in. systemów sieciowych) należy stosować komunikację asynchroniczną[3], co skutkuje brakiem pewności, iż zmiany zostały wykonane na wszystkich podsystemach. Często w przypadku niedostępności jednego systemu Integrator próbuje wykonać wycofanie zmian, jakie wykonał na innych systemach. Nie zawsze taka akcja kończy się pełnym sukcesem, co skutkuje powstaniem niespójności danych.

1.4. Cel wykrywania niespójności danych w rozproszonych systemach transakcyjnych

Niska jakość danych klienta kosztuje wszystkie firmy zlokalizowane w USA w przybliżeniu około 600 miliardów dolarów amerykańskich rocznie.

Główne cele wykrywania niespójności danych w rozproszonych systemach transakcyjnych są następujące [4]:

1. Ograniczenie oszustw, wyłudzeń.
2. Ograniczenie liczby reklamacji.
3. Podniesienie poziomu świadczenia usług.
4. Zwiększenie konkurencyjności na rynku.
5. Zwiększenie zadowolenia klientów.
6. Zwiększenie wpływów finansowych dla firmy.

1.5. Metody wykrywania niespójności danych w rozproszonych systemach transakcyjnych

Obecnie na rynku dostępnych jest wiele mechanizmów do wykrywania niespójności danych w rozproszonych systemach transakcyjnych. Do najbardziej popularnych zaliczyć

[3] Komunikacja asynchroniczna – rodzaj komunikacji, w której system wysyłający żądanie nie czeka synchronicznie na odpowiedź z systemu dziedzinowego. W takiej komunikacji system dziedzinowy nie ma ustalonych limitów czasowych na odpowiedź.

można cykliczną synchronizację systemów slave z systemami master [3]. Metoda polega na pobraniu ze wszystkich systemów danych, umieszczeniu ich w jednej lokalizacji i sprawdzeniu poprawności na poziomie danych prostych. W przypadku stwierdzenia niespójności danych informacja zostaje zapisana w końcowym raporcie z błędami. Jest to najbardziej popularna metoda do wykrywania niespójności danych.

Inne metody do wykrywania niespójności danych są następujące [4]:

1. Cykliczna synchronizacja pomiędzy systemami dziedzinowymi (zakłada cykliczną synchronizację danych między systemami przechowującymi podobne dane za pomocą wbudowanych procedur PL/SQL).
2. Synchronizacja danych ad-hoc (zakłada akceptację niespójności danych; niespójności poprawiane ręcznie po wychwyceniu ich przez człowieka).
3. Metoda oparta na robotach cyklicznie przeszukujących bazy danych systemów master i slave, naprawiających niespójności w oparciu o reguły biznesowe i porównywanie danych między tymi systemami.

Powyższe metody są proste w implementacji i częściowo operują na regułach biznesowych[4] [4]. Jednakże mają one wiele wad, które stały się bodźcem do rozpoczęcia badań nad nową metodą do wykrywania niespójności danych. Jedną z głównych wad jest czasochłonność procesu przeszukiwania danych. Czas potrzebny do przeszukania wszystkich danych w jednej z polskich sieci telekomunikacyjnych wynosi obecnie ponad 26 godzin przy wolumenie klientów równym ponad 12 milionów. Drugą poważną wadą powyższych mechanizmów jest operowanie na wszystkich danych, a nie na danych ostatnio modyfikowanych.

2. Nowa metoda do wykrywania niespójności danych, wykorzystująca reguły asocjacyjne (algorytm A priori)

2.1. Główne założenia nowej metody

Jednym z głównych założeń nowej metody jest ograniczenie wolumenu danych jedynie do danych ostatnio modyfikowanych. Założono tutaj, iż w danych nie modyfikowanych od ostatniego sprawdzenie nie ma żadnych niespójności danych. Drugim bardzo ważnym założeniem, na którym bazuje cała koncepcja rozwiązania jest założenie, iż w danych ostatnio modyfikowanych jedynie znikomy procent danych posiada niespójności danych. Dzięki temu założeniu do nowej metody został wykorzystany algorytm A priori. Te dwa założenia mają duży wpływ na szybkość i efektywność działania nowej metody. Skutkami niepożądanymi tych dwóch założeń może być sytuacja, w której nowa metoda nie wykryje wszystkich

[4] Reguła biznesowa – zbiór warunków logicznych, opisujących językiem logiki matematycznej poprawne i niepoprawne cechy danego obiektu biznesowego.

niespójności danych. Zostało to bardziej szczegółowo opisane w następnych sekcjach tego rozdziału.

2.2. Opis metody wykrywania niespójności danych

Zaproponowana metoda wykrywania niespójności danych została podzielona na dwie fazy. Pierwsza faza odpowiada za usunięcie ze zbioru danych ostatnio modyfikowanych danych poprawnych. Druga faza odpowiada za wyłuskanie z danych pozostałych z fazy pierwszej wszystkich niespójności danych.

Poniżej zostały opisane szczegółowo dwie fazy nowej metody.

2.2.1. Faza 1

Pierwszym etapem tej fazy jest pobranie ze wszystkich systemów slave danych modyfikowanych od ostatniego sprawdzenia. Następnie pobranie ze wszystkich systemów master wszystkich danych. Założono tutaj, iż jedynie w danych z systemów slave mogą występować niespójności danych, a dane w systemach master są zawsze poprawne. Założenie to jest konsekwencją założenia z fazy 2, które zakłada, iż w przypadku wykrycia niespójności danych pomiędzy systemem master i slave dane z systemu master traktujemy jako poprawne.

Następnym etapem fazy pierwszej jest przygotowanie danych na wejście do algorytmu A priori. Z uwagi na fakt, iż dane z systemów slave zawierają różne dane (np. dane o klientach, dane o rachunkach klientów, dane o aktywnych usługach), należy rozdzielić dane na podzbiory, zawierające te same biznesowe dane. Następnie dla każdego podzbioru należy stworzyć dane w odpowiedniej formie jako dane wejściowe do algorytmu A priori. Poniżej został opisany w formie krokowej algorytm budowy tych danych dla pojedynczej danej.

Algorytm budowy danych do algorytmu A priori:

1. Pobierz dla danych z systemu slave odpowiednie dane systemu master.
2. Zapisz dane w jednej linii za pomocą ustalonych znaków.
3. Powtórz kroki 1 i 2 dla reszty danych z systemu slave, używając tej samej składni.

Z uwagi na fakt, iż algorytm A priori będzie uruchamiany dla całego podzbioru, należy zapewnić stały format danych systemu slave i master.

Poniżej został przedstawiony przykład budowy danych na wejście algorytmu A priori dla następujących danych:

```
 ”
system slave:
format:id klienta:lista aktywności usług oddzielona dwukropkiem (0 – nieaktywna;
1 aktywna)
…
1256:0:1:1:1:0:0:1:0:0:1
1521:0:1:1:1:0:1:0:1:1:1
…

System master:
```

```
format:id klienta:lista danych sieciowych odzwierciedlająca usługi oddzielona
dwukropkiem
1256:5:4:1:1:1:3:2:1:1:1:1:1:2:2:4
1521:4:2:2:2:1:3:2:1:1:1:1:1:3:2:4

Budowa danych dla klienta 1256 i 1521:
1256 0:1:1:1:0:0:1:0:0:1:5:4:1:1:1:3:2:1:1:1:1:1:2:2:4
1521 0:1:1:1:0:1:0:1:1:1:4:2:2:2:1:3:2:1:1:1:1:1:3:2:4
"
```

Następnym etapem pierwszej fazy jest wyodrębnienie wszystkich zbiorów częstych występujących w każdym podzbiorze przygotowanym w poprzednim etapie za pomocą algorytmu A priori. Wynikiem działania algorytmu A priori jest lista reguł asocjacyjnych, które będą opisywały wszystkie zbiory częste.

Algorytm A priori jest procesem dwufazowym. W pierwszej fazie znajdywane są wszystkie zbiory częste. Natomiast druga faza polega na utworzeniu na podstawie zbiorów częstych reguł asocjacyjnych, które spełniają warunek minimalnego wsparcia i ufności.

Wsparcie dla danej reguły asocjacyjnej A => B (jeśli A to B) jest wartością procentową transakcji w całym zbiorze, które zawierają A i B. Natomiast ufność dla danej reguły asocjacyjnej A => B (jeśli A to B) jest miarą dokładności reguły, określoną jako procent transakcji zawierających A, które również zawierają B[1]. Obie wartości opisane są następującymi wzorami:

$$wsparcie(A => B) = P(A \cap B) = \frac{liczba\_transakcji\_zaw\_A\_i\_B}{całałkowita\_liczba\_transakcji} \tag{1}$$

$$ufnosc(A => B) = P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{liczba\_transakcji\_zaw\_A\_i\_B}{liczba\_transakcji\_zaw\_A} \tag{2}$$

W celu wyznaczenia reguł asocjacyjnych, które opisują dane poprawne, należy ustalić odpowiednie minimalne wartości dla powyższych dwóch współczynników. Standardowo wsparcie (1) powinno być większe niż 20%, natomiast minimalny poziom ufności (2) powinien wynosić 70% [1]. Powyższe wartości wsparcia i poziomu ufności są brane jako wartości początkowe do etapu testów gotowego rozwiązania. W zależności od jakości danych wartości wsparcia i poziomu ufności należy modyfikować w taki sposób, aby możliwie wszystkie niespójne dane nie dostały się do żadnego zbioru częstego. Reguły asocjacyjne powstają w wyniku działania programu napisanego w języku R[5]. Program na podstawie danych przygotowanych w pierwszym etapie fazy pierwszej zwraca jako wynik działania listę reguł asocjacyjnych spełniających minimalne wsparcie i poziom ufności.

Poniżej został przedstawiony fragment kodu w języku R, który służy do wyznaczania reguł asocjacyjnych z minimalnym poziomem wsparcia 20% i minimalną ufnością 70%:

[5] Język R – język programowania, wykorzystywany do elementów statysyki (http://www.r-project.org/)

```
data = read.table("data.txt", sep = ",", header = TRUE);
library(arules);
trans <- as(data, "transactions");
rules <- priori(trans, parameter = list(supp = 0.2, conf = 0.7, target =
"rules"));
summary(rules);
inspect(rules);
```

Na podstawie reguł asocjacyjnych możliwe jest wydzielenie z całego zbioru krotek, występujących częściej niż inne. Zgodnie z głównym założeniem danych zawierających błędne dane jest stosunkowo mało w stosunku do danych poprawnych, dlatego dane, które spełniają powstałe reguły asocjacyjne, są danymi poprawnymi. Z tego powodu nie przechodzą do drugiej fazy algorytmu.

2.2.2. Faza 2

Na wejście do fazy 2 zostaje przekazany podzbiór danych modyfikowanych od ostatniego sprawdzenia, który nie zawiera danych częstych. Część z tych danych zawiera niespójności danych.

Druga faza opiera swoje działanie na regułach biznesowych, opisujących poprawne obiekty biznesowe. Dzięki tym regułom można w prosty i szybki sposób znaleźć wszystkie niespójności danych.

Reguła biznesowa jest to zbiór warunków logicznych, opisujących językiem logiki matematycznej poprawne i niepoprawne cechy danego obiektu biznesowego. Reguły biznesowe opisują poprawne obiekty za pomocą wyrażeń następujących typów:

1. Jeśli A, to B (np. Jeśli usługa o id 1 jest włączona to plan taryfowy mix).
2. Jeśli A, to nie B (np. Jeśli usługa o id 1 jest włączona to nie plan taryfowy mix).

Gdzie A i B są dowolnymi stwierdzeniami.

Reguły biznesowe tworzone są w sposób manualny przez analityków na podstawie wiedzy biznesowej. Reguły biznesowe tworzone są osobno dla każdego podzbioru danych, gdyż każdy podzbiór danych zawiera inne biznesowo dane. Poniżej przedstawiono przykład dwóch reguł biznesowych dla podzbioru, zawierającego dane o usługach:

Treść reguły:

Jeśli usługa 1 w systemie A jest aktywna, to statusy usług 5 i 6 w systemie B są odpowiednio równe: 4 i 6.

Zapis warunkowy reguły:

u[system=A][id=1][isActive]=active=>u[system=B][id=5][state]=4andu[system=B][id=6][state] = 5

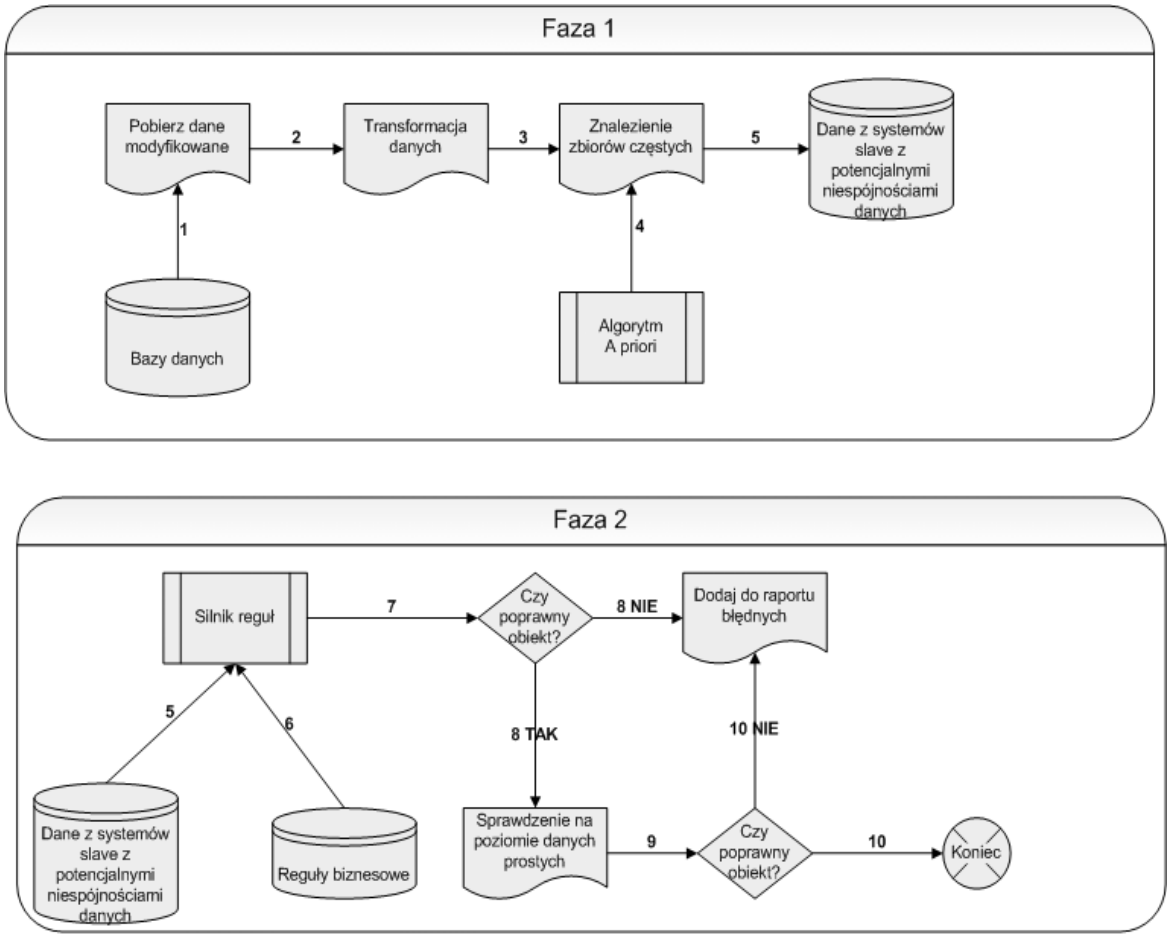
Opis biznesowy reguły:

Jeśli w systemie master A usługa o identyfikatorze id = 1 jest aktywna, to w systemie slave B usługa o identyfikatorze id = 5 ma stan 4, a usługa o identyfikatorze id = 6 ma stan 5.

Za pomocą zbioru reguł biznesowych uruchomionych na silniku reguł[6] ze wszystkich danych z systemu slave, które przeszły do drugiej fazy, zostaje wyłuskana część danych, które zawierają niespójności danych z systemem master [3].

W ostatnim etapie fazy drugiej następuje sprawdzenie reszty danych na poziomie danych prostych. Do tego etapu zostaną zakwalifikowane dane, które nie zawierają się w zbiorach częstych (pierwsza faza) oraz nie zostały odrzucone jako dane poprawne w pierwszym etapie drugiej fazy. Wszystkie niepoprawne dane zapisywane są w raporcie z błędnymi danymi wraz z informacją o poprawnej wartości.

2.2.3. Schemat blokowy



Rys. 1. Schemat blokowy
Fig. 1. Flowchart

Poniżej został przedstawiony schemat blokowy działania zaproponowanej metody wykrywania niespójności danych. Został on podzielony na dwie fazy, które opisano szczegółowo w poprzednich sekcjach tego artykułu.

Faza 1 składa się z następujących kroków:

[6] Silnik reguł – dedykowany system do wykonywania na obiektach walidacji za pomocą reguł biznesowych. Obecnie popularnym silnikiem reguł jest Drools.

1. Pobranie danych modyfikowanych z systemów slave oraz wszystkich danych z systemów master.
2. Podzielenie danych z systemów slave na podzbiory zgodne biznesowo.
3. Wykonanie transformacji danych na wejście algorytmu A priori.
4. Za pomocą aplikacji, wykorzystującej algorytm A priori, wyliczenie reguł asocjacyjnych dla zbiorów częstych.
5. Za pomocą reguł asocjacyjnych podzielenie podzbiorów danych z systemów slave na dane poprawne i dane potencjalnie zawierające niespójności.

Faza 2 składa się z następujących kroków:

1. Za pomocą silnika reguł na podstawie reguł biznesowych sprawdzenie, czy obiekty biznesowe powstałe z danych prostych z systemów slave są poprawne pod względem biznesowym.
2. Zapisanie danych niepoprawnych w raporcie z danymi błędnymi i usunięcie ich ze zbioru danych.
3. Pozostawienie danych poprawnych w zbiorze danych.
4. Po wykonaniu punktów 1 i 2 dla każdego obiektu biznesowego wykonanie na pozostałych danych sprawdzenia spójności danych na poziomie danych prostych.
5. Zapisanie danych niepoprawnych w raporcie z danymi błędnymi i usunięcie ich ze zbioru danych.
6. Raport końcowy zawiera informacje o danych z systemów slave, które posiadają niespójności danych wraz z informacją, jaka powinna być poprawna wartość.

2.2.4. Zalety i wady nowej metody

Zaproponowana metoda wykrywania niespójności danych w rozproszonych systemach transakcyjnych posiada wiele zalet i wad. Jedna z największych wad wynika wprost z głównego założenia nowej metody, iż dane błędne nigdy nie znajdą się w zbiorach częstych (patrz rozdział 2.1). Jeśli taka sytuacja wystąpi, to nie zostaną wykryte wszystkie niespójności danych, gdyż część danych niepoprawnych zostanie zakwalifikowana w pierwszej fazie do danych poprawnych.

Poniżej wypunktowano wszystkie wady i zalety nowej metody wykrywania niespójności danych w rozproszonych systemach transakcyjnych.

Zalety:

1. Ograniczenie wolumenu danych z systemów slave do szukania niespójności danych do danych zmodyfikowanych od ostatniego sprawdzenia.
2. Szybkie odfiltrowanie z danych z systemów slave większej części danych poprawnych.

3. Zmniejszenie czasu do wykrywania niespójności danych poprzez operowanie na obiektach biznesowych z wykorzystaniem reguł biznesowych, opisujących poprawne zachowanie oraz cechy obiektów.
4. Wykonanie sprawdzenia danych pomiędzy systemami slave a master na poziomie danych prostych tylko dla danych pozostałych w ostatnim etapie drugiej fazy metody.

Wady:

1. Istnieje ryzyko odfiltrowania w pierwszej fazie nie tylko danych poprawnych, ale i danych zawierających niespójności danych. Taka sytuacja będzie miała miejsce, w przypadku, gdy dane niepoprawne zostaną zakwalifikowane do zbioru częstego przez algorytm A priori (zostało to szczegółowo opisane w rozdziale 2.1).
2. Jeśli w danych ostatnio modyfikowanych większość danych będzie błędna, to pierwsza faza nowej metody nie odfiltruje danych poprawnych, gdyż nie wejdą do zbiorów częstych. W takim przypadku faza 1 algorytmu nie tylko nie przyspieszy całego procesu, ale go spowolni.

3. Zakończenie

Zaproponowana nowa metoda wykrywania niespójności danych w rozproszonych systemach transakcyjnych jest metodą alternatywną do tradycyjnych metod przeszukiwania danych w celu poprawienia niespójności danych. Nowa metoda zakłada, iż w danych ostatnio modyfikowanych jedynie niewielki procent danych zawiera niespójności danych. Jest to założenie poprawne dla stabilnych rozproszonych systemów transakcyjnych, w których nie tworzy się duży wolumen niespójności danych. Drugim bardzo ważnym założeniem nowej metody jest operowanie nie na danych prostych, ale na obiektach biznesowych. Sprawdzanie poprawności danych odbywa się za pomocą reguł biznesowych uruchamianych na dedykowanym silniku reguł. Jedynie na danych, które przeszły wszystkie walidacje reguł biznesowych, następuje sprawdzenie na poziomie danych prostych.

Zaproponowana metoda może być wykorzystana w sieciach teleinformatycznych do wykrywania niespójności danych. Sieci teleinformatyczne charakteryzują się dużym wolumenem danych oraz stosunkowo dużą zmiennością danych. Wielu klientów w takich sieciach często zmienia swoje usługi, profil, taryfę w zależności od swoich potrzeb i aktualnych promocji. Niniejszy artykuł przedstawia koncepcję rozwiązania nowej metody do wykrywania niespójności danych. Z uwagi na brak gotowego rozwiązania, implementującego nową metod e, nie zostały przedstawione jakiekolwiek testy oraz opis funkcjonalny działającego produktu. Obecnie trwają intensywne prace analityczno-projektowe nad docelowym rozwiązaniem.

BIBLIOGRAFIA

1. Larose D.: Odkrywanie wiedzy z danych. Wprowadzenie do eksploracji danych. Wydawnictwo naukowe PWN, Vol. 1, 2006, s. 185÷207.
2. Tamer Özsu M., Valduriez P.: Distributed DBMS. Vol. 1, 1998, s. 3÷14.
3. Kamra A., Bertino E.: Responding to Anomalous Database Requests. Springer Berlin/ Heidelberg, 2008, s. 50÷66.
4. Mosiej Ł.: Detecting data anomalies methods in distributed systems. Proceedings of SPIE, the International Society for Optical Engineering, Vol. 7502, 2009.

Recenzent: Dr inż. Marcin Gorawski

Wpłynęło do Redakcji 19 stycznia 2010 r.

Abstract

A distributed system is a computer system based on cooperating subsystems. All subsystems can be divided into two groups: slave and master systems. The master system holds the critical data for objects. When the incoherent data for the object is found in slave system, it can be repaired by retrieving the correct data from master system [4]. Nowadays there is many ways to find data inconsistency in distributed systems. The most popular method is based on cyclic synchronization. This method is easy to implement, but it is time consuming.

This article is about new detection data inconsistency method. The main foundation is that in whole modify data there is not to much wrong data. This method consists with two phase and it is based on association rules, which is one of data mining methods. In that method algorithm A priori is used to filter from whole data good data. The support and the confidence should be enough high to get all frequent sets. The next step of this method is to build object from simple data and try to find wrong object using business rules. The business rule is a set of logical conditions which describe correct and incorrect features of business objects.

This method has many advantages. The main advantage is to reduce the volume of checking data. This method operates only on modified data from slave systems. The second advantage is to filter in first phase from whole data good data. The main disadvantage is the

possibility that in some situation new method will not find all data inconsistency. It will happen, when wrong data will be in one of frequent sets.

Adres

Łukasz MOSIEJ: Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, Polska, lukasz.mosiej@gmail.com .