STUDIA INFORMATICA 2010
Volume 31 Number 2A (89)

Marek MIŁEK, Bożena MAŁYSIAK-MROZEK, Dariusz MROZEK
Politechnika Śląska, Instytut Informatyki

ARCHITEKTURA SYSTEMU I PRAKTYCZNE ASPEKTY IMPLEMENTACJI HURTOWNI DANYCH ROZMYTYCH

Streszczenie. Wprowadzenie rozmytości do systemów hurtowni danych pozwala na przetwarzanie danych na wyższym poziomie abstrakcji i wprowadza możliwość analizy danych o nieprecyzyjnym charakterze. Ponadto, umożliwia wyrażanie różnych wskaźników biznesowych w języku naturalnym przy użyciu ogólnych sformułowań typu: *dużo*, *mało*, *około 10*, *prawie wszyscy* i in., reprezentowanych przez odpowiednie funkcje przynależności. Implementacja hurtowni danych rozmytych i narzędzi analizy danych wykorzystujących elementy logiki rozmytej napotyka na szereg problemów natury technicznej, występujących po stronie istniejących systemów zarządzania bazą danych. W niniejszym artykule, na przykładzie systemu zrealizowanego przez autorów, zaprezentowano architekturę i praktyczne aspekty implementacji hurtowni danych rozmytych.

Słowa kluczowe: hurtownie danych, zbiory rozmyte, systemy wspomagania decyzji

SYSTEM ARCHITECTURE AND PRACTICAL IMPLEMENTATION OF FUZZY DATA WAREHOUSE

Summary. Incorporation of fuzziness into data warehouse systems gives the opportunity to process data at higher level of abstraction and improves the analysis of imprecise data. It also gives the possibility to express business indicators in natural language using terms, like: *high*, *low*, *about 10*, *almost all*, etc., represented by appropriate membership functions. There are many technical, server-side problems that appear while developing the Fuzzy Data Warehouse with the use of existing database management systems (DBMSs). In the paper, we show architecture and practical aspects of the implementation of the Fuzzy Data Warehouse system based on our own personal experiences.

Keywords: data warehouse, fuzzy sets, fuzzy logic, decision support systems

1. Wprowadzenie

Hurtownia danych rozmytych stanowi repozytorium danych, które przechowuje zarówno dane precyzyjne, jak i dane rozmyte oraz pozwala na klasyczne i rozmyte przetwarzanie zgromadzonych w niej danych [1], [2]. Wprowadzenie rozmytości do systemów hurtowni danych pozwala na przetwarzanie danych na wyższym poziomie abstrakcji i analizę danych o nieprecyzyjnym charakterze. Ponadto, umożliwia wyrażanie różnego rodzaju wskaźników procesowych, społecznych i biznesowych w języku naturalnym, przy użyciu ogólnych sformułowań typu: *dużo*, *mało*, *około 10*, *prawie wszyscy* i in., reprezentowanych przez odpowiednie funkcje przynależności [3], [4].

W procesie implementacji hurtowni danych rozmytych i narzędzi analizy danych wykorzystujących elementy logiki rozmytej (narzędzi klasy FOLAP [1]) napotyka się na szereg problemów natury technicznej, występujących po stronie istniejących systemów zarządzania bazą danych (SZBD). Rozważając wprowadzenie rozmytości do systemów hurtowni danych, problemy te są dwojakiego rodzaju.

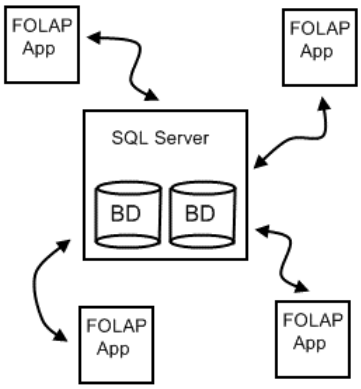
Pierwszym z nich jest brak spójnego sposobu przechowywania liczb rozmytych w bazach danych i odpowiedniego typu danych, który reprezentowałby tego rodzaju dane w bazie hurtowni. Pociąga to za sobą szereg konsekwencji. Jeśli wyobrazimy sobie, że posiadamy rozmyte atrybuty w wymiarach analitycznych hurtowni danych, to musimy zastosować grupowanie rozmytych danych podczas procesu agregacji danych. System hurtowni danych rozmytych powinien również umożliwiać filtrowanie z wykorzystaniem rozmytych atrybutów, co jest związane z wycinaniem fragmentów wielowymiarowej kostki danych (ang. *slicing and dicing*). Ponadto, posiadając w hurtowni tzw. rozmyte miary (miary zdefiniowane na kolumnach przechowujących liczbowe dane rozmyte), musimy zaimplementować arytmetykę liczb rozmytych [5] w celu agregacji danych.

Drugim problemem jest brak możliwości rozmytego przetwarzania danych dokładnych przechowywanych w hurtowni danych. Takie przetwarzanie musiałoby obejmować rozmyte grupowanie danych precyzyjnych oraz rozmyte filtrowanie wierszy i grup. Pozwoliłoby to na skoncentrowanie w tych samych grupach podobnych danych, np. ludzi w podobnym przedziale wiekowym oraz obliczenie dla nich wartości wybranych wskaźników.

W niniejszym artykule, na przykładzie systemu zrealizowanego przez autorów, zaprezentowano architekturę i praktyczne elementy implementacji hurtowni danych rozmytych. Omówiono również budowę i zasady działania aplikacji analitycznej FDW Browser, należącej do klasy Fuzzy-OLAP (FOLAP, od ang. *Fuzzy On-Line Analytical Processing*), współpracującej z hurtownią danych rozmytych.

Hurtownia danych rozmytych i aplikacja analityczna FDW Browser tworzą razem system informacyjny, który stanowi próbę wyznaczenia nowej odmiany systemów analitycznych,

w których możliwe jest przeprowadzanie rozmytego analitycznego przetwarzania danych dokładnych i rozmytych. Opracowany przez autorów system należy do systemów o cechach architektury *ROLAP* [6], [7], w której bezpośrednim źródłem informacji jest starannie zaprojektowana hurtownia danych rozmytych, pracująca w relacyjnym systemie zarządzania bazą danych. System zaprojektowano w architekturze klient-serwer (rys. 1), w której dopuszczalna jest jednoczesna praca wielu instancji aplikacji analitycznych korzystających z zasobów serwera pracującego w tej samej sieci komputerowej. Poszczególne elementy systemu zostały opisane w dalszej części pracy.



Rys. 1. Współpraca aplikacji analitycznej FOLAP z hurtownią danych.
Fig. 1. Interoperability of the FOLAP application and data warehouse.

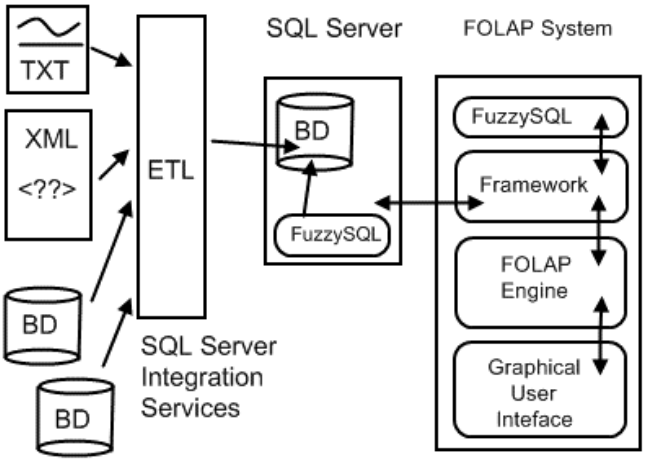
2. Architektura systemu analitycznego współpracującego z hurtownią danych rozmytych

Architektura aplikacji analitycznej należącej do klasy *FOLAP* jest podzielona na kilka modułów, które wszystkie razem tworzą pewną hierarchię zależności (rys. 2). Fundamentalne miejsce zajmuje moduł rozszerzający możliwości języka SQL (*FuzzySQL*). Moduł ten jest instalowany na serwerze bazy danych i umożliwia przetwarzanie danych z wykorzystaniem logiki rozmytej.

Aplikacja analityczna FOLAP również korzysta z tego modułu, między innymi po to, aby dane rozmyte były jednakowo reprezentowane zarówno w środowisku aplikacji analitycznej, jak i w obszarze serwera bazy danych.

Wszystkie mechanizmy niskopoziomowe aplikacji, na których oparty jest system, zebrano w jednym module *Framework*, stanowiącym szkielet dla wyższych warstw aplikacji. Moduł ten zawiera między innymi funkcję dynamicznego generowania zapytań *SQL* do bazy danych. Wyżej w hierarchii znajduje się moduł wielowymiarowego przetwarzania danych FOLAP (*FOLAP Engine*), którego rolą jest umożliwienie zadawania zapytań analitycznych. Na samym szczycie znajduje się moduł interfejsu użytkownika aplikacji, który skupia głów-

nie mechanizmy związane z oknami aplikacji oraz korzysta z elementów zdefiniowanych na niższych szczeblach hierarchii systemu. Hierarchię warstw systemu przedstawiono na rysunku 2, na którym pokazano również, w jaki sposób dane mogą być wprowadzane do systemu. Opis cech funkcjonalnych aplikacji analitycznej FDW Browser został natomiast przedstawiony w [1].



Rys. 2. Wielowarstwowa architektura aplikacji klasy FOLAP.
Fig. 2. Multitier architecture of the FOLAP application.

2.1. Moduł rozszerzenia języka SQL (FuzzySQL)

Rozszerzenie języka SQL stanowi zbiór elementów dostępnych w zakresie bazy danych, wśród których zaimplementowano:

- Typ liczby rozmytej LR umożliwiający przechowywanie danych rozmytych.
- Funkcje logiczne i arytmetyczne na liczbach rozmytych, umożliwiające porównanie liczb rozmytych oraz ich dodawanie, odejmowanie i dzielenie.
- Procedury grupowania rozmytego oraz lingwistycznego liczb dokładnych oraz grupowania liczb rozmytych [8], [9].
- Funkcje agregujące dla danych rozmytych, m.in. FSUM, FAVG, FMIN, FMAX.

2.2. Szkielet aplikacji analitycznej FDW Browser (moduł Framework)

Moduł *Framework* stanowi fundament aplikacji analitycznej. Znajduje się najniżej w hierarchii, jeśli nie brać pod uwagę modułu rozszerzenia *SQL*, który głównie dedykowany jest serwerowi bazy danych. Modułu *Framework* umożliwia:

- Dostęp do schematu bazy danych
- Tworzenie logicznej reprezentacji zapytania
- Dynamiczne generowanie zapytania SQL

- Zarządzanie zmiennymi lingwistycznymi
- Konwersję do liczb rozmytych

2.2.1. Dostęp do schematu bazy danych

Ważnym elementem systemu jest moduł pośredniczący w dostępie do schematu bazy danych wraz z możliwością odczytu zależności pomiędzy tabelami. Dostęp do listy tabel jest ważny w momencie tworzenia widoków źródła danych, natomiast dzięki zależnościom pomiędzy tabelami możliwa staje się automatyczna identyfikacja tabel wymiarów i faktów w kreatorze definiowania modelu wielowymiarowej kostki danych.

2.2.2. Tworzenie logicznej reprezentacji zapytania

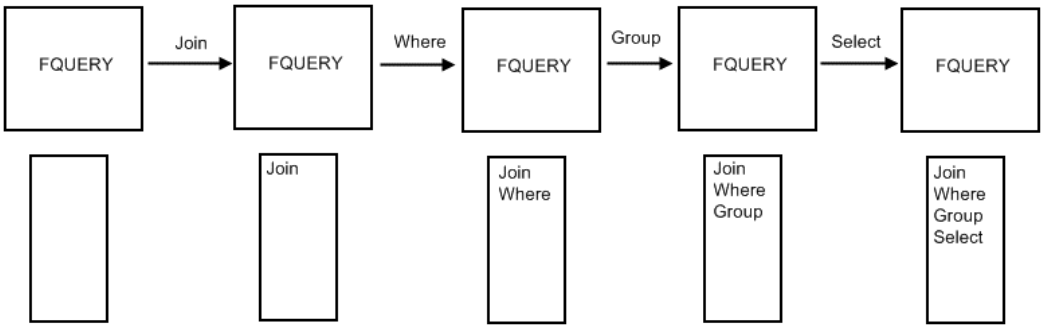
Mechanizm ten ma na celu umożliwienie konstrukcji zapytania bez potrzeby jego zapisywania w postaci kodu SQL. Na samym początku tworzony jest element reprezentujący zapytanie **FQuery** i dotyczy on zawsze jednej wybranej tabeli. Za pomocą odpowiednich przekształceń za pomocą wywołań określonych funkcji możliwe jest wykonywanie następujących operacji (w nawiasach podano nazwy tych funkcji):

- Wybór kolumn (**Select**)
- Złączenie z inną tabelą (**Join**)
- Grupowanie względem wybranej kolumny (**GroupBy**)
- Grupowanie rozmyte względem wybranej kolumny (**FuzzyGroupBy**)
- Określenie warunków filtrujących (**Where**)
- Porządkowanie elementów względem wybranej kolumny (**OrderBy**)

Obiekt reprezentujący zapytanie gromadzi wszystkie wykonane na nim przekształcenia w postaci odpowiednich wyrażeń (**QueryExpression**). Wyrażenia te są obiektami klas pochodnych od **QueryExpression** i w momencie realizacji zapytania stanowią elementy identyfikujące poszczególne fragmenty zapytania.

Na rysunku 3 zaprezentowano zasadę działania mechanizmu tworzenia zapytania. Każde wywołanie funkcji buduje nową reprezentację zapytania. Mechanizm nie narzuca konieczności wywoływania wszystkich dostępnych operatorów funkcyjnych, gdyż oczywiste jest, że nie zawsze zachodzi taka potrzeba. Parametrami wywołań funkcji są elementy, które określają, w jaki sposób poszczególne fragmenty wyrażeń mają zostać skonstruowane. W przypadku instrukcji **Select** parametrami wywołania funkcji są kolumny. Wywołanie funkcji **Join** wymaga podania tabeli oraz kolumn, na podstawie których wykonywane jest złączenie tabeli bazowej oraz drugiej tabeli. Pozostałe funkcje w analogiczny sposób wymagają odpowiednich parametrów właściwych dla fragmentu instrukcji SQL, które reprezentują.

Choć nie jest konieczne wywołanie wszystkich tych funkcji, należy wziąć pod uwagę konieczność wywoływania ich w podanej kolejności, aby zachować poprawność konieczną w momencie przetwarzania obiektu na zapytanie skierowane do bazy hurtowni danych.



Rys. 3. Przykład etapowego tworzenia obiektu FQuery reprezentującego zapytanie
Fig. 3. Phases in formation of the FQuery object representing analytical query

2.2.3. Dynamiczne generowanie zapytania SQL

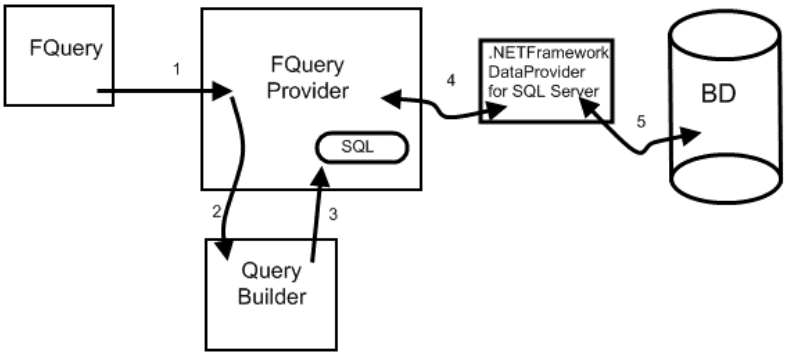
Inicjatorem wykonania zapytania w obszarze egzekutora zapytań bazy danych jest moduł **FQueryProvider**. Rolą tego elementu jest umożliwienie przekształcenia obiektu zapytania **FQuery** na postać zapytania SQL. Procesem zamiany zapytania na reprezentację możliwą do zrealizowania przez interpreter egzekutora zapytań bazy danych zajmuje się moduł **QueryBuilder**.

Na rysunku 4 przedstawiono w kolejnych krokach przebieg wykonania zapytania. Proces ten wykonywany jest w następujących etapach:

1. Przekazanie obiektu reprezentującego zapytanie (**FQuery**) do zrealizowania.
2. Przekazanie zapytania do modułu **QueryBuilder** i translacja zapytania do postaci zapytania w języku SQL.
3. Zwrócenie instrukcji zapytania SQL.
4. Realizacja zapytania z użyciem dostawcy usług komunikacji z serwerem bazy danych.
5. Komunikacja środowiska klienta z serwerem z użyciem właściwego dla tej operacji protokołu.

2.2.4. Zarządzanie zmiennymi lingwistycznymi

W aplikacji FDW Browser istnieje możliwość zarządzania zmiennymi lingwistycznymi oraz ich wartościami. Ze względu na to, że algorytmy przetwarzania rozmytego, wykonywane po stronie serwera bazy danych, w niektórych przypadkach wymagają dostępu do zmiennych lingwistycznych, zmienne te oraz ich wartości przechowywane są w specjalnie przeznaczonych do tego tabelach. Pomocniczy moduł **FuzzyDBHelper**, będący elementem aplikacji analitycznej, spełnia rolę pośrednika pomiędzy bazą danych a interfejsem użytkownika, umożliwiając pełne zarządzanie zgromadzonymi zmiennymi oraz wartościami rozmytymi.



Rys. 4. Etapy wykonania zapytania
Fig. 4. Phases of query execution

2.2.5. Konwersja do liczb rozmytych

Dodatkowym elementem systemu jest moduł **DataConversionHelper** przeprowadzający proces konwersji wartości dokładnych do wartości rozmytych. W ramach tej operacji powinny zostać przetworzone wartości wszystkich wierszy w wybranej kolumnie tabeli. Przez wybór zmiennej lingwistycznej wybierany jest zestaw możliwych wartości, jakie może przyjąć wartość dokładna po konwersji do liczby rozmytej. W przypadku każdej konwertowanej liczby algorytm klasyfikuje ją do odpowiedniej wartości lingwistycznej w taki sposób, aby wartość ta odpowiadała jej w jak największym stopniu zgodności.

W drugim przypadku może istnieć potrzeba konwersji z typu tekstowego do wartości rozmytej. Oprócz podania kolumny źródłowej oraz zmiennej lingwistycznej algorytm zakłada podanie wzorca, za pomocą którego jest realizowane odpowiednie dopasowanie reprezentacji tekstowej do reprezentacji typu rozmytego **LR**.

2.3. Moduł procesora wielowymiarowego FOLAP (FOLAP Engine)

Wielowymiarowy procesor FOLAP posiada wbudowane mechanizmy umożliwiające przetwarzanie zapytań analitycznych, a także inne elementy konieczne do prawidłowej pracy i działania systemu analitycznego. Elementy składające się na procesor FOLAP umożliwiają konstruowanie, reprezentację oraz translację zapytania do postaci, którą dalej będzie można przetworzyć na zapytanie kierowane do serwera bazy danych.

Moduł zawiera elementy, za pomocą których można zdefiniować wielowymiarowy logiczny model kostki danych. Istnieje ponadto mechanizm wspomagający tworzenie takich modeli. Choć moduł ten dedykowany jest głównie funkcjom tworzenia i przetwarzania analitycznych zapytań, to zawiera on również inne funkcje ważne z perspektywy systemu, takie jak zarządzanie sesją aplikacji, czy mechanizmy tworzenia wykresów.

2.3.1. Elementy wielowymiarowego modelu

Elementy wielowymiarowego modelu kostki danych są oparte na obiektach bazy danych takich jak tabele, kolumny, czy funkcje agregujące. Wielowymiarowy model kostki danych definiowany jest poprzez następujące elementy.

Kostka danych (obiekt **Cube**)

- Nazwa kostki
- Fakty (obiekty **Fact**)
  - miary (zbiór obiektów **Measure**)
  - grupy miar (zbiór obiektów **MeasureGroup**)
- Wymiary (zbiór obiektów **Dimension**)
  - nazwa wymiaru
  - atrybuty (zbiór obiektów **Attribute**)
- Hierarchie (zbiór obiektów **Hierarchy**)
  - nazwa hierarchii
  - atrybuty (uporządkowany zbiór obiektów klasy **Attribute**)

2.3.2. Mechanizmy wspomagania tworzenia modelu wielowymiarowego

Mechanizmy te ułatwiają tworzenie struktury kostki danych. Na początku tego procesu powinien być znany podzbiór tabel, z którego ma zostać zbudowany wielowymiarowy model. Odpowiednie mechanizmy pozwalają wstępnie utworzyć strukturę kostki, bazując na zależnościach, które wynikają z powiązań pomiędzy tabelami w bazie danych. Tabela faktów jest zwykle wybierana na podstawie liczby występujących w niej kluczy obcych. Wybór taki wynika z charakteru modelu, jaki przyjmuje zazwyczaj schemat hurtowni danych[1]. Sugerowany przez system wybór może ulec zmianie, zgodnie z wolą użytkownika aplikacji analitycznej.

2.3.3. Silnik przetwarzania analitycznego

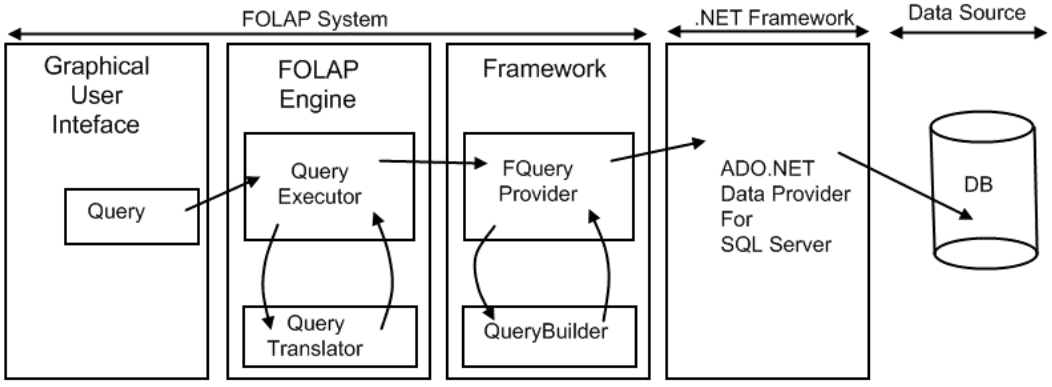
Zapytanie analityczne utworzone w interfejsie graficznym przez użytkownika jest reprezentowane w aplikacji analitycznej FDW Browser jako obiekt **Query**. Obiekt ten przechowuje między innymi wybrane przez użytkownika miary oraz grupy miar. Wyniki analiz i podsumowań są prezentowane dwuwymiarowo, zatem już na tym etapie zbiór atrybutów dzielony jest na dwa podzbiory. Podzbiór pierwszy określa wybór atrybutów, na podstawie których utworzone grupy prezentowane będą w wierszach. Drugi podzbiór w analogiczny sposób określa wybór atrybutów, lecz prezentacja przebiegać będzie w kolumnach.

[1] W schemacie płatka śniegu lub gwiazdy [5] centralne miejsce zajmuje tabela faktów, która zazwyczaj charakteryzuje się tym, że posiada najwięcej powiązań z innymi tabelami.

Podczas wyboru atrybutu automatycznie dołączany jest również wymiar, z którego ten atrybut pochodzi. Podobnie jak w przypadku atrybutów, występuje podział między hierarchiami, które mogą występować zarówno w wierszach, jak i w kolumnach. Ważnym elementem zapytania jest też lista warunków, które powinny spełniać analizowane dane. Oprócz tego istnieje możliwość sortowania danych względem wybranego atrybutu wraz z określeniem porządku sortowania.

Logiczna postać zapytania analitycznego w postaci obiektu **Query** stanowi najwyższą warstwę abstrakcji w całym mechanizmie zapytań. Zanim zapytanie zostanie wysłane do serwera bazy danych, wcześniej zostaje przetłumaczone do obiektowej reprezentacji zapytania SQL reprezentowanej przez obiekt **FQuery**. Obiekt zapytania inicjowany jest na podstawie tabeli faktów. Następnie znając wszystkie tabele wymiarów biorące udział w zapytaniu, można wykonać odpowiednie złączenia za pomocą wywołania funkcji *Join*. Podobnie wszystkie warunki zapytania są dołączane do zapytania za pomocą funkcji *Where*. Atrybuty, względem których następuje grupowanie, dołączane są za pomocą funkcji *GroupBy* lub *FuzzyGroupBy* w zależności od tego, jaki tryb grupowania jest wybrany dla danego atrybutu. Jeśli konieczne jest sortowanie, to odpowiednie parametry określające kolejność wyników dołączane są za pomocą funkcji *OrderBy*. Na samym końcu lista miar oraz atrybutów grupowania wybierana jest za pomocą funkcji *Select*.

Realizacją zapytania analitycznego zajmuje się moduł wykonania zapytań **QueryExecutor**. Moduł ten wykorzystywany jest w momencie, gdy konieczne jest pobranie danych. Zamianą zapytania analitycznego do postaci obiektu **FQuery** zajmuje się translator zapytań **QueryTranslator**. Obiektowa postać zapytania SQL jest następnie przekazywana do modułów odpowiedzialnych za dynamiczne utworzenie, a następnie wykonanie zapytania SQL. Cały proces zilustrowany został na rysunku 5.



Rys. 5. Etapy przetwarzania zapytania analitycznego
Fig. 5. Phases of analytical query processing

2.3.4. Menadżer wykresów

Aplikacja analityczna FDW Browser umożliwia prowadzenie procesu analizy z wykorzystaniem wykresów, co jest niezwykle istotne dla analizy danych w każdej hurtowni danych, w tym z hurtowni danych rozmytych. Elementem odpowiedzialnym za zdefiniowanie obszarów wykresów jest menadżer wykresów **ChartManager**. Moduł ten na podstawie pobranych danych oraz informacji o tym, jakie miary oraz atrybuty biorą udział w analizie, odpowiednio konstruuje wykresy. W procesie tym uwzględnia fakt, że niektóre atrybuty lub miary mają charakter rozmyty. Menadżer wykresów umożliwia wyświetlanie wykresów dwóch typów: słupkowych oraz kołowych. W przypadku wykresu kołowego ważną informacją, jaką użytkownik może wyświetlić, jest udział procentowy danej wartości na tle wszystkich wyników. Przykłady wykresów znajdują się w opisie cech funkcjonalnych aplikacji analitycznej FDW Browser, przedstawionym w [1].

3. Podsumowanie

Budowa systemu hurtowni danych, który umożliwiałby przechowywanie i przetwarzanie danych rozmytych oraz rozmyte przetwarzanie danych dokładnych, nie jest procesem prostym. Od wielu lat trwają prace nad budową podobnych rozwiązań w obszarze tradycyjnych baz danych [10-13], napotykając na podobne problemy, na które napotykają autorzy poszerzając skalę zastosowań do obszaru hurtowni danych. Jednak autorzy wierzą, że takie podejście do przetwarzania i analizy danych ma wiele zalet. Przy tak dużej liczbie danych, jakie gromadzi się w systemach hurtowni danych, wprowadzenie technik rozmytego przetwarzania danych pozwala na bardziej ogólną analizę niż w przypadku klasycznych hurtowni danych [6], [7]. Naturalne również wydaje się zastosowanie typów i technik rozmytych kolekcjonując nieprecyzyjne dane, np. dane z systemów pomiarowych, dzięki czemu wprowadza się do procesu analizy pewną dozę tolerancji. W tym zakresie przedstawiony system hurtowni danych rozmytych i aplikacja analityczna FDW Browser jest rozwiązaniem pionierskim.

BIBLIOGRAFIA

1. Miłek M., Małysiak-Mrozek B., Mrozek D.: Hurtownia danych rozmytych: podstawy teoretyczne i praktyczne aspekty użycia. Studia Informatica. Vol. 24, No. 2A(53), s. 179÷190, Gliwice 2010, (publikacja w bieżącym wydaniu).

2. Małysiak-Mrozek B., Mrozek D., Kozielski S.: Processing of Crisp and Fuzzy Measures in the Fuzzy Data Warehouse for Global Natural Resources. LNAI, Springer, Heidelberg 2010 (w publikacji).
3. Zadeh L.A.: Fuzzy sets. Information and Control. 1965, 8 (3), s. 338÷353.
4. Dubois D., Prade H.: Fundamentals of fuzzy sets. Kluwer Academic Publisher, 2000.
5. Bouchon-Meunier B., Yager R.R., Zadeh L.A.: Fuzzy logic and soft computing. Advances in Fuzzy Systems, Application and Theory Vol. 4, Singapore 1995.
6. Kimball R., Reeves L., Margy R., Thornthwaite W.: The Data Warehouse Lifecycle Toolkit. John Wiley & Sons, 1998.
7. Ponniah P.: Data Warehousing Fundamentals. A Comprehensive Guide for IT Professionals. John Wiley and Sons, 2001.
8. Małysiak-Mrozek B., Mrozek D., Kozielski S.: Data Grouping Process in Extended SQL Language Containing Fuzzy Elements. Advances in Intelligent and Soft Computing Vol. 59, Springer Verlag GmBH, 2009, s. 247÷256.
9. MacQueen J.B.: Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1967, Vol. 1, s. 281÷297.
10. Bosc P., Pivert O.: SQLf: A Relational Database Language for Fuzzy Querying. IEEE Transactions on Fuzzy Systems. 1995, Vol. 3, No 1.
11. Kacprzyk J., Zadrozny S.: SQLf and FQUERY for Access. IFSA World Congress and 20th NAFIPS International Conference. 2001, s. 2464÷2469.
12. Małysiak B.: Fuzzy Values in SQL Queries Submitted to Databases. Studia Informatica. Vol. 24, No. 2A(53), s. 179÷190, Gliwice 2003.
13. Małysiak B., Mrozek D., Kozielski S.: Processing Fuzzy SQL Queries with Flat, Context-Dependent and Multidimensional Membership Functions. Proc. of 4th IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada. ACTA Press, 2005, s. 36÷41.

Recenzent: Prof. dr hab. inż. Jerzy Klamka

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

Fuzzy Data Warehouse (FDW) is a data repository, which contains fuzzy data and allows a fuzzy processing of the data. Incorporating the fuzziness into data warehouse systems gives the opportunity to process data at higher level of abstraction and improves the analysis of imprecise data. It also gives the possibility to express various social and business indicators in natural language using terms, like: *high*, *low*, *about 10*, *almost all*, etc., represented by appropriate membership functions.

There are many technical, server-side problems that appear while developing the Fuzzy Data Warehouse with the use of existing database management systems (DBMSs). The main problem is the lack of the common standard data type for storing fuzzy data in database management systems. This has several consequences that can be divided into two separate groups. The first group covers the fuzzy processing of crisp values that we store in the data warehouse. The processing must cover fuzzy grouping of crisp data, fuzzy filtering of rows and groups. The second group is related to processing of fuzzy values that we can collect in the data warehouse. If we imagine we have fuzzy attributes in dimensions of data warehouse, we have to use grouping by fuzzy attributes while aggregating data. We also need a possibility to filter data according to fuzzy filtering criteria, which is associated with slicing and dicing on dimensions in multidimensional cubes. Moreover, having fuzzy measures (measures defined on columns that store fuzzy data), we must implement the arithmetic of fuzzy numbers in order to aggregate fuzzy data.

In the paper, we show the architecture and practical aspects of the implementation of the Fuzzy Data Warehouse system based on our own personal experiences.

Adresy

Marek MIŁEK: student Politechniki Śląskiej, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, marek.milek@gmail.com .

Bożena MAŁYSIAK-MROZEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, bozena.malysiak@polsl.pl .

Dariusz MROZEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, dariusz.mrozek@polsl.pl .