On-Line Data Mining

Maciej Zakrzewicz

Politechnika Poznańska
Instytut Informatyki
ul. Piotrowo 3a, 60-965 Poznań
mzakrz@cs.put.poznan.pl

Streszczenie

Celem eksploracji danych (Data Mining) jest zautomatyzowane odkrywanie statystycznych zależności i schematów w bardzo dużych bazach danych, a następnie ich reprezentacja w formie reguł logicznych, drzew decyzyjnych lub sieci neuronowych. Mimo, iż stosowane do tych celów algorytmy są bardzo obliczeniochłonne i zwykle wykonywane w trybie wsadowym, to jednak dynamiczny rozwój technologii sprzętowych pozwolił na postęp w kierunku **bieżącej eksploracji danych** (**On-Line Data Mining**). Bieżąca eksploracja danych rozszerza funkcjonalność klasycznych systemów baz danych i umożliwia implementację nowej generacji aplikacji dla analizy danych i wspomagania podejmowania decyzji.

W artykule przedstawiono system ***RD2***, który jest prototypowym środowiskiem dla bieżącej eksploracji danych, rozbudowującym własności *RDBMS Oracle7*. Scharakteryzowano kluczowe technologie wykorzystane podczas budowy modułów oprogramowania: sieciowego, analizy językowej, dostępu do bazy danych, eksploracji danych. Zilustrowane zostały także zastosowania prezentowanego środowiska w dziedzinach marketingu, zarządzania i bankowości.

Czym jest data mining?

Data mining, czyli eksploracja danych, jest nowoczesną technologią, służącą do zautomatyzowanego odkrywania statystycznych zależności i schematów w bardzo dużych bazach danych [AIS93], [FPS96], [Zak97]. Odkrywane, wcześniej nieznane zależności i schematy, przedstawiane najczęściej w formie reguł logicznych, drzew decyzyjnych lub sieci neuronowych mogą posiadać dużą wartość ekonomiczną i mogą być użyte do wspomagania podejmowania decyzji finansowych i marketingowych w przedsiębiorstwie. Często żartobliwie charakteryzuje się eksplorację danych nie jako "poszukiwanie reguł w bazach danych", ale jako "poszukiwanie pieniędzy w bazach danych" (ang. "mining for dollars"). Mimo, iż eksploracja danych wyrosła na gruncie sztucznej inteligencji i uczenia maszynowego, to jednak rozmiary stawianych przed nią problemów powodują konieczność opracowania zupełnie nowych, wyrafinowanych algorytmów, metod i architektur. Na rynku pojawia się coraz więcej produktów noszących etykietę "Data Mining", które potrafią współpracować z popularnymi systemami baz danych w celu rozwiązania specyficznego problemu decyzyjnego. Ze względu na dość luźną integrację takich systemów z systemami baz danych (klient/serwer) osiągana efektywność jest zwykle niesatysfakcjonująca dla aplikacji, które powinny szybko reagować na zmieniającą się rzeczywistość (np. decyzje o zakupach na rynkach kapitałowych).

Najczęściej przedstawianym i najbardziej chyba udanym przykładem zastosowania eksploracji danych jest problem **analizy koszyka zakupów** (ang. market basket analysis) [AIS93], [MZ97], [SA95], [SA96]. Analiza koszyka zakupów dotyczy bazy danych gromadzącej zapisy o zakupach dokonywanych przez kolejnych klientów supermarketu: dla każdej transakcji zakupu odnotowywane są wszystkie zakupione produkty (koszyk zakupów). Celem analizy koszyka zakupów jest odkrycie w bazie danych tych zbiorów produktów, które są najczęściej kupowane wspólnie, np. zbioru {"szklanka", "karafka", "pojemnik do lodu"}. Taka wiedza o zależnościach zachodzących pomiędzy sprzedawanymi produktami może być następnie użyta do planowania

skutecznych kampanii promocyjnych, rozmieszczania produktów na półkach, projektowania katalogów wysyłkowych itp.. Najczęściej wyniki analizy koszyka zakupów przedstawiane są w formie tzw. **reguł asocjacyjnych**, np.:

IF produkt="szklanka" AND produkt="karafka" THEN produkt="pojemnik do lodu"

Powyższa reguła orzeka, że kiedy klient kupuje (albo będzie kupować) produkty "szklanka" i "karafka", to jest wysoce prawdopodobne, że kupuje (albo kupi) on również produkt "pojemnik do lodu".

Co data mining ma wspólnego z bazami danych? OLAM.

Coraz bardziej powszechne są opinie, że eksploracja danych uzupełnia obecną funkcjonalność systemów baz danych [IM96]. Z punktu widzenia użytkownika optymalna byłaby pełna integracja obu technologii i traktowanie odkrywania zależności i schematów na równi z wykonywaniem zapytań do bazy danych. Integracja taka dotyczyłaby wspólnego deklaratywnego języka dla zapytań i eksploracji, unifikacji logicznej reprezentacji danych relacyjnych i regułowych, współpracy na poziomie wewnętrznych mechanizmów transakcyjności, współbieżności, ograniczeń integralnościowych, indeksowania, itp.. Niewątpliwa poprawa efektywności eksploracji danych byłaby efektem umiejscowienia algorytmów jak najbliżej danych, na których te algorytmy operują. Ponadto, w nieskomplikowany sposób, obecne aplikacje biznesowe mogłyby zostać rozszerzone o funkcje eksploracji dodane na poziomie systemu bazy danych, co wpłynęłoby na relatywnie niski koszt takiego skoku technologicznego. W efekcie, użytkownicy mogliby posługiwać się eksploracją danych tak, jak dziś posługują się bieżącą analizą *OLAP* (ang. On-Line Analytical Processing) - szybko i skutecznie [CD97], [Wid95]. Tę generację systemów eksploracji danych, szybkich, wielodostępnych i zintegrowanych z systemami baz danych nazywa się często *OLAM* (ang. On-Line Analytical Mining).

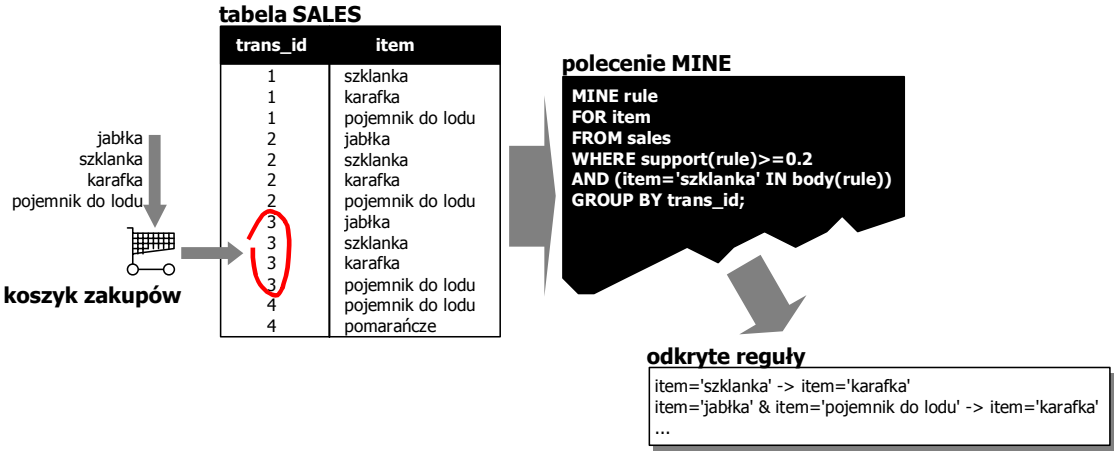
OLAM w praktyce: system RD2

System *RD2* jest prototypowym rozszerzeniem serwera Oracle7 o funkcje eksploracji danych, realizującym idee przetwarzania *OLAM* [ZJJ+98]. Powstał on i jest rozwijany na Politechnice Poznańskiej, a stanowi głównie platformę dla implementacji wyników prowadzonych badań naukowych [MZ97], [MZ97b], [MZ98], [WZ98]. Dzięki skalowalności przyjętych rozwiązań możliwe jest jego wykorzystanie w połączeniu z komercyjnymi bazami i hurtowniami danych, a przez to rozwijanie aplikacji dynamicznego wspomagania podejmowania decyzji

Z punktu widzenia użytkownika, system *RD2* jest transparentnym dodatkiem do serwera Oracle7, poszerzającym język SQL o kilka dodatkowych poleceń, przy użyciu których możliwe jest odkrywanie reguł logicznych zachodzących w tabelach bazy danych, ich przetwarzanie, składowanie i późniejsze wykorzystywanie. Z punktu widzenia programisty, *RD2* dostarcza uniwersalnego interfejsu *API*, służącego do łatwej budowy aplikacji analitycznych posiadających pełen dostęp zarówno do funkcji eksploracji danych, jak i do danych relacyjnych, których ta eksploracja dotyczy. Wreszcie z punktu widzenia technologii informatycznej, *RD2* jest sieciowym oprogramowaniem klient/serwer, współpracującym z serwerem Oracle7, wykorzystującym tak odległe od siebie rozwiązania jak analiza leksykalna i składniowa, odkrywanie logicznych reguł asocjacyjnych, komunikacja sieciowa TCP/IP czy niskopoziomowy dostęp do bazy danych.

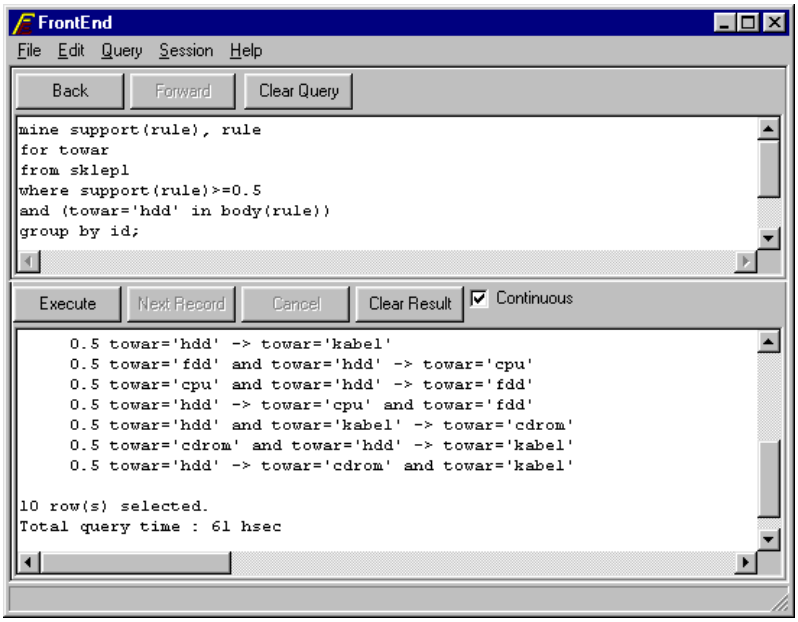
RD2 dla użytkownika

Korzystając z systemu *RD2/Oracle7*, użytkownik dysponuje rozszerzonym w składni językiem *SQL* [MZ97b]. Podczas tworzenia nowych tabel może definiować atrybuty, służące do późniejszego przechowywania reguł asocjacyjnych, np. atrybuty typu *RULE OF VARCHAR2* (reguły, których elementy są ciągami znaków), *RULE OF NUMBER* (reguły, których elementy są liczbami), *RULE OF DATE* (reguły, których elementy są datami). Tabele przechowujące reguły asocjacyjne mogą być wypełniane regułami wprowadzanymi przez użytkownika (rozszerzone polecenie *INSERT*), jak też regułami automatycznie odkrywanymi w innych tabelach. Nowe polecenie *MINE* pozwala na formułowanie szczegółowych żądań odkrywania reguł asocjacyjnych spełniających zadane, złożone kryteria dotyczące parametrów statystycznych reguły, jej zawartości, długości, itp. Poniżej, na rysunku 1, przedstawione zostało przykładowe polecenie *MINE*, odkrywające w tabeli *SALES* wszystkie reguły asocjacyjne potwierdzane (ang. support) przez co najmniej 20% wszystkich transakcji zakupu i zawierające produkt "szklanka" w części warunkowej (ang. body). Każda transakcja zakupu zapisana jest w atrybucie *ITEM* w rekordach zgrupowanych według atrybutu *TRANS_ID*.



Rys. 1. Polecenie MINE: odkrywanie specyficznych reguł asocjacyjnych

Podstawowym narzędziem komunikacji użytkownika z systemem *RD2/Oracle7* przy użyciu rozszerzonego języka *SQL* jest program ***RD2 FrontEnd***. Koncepcyjnie zbliżony do *Oracle SQL Plus*, cechuje się bardziej funkcjonalnym interfejsem, pracuje w środowisku *Microsoft Windows 95/NT* i pozwala na pracę w trybie klient/serwer w sieci *TCP/IP*. Okno programu *RD2 FrontEnd* zostało przedstawione na rysunku 2.



Rys. 2. Program RD2 FrontEnd: narzędzie komunikacji użytkownika z RD2

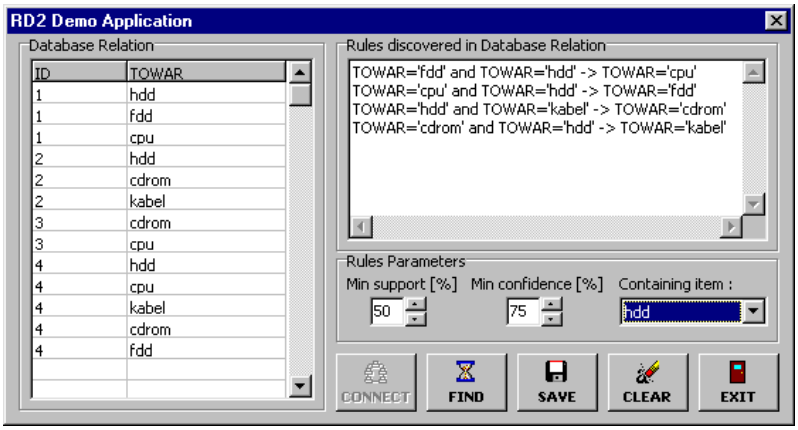
RD2 dla programisty

Funkcje systemu *RD2* mogą być wykorzystywane przez aplikacje użytkowe współpracujące ze specjalizowanym interfejsem *RD2 API* (ang. Application Programmer's Interface). Operacje dostępne na poziomie interfejsu obejmują: nawiązanie i zamknięcie zdalnego połączenia sieciowego, otwarcie i zamknięcie rozszerzonego kursora (kursora opartego dowolnym poleceniu rozszerzonego *SQL*, w tym również *MINE*), pobranie rezultatów polecenia rozszerzonego *SQL*, itp.. Interfejs *RD2 API* dostarczony został w postaci biblioteki *DLL* systemu *Microsoft Windows* i w związku z tym może być obsługiwany przez dowolne kompilatory pracujące w tym systemie. W efekcie programista uzyskuje możliwość tworzenia aplikacji dla eksploracji danych w językach takich jak: *Visual Basic*, *Borland Delphi*, *Borland C++/Builder*, *Oracle Developer/2000* itd.. Poniżej, na rysunku 3, przedstawiono fragment przykładowego kodu źródłowego aplikacji

dla eksploracji danych, która odkrywa reguły asocjacyjne zgodnie z założeniami z rysunku 1, a ich treść wyświetla na standardowym urządzeniu wyjściowym. Na rysunku 4 pokazano okno prostej przykładowej aplikacji użytkowej zbudowanej na platformie *RD2*. Aby z niej efektywnie korzystać, użytkownik nie musi posiadać żadnych doświadczeń ani w zakresie eksploracji danych, ani w zakresie formułowania poleceń w języku *SQL*.

```
/* nawiązanie połączenia z systemem RD2 */
session_id = SessionOpen("scott", "tiger", "ora7.pl", "cactus.db.com.pl");
/* wysłanie do RD2 treści polecenia MINE dla odkrywania reguł asocjacyjnych
   - otwarcie kursora */
cursor_id = QueryOpen(session_id,"mine rule for item from sales ... group by trans_id);
/* pobieranie kolejnych pozycji wyniku polecenia aż do ich wyczerpania*/
while not QueryEOD(session_id, cursor_id) {
  /* pobranie kolejnej pozycji wyniku z kursora */
  QueryNext(session_id, cursor_id);
  /* wyświetlenie pierwszej wartości bieżącej pozycji wyniku polecenia */
  printf("%s\n", FieldValue(session_id, cursor_id, 1);
}
/* zamknięcie wykorzystanego kursora */
QueryClose(session_id, cursor_id);
/* zakończenie połączenia z systemem RD2 */
SessionClose(session_id);
```

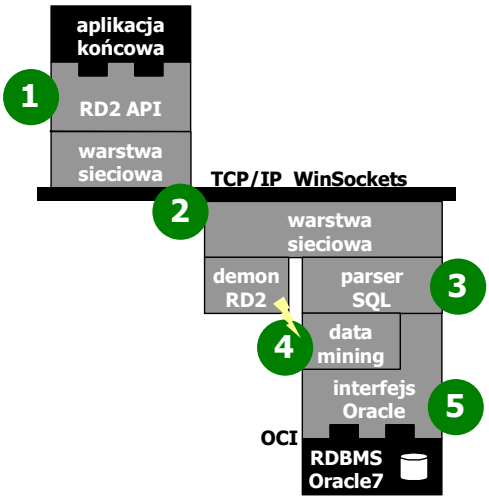
Rys. 3. Przykładowy kod aplikacji dla eksploracji danych zbudowanej w oparciu o *RD2 API*



Rys. 4. Demonstracyjna aplikacja użytkowa zbudowana na platformie *RD2*

Architektura wewnętrzna RD2

System *RD2* posiada budowę modularną, pozwalając na łatwą podmianę wyraźnie wyróżnionych komponentów w celu nieskomplikowanej adaptacji do np. odmiennego protokołu sieciowego, czy współpracy z innym systemem bazy danych. Oprogramowanie zostało przygotowane w języku C++, z użyciem kompilatorów *Borland C++ 4.5* i *Borland C++ Builder 2.0*, a funkcjonuje w konfiguracji klient/serwer, na platformie *Microsoft Windows 95/NT*. Schemat poglądowy architektury wewnętrznej *RD2* został zamieszczony na rysunku 5. Aplikacje użytkowe przekazują treści poleceń rozszerzonego języka *SQL* przez interfejs *RD2 API* (**1**), którego funkcje są odwzorowywane w funkcje warstwy komunikacji sieciowej (**2**). Po przesłaniu przez sieć komunikacyjną, polecenia rozszerzonego *SQL* są rozpoznawane przez moduł parsera (**3**) (analizatora leksykalno-składniowego). Następnie, w zależności od charakteru analizowanego polecenia, do jego wykonania angażowany jest moduł eksploracji danych (**4**) lub wyłącznie interfejs bazy danych *Oracle* (**5**). Dla każdej sesji użytkownika tworzony jest niezależny proces serwera *RD2* (**2**+**3**+**4**+**5**), a koordynacją nawiązywania połączeń zajmuje się niewielki moduł demona *RD2*.



Rys. 5. Wewnętrzna architektura systemu *RD2*

Technologie RD2: jak rozwiązać komunikację sieciową? Windows Sockets.

Ze względu na konieczność zachowania niezależności od wykorzystywanego systemu bazy danych oraz podniesienia wydajności transmisji, system *RD2* nie wykorzystuje oprogramowania sieciowego *Oracle SQL*Net V2*. Dla przesyłania treści poleceń *SQL*, ich wyników oraz komunikatów sterujących (opracowany protokół *RD2 Net*) zastosowano sieciowy interfejs programowy *Windows Sockets*, który bazuje na popularnym modelu gniazd BSD (ang. BSD sockets) [HTA+93]. Dla pobieżnego zilustrowania zasad programowania z użyciem *Windows Sockets*, na rysunku 6 zamieszczono fragment kodu przykładowego programu, przesyłającego krótki komunikat tekstowy do komputera *cactus.db.com.pl*, na port o numerze *9999*.

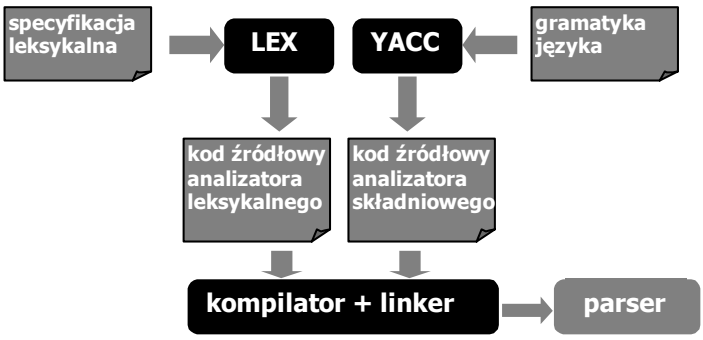
```
int sockfd;
struct sockaddr_in serv_addr;

/* ustalenie adresu odbiorcy, numeru portu i typu transmisji */
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = inet_addr("cactus.db.com.pl");
serv_addr.sin_port = htons(9999);
sockfd = socket(AF_INET, SOCK_STREAM, 0));
/* nawiązanie połączenia z odbiorcą */
connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
/* przesłanie pakietu sieciowego z przykładowym komunikatem */
send(sockfd, "Tekst do wysłania", MSG_SIZE, 0);
/* zamknięcie połączenia */
closesocket(sockfd);
```

Rys. 6. Fragment przykładowego kodu programu komunikacji sieciowej z *Windows Sockets*

Technologie RD2: jak wykonać analizę leksykalną i składniową? Lex i Yacc.

Ponieważ system *RD2* realizuje żądania wyrażone w rozszerzonym języku *SQL*, dlatego konieczne jest posiadanie oprogramowania parsera rozpoznającego dodatkowe polecenia (np. *MINE*) i dokonującego ich rozkładu leksykalnego i składniowego. Dla przygotowania kodu źródłowego parsera dla *RD2* wykorzystano znane z systemu *UNIX* narzędzia *Lex* i *Yacc* (rysunek 7). *Lex* jest programem narzędziowym, który na podstawie przygotowanej przez programistę specyfikacji automatycznie generuje kod źródłowy tzw. analizatora leksykalnego, czyli programu rozpoznającego słowa kluczowe języka. Analogicznie, *Yacc* jest programem narzędziowym, który na podstawie dostarczonej gramatyki języka automatycznie generuje kod źródłowy tzw. analizatora składniowego, czyli programu, który rozpoznaje całe polecenia języka. Użycie narzędzi *Lex* i *Yacc* pozwoliło na elastyczność modyfikacji składni rozszerzeń języka *SQL* już w czasie realizacji projektu. Nie bez

znaczenia jest również łatwość dodawania poleceń o nowej składni bez konieczności dokonywania globalnych zmian kodu źródłowego.



Rys. 7. Budowa parsera przy użyciu narzędzi *Lex* i *Yacc*

Technologie RD2: jak *C* komunikuje się z *Oracle*? OCI.

Eksploracja danych wymaga sprawnego i intensywnego dostępu do dużych wolumenów danych składowanych w bazie danych. Aby zapewnić dostęp do bazy danych na najniższym poziomie logicznym, funkcje *RD2* korzystają z programowego interfejsu *OCI* (*Oracle Call Interface*). *OCI* pozwala programiście na manipulację danymi zgromadzonymi w bazie danych *Oracle* z poziomu programu napisanego w języku wysokiego poziomu. Programista realizuje każde polecenie *SQL* za pomocą kolejno wywoływanych funkcji *OCI*, realizujących następujące zadania: analizę leksykalną i składniową, wiązanie zmiennych wejściowych, wiązanie zmiennych wyjściowych, wykonanie polecenia i pobranie rekordów wyniku. Przykładowy fragment kodu programu wykorzystującego *OCI* zamieszczono na rysunku 8.

```
/* treść bloku PL/SQL do wykonania */
char stmt[] = "begin my_procedure(:x, :y); end;";
int x;
float y;

/* analiza leksykalna i składniowa bloku PL/SQL */
if (oparse(&cda, stmt, -1, 1, 2)) oci_error(&cda);
/* wiązanie i podstawienie parametrów wywołania procedury my_procedure */
if (obndrv(&cda, ":x", -1, &xnum, sizeof (int), 3, -1, 0, 0, -1, -1)) oci_error(&cda);
if (obndrv(&cda, ":y", -1, &ynum, sizeof (float), 4, -1, 0, 0, -1, -1)) oci_error(&cda);
xnum = 10;
ynum = 5.25;
/* wykonanie bloku PL/SQL */
if (oexec(&cda)) oci_error(&cda);
/* zatwierdzenie transakcji */
if (ocom(&lda)) oci_error(&cda);
```

Rys. 8. Fragment przykładowego wykorzystania interfejsu *OCI* dla wykonania bloku *PL/SQL*

Technologie RD2: jak odkrywać reguły asocjacyjne?

Znanych jest wiele algorytmów odkrywania reguł asocjacyjnych, różniących się efektywnością i klasą zastosowań [AIS93], [AS94], MTV94], [SA95], [SA96]. *RD2* wykorzystuje zoptymalizowaną wersję najpopularniejszego algorytmu, nazywanego *Apriori* [AS94], [MZ97]. Ogólna zasada działania algorytmu *Apriori* polega na znajdowaniu najczęściej występujących zbiorów - tzw. zbiorów częstych, a następnie generowaniu reguł asocjacyjnych na podstawie każdego ze znalezionych zbiorów częstych. Z punktu widzenia złożoności czasowej kluczowym podproblemem jest tutaj znajdowanie zbiorów częstych - wymaga ono wykonania *N* lub *N+1* pełnych odczytów całej bazy danych, gdzie *N* jest rozmiarem największego znalezionego zbioru częstego. Zastosowana w *RD2* optymalizacja algorytmu *Apriori* pozwala częściowo ograniczyć zakres

przeszukiwań bazy danych oraz liczbę iteracji. W efekcie, ograniczenia narzucane przez użytkownika na odkrywane reguły asocjacyjne są skutecznie wykorzystywane do poprawy czasu odpowiedzi systemu, co pozwala na wysoce interaktywną pracę całego środowiska.

Wachlarz zastosowań RD2: kredyty, ubezpieczenia, marketing.

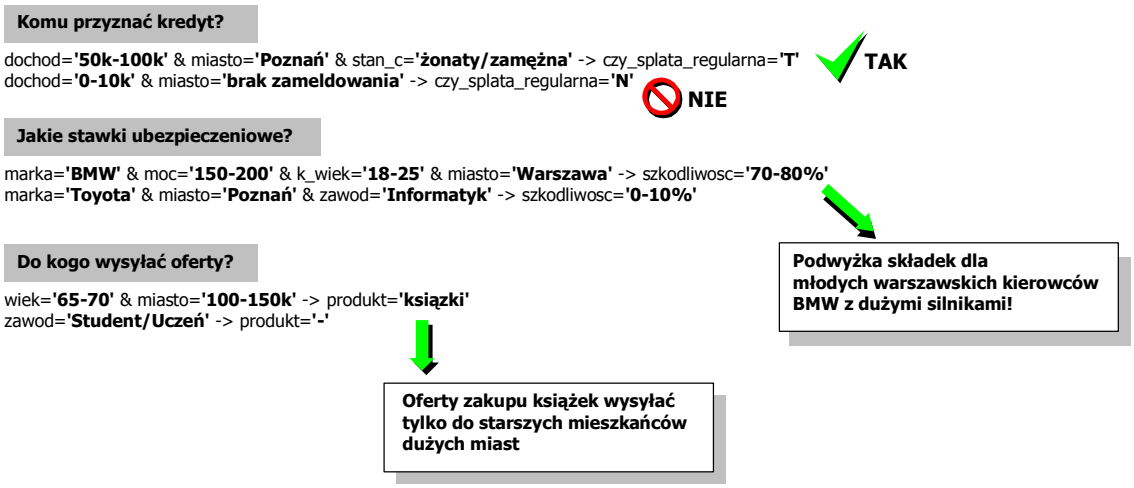
Lista możliwych zastosowań eksploracji danych, a tym samym zastosowań systemu *RD2* jest bardzo długa i jest systematycznie poszerzana. Poniżej przedstawiono kilka "sztandarowych" przykładów z dziedzin bankowości, ubezpieczeń i marketingu.

Banki gromadzą wiele szczegółowych informacji dotyczących swoich kredytobiorców: dochody, zawód, miejsce pracy, wiek, miejsce zamieszkania, wydatki itp. Gdy klient banku ubiega się o przyznanie kredytu, wtedy informacje te mogą być użyteczne dla podjęcia właściwej decyzji: czy przyznać kredyt, osiągać z tego tytułu zyski, ale ryzykować bankructwo klienta, czy też nie przyznawać kredytu, a przez to pomniejszyć swoje potencjalne zyski z odsetek. Praktyka dowodzi, że to właśnie eksploracja danych może zostać wykorzystana dla znalezienia quasi-optymalnego rozwiązania powyższego problemu. W wyniku analizy baz danych o kredytach przyznawanych w przeszłości generowane są reguły asocjacyjne uzależniające regularną spłatę kredytów od pewnych cech osobowych klienta. Następnie, znalezione reguły są aplikowane do nowych kredytobiorców, pozwalając na predykcję przyszłego postępowania kredytobiorcy wobec banku przyznającego kredyt.

Firmy ubezpieczeniowe są zainteresowane takim ustalaniem składek ubezpieczeniowych *OC/AC*, aby były one proporcjonalne do prawdopodobieństwa spowodowania wypadku przez ubezpieczonego. Również tutaj eksploracja danych może być wysoce użyteczna dla znalezienia prostych zależności pomiędzy np. marką pojazdu, mocą silnika, wiekiem kierowcy, miejscem zamieszkania, itp. a szkodliwością wyrażaną przez kwotę wypłaconych odszkodowań. Reguły asocjacyjne odkrywane w bazie danych o wypłacanych odszkodowaniach stanowią wiarygodną podstawę dla opracowania nowych taryf ubezpieczeniowych.

Coraz powszechniej spotykamy się z różnymi formami zdalnej promocji, polegającej na wysyłaniu potencjalnym klientom propozycji zakupu określonych towarów. Aby taki mechanizm cechował się wysoką skutecznością, klienci, na których kierowana jest promocja, muszą podlegać specjalnej selekcji. Taka selekcja może być dokonywana na podstawie zapisów o dotychczasowych efektach prowadzonych akcji promocyjnych (eksploracja danych!), a efektem będzie wysoki odsetek klientów, którzy odpowiadają na wysyłane propozycje.

Na rysunku 9 przedstawiono przykłady reguł asocjacyjnych, które obrazują opisane powyżej klasy zastosowań. Każda z reguł reprezentuje zależności odkryte w posiadanych bazach danych i każda może być wykorzystana do predykcji przyszłych wartości i zachowań.



Rys. 9. Reguły asocjacyjne w różnych zastosowaniach

Podsumowanie

W artykule przedstawiono systemowe środowisko *RD2* dla bieżącej eksploracji danych, które rozszerza funkcjonalność systemu zarządzania bazą danych *Oracle*. Pomimo, iż *RD2* stanowi prototyp badawczy wykorzystywany do oceny efektywności rozwijanych technik i algorytmów, to jednak jego własności mogą być

komercyjnie wykorzystywanie do budowy aplikacji dla eksploracji danych. Jego ścisła, lecz nie nierozłączna integracja z systemem zarządzania bazą danych *Oracle7* gwarantuje szerokie możliwości wykorzystania w systemach wspomagania podejmowania decyzji bazujących na popularnych hurtowniach danych lub bazach danych. Do budowy *RD2* wykorzystano szereg różnorodnych zaawansowanych technologii informatycznych, co przy jego modularnej architekturze umożliwia łatwe rozszerzanie lub migrację. System powstał w Instytucie Informatyki Politechniki Poznańskiej.

Referencje

[AIS93] R. Agrawal, T. Imielinski, A. Swami, “Mining Association Rules Between Sets of Items in Large Databases”, Proc. ACM SIGMOD Conference, pp. 207-216, Washington DC, USA, May 1993

[AMS+96] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, T. Bollinger, "The Quest Data Mining System", Proc. of the 2nd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon, August 1996

[AS94] R. Agrawal, R. Srikant, “Fast Algorithms for Mining Association Rules”, Proc. 20th Int’l Conf. Very Large Data Bases, pp. 478-499, Santiago, Chile, 1994

[CD97] S. Chaudhuri, U. Dayal, "An Overview of Data Warehousing and OLAP Technology", SIGMOD Record, Vol. 26, No. 1, March 1997

[FPS96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, The KDD Process for Extracting Useful Knowledge from Volumes of Data, Comm. of the ACM, Vol. 39, No. 11, November 1996

[HTA+93] M. Hall, M. Towfiq, G. Arnold, D. Treadwell, H. Sanders *Windows Sockets - An Open Interface for Network Programming under Microsoft® Windows™ version 1.1*, 1993

[IM96] T. Imielinski T., Manilla H., A Database Perspective on Knowledge Discovery, Comm. of the ACM, Vol. 39, No. 11, November 1996

[MTV94] H. Manilla, H. Toivonen, A. Inkeri Verkamo, “Efficient Algorithms for Discovering Association Rules”, Proc. AAAI Workshop Knowledge Discovery in Databases, pp. 181-192, July 1994

[MZ97] T. Morzy, M. Zakrzewicz, "Constraints-Driven Algorithm for Mining Association Rules On Demand", Technical Report RA-004/97, Poznań University of Technology, 1997

[MZ97b] T. Morzy, M. Zakrzewicz, "SQL-Like Language For Database Mining", 1st International Conference on Advances in Databases and Information Systems, pp. 311-317, St. Petersburg, Sept. 1997

[MZ98] T. Morzy, M. Zakrzewicz, "Group Bitmap Index: A Structure for Association Rules Retrieval", Proc. of 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press, New York, August 1998

[SA95] R. Srikant, R. Agrawal, “Mining Generalized Association Rules”, Proc. 21th Int’l Conf. Very Large Data Bases, pp. 407-419, Zurich, Switzerland, Sept. 1995

[SA96] R. Srikant, R. Agrawal, “Mining Quantitative Association Rules in Large Relational Tables”, Proc. 1996 ACM SIGMOD Conference, Montreal, Canada, June 1996

[Wid95] J. Widom, "Research Problems in Data Warehousing", Proc. 4th International CIKM Conference, 1995

[WZ98] M. Wojciechowski, M. Zakrzewicz, "Itemset Materializing for Fast Mining of Association Rules", Proc. of 2nd International Conference on Advances in Databases and Information Systems, Lecture Notes in Computer Science, vol. 1575, pp. 284-295, Poznań, 1998

[Zak97] M. Zakrzewicz, "Data Mining i odkrywanie wiedzy w bazach danych", Materialy konf. Polish Oracle Users Group PLOUG'97, str. 57-67, Zakopane, 1997

[ZJJ+98] M. Zakrzewicz, T. Januszewski, T. Jańczuk, Ł. Józefowski, W. Kaczmarek, W. Szczęsny, "RD2: Prototypowy system odkrywania reguł asocjacyjnych w relacyjnych bazach danych", Raport techniczny, Politechnika Poznańska, 1998