Analityczne rozszerzenia SQL w rozproszonej, homogenicznej strukturze danych

Marcin Gorawski[1], **Ewa Płuciennik**[1]

Streszczenie: Artykuł przedstawia zaproponowane przez autorów rozszerzenie języka SQL pozwalające na eksplorację danych oraz budowę ich modeli analitycznych w ramach rozproszonej, homogenicznej struktury danych. Dane podzielone z wykorzystaniem metody podziału poziomego kompletnego przechowywane są w autonomicznych węzłach. Przedstawione rozwiązanie obejmuje budowę modeli lokalnych oraz ich scalanie w modele globalne. Struktura logiczna systemu składa się ze zbioru operatorów realizujących operacje zdefiniowane w ramach zaproponowanej składni oraz metadanych przechowujących m.in. zbudowane modele danych, wyniki ich testowania oraz inne niezbędne informacje. Przedstawiono również przykład eksploracji z wykorzystaniem zaproponowanych rozszerzeń oraz wyniki wstępnych testów systemu.

Słowa kluczowe: SQL, rozproszona eksploracja danych, rozproszone bazy danych.

1. Wstęp

Od wielu lat w dziedzinie baz danych daje się zaobserwować tendencja do gromadzenia coraz większych ilości informacji oraz umożliwienia ich efektywnego przetwarzania oraz analizowania. Informacje te stanowią źródło danych dla systemów bezpośredniego przetwarzania analitycznego (OLAP) oraz systemów wspierania podejmowania decyzji (DSS). Użyteczność tego typu systemów oceniana jest przez pryzmat jakości uzyskiwanych wyników oraz wydajności, rozumianej jako czas realizacji analiz danych. Istnieje kilka sposobów poprawy efektywności przetwarzania danych: odpowiednie indeksowanie, widoki zmaterializowane, partycjonowanie danych oraz przetwarzanie rozproszone (patrz Bernardino, Madeira, 2000). Ostatnia z wymienionych technik jest szczególnie obiecująca, ponieważ modyfikacje sprzętowe nie gwarantują takiej poprawy wydajności jaką może zapewnić podzielenie procesu przetwarzania na części i wykonanie ich równolegle (patrz Ullman, Widom, 1997). Przetwarzanie równoległe może być realizowane w ramach systemów wieloprocesorowych czy wielokomputerowych, również rozproszonych geograficznie.

Biorąc pod uwagę następujące aspekty związane z obecnymi tendencjami w dziedzinie przetwarzania danych:

- analizowane dane są gromadzone w bazach danych (najczęściej relacyjnych),
- efektywność przetwarzania danych można znacząco poprawić zrównoleglając proces analizy,

[1] Instytut Informatyki, Politechnika Śląska, ul. Akademicka 16, 44-100 Gliwice
e-mail: {M.Gorawski, E.Pluciennik}@polsl.pl

- przetwarzanie rozproszone staje się koniecznością w przypadku instytucji działających w oparciu o autonomiczne oddziały, często rozproszone geograficznie,
- podstawowym sposobem dostępu do danych gromadzonych w bazach danych jest język zapytań np.: SQL,
- analiza danych coraz częściej obejmuje eksplorację danych, której wynikiem są analityczne modele danych pozwalające określać globalne cechy danych czy przewidywać pewne ich cechy (patrz Hand, Mannila, Smyth, 2005),

autorzy podjęli się próby stworzenia schematu realizacji podstawowej jak i zaawansowanej analizy danych z poziomu zapytań języka SQL w rozproszonej strukturze danych.

Istnieje kilka rodzajów rozproszonych struktur danych. Dwa podstawowe opierają się na podziale pionowym oraz poziomym danych. Podział pionowy zakłada podział relacji operatorem projekcji, poziomy selekcji. Prezentowane rozwiązanie skupia się na podziale poziomym kompletnym (patrz Elmasri, Navathe, 1999). W tym przypadku krotki relacji przydzielane do poszczególnych węzłów systemu muszą spełniać warunek zwany strażnikiem. Warunek strażnika ma postać: $\delta_{c_i}(R)$, gdzie R oznacza relację zawierającą wszystkie krotki, C_i oznacza warunek jaki musi spełniać krotka przydzielona do i-tego węzła, przy czym w relacji R nie istnieje krotka jednocześnie spełniająca warunki z dwóch różnych węzłów:
$\neg \exists\ t[R] : i \neq j, t[R] \in \delta_{c_i}(R) \wedge t[R] \in \delta_{c_j}(R)$.

W ramach dotychczasowych prac określono i zaimplementowano (w środowisku Oracle10g/Java) schemat realizacji zapytań SQL do rozproszonej, homogenicznej struktury danych, uwzględniający realizację funkcji agregujących zarówno dystrybucyjnych, algebraicznych jak i holistycznych bez konieczności przesyłania danych między węzłami. Uzyskane wyniki zostały opublikowane we wcześniejszych pracach autorów (patrz Gorawski, Płuciennik, 2005, 2006, 2007). W niniejszej pracy przedstawiono propozycję analitycznego rozszerzenia języka SQL i sposób jego realizacji w rozproszonej strukturze danych. Podjęto również dyskusję nad problemem scalania wyników rozproszonej eksploracji danych.

2. Eksploracja danych z poziomu języka zapytań do bazy danych

Propozycje analitycznych języków zapytań do baz danych lub rozszerzeń samego SQL'a pojawiły się już w latach 90-tych. Rozszerzenia tego typu nie zostały do tej pory ujęte w standardzie SQL, a wymienione poniżej, wybrane rozwiązania stanowią prace badawcze. W roku 2001 pojawił się standard SQL/MM (patrz ISO/IEC JTC1 SC32, 2001). Definiuje on zagadnienia związane z eksploracją danych w oparciu o obiektowe cechy baz danych (np.: możliwości definiowania typów oraz metod) oraz XML jako język definicji struktur danych, nie obejmuje jednak żadnych nowych konstrukcji samego SQL'a. DMQL (Data Mining Query Language) został zaprojektowany na Uniwersytecie Simona Frasera w Kanadzie i częściowo zaimplementowany systemie DBMiner (patrz Fu, Han, Koperski, Wang, Zaiane 1996).

MSQL powstał na Uniwersytecie Rutgers w USA (patrz Imieliński, Virmani, 1999). Na Politechnice w Turynie stworzono, we współpracy z Politechniką Mediolańską operator MineRule stanowiący rozszerzenie SQL'a (patrz Ceri, Meo, Psaila,1996 i 1998). Na Politechnice Poznańskiej prowadzono prace nad MineSQL będącym rozszerzeniem SQL wspomagającym generowanie reguł (patrz Morzy, Zakrzewicz, 1997). Jedną z najnowszych propozycji jest Mql - algebraiczny język zapytań służący do eksploracji danych (patrz Baglioni, Turini, 2005). Podejmowano również udane próby implementacji algorytmów eksploracji danych przy pomocy standardowych elementów SQL'a (patrz Cereghini, Ordonez, 2000 oraz Dunemann, Sattler, 2001).

2.1. Analityczne rozszerzenie SQL'a

Zaproponowane przez autorów rozszerzenie uwzględnia możliwość budowy modeli predykcji, klastrowania, klasyfikacji i asocjacji. Zbudowane modele można testować (modele klasyfikacji, predykcji) oraz stosować na nowych danych (modele klasyfikacji, predykcji, klastrowania). Budowę modelu przeprowadza się z wykorzystaniem dowolnego algorytmu zaimplementowanego na poziomie bazy danych. Uwzględniono również możliwość określenia metody dyskretyzacji danych, w przypadku gdy część ich atrybutów jest numeryczna i może przyjmować dowolną war-tość z danego zakresu. Składnia rozszerzenia przedstawia się następująco:

```
<zbiór_danych>
<rodzaj_modelu> <rodzaj_operacji> <nazwa_modelu>
[TARGET <atrybut_docelowy>] [OBJECT <identyfikator_obiektu>]
WITH (<lista_parametrów_modelu>)
BY <lista_atrybutów_przestrzeni_analizy>
[DISC WITH (<lista_parametrów_dyskretyzacji>)]

<zbiór_danych> ::= SELECT {* || <lista_kolumn>} FROM <tabela>
<tabela> ::= <tabela_bazy_danych> || <widok>
<rodzaj_modelu> ::= PREDICT || CLUSTER || CLASSIFY || ASSOC
<rodzaj_operacji> ::= TRAIN || TEST || APPLY
<lista_parametrów_modelu> ::= <algorytm> [,<parametry_algorytmu>]
<lista_parametrów_dyskretyzacji>
  ::= <metoda_dyskretyzacji> [,<parametry_metody>]
```

Zakładamy, że wszystkie dane potrzebne do budowy (testowania, stosowania) modelu zgromadzone są w pojedynczej tabeli bądź widoku opartym na większej liczbie tabel. Fraza BY określa wymiary przestrzeni analizy czyli atrybuty danych, które ma uwzględniać model. Opcjonalna fraza TARGET określa atrybut docelowy (klasę) w przypadku modeli klasyfikacji. Również opcjonalna, fraza OBJECT określa atrybut jednoznacznie identyfikujący krotki danych. Atrybut ten wymagany jest w przypadku testowania bądź stosowania modelu na nowych danych.

3. Rozproszona eksploracja danych

W literaturze (patrz Chan, Prodromidis, Stolfo,2000) spotyka się trzy główne podejścia do problemu rozproszonej eksploracji danych:

- przeniesienie wszystkich danych w jedno miejsce - w zasadzie nierealne do zastosowania ze względu na ogromne ilości danych używane w eksploracji, choć dające najbardziej dokładne rezultaty,
- zgromadzenie w jednym miejscu reprezentantów danych lokalnych - problemem jest tu sposób wyboru reprezentantów,
- metauczenie – uczenie się z modeli lokalnych.

Dla ostatniego podejścia stosowane są trzy techniki metauczenia:

- głosowanie większością, w którym każdy z modeli lokalnych ma równoważny głos,
- głosowanie ważone, w którym (na podstawie wcześniejszych obserwacji) nadaje się głosom modeli lokalnych odpowiednie wagi,
- głosowanie z sędzią, który podejmuje decyzje w przypadku, kiedy na podstawie głosowania modeli lokalnych nie udało się określić jednoznacznie np.: klasy obiektu.

Innym podejściem jest modyfikowanie istniejących algorytmów do wersji rozproszonych, których działanie opiera się m.in. na uzgadnianiu, w trakcie realizacji statystyk potrzebnych do przeliczania wartości parametrów, z których korzysta dany algorytm (patrz Caragea, Honavar, Silvescu, 2004).

3.1. Scalanie modeli lokalnych w model globalny

Zaproponowane przez autorów podejście polega na próbie scalenia modeli lokalnych w jeden model globalny. Podstawowym założeniem jest to, że modele lokalne budowane są w postaci zbiorów reguł decyzyjnych kompletnych ze względu na wymiary przestrzeni analizy. Poza tym modele muszą być zgodne pod względem struktury i zawartości zbiorów treningowych, przestrzeni analizy (jej wymiarów) oraz ewentualnych parametrów dyskretyzacji.

Model stanowi zbiór reguł postaci:

Jeżeli P to K: $P \rightarrow K$, gdzie P oznacza przesłankę reguły, K jej konkluzję (w przypadku modelu klasyfikacji jest to określona wartość atrybutu docelowego).

Dla n-wymiarowej przestrzeni analizy o A_1 do A_n atrybutach przyjmujących wartości ze zbiorów, odpowiednio $a_1, \ldots, a_n$ oraz atrybutu docelowego C przyjmującego wartości ze zbioru c:

$P : A_1 \in (sub(a_1))\ and\ A_2 \in (sub(a_2))\ and\ \ldots\ and\ A_n \in (sub(a_n))\ K : C = c^i$, gdzie $sub(a)$ oznacza podzbiór zbioru a.

Reguła jest kompletna ze względu na wymiary przestrzeni analizy jeśli w przesłance występują wszystkie atrybuty tej przestrzeni.

Jeżeli wartość danego atrybutu A_i przestrzeni nie jest istotna dla danej reguły, to w przesłance warunek dla tego atrybutu przyjmuje postać:

$sub(a_i) = a_i, P : A_i \in (a_i)$

Scalenie modeli lokalnych sprowadza się do scalenia ich zbiorów reguł. W przypadku tej operacji możemy mieć do czynienia z dwoma rodzajami reguł wymagającymi odpowiedniego przetworzenia. Pierwszym rodzajem są reguły sprzeczne.

Reguła $R_1 : P_1 \rightarrow K_1$ oraz reguła $R_2 : P_2 \rightarrow K_2$ są sprzeczne jeżeli:

$K1 \neq K2$ oraz podzbiory dla wszystkich wartości atrybutów P_1 i P_2 mają niepuste części wspólne:

$\forall A_i, i \in (1..n) : P_1 : sub(a_i) \cap P_2 : sub(a_i) \neq \oslash$

Drugi rodzaj stanowią tzw. podreguły.

Reguła $R_1 : P_1 \rightarrow K_1$ jest podregułą reguły $R_2 : P_2 \rightarrow K_2$ jeżeli: $K_1 \equiv K_2$ oraz podzbiory wartości wszystkich atrybutów P_1 są podzbiorami wartości odpowiednich atrybutów P_2:

$\forall A_i, i \in (1..n) : \exists P_1 : sub(a_i) \subset P_2 : sub(a_i)$

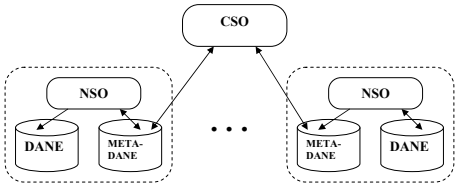
Podreguły czyli reguły, które zawierają się w innej regule (zwanej nadrzędną) należy usunąć z modelu globalnego. W przypadku reguł sprzecznych można zastosować następujące strategie usuwania sprzeczności:

- usunąć obie reguły z modelu globalnego,
- usunąć jedną z nich,
- rozdzielić reguły usuwając z obu części wspólne przesłanek dla przynajmniej jednego atrybutu,
- rozdzielić reguły usuwając z jednej z nich części wspólne przesłanek dla przynajmniej jednego atrybutu,
- nie usuwać reguł sprzecznych (ani podreguł) i zastosować głosowanie większością.

Pierwszy i ostatni sposób jest najprostszy, pozostałe wymagają decyzji co usuwać. Może zostać ona pojęta w oparciu o zawartość globalnej tabeli liczności, przechowującej informacje o globalnej liczbie krotek ze wszystkimi kombinacjami wartości atrybutów przestrzeni analizy (sekcja 4.2).

4. Komponenty systemu

Na system rozproszonej eksploracji danych z poziomu SQL'a składają dwa główne komponenty: operatory realizujące elementy zaproponowanej składni oraz metadane przechowujące niezbędne informacje. Rysunek 1 przedstawia ogólny schemat logiczny systemu. Na poszczególne węzły systemu składają się dane lokalne, metadane oraz NSO (operatory strony węzła). Zapytania są zadawane do tak rozproszonej struktury poprzez klienta SQL, w którym zostały zaimplementowane operatory globalne CSO (operatory strony klienta).



Rysunek 1. Struktura logiczna systemu

4.1. Operatory rozproszonej analizy danych

Operacje konieczne do realizacji zaproponowanej składni SQL w ramach rozproszonej struktury danych są definiowane w postaci operatorów ogólnych i ich podoperatorów realizujących bardziej szczegółowe etapy przetwarzania. Operatory ogólne dzielą się na operatory strony klienta i strony węzła. Operatory strony klienta to:

- operator metadanych modelu - wypełnia metadane modelu na podstawie zapytania analitycznego,
- operator scalania modeli,
- operator uzgodnienia zakresów dyskretyzacji - podaje globalne minimalne i maksymalne wartości atrybutów numerycznych,
- operator uzgodnienia list wartości atrybutów - dla każdego z wyliczeniowych atrybutów przestrzeni analizy tworzy globalne listy ich dopuszczalnych wartości,
- operator globalnej tabeli liczności - scala lokalne tabele liczności.

Operatory strony węzłów:

- operator dyskretyzacji,
- operator budowy modelu i wszystkie niezbędne jego operatory podrzędne,
- operator testowania modelu i wszystkie niezbędne jego operatory podrzędne,
- operator stosowania i wszystkie niezbędne jego operatory podrzędne.

Operator budowy modelu MB można zdefiniować następująco:
$MB(model) \rightarrow rule\_set$
Model określa nazwę modelu, która służy do pobrania wszystkich niezbędnych o nim informacji z metadanych (rysunek 2):
$\delta_{(model\_name=model)}(MODELS)$
$\delta_{(model\_name=model)}(MODEL\_DIMS)$
$alg \leftarrow \pi_{(alg\_id)}\delta_{(model\_name=model)}(MODELS)$
$\delta_{(alg\_id=alg)}(ALGORITHMS)$
$\delta_{(alg\_id=alg)}(ALG\_PARAMS)$
Podoperatory MB stanowią:

- operator budowy drzewa decyzyjnego BDT – opcjonalny, dla rodzaju modelu CLUSTER lub CLASSIFY, tworzy relację drzewa,
- operator redukcji drzewa decyzyjnego RDT – opcjonalny, redukuje drzewo decyzyjne scalając liście homogeniczne, usuwając nadmiarowe,
- operator generacji reguł GR z drzewa decyzyjnego - tworzy lokalne zbiory reguł (rule_set).

Podoperatory operatora CR (scalania modeli w postaci reguł) to:

- operator tworzenia postaci kompletnej reguły,
- operator wyszukiwania reguł sprzecznych,
- operator wyszukiwania podreguł.

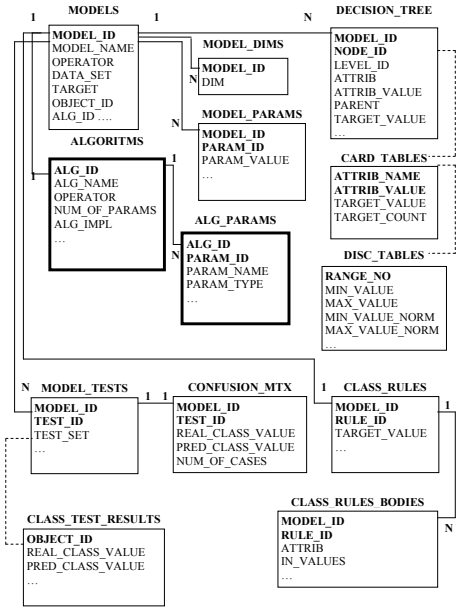
Pierwszym etapem procesu scalania modeli lokalnych jest utworzenie zbioru zawierającego wszystkie reguły modeli lokalnych oraz uzupełnienie ich do postaci kompletnej ze względu na wymiary analizy. Etap drugi polega na iteracyjnym stosowaniu operatorów wyszukiwania reguł sprzecznych oraz podreguł, a następnie zastosowaniu, dla znalezionych reguł sprzecznych jednej ze strategii usuwania sprzeczności. Warunkiem zakończenia procesu jest, nie zmieniająca się w kolejnych iteracjach, liczba pozostawionych reguł.

Złożoność obliczeniowa całego procesu obejmującego budowę modeli lokalnych oraz ich scalanie stanowi sumę złożoności algorytmu budowy modeli lokalnych (etap realizowany równolegle) oraz złożoności algorytmu scalania. Złożoność scalania, z kolei zależy głównie od ilości reguł, w mniejszym stopniu od liczby atrybutów przestrzeni analizy oraz liczności zbiorów dopuszczalnych wartości atrybutów. W jednej iteracji algorytmu scalania każda reguła jest sprawdzana tylko raz, a liczba samych iteracji jest różna, w testach nie przekroczyła jednak dziesięciu.

4.2. Metadane systemu

Metadane systemu można podzielić następująco:

- metadane pierwotne (podstawowe) - wypełnione wcześniej i umieszczone na każdym z węzłów, przechowują informacje zaimplementowanych algorytmów eksploracji oraz operatorów je realizujących,
- metadane bieżącej analizy (wtórne) - przechowują informacje o zbudowanych modelach, wynikach ich testów i stosowania, można je podzielić na:
  - lokalne (modele lokalne),
  - zaprzyjaźnione (modele z innych węzłów systemu),
  - globalne (scalone modele oraz globalne zbiory pomocnicze),
- metadane pomocnicze (robocze) - przechowują niezbędne informacje tymczasowe.



Rysunek 2. Fragment struktury metadanych

Na rysunku 2 pokazano fragment metadanych. Tabele zaznaczone pogrubioną kreską to metadane pierwotne. Tabele, których powiązania zaznaczono linią przerywaną należą do metadanych pomocniczych. Są to m.in. tabele przedziałów dyskretyzacji DISC_TABLE oraz tabele liczności atrybutów CARD_TABLE. Tabele liczności są tworzone w trakcie budowy drzewa decyzyjnego, wspomagają one obliczenia związane z określaniem warunków dzielenia węzła drzewa na węzły potomne. Tabela liczności dla każdego z wymiarów przestrzeni analizy, każdej jego wartości oraz każdej wartości atrybutu docelowego podaje liczbę krotek w zbiorze treningowym bądź testowym. Pozostałe stanowią metadane bieżącej analizy.

5. Testy klasyfikacji

Zaproponowane rozszerzenie SQL'a oraz wszystkie niezbędne operatory i metadane systemu zostały zaimplementowane w środowisku Oracle10g z wykorzystaniem języka Java oraz PL/SQL. Testy budowy i scalania modeli przeprowadzono dla modeli klasyfikacji budowanych na zbiorach z danymi ze spisu ludności, które są wykorzystywane jako zbiory przykładowe do eksploracji danych z wykorzystaniem Oracle Data Mining. Zbiór treningowy podzielono losowo na trzy części (węzły systemu).

5.1. Przykład testowy

W ramach testów wykorzystano następujące zapytanie analityczne:

```
SELECT * FROM census_build
CLASSIFY TRAIN model_3
TARGET class OBJECT person_id
WITH (ID3)
BY relationship, marital_status, occupation, sex
```

Wynikiem jego realizacji jest zbudowanie lokalnych modeli klasyfikacji danych ze zbioru *census_build* w 4-wymiarowej przestrzeni analizy, atrybut docelowy stanowi pole *class*. Krotka (rekord tabeli) danych jest identyfikowana jednoznacznie przez pole *person_id*. Modele lokalne budowane są za pomocą bezparametrowego algorytmu ID3 (patrz Quinlan, 1986). Po zbudowaniu modeli lokalnych zbiory reguł je prezentujących są scalane w model globalny. Algorytm ID3 został zaimplementowany przy pomocy następujących podoperatorów:

- wstawiania węzła drzewa - insertTreeNode(model_id, node_id, level_id, attrib, attrib_value, parent, target_value),
- sprawdzania homogeniczności danych w węźle pod względem wartości atrybutu docelowego - checkNodeHomogenity(model_id, node_id),
- tworzenia dla węzła tabeli liczności - countAttribs(model_id, node_id) → card_table,
- wyboru atrybutu decyzyjnego dla kolejnych węzłów drzewa - getHighestGainAttrib(card_table):
  - wyznaczania zysku informacyjnego dla danego atrybutu - countGain(card_table, attrib),
  - wyznaczania entropii danych w węźle - countEntropy(counts), counts – zbiór liczności.

5.2. Wyniki testów

W ramach testów budowano trzy modele lokalne (ML_1, ML_2, ML_3) oraz model MGT, którego zbiorem treningowym był zbiór stanowiący sumę zbiorów treningowych dla modeli lokalnych. W tabeli 1 zebrano cechy, zbudowanych dla wymienionych wyżej modeli drzew decyzyjnych. Liczba reguł modeli lokalnych odpowiada liczbie liści w drzewie decyzyjnym po zastosowaniu operatora redukcji. Modele lokalne scalano w model globalny stosując dwie strategie: usuwanie reguł sprzecznych (model MG_UR) oraz ich rozdzielanie (model MG_RR).

Model, w którym usunięto reguły sprzeczne składał się z 27 reguł, a model, w którym je rozdzielono z 51. Tabela 2 pokazuje wyniki testowania modeli tym samym zbiorem testującym. Najlepszą dokładność uzyskał oczywiście model MGT - model powstaje w przypadku przeniesienia wszystkich danych w jedno miejsce. Średnia dokładność modeli lokalnych wyniosła 0,86. Dokładność modeli scalanych była gorsza o około 4,2 %.

Na jakość modeli scalanych największy wpływ ma dobór strategii postępowania z regułami sprzecznymi. Celem jest uzyskanie dokładności modeli scalanych

Tabela 1. Zbudowane modele klasyfikacji

model	**liczba rekordów treningowych**	**drzewo nie zredukowane**			**drzewo zredukowane**		
		węzły	liście	poziomy	węzły	liście	poziomy
ML_1	878	115	91	5	39	27	5
ML_2	950	120	95	5	41	33	5
ML_3	904	121	95	5	44	33	5
MG_T	2732	195	147	5	84	69	5

Tabela 2. Wyniki testów modeli klasyfikacji

Model	dokładność
ML_1	0,86
ML_2	0,83
ML_3	0,88
MG_T	0,88
MG_RR	0,82
MG_UR	0,81

zbliżonej do średniej dokładności modeli lokalnych. Określenie najlepszej metody postępowania z regułami sprzecznymi wymaga przeprowadzenia dalszych testów. Oprócz testowania osobno każdej ze strategii autorzy planują testowanie metody adaptacyjnej postępowania z regułami sprzecznymi, która polega na tym, że w każdej kolejnej iteracji scalania (lub nawet dla każdej konkretnej pary reguł) dokonuje się wyboru strategii usuwania sprzeczności w oparciu o globalną tabelę liczności lub inne statystyki.

6. Podsumowanie

Zaproponowane przez autorów rozszerzenie zostało zaimplementowane w części związanej z budową modeli klasyfikacji. System jest dedykowany rozproszonym strukturom danych, ale umożliwia również, dla struktur lokalnych zrównoleglenie przetwarzania danych bez konieczności stosowania równoległych wersji algorytmów eksploracji. Jego główną zaletą jest połączenie prostoty języka deklaratywnego, jakim jest SQL z możliwością zaawansowanej, rozproszonej analizy danych. Sposób implementacji zdefiniowanych rozszerzeń jest uniwersalny w ramach relacyjnych baz danych - nie wykorzystuje on żadnych specyficznych cech związanych np.: z interfejsem obiektowym. Dalszych testów oraz dopracowania wymagają strategie scalania modeli lokalnych w globalny tak, aby uzyskiwać modele jakością zbliżone do modeli lokalnych. Uzyskane strategie można będzie zaadaptować do scalania modeli klastrowania oraz reguł asocjacji i, być może, predykcji, pa-

miętając oczywiście o założeniu dotyczącym postaci modelu. Skalowanie systemu jest realizowane poprzez dodanie kolejnych węzłów, które przejmują część danych. Z administracyjnego punktu widzenia sprowadza się to do rozszerzenia informacji o strukturze systemu o dane dotyczące sposobu komunikacji (adres, port, użytkownik, itp.) z nowymi węzłami.

Literatura

Baglioni, M and Turini, F. (2005) MQL: An Algebraic Query Language for Knowledge Discovery. *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 3-540-20119-X, vol.LNAI 2829, 225-236.

Bernardino, J and Madeira, H. (2000) Data Warehousing and OLAP: Improving Query Perform-ance Using Distributed Computing. *12th Conference on Advanced Information Systems Engineering.* Stockholm, Sweden.

Caragea, D, Honavar, V. and Silvescu, A. (2004) A Framework for Learning from Distributed Data Using Sufficient Statistics and Its Application to Learning Decision Trees. *Int. J. Hybrid Intell. Syst.*,vol. 1, no. 2, 80-89.

Cereghini, P and Ordonez, C. (2000) SQLEM: Fast Clustering in SQL using the EM Algorithm. *SIGMOD Conference*, 559-570.

Ceri, G, Meo, R. and Psaila, G.(1996) A New SQL-like Operator for Mining Association Rules.Bombaj, Indie. *Proc. VLDB*, 122-133.

Ceri, G, Meo, R. and Psaila, G.(1998) An Extention to SQL for Mining Association Rules. *Data Mining and Knowledge Discovery.*

Chan, P, Prodromidis, A. and Stolfo, G.(2000) Meta-learning in distributed data mining systems: Issues and approaches. *Advances of Distributed Data Mining.* AAAI Press.

Dunemann, O and Sattler, K. (2001) SQL Database Primitives for Decision Tree Classifiers. *Proc. of the 10th ACM CIKM Int. Conf. on Information and Knowledge Management*, 379-386.

Elmasri, R.and Navathe, S. (1999) *Fundamentals of Database Systems.* Third Edition. Addison-Wesley.

Fu, Y., Han, J., Koperski, K., Wang, W. and Zaiane, O. (1996) DMQL: A Data Mining Query Language for Relational Database. *Proc. Of SIGMOD Workshop DMKD.* Montreal, Kanada.

Gorawski, M. and Płuciennik, E. (2005) Analiza danych w systemie telemetrycznych hurtowni danych z wykorzystaniem rozszerzeń SQL/Oracle10g. *Bazy Danych - Modele, technologie, narzędzia. Analiza danych i wybrane zastosowania* ISBN 83-206-1572-0, pp. 59-67, WKŁ.

Gorawski, M. and Płuciennik, E. (2006) Realizacja zapytań SQL w rozproszonym środowisku agentów programowych. *Bazy Danych - Struktury, Algorytmy, Metody. Nowe technologie baz danych*, ISBN 83-206-1611-5, pp. 149-155, WKŁ 2006

Gorawski, M. and Płuciennik, E. (2006) Spatial Telemetric Data Warehouse and Software Agents as Environment to Distributed Execute SQL Queries. *Proceedings of the International Multiconference on Computer Science and Information Technology*, ISSN 1896-7094, pp. 243-252

Gorawski, M. and Płuciennik, E. (2007) Ogólny schemat realizacji zapytań SQL w homogenicznej rozproszonej strukturze danych. *Bazy Danych - Nowe technologie. Archtektura, metody formalne i zaawansowana analiza danych*, ISBN 978-83-206-1648-4, pp. 137-145, WKŁ 2007

Hand, D., Mannila, H and Smyth, P. (2005) *Eksploracja danych.* Wydawnictwa Naukowo Techniczne, Warszawa.

Imieliński, T. and Virmani, A. (1999) MSQL: A Query Language for Database Mining *Journal Data Mining and Knowledge Discovery.* 373-408.

ISO/IEC JTC1 SC32 (2001) FCD 13249-6 SQL/MM-Part 6: Data Mining

Morzy, T. and Zakrzewicz, M. (1997) SQL-Like Language for Database Mining *Proc. of the First East-European Symposium on Advances in Databases and Information Systems* 311-317.

Quinlan, R. (1986 Induction of decision trees *Machine Learning.* vol. 1, pp. 81-106.

Ullman, J.and Widom, J. (1997) *A First Course in Database Systems.* Prentice-Hall.

SQL analitycal enhancement in a distributed, homogenic data structure

Article presents analytical SQL syntax enhancement allowing data mining and data model building in a distributed, homogenic data structure. Data divided according to the complete horizontal fragmentation are stored at autonomous nodes. Presented solution includes local model building and its combining into global models. Logical system structure consists of set of operators realizing operation defined within the confines of proposed SQL syntax and meta-data storing built models, its tests results and other necessary information. An example of data mining model building and preliminary system tests are presented.