STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Ewa PŁUCIENNIK-PSOTA
Silesian University of Technology, Institute of Computer Science

METADATA OF A DISTRIBUTED DATA MINING SYSTEM

Summary. Distributed computing and data mining are two elements essential for many commercial and scientific organizations. This article presents metadata of a distributed data mining system based on combining local models created in distributed nodes into global models. This metadata is divided into following categories: primal, derivative, work and test data for local and global models.

Keywords: metadata, databases, distributed data mining, SQL

METADANE ROZPROSZONEGO SYSTEMU EKSPLORACJI DANYCH

Streszczenie. Rozproszone przetwarzanie oraz eksploracja danych są obecnie zasadniczymi elementami działalności wielu jednostek naukowych, a także organizacji komercyjnych. Artykuł opisuje metadane wykorzystane w systemie rozproszonej eksploracji danych, opartym na scalaniu modeli lokalnych, powstających w rozproszonych węzłach systemu, w modele globalne. Obejmują one metadane pierwotne, wynikowe, robocze, a także testowe dla modeli lokalnych oraz globalnych.

Słowa kluczowe: metadane, bazy danych, rozproszona eksploracja danych, SQL

1. Introduction

Metadata is simply defined as “data about data” [1]. Metadata is widely used in contemporary data storage systems. Metadata helps to describe structure of information so it is easy to access, retrieve and understand. Format of metadata is not standardized and it depends on what kind of data it describes. For example, in Geographic Information Systems metadata is defined as a formal dataset description – a kind of a summary of a set [2]. Metadata can be stored in a text file (or any other binary file), database or data warehouse [3], in a form of an

XML document, records or simple descriptions. Metadata presented in the article applies to a distributed data mining system.

Distributed data mining systems (DDMS) are created for exploring huge amount of data. Knowledge which can be retrieved from the data is valuable and requires appropriate storage and management. Metadata is the key element for efficient knowledge management. Information which should be stored as metadata in DDMS encompasses:

- system structure - description of distributed nodes: how nodes can be accessed, what kind of data storage they offer etc.,
- data mining algorithms which can be used to analyze the data,
- knowledge - results of data mining task in a form of a data model (classification rules, decision tree, clusters etc.).

In [3] authors proposed for their Distributed Data Mining Tool a metadata warehouse which stores all information in a form of XML documents. This information concerns, aside of the mentioned above elements, calculation capabilities of each node (CPU type, frequency, memory capacity etc.). Results of the data mining tasks are stored in PMML (Predictive Model Markup Language) format. Authors of the grid system presented in [4] proposed usage of a knowledge map which stores so-called meta-knowledge (knowledge about knowledge). Knowledge map can be defined as some kind of a guide for a set of resources [5]. Meta-knowledge is responsible for detecting the sources of knowledge and representing their relationships with concepts of a given application domain. Relationships are stored as tuples and the data mining task results as files. In [6] authors discussed an XML metadata model for the heterogeneous resource management in Knowledge Grid. Stored information is divided into three repositories. Knowledge Metadata Repository describes features of the data, software and tools. Results of the data mining tasks are stored in Knowledge Base Repository. Knowledge Execution Plan Repository stores the tasks execution plans over the grid.

Metadata presented in this article is stored in a form of database tables and XML files. XML files are used only for storing information how to connect with the distributed system nodes. This approach allows metadata to be queried with SQL.

2. Distributed Data Mining System

Distributed data mining system (DDMS) whose metadata is presented in this article has been described in [7, 8 and 9]. Basic features of proposed solution are a complete horizontal data fragmentation and a model form understood by a human being. Building global data model consists of two stages. In the first one local models are built in a parallel manner. Second one

consists of combining these models into a global data picture in one of the system's nodes. There is no need for constructing distributed version of data mining algorithm.

An analysis of a huge amount of information is feasible only if information systems are used. First, information needs to be accumulated and stored in a persistent structure enabling effective data access and management. The most popular storage way is a relational database. Regarding the circumstances mentioned above, the system was designed to allow querying distributed data with SQL [10]. The system also uses an enhancement of SQL for distributed data mining. This enhancement was implemented in a form of operators. Detailed description of the enhancement can be found in [11].

The system is composed of two main elements: enhancement SQL operators (which are responsible for building, testing and applying data models) and metadata for storing any necessary information. Fig. 1 presents the general logical system structure. Individual system's nodes are composed of local data, metadata and node side operators (NSO). Client side operators are implemented in the client application.

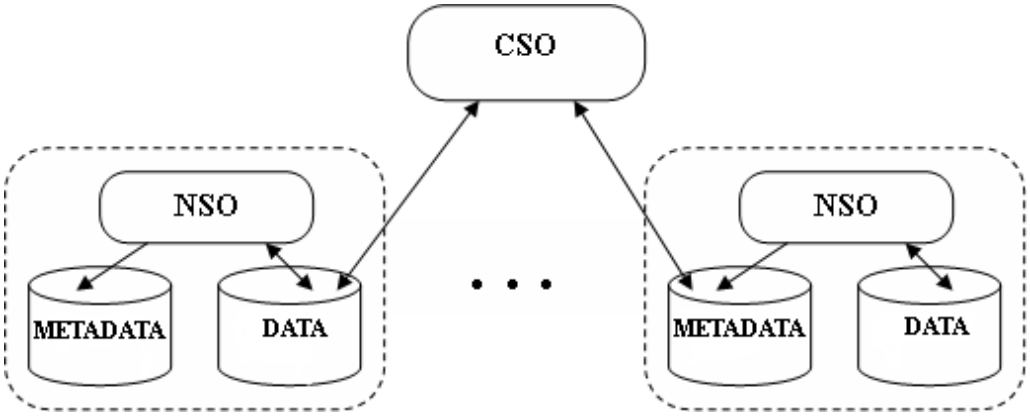

Fig. 1. Logical system structure
Rys. 1. Logiczna struktura systemu

CSOs are responsible for executing basic SQL queries and some part of enhanced SQL syntax, for example synchronizing the global discretization range (for all local models which are combined into a global one, ranges used for discretization must be the same). Data mining process is conducted by NSOs which are responsible for building, testing and applying local models and combining them into global models. System metadata is used for enhanced SQL queries. Information stored in metadata encompasses methods of discretization, missing values complement and model building and combining. Built data models are also stored in metadata.

3. Metadata Structure

Presented metadata structure was designed by the author from ground up placing emphasis on enabling to store information for distributed data mining models built by means of enhanced SQL. System metadata can be divided as follows:

- Primal (basic) metadata – placed on each system's node, completed with information about the implemented data mining algorithms and combining algorithms.
- Derivative metadata – stores information about the built models, testing and applying models results. This metadata can be divided into following categories:
  - local – for storing the locally build models,
  - related – the local models transferred from the other nodes,
  - global – combined (global) models and global additional sets.
- Work (auxiliary) metadata – stores any necessary temporary information.

Created data mining models are also stored in metadata – they constitute a special part of the metadata repository called knowledge about data.

Operators access metadata, get information from metadata and also modify or insert information. Main client side operators are:

- Model metadata operator – inserts model metadata (model name, train set, chosen algorithm etc.) on the basis of a data mining query.
- Discretization range synchronization operator – determines the global minimum and maximum values for numerical attributes.
- Values lists synchronization operator – determines global valid values lists for all enumerated attributes.

Node side operators are:

- Discretization operator – responsible for continuous value discretization with global ranges.
- Building local model operator (using a given algorithm) and all necessary suboperators.
- Combining model (using a given algorithm) operator and all necessary suboperators.
- Testing local/global model operator and all necessary suboperators.
- Applying local/global model operator and all necessary suboperators.

Detailed description of the operators and suboperators is beyond the scope of this article. Here is an example how the model metadata operator, building local model operator and combining model operator affect metadata. SQL queries presented in Fig. 2 return basic information inserted into metadata during the local models building and combining into a global model. Retrieved information is shown in fig. 3.

```
SELECT model_id as global_model, operator, object_id, basic_rule_set,
alg_id as comb_alg, target, target_vals as target_values
FROM ddm_gl_models
WHERE model_id = 52;

SELECT local_model_id as local_models, operator,
data_set as local_data_sets, alg_id as alg, object_id, target,
target_vals as local_target_values
FROM ddm_gl_model_locals lm, ddm_models m
WHERE lm.model_id = 52 and lm.local_model_id = m.model_id;
```

Fig. 2. SQL queries for retrieving a basic model information
Rys. 2. Zapytania SQL zwracające podstawowe informacje o modelach

```
GLOBAL_MODEL OPERATOR OBJECT_ID BASIC_RULE_SET COMB_ALG TARGET  TARGET_VALUES
------------ -------- --------- -------------- -------- ------- -------------
          52 classify car_id    ddm_brs_gl_52  2        class_n 1#2#3#4#

LOCAL_MODELS OPERATOR LOCAL_DATA_SETS ALG OBJECT_ID TARGET  LOCAL_TARGET_VALUES
------------ -------- --------------- --- --------- ------- -------------------
          53 classify   cars_train_1  1   car_id    class_n 3#4#
          54 classify   cars_train_2  1   car_id    class_n 3#4#
          55 classify   cars_train_3  1   car_id    class_n 1#2#4#
```

Fig. 3. Basic model information retrieved from metadata
Rys. 3. Podstawowe informacje o modelach otrzymanych z metadanych

Local classification models (in Fig 3.) were trained with three different parts of *cars_train* set (this set is a part of the UCI Machine Learning Repository [12]) using the algorithm with *id*=1 and target attribute *class_n*. Classified objects are identified by *car_id*. These models were then combined into a global model with *id*=52. The effect of combining using the algorithm with *id*=2 was basic rule set *ddm_brs_gl_52*. The above example also demonstrates effects of applying the values lists synchronization operator for the target of classification (*local_target_values* vs. *target_values*).

3.1. Primal Metadata

Primal metadata stores information about the implemented data mining algorithms, combining local models strategies and also discretization and missing values completion methods. This metadata encompasses the following tables:

- DDM_ALGORITHMS (alg_id, alg_name, operator, num_of_params, alg_impl),
- DDM_ALG_PARAMS (*alg_id*, *param_id*, *param_name*, *param_type*),
- DDM_COMB_STR(*cstr_id*, *cstr_name*, *operator*, *num_of_params_cstr_impl*),
- DDM_CSTR_PARAMS (*cstr_id*, *param_id*, *param_name*, *param_type*),
- DDM_DISC_MET (*disc_id*, *disc_name*, *num_of_params*, *disc_impl*),
- DDM_DISC_PARAMS (*disc_id*, *param_id*, *param_name*, *param_type*),
- DDM_MV_COMP (*mv_id*, *mv_name*, *num_of_params*, *mv_impl*),
- DDM_MV_PARAMS (*mv_id*, *param_id*, *param_name*, *param_type*).

Information stored in these tables encompasses ids, names and parameters of the implemented procedures. Table field ended with suffix *_impl* (implementation field) is used for

storing information how procedure, which implements a given operator, must be called. This information has a form of a pattern allowing the correct procedure call and describing its parameters. Presented system was implemented with Oracle database. Operators procedures were implemented as PL/SQL stored procedures. In Oracle database operators can also be implemented as Java stored procedures. The implementation field allows using different databases as systems nodes (system can be heterogeneous), provided the correct call pattern is set. The implementation field format is presented in fig. 4.

```
#N:<full_procedure_name>({[#P<consecutive_number>:<parameter_name>,]|
                          [#D<consecutive_number>:<parameter_call_value>,]})
```

Fig. 4. Implementation field format
Rys. 4. Format pola implementacji

Here is an example how the classification algorithm ID3 is described in the primal metadata. Fig.5 shows a SQL query which returns the algorithm description.

```
SELECT * FROM ddm_algorithms WHERE alg_name = 'ID3'

ALG_ID ALG_NAME OPERATOR NUM_OF_PARAMS ALG_IMPL
------ -------- -------- ------------- ----------------------------------------
---
     1 ID3      CLASSIFY             0 #N:ddm_classify.buildTree(#P1:schema,
#P2:modelType, #P3:modelId, #D1:1,#D2:1)
```

Fig. 5. Retrieving an algorithm description
Rys. 5. Uzyskiwanie informacji o algorytmie

Information stored in the *alg_impl* field (see Fig.5) means that to use classification algorithm ID3 one needs to call *ddm_classify.buildTree* procedure with five parameters. First three must be given at the time of calling (they are set automatically without user interaction). Last two have to be set to 1 (these two parameters are *level_id* and *node_id – buildTree* procedure is recursive and these values are initial values). The ID3 algorithm itself has no parameters [13] (*num_of_params*=0).

3.2. Derivative Metadata

Derivative metadata stores knowledge about data in a form of classification rules, decision trees or cluster histograms. This knowledge is an effect of building the data models using data mining and combining the algorithms implemented in the system.

Figure 6 presents a basic structure of metadata which stores information about built global models. This information encompasses:

- basic description of a global model – DDM_GL_MODELS table,
- dimensions of an analysis space – DDM_GL_MODELS_DIMS table,
- local models which were combined into the global one – DDM_GL_MODEL_LOCALS table,

- method used for combining the local models – DDM_GL_COMB table,
- results of testing and the confusion matrix for the classification models – DDM_GL_MODEL_TESTS and DDM_GL_CONF_MTX tables.

Local models can be combined into a global model with different strategies. Each global model created in the system is stored in a separate table called global model form table (GMFT).

GMFT has defined structure and is created as a first step of building global model process. Names of these tables have constant prefix and variable suffix. The suffix describes the model identifier and chosen combining strategy.

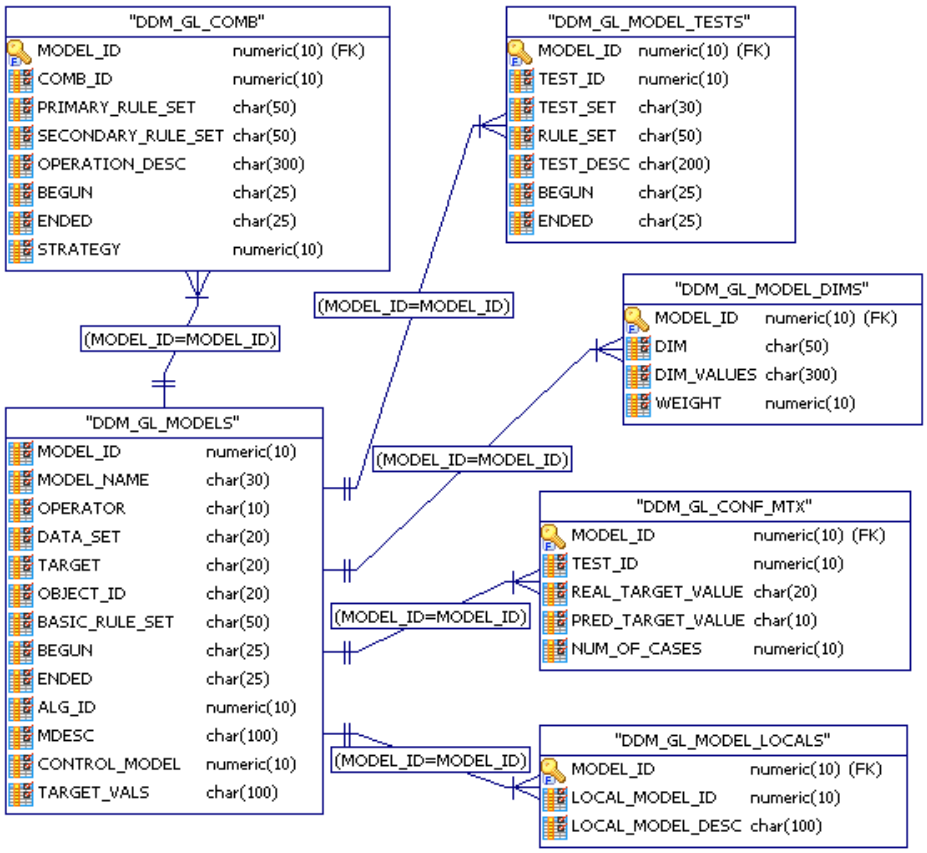

Fig. 6. Basic structure of derivative metadata for the global models
Rys. 6. Struktura podstawowych metadanych wynikowych dla modeli globalnych

For classification model GMFT has prefix DDM_BRS_GL. BRS means basic rule set – the simplest strategy for combining classification models implemented in the system joins rules from the local models into one set [7 and 8] called basic rule set. Classification GMFT has a structure designed for rules storing. Basic fields of the table are: *rule_no* – identifier of a rule in a global model, *model_id* – rule local model, *rule_id* – rule identifier in a local model, *target_value* – conclusion of a rule, *attrib* – attribute used in a rule's premise, *in_values* – values of an attribute used in a rule's premise.

A rule can be show using PL/SQL procedure. Fig. 7 presents *showClassificationRule* procedure call.

```
ddm_get_model_info.showClassificationRule(72,1)

rule no: 1 from model 72
IF DOORS IN (2,3) LUG_BOOT IN (med) BUYING IN (med) MAINT IN (low) PERSONS IN (4)
SAFETY IN (high) THEN class_ch = good
```

Fig. 7. Retrieving classification rule description with a stored procedure
Rys. 7. Uzyskiwanie informacji o regule klasyfikacji z wykorzystaniem procedury składowanej

Here is an example of classification GMFTs created for a model with *id*=1:

- DDM_BRS_GL_1 – basic rule set,
- DDM_BRS_GL_1_WSR – basic rule set devoid of subrules [7 and 8],
- DDM_BRS_GL_1_ASR – basic rule set where absorption of the rules was conducted [7 and 8]; this GMFT has an additional field *absorbed* which stores the number of rules absorbed by a given rule.

During a global classification model testing some auxiliary tables are created. For example the tables with DDM_CL_T_GL prefix. These tables contain results of applying a classification model on a test set (sets). The results of testing encompasses the value of class determined by a model (*pred_class_value*), identifier of a rule (rules) which classified object into a class (*rules*) and object real class value (*real_class_value*). On the basis of this information the data confusion matrix and model accuracy are calculated.

```
select attrib, attrib_val, round(perc,2)||'%' as perc from dm_gl_cl_desc_1_CTR
where node_id = 12513
order by attrib, attrib_val

ATTRIB   ATTRIB_VAL    PERC
-------- ------------ -------------
BUYING   HIGH         0%                  MAINT    HIGH         23,44%
BUYING   LOW          100%                MAINT    LOW          34,38%
BUYING   MED          0%                  MAINT    MED          26,56%
BUYING   VHIGH        0%                  MAINT    VHIGH        15,63%
DOORS    2            0%                  PERSONS  2            21,88%
DOORS    3            0%                  PERSONS  4            42,19%
DOORS    4            60,94%              PERSONS  MORE         35,94%
DOORS    5MORE        39,06%              SAFETY   HIGH         37,5%
LUG_BOOT BIG          40,63%              SAFETY   LOW          32,81%
LUG_BOOT MED          31,25%              SAFETY   MED          29,69%
LUG_BOOT SMALL        28,13%
```

Fig. 8. Retrieving a cluster description
Rys. 8. Uzyskiwanie opisu o skupieniu

For clustering model GMFT has prefix DDM_GL_CL_DESC. Suffix describes model identifier and chosen combining strategy [9]: _CTR, _MAX, _MIN, _MF, _EBM, _EFM. Global clustering models have a form of a probabilistic description. Fig. 8 presents SQL query which returns a percentage of attribute values for one cluster.

Classification models have a form of decision trees (DDM_DECISION_TREE table) and a set of rules generated from these trees (DDM_CLASS_RULES and DDM_CLASS_BODIES tables). SQL query presented in fig. 9 shows one of the rules for model no. 53.

```
select cr.model_id, cr.rule_id, target_value, attrib, in_values from
ddm_class_rules cr, ddm_class_rules_bodies crb
where cr.model_id = crb.model_id and cr.rule_id = crb.rule_id and cr.model_id=53
and cr.rule_id=1;

MODEL_ID RULE_ID TARGET_VAL ATTRIB   IN_VALUES
-------- ------- ---------- -------- ---------
      53       1 acc        DOORS    2#
      53       1 acc        LUG_BOOT big#med#
      53       1 acc        MAINT    low#
      53       1 acc        PERSONS  more#
      53       1 acc        SAFETY   high#
```

Fig. 9. Retrieving classification rule description with a SQL query
Rys. 9. Uzyskiwanie informacji o regule klasyfikacji zapytaniem

Figure 10 shows SQL query for retrieve decision tree for model no. 53. Fragment of this tree is also presented.

```
select (LPAD(' ',2*(LEVEL_ID-1))||level_id) as tree_level , (LPAD('
',2*(LEVEL_id-1))||attrib||' = '||attrib_value||' -> '||target_value) as
node_condition
from ddm_decision_tree where model_id=53
start with node_id=1 connect by prior node_id=parent

TREE_LEVEL NODE_CONDITION
------------------------
1 ROOT = - ->
  2 MAINT = high -> unacc
  2 MAINT = low ->
    3 SAFETY = high ->
      4 PERSONS = 2 -> unacc
      4 PERSONS = 4 -> acc
      4 PERSONS = more ->
        5 DOORS = 2 ->
          6 LUG_BOOT = big -> acc
          6 LUG_BOOT = med -> acc
```

Fig. 10. Retrieving decision tree with a SQL query
Rys. 10. Uzyskiwanie drzewa decyzyjnego zapytaniem SQL

Figure 11 shows how to retrieve rules with "*SAFETY = high*" condition in rule's premise.

```
select cr.rule_id, target_value
from ddm_class_rules cr, ddm_class_rules_bodies crb
where cr.model_id = crb.model_id and cr.rule_id = crb.rule_id and cr.model_id=53
and crb.attrib like 'SAFETY' and crb.in_values like 'high%';

RULE_ID TARGET_VAL
---------- ----------
         1 3
         2 4
         7 3
         8 4
        13 3
        17 3
```

Fig. 11. Retrieving classification rules with a given attribute value in the premise
Rys. 11. Uzyskiwanie reguł klasyfikacji z konkretną wartością atrybutu w przesłance

Retrieving information about rules, decision trees or clusters using SQL queries requires knowledge about metadata structure. Using only SQL queries seems to be complicated. To make querying metadata easier additional stored procedures (example in fig. 7) and views were created. Figure 12 shows a view creation (which is automated) and using – result is the same as in fig. 10.

```
create view dt_model_53 as select (LPAD(' ',2*(LEVEL_ID-1))||level_id) as
tree_level , (LPAD(' ',2*(LEVEL_id-1))||attrib||' = '||attrib_value||' ->
'||target_value) as node_condition
from ddm_decision_tree_53 where model_id=53
start with node_id=1 connect by prior node_id=parent;

select * from dt_model_53;
```

Fig. 12. Retrieving decision tree with a simple SQL query
Rys. 12. Uzyskiwanie drzewa decyzyjnego prostym zapytaniem SQL

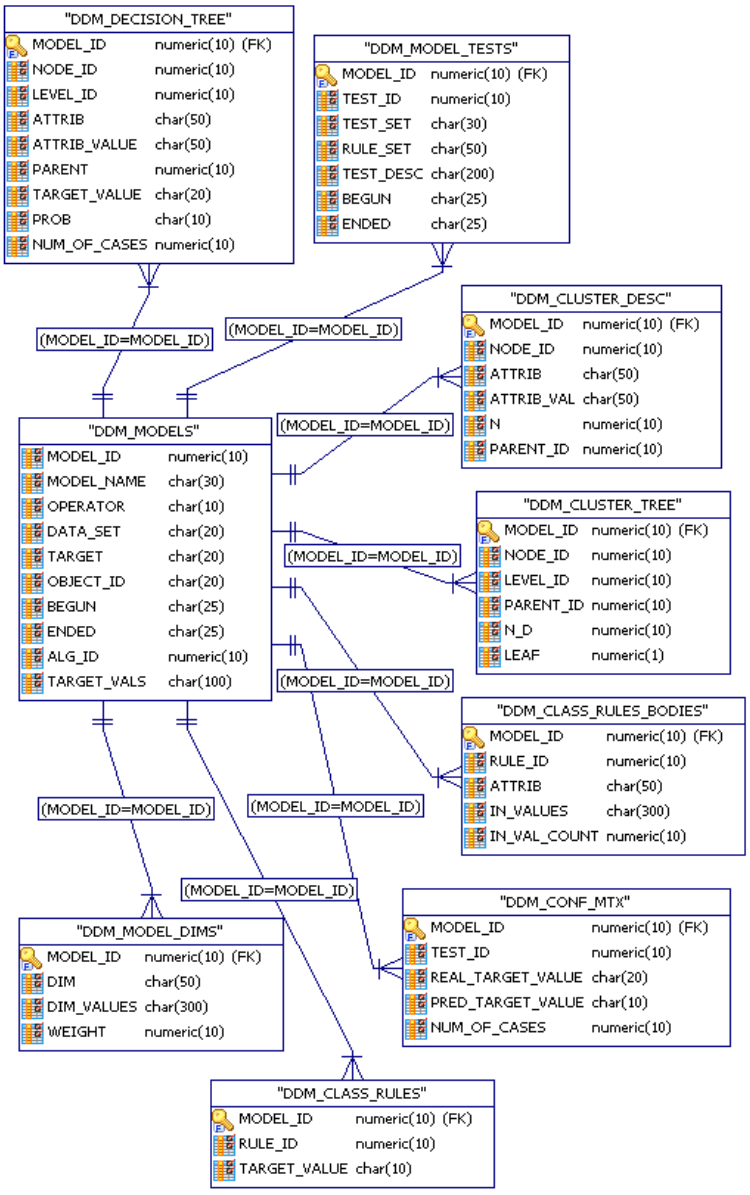

Fig. 13. Local metadata schema
Rys. 13. Lokalne metadane systemu

Figure 13 presents metadata for storing basic information about the local models and the models themselves. The local clustering models are also stored in a form of trees (DDM_CLUSTER_TREE table) and probabilistic cluster descriptions (DDM_CLUSTER_DESC table).

4. Conclusions

The basic assumption established during the system development was usage of one standard method of creating and querying data and metadata (knowledge). Presented metadata stores information in a form of database tables. Structure of metadata is easy to create with SQL scripts on any SQL database. Metadata can be queried with SQL.

The proposed metadata structure is not final. At present, it allows to store classification rules, decision trees, clusters trees and probabilistic cluster description, so it has to be complemented as new model building algorithms will be implemented in the system. During the system tests it turned out that storing rules and decision trees of local models in single table causes fast table size growth. The tables become difficult to query because of the time needed for query execution. One solution is to create adequate indices for these tables. Another one is to use separate local model form tables (LMFT). This solution was tested with global models and turned out to be useful and so far efficient.

The system needs more tests and implementation of other data mining algorithms but can be a good base for building heterogeneous, distributed data mining system.

BIBLIOGRAPHY

1. Kimball R., Reeves L., Margy R., Warren T.: The Data Warehouse Lifecycle Toolkit. Wiley & Sons, 1998.
2. Longley P. A., Goodchild M. F., Maguire D. J., Rhind D. W.: Geographic Information Systems and Science. John Wiley & Sons Ltd, 2005.
3. Ma Y., Yu S., Li D., Liu L.: Implementation of Metadata Warehouse Used in a Distributed Data Mining Tool. International Conference on Challenges in Environmental Science and Computer Engineering, 2010, Vol. 2, p. 343÷346.
4. Le-Khac N. A., Aouad L. M., Kechadi M. T.: An efficient Knowledge Management Tool for Distributed Data Mining. International Journal of Computational Intelligence Research, 2009, Vol. 5, No. 1.

5. Nesbit J. C., Adesope O. O.: Learning With Concept and Knowledge Maps: A Meta-Analysis. Review of Educational Research, Vol. 76, No. 3, 2006, p. 413÷448.
6. Mastroianni C., Talia D. Trunfio P.: Metadata for Managing Grid Resources in Data Mining Applications. Journal of Grid Computing, Vol. 2, No. 1, 2004, p. 85÷102.
7. Gorawski M., Płuciennik E.: Analytical Models Combining Methodology with Classification Model Example. First International Conference on Information Technology, 2008.
8. Gorawski M., Pluciennik-Psota E.: Distributed Data Mining Methodology with Classification Model Example. Lecture Notes in Artificial Intelligence (eds.): Nguyen N. T., Kowalczyk R. Chen S.M. ICCCI 2009, LNCS Vol. 5796, XVII, p. 107÷117.
9. Gorawski M., Płuciennik-Psota E.: Distributed Data Mining Methodology for Clustering and Classification Model. Rutkowski L. et al. (eds.): ICAISC 2010, Part I, LNAI 6113, Springer-Verlag Berlin Heidelberg 2010, p. 323÷330.
10. Gorawski M., Płuciennik E.: Distributed SQL in Spatial Data Warehouse and Software Agents Environment. International Transactions on Systems Science and Applications. 2008, Vol. 3, No. 4, p. 307÷313.
11. Gorawski M., Płuciennik E.: Distributed Data Mining by Means of SQL Enhancement. OTM Workshops 2008, ODBASE International Conference, 2008, p. 34÷35.
12. Asuncion A., Newman D. J.: UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science, http://www.ics.uci.edu/~mlearn /MLRepository.html (2007).
13. Quinlan R.: Induction of Decision Trees. Machine Learning, vol. 1, 1986, p. 81÷106.

Recenzenci: Prof. dr hab. inż. Tadeusz Morzy
Dr inż. Paweł Sitek

Wpłynęło do Redakcji 16 stycznia 2011 r.

Omówienie

Rozproszone przetwarzanie oraz eksploracja danych są obecnie jednymi z najpopularniejszych technik obróbki danych. Metadane można zdefiniować jako „dane o danych”. Są one również szeroko stosowane we współczesnych systemach przechowywania danych. Pomagają opisać strukturalnie przechowywane informacje. Ułatwiają dostęp do informacji oraz pozwalają lepiej ją zrozumieć. Metadane mogą być przechowywane w dowolnej formie: plików tekstowych, XML, tabel bazy danych itp.

Artykuł opisuje metadane wykorzystane w systemie rozproszonej eksploracji danych, opartym na scalaniu modeli lokalnych, powstających w rozproszonych węzłach systemu, w modele globalne. Zarówno metadane, jak i dane systemu są przechowywane w tabelach bazy danych. Dostęp do nich jest realizowany z poziomu języka SQL. Metadane obejmują metadane pierwotne, wynikowe, robocze, a także testowe dla modeli lokalnych oraz globalnych. Na poszczególne węzły systemu składają się dane lokalne, metadane oraz operatory strony węzła. Zapytania (również eksploracyjne z rozszerzoną składnią SQL) są zadawane do rozproszonej struktury poprzez aplikację kliencką, w której zostały zaimplementowane operatory strony klienta. Operatory strony klienta są odpowiedzialne za realizację podstawowej składni zapytania SQL oraz za część elementów składni rozszerzonej. Zaawansowana analiza danych jest realizowana przez operatory strony węzła odpowiedzialne za budowę, testowanie czy stosowanie modeli lokalnych oraz ich scalanie w model globalny.

Metadane systemu mają zastosowanie w przypadku realizowania zapytań SQL o rozszerzonej składni. Przechowują one dane dotyczące zaimplementowanych metod obróbki danych (m.in. dyskretyzacji czy budowy modelu) oraz zbudowane w ramach eksploracji modele. Na metadane systemu składają się: metadane pierwotne (podstawowe) przechowujące informacje o zaimplementowanych algorytmach eksploracji oraz operatorach, które je realizują; metadane bieżącej analizy (wtórne), przechowujące informacje o zbudowanych modelach, wynikach ich testów i stosowania; metadane robocze, przechowujące niezbędne informacje tymczasowe. Metadane wtórne można podzielić na: lokalne (modele lokalne), zaprzyjaźnione (modele z innych węzłów systemu), globalne (scalone modele oraz globalne zbiory pomocnicze).

Address

Ewa PŁUCIENNIK-PSOTA: Silesian University of Technology, Institute of Computer Science, Akademicka 16, 44-100 Gliwice, Poland, Ewa.Pluciennik@polsl.pl.