Final Committee Draft ISO/IEC FCD	
Date: **2005-05-11**	Reference number: ISO/JTC 1/SC 32**N1310**
Supersedes document SC 32N1155	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 32 Data Management and Interchange Secretariat: USA (ANSI)	Circulated to P- and O-members, and to technical committees and organizations in liaison for voting (P-members only) by: **2005-09-12** Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated.

ISO/IEC FCD 13249-6:200x(E)

Title: Information technology - Database languages - SQL Multimedia and Application Packages - Part 6: Data Mining 2nd ed

Project: 1.32.04.02.06.00

Introductory note: The attached document is hereby submitted for a four-month letter ballot to the National Bodies of ISO/IEC JTC 1/SC 32. The ballot starts 2005-05-11.

Medium: E

No. of pages: 305

Address Reply to: Douglas Mann, Secretary, ISO/IEC JTC 1/SC 32

Farance Inc *, 360 Pelissier Lake Road, Marquette, MI 49855-9678, United States of America

Telephone: +1 906-249-9275; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/

Final Committe Draft
Ballot 2nd Edition

ISO/IEC JTC 1/SC 32 N 1310

Date: 2005-05-09

ISO/IEC 13249-6:2005(E)

ISO/IEC JTC 1/SC 32/WG 4

Secretariat: U.S.A.

Information technology — Database languages — SQL Multimedia and Application Packages — Part 6: Data Mining

Document type: Final Committee Draft
Document subtype: Not applicable
Document stage: (00) Preliminary
Document language: E

Blank page

Contents Page

Figures

Page

Tables

Page

Blank page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 13249 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 13249-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32*, Data management services*.

ISO/IEC 13249 consists of the following parts, under the general title *Information technology — Database languages — SQL Multimedia and Application Packages*:

— *Part 1: Framework*

— *Part 2: Full-text*

— *Part 3: Spatial*

— *Part 5: Still Image*

— *Part 6: Data Mining*

Annexes A to C of this part of ISO/IEC 13249 are for information only.

Blank page

Introduction

The purpose of this International Standard is to define multimedia and application specific types and their associated routines using the user-defined features in ISO/IEC 9075.

SQL/MM is structured as a multi-part standard. At present it consists of the following parts:

Part 1: Framework

Part 2: Full-Text

Part 3: Spatial

Part 5: Still Image

Part 6: Data Mining

The organization of this part of ISO/IEC 13249 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 13249, constitute provisions of this part of ISO/IEC 13249.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.

4) Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.

5) Clause 5, “Data Mining Data Types”, defines the user-defined types and associated routines for the data definitions of data mining.

6) Clause 6 “Association Rules”, defines the user-defined types and associated routines for the search for association rules.

7) Clause 7, “Clustering”, defines the user-defined types and associated routines for clustering.

8) Clause 8, “Classification”, defines the user-defined types and associated routines for classification.

9) Clause 9, “Regression”, defines the user-defined types and associated routines for regression.

10) Clause 10, “Status Codes”, defines the SQLSTATE codes used in this part of ISO/IEC 13249.

11) Clause 11, “Conformance“, defines the criteria for conformance to this part of ISO/IEC 13249.

A) Annex A, “Implementation-defined elements”, is an informative annex. It lists those features for which the body of this part of ISO/IEC 13249 states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementor shall provide in each case.

B) Annex B, “Implementation-dependent elements”, is an informative annex. It lists those features for which the body of this part of ISO/IEC 13249 states explicitly that the meaning or effect on the database is implementation-dependent.

C) Annex C, “A scenario using the classification technique”, is an informative annex. It presents explanatory material on how to use this part of 13249.

Blank page

Information technology — Database languages — SQL Multimedia and Application Packages —

Part 6: Data Mining

1 Scope

This part of ISO/IEC 13249:

a) introduces the Data Mining part of ISO/IEC 13249,

b) gives the references necessary for this part of ISO/IEC 13249,

c) defines notations and conventions specific to this part of ISO/IEC 13249,

d) defines concepts specific to this part of ISO/IEC 13249,

e) defines data mining user-defined types and their associated routines.

The data mining user-defined types defined in this part adhere to the following:

— A data mining user-defined type is generic to data mining data handling. It addresses the need to store, manage and retrieve information based on elements such as data mining models, data mining settings, and data mining test results.

— A data mining user-defined type does not redefine the database language SQL directly or in combination with another data mining data type.

Blank page

2 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of ISO/IEC 13249. For dated references, subsequent amendments to, or revisions of any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 13249 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative documents referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

2.1 International standards

ISO/IEC 13249-1:2006, Information technology — Database languages — SQL Multimedia and Application Packages — Part 1: Framework.

2.2 Publicly-available specifications

Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004, http://www.w3.org/TR/REC-xml-2004-02-04.

Predictive Model Markup Language (PMML) 3.0, http://www.dmg.org/pmml-v3-0.html

Blank page

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 1

This part of ISO/IEC 13249 makes use of all terms defined in part 1 of ISO/IEC 13249.

3.1.2 Definitions provided in Part 6

For the purposes of this part of ISO/IEC 13249, the following definitions apply.

3.1.2.1
application phase
Phase, during which input rows are evaluated against a data mining model and one or more values are computed, for example the identification of the cluster for data mining clustering, the predicted value for data mining classification or data mining regression and inferred items for association rules.

3.1.2.2
association rule
Relationship between two sets of items, which appear frequently (according to a specified threshold), i.e. there is a significant number of groups containing the combination of the values, describing how the presence of one of the sets (the rule head) in a group depends on the presence of the other set (the rule body).
Note: In this document, the term “rule” is sometimes used as a synonym for “association rule” where the meaning is unambiguous from the context.

3.1.2.3
association rule body
The set of conditions in an association rule. The body of the association rule “Z, if X and Y” is {X, Y}.
Note: In this document, the term “rule body” is sometimes used as a synonym for “association rule body” where the meaning is unambiguous from the context.

3.1.2.4
association rule confidence
The percentage of transactions that contain all items of the rule out of those transactions that contain the rule body.
Note: The confidence of the association rule “X and Y implies Z” is 100 times the number of transactions that contain X, Y and Z divided by the number of transactions that contain X and Y. In this document, the terms “rule confidence” or “confidence” are sometimes used as synonyms for “association rule confidence” where the meaning is unambiguous from the context.

3.1.2.5
association rule filter
A specification defining how to compute a subset of a given set of association rules.
Note: In this document, the terms “rule filter” or “filter” are sometimes used as synonyms for “association rule filter” where the meaning is unambiguous from the context.

3.1.2.6
association rule head
The implication of an association rule. The head of the association rule “Z, if X and Y” is Z.
Note: In this document, the terms “rule head” or “head” are sometimes used as synonyms for “association rule head” where the meaning is unambiguous from the context.

3.1.2.7
association rule item constraint
A restriction of the contents of an association rule.
Such a restriction can either mandate or bar the presence of a particular item in an association rule. A restriction may apply to the complete rule or only to its body or its head.
Note: In this document, the term “item constraint” is sometimes used as a synonym for “association rule item constraint” where the meaning is unambiguous from the context.

3.1.2.8
association rule lift
A property of an association rule measuring how much more likely the rule occurs when the rule body is present than statistically expected.
Note: In this document, the terms “rule lift” or “lift” are sometimes used as synonyms for “association rule lift” where the meaning is unambiguous from the context.

3.1.2.9
association rule model
The result of data mining association rule discovery, which is a set of association rules.
Note: In this document, the term “rule model” is sometimes used as a synonym for “association rule model” where the meaning is unambiguous from the context.

3.1.2.10
association rule support
The percentage of transactions that contain all items of an association rule.
Note: In this document, the terms “rule support” or “support” are sometimes used as synonyms for “association rule support” where the meaning is unambiguous from the context. The association rule support of the association rule “X and Y implies Z” is 100 times the number of transactions that contain X, Y and Z divided by the total number of transactions.

3.1.2.11
categorical field type
A field being of categorical field type supports only the <equal operator> as a comparison operator.
Note: There is no defined order, and no arithmetic operations are supported.

3.1.2.12
class label
Target field used in data mining classification to create a classification model, whose application to data without a class label allows it to predict a value for this class label.
Note: The class label shall have a categorical field type.

3.1.2.13
classification cost rate
Maximal classification error tolerated during the validation phase of data mining classification, above which the classification model is considered not good enough.

3.1.2.14
classification error
Percentage of wrongly predicted values among the total number of values predicted during the validation phase of the classification technique.
Note: This error is returned at the end of the test phase.

3.1.2.15
classification model
The result of data mining classification.

3.1.2.16
cluster homogeneity
The property of a cluster describing how similar (on a scale between 0 and 1) the records in a cluster are.
Note: This is a measure of the degree of similarity between the records of a cluster according to the criteria for similarity between records.

3.1.2.17
clustering model
The result of data mining clustering.

3.1.2.18
data mining application task
Abstraction for the complete data needed to apply a data mining model, containing values of the data mining model, the data mining data for input and output, and the result field used for the operation.

3.1.2.19
data mining association rule discovery
A data mining technique that searches for common occurrences of different items in data.
Note: In this document, the terms “association rule discovery” or “rule discovery” are sometimes used as synonyms for “data mining association rule discovery” where the meaning is unambiguous from the context.
Example of usage: store layout.

3.1.2.20
data mining classification
Data mining technique that computes a classification model such that the class label value can be predicted by using the model and input field values without the class label value.
Note: In this document, the term “classification” is sometimes used as a synonym for “data mining classification” where the meaning is unambiguous from the context.
Example of usage: insurance risk prediction.

3.1.2.21
data mining cluster
A set of input rows with common characteristics (see data mining clustering).
Note: In this document, the term “cluster” is sometimes used as a synonym for “data mining cluster” where the meaning is unambiguous from the context.

3.1.2.22
data mining clustering
A data mining technique that discovers sets of input rows with common characteristics - the *data mining clusters*, so that rows are as homogeneous as possible inside a data mining cluster and as heterogeneous as possible between two data mining clusters.
Note: In this document, the term “clustering” is sometimes used as a synonym for “data mining clustering” where the meaning is unambiguous from the context.
Example of usage: customer mailings.

3.1.2.23
data mining data
Description of data contained in tables that represents the metadata required for access to the data for data mining training, test or application runs.

3.1.2.24
data mining data type
A data mining data type represents an abstraction of a table. A data mining data type maps the input field names (i.e. column names of the table) to alias names. These alias names will be compared, before the data mining run, to the field names used in the specified logical data specification.

3.1.2.25
data mining model
The result of a data mining run of a data mining technique over a given set of data, which contains the usable correlations discovered in the data.
Note: The data mining model can also be used alone as a substitute of the original large amount of data to associate, classify or predict the behavior of additional data (in application and testing phases) .
Note: In this document, the terms “mining model” or “model” are sometimes used as a synonym for “data mining model” where the meaning is unambiguous from the context.

3.1.2.26
data mining regression
Data mining technique similar to data mining classification except for the numerical type of the target field, which computes a regression model allowing to predict a numerical value.
Note: The predicted value might not be identical with any value contained in the data used to build the model.
Note: In this document, the term “regression” is sometimes used as a synonym for “data mining regression” where the meaning is unambiguous from the context.
Example of usage: customer ranking.

3.1.2.27
data mining run
Computation of a data mining model using data mining settings.
Note: According to the data mining technique, a data mining run may contain a training phase, a validation phase and a testing phase. A data mining run is called a training run when it contains a training phase, and a test run when it contains only a test phase.

3.1.2.28
data mining segmentation
synonym for data mining clustering.
Note: Data mining clustering is the term used in this part of ISO/IEC 13249.

3.1.2.29
data mining sequence
Combination of field values that appears frequently in the same order in the input data (according to a specified threshold), i.e. there is a significant number of groups containing the combination of the values.
Note: In this document, the term “sequence” is sometimes used as a synonym for “data mining sequence” where the meaning is unambiguous from the context.

3.1.2.30
data mining sequence rule
Combination of field values that appears frequently in the input data (according to a specified threshold), where two data mining sequences are related such that the occurrence of the first influences the frequency of occurrence of the second.
Note: In this document, the terms “sequence rule” or “rule” are sometimes used as synonyms for “data mining sequence rule” where the meaning is unambiguous from the context.

3.1.2.31
data mining sequence rule discovery
A data mining technique similar to association rule discovery that searches for repeated joint occurrences of items in data and correlations between these items.
Note: In this document, the terms “sequence rule discovery” or “sequence discovery” are sometimes used as synonyms for “data mining sequence rule discovery” where the meaning is unambiguous from the context.
Example of usage: warranty and quality management.

3.1.2.32
data mining settings
Settings used in the data mining run that specify the logical data specification and parameters depending on the data mining technique.
Note: In this document, the terms “mining settings” or “settings” are sometimes used as a synonym for “data mining settings” where the meaning is unambiguous from the context.

3.1.2.33
data mining task
Abstraction for the complete data needed to compute, test or apply a data mining model, containing the data mining settings and the data mining data type.
Note: In this document, the terms “mining task” or “task” are sometimes used as a synonym for “data mining task” where the meaning is unambiguous from the context.

3.1.2.34
data mining technique
Four types of data mining techniques are distinguished: data mining association rule discovery, data mining clustering, data mining classification and data mining regression.

3.1.2.35
data mining test result
Statistical result returned at the end of the testing phase that contains an evaluation of the correctly and incorrectly predicted values in the test data.
Note: This result can be considered a quality criterion of the data mining model.
Note: In this document, the term “test result” is sometimes used as a synonym for “data mining test result” where the meaning is unambiguous from the context.

3.1.2.36
data mining test task
Abstraction for the complete data needed to test a data mining model, containing values of the data mining model and the data mining data used for the operation.
Note: In this document, the term “test task” is sometimes used as a synonym for “data mining test task” where the meaning is unambiguous from the context.

3.1.2.37
data warehouse
A collection of data for decision support.

3.1.2.38
export
To transform the value of a user-defined type into a specific format of data.

3.1.2.39
gains chart
A graph showing the ranking quality of a classification or regression model.

3.1.2.40
import
To generate a value of a user-defined type from a specific format of data.

3.1.2.41
input field
Field of the row type of an input row.

3.1.2.42
input field importance
The importance of the values of a field in the training data for a data mining model.

3.1.2.43
input row
A row used as input for a data mining run or an application of a data mining model.
Note: The expected row type for a data mining run is determined by the logical data specification of the settings value or –in case of a test run– by the logical data specification and the target field of the data mining model. It is determined by the logical data specification of the data mining model.

3.1.2.44
logical data specification
Part of the data mining settings that lists field names and field types to be used during the data mining run.
Note: These fields will be compared before the data mining run to the field names or aliases defined in the data mining data type.

3.1.2.45
logical field
A placeholder for an input field whose properties are given by the methods of the logical data specification type (DM_LogicalDataSpec).

3.1.2.46
logical field type
Type of a logical field, either categorical or numerical, that defines the way the data mining field values will be compared and managed during the data mining run.
Note: In this document, the term “field type” is sometimes used as a shorthand for “logical field type” where the meaning is unambiguous from the context.

3.1.2.47
numerical field type
A field being of numerical field type supports all operations of <numerical value expression>s.

3.1.2.48
predicted value
The value of the target field predicted using a data mining classification or regression model.
Note: The predicted value can either be a categorical or numerical value.

3.1.2.49
ranking quality
A measure of a classification or regression model's capability to correctly order records based on the predicted value of their target fields.

3.1.2.50
regression model
The result of data mining regression.

3.1.2.51
root mean squared error
A measure of the predictive accuracy of a regression model. It is the square root of the mean of the squared differences of the actual values and the predicted values.

3.1.2.52
rule model
An association rule model or a sequence rule model.

3.1.2.53
sequence rule model
The result of data mining sequence rule discovery.
Note: In this document, the term “rule model” is sometimes used as a synonym for “sequence rule model” where the meaning is unambiguous from the context.

3.1.2.54
similarity scale
A scale for a numerical field used to compute similarities between record values for data mining clustering.
Note: The similarity scale is used to map the absolute difference between two real values to a similarity. It is 50% when the values differ by the similarity scale.

3.1.2.55
source table
A table containing the data on which data mining is done. The source table is either a base table or a view.

3.1.2.56
target field
The special field used in data mining classification and regression whose value will be predicted.

Note: In classification, the target field is sometimes also called “class label” and has a categorical field type. In regression, the target field has a numerical field type.

3.1.2.57
testing phase
Phase that reads a set of input rows containing values for the target field, evaluates each input row as in the application phase, and compares the predicted value to the actual value in the target field.
Note: The testing phase is only used for data mining classification and regression. Results of the testing phase are for example used by a data mining user or a data mining application to determine whether a model can safely be applied based on its quality.

3.1.2.58
training data
Data used as input for the training phase.

3.1.2.59
training phase
Common to all data mining techniques, this is the phase in which the data mining model is computed.

3.1.2.60
validation phase
Part of the training phase for data mining classification and regression techniques, which uses the calculated data mining model and another set of input rows to test these input rows against the data mining model, as described in the testing phase.
Note: Results of the validation phase are for example used to decide when the algorithm should terminate.

3.2 Notations

3.2.1 Notations provided in Part 1

The notations used in this part of ISO/IEC 13249 are defined in part 1 of ISO/IEC 13249.

3.2.2 Notations provided in Part 6

This part of ISO/IEC 13249 uses the prefix 'DM_' for user-defined type, attribute and SQL-invoked routine names.

This part of ISO/IEC 13249 uses the following short forms in user-defined type names and in SQL-invoked routine names:

- “appl” for “application”,
- “bld” for “build”,
- “clas” for “classification”,
- “clus” for “clustering”,
- “conf” for “confidence”,
- “dataSpec” for “logical data specification”,
- “def” for “define”,
- “diff” for “difference”,
- “exp” for “export”,
- “fld” for “field”,
- “imp” for “import”,
- “mdl” for “model”,
- “mdlSpec” for “model data specification”,
- “num” for “number of”,
- “outp” for “output”,
- “pred” for “predicted” or “prediction”,
- “reg” for “regression”,
- “seq” for “sequence”,
- “sim” for “similarity”,
- “spec” for “specification”,
- “stddev” for “standard deviation”,
- “trn” for “training”,
- “val” for “validation”.

3.3 Conventions

All conventions used defined in part 1 of ISO/IEC 13249 apply to this part of ISO/IEC 13249.

In addition, the following naming conventions for methods are introduced. These placeholders are used in the formulation:

- <verb> any action verb or an abbreviation thereof, for example “apply” or “imp”.
- <technique> abbreviation of a mining technique, “rule”, “clus”, “clas” or “reg”.
- <method> descriptor of the method
- <property> some property of a value

1) Method names shall have the form

```
DM_<verb><technique><method>,
```

where <technique> may be omitted for methods that are unique to a mining technique.

2) Individual properties are set by a methods of the form

```
DM_set<technique><property>.
```

3) Individual properties are reset by invoking the setter method with null as its property.

4) Values are added to and removed from list-valued properties using the methods

```
DM_add<technique><property>,
DM_rem<technique><property>.
```

Blank page

4 Concepts

4.1 Data mining techniques

Data mining is the process of discovering hidden, previously unknown and usable correlations in data. The data is analyzed without the necessity of any hypothesis (expected result). Data mining delivers knowledge that can be used for a better understanding of the data. Four data mining techniques have been commonly accepted so far and are widely used in applications:

1) Rule Discovery

   a) Association rule discovery:
      Given a set of purchase transactions (baskets), which contain a set of items, find rules of the form: If a purchase transaction contains item X and item Y then the purchase transaction also contains item Z in N% of all purchase transactions.

      Example application: planning of store layout.

   b) Sequence rule discovery:
      Given sets of purchase transactions (baskets), which contain sets of items, further given the customer identity for each of these transactions, find rules of the form:
      If a customer purchases item X and item Y then a later transaction by the same customer also contains item Z for N% of all customers.

      Example application: analysis of customers' buying behavior in an online store.

2) Clustering:
   Given a set of input rows with a set of fields, find sets of input rows with common characteristics. These sets are called clusters. Characterize each cluster by input field values and rank the input fields such that the most distinguishing fields come first.

   Example application: customer mailings.

3) Classification
   Given a set of input rows with a set of fields and a special field, the class label, compute a classification model such that the class label can be predicted by using the model and a set of input field values without the class label. Optimize the model such that a class label can be predicted with a minimal number of field values.

   Example application: insurance risk prediction.

4) Regression
   Regression is very similar to classification except for the type of the predicted value. Rather than predicting a class label, regression is predicting a continuous value. Hence, the predicted value might not be identical with any value contained in the data used to build the model.

   Example application: customer ranking.

4.2 Computational phases

Four major phases can be distinguished in data mining: the training phase, the model introspection phase, the test phase, and the application phase. The following subsections will briefly describe the major characteristics of these phases.

4.2.1 Training phase

The training phase is common to all data mining techniques. It is the phase in which the *data mining model* is computed.

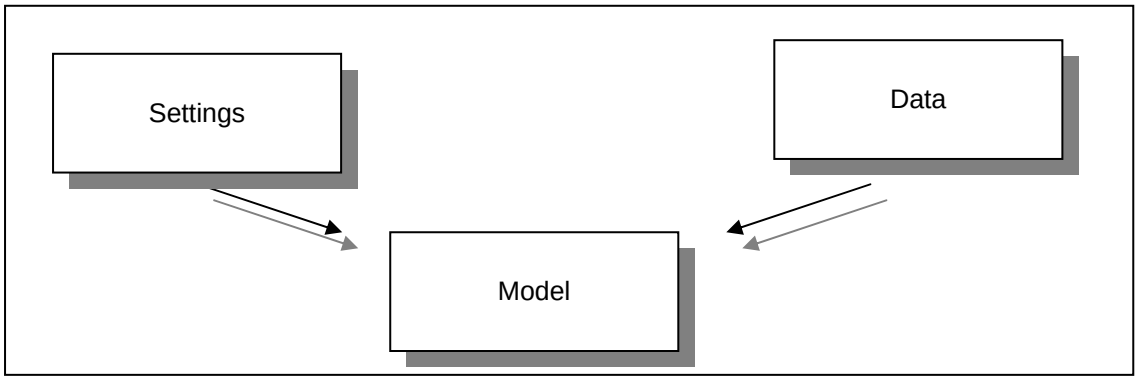

Figure 1 — The training phase

Essentially, data mining needs some *settings* and a set of input rows to compute a data mining model. However, it is important to note that usually several passes over the input data are needed before a model can be generated. Furthermore, most data mining techniques need to be able to identify fields to assign data mining related information like data mining types and special handling of a field as in case of predictive techniques.

4.2.2 Model introspection phase

The model introspection phase is common to all data mining techniques. It is the phase in which the *data mining model* is scrutinized with the goal to understand the correlations in the data that were uncovered during the training phase.

This is done for two purposes, model interpretation and model evaluation:

- Showing the regularities hidden in the data, helps to interpret the model,
- Showing statistical properties, helps to judge the quality of a model.

As an example for model interpretation, see the table below that can be generated from an association rule model using the table function DM_getRules:

Column Name:	**ID**	**HEAD NAME**	**HEAD**	**BODYID**	**LENGTH**	**BODY TEXT**	**SUPPORT**	**CONFIDENCE**	**LIFT**
Column Type:	INTEGER	CHARACTER VARYING	CHARACTER VARYING	INTEGER	INTEGER	CHARACTER VARYING	REAL	REAL	REAL
...	1	scarf	11307	1	3	jeans, hat	4.2	65.0	3.8
example rows:	2	scarf	11307	17	4	shoes, coat, hat	5.8	82.0	0.7
	3	jeans	9891	3	2	trousers	3.0	10.0	0.6
...	4	jeans	9891	17	4	shoes, coat, hat	10.5	55.0	2.0

Table 1 **— Model interpretation**

There is a row in **Table** 1 **— Model interpretation** above for each rule in the association rule model. ID is a number generated to identify the rules; HEADNAME holds the head item of the rule after application of a name mapping (if any), HEAD uses the original item before name mappings. BODYID is a generated identifier to allow joining the resulting table with a table generated by DM_getRuleBodies(). BODYTEXT is an implementation-dependent textual description of the rule body; it is provided for print-out purposes. The remaining columns further qualify the rules by their length (total number of items), as well as the support, confidence and lift values.

The value 4.2 in the SUPPORT column in the first row of table Table 1 above shows that jeans, hat and scarf are contained in 4.2% of all input transactions. The value 65.0 in the CONFIDENCE column shows that scarf occurs in 65% of all transactions that contain jeans and hat. The value 3.8 in the LIFT column shows that scarf occurs 3.8 times as often with jeans and hat as would be expected without any correlation with these items.

As an example for model evaluation consider the problem of determining the risk of a customer canceling an insurance policy. In a database of 100,000 customers, 10,000 actually left the company. A classification model is computed from these data to rank the customers by their propensity to leave. In order to evaluate the quality of the model a gains chart is created. A gains chart shows the ranking quality of a predictive model, i.e. its capability to correctly order records based on the predicted value of their target fields. The method DM_getGainsChart computes the table below from a classification model:

Column Name:	**ROWCOUNT**	**SUMACTUAL**	**THRESHOLD**
Column Type:	BIGINT	DOUBLE	DOUBLE
...	23,750	4,951	0.89
example rows:	31,250	6,408	0.83
	40,830	8,155	0.78
	51,670	8,835	0.27
	62,080	9,223	0.16
...	100,000	10,000	0.09

Table 2 **— Gains chart information**

In the example, from the 23,750 most likely to leave (as expected by the model) 4,951 actually left. If 31,250 customers are considered, 6,408 of them leave, etc. The threshold column holds the minimum confidence of belonging to this group.

Typically, these figures are presented as a chart in the following form:

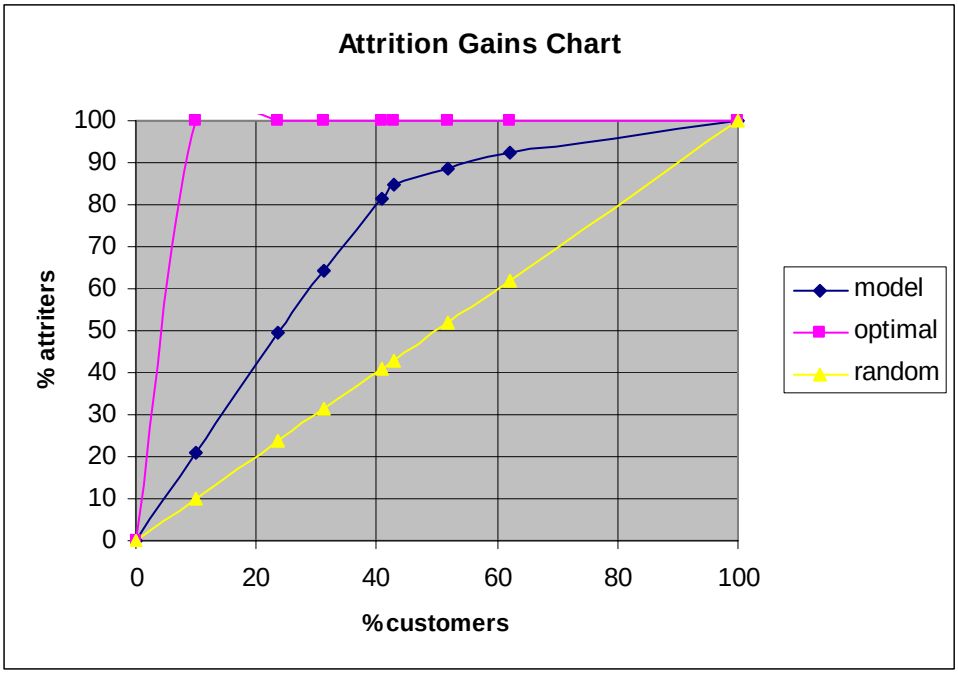

Figure 2 — Gains chart

The graphic shows, for example, that in a group of about 25% of the customers selected by the model, there are likely to be about half of the overall attriters (the customers who have left the company). In contrast, random selection would only deliver a quarter, and with perfect knowledge one would identify all of them in just 10% of the customers.

4.2.3 Application phase

During the application of a model, input rows are evaluated against a model and one or more values are computed. For instance, applying a clustering model to a customer input row would assign a cluster id and a quality value to the customer input row. In case of classification a class label and a confidence value is computed for the input row.

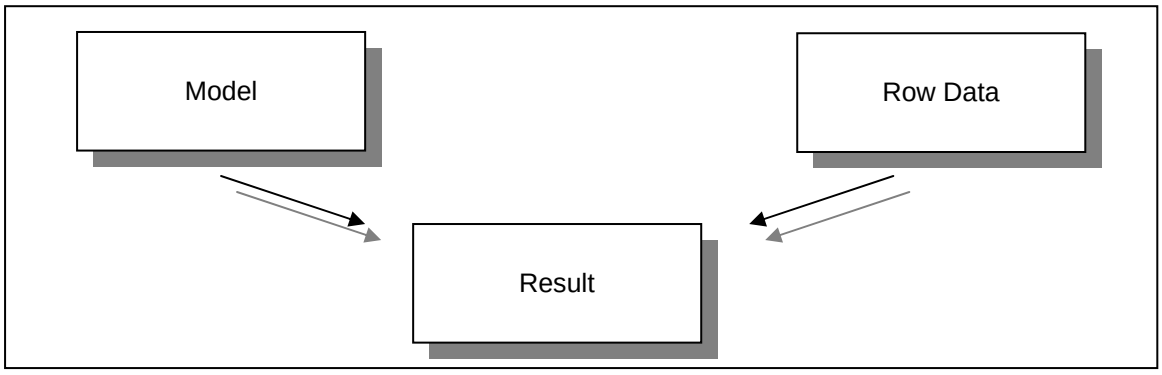

Figure 3 — The application phase

To apply the model correctly to input rows, the fields of the input rows have to be assigned to the fields that were identified as the relevant fields during the training phase. Hence, it is not sufficient to pass only the field values.

The application of association rule models is different from that of the other data mining functions. The model is applied to an item set. As output, a table is generated holding all the items that can be inferred from the input set using one or more rules in the model. In that way any inferred items can be delivered together with additional properties. In particular these are the support of the inferred item and its confidence. The output table has the following layout:

Column Name:	ITEM	SUPPORT	CONFIDENCE
Column Type:	CHARACTER VARYING	REAL	REAL
...	AddOnMemory	5.8	79.5
example rows:	Tripod	0,9	11.3
...	PhotoPrinter	0.8	10.1

Table 3 **— Items inferred from an association rule model**

In the case of bulk application of rule models, using DM_RuleApplTask, the input sets are defined as in association rule training by defining a group column. For each of the sets defined in that way, items may be inferred, which are then grouped together using a group column as well. Then, the output table has the following layout:

Column Name:	**GROUP**	**ITEM**	**SUPPORT**	**CONFIDENCE**
Column Type:	CHARACTER VARYING	CHARACTER VARYING	REAL	REAL
...	**Basket1**	**AddOnMemory**	**5.8**	**79.5**
	Basket2	**Tripod**	**0,9**	**11.3**
example rows:	**Basket2**	**PhotoPrinter**	**0.8**	**10.1**
	Basket3	**King's Gambit**	**1.9**	**89.1**
	Basket3	**Endgame Theory**	**4,9**	**71.4**
...	**Basket3**	**Die Schachnovelle**	**0.8**	**44.4**

Table 4 **— Groups of items inferred from an association rule model**

4.2.4 Testing phase

Only the techniques for classification and regression have a test phase. To test means to check the quality of the prediction using the data mining model. So, during test, a set of input rows with the target field is read and for each input row the application of the mining model is invoked. Then, the predicted value is compared to the actual value in the target field. When all input rows are read, a statistical result about the false predictions is computed and returned as the test result.

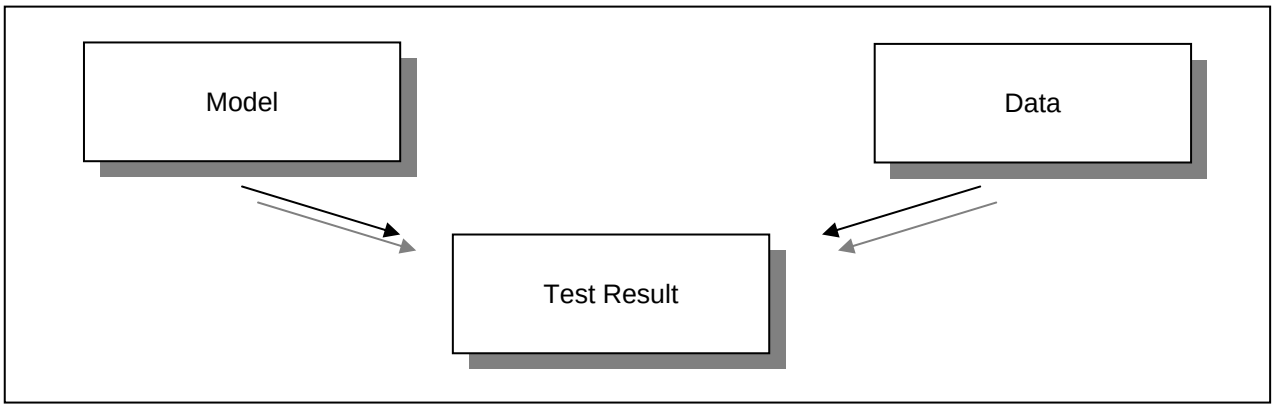

Figure 4 — The test phase

4.3 Mapping on user-defined types

4.3.1 Target application

The typical application scenario for the user-defined types introduced in this part of ISO/IEC 13249 is a data warehouse application. Warehouse applications typically need to flexibly execute different tasks on different data sets. For this reason, this standard proposes many different user-defined types that can fit together in different ways.

The approach introduced in the following subsections optimizes on flexibility and tries to make independent tasks as autonomous as possible. One might argue that this flexibility might result in unnecessary redundancy. However, this part of ISO 13249 defines an interface for data mining rather than an actual design for storing the information. Thus, redundancy is not directly implied by the introduced user-defined types.

4.3.2 The relationships of the user-defined types

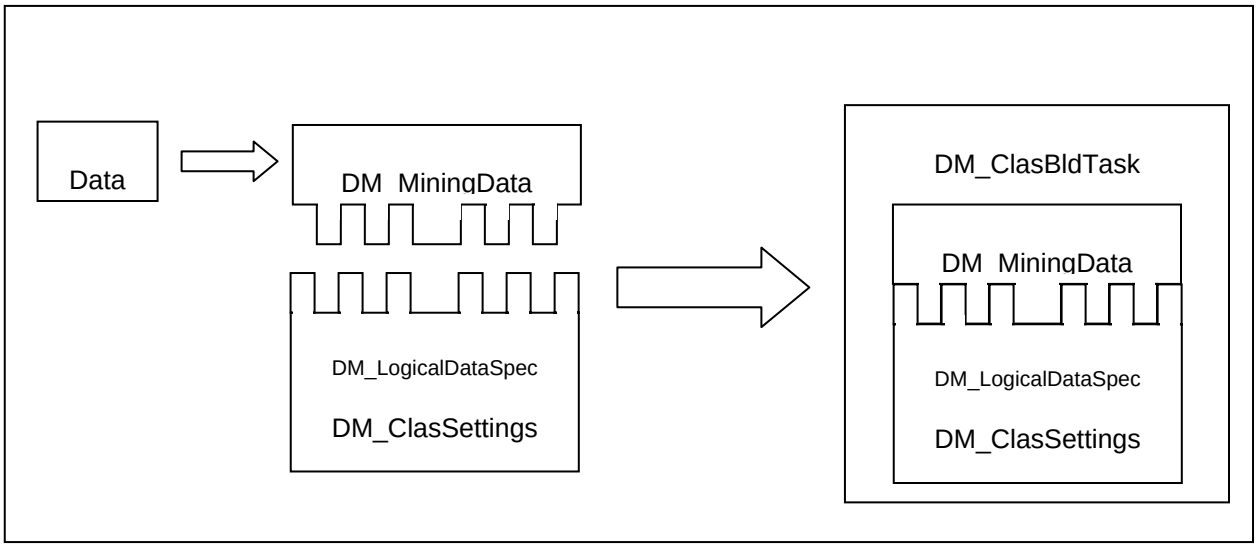

Figure 5 — Overview of user-defined types for the training phase

Before the user-defined types of part 6 of ISO 13249 are introduced, an example is given how the types are related. This should help in understanding the overall structure and the details of the remainder of this clause.

Figure 5 — Overview of user-defined types for the training phase shows the user-defined types (UDTs) introduced to define all the information needed for a data mining training run. The training phase of classification is used as an example to show how the types interact. Either starting from the data or the settings side, the goal is to define a mining task containing all information actually needed to do the training of a classification model. If the first step is to define the mining settings, then the mining settings are defined using a set of fields – the logical data specification. In a next step, a representation of real data (the DM_MiningData) is assigned, which completes the definition of a DM_ClasBldTask object. However, one can also start by defining a DM_MiningData value. The logical data specification can simply be derived from this mining data value and can then be used to define a mining settings. Finally, the settings with the logical data specification and the representation of the real data are connected in the classification build task object.

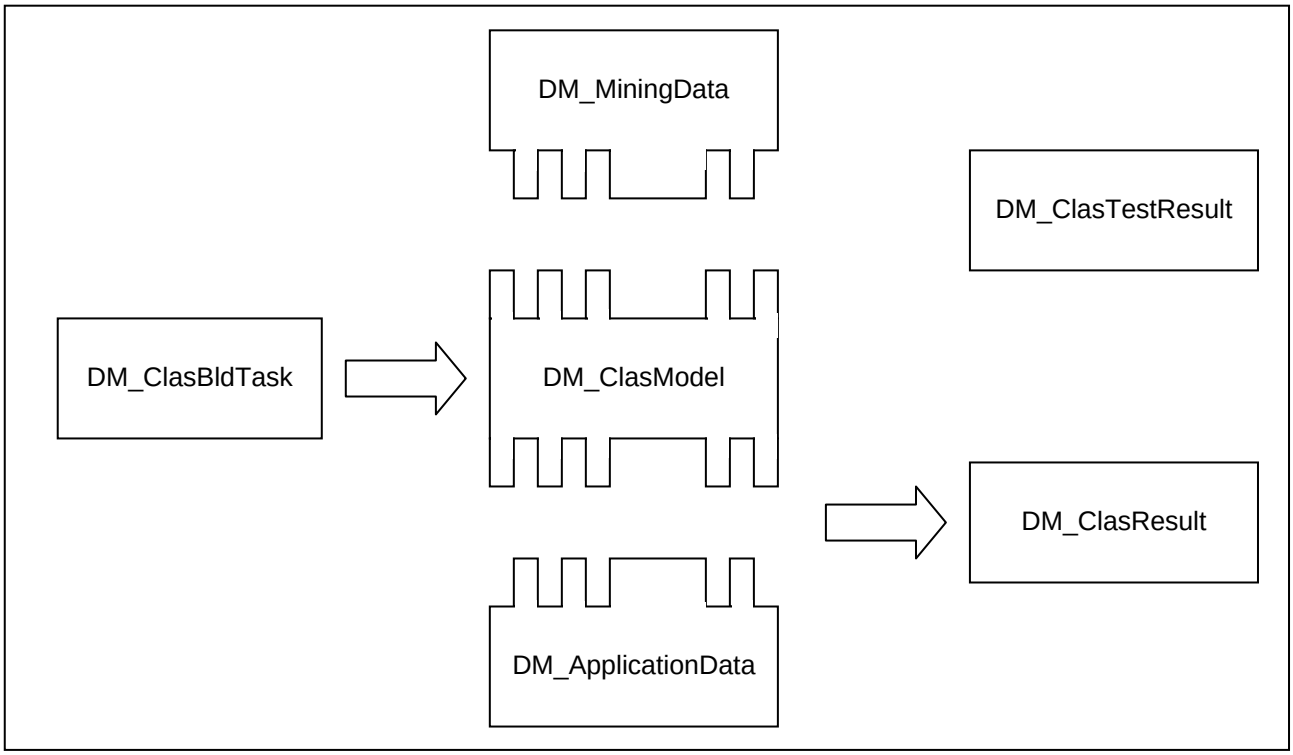

Figure 6 — Overview of user-defined types for training, test and application

The user-defined types introduced in **Figure 5 — Overview of user-defined types for the training phase** were introduced to allow for the storage and flexible combinations of definitions used for data mining. The UDTs shown in **Figure 6 — Overview of user-defined types for training, test and application** are defined to hold the results of either a data mining training, a data mining test, or a data mining application run.

The data mining training creates a data mining model from a mining task. This model has a logical data specification (not necessarily identical to the logical data specification of the data mining task). Objects of type mining data or application data shall match this specification in order to be usable for a subsequent test or application of the model. Since test and application create different results, two different user-defined types are introduced to represent these results.

Models can be applied to individual rows using the user-defined type DM_ApplicationData, but in a warehouse environment, one often wants to schedule the application of all rows of a table. To that end, an application task is used, which encapsulates the necessary information and can check at creation time whether the data specification of the model fits the input data. The result of executing the application task is a table containing a row that holds the model application results.

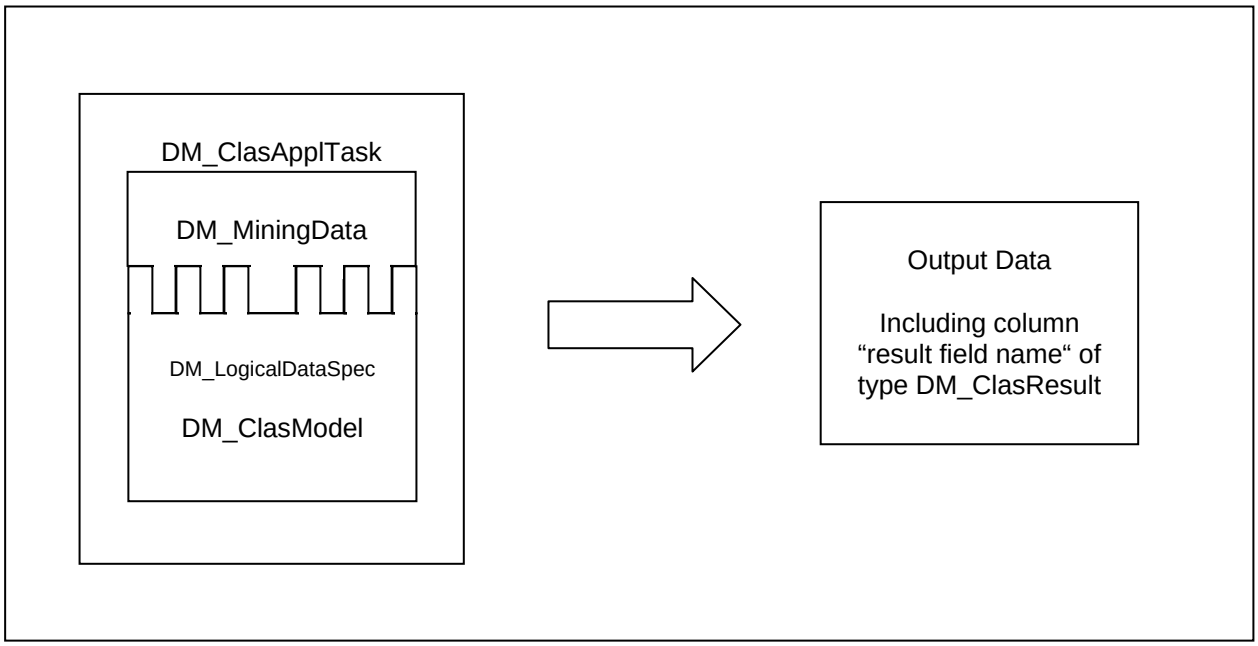

Figure 7 — Overview of user-defined types for bulk application

Figure 7 — Overview of user-defined types for bulk application shows how the user-defined types interact for the example of DM_ClasApplTask.

Similarly, model tests can be scheduled using test tasks (DM_ClasTestTask of DM_RegTestTask). This also ensures compatibility of model and input data at creation time. The result of running a test task is a single test result value (DM_ClasTestResult or DM_regTestResult).

4.3.3 User-defined types that are not related to mining techniques

The user-defined types for mining data, logical data specification, and mining application data represent information that is not mining technique specific. These types provide the basic infrastructure for the mining technique specific types.

1) The DM_MiningData user-defined type is an abstraction for real data contained in tables or views. A value of this type represents the metadata to access the real tables for a later data mining training, test or application run. This might include internal transformations used to clean or pre-process data in order to increase the quality of data mining models or data mining test results.

2) The DM_LogicalDataSpec user-defined type defines the input fields used by data mining training, data mining test, or data mining application runs. It allows the specification of data mining field related information. The most important information of this category is the association of a logical field type. The logical field type defines how an input field is handled by the data mining techniques. For instance, a field can be declared as categorical. This tells the data mining algorithms that there is no order defined for the values of this type and that by default only values contained in the input data are also valid values.

3) The DM_ApplicationData user-defined type is introduced as a container for data used to apply a data mining model. Basically, it is an abstraction for a set of values with associated names representing a single row of input data. This type is needed because the input values to apply a data mining model shall be mapped onto the logical data specification of the mining model.

4.3.4 User-defined types that are related to mining techniques

The types introduced in Subclause 4.3.3, “User-defined types that are not related to mining techniques”. do not provide any methods to compute any mining related information. These types can only be used to define the metadata needed for later data mining functions. The following categories of user-defined types are used for data mining processing. For each mining technique, there is at least one type in each of the following categories:

1) The mining task types like the DM_ClasBldTask type are intended to provide all information actually needed to start a data mining training, test or application run to compute, test or apply a data mining model. In a warehouse environment, this type might be used to store fully defined data mining runs such that the actual computation can be scheduled.

2) The mining model type (e.g. DM_ClasModel) is defined as an abstraction for an actual data mining model. It provides methods to access the properties of the model as well as methods to apply or to test the model. Values of the mining model type can only be generated by either using the build method of a corresponding mining task value or by importing the model from somewhere else.

   SQL/MM data mining requires support for data mining models in PMML format (see subclause 2.2 Publicly-available specifications on page 3). This format is based on XML and widely used for the exchange of data mining models. The internal representation of data mining models is implementation-dependent, but it must be possible to export a model to a PMML format and to import a model from a PMML format

3) The data mining test result type (e.g. DM_ClasTestResult) is introduced to hold the result information of a test run computed for a data mining model. A separate type was introduced because a test result might consist of an arbitrary number of values resulting from the invocation of a method having the characteristics of an aggregate function. That means that a data mining test run reads a set of input rows, applies a previously computed model and aggregates information about the quality of the prediction. The data mining test result represents the aggregate after the last input row was read.

4) The data mining application result type (e.g. DM_ClasResult) is also introduced because the application of a data mining model might also return multiple values, which is currently not supported by part 2 of ISO/IEC 9075. The computational pattern of the data mining application mode is pretty simple in the sense that a value of DM_ApplicationData is used in the apply method of a data mining model to compute a value of type application result. The methods of the application result value can then be used to query the details.

5) The data mining settings types (e.g. DM_ClusSettings) are introduced to hold all the information necessary to specify a data mining run. In particular, the settings types support the following parameters:

   a) DM_ClasSettings:
   The purpose of classification is to predict the class label value of previously unseen data of which this value is unknown. From a mathematical standpoint, the number of wrong predictions it leads to determines the quality of a classification model. The *classification cost rate* defines the maximal classification error tolerated during the validation phase, above which the model is considered unsatisfactory. In practice, however, some wrong predictions will cause large damage, while others can be tolerated more easily. To take such knowledge into account, the specific *cost* for the wrong prediction of one class label value instead of another can be specified.

   b) DM_ClusSettings:
   The purpose of clustering is the discovery of sets of input rows with common characteristics - the *clusters*. A basic operation in clustering is the computation of distance or similarity between pairs of records. This is determined by comparing their individual field values.
   It is not always useful to assign the same importance to all fields. In order to let the user define the relative importance of the fields, *field weights* are often used to define the relative importance of the fields.
   For the comparison of numerical field values, functions are used such as the absolute difference. This may produce poor results if the variances are largely different between different fields. A *similarity scale* is a value to allow the algorithm to compensate for this. If two values are one similarity scale apart, they should have medium similarity, i.e. their similarity becomes 0.5 in a range from 0 to 1.

   c) DM_RegSettings:
   Regression is a data mining technique similar to classification except for the numerical type of the target field. It computes a regression model allowing to predict numerical values. In order to specify the maximum allowed squared Pearson correlation coefficient for the training phase, the parameter *r-squared* is used.

   d) DM_RuleSettings:
   The association rules function discovers combinations of field values that appear frequently. The *minimum support* parameter specifies to compute only rules that occur in a sufficiently large percentage of the cases. *Minimum confidence* allows only rules that hold with a given minimal probability, and the *maximum rule length* puts a restriction on the number of values that are allowed in a single rule.

6) The rule filter type DM_RuleFilter holds all the information necessary to specify a subset of association or sequence rules. Using this type allows the user to select a specific subset of rules from a rule model of type DM_RuleModel. Alternatively, values of type DM_RuleFilter can be added to rule settings. As a consequence, only rules that satisfy the filter conditions are kept during the training phase.

5 Data Mining Data Types

5.1 DM_LogicalDataSpec Type and Routines

5.1.1 DM_LogicalDataSpec Type

Purpose

DM_LogicalDataSpec is an abstraction for a set of data mining fields identified by their name. Each of the fields has also an associated logical field type. The type DM_LogicalDataSpec is introduced to represent the input data needed for both data mining training and data mining test.

Definition

```
CREATE TYPE DM_LogicalDataSpec
  AS (
        DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
     )
  INSTANTIABLE
  NOT FINAL

METHOD DM_addDataSpecFld
         (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_LogicalDataSpec
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_remDataSpecFld
         (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_LogicalDataSpec
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getNumFields()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getFldName (position INTEGER)
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_setFldType
         (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength),
          miningType INTEGER)
  RETURNS DM_LogicalDataSpec
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```

```
METHOD DM_getFldType
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS SMALLINT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_isCompatible (dataSpec DM_LogicalDataSpec)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_LogicalDataSpec*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_LogicalDataSpec* type provides for public use:

   a) a method *DM_addDataSpecFld(CHARACTER VARYING),*

   b) a method *DM_remDataSpecFld(CHARACTER VARYING),*

   c) a method *DM_getNumFields(),*

   d) a method *DM_getFldName(INTEGER),*

   e) a method *DM_setFldType(CHARACTER VARYING, INTEGER),*

   f) a method *DM_getFldType(CHARACTER VARYING),*

   g) a method *DM_isCompatible(DM_LogicalDataSpec).*

5.1.2 DM_addDataSpecFld Method

Purpose

Add a field with the specified name to the DM_LogicalDataSpec value.

Definition

```
CREATE METHOD DM_addDataSpecFld
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_LogicalDataSpec
  FOR DM_LogialDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_addDataSpecFld* takes the following input parameter:

   a) a *CHARACTER VARYING* value *fieldName*.

2) The result of the invocation *DM_addDataSpecFld(fieldName)* is determined as follows:

   Case:

   a) If *fieldName* is the null value, then *SELF* is returned.

   b) If a field having a name equal to *fieldName* is contained in the set of fields of *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field already defined*.

   c) Otherwise, it is a value of type *DM_LogicalDataSpec* consisting of the set of fields contained in *SELF* followed by a new field with *fieldName* as its name. The type of the field is set to the implementation-defined default type. The position of the field is set to an implementation-dependent value.

5.1.3 DM_remDataSpecFld Method

Purpose

Remove the mining field with the specified name from the DM_LogicalDataSpec value.

Definition

```
CREATE METHOD DM_remDataSpecFld
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_LogicalDataSpec
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_remDataSpecFld* takes the following input parameter:

   a) a *CHARACTER VARYING* value *fieldName*.

2) The result of the invocation *DM_remDataSpecFld(fieldName)* is determined as follows:

   Case:

   a) If *fieldName* is the null value, then *SELF* is returned.

   b) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) Otherwise, it is the *DM_LogicalDataSpec* value derived from *SELF* by deleting the field with a name equal to *fieldName* from the set of fields contained in *SELF*. The positions of the fields in the returned value are implementation-dependent.

5.1.4 DM_getNumFields Method

Purpose

Returns the number of fields of the DM_LogicalDataSpec value.

Definition

```
CREATE METHOD DM_getNumFields()
  RETURNS INTEGER
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumFields()* is the number of fields contained in the set of fields of *SELF*.

5.1.5 DM_getFldName Method

Purpose

Returns the name of the data mining field of the DM_LogicalDataSpec value at the specified position.

Definition

```
CREATE METHOD DM_getFldName (position INTEGER)
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_getFldName* takes the following input parameter:

   a) an *INTEGER* value *position*.

2) The result of the invocation *DM_getFldName(position)* is determined as follows:

   Case:

   a) If *position* is greater than 0 (zero) and less than or equal to the result of a call of *DM_getNumFields()*, then the name of the mining field with the associated number *position*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

5.1.6 DM_setFldType Method

Purpose

Sets the field type of the field with the specified name contained in the DM_LogicalDataSpec value.

Definition

```
CREATE METHOD DM_setFldType
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength),
           miningType INTEGER)
  RETURNS DM_LogicalDataSpec
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setFldType* takes the following input parameters:

   a) a *CHARACTER VARYING* value *fieldName*,

   b) an *INTEGER* value *miningType*.

2) The following values for field types are defined:

Field type	Code for field type
DM_Categorical	0 (zero)
DM_Numerical	1 (one)

Table 5 — Values for field types

3) The result of the invocation *DM_setFldType(fieldName, miningType)* is determined as follows:

   Case:

   a) If *fieldName* is the null value, then *SELF* is returned.

   b) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) If *miningType* is the null value, then the field type of the field with name *fieldName* of the set of fields of *SELF* is reset to the implementation-defined default.

   d) If *miningType* is not a value defined in Table 5 — Values for field types, then an exception condition is raised: *SQL/MM Data Mining Exception – parameter out of range.*

e) Otherwise, the field type of the field with a name equal to *fieldName* of the set of fields of *SELF* is set to the value of *miningType* (see Table 5 — Values for field types).

5.1.7 DM_getFldType Method

Purpose

Returns the type of the data mining field with the specified name of the DM_LogicalDataSpec value.

Definition

```
CREATE METHOD DM_getFldType
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS SMALLINT
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_getFldType* takes the following input parameter:

   a) a *CHARACTER VARYING* value *fieldName*.

2) The result of the invocation *DM_getFldType(fieldName)* is determined as follows:

   Case:

   a) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   b) If *SELF* is equal to *SELF.DM_setFldType(fieldName*, *v)* for some SMALLINT value *v* representing a field type, then *v*. For defined values for the field type see Table 5 — Values for field types on page 31.

   c) Otherwise, the null value.

5.1.8 DM_isCompatible Method

Purpose

Determine whether a logical data specification is compatible with the existing logical data specification.

Definition

```
CREATE METHOD DM_isCompatible (dataSpec DM_LogicalDataSpec)
  RETURNS INTEGER
  FOR DM_LogicalDataSpec
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_isCompatible* takes the following parameter:

   a) a *DM_LogicalDataSpec* value *dataSpec*.

2) The result of the invocation *DM_isCompatible(dataSpec)* is determined as follows:

   a) 1 (one) if *dataSpec* is compatible with *SELF*. This is the case if for each field F in *SELF*, there is a field G in *dataSpec* with an identical name. The type of G shall be *DM_Numerical* whenever the type of F is *DM_Numerical*.

   b) 0 (zero) otherwise.

5.2 DM_MiningData Type and Routines

5.2.1 DM_MiningData Type

Purpose

DM_MiningData is a description of data contained in tables, which represents the metadata required for access to the data for data mining training, test or application runs.

Definition

```
CREATE TYPE DM_MiningData
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defMiningData
          (tableName CHARACTER VARYING(DM_MaxTableNameLength))
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_setFldAlias
          (dataColumn CHARACTER VARYING(DM_MaxFieldNameLength),
           aliasName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_MiningData
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_genDataSpec()
  RETURNS DM_LogicalDataSpec
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_MiningData.*

2) *DM_MaxTableNameLength* is the implementation-defined maximum number of characters for the name of a table.

3) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.

4) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_MiningData* type provides for public use:

   a) a method *DM_defMiningData(CHARACTER VARYING)*,

   b) a method *DM_setFldAlias(CHARACTER VARYING, CHARACTER VARYING)*,

c) a method *DM_genDataSpec()*.

5.2.2 DM_defMiningData Method

Purpose

Return the DM_MiningData value corresponding to the source table determined by the given name.

Definition

```
CREATE STATIC METHOD DM_defMiningData
          (tableName CHARACTER VARYING(DM_MaxTableNameLength))
  RETURNS DM_MiningData
  FOR DM_MiningData
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxTableNameLength* is the implementation-defined maximum number of characters for the name of a table.

Description

1) The method *DM_defMiningData* takes the following input parameter:

   a) a *CHARACTER VARYING* value *tableName*.

2) The result of the invocation *DM_defMiningData(tableName)* is determined as follows:

   a) If *tableName* identifies a valid table, then a value of *DM_MiningData* determined by the identified table. For each column of the table of type character, character varying, datetime, or numerical a field description is created in the *DM_MiningData* value. The field has an associated name and an associated alias name, which are both identical to the name of the column in the table.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid table name*.

5.2.3 DM_setFldAlias Method

Purpose

Define an alias name for a field contained in a DM_MiningData value.

Definition

```
CREATE METHOD DM_setFldAlias
          (dataField CHARACTER VARYING(DM_MaxFieldNameLength),
           aliasName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_MiningData
  FOR DM_MiningData
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setFldAlias* takes the following input parameters:

   a) a *CHARACTER VARYING* value *dataField*,

   b) a *CHARACTER VARYING* value *aliasName*.

2) The result of the invocation *DM_setFldAlias(dataField*, *aliasName*) is determined as follows:

   a) If *dataField* is the null value, then *SELF*.

   b) If *aliasName* is not the null value and there is a field contained in *SELF* with an alias name equal to *aliasName*, then an exception condition is raised: *SQL/MM Data Mining exception – alias already in use.*

   c) If *dataField* is a valid name of a field in *SELF* and *aliasName* is not the null value, then the alias of that field is set to the value of *aliasName.*

   d) If *dataField* is a valid name of a field in *SELF* and it is not the alias of any other field in *SELF*, and *aliasName* is the null value, then the alias of that field is set to the value of *dataField.*

   e) If *dataField* is the alias of some other field in *SELF* and *aliasName* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – alias already in use.*

   f) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid field name.*

5.2.4 DM_genDataSpec Method

Purpose

Generates value of type DM_genDataSpec.

Definition

```
CREATE METHOD DM_genDataSpec()
   RETURNS DM_LogicalDataSpec
   FOR DM_MiningData
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_genDataSpec()* is a value of type *DM_LogicalDataSpec* consisting of a set of fields derived from the set of fields contained in *SELF*. For each field in *SELF*, a field is contained in the *DM_LogicalDataSpec* value. Each field in the *DM_LogicalDataSpec* has an associated name, which is identical to the alias name of the corresponding field in *SELF*.

2) Each field in the *DM_LogicalDataSpec* value has an associated default field type derived from the data type of the field contained in *SELF*. The field type is derived according to the following mapping:

   Case:

   a) If the field in *DM_MiningData* is of numerical type, then the field type is set to DM_Numerical (see Table 5 — Values for field types in Subclause 5.1.6, “DM_setFieldType Method”.).
   Note: The exact mapping of datetimes to numbers for the processing by a data mining algorithm is implementation-defined.

   b) Otherwise, the field type is set to DM_Categorical (see Table 5 — Values for field types in Subclause 5.1.6, “DM_setFieldType Method”.).

5.3 DM_ApplicationData Type and Routines

5.3.1 DM_ApplicationData Type

Purpose

The DM_ApplicationData type is the description of a single input row that can be used for the application of a data mining model.

Definition

```
CREATE TYPE DM_ApplicationData
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_genDataSpec()
  RETURNS DM_LogicalDataSpec
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_impApplData
          (inputRow CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ApplicationData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for *DM_content* of a *DM_ApplicationData*.

Description

1) The *DM_ApplicationData* type provides for public use:

   a) a method *DM_genDataSpec()*,

   b) a method *DM_impApplData(CHARACTER LARGE OBJECT(DM_MaxContentLength))*.

5.3.2 DM_genDataSpec Method

Purpose

Return the DM_LogicalDataSpec corresponding to SELF.

Definition

```
CREATE METHOD DM_genDataSpec ()
  RETURNS DM_LogicalDataSpec
  FOR DM_ApplicationData
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_genDataSpec()* is a value of type *DM_LogicalDataSpec* with a field for each input value in the input data of *DM_impApplData*. Each of these fields has the field type DM_Numerical if the given value for a field represents a numerical value.

5.3.3 DM_impApplData Method

Purpose
Return the DM_ApplicationData value containing field names and their associated values for a single input row corresponding to the given input parameter.

Definition

```
CREATE STATIC METHOD DM_impApplData
          (inputRow CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ApplicationData
  FOR DM_ApplicationData
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ApplicationData*.

Description

1) The method *DM_impApplData* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputRow*.

2) The value of *inputRow* is a valid representation of application input data if it is a well-formed XML element (see XML 1.0). It shall be a single element and match the declaration of type 'row' as defined by:

```
<!ELEMENT row    (column*) >
<!ELEMENT column (#PCDATA) >
<!ATTLIST column
    name CDATA REQUIRED
    null ( true | false ) "false"
>
```

   A value of the attribute "name" is interpreted as the name of mining field. The content of an element is interpreted as the value of the named mining field. The order of the elements is not relevant.
   If the data for some field is null, then it is represented by an element where the attribute “null” has the value “true”.

3) The result of the invocation *DM_impApplData(inputRow)* is determined as follows:

   Case:

   a) if *inputRow* is a valid representation of application input data, then a value of *DM_ApplicationData* corresponding to the input data.

4) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid application input data format.*

6 Association Rules

6.1 DM_RuleSettings Type and Routines

6.1.1 DM_RuleSettings Type

Purpose

The DM_RuleSettings type is the description of the settings that are used to generate a rule model.

Definition

```
CREATE TYPE DM_RuleSettings
  AS (
        DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impRuleSettings
         (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RuleSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expRuleSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setRuleFilter(ruleFilter DM_RuleFilter)
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleFilter()
  RETURNS DM_RuleFilter
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setMinSupport(support DOUBLE PRECISION)
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMinSupport()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```

```
METHOD DM_setMinConf(confidence DOUBLE PRECISION)
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMinConf()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setMaxLength(maxRuleLength INTEGER)
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMaxLength()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_useRuleDataSpec(logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleDataSpec()
  RETURNS DM_LogicalDataSpec
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setGroup(groupField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getGroup()
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setSequence
      (sequenceField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RuleSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```



```
METHOD DM_getSequence()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleSettings*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_RuleSettings* type provides for public use:

   a) a method *DM_impRuleSettings(CHARACTER LARGE OBJECT(DM_MaxContentLength))*,

   b) a method *DM_expRuleSettings()*,

   c) a method *DM_setRuleFilter(DM_RuleFilter)*,

   d) a method *DM_getRuleFilter()*,

   e) a method *DM_setMinSupport(DOUBLE PRECISION)*,

   f) a method *DM_getMinSupport()*,

   g) a method *DM_setMinConf(DOUBLE PRECISION)*,

   h) a method *DM_getMinConf()*,

   i) a method *DM_setMaxLength(INTEGER)*,

   j) a method *DM_getMaxLength()*,

   k) a method *DM_useRuleDataSpec(DM_LogicalDataSpec)*,

   l) a method *DM_getRuleDataSpec()*,

   m) a method *DM_setGroup(CHARACTER VARYING)*,

   n) a method *DM_getGroup()*,

   o) a method *DM_setSequence(CHARACTER VARYING)*,

   p) a method *DM_getSequence()*.

6.1.2 DM_impRuleSettings Method

Purpose

Return a specified value of type DM_RuleSettings.

Definition

```
CREATE STATIC METHOD DM_impRuleSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleSettings*.

Description

1) The method *DM_impRuleSettings* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputSettings.* The format of this value is implementation-dependent, but for any one implementation the format is the same as that used by *DM_expRuleSettings*.

2) The result of the invocation *DM_impRuleSettings(inputSettings)* is determined as follows:

   Case:

   a) If there is a value v of type *DM_RuleSettings* such that *inputSettings* = v.*DM_expRuleSettings()*, then v.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid settings format*.

6.1.3 DM_expRuleSettings Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the settings for association rule or sequence rule discovery contained in the DM_content value of the DM_RuleSettings value.

Definition

```
CREATE METHOD DM_expRuleSettings()
   RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleSettings*.

Description

1) The result of the invocation *DM_expRuleSettings()* is a *CHARACTER LARGE OBJECT* value representing the settings for association rule or sequence rule discovery contained in *SELF*. This representation is implementation-dependent.

6.1.4 DM_setRuleFilter Method

Purpose

This method sets a rule filter in a value of a settings object for association rule or sequence rule discovery. Rule models created using the resulting value of DM_RuleSettings shall conform to the restrictions set forth in the filter.

Definition

```
CREATE METHOD DM_setRuleFilter(ruleFilter DM_RuleFilter)
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setRuleFilter* takes the following input parameter:

   a) a *DM_RuleFilter* value *ruleFilter*.

2) The result of the invocation *DM_setRuleFilter(ruleFilter)* is determined as follows:

   Case:

   a) If *ruleFilter* is the null value, then a value of type DM_RuleSettings without any filter for rule restriction.

   b) Otherwise, a value v of type *DM_RuleSettings* that is identical to *SELF* except that the method invocation v.*DM_getRuleFilter() = ruleFilter*.

6.1.5 DM_getRuleFilter Method

Purpose

This method returns the rule filter contained in SELF.

Definition

```
CREATE METHOD DM_getRuleFilter()
  RETURNS DM_RuleFilter
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleFilter()* is determined as follows:

   Case:

   a) If no rule filter is contained in *SELF*, then the null value.

   b) Otherwise, the value of type *DM_RuleFilter* contained in *SELF*.

6.1.6 DM_setMinSupport Method

Purpose

Sets the minimum support in a value of a settings object for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_setMinSupport(support DOUBLE PRECISION)
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setMinSupport* takes the following input parameter:

   a) a *DOUBLE PRECISION* value *support*.

2) The result of the invocation *DM_setMinSupport(support)* is determined as follows:

   Case:

   a) If *support* is negative or greater than 100, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   b) Otherwise a value of type *DM_RuleSettings* copied from *SELF* that will return the value of the parameter *support* if *DM_getMinSupport()* is called on it.

6.1.7 DM_getMinSupport Method

Purpose

Returns the minimum support that was specified as a parameter for the search for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_getMinSupport()
  RETURNS DOUBLE PRECISION
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getMinSupport()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setMinSupport(*v*)* for some *DOUBLE PRECISION* value v, then v.

   b) Otherwise, the null value, indicating that a numerical value for the minimum support will be determined during the data mining run.

6.1.8 DM_setMinConf Method

Purpose

Sets the minimum confidence in a value of a settings object for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_setMinConf(confidence DOUBLE PRECISION)
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setMinConf* takes the following input parameter:

   a) a *DOUBLE PRECISION* value *confidence*.

2) The result of the invocation *DM_setMinConf(confidence)* is determined as follows:

   Case:

   a) If *confidence* is negative or greater than 100, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of rang*e.

   b) Otherwise, a value v of type *DM_RuleSettings* that is identical to *SELF* except that the method invocation v.*DM_getMinConf() = confidence.*

6.1.9 DM_getMinConf Method

Purpose

Returns the minimum confidence in a value of a settings object for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_getMinConf()
   RETURNS DOUBLE PRECISION
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getMinConf()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setMinConf*(v*)* for some *DOUBLE PRECISION* value v, then v.

   b) Otherwise, the null value, to indicate that the minimum confidence will be determined during the data mining run.

6.1.10 DM_setMaxLength Method

Purpose

Sets the maximum rule length in a value of a settings object for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_setMaxLength (ruleLength INTEGER)
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setMaxLength* takes the following input parameter:

   a) an *INTEGER* value *ruleLength*.

2) The result of the invocation *DM_setMaxLength(ruleLength)* is determined as follows:

   Case:

   a) If *ruleLength* is less than 2, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of rang*e.

   b) Otherwise, a value v of type *DM_RuleSettings* that is identical to *SELF* except that the method invocation v.*DM_getMaxLength() = ruleLength.*

6.1.11 DM_getMaxLength Method

Purpose

Returns the maximum rule length in a value of a settings object for association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_getMaxLength()
   RETURNS INTEGER
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getMaxLength()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setMaxLength*(v) for some *INTEGER* value v > 1 (one), then v.

   b) Otherwise, the null value, to indicate that the rule length is unlimited.

6.1.12 DM_useRuleDataSpec Method

Purpose

Specify a DM_LogicalDataSpec value for the DM_RuleSettings value. The DM_LogicalDataSpec value determines valid DM_MiningData values, which can be processed in a training run.

Definition

```
CREATE METHOD DM_useRuleDataSpec
          (logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_useRuleDataSpec* takes the following input parameter:

   a) a *DM_LogicalDataSpec* value *logicalDataSpecification*.

2) The result of the invocation *DM_useRuleDataSpec(logicalDataSpecification)* is determined as follows:

   Case:

   a) If *logicalDataSpecification* is the null value, then a *DM_RuleSettings* value containing no logical data specification but otherwise identical to *SELF*.

   b) If *SELF.DM_getGroup()* = group (not null) and group is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   c) If *SELF.DM_getSequence()* = sequence (not null) and sequence is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   d) Otherwise, it is the *DM_RuleSetting*s value containing *logicalDataSpecification* as the logical data specification.

6.1.13 DM_getRuleDataSpec Method

Purpose

Returns the logical data specification defined for a rule settings.

Definition

```
CREATE METHOD DM_getRuleDataSpec()
  RETURNS DM_LogicalDataSpec
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleDataSpec()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_useRuleDataSpec(*v*)* for some *DM_LogicalDataSpec* value v, then v.

   b) Otherwise, the null value.

6.1.14 DM_setGroup Method

Purpose

Specifies the field used to identify the group (e.g. purchase transaction) for mining association rules or sequence rules. The group field will hold information to group all the items that belong to one item set. In the case of sequence rule discovery, ordering by the group field will define the order of the sequences.

Definition

```
CREATE METHOD DM_setGroup()
         (groupField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setGroup* takes the following input parameter:

   a) a *CHARACTER VARYING* value *groupField*.

2) The result of the invocation *DM_setGroup(groupField)* is determined as follows:

   Case:

   a) If *groupField* is the null value, then a value of type *DM_RuleSettings* not containing a group field but otherwise identical to *SELF*.

   b) If *SELF.DM_getRuleDataSpec()* = ds (not null) and no field with name *groupField* is contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) Otherwise, it is the *DM_RuleSettings* value containing the field named *groupField* as the grouping field.

6.1.15 DM_getGroup Method

Purpose

Returns the group field name defined for a rule settings.

Definition

```
CREATE METHOD DM_getGroup()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getGroup()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setGroup*(v*)* for some *CHARACTER VARYING* value v, then v.

   b) Otherwise, it is the null value.

6.1.16 DM_setSequence Method

Purpose

Defines the field used to identify the item sets belonging to the same sequence (e.g. of purchase transactions) for sequence rule discovery. This also specifies that data mining sequence rule discovery shall be performed. The sequence field will hold information to group all the item sets that belong to one sequence.

Definition

```
CREATE METHOD DM_setSequence()
          (sequenceField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RuleSettings
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setSequence* takes the following input parameter:

   a) a *CHARACTER VARYING* value *sequenceField*.

2) The result of the invocation *DM_setSequence(sequenceField)* is determined as follows:

   Case:

   a) If *sequenceField* is the null value, then a value of type *DM_RuleSettings* not containing a sequence field but otherwise identical to *SELF*.

   b) If *SELF.DM_getRuleDataSpec()* = ds (not null) and no field with name *sequenceField* is contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) Otherwise, it is the *DM_RuleSettings* value containing the field named *sequenceField* as the sequence field.

6.1.17 DM_getSequence Method

Purpose

Returns the sequence field name defined for a rule settings. If the null value is returned, then no sequence field is contained in SELF, and association rule discovery shall be performed.

Definition

```
CREATE METHOD DM_getSequence()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getSequence()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setSequence(*v*)* for some *CHARACTER VARYING* value v, then v.

   b) Otherwise, it is the null value.

6.2 DM_RuleBldTask type and Routines

6.2.1 DM_RuleBldTask Type

Purpose

A DM_RuleBldTask represents the information about a search for association rules or sequence rules, in particular the input data and the parameter settings. The type provides a method for computing a data mining rule model.

Definition

```
CREATE TYPE DM_RuleBldTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defRuleBldTask
          (trainData DM_MiningData, settings DM_RuleSettings)
  RETURNS DM_RuleBldTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleTrnData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRuleSettings()
  RETURNS DM_RuleSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_buildRuleModel()
  RETURNS DM_RuleModel
  LANGUAGE SQL
  NOT DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleBldTask*

Description

1) The type *DM_RuleBldTask* provides for public use:

   a) a method *DM_defRuleBldTask(DM_MiningData, DM_RuleSettings)*,

   b) a method *DM_getRuleTrnData()*,

   c) a method *DM_getRuleSettings()*,

   d) a method *DM_buildRuleModel()*.

6.2.2 DM_defRuleBldTask Method

Purpose

Given a DM_MiningData value and a DM_RuleSettings value return the corresponding DM_RuleBldTask value.

Definition

```
CREATE STATIC METHOD DM_defRuleBldTask
          (trainData DM_MiningData, settings DM_RuleSettings)
  RETURNS DM_RuleBldTask
  FOR DM_RuleBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defRuleBldTask* takes the following input parameters:

   a) a *DM_MiningData* value *trainData*,

   b) a *DM_RuleSettings* value *settings*.

2) The result of the invocation *DM_defRuleBldTask(trainData, settings)* is determined as follows:

   Case:

   a) If *settings* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null settings*.

   b) If *trainData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *settings.DM_getRuleDataSpec().isCompatible(trainData.DM_genDataSpec())* equals 1 (one), then a *DM_RuleBldTask* value bt such that bt.*DM_getRuleTrnData()* equals *trainData* and bt.*DM_getRuleSettings()* equals *settings*.

   d) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – data and data specification not compatible*.

6.2.3 DM_getRuleTrnData Method

Purpose

Return the DM_MiningData value representing the training data for the given DM_RuleBldTask value.

Definition

```
CREATE METHOD DM_getRuleTrnData()
  RETURNS DM_MiningData
  FOR DM_RuleBldTask
  BEGIN
  --
  --!! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleTrnData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for training, then the null value.

   b) Otherwise, it is the *DM_MiningData* value contained in *SELF*.

6.2.4 DM_getRuleSettings Method

Purpose

Return the value of type DM_RuleSettings representing the settings of association rule or sequence rule discovery.

Definition

```
CREATE METHOD DM_getRuleSettings()
   RETURNS DM_RuleSettings
   FOR DM_RuleBldTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRuleSettings()* is:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_RuleSettings* value, then the null value.

   b) Otherwise, it is the *DM_RuleSettings* value representing the settings to be used for the training phase.

6.2.5 DM_buildRuleModel Method

Purpose

Return a specified value of type DM_RuleModel.

Definition

```
CREATE STATIC METHOD DM_buildRuleModel()
  RETURNS DM_RuleModel
  FOR DM_RuleBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_buildRuleModel()* is determined as follows:

   Case:

   a) If *DM_buildRuleModel()* is successful, then an implementation-dependent value of type *DM_RuleModel* determined by the information contained in *SELF.*

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model computation failed*.

6.3 DM_RuleModel Type and Routines

6.3.1 DM_RuleModel Type

Purpose

The DM_RuleModel type represents models that are the result of the search for association rules or sequence rules.

Definition

```
CREATE TYPE DM_RuleModel
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impRuleModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RuleModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRuleMdlType ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_expRuleModel ()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_applyRuleModel
             (inputItemSet CHARACTER VARYING(DM_MaxItemSetLength))
  RETURNS TABLE (ITEM CHARACTER VARYING(DM_MaxItemLength),
                 SUPPORT REAL, CONFIDENCE REAL)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getNumItemSets ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getNumSequences ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD DM_getNumRules ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_filterRules (ruleFilter DM_RuleFilter)
  RETURNS DM_RuleModel
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getNumTransacts ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getItemSets ()
  RETURNS TABLE (ITEMSETID INTEGER, SUPPORT REAL, LIFT REAL,
                 ITEM CHARACTER VARYING(DM_MaxItemLength),
                 ITEMNAME CHARACTER VARYING(DM_MaxItemLength))
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRules ()
  RETURNS TABLE ( ID INTEGER,
                  HEADNAME CHARACTER VARYING(DM_MaxItemLength),
                  HEAD CHARACTER VARYING(DM_MaxItemLength),
                  BODYID INTEGER, LENGTH INTEGER,
                  BODYTEXT CHARACTER VARYING(DM_MaxBodyLength),
                  SUPPORT REAL, CONFIDENCE REAL, LIFT REAL)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRuleBodies ()
  RETURNS TABLE ( BODYID INTEGER,
                  ITEM CHARACTER VARYING(DM_MaxItemLength),
                  ITEMNAME CHARACTER VARYING(DM_MaxItemLength))
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getSequences ()
  RETURNS TABLE (SEQID INTEGER, NUMITEMSETS INTEGER, NUMITEMS INTEGER,
                 SUPPORT REAL, LIFT REAL,
                 MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL,
                 SEQTEXT CHARACTER VARYING(DM_MaxBodyLength))
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getSeqDetails ()
  RETURNS TABLE (SEQID INTEGER, POSITION INTEGER, ITEMSETID INTEGER,
                 ITEMSETTEXT CHARACTER VARYING(DM_MaxBodyLength),
                 MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL,
                 ITEMSETTEXT CHARACTER VARYING(DM_MaxBodyLength))
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```



```
METHOD DM_getSeqRules ()
   RETURNS TABLE (SEQRULEID INTEGER,
                  BODYSEQID INTEGER, HEADSEQID INTEGER,
                  BODYSEQTEXT CHARACTER VARYING(DM_MaxBodyLength),
                  HEADSEQTEXT CHARACTER VARYING(DM_MaxBodyLength),
                  NUMITEMSETS INTEGER, NUMITEMS INTEGER,
                  SUPPORT REAL, CONFIDENCE REAL, LIFT REAL,
                  MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_getRuleBldTask ()
   RETURNS DM_RuleBldTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleModel*.
2) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an entry in the HEAD or HEADNAME columns of table DM_getRules() as well as the ITEM or ITEMNAME columns of table DM_getRuleBodies() and DM_applyRuleModel(itemset).
3) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for an entry in the BODYTEXT column of table DM_getRules().
4) *DM_MaxItemSetLength* is the implementation-defined maximum number of characters for a value of an item set.

Description

1) The *DM_RuleModel* type provides for public use:
   a) a method *DM_impRuleModel(CHARACTER LARGE OBJECT(DM_MaxContentLength)),*
   b) a method *DM_getRuleMdlType(),*
   c) a method *DM_expRuleModel(),*
   d) a method *DM_applyRuleModel(CHARACTER VARYING(DM_MaxItemSetLength)),*
   e) a method *DM_getNumItemSets(),*
   f) a method *DM_getNumSequences(),*
   g) a method *DM_getNumRules(),*
   h) a method *DM_filterRules(DM_RuleFilter),*
   i) a method *DM_getNumTransacts(),*
   j) a method *DM_getItemSets(),*
   k) a method *DM_getRules(),*
   l) a method *DM_getRuleBodies(),*
   m) a method *DM_getSequences(),*
   n) a method *DM_getSeqDetails(),*
   o) a method *DM_getSeqRules(),*
   p) a method *DM_getRuleBldTask().*

6.3.2 DM_impRuleModel Method

Purpose

Return a specified value of type DM_RuleModel.

Definition

```
CREATE STATIC METHOD DM_impRuleModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RuleModel
  FOR DM_RuleModel
  BEGIN
  --
  --!! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleModel*.

Description

1) The method *DM_impRuleModel* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputModel*.

2) The result of the invocation *DM_impRuleModel(inputModel)* is determined as follows:

   Case:

   a) If *inputModel* contains an XML document defined by the PMML schema with exactly one AssociationModel or SequenceModel element, then a value of type *DM_RuleModel* representing the given PMML model.

   b) If *inputModel* contains an XML document defined by the PMML schema with more than one AssociationModel or SequenceModel element, then a value of type *DM_RuleModel* representing the first of the models.

   c) Otherwise, an exception condition is raised: SQL/MM Data Mining exception – invalid import format.

6.3.3 DM_getRuleMdlType Method

Purpose

Returns the type of a rule model, which can be either an association rule model or a sequence rule model.

Definition

```
CREATE METHOD DM_getRuleMdlType()
   RETURNS INTEGER
   FOR DM_RuleModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRuleMdlType()* is determined as follows:

   Case:

   a) 1 (one) if *SELF* contains an association rule mode.

   b) 2, otherwise. In this case *SELF* contains a sequence rule model.

6.3.4 DM_expRuleModel Method

Purpose

Return the CHARACTER LARGE OBJECT representation of the rule model corresponding to SELF.

Definition

```
CREATE METHOD DM_expRuleModel ()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleModel*.

Description

1) The result of the invocation *DM_expRuleModel ()* is a *CHARACTER LARGE OBJECT* value that contains an XML document conforming to the PMML schema with exactly one AssociationModel or SequenceModel element, representing the rule model contained in *SELF*.

6.3.5 DM_applyRuleModel Method

Purpose

This method computes the result of applying an association rule model to an item set.

Definition

```
CREATE METHOD DM_applyRuleModel
          (itemSet CHARACTER VARYING(DM_MaxItemSetLength))
   RETURNS TABLE (ITEM CHARACTER VARYING(DM_MaxItemLength),
                  SUPPORT REAL, CONFIDENCE REAL)
   FOR DM_RuleModel
   BEGIN
   --
   --!! See Description
   --
   END
```

Definitional Rules

1) *DM_ MaxItemSetLength* is the implementation-defined maximum number of charaters for a value of an item set.

2) *DM_ MaxItemLength* is the implementation-defined maximum number of charaters for an entry in the ITEM column of the table returned by DM_applyRuleModel.

Description

1) The method *DM_applyRuleModel* takes the following input parameter:

   a) a *CHARACTER VARYING* value *itemSet*.

2) The result of the invocation *DM_applyRuleModel(itemSet)* is determined as follow:

   Case:

   a) If *SELF.DM_getRuleMdlType()* equals 2, an exception condition is raised: *SQL/MM Data Mining exception – sequence model cannot be applied to item sets.*

   b) If *SELF.DM_getRuleMdlType()* equals 1 and *itemSet* is a valid representation of an input item set for rule model application, then a table containing a row for each item that can be inferred from *itemSet* using the association rule model contained in *SELF*.
   The value of *itemSet* is a valid representation of an item set input for rule model application if it is a well-formed XML element (see XML 1.0). The element shall match the declaration of type 'itemset' as defined by:
   ```
   <!ELEMENT Itemset (Item*) >
   <!ELEMENT Item    (#PCDATA) >
   ```

   The content of an element `item` is interpreted as a potential item of an association rule. The order of the elements is not relevant.

   c) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid application input data format.*

6.3.6 DM_getNumItemSets Method

Purpose

Returns the number of item sets contained in the DM_content of a value of DM_RuleModel.

Definition

```
CREATE METHOD DM_getNumItemSets ()
  RETURNS INTEGER
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumItemSets()* is determined as follows:

   Case:

   a) If *SELF.DM_content* is the null value, then the null value.

   b) Otherwise, the number of item sets contained in *SELF.DM_content*.

6.3.7 DM_getNumSequences Method

Purpose

Returns the number of sequences contained in the DM_content of a value of DM_RuleModel.

Definition

```
CREATE METHOD DM_getNumSequences()
  RETURNS INTEGER
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumSequences()* is determined as follows:

   Case:

   a) If *SELF.DM_content* is the null value, then the null value.

   b) If *SELF.*DM_getRuleType() = 1 (one), then 0 (zero).

   c) Otherwise, the number of sequences contained in *SELF.DM_content*.

6.3.8 DM_getNumRules Method

Purpose

Returns the number of rules contained in the DM_content of a value of DM_RuleModel.

Definition

```
CREATE METHOD DM_getNumRules ()
  RETURNS INTEGER
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumRules()* is determined as follows:

   Case:

   a) If *SELF.DM_content* is the null value, then the null value.

   b) Otherwise, the number of association rules or sequence rules contained in *SELF.DM_content*.

6.3.9 DM_filterRules Method

Purpose

This method returns a value of type DM_RuleModel, which contains a subset of the rules contained in SELF. The subset is generated according to the constraints specified in the input value of type DM_RuleFilter.

Definition

```
CREATE METHOD DM_filterRules(ruleFilter DM_RuleFilter)
  RETURNS DM_RuleModel
  FOR DM_RuleModel
  BEGIN
  --
  --!! See Description
  --
  END
```

Description

1) The method *DM_filterRules* takes the following input parameter:

   a) a *DM_RuleFilter* value *ruleFilter*.

2) The result of the invocation *DM_filterRules(ruleFilter)* is determined as follows:

   Case:

   a) *SELF*, if *ruleFilter* is the null value.

   b) Otherwise, a value of type *DM_RuleModel*, which contains a subset of the rules contained in *SELF*. Each rule in the result value conforms to the restrictions set forth in *ruleFilter*. And all rules in *SELF* conforming to the restrictions in *ruleFilter* are contained in the result.

6.3.10 DM_getNumTransacts Method

Purpose

Returns the number of transactions on which the DM_ruleModel value SELF is based.

Definition

```
CREATE METHOD DM_getNumTransacts ()
  RETURNS INTEGER
  FOR DM_RuleModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getNumTransacts()* is an *INTEGER* value giving the number of transactions on which *SELF* is based.

6.3.11 DM_getItemSets Method

Purpose

Returns a table of all item sets contained in SELF. For each item set, there are as many rows in that table as the set contains items. The support and the lift of the item sets are contained in the appropriate columns.

ITEMSETID is a number generated to identify the item sets and to group their member items.

SUPPORT is the support of the item set, i.e. the percentage of input item sets in which the particular item set is contained,

LIFT is the ratio between the actual support of the item set and the expected support. The expected support is the support when the occurrences of all items are statistically independent, so it can be calculated as the product of the items' supports.

ITEM is a member of an item set. All ITEM values with the same ID value make up an item set.

ITEMNAME is the display name of ITEM as defined by PMML. If none is contained in SELF, the value of ITEMNAME is equal to that of ITEM.

Definition

```
CREATE METHOD DM_getItemSets()
   RETURNS TABLE (ITEMSETID INTEGER, SUPPORT REAL, LIFT REAL,
                  ITEM CHARACTER VARYING(DM_MaxItemLength),
                  ITEMNAME CHARACTER VARYING(DM_MaxItemLength))
   FOR DM_RuleModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an item in an item name in an item set.

Description

1) The result of the invocation *DM_getItemsets()* is a table of item sets contained in *SELF*. For each item set, there are as many rows in the table as the set contains items.

6.3.12 DM_getRules Method

Purpose

Returns a table of association rules contained in SELF. There is a row in that table for each rule in SELF whose head consists of exactly one item. The length (total number of items), as well as the support, confidence and lift values of these rules are contained in the appropriate columns. ID is a number generated to identify the rules; HEADNAME holds the head item of the rule after application of a name mapping (if any), HEAD uses the original item before name mappings. BODYID is a generated identifier to allow joining the resulting table with a table generated by DM_getRuleBodies(). BODYTEXT is an implementation-dependent textual description of the rule body; it is provided for print-out purposes. LENGTH is a count of all items in the head and body, SUPPORT is the rule support, and CONFIDENCE is the rule confidence, both are percentage values between 0 and 100. LIFT indicates how much more confidence the rule has as might be expected if the items were statistically independent; the value is given as a positive factor.

For each complex rule, i.e. a rule with longer head, there is one table entry per head item representing a simple derived rule with the same body but only one of the head items. In this case, the length will be provided, but support, confidence and lift will have null values. All entries derived from the same complex rule will have the same rule id.

Definition

```
CREATE DM_getRules ()
  RETURNS TABLE (ID INTEGER,
                 HEADNAME CHARACTER VARYING(DM_MaxItemLength),
                 HEAD CHARACTER VARYING(DM_MaxItemLength),
                 BODYID INTEGER, LENGTH INTEGER,
                 BODYTEXT CHARACTER VARYING(DM_MaxBodyLength),
                 SUPPORT REAL, CONFIDENCE REAL, LIFT REAL)
  FOR DM_RuleModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an entry in the HEAD or HEADNAME columns of table *DM_getRules()*.

2) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for an entry in the BODYTEXT column of table *DM_getRules().*

Description

1) The result of the invocation *DM_getRules()* is determined as follows:

   Case:

   a) If *SELF.DM_getRuleMdlType()* equals 1 (one), then a table containing a row for each association rule contained in *SELF*, whose head consists of exactly one item. For each complex rule, i.e. a rule with longer head, there is one row for each of the head items.

   b) Otherwise, the null value.

6.3.13 DM_getRuleBodies Method

Purpose

Returns a table containing the association rule bodies contained in SELF.

Definition

```
CREATE METHOD DM_getRuleBodies ()
   RETURNS TABLE (BODYID INTEGER,
                  ITEMNAME CHARACTER VARYING(DM_MaxItemLength),
                  ITEM CHARACTER VARYING(DM_MaxItemLength))
   FOR DM_RuleModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Definitional Rules

1) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an entry in the ITEM or ITEMNAME columns of table *DM_getRuleBodies()*.

Description

1) The result of the invocation *DM_getRuleBodies()* is a table, which, for each distinct BODYID value in *SELF.DM_getRules(),* contains as many rows as there are items in that rule body.

6.3.14 DM_getSequences Method

Purpose

Returns a table of all sequences contained in SELF. For each sequence, there is one row in that table. The length, the total number of items, the support, the lift, the mean time difference between item sets in the sequence, the standard deviation of the time differences, and a textual version of the sequence are contained in the appropriate columns.

SEQID	is a number generated to identify the sequence,
NUMITEMSETS	is the number of item sets that make up the sequence,
NUMITEMS	is the total number of items in all items sets of the sequence,
SUPPORT	is the support of the sequence, i.e. the percentage of input sequences in which the particular sequence is contained,
LIFT	is the ratio between the actual support of the sequence and the expected support. The expected support is the support when the occurrences of all item sets are statistically independent, so it can be calculated as the product of the item set supports.
MEANTIMEDIFF	is the average time difference between the first and the last item set in the sequence with SEQID (total duration of the sequence). The average is computed over all sequences containing the sequence with SEQID as a subsequence.
STDDEVTIMEDIFF	is the standard deviation of the time differences between two subsequent steps in the sequence. The standard deviation is computed over all sequences containing the sequence with SEQID as a subsequence.
SEQTEXT	is a textual description of the sequence.

Definition

```
CREATE METHOD DM_getSequences ()
  RETURNS TABLE (SEQID INTEGER, NUMITEMSETS INTEGER, NUMITEMS INTEGER,
                 SUPPORT REAL, LIFT REAL,
                 MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL,
                 SEQTEXT CHARACTER VARYING(DM_MaxBodyLength))
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for the textual description of a sequence.

Description

1) The result of the invocation *DM_getSequences()* is determined as follows:

   Case:

   a) If *SELF.DM_getRuleMdlType()* equals 1 (one), then the null value.

   b) Otherwise, a table of sequences contained in *SELF*. For each sequence, there is a row in the resulting table.

6.3.15 DM_getSeqDetails Method

Purpose

Returns a table of detailed information about all sequences contained in SELF. For each sequence, there are as many rows in that table as the sequence contains item sets. An ID for each item set, its position, a textual representation of the item set, the average time difference between the items in each item set and the standard deviation thereof are contained in the appropriate columns.

SEQID	is an integer value identifying the sequence. The values coincide with the ID values generated by SELF.DM_getSequences().
ITEMSETID	is an integer value identifying an item set. The values coincide with the ID values generated by SELF.DM_getItemsets().
POSITION	is a positive integer indicating the position of the item set within the sequence.
ITEMSETTEXT	is a textual description of the item set with ITEMSETID.
MEANTIMEDIFF	is the average time difference to the predecessor item in the sequence. The average is computed over the same step in all sequences containing the sequence with SEQID as a subsequence. The value is null for the item set with position 1.
STDDEVTIMEDIFF	is the standard deviation of the time difference to the predecessor item in the sequence. The standard deviation is computed over the same step in all sequences containing the sequence with SEQID as a subsequence.

Definition

```
CREATE METHOD DM_getSeqDetails()
       RETURNS TABLE(SEQID INTEGER, POSITION INTEGER, ITEMSETID INTEGER,
                     ITEMSETTEXT CHARACTER VARYING(DM_MaxBodyLength),
                     MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL)
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for the textual description of an item set.

Description

1) The result of the invocation *DM_getSeqDetails()* is determined as follows:

   Case:

   a) If *SELF.DM_getRuleMdlType()* equals 1 (one), then the null value.

   b) Otherwise, a table with detailed information about all sequences contained in *SELF*.

6.3.16 DM_getSeqRules Method

Purpose

Returns a table of sequence rules contained in SELF. For each sequence rule, there is one row in that table. The support, confidence and lift of the sequence rules are contained in the appropriate columns. In addition, there is the length of the rule (number of item sets) and the total number of items, the mean time difference and the standard deviation thereof. Finally, textual representations of the rule bodies and heads are provided, as well as their ID's for reference.

SEQRULEID	is a number generated to identify the sequence rule,
BODYSEQID	is a number identifying the body sequence of the sequence rule,
HEADSEQID	is a number identifying the head sequence of the sequence rule,
BODYSEQTEXT	is a textual representation of the body sequence of the sequence rule.
HEADSEQTEXT	is a textual representations of the head sequence of the sequence rule.
NUMITEMSETS	is the number of item sets in the sequence rule.
NUMITEMS	is the total number of items in the sequence rule.
SUPPORT	is the support of the sequence rule, i.e. the percentage of input sequences in which both the body sequence and the head sequence are contained in that order,
CONFIDCENCE	is the confidence of the sequence rule, i.e. the percentage of input sequences in which both the body sequence and the head sequence are contained in that order with respect to the input sequences containing the body sequence.
LIFT	is the ratio between the confidence of the sequence rule and the support of the head sequence.
MEANTIMEDIFF	is the average time difference between the body sequence and the head sequence in the sequence rule. The average is computed over all input sequences containing the sequence rule.
STDDEVTIMEDIFF	is the average time difference between the body sequence and the head sequence in the sequence rule. The average is computed over all input sequences containing the sequence rule.

Definition

```
CREATE METHOD DM_getSeqRules ()
  RETURNS TABLE (SEQRULEID INTEGER,
                 BODYSEQID INTEGER, HEADSEQID INTEGER,
                 BODYSEQTEXT CHARACTER VARYING(DM_MaxBodyLength),
                 HEADSEQTEXT CHARACTER VARYING(DM_MaxBodyLength),
                 NUMITEMSETS INTEGER, NUMITEMS INTEGER,
                 SUPPORT REAL, CONFIDENCE REAL, LIFT REAL,
                 MEANTIMEDIFF REAL, STDDEVTIMEDIFF REAL)
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for the textual description of a sequence rule.

Description

1) The result of the invocation *DM_getSeqRules()* is determined as follows:

   Case:

   a) If *SELF.DM_getRuleMdlType()* equals 1 (one), then the null value.

   b) Otherwise, a table of sequence rules contained in *SELF*. For each sequence rule, there is one row in the table.

6.3.17 DM_getRuleBldTask method

Purpose

Return the *DM_RuleBldTask* value contained in the *DM_RuleModel* value.

Definition

```
CREATE METHOD DM_getRuleBldTask ()
  RETURNS DM_RuleBldTask
  FOR DM_RuleModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleBldTask()* is determined as follows:

   a) If *SELF* does not contain the information about the *DM_RuleBldTask* value that was used to compute *SELF*, then the null value.

   b) Otherwise, it is the *DM_RuleBldTask* value contained in *SELF* that was used to compute the model.

6.4 DM_RuleFilter Type and Routines

6.4.1 DM_RuleFilter Type

Purpose

The DM_RuleFilter type represents a specification of a subset of a given data mining rule model.

Definition

```
CREATE TYPE DM_RuleFilter
  AS (
        DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_addRangeConstr
         (restrictionCriterion INTEGER,
          lowerBound REAL, upperBound REAL, operator INTEGER)
  RETURNS DM_RuleFilter
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_remRangeConstr(restrictionCriterion INTEGER)
RETURNS DM_RuleFilter
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRangeConstrs()
  RETURNS TABLE(RESTRICTIONCRITERION INTEGER,
                LOWERBOUND REAL, UPPERBOUND REAL, OPERATOR INTEGER)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL

METHOD DM_setMaxNumRules (maxNumRules INTEGER)
  RETURNS DM_RuleFilter
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMaxNumRules()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_addCountConstr(preference INTEGER,
                         restrictionCriterion INTEGER, sense INTEGER)
RETURNS DM_RuleFilter
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```

```
METHOD DM_getCountConstrs()
   RETURNS TABLE(PREFERENCE INTEGER,
                 RESTRICTIONCRITERION INTEGER, SENSE INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL

METHOD DM_remCountConstr(preference INTEGER)
RETURNS DM_RuleFilter
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_setItemConstr
          (item CHARACTER VARYING(DM_MaxItemLength),
           constraintType INTEGER, disjunctiveGroup INTEGER)
   RETURNS DM_RuleFilter
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getItemConstrs()
   RETURNSTABLE(ITEM CHARACTER VARYING(DM_MaxItemLength),
                CONSTRAINTTYPE INTEGER, disjunctiveGroup INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RuleFilter*.

2) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an item in a constraint specification for a rule filter.

Description

1) The *DM_RuleFilter* type provides for public use:

   a) a method *DM_addRangeConstr(INTEGER, REAL, REAL, INTEGER),*

   b) a method *DM_remRangeConstr(INTEGER),*

   c) a method *DM_getRangeConstrs(),*

   d) a method *DM_setMaxNumRules(INTEGER),*

   e) a method *DM_getMaxNumRules(),*

   f) a method *DM_addCountConstr(INTEGER, INTEGER, INTEGER),*

   g) a method *DM_remCountConstr(INTEGER),*

   h) a method *DM_getCountConstrs(),*

   i) a method *DM_setItemConstr(CHARACTER VARYING(DM_MaxItemLength), INTEGER, INTEGER),*

   j) a method *DM_getItemConstrs().*

6.4.2 DM_addRangeConstr Method

Purpose

This method sets the boundary values for some property of SELF. Rule models created with the use of the resulting filter will contain only item sets, sequences and rules, whose values for that property are within the given range.

Definition

```
CREATE METHOD DM_addRangeConstr
                    (restrictionCriterion INTEGER,
                     lowerBound REAL, upperBound REAL, operator INTEGER)
 RETURNS DM_RuleFilter
 FOR DM_RuleFilter
 BEGIN
 --
 --!! See Description
 --
 END
```

Description

1) The method *DM_addRangeConstr* takes the following input parameters:

   a) An *INTEGER* value *restrictionCriterion*,

   b) a *REAL* value *lowerBound,*

   c) a *REAL* value *upperBound,*

   d) an *INTEGER* value *operator*.

2) The following numerical values for the input values are defined:

Operator	Code for operator	Description
is in closed	1 (one)	All values between the boundaries are allowed, including the boundaries.
is in open	2	All values between the boundaries are allowed, not including the boundaries.
not in closed	-1	All values outside the boundaries are allowed, not including the boundaries.
not in open	-2	All values outside the boundaries are allowed, including the boundaries.

Table 6 — Values for operators

Operator shall be one of the values 1, 2, -1 or -2 with the meaning as described in Table 6 — Values for operators.

Criterion	Code for criterion	Minimum value	Maximum value
support	1 (one)	0 (zero)	100
confidence	2	0 (zero)	100
lift	3	0 (zero)	unbounded
support * confidence	4	0 (zero)	10,000
number of items	5	2	unbounded
step time	6	0 (zero)	unbounded
total time	7	0 (zero)	unbounded

Table 7 — Values for restriction criteria

RestrictionCriterion shall be a value between 1 and 7. For each of these values, *minimum* and *maximum* shall be between the minimum and maximum values in Table 7 — Values for restriction criteria.

3) The result of the invocation *DM_addRangeConstr(restrictionCriterion, lowerBound, upperBound, operator)* is determined as follows:

Case:

a) If *operator* is not 1, 2, -1 or -2, *restrictionCriterion* is not positive or *restrictionCriterion* > 7, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

b) If *lowerBound* > *upperBound* or either *lowerBound* or *upperBound* is outside the legal range defined for *restrictionCriterion* in Table 7 — Values for restriction criteria above, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

c) Otherwise, a value v of type *DM_RuleFilter* that is identical to *SELF* except that “select LOWERBOUND, UPPERBOUND, OPERATOR from (table v.*DM_getRangeConstrs()* as T) where RESTRICTIONCRITERION = *restrictionCriterion*” is “*lowerBound, upperBound, operator*”.

6.4.3 DM_remRangeConstr Method

Purpose

This method removes the range constraint for a given restriction criterion.

Definition

```
CREATE METHOD DM_remRangeConstr(restrictionCriterion INTEGER)
  RETURNS DM_RuleFilter
  FOR DM_RuleFilter
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_ remRangeConstr* takes the following input parameter:

   a) An *INTEGER* value *restrictionCriterion*.

2) The numerical values for *restrictionCriterion* are defined in Table 7 — Values for restriction criteria on page 90.

3) The result of the invocation *DM_remRangeConstr(restrictionCriterion)* is a value v of type *DM_RuleFilter* that is identical to *SELF* except that “select RESTRICTIONCRITERION from (table v.*DM_getRangeConstrs()* as T) where RESTRICTIONCRITERION = *restrictionCriterion*” is empty.

6.4.4 DM_getRangeConstrs Method

Purpose

This method returns the range constraints defined in SELF.

Definition

```
CREATE METHOD DM_getRangeConstrs()
  RETURNS TABLE (RESTRICTIONCRITERION INTEGER, LOWERBOUND REAL,
                 UPPERBOUND REAL, OPERATOR INTEGER)
  FOR DM_RuleFilter
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRangeConstrs()* is a table containing a row for each restriction criterion, for which a range constraint is contained in *SELF*. The table has four columns holding the restriction criterion, the lower and upper bound, as well as the operator.

6.4.5 DM_setMaxNumRules Method

Purpose

This method sets the maximum number of rules for SELF. Rule models created with the use of this filter will contain at most that number of rules. If more rules are available, rules are selected according to the selection criteria used in count constraints in SELF.

Definition

```
CREATE METHOD DM_setMaxNumRules(maxNumRules INTEGER)
  RETURNS DM_RuleFilter
  FOR DM_RuleFilter
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setMaxNumRules* takes the following input parameters:

   a) An *INTEGER* value *maxNumRules*.

2) The result of the invocation *DM_setMaxNumRules(maxNumRules)* is determined as follows:

   Case:

   a) If *maxNumRules* is the null value, then a value of type *DM_RuleFilter* that contains no upper bound for the number of rules and is otherwise identical to *SELF*.

   b) If *maxNumRules* is not positive, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   c) Otherwise, a value v of type *DM_RuleFilter* that is identical to *SELF* except that the method invocation v.*DM_getMaxNumRules() = maxNumRules*.

6.4.6 DM_getMaxNumRules Method

Purpose

This method returns the value for the maximum number of rules of SELF.

Definition

```
CREATE METHOD DM_getMaxNumRules()
  RETURNS INTEGER
  FOR DM_RuleFilter
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getMaxNumRules()* is determined as follows:

   Case:

   a) If *SELF=SELF.DM_setMaxNumRules(v)* for some *INTEGER* value v, then v.

   b) Otherwise, the null value, indicating that no upper bound for the number of rules is contained in *SELF*.

6.4.7 DM_addCountConstr Method

Purpose

This method adds a count constraint for a given restriction criterion.

Definition

```
CREATE METHOD DM_addCountConstr
                (preference INTEGER, restrictionCriterion INTEGER,
                 ordering INTEGER)
  RETURNS DM_RuleFilter
  FOR DM_RuleFilter
  BEGIN
   --
   -- !! See Description
   --
  END
```

Description

1) The method *DM_addCountConstr* takes the following input parameters:
   a) An *INTEGER* value *preference*,
   b) An *INTEGER* value *restrictionCriterion*,
   c) An *INTEGER* value *ordering*.

2) The numerical values for *restrictionCriterion* are defined in Table 7 — Values for restriction criteria on page 90.

3) The result of the invocation *DM_addCountConstr(preference, restrictionCriterion, ordering)* is determined as follows:

   Case:

   a) If *preference* is not positive, *ordering is not 1 (one) or -1 (minus 1), restrictionCriterion* is not one of the values defined in table **tref** on page **pref**, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.
   b) Otherwise, a value v of type *DM_RuleFilter* that is identical to *SELF* except that “select PREFERENCE, ORDERING from (table v.*DM_getCountConstrs()* as T) where RESTRICTIONCRITERION = *restrictionCriterion*” is “*preference, ordering*”.

6.4.8 DM_remCountConstr Method

Purpose

This method removes the count constraint for a given restriction criterion.

Definition

```
CREATE METHOD DM_remCountConstr(restrictionCriterion INTEGER)
   RETURNS DM_RuleFilter
   FOR DM_RuleFilter
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The method *DM_remCountConstr* takes the following input parameter:

   a) An *INTEGER* value *restrictionCriterion*.

2) The numerical values for *restrictionCriterion* are defined in Table 7 — Values for restriction criteria on page 90.

3) The result of the invocation *DM_remCountConstr(restrictionCriterion)* is a value v of type *DM_RuleFilter* that is identical to *SELF* except that “select PREFERENCE from (table v.*DM_getCountConstrs()* as T) where RESTRICTIONCRITERION = *restrictionCriterion*” is empty.

6.4.9 DM_getCountConstrs Method

Purpose

This method removes the count constraint for a given restriction criterion.

Definition

```
CREATE METHOD DM_getCountConstrs(restrictionCriterion INTEGER)
  RETURNS TABLE(PREFERENCE INTEGER, RESTRICTIONCRITERION INTEGER,
                ORDERING INTEGER)
  FOR DM_RuleFilter
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRangeConstrs()* is a table containing a row for each restriction criterion, for which a range constraint is contained in *SELF*. The table has three columns holding the preference, the restriction criterion and the ordering.

6.4.10 DM_setItemConstr Method

Purpose

This method sets a constraint for SELF regarding a particular item. Rule models created with the use of this filter will conform to this item constraint.

For item constraints within the same disjunctive group, at least one of the conditions shall hold. The constraints defined in different disjunctive groups or defined with null as the disjunctive group shall hold simultaneously.

Definition

```
CREATE METHOD DM_setItemConstr
         (item CHARACTER VARYING(DM_MaxItemLength),
          constraintType INTEGER, disjunctiveGroup INTEGER)
  RETURNS DM_RuleFilter
  FOR DM_RuleFilter
  BEGIN
  --
  --!! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an item in an item constraint specification for an association rule filter.

Description

1) The method *DM_setItemConstr* takes the following input parameters:

   a) a *CHARACTER VARYING* value *item*,

   b) an *INTEGER* value *constraintType,*

   c) an *INTEGER* value *disjunctiveGroup*.

2) The following values for constraint types are defined:

Constraint type	Code for constraint type
shall appear in body	1 (one)
shall appear in head	2
shall appear in rule	3
shall not appear in body	-1
shall not appear in head	-2
shall not appear in rule	-3

Table 8 — Values for constraint types

3) The result of the invocation *DM_setItemConstr(item, constraintType, disjunctiveGroup)* is determined as follows:

   Case:

a) If *constraintType* is 0 (zero) or the absolute value of *constraintType* > 3, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

b) Else, if *constraintType, item and disjunctiveGroup* are all null, then a value v of type *DM_ruleFilter* with no item constraints, such that "select count(*) from (table v.*DM_getItemConstrs()* as T)" equals 0 (zero)..

c) Else, if *constraintType and item* are null, then a value v of type *DM_ruleFilter* with no item constraints in *disjunctiveGroup*, such that "select count(*) from (table v.*DM_getItemConstrs()* as T) where DISJUNCTIVEGROUP = *disjunctiveGroup"* equals 0 (zero).

d) Else, if *constraintType and disjunctiveGroup* are null, then a value v of type *DM_ruleFilter* with no independent item constraint for *item*, such that "select count(*) from (table v.*DM_getItemConstrs()* as T) where ITEM = *item* and DISJUNCTIVEGROUP = null*"* equals 0 (zero).

e) Else, if *item* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

f) Else, if *constraintType* is the null value, then a value v of type *DM_ruleFilter* with no item constraint for *item* in *disjunctiveGroup*, such that "select CONSTRAINTTYPE from (table v.*DM_getItemConstrs()* as T) where ITEM = *item* and DISJUNCTIVEGROUP = *disjunctiveGroup*" equals 0 (zero).

g) Otherwise, a value v of type *DM_RuleFilter* such that "select CONSTRAINTTYPE, DISJUNCTIVEGROUP from (table v.*DM_getItemConstrs()* as T) where ITEM = *item*" returns "*constraintType, disjunctiveGroup"* and *SELF.DM_getItemConstrs()* is otherwise equal to v.*DM_getItemConstrs().*

4) Note that for each item, there is an implicit disjunctive group "null(item)", which consists only of this item. Two item constraints with null as their disjunctive group, but with different items, are independent item constraints and shall both be satisfied.

5) *Overview of results:*

DM_setItemConstr(item, constraintType, disjunctiveGroup)	sets item constraint within disjunctive group
DM_setItemConstr(item, constraintType, null)	sets independent constraint for item
DM_setItemConstr(item, null, disjunctiveGroup)	removes item from disjunctive group
DM_setItemConstr(null, constraintType, disjunctiveGroup)	error
DM_setItemConstr(item, null, null)	removes item from its null group
DM_setItemConstr(null, constraintType, null)	error
DM_setItemConstr(null, null, disjunctiveGroup)	removes disjunctive group
DM_setItemConstr(null, null, null)	removes all item constraints

6.4.11 DM_getItemConstrs Method

Purpose

This method returns a table containing all item constraints contained in SELF.

Definition

```
CREATE METHOD DM_getItemConstrs()
  RETURNS TABLE (ITEM CHARACTER VARYING(DM_MaxItemLength),
                 CONSTRAINTTYPE INTEGER, DISJUNCTIVEGROUP)
  FOR DM_RuleFilter
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxItemLength* is the implementation-defined maximum number of characters for an item in an item constraint specification for an association rule filter.

Description

1) The result of the invocation *DM_getItemConstrs()* is a table containing a row for each item constraint contained in *SELF*. The table has three columns holding the item, the constraint type and the disjunctive group, respectively.

2) Note that two item constraints with null as their disjunctive group, but with different items, are independent item constraints and shall both be satisfied.

6.5 DM_RuleApplTask Type and Routines

6.5.1 DM_RuleApplTask Type

Purpose

The type DM_RuleApplTask is a representation of all the information that constitutes an association rule application task; in particular it contains the mining model and the data mining data.

Definition

```
CREATE TYPE DM_RuleApplTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defRuleApplTask
          (model DM_RuleModel, inputData DM_MiningData,
           groupField CHARACTER VARYING(DM_MaxFieldAliasLength),
           outputData DM_MiningData)
  RETURNS DM_RuleApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleMdl()
  RETURNS DM_RuleModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleApplData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleGroupFld()
  RETURNS CHARACTER VARYING
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRuleOutpData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_applyRuleMdl()
  RETURNS DM_RuleApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a *DM_RuleApplTask*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a filed name within a logical data specification.

Description

1) The type *DM_RuleApplTask* provides for public use:

   a) a method *DM_defRuleApplTask(DM_RuleModel, DM_MiningData, CHARACTER VARYING, DM_MiningData),*

   b) a method *DM_getRuleMdl(),*

   c) a method *DM_getRuleApplData(),*

   d) a method *DM_getRuleOutpData(),*

   e) a method *DM_applyClusMdl().*

6.5.2 DM_defRuleApplTask Method

Purpose

Return the DM_RuleApplTask value containing the given DM_RuleModel value and the given DM_MiningData values for input and output.

Definition

```
CREATE STATIC METHOD DM_defRuleApplTask
          (model DM_RuleModel, inputData DM_MiningData,
           groupField CHARACTER VARYING(MaxFieldAliasLength),
           outputData DM_MiningData)
  RETURNS DM_RuleApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_defRuleApplTask* takes the following input parameters:

   a) a *DM_RuleModel* value *model*.

   b) a *DM_MiningData* value *inputData*,

   c) a *CHARACTER VARYING* value *groupField*,

   d) a *DM_MiningData* value *outputData*,

2) The result of the invocation *DM_defRuleApplTask(model, inputData, outputData)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *model.DM_getRuleMdlType()* equals 2, an exception condition is raised: *SQL/MM Data Mining exception – sequence model cannot be applied to item sets.*

   c) If *inputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   d) If no field with name *groupField* is contained in the logical data specification *inputData.genDataSpec(),* then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   e) If a field with name *groupField* is contained in the logical data specification *inputData.genDataSpec()* and the field type of that field is not categorical, then an exception condition is raised: *SQL/MM Data Mining exception – field not categorical*.

f) If *outputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null output data*.

g) If “GROUP” is not the alias of a field of type *CHARACTER VARYING* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field.*

h) If “ITEM” is not the alias of a field of type *CHARACTER VARYING* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field.*

i) If “SUPPORT” is not the alias of a field of type *REAL* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field.*

j) If “CONFIDENCE” is not the alias of a field of type *REAL* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field.*

k) Otherwise, the *DM_RuleApplTask* value containing the given arguments.

6.5.3 DM_getRuleMdl Method

Purpose

Return the DM_RuleModel value representing the association rule model contained in the association rule application task.

Definition

```
CREATE METHOD DM_getRuleMdl()
   RETURNS DM_RuleModel
   FOR DM_RuleApplTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRuleMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_RuleModel* value, then the null value.

   b) Otherwise, it is the *DM_RuleModel* value contained in *SELF*.

6.5.4 DM_getRuleApplData Method

Purpose

Return the DM_MiningData value for input contained in SELF.

Definition

```
CREATE METHOD DM_getRuleApplData()
  RETURNS DM_MiningData
  FOR DM_RuleApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleApplData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for input contained in *SELF*.

6.5.5 DM_getRuleOutpData Method

Purpose

Return the DM_MiningData value for output contained in SELF.

Definition

```
CREATE METHOD DM_getRuleOutpData()
  RETURNS DM_MiningData
  FOR DM_RuleApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRuleOutpData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for output, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for output contained in *SELF*.

6.5.6 DM_getRuleGroupFld method

Purpose

Return the CHARACTER VARYING value for the group field contained in SELF.

Definition

```
CREATE METHOD DM_getRuleGroupFld()
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  FOR DM_RuleApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getRuleGroupFld()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *CHARACTER VARYING* value for the group field, then the null value.

   b) Otherwise, it is the *CHARACTER VARYING* value for the group field contained in *SELF*.

6.5.7 DM_applyRuleMdl method

Purpose

Apply the association rule model in the DM_RuleApplTask value to each set of items in the table determined by the input data in the DM_RuleApplTask value. The item sets are formed by all items with the same value in the group field.

Definition

```
CREATE METHOD DM_applyRuleMdl()
   RETURNS DM_RuleApplTask
   FOR DM_RuleApplTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_applyRuleMdl()* is determined as follows:

   Case:

   a) If the application run is successful, then SELF.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model application failed*.

2) In a successful application run, for each item set of the table determined by the input data mining data contained in *SELF* zero or more rows are inserted into the table determined by the output data mining data contained in *SELF*. The column values are determined as follows:

   a) The values for the column “GROUP” are the values from *SELF.DM_getRuleOutpData()* whose alias name is *SELF.DM_getRuleGroupFld().* Rows with identical values G in that column contain inferred items from the input item set corresponding to the stem set with group column value G,

   b) The values for the column “ITEM” are the item values inferred from *SELF.DM_getRuleMdl()* for the transaction defined by the value in the column “GROUP”,

   c) The values for the column “SUPPORT” indicate the support for the value in the column “ITEM”,

   d) The values for the column “CONFIDENCE” indicate the confidence for the value in the column “ITEM”.

Blank page

7 Clustering

7.1 DM_ClusSettings Type and Routines

7.1.1 DM_ClusSettings Type

Purpose

The DM_ClusSettings type is the description for the settings that are used to generate a clustering model.

Definition

```
CREATE TYPE DM_ClusSettings
  AS (
        DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impClusSettings
         (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClusSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expClusSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setMaxNumClus (maxNumClusters INTEGER)
  RETURNS DM_ClusSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMaxNumClus()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setFldWeight
         (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength),
          weight DOUBLE PRECISION)
  RETURNS DM_ClusSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getFldWeight
         (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD DM_setFldSimScale
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength),
           similarityScale DOUBLE PRECISION)
   RETURNS DM_ClusSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getFldSimScale
          (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength)
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_useClusDataSpec(logicalDataSpecification DM_LogicalDataSpec)
   RETURNS DM_ClusSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClusDataSpec()
   RETURNS DM_LogicalDataSpec
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusSettings*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_ClusSettings* type provides for public use:

   a) a method *DM_impClusSettings (CHARACTER LARGE OBJECT(DM_MaxContentLength))*,

   b) a method *DM_expClusSettings*(),

   c) a method *DM_setMaxNumClus(INTEGER),*

   d) a method *DM_getMaxNumClus(),*

   e) a method *DM_setFldWeight(CHARACTER VARYING, DOUBLE PRECISION),*

   f) a method *DM_getFldWeight(CHARACTER VARYING),*

   g) a method *DM_setFldSimScale(CHARACTER VARYING, DOUBLE PRECISION),*

   h) a method *DM_getFldSimScale(CHARACTER VARYING),*

   i) a method *DM_useClusDataSpec(DM_LogicalDataSpec),*

   j) a method *DM_getClusDataSpec().*

7.1.2 DM_impClusSettings Method

Purpose

Return a specified value of type DM_ClusSettings.

Definition

```
CREATE STATIC METHOD DM_impClusSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClusSettings
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusSettings*.

Description

1) The method *DM_impClusSettings* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputSettings*. The format of this value is implementation-dependent, but for any one implementation the format is the same as that used by *DM_expClusSettings*.

2) The result of the invocation *DM_impClusSettings(inputSettings*) is determined as follows:

   Case:

   a) If there is a value v of type *DM_ClusSettings* such that *inputSettings* = v.*DM_expClusSettings(),* then v.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid settings format*.

7.1.3 DM_expClusSettings Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the clustering settings contained in the DM_content value of the DM_ClusSettings value.

Definition

```
CREATE METHOD DM_expClusSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusSettings*.

Description

1) The result of the invocation *DM_expClusSettings()* is a *CHARACTER LARGE OBJECT* value representing the clustering settings contained in *SELF*. This representation is implementation-dependent.

7.1.4 DM_setMaxNumClus Method

Purpose

Specifies the maximum number of clusters for a DM_ClusSettings value.

Definition

```
CREATE METHOD DM_setMaxNumClus(maxNumClusters INTEGER)
  RETURNS DM_ClusSettings
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setMaxNumClus* takes the following input parameter:

   a) an *INTEGER* value *maxNumClusters*.

2) The result of the invocation *DM_setMaxNOClus(maxNumClusters)* is determined as follows:

   Case:

   a) If *maxNumClusters* is the null value, then a value of type *DM_ClusSettings* containing no upper limit for the number of clusters.

   b) If *maxNumClusters* is not strictly positive, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   c) Otherwise, it is the *DM_ClusSettings* value containing *maxNumClusters* as the upper limit for the number of clusters.

7.1.5 DM_getMaxNumClus Method

Purpose

Return the maximum number of clusters specified in SELF.

Definition

```
CREATE METHOD DM_getMaxNumClus()
  RETURNS INTEGER
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getMaxNumClus()* is determined as follows:

   a) If *SELF* is equal to *SELF.DM_setMaxNumClus(*v*)* for some INTEGER value v, then v.

   b) Otherwise, it is the null value.

7.1.6 DM_setFldWeight Method

Purpose

Specifies the weight for a given field of a DM_ClusSettings value.

Definition

```
CREATE METHOD DM_setFldWeight
          (fieldName CHARACTER VARYING(DM_MaxAliasNameLength)
           weight DOUBLE PRECISION)
  RETURNS DM_ClusSettings
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setFldWeight* takes the following input parameters:

   a) a *CHARACTER VARYING* value *fieldname*,

   b) a *DOUBLE PRECISION* value *weight*.

2) The result of the invocation *DM_setFldWeight(fieldname, weight)* is determined as follows:

   Case:

   a) If *fieldName* is the null value, then *SELF*.

   b) If *SELF.DM_getClusDataSpec()* = ds (not null) and no field with name *fieldName* is contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) If *weight* is the null value, then a value of type *DM_ClusSettings* with weight 1.0 (default value) for the field *fieldName* but otherwise identical to *SELF*.

   d) If *weight* is not strictly positive, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   e) Otherwise, a value v of type *DM_ClusSettings* that is identical to *SELF* except that the method invocation v.*DM_getFldWeight(fieldname) = weight.*

7.1.7 DM_getFldWeight Method

Purpose

Returns the weight of a given field of a DM_ClusSettings value.

Definition

```
CREATE METHOD DM_getFldWeight
          (fieldName CHARACTER VARYING(DM_MaxAliasNameLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_getFldWeight* takes the following input parameter:

   a) a *CHARACTER VARYING* value *fieldname*.

2) The result of the invocation *DM_getFldWeight(fieldname)* is determined as follows:

   Case:

   a) If *SELF.DM_getClusDataSpec()* is the null value or *fieldName* is not equal to the name of any field contained in the logical data specification of *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   b) If *SELF* is equal to *SELF.DM_setFldWeight(fieldname,* v*)* for some *DOUBLE PRECISION* value *v* such that v is not null, then *v*.

   c) Otherwise, 1.0 (default weight).

7.1.8 DM_setFldSimScale Method

Purpose

Specifies the similarity scale for a given field of a DM_ClusSettings value. If two values are one similarity scale apart, they are considered to have medium similarity, i.e. their similarity becomes 0.5 in a range from 0 to 1.

Definition

```
CREATE METHOD DM_setFldSimScale
          (fieldName CHARACTER VARYING(DM_MaxAliasNameLength)
           similarityScale DOUBLE PRECISION)
  RETURNS DM_ClusSettings
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setFldSimScale* takes the following input parameters:

   a) a *CHARACTER VARYING* value *fieldname*,

   b) a *DOUBLE PRECISION* value *similarityScale*.

2) The result of the invocation *DM_setFldSimScale(fieldname, similarityScale)* is determined as follows:

   Case:

   a) If *fieldName* is the null value, then *SELF*.

   b) If *SELF.DM_getClusDataSpec()* = ds (not null) and no field with name *fieldName* is contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) If *SELF.DM_getClusDataSpec()* = ds (not null), *fieldName* is contained in ds and ds.*DM_getFldType(fieldName)* = 0 (zero), then an exception condition is raised: *SQL/MM Data Mining exception – field not numerical*.

   d) If *similarityScale* is the null value, then a value of type *DM_ClusSettings* with no similarity scale for the field *fieldName,* indicating that a default numerical value for the similarity scale will be determined during the data mining run.

   e) If *similarityScale* is not strictly positive, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   f) Otherwise, a value v of type *DM_ClusSettings* that is identical to *SELF* except that the method invocation v.*DM_getFldSimScale(fieldname) = similarity.*

7.1.9 DM_getFldSimScale Method

Purpose

Returns the similarity scale of a given field of a DM_ClusSettings value.

Definition

```
CREATE METHOD DM_getFldSimScale
          (fieldName CHARACTER VARYING(DM_MaxAliasNameLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_getFldSimScale* takes the following input parameter:

   a) a *CHARACTER VARYING* value *fieldname*,

2) The result of the invocation *DM_getFldSimScale(fieldname)* is determined as follows:

   Case:

   a) If *SELF.DM_getClusDataSpec()* is the null value or *fieldName* is not equal to the alias name of any field contained in the logical data specification of *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   b) If *SELF* is equal to *SELF.DM_setFldSimScale(fieldname,* v*)* for some positive *DOUBLE PRECISION* value v, then v.

   c) Otherwise, the null value, indicating that a default numerical value for the similarity scale will be determined during the data mining run.

7.1.10 DM_useClusDataSpec Method

Purpose

Specify a DM_LogicalDataSpec value for the DM_ClusSettings value. The DM_LogicalDataSpec value determines valid DM_MiningData values, which can be processed in a training run.

Definition

```
CREATE METHOD DM_useClusDataSpec
          (logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_ClusSettings
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_useClusDataSpec* takes the following input parameter:

   a) a *DM_LogicalDataSpec* value *logicalDataSpecification*.

2) The result of the invocation *DM_useClusDataSpec (logicalDataSpecification)* is determined as follows:

   Case:

   a) If *logicalDataSpecification* is the null value, then a *DM_ClusSettings* value containing no logical data specification but otherwise identical to *SELF*.

   b) If, for some *CHARACTER VARYING* value fieldName, *SELF.DM_getFldWeight*(fieldName) is not 1.0 and fieldName is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   c) If, for some *CHARACTER VARYING* value fieldName, *SELF.DM_getFldSimScale*(fieldName) = s (not null) and fieldName is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   d) Otherwise, it is the *DM_ClusSetting*s value containing *logicalDataSpecification* as the logical data specification.

7.1.11 DM_getClusDataSpec Method

Purpose

Returns the logical data specification defined for a clustering settings.

Definition

```
CREATE METHOD DM_getClusDataSpec()
  RETURNS DM_LogicalDataSpec
  FOR DM_ClusSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusDataSpec()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_useClusDataSpec*(v) for some *DM_LogicalDataSpec* value v, then v.

   b) Otherwise, the null value.

7.2 DM_ClusBldTask type and Routines

7.2.1 DM_ClusBldTask Type

Purpose

The type DM_ClusBldTask represents the information about a clustering task, in particular the input data and the parameter settings. Furthermore, it provides a method for computing a clustering model.

Definition

```
CREATE TYPE DM_ClusBldTask
   AS (
          DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
   )
   INSTANTIABLE
   NOT FINAL

STATIC METHOD DM_defClusBldTask
           (inputData DM_MiningData, settings DM_ClusSettings)
   RETURNS DM_ClusBldTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClusTrnData()
   RETURNS DM_MiningData
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_getClusSettings()
   RETURNS DM_ClusSettings
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_buildClusModel()
   RETURNS DM_ClusteringModel
   LANGUAGE SQL
   NOT DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusBldTask*

Description

1) The type *DM_ClusBldTask* provides for public use:

   a) a method *DM_defClusBldTask(DM_MiningData, DM_ClusSettings)*,

   b) a method *DM_getClusTrnData()*,

   c) a method *DM_getClusSettings()*,

   d) a method *DM_buildClusModel().*

7.2.2 DM_defClusBldTask Method

Purpose

Return the DM_ClusBldTask value determined by both the given DM_MiningData value and the given DM_ClusSettings value.

Definition

```
CREATE STATIC METHOD DM_defClusBldTask
          (inputData DM_MiningData, settings DM_ClusSettings)
  RETURNS DM_ClusBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defClusBldTask* takes the following input parameters:

   a) a *DM_MiningData* value *inputData*,

   b) a *DM_ClusSettings* value *settings*.

2) The result of the invocation *DM_defClusBldTask(inputData, settings)* is determined as follows:

   Case:

   a) If *settings* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null settings*.

   b) If *inputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *settings.DM_getClusDataSpec().isCompatible(trainData.DM_genDataSpec())* equals 1 (one), then a *DM_ClusBldTask* value bt such that bt.*DM_getClusTrnData()* equals *trainData* and bt.*DM_getClusSettings()* equals *settings*.

   d) Otherwise, an exception condition is raised: SQL/MM Data Mining exception – data and data specification not compatible.

7.2.3 DM_getClusTrnData Method

Purpose

Return the DM_MiningData value representing the training data for the clustering task.

Definition

```
CREATE METHOD DM_getClusTrnData()
   RETURNS DM_MiningData
   FOR DM_ClusBldTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getClusTrnData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value representing the input data to be used for training.

7.2.4 DM_getClusSettings Method

Purpose

Return the DM_ClusSettings value representing the settings of the clustering task.

Definition

```
CREATE METHOD DM_getClusSettings()
  RETURNS DM_ClusSettings
  FOR DM_ClusBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusSettings()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_ClusSettings* value, then the null value.

   b) Otherwise, it is the *DM_ClusSettings* value representing the settings to be used for training.

7.2.5 DM_buildClusModel Method

Purpose

Return the DM_ClusteringModel value representing the result of a clustering training run given SELF.

Definition

```
CREATE METHOD DM_buildClusModel()
   RETURNS DM_ClusteringModel
   FOR DM_ClusBldTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_buildClusModel()* is determined as follows:

   Case:

   a) If the run of this mining function is successful, then an implementation-dependent value of type *DM_ClusteringModel* representing the result of the clustering run and recalling the settings and input data that were used.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model computation failed.*

7.3 DM_ClusteringModel Type and Routines

7.3.1 DM_ClusteringModel Type

Purpose

The DM_ClusteringModel type represents models that are the result of clustering.

Definition

```
CREATE TYPE DM_ClusteringModel
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impClusModel
         (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClusteringModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expClusModel ()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getNumClusters ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getClusters ()
  RETURNS TABLE (ID INTEGER,
                 NAME CHARACTER VARYING(DM_MaxClusterNameLength),
                 SIZE BIGINT, HOMOGENEITY REAL)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_applyClusModel (inputRow DM_ApplicationData)
  RETURNS DM_ClusResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```



```
METHOD DM_getClusTask ()
   RETURNS DM_ClusBldTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClusMdlSpec ()
   RETURNS DM_LogicalDataSpec
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT

METHOD DM_getFields ()
   RETURNS TABLE (COLNAME VARCAHR(DM_MaxFieldNameLength),
                  FIELDNAME VARCAHR(DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusteringModel*.

2) *DM_MaxClusterNameLength* is the implementation-defined maximum number of characters of a cluster name.

Description

1) The *DM_ClusteringModel* provides for public use:

   a) a method *DM_impClusModel(CHARACTER LARGE OBJECT(DM_MaxContentLength))*,

   b) a method *DM_expClusModel(),*

   c) a method *DM_getNumClusters(),*

   d) a method *DM_getClusters(),*

   e) a method *DM_getMdlQuality(),*

   f) a method *DM_applyClusModel(DM_ApplicationData),*

   g) a method *DM_getClusTask(),*

   h) a method *DM_getClusMdlSpec(),*

   i) a method *DM_getFields().*

7.3.2 DM_impClusModel Method

Purpose

Return the DM_ClusteringModel value determined by the given string.

Definition

```
CREATE STATIC METHOD DM_impClusModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClusteringModel
  FOR DM_ClusteringModel
  BEGIN
  --
  --!! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusteringModel*.

Description

1) The method *DM_impClusModel* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputModel*.

2) The result of the invocation *DM_impClusModel*(*inputModel*) is determined as follows:

   Case:

   a) If *inputModel* contains an XML document defined by the PMML schema with exactly one ClusteringModel element, then a value of type *DM_ClusteringModel* representing the given PMML model.

   b) If *inputModel* contains an XML document defined by the PMML schema with more than one ClusteringModel element, then a value of type *DM_ClusteringModel* representing the first of the models.

   c) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid import format*.

7.3.3 DM_expClusModel Method

Purpose

Return a string representing the clustering model of the DM_ClusteringModel value.

Definition

```
CREATE METHOD DM_expClusModel ()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  FOR DM_ClusteringModel
  BEGIN
  --
  --!! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusteringModel*.

Description

1) The result of the invocation *DM_expClusModel ()* is a *CHARACTER LARGE OBJECT* value that contains an XML document conforming to the PMML schema with exactly one ClusteringModel element, representing the clustering model contained in *SELF*.

7.3.4 DM_getNumClusters Method

Purpose

Returns the number of clusters contained in the DM_content of a value of type DM_ClusteringModel.

Definition

```
CREATE METHOD DM_getNumClusters ()
  RETURNS INTEGER
  FOR DM_ClusteringModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumClusters()* is the number of clusters contained in *SELF.DM_content.*

7.3.5 DM_getClusters Method

Purpose

Returns a table of clusters contained in SELF. For each cluster, its size, name and homogeneity are contained in the table. The cluster size is given as an integer value, its name is an implementation-defined string characterizing the cluster, and its homogeneity is a value between 0 and 1 measuring how similar the rows are that belong to the cluster.

Definition

```
CREATE METHOD DM_getClusters ()
  RETURNS TABLE (ID INTEGER,
                 NAME CHARACTER VARYING(DM_MaxClusterNameLength),
                 SIZE BIGINT, HOMOGENEITY REAL)
  FOR DM_ClusteringModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxClusterNameLength* is the implementation-defined maximum number of characters of a cluster name.

Description

1) The result of the invocation *DM_getClusters()* is a table containing a row for each cluster contained in *SELF*.

7.3.6 DM_getMdlQuality Method

Purpose

Returns a value between 0 and 1 indicating the quality of the clustering model. This value is a measure of the homogeneity of the clusters contained in the DM_content of a value of type DM_ClusteringModel. 1 denotes maximal homogeneity, i.e. no discernable distinction, and 0 denotes no discernable commonalities between records that belong to the same cluster.

Definition

```
CREATE METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_ClusteringModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getMdlQuality()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain quality information, then the null value.

b) Otherwise, a *DOUBLE PRECISION* number between 0 (zero) and 1 (one), the overall quality of the clustering model.

7.3.7 DM_applyClusModel Method

Purpose

Return the DM_ClusResult value representing the result of applying the clustering model contained in SELF to the given DM_ApplicationData value.

Definition

```
CREATE METHOD DM_applyClusModel (inputRow DM_ApplicationData)
  RETURNS DM_ClusResult
  FOR DM_ClusteringModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_applyClusModel* takes the following input parameter:

   a) a *DM_ApplicationData* value *inputRow*.

2) The result of the invocation *DM_applyClusModel(inputRow)* is the *DM_ClusResult* value representing the result of an application of *SELF* to *inputRow*.

3) Note that field values missing in *inputRow* are substituted by null values. Field values in *inputRow* for fields not contained in *SELF.DM_getClusDataSpec()* are discarded.

7.3.8 DM_getClusTask method

Purpose

Return the DM_ClusBldTask value used to create this model.

Definition

```
CREATE METHOD DM_getClusTask ()
  RETURNS DM_ClusBldTask
  FOR DM_ClusteringModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusTask ()* is determined as follows:

   a) If *SELF* does not contain the *DM_ClusBldTask* value that was used to create the model, then the null value.

   b) Otherwise, it is the *DM_ClusBldTask* value that was used to compute the model.

7.3.9 DM_getClusMdlSpec method

Purpose

Return the DM_LogicalDataSpec value representing the set of fields needed for an application of the DM_ClusteringModel value.

Definition

```
CREATE METHOD DM_getClusMdlSpec ()
  RETURNS DM_LogicalDataSpec
  FOR DM_ClusteringModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusMdlSpec()* is the *DM_LogicalDataSpec* value representing the set of fields needed for an application of this model.

7.3.10 DM_getFields

Purpose

Returns a table of fields contained in SELF. For each field, its alias name, its mining type and its importance for SELF are given.

Definition

```
METHOD DM_getFields ()
   RETURNS TABLE (COLNAME VARCAHR(DM_MaxFieldNameLength),
                  FIELDNAME VARCAHR(DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   FOR DM_ClusteringModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Definitional Rules

1) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getFields()* is a table containing a row for each input field in the model used for model building.

7.4 DM_ClusApplTask Type and Routines

7.4.1 DM_ClusApplTask Type

Purpose

The type DM_ClusApplTask is a representation of all the information that constitutes a clustering application task, in particular it contains the mining model and the data mining data.

Definition

```
CREATE TYPE DM_ClusApplTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defClusApplTask
          (model DM_ClusteringModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_ClusApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClusMdl()
  RETURNS DM_ClusteringModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClusApplData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClusOutpData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClusResFld()
  RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_applyClusMdl()
  RETURNS DM_ClusApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a DM_ClusApplTask.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for the alias name of a field.

Description

1) The type *DM_ClusApplTask* provides for public use:

   a) a method *DM_defClusApplTask(DM_ClusteringModel, DM_MiningData, DM_MiningData, CHARACTER VARYING),*

   b) a method *DM_getClusMdl(),*

   c) a method *DM_getClusApplData(),*

   d) a method *DM_getClusOutpData(),*

   e) a method *DM_getClusResFld(),*

   f) a method *DM_applyClusMdl().*

7.4.2 DM_defClusApplTask Method

Purpose

Return the DM_ClusApplTask value containing the given DM_ClusteringModel value, the given DM_MiningData values for input and output, and the given result field value.

Definition

```
CREATE STATIC METHOD DM_defClusApplTask
          (model DM_ClusteringModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING (DM_MaxFieldAliasLength))
  RETURNS DM_ClusApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_defClusApplTask* takes the following input parameters:

   a) a *DM_ClusteringModel* value *model*.

   b) a *DM_MiningData* value *inputData*,

   c) a *DM_MiningData* value *outputData*,

   d) a *CHARACTER VARYING* value *resultField*,

2) The result of the invocation *DM_defClusApplTask(model, inputData, outputData, resultField)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *inputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *outputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null output data*.

   d) If *resultField* is not the alias of a field of type *DM_ClusResult* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field*.

   e) If *model.DM_getClusMdlSpec().DM_isCompatible(inputData.genDataSpec())* returns FALSE, then an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible*.

   f) Otherwise, the *DM_ClusApplTask* value containing the given arguments.

7.4.3 DM_getClusMdl Method

Purpose

Return the DM_ClusteringModel value representing the clustering model contained in the clustering application task.

Definition

```
CREATE METHOD DM_getClusMdl()
  RETURNS DM_ClusteringModel
  FOR DM_ClusApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_ClusteringModel* value, then the null value.

   b) Otherwise, it is the *DM_ClusteringModel* value contained in *SELF*.

7.4.4 DM_getClusApplData Method

Purpose

Return the DM_MiningData value for input contained in SELF.

Definition

```
CREATE METHOD DM_getClusApplData()
  RETURNS DM_MiningData
  FOR DM_ClusApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusApplData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for input contained in *SELF*.

7.4.5 DM_getClusOutpData Method

Purpose

Return the DM_MiningData value for output contained in SELF.

Definition

```
CREATE METHOD DM_getClusOutpData()
  RETURNS DM_MiningData
  FOR DM_ClusApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusOutpData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for output, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for output contained in *SELF*.

7.4.6 DM_getClusResFld method

Purpose

Return the CHARACTER VARYING value for the result field contained in SELF.

Definition

```
CREATE METHOD DM_getClusResFld()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   FOR DM_ClusApplTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getClusResFld()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *CHARACTER VARYING* value for the result field, then the null value.

   b) Otherwise, it is the *CHARACTER VARYING* value for the result field contained in *SELF*.

7.4.7 DM_applyClusMdl method

Purpose

Apply the clustering model in the DM_ClusApplTask value to each row of the table determined by the input data in the DM_ClusApplTask value. The results of the application are written to the table determined by the output data. In addition, any number of columns from the input table can be copied to the output table.

Definition

```
CREATE METHOD DM_applyClusMdl()
  RETURNS DM_ClusApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_applyClusMdl()* is determined as follows:

   Case:

   a) If the application run is successful, then *SELF*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model application failed*.

2) In a successful application run, for each row of the table determined by the input data mining data contained in *SELF* one row is inserted into the table determined by the output data mining data contained in *SELF*. The column values are determined as follows:

   a) The value for the column, corresponding to the field of *SELF.DM_getClusOutpData()* whose alias name is *SELF.DM_getClusResFld()*, is the result of the application of the clustering model to the input row. This is a value of type *DM_ClusResult.*

   b) For each field of *SELF.DM_getClusOutpData()*, whose alias name is the same as an alias name of a field of *SELF.DM_getClusApplData()*, the value of the corresponding output column is the same as the value of the corresponding column of the input row.

   c) Otherwise, the null value.

7.5 DM_ClusResult Type and Routines

7.5.1 DM_ClusResult Type

Purpose

The DM_ClusResult type is the description of the result of an application run of a clustering model.

Definition

```
CREATE TYPE DM_ClusResult
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_getClusterID()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getQuality()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClusResult*.

Description

1) The *DM_ClusResult* type provides for public use:

   a) a method *DM_getClusterID(),*

   b) a method *DM_getQuality().*

7.5.2 DM_getClusterID Method

Purpose

Returns the cluster identification number contained in the DM_ClusResult value.

Definition

```
CREATE METHOD DM_getClusterID()
  RETURNS INTEGER
  FOR DM_ClusResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClusterID()* is an *INTEGER* value representing the cluster identification number contained in *SELF*. This number identifies the cluster that best fits the input row used to generate *SELF*, i.e. the optimal cluster with respect to the clustering criterion of the specific implementation.

7.5.3 DM_getQuality Method

Purpose

Returns the quality value contained in the DM_ClusResult value.

Definition

```
CREATE METHOD DM_getQuality()
   RETURNS DOUBLE PRECISION
   FOR DM_ClusResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getQuality()* is the *DOUBLE PRECISION* value representing the quality value computed for the prediction of the cluster identification contained in *SELF*. This quality indicates how well the input row used to generate *SELF* fits into the cluster identified by the return value of *DM_getClusterID().*

Blank page

8 Classification

8.1 DM_ClasSettings Type and Routines

8.1.1 DM_ClasSettings Type

Purpose

The DM_ClasSettings type is the description for the settings that are used to generate a classification model. It defines a target field and parameters guiding the algorithm.

Definition

```
CREATE TYPE DM_ClasSettings
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impClasSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClasSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expClasSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setClasCostRate(costRate DOUBLE PRECISION)
  RETURNS DM_ClasSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getClasCostRate()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setClasCost
          (actualClass CHARACTER VARYING(DM_MaxClassLabelValueLength),
           predictedClass CHARACTER VARYING(DM_MaxClassLabelValueLength),
           cost DOUBLE PRECISION)
  RETURNS DM_ClasSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```

```
METHOD DM_getClasCost
          (actualClass CHARACTER VARYING(DM_MaxClassLabelValueLength),
           predictedClass CHARACTER VARYING(DM_MaxClassLabelValueLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_useClasDataSpec(logicalDataSpecification DM_LogicalDataSpec)
   RETURNS DM_ClasSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClasDataSpec()
   RETURNS DM_LogicalDataSpec
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_setClasTarget
          (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS DM_ClasSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClasTarget()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasSettings*.

2) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for the name of a class.

3) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_ClasSettings* type provides for public use:

   a) a method *DM_impClasSettings(CHARACTER LARGE OBJECT(DM_MaxContentLength)),*

   b) a method *DM_expClasSettings()*,

   c) a method *DM_setClasCostRate(DOUBLE PRECISION*),

   d) a method *DM_getClasCostRate(),*

e) a method *DM_setClasCost(CHARACTER VARYING, CHARACTER VARYING, DOUBLE PRECISION),*

f) a method *DM_getClasCost(CHARACTER VARYING, CHARACTER VARYING),*

g) a method *DM_useClasDataSpec(DM_LogicalDataSpec)*,

h) a method *DM_getClasDataSpec()*,

i) a method *DM_setClasTarget(CHARACTER VARYING)*,

j) a method *DM_getClasTarget()*.

8.1.2 DM_impClasSettings Method

Purpose

Return a specified value of type DM_ClasSettings.

Definition

```
CREATE STATIC METHOD DM_impClasSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClasSettings
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasSettings*.

Description

1) The method *DM_impClasSettings* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputSettings.* The format of this value is implementation-dependent, but for any one implementation the format is the same as that used by *DM_expClasSettings.*

2) The result of the invocation *DM_impClasSettings(inputSettings)* is determined as follows:

   Case:

   a) If there is a value v of type *DM_ClasSettings* such that *inputSettings* = v.*DM_expClasSettings(),* then v.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid settings format*.

8.1.3 DM_expClasSettings Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the classification settings contained in the DM_content value of the DM_ClasSettings value.

Definition

```
CREATE METHOD DM_expClasSettings()
   RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
   FOR DM_ClasSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasSettings*.

Description

1) The result of the invocation *DM_expClasSettings()* is a *CHARACTER LARGE OBJECT* value representing the classification settings contained in *SELF*. This representation is implementation-dependent.

8.1.4 DM_setClasCostRate Method

Purpose

Return a value of type DM_ClasSettings by specifying an error percentage for a training run. The classification cost rate value specifies the percentage of incorrect predictions that can be tolerated in a classification model on validation data. The training run may then terminate when the tolerated costRate is not exceeded on validation data.

Note that any values set by DM_setClasCost are not taken into account.

Definition

```
CREATE METHOD DM_setClasCostRate (costRate DOUBLE PRECISION)
  RETURNS DM_ClasSettings
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setClasCostRate* takes the following input parameter:

   a) a *DOUBLE PRECISION* value *costRate*.

2) The result of an invocation *DM_setClasCostRate(costRate)* is determined as follows:

   Case:

   a) If *costRate* is negative or greater than 100, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   b) Otherwise, it is the *DM_ClasSettings* value containing *costRate* as tolerated percentage of wrong predictions.

8.1.5 DM_getClasCostRate Method

Purpose

Return the error percentage (classification cost rate) contained in SELF.

Definition

```
CREATE METHOD DM_getClasCostRate()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasCostRate()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setClasCostRate(*v*)* for some *DOUBLE PRECISION* value v, then v.

   b) Otherwise, the null value, indicating that a numerical value for the classification cost rate will be determined during the data mining run.

8.1.6 DM_setClasCost Method

Purpose

Return a specified value of type DM_ClasSettings.

Definition

```
CREATE METHOD DM_setClasCost
          (actualClass CHARACTER VARYING(DM_MaxClassLabelValueLength)
           predictedClass CHARACTER VARYING(DM_MaxClassLabelValueLength)
  cost DOUBLE PRECISION)
  RETURNS DM_ClasSettings
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for the name of a class.

Description

1) The method *DM_setClasCost* takes the following input parameters:

   a) a *CHARACTER VARYING* value *actualClass,*

   b) a *CHARACTER VARYING* value *predictedClass,*

   c) a *DOUBLE PRECISION* value *cost.*

2) The result of the invocation *DM_setClasCost(actualClass, predictedClass, cost)* is determined as follows:

   Case:

   a) If *cost* is the null value, a value of *DM_ClasSettings*, with default cost assigned to the wrong prediction of an actual class *actualClass* as the predicted class *predictedClass*. The default cost is 1.0 if *actualClass* and *predictedClass* are different, 0.0 if they are equal.

   b) If *actualClass* equals *predictedClass* and *cost* is not equal to 0.0, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   c) If *actualClass* and *predictedClass* are different and *cost* is negative, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   d) Otherwise, a value v of type *DM_ClasSettings* that is identical to *SELF* except that the method invocation v.*DM_getClasCost(actualClass, predictedClass) = cost.*

8.1.7 DM_getClasCost Method

Purpose

Return the cost of a wrong prediction of an actual value of the class label (actualClass) as the predicted value of the class label (predictedClass).

Definition

```
CREATE METHOD DM_getClasCost
          (actualClass CHARACTER VARYING(DM_MaxClassLabelValueLength)
  predictedClass CHARACTER VARYING(DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for the name of a class.

Description

1) The method *DM_getClasCost* takes the following input parameters:

   a) a *CHARACTER VARYING* value *actualClass,*

   b) a *CHARACTER VARYING* value *predictedClass.*

2) The result of the invocation *DM_getClasCost(actualClass, predictedClass)* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setClasCost(actualClass, predictedClass,* v*)* for some *DOUBLE PRECISION* value *v* such that v is not null, then v.

   b) Otherwise, the default cost. The default cost is 1.0 if *actualClass* and *predictedClass* are different, 0.0 if they are equal.

8.1.8 DM_useClasDataSpec Method

Purpose

Specify a DM_LogicalDataSpec value for the DM_ClasSettings value. The DM_LogicalDataSpec determines valid DM_MiningData values, which can be processed in a training run.

Definition

```
CREATE METHOD DM_useClasDataSpec
          (logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_ClasSettings
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_useClasDataSpec* takes the following input parameter:

   a) a *DM_LogicalDataSpec* value *logicalDataSpecification*.

2) The result of the invocation *DM_useClasDataSpec(logicalDataSpecification)* is determined as follows:

   Case:

   a) If *logicalDataSpecification* is the null value, then a *DM_ClasSettings* value containing no logical data specification but otherwise identical to *SELF*.

   b) If *SELF.DM_getClasTarget()* = targetField (not null) and targetField is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   c) If *SELF.DM_getClasTarget()* = targetField (not null), targetField is contained in *logicalDataSpecification* and *logicalDataSpecification.DM_getFldType(targetField)* = 1 (one), then an error condition is raised: *SQL/MM Data Mining exception - field not categorical*.

   d) Otherwise, it is a value of type *DM_ClasSetting*s containing *logicalDataSpecification* as the logical data specification of *SELF*.

8.1.9 DM_getClasDataSpec Method

Purpose

Returns the logical data specification defined for a classification settings.

Definition

```
CREATE METHOD DM_getClasDataSpec()
  RETURNS DM_LogialDataSpec
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasDataSpec()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_useClasDataSpec(*v*)* for some *DM_LogicalDataSpec* value v such that v is not null, then v.

   b) Otherwise, it is the null value.

8.1.10 DM_setClasTarget Method

Purpose

Specify the target (class label) field for a DM_ClasSettings value.

Definition

```
CREATE METHOD DM_setClasTarget
          (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_ClasSettings
  FOR DM_ClasSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setClasTarget* takes the following input parameter:

   a) a *CHARACTER VARYING* value *targetField*.

2) The result of the invocation *DM_setClasTarget(targetField)* is determined as follows:

   Case:

   a) If *targetField* is the null value, then a value of type *DM_ClasSettings* not containing a target field but otherwise identical to *SELF*.

   b) If *SELF.DM_getClasDataSpec()* = ds (not null) and no field with name *targetField* is contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) If *SELF.DM_getClasDataSpec()* = ds (not null), *targetField* is contained in ds and ds.*DM_getFldType(targetField)* = 1 (one), then an error condition is raised: *SQL/MM Data Mining exception - field not categorical*.

   d) Otherwise, it is the *DM_ClasSettings* value containing the field named *targetField* as the target (class label) field.

8.1.11 DM_getClasTarget Method

Purpose

Returns the target field name defined for a classification settings.

Definition

```
CREATE METHOD DM_getClasTarget()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   FOR DM_ClasSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getClasTarget()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setClasTarget(*v*)* for some *CHARACTER VARYING* value v, then v.

   b) Otherwise, the null value.

8.2 DM_ClasBldTask Type and Routines

8.2.1 DM_ClasBldTask Type

Purpose

The type DM_ClasBldTask is an abstraction of all the information that constitutes a classification task, in particular the input data and the parameter settings. Furthermore, it provides a method for computing a classification model.

Definition

```
CREATE TYPE DM_ClasBldTask
  AS (
        DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defClasBldTask
          (trainData DM_MiningData, validationData DM_MiningData,
           settings DM_ClasSettings)
  RETURNS DM_ClasBldTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClasTrnData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasValData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasSettings()
  RETURNS DM_ClasSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_buildClasModel()
  RETURNS DM_ClasModel
  LANGUAGE SQL
  NOT DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasBldTask*

Description

1) The type *DM_ClasBldTask* provides for public use:

   a) a method *DM_defClasBldTask(DM_MiningData, DM_MiningData, DM_ClasSettings)*,

b) a method *DM_getClasTrnData()*,

c) a method *DM_getClasValData()*,

d) a method *DM_clasSettings()*,

e) a method *DM_buildClasModel()*.

8.2.2 DM_defClasBldTask Method

Purpose

Return a specified value of type DM_ClasBldTask.

Definition

```
CREATE STATIC METHOD DM_defClasBldTask
          (trainData DM_MiningData, validationData DM_MiningData,
           settings DM_ClasSettings)
  RETURNS DM_ClasBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defClasBldTask* takes the following input parameters:

   a) a *DM_MiningData* value *trainData*,

   b) a *DM_MiningData* value *validationData*,

   c) a *DM_ClasSettings* value *settings*.

2) The result of the invocation *DM_defClasBldTask(trainData, validationData, settings)* is determined as follows:

   Case:

   a) If *settings* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null settings*.

   b) If *trainData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *validationData* is the null value, then a value of type *DM_ClasBldTask* determined by the given parameters where no validation data is defined. An implementation-dependent subset of the training data is used for internal validation in this case.

   d) If *settings.DM_getClasDataSpec().isCompatible(trainData.DM_genDataSpec())* equals 1 (one) and *settings.DM_getClasDataSpec().isCompatible(validationData.DM_genDataSpec())* equals 1 (one), then a *DM_ClasBldTask* value bt such that bt.*DM_getClasTrnData()* equals *trainData,* bt.*DM_getClasValData()* equals *validationData* and bt.*DM_getClasSettings()* equals *settings*.

   e) Otherwise, an exception condition is raised: SQL/MM Data Mining exception – data and data specification not compatible.

8.2.3 DM_getClasTrnData Method

Purpose

Return the value of type DM_MiningData representing the training data for the classification task.

Definition

```
CREATE METHOD DM_getClasTrnData()
   RETURNS DM_MiningData
   FOR DM_ClasBldTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getClasTrnData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for training, then the null value.

   b) Otherwise, it is the *DM_MiningData* value representing the data to be used for training.

8.2.4 DM_getClasValData Method

Purpose

Return the value of type DM_MiningData representing the validation data for the classification task.

Definition

```
CREATE METHOD DM_getClasValData()
  RETURNS DM_MiningData
  FOR DM_ClasBldTask
  BEGIN
  --
  --!! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasValData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for validation, then the null value.

   b) Otherwise, it is the *DM_MiningData* value representing the data to be used for validation.

8.2.5 DM_getClasSettings Method

Purpose

Return the value of type DM_ClasSettings representing the settings of the classification task.

Definition

```
CREATE METHOD DM_getClasSettings()
  RETURNS DM_ClasSettings
  FOR DM_ClasBldTask
  BEGIN
  --
  --!! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasSettings()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_ClasSettings* value, then the null value.

   b) Otherwise, it is the *DM_ClasSettings* value representing the settings to be used for the training phase.

8.2.6 DM_buildClasModel Method

Purpose

Return a specified value of type DM_ClasModel.

Definition

```
CREATE STATIC METHOD DM_buildClasModel()
  RETURNS DM_ClasModel
  FOR DM_ClasBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_buildClasModel()* is determined as follows:

   Case:

   a) If the training run is successful, then an implementation-dependent value of type *DM_ClasModel*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model computation failed*.

8.3 DM_ClasModel Type and Routines

8.3.1 DM_ClasModel Type

Purpose

The DM_ClasModel type represents a classification model.

Definition

```
CREATE TYPE DM_ClasModel
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impClasModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClasModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expClasModel()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_applyClasModel(inputRow DM_ApplicationData)
  RETURNS DM_ClasResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_testClasModel(inputData DM_MiningData)
  RETURNS DM_ClasTestResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getClasCostRate()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_isValidated ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```

```
METHOD DM_getRankQuality
          (class CHARACTER VARYING (DM_MaxClassLabelValueLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getPredAccuracy
          (class CHARACTER VARYING (DM_MaxClassLabelValueLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getReliability ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getMdlQuality ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getGainsChart ()
   RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getClasTask()
   RETURNS DM_ClasBldTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getClasMdlSpec()
   RETURNS DM_LogicalDataSpec
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getFields ()
   RETURNS TABLE (COLNAME CHARACTER VARYING (DM_MaxFieldNameLength),
                  FIELDNAME CHARACTER VARYING (DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getClasTarget()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
```


Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasModel*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_ClasModel* provides for public use:

   a) a method *DM_impClasModel(CHARACTER LARGE OBJECT(DM_MaxContentLength)),*

   b) a method *DM_expClasModel(),*

   c) a method *DM_applyClasModel(DM_ApplicationData),*

   d) a method *DM_testClasModel(DM_MiningData),*

   e) a method *DM_getClasCostRate(),*

   f) a method *DM_isValidated(),*

   g) a method *DM_getRankQuality(CHARACTER VARYING),*

   h) a method *DM_getAccuracy(CHARACTER VARYING),*

   i) a method *DM_getReliability(),*

   j) a method *DM_getMdlQuality(),*

   k) a method *DM_getGainsChart(),*

   l) a method *DM_getClasTask(),*

   m) a method *DM_getClasMdlSpec(),*

   n) a method *DM_getFields(),*

   o) a method *DM_getClasTarget().*

8.3.2 DM_impClasModel Method

Purpose

Return a specified value of type DM_ClasModel.

Definition

```
CREATE STATIC METHOD DM_impClasModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_ClasModel
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasModel*.

Description

1) The method *DM_impClasModel* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputModel*.

2) The result of the invocation *DM_impClasModel*(*inputModel*) is determined as follows:

   Case:

   a) If *inputModel* contains an XML document defined by the PMML schema with exactly one NaiveBayesModel element, TreeModel element or NeuralNetwork element with categorical target, then a value of type *DM_ClasModel* representing the given PMML model.

   b) If *inputModel* contains an XML document defined by the PMML schema with more than one NaiveBayesModel element, TreeModel element or NeuralNetwork element with categorical target, then a value of type *DM_ClasModel* representing the first of the models.

   c) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid import format*.

8.3.3 DM_expClasModel Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the classification model contained in the DM_content value of the DM_ClasModel value.

Definition

```
CREATE METHOD DM_expClasModel()
   RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
   FOR DM_ClasModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasModel*.

Description

1) The result of the invocation *DM_expClasModel()* is a *CHARACTER LARGE OBJECT* value that contains an XML document conforming to the PMML schema with exactly one NaiveBayesModel element, TreeModel element or NeuralNetwork element with categorical target, representing the classification model contained in *SELF*.

8.3.4 DM_applyClasModel Method

Purpose

Return the result of applying the classification model contained in SELF to a given value of DM_ApplicationData.

Definition

```
CREATE METHOD DM_applyClasModel(inputRow DM_ApplicationData)
  RETURNS DM_ClasResult
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_applyClasModel* takes the following input parameter:

   a) a *DM_ApplicationData* value *inputRow*.

2) The result of the invocation *DM_applyClasModel(inputRow)* is the *DM_ClasResult* value representing the result of an application of *SELF* to *inputRow*.

3) Note that field values missing in *inputRow* are substituted by null values. Field values in *inputRow* for fields not contained in *SELF.DM_getClasDataSpec()* are discarded.

8.3.5 DM_testClasModel Method

Purpose

Return the result of testing the classification model contained in SELF, using a value of DM_MiningData.

Definition

```
CREATE METHOD DM_testClasModel(inputData DM_MiningData)
  RETURNS DM_ClasTestResult
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_testClasModel* takes the following input parameter:

   a) a *DM_MiningData* value *inputData*.

2) The result of the invocation *DM_testClasModel(inputData)* is determined as follows: :

   Case:

   a) If *SELF.DM_getClasDataSpec().DM_isCompatible(inputData.genLogDataSpec())* returns TRUE, then the result is the *DM_ClasTestResult* value representing the result of testing of *SELF* with *inputData* as test data.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible.*

8.3.6 DM_getClasCostRate Method

Purpose

Return the classification cost rate of the classification model computed during the training phase using the validation data specified in the DM_ClasBldTask value used to generate the model.

Definition

```
CREATE METHOD DM_getClasCostRate()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasCostRate()* is determined as follows:

   Case:

   a) If *SELF.DM_isValidated()* equals 0 (zero), then the classification cost rate on the training data.

   b) Otherwise, the classification cost rate of the classification model contained in *SELF* computed using the validation data.
   Note that the validation data may have been provided when the model was created or has been selected as an implementation-dependent subset of the training data.

8.3.7 DM_isValidated Method

Purpose

Indicates whether validation data have been used during the training phase so that quality measures are based on validation rather than training data.

Definition

```
CREATE METHOD DM_isValidated ()
   RETURNS INTEGER
   FOR DM_ClasModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Description

1) The result of the invocation *DM_isValidated()* is determined as follows:

Case:

a) 1 (one), if validation data have been used during training of the input model.

b) Otherwise, 0 (zero).

8.3.8 DM_getRankQuality method

Purpose

Returns a value between -1 and 1 indicating the ranking quality of the classification model with respect to a specified value *class* of the target field. This method returns a value which compares the order of the data records when sorted by their confidence values (for the prediction of *class*) with a random order of the same data records. 1 is assigned to an optimal model, by which the records are ordered exactly in the order of their actual values. Positive values indicate that the model is better than random. 0 is returned for models that work no better than random ordering. Negative values indicate that the ordering is worse than random. -1 means that the order is inverse to the actual values.

This value is based on validation data if possible, otherwise on training data.

Definition

```
CREATE METHOD DM_getRankQuality
       (class CHARACTER VARYING(DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClasModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The result of the invocation *DM_getRankQuality(class)* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain ranking quality information, then the null value.

b) If *class* is not predicted by the model, an exception condition is raised *SQL/MM Data Mining exception – parameter out of range*.

c) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between -1 and 1 (one), the ranking quality of the model, based on an implementation-dependent subset of the training data.

d) Otherwise, a double precision number between -1 and 1 (one), the ranking quality of the model, based on validation data.

8.3.9 DM_getPredAccuracy Method

Purpose

Returns a value between 0 and 1 indicating the model's accuracy to predict a specific value *class* of the target field. The value provides the probability that a prediction of *class* is correct, and it is computed as (# of correct predictions of *class*) / (total # of records with target field value *class*).

This value is based on validation data if possible, otherwise on training data.

Definition

```
CREATE METHOD DM_getPredAccuracy
       (class CHARACTER VARYING(DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClasModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The result of the invocation *DM_getPredAccuracy()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain predictive accuracy information, then the null value.

b) If *class* is not predicted by the model, an exception condition is raised *SQL/MM Data Mining exception – parameter out of range*.

c) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between 0 (zero) and 1 (one), the predictive accuracy of the model with respect to *class*, based on an implementation-dependent subset of the training data.

d) Otherwise, a double precision number between 0 (zero) and 1 (one), the predictive accuracy of the model with respect to *class*, based on validation data.

8.3.10 DM_getReliability Method

Purpose

Returns a value between 0 and 1, which is a measure of the model's capability to correctly predict unknown data, compared to its prediction capability on the training data. Hence, values considerably smaller than 1 indicate that the model may be overtrained. A value of 1 means that the error on the test data is not larger than that on the training data.

Definition

```
CREATE METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getReliability()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain reliability information, especially if *SELF.DM_isValidated()* = 0 (zero), then the null value.

b) 1.0, if the error rate for validation data is smaller than that for training data.

c) Otherwise, classification error (training data) / classification error (validation data).

8.3.11 DM_getMdlQuality Method

Purpose

Returns a value between 0 and 1 indicating the overall quality of the classification model. A value of 1 is returned for a perfect model. 0 indicates that the model is no better than guessing. This value is based on validation data if possible, otherwise on training data.

It is assumed that the overall quality value takes the individual quality values into account, i.e. ranking quality, accuracy and reliability.

Note that the quality value depends not only on the model itself but also on the validation or test data used. Even for a high quality value, it can only be assumed that model application produces high quality results, if the application data is similar to the validation data used.

Definition

```
CREATE METHOD DM_getMdlQuality ()
   RETURNS DOUBLE PRECISION
   FOR DM_ClasModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Description

1) The result of the invocation *DM_getMdlQuality()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain quality information, then the null value.

b) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between 0 (zero) and 1 (one), the overall quality of the model, based on an implementation-dependent subset of the training data.

c) Otherwise, a double precision number between 0 (zero) and 1 (one) is returned, the overall quality of the model, based on validation data.

8.3.12 DM_getGainsChart method

Purpose

Returns a table containing gains chart information for SELF.

Definition

```
CREATE METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  FOR DM_ClasModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getGainsChart()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain gains chart information, then the null value.

   b) If *SELF.DM_isValidated()* = 0 (zero), then a table containing gains chart information for *SELF*, based on an implementation-dependent subset of the training data.

   c) Otherwise, a table containing gains chart information for *SELF*, based on validation data.

8.3.13 DM_getClasTask method

Purpose

Return the DM_ClasBldTask value used to create this model.

Definition

```
CREATE METHOD DM_getClasTask()
   RETURNS DM_ClasBldTask
   FOR DM_ClasModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getClasTask()* is determined as follows:

   a) If *SELF* does not contain the *DM_ClasBldTask* value that was used to create the model, then the null value.

   b) Otherwise, it is the *DM_ClasBldTask* value that was used to compute the model.

8.3.14 DM_getClasMdlSpec method

Purpose

Return the DM_LogicalDataSpec value specifying the set of fields for an application of this model.

Definition

```
CREATE METHOD DM_getClasMdlSpec ()
  RETURNS DM_LogicalDataSpec
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasMdlSpec()* is the *DM_LogicalDataSpec* value representing the set of fields needed for an application of this model.

8.3.15 DM_getFields

Purpose

Return a table of fields contained in SELF. For each field, its alias name, its mining type and its importance for SELF are given.

Definition

```
METHOD DM_getFields ()
   RETURNS TABLE (COLNAME CHARACTER VARYING(DM_MaxFieldNameLength),
                  FIELDNAME CHARACTER VARYING (DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   FOR DM_ClasModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Definitional Rules

1) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getFields()* is a table containing a row for each input field in the model used for model building.

8.3.16 DM_getClasTarget method

Purpose

Return the name of the target field used when training the classification model.

Definition

```
CREATE METHOD DM_getClasTarget()
  RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getClasTarget()* is the name of the target field used when training the classification model.

8.4 DM_ClasTestTask Type and Routines

8.4.1 DM_ClasTestTask Type

Purpose

The type DM_ClasTestTask is a representation of all the information that constitutes a classification test task, in particular it contains the mining model and the data mining data or input.

Definition

```
CREATE TYPE DM_ClasTestTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defClasTestTask
          (model DM_ClasModel, inputData DM_MiningData)
  RETURNS DM_ClasTestTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClasTestMdl()
  RETURNS DM_ClasModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasTestData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_testClasMdl ()
  RETURNS DM_ClasTestResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a *DM_ClasTestTask.*

Description

1) The type DM_ClasTestTask provides for public use:

   a) a method *DM_defClasTestTask(DM_ClasModel, DM_MiningData),*

   b) a method *DM_getClasTestMdl(),*

   c) a method *DM_getClasTestData(),*

   d) a method *DM_testClasMdl().*

8.4.2 DM_defClasTestTask Method

Purpose

Return the DM_ClasTestTask value containing the given DM_ClasModel value and the given DM_MiningData value.

Definition

```
CREATE STATIC METHOD DM_defClasTestTask
          (model DM_ClasModel, inputData DM_MiningData)
  RETURNS DM_ClasTestTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defClasTestTask* takes the following input parameters:

   a) a *DM_ClasModel* value *model*.

   b) a *DM_MiningData* value *data*,

2) The result of the invocation *DM_defClasTestTask(model, data)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *data* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *model.DM_getClasMdlSpec().DM_isCompatible(data.genDataSpec())* returns FALSE, then an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible*.

   d) Otherwise, the *DM_ClasTestTask* value containing the given arguments.

8.4.3 DM_getClasTestData Method

Purpose

Return the DM_MiningData value for input contained in SELF.

Definition

```
CREATE METHOD DM_getClasTestData()
   RETURNS DM_MiningData
   FOR DM_ClasTestTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getClasTestData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value containing the input.

8.4.4 DM_getClasTestMdl Method

Purpose

Return the DM_ClasModel value contained in the classification test task.

Definition

```
CREATE METHOD DM_getClasTestMdl()
  RETURNS DM_ClasModel
  FOR DM_ClasTestTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasTestMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_ClasModel* value, then the null value.

   b) Otherwise, it is the *DM_ClasModel* value contained in *SELF*.

8.4.5 DM_testClasMdl Method

Purpose

Test the classification model in the DM_ClasTestTask value using the table determined by the input data in the DM_ClasTestTask value

Definition

```
CREATE METHOD DM_testClasMdl()
   RETURNS DM_ClasTestResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_testClasMdl()* is determined as follows:

   Case:

   a) If the test run is successful, then a value of type *DM_ClasTestResult* holding the outcome of the test.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model test failed*.

8.5 DM_ClasTestResult Type and Routines

8.5.1 DM_ClasTestResult Type

Purpose

The DM_ClasTestResult type is the description of the result of a test run of a classification model.

Definition

```
CREATE TYPE DM_ClasTestResult
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_getClasError()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRankQuality
         (class CHARACTER VARYING (DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getPredAccuracy
         (class CHARACTER VARYING (DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getNumClasses()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```



```
METHOD DM_getClass(position INTEGER)
   RETURNS CHARACTER VARYING (DM_MaxClassLabelValueLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getClasPreds(actualClass INTEGER, predictedClass INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasTestResult*.

2) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The *DM_ClasTestResult* type provides for public use:

   a) a method *DM_getClasError(),*

   b) a method *DM_getRankQuality(CHARACTER VARYING),*

   c) a method *DM_getAccuracy(CHARACTER VARYING),*

   d) a method *DM_getReliability(),*

   e) a method *DM_getMdlQuality(),*

   f) a method *DM_getGainsChart(),*

   g) a method *DM_getNumClasses(),*

   h) a method *DM_getClass(INTEGER),*

   i) a method *DM_getClasPreds(INTEGER, INTEGER).*

8.5.2 DM_getClasError Method

Purpose

Returns the classification error value contained in the DM_ClasTestResult value.

Definition

```
CREATE METHOD DM_getClasError()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasTestResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasError()* is the *DOUBLE PRECISION* value representing the percentage of false classifications computed during a test of a classification model.

8.5.3 DM_getRankQuality Method

Purpose

Returns a value between -1 and 1 indicating the ranking quality of the classification model on which the test result is based with respect to a specified value *class* of the target field. This method returns a value which compares the order of the data records when sorted by their confidence values (for the prediction of *class*) with a random order of the same data. 1 is assigned to an optimal model, by which the records are ordered exactly in the order of their actual values. Positive values indicate that the model is better than random. 0 is assigned to models that work no better than random ordering. Negative values indicate that the ordering is worse than random. -1 means that the order is inverse to the actual values.

This value is based on test data.

Definition

```
CREATE METHOD DM_getRankQuality
       (class CHARACTER VARYING(DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClasTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The result of the invocation *DM_getRankQuality(class)* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain ranking quality information, then the null value.

b) If *class* is not predicted by the model on which the test result is based, an exception condition is raised *SQL/MM Data Mining exception – parameter out of range*.

c) Otherwise, a double precision number between -1 and 1 (one) is returned, the ranking quality of the model on which the test result is based. This value is based on test data.

8.5.4 DM_getPredAccuracy Method

Purpose

Returns a value between 0 and 1 indicating the accuracy to predict a specific value *class* of the target field. The value provides the probability that a prediction of *class* is correct, and it is computed as (# of correct predictions of *class*) / (total # of records with target field value *class*). This value is based on test data.

Definition

```
CREATE METHOD DM_getPredAccuracy
       (class CHARACTER VARYING(DM_MaxClassLabelValueLength))
  RETURNS DOUBLE PRECISION
  FOR DM_ClasTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The result of the invocation *DM_getPredAccuracy(class)* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain predictive accuracy information, then the null value.

b) If *class* is not predicted by the model on which the test result is based, an exception condition is raised *SQL/MM Data Mining exception – parameter out of range*.

c) Otherwise, a double precision number between 0 (zero) and 1 (one), the predictive accuracy with respect to *class*.

8.5.5 DM_getReliability Method

Purpose

Returns a value between 0 and 1, which is a measure of the estimated capability to correctly predict unknown data, compared to the prediction capability on the training data. Hence, values considerably smaller than 1 indicate that the model may be overtrained. A value of 1 means that the error on the test data is not larger than that on the training data.

Definition

```
CREATE METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getReliability()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain reliability information, then the null value.

b) 1.0, if the error rate for the test data is smaller than that for the training data.

c) Otherwise, classification error (training data) / classification error (test data).

8.5.6 DM_getMdlQuality Method

Purpose

Returns a value between 0 and 1 indicating the overall quality of the classification model on which the test result is based. A value of 1 is returned for a perfect model. 0 indicates that the model is no better than guessing.

This value is based on test data. Note that the quality value depends not only on the model itself but also on test data used. Even for a high quality value, it can only be assumed that model application produces high quality results, if the application data is similar to the test data used.

It is assumed that the overall quality value takes the individual quality values into account, i.e. ranking quality, accuracy and reliability.

Definition

```
CREATE METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getMdlQuality()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain quality information, then the null value.

b) Otherwise, a double precision number between 0 (zero) and 1 (one), the overall quality of the model on which the test result is based.

8.5.7 DM_getGainsChart Method

Purpose

Returns a table containing gains chart information for SELF.

Definition

```
CREATE METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  FOR DM_ClasTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END

Description
```

1) The result of the invocation *DM_getGainsChart()* is determined as follows:

   Case:

   a) A table containing gains chart information for *SELF*, if present in the model.

   b) Otherwise, the null value.

8.5.8 DM_getNumClasses Method

Purpose

Returns the number of values encountered in the target field (class label) when training the classification model.

Definition

```
CREATE METHOD DM_getNumClasses()
  RETURNS INTEGER
  FOR DM_ClassificationResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getNumClasses()* is the number of values encountered in the target field (class label).

8.5.9 DM_getClass Method

Purpose

Returns the value of the class label in the DM_ClassificationResult type at the specified position.

Definition

```
CREATE METHOD DM_getClass(position INTEGER)
   RETURNS CHARACTER VARYING(DM_MaxClassLabelValueLength)
   FOR DM_ClassificationResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The method *DM_getClass* takes the following input parameter:

   a) an *INTEGER* value *position*.

2) The result of the invocation *DM_getClass(position)* is determined as follows:

   Case:

   a) If *position* is greater than 0 (zero) and less than or equal to *SELF.DM_getNumClasses(),* then the value of the class label with the associated number *position*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

8.5.10 DM_getClasPreds Method

Purpose

Returns the number of predictions of a class label value (given by position) made for input rows with an actual class label value, which is also given by position. The values SELF.DM_getClasPreds(n,m) provide the confusion matrix of SELF.

Definition

```
CREATE METHOD DM_getClasPreds
         (actualClass INTEGER, predictedClass INTEGER)
  RETURNS INTEGER
  FOR DM_ClassificationResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_getClasPreds* takes the following input parameters:

   a) an *INTEGER* value *actualClass,*

   b) an *INTEGER* value *predictedClass.*

2) The result of the invocation *DM_getClasPreds(actualClass, predictedClass)* is determined as follows:

   Case:

   a) If both *actualClass* and *predictedClass* are greater than 0 (zero) and less than or equal to *SELF.DM_getNumClasses(),* then the number of predictions of the class label value with position *predictedClass* made for input rows with an actual class label value with position *actualClass*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

8.6 DM_ClasApplTask Type and Routines

8.6.1 DM_ClasApplTask Type

Purpose

The type DM_ClasApplTask is a representation of all the information that constitutes a classification application task, in particular it contains the mining model and the data mining data for input and output.

Definition

```
CREATE TYPE DM_ClasApplTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defClasApplTask
          (model DM_ClasModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_ClasApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getClasMdl()
  RETURNS DM_ClasModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasApplData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasOutpData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getClasResFld()
  RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_applyClasMdl()
  RETURNS DM_ClasApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a DM_ClasApplTask.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The type *DM_ClasApplTask* provides for public use:

   a) a method *DM_defClasApplTask(DM_ClasModel, DM_MiningData, DM_MiningData, CHARACTER VARYING),*

   b) a method *DM_getClasMdl(),*

   c) a method *DM_getClasApplData(),*

   d) a method *DM_getClasOutpData(),*

   e) a method *DM_getClasResFld(),*

   f) a method *DM_applyClasMdl().*

8.6.2 DM_defClasApplTask Method

Purpose

Return the DM_ClasApplTask value containing the given DM_ClasModel value, the given DM_MiningData values for input and output, and the given result field value.

Definition

```
CREATE STATIC METHOD DM_defClasApplTask
          (model DM_ClasModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING (DM_MaxFieldAliasLength))
  RETURNS DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_defClasApplTask* takes the following input parameters:

   a) a *DM_ClasModel* value *model*.

   b) a *DM_MiningData* value *inputData*,

   c) a *DM_MiningData* value *outputData*,

   d) a *CHARACTER VARYING* value *resultField*,

2) The result of the invocation *DM_defClasApplTask(model, inputData, outputData, resultField)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *inputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *outputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null output data*.

   d) If *resultField* is not the alias of a field of type *DM_ClasResult* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field*.

   e) If *model.DM_getClasMdlSpec().DM_isCompatible(inputData.genDataSpec())* returns FALSE, then an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible*.

   f) Otherwise, the *DM_ClasApplTask* value containing the given arguments.

8.6.3 DM_getClasMdl Method

Purpose

Return the DM_ClasModel value representing the classification contained in the classification application task.

Definition

```
CREATE METHOD DM_getClusMdl()
  RETURNS DM_ClasModel
  FOR DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_ClasModel* value, then the null value.

   b) Otherwise, it is the *DM_ClasModel* contained in *SELF*.

8.6.4 DM_getClasApplData Method

Purpose

Return the DM_MiningData for input contained in SELF.

Definition

```
CREATE METHOD DM_getClasApplData()
  RETURNS DM_MiningData
  FOR DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasApplData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for input contained in *SELF*.

8.6.5 DM_getClasOutpData Method

Purpose

Return the DM_MiningData value for output contained in SELF.

Definition

```
CREATE METHOD DM_getClasOutpData()
  RETURNS DM_MiningData
  FOR DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getClasOutpData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for output, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for output contained in *SELF*.

8.6.6 DM_getClasResFld method

Purpose

Return the CHARACTER VARYING value for the result field contained in SELF.

Definition

```
CREATE METHOD DM_getClasResFld()
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  FOR DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getClasResFld()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *CHARACTER VARYING* value for the result field, then the null value.

   b) Otherwise, it is the *CHARACTER VARYING* value for the result field contained in SELF.

8.6.7 DM_applyClasMdl method

Purpose

Apply the classification model in the DM_ClasApplTask value to each row of the table determined by the input data in the DM_ClasApplTask value. The results of the application are written to the table determined by the output data. In addition any number of columns from the input table can be copied to the output table.

Definition

```
CREATE METHOD DM_applyClasMdl()
  RETURNS DM_ClasApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_applyClasMdl()* is determined as follows:

   Case:

   a) If the application run is successful, then *SELF*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model application failed*.

2) In a successful application run, for each row of the table determined by the input data mining data contained in *SELF* one row is inserted into the table determined by the output data mining data contained in *SELF*. The column values are determined as follows:

   a) The value for the column, corresponding to the field of *SELF.DM_getClasOutpData()* whose alias name is *SELF.DM_getClasResFld()*, is the result of the application of the classification model to the input row. This is a value of type *DM_ClasResult.*

   b) For each field of *SELF.DM_getClasOutpData()*, whose alias name is the same as an alias name of a field of *SELF.DM_getClasApplData()*, the value of the corresponding output column is the same as the value of the corresponding column of the input row.

   c) Otherwise, the null value.

8.7 DM_ClasResult Type and Routines

8.7.1 DM_ClasResult Type

Purpose

The DM_ClasResult type is the description of the result of an application run of a classification model.

Definition

```
CREATE TYPE DM_ClasResult
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_getPredClass()
  RETURNS CHARACTER VARYING(DM_MaxClassLabelValueLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getConfidence()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_ClasResult*.

2) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The *DM_ClasResult type* provides for public use:

   a) a method *DM_getPredClass(),*

   b) a method *DM_getConfidence().*

8.7.2 DM_getPredClass Method

Purpose

Returns the predicted class contained in the DM_ClasResult value.

Definition

```
CREATE METHOD DM_getPredClass()
  RETURNS CHARACTER VARYING(DM_MaxClassLabelValueLength)
  FOR DM_ClasResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label.

Description

1) The result of the invocation *DM_getPredClass()* is determined as follows:

   Case:

   a) If the result does not contain the predicted class, then the null value.

   b) Otherwise, a *CHARACTER VARYING* value representing the value of a class label contained in *SELF*. This class is the one predicted by the application of a classification model.

8.7.3 DM_getConfidence Method

Purpose

Returns the confidence value contained in the DM_ClasResult value.

Definition

```
CREATE METHOD DM_getConfidence()
   RETURNS DOUBLE PRECISION
   FOR DM_ClasResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getConfidence()* is is determined as follows:

   Case:

   a) If the result does not contain the confidence for the prediction of *SELF.DM_getPredClass()*, then the null value.

   b) Otherwise, a *DOUBLE PRECISION* value representing the confidence value computed for the prediction of the class label value contained in *SELF*.

Blank page

9 Regression

9.1 DM_RegSettings Type and Routines

9.1.1 DM_RegSettings Type

Purpose

The DM_RegSettings type is the description of the settings that are used to generate a regression model. It defines a target field and parameters guiding the algorithm.

Definition

```
CREATE TYPE DM_RegSettings
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impRegSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RegSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expRegSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_setRegRSquared(Rsquared DOUBLE PRECISION)
  RETURNS DM_RegSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRegRSquared()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_useRegDataSpec(logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_RegSettings
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegDataSpec()
  RETURNS DM_LogicalDataSpec
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```

```
METHOD DM_setRegTarget
          (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS DM_RegSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getRegTarget()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *content* of a *DM_RegSettings*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_RegSettings* type provides for public use:

   a) a method *DM_impRegSettings(CHARACTER LARGE OBJECT(DM_MaxContentLength)),*

   b) a method *DM_expRegSettings()*,

   c) a method *DM_setRegRSquared(DOUBLE PRECISION),*

   d) a method *DM_getRegRSquared(),*

   e) a method *DM_useRegDataSpec(DM_LogicalDataSpec),*

   f) a method *DM_getRegDataSpec(),*

   g) a method *DM_setRegTarget(CHARACTER VARYING),*

   h) a method *DM_getRegTarget().*

9.1.2 DM_impRegSettings Method

Purpose

Return a specified value of type DM_RegSettings.

Definition

```
CREATE STATIC METHOD DM_impRegSettings
          (inputSettings CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RegSettings
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegSettings*.

Description

1) The method *DM_impRegSettings* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputSettings*. The format of this value is implementation-dependent, but for any one implementation the format is the same as that used by *DM_expRegSettings*.

2) The result of the invocation *DM_impRegSettings(inputSettings*) is determined as follows:

   Case:

   a) If there is a value v of type *DM_RegSettings* such that *inputSettings* = v.*DM_expRegSettings()*, then v.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid settings format*.

9.1.3 DM_expRegSettings Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the regression settings contained in the DM_content value of the DM_RegSettings value.

Definition

```
CREATE METHOD DM_expRegSettings()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegSettings*.

Description

1) The result of the invocation *DM_expRegSettings()* is a *CHARACTER LARGE OBJECT* value representing the regression settings contained in *SELF*. This representation is implementation-dependent.

9.1.4 DM_setRegRSquared Method

Purpose

Return a DM_RegSettings value with the specified maximum tolerated squared Pearson correlation coefficient for the training phase of a data mining run of data mining regression. This coefficient can be tolerated in the data mining regression model for the error on verification data.

Definition

```
CREATE METHOD DM_setRegRSquared(Rsquared DOUBLE PRECISION)
  RETURNS DM_RegSettings
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_setRegRSquared* takes the following input parameter:

   a) a *DOUBLE PRECISION* value *rsquared*.

2) The result of the invocation *DM_setRegRSquared(Rsquared)* is determined as follows:

   Case:

   a) If *Rsquared* is negative or greater than 1, then an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

   b) Otherwise, a value of type *DM_RegSettings* containing *Rsquared* as the maximum allowed squared Pearson correlation coefficient.

9.1.5 DM_getRegRSquared Method

Purpose

Return the r-squared value contained in SELF.

Definition

```
CREATE METHOD DM_getRegRSquared()
  RETURNS DOUBLE PRECISION
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegRSquared()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setRegRSquared(*v*)* for some *DOUBLE PRECISION* value v, then v.

   b) Otherwise, the null value, indicating that a numerical value for the r-squared value will be determined during the data mining run.

9.1.6 DM_useRegDataSpec Method

Purpose

Specify a DM_LogicalDataSpec value for the DM_RegSettings value. The DM_LogicalDataSpec value determines valid DM_MiningData values, which can be processed in a training run.

Definition

```
CREATE METHOD DM_useRegDataSpec
          (logicalDataSpecification DM_LogicalDataSpec)
  RETURNS DM_RegSettings
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_useRegDataSpec* takes the following input parameter:

   a) a *DM_LogicalDataSpec* value *logicalDataSpecification*.

2) The result of the invocation *DM_useRegDataSpec(logicalDataSpecification)* is determined as follows:

   Case:

   a) If *logicalDataSpecification* is the null value, then a *DM_RegSettings* value containing no logical data specification but otherwise identical to *SELF*.

   b) If *SELF.DM_getRegTarget()* = targetField (not null) and targetField is not contained in *logicalDataSpecification,* then an error condition is raised: *SQL/MM Data Mining exception - field not defined in data specification*.

   c) If *SELF.DM_getRegTarget()* = targetField (not null), targetField is contained in *logicalDataSpecification* and *logicalDataSpecification.DM_getFldType*(targetField) = 0 (zero), then an error condition is raised: *SQL/MM Data Mining exception - field not numerical*.

   d) Otherwise, it is the *DM_RegSettings* value containing *logicalDataSpecification* as the logical data specification.

9.1.7 DM_getRegDataSpec Method

Purpose

Returns the logical data specification defined for a regression settings.

Definition

```
CREATE METHOD DM_getRegDataSpec()
  RETURNS DM_LogicalDataSpec
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegDataSpec()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_useRegDataSpec(*v*)* for some *DM_LogicalDataSpec* value v, then v.

   b) Otherwise, the null value.

9.1.8 DM_setRegTarget Method

Purpose

Specify a target (predicted) field for the DM_RegSettings value.

Definition

```
CREATE METHOD DM_setRegTarget
          (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RegSettings
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_setRegTarget* takes the following input parameter:

   a) a *CHARACTER VARYING* value *targetField*.

2) The result of the invocation *DM_setRegTarget(targetField)* is determined as follows:

   Case:

   a) If *targetField* is the null value, then a value of type *DM_RegSettings* not containing a target field but otherwise identical to *SELF*.

   b) If *SELF.DM_getRegDataSpec()* = ds (not null) and *targetField* is not equal to the name of any field contained in ds, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.

   c) If *SELF.DM_getRegDataSpec()* = ds (not null), *targetField* is contained in ds and ds.*DM_getFldType(targetField)* = 0 (zero), then an error condition is raised: *SQL/MM Data Mining exception - field not numerical*.

   d) Otherwise, it is the *DM_RegSettings* value containing the field named *targetField* as the target field.

9.1.9 DM_getRegTarget Method

Purpose

Returns the target field name defined for a regression settings.

Definition

```
CREATE METHOD DM_getRegTarget()
  RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getRegTarget()* is determined as follows:

   Case:

   a) If *SELF* is equal to *SELF.DM_setRegTarget(v)* for some *CHARACTER VARYING* value v, then v.

   b) Otherwise, the null value.

9.2 DM_RegBldTask type and Routines

9.2.1 DM_RegBldTask Type

Purpose

The type DM_RegBldTask represents the information to invoke the training of a regression model. In particular, it consists of a representation of the input data and the parameter settings.

Definition

```
CREATE TYPE DM_RegBldTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defRegBldTask
          (trainData DM_MiningData, validationData DM_MiningData,
           settings DM_RegSettings)
  RETURNS DM_RegBldTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegTrnData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRegValData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRegSettings()
  RETURNS DM_RegSettings
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_buildRegModel()
  RETURNS DM_RegressionModel
  LANGUAGE SQL
  NOT DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegBldTask*

Description

1) The type *DM_RegBldTask* provides for public use:

   a) a method *DM_defRegBldTask(DM_MiningData, DM_MiningData, DM_RegSettings)*,

   b) a method *DM_getRegTrnData()*,

c) a method *DM_getRegValData()*,

d) a method *DM_getRegSettings()*,

e) a method *DM_buildRegModel()*.

9.2.2 DM_defRegBldTask Method

Purpose

Return the DM_RegBldTask value determined by the given input parameters.

Definition

```
CREATE STATIC METHOD DM_defRegBldTask
          (trainData DM_MiningData, validationData DM_MiningData,
           settings DM_RegSettings)
  RETURNS DM_RegBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defRegBldTask* takes the following input parameters:

   a) a *DM_MiningData* value *trainData*,

   b) a *DM_MiningData* value *validationData*,

   c) a *DM_RegSettings* value *settings*.

2) The result of the invocation *DM_defRegBldTask(trainData, validationData, settings)* is determined as follows:

   Case:

   a) If *settings* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null settings*.

   b) If *trainData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *validationData* is the null value, then a value of type *DM_RegBldTask* determined by the given parameters where no validation data is defined. It is implementation-dependent which subset of the training data is used for internal validation in this case.

   d) If *settings.DM_getRegDataSpec().isCompatible(trainData.DM_genDataSpec())* equals 1 (one) and *settings.DM_getRegDataSpec().isCompatible(validationData.DM_genDataSpec())* equals 1 (one), then a *DM_RegBldTask* value bt such that bt.*DM_getRegTrnData()* equals *trainData,* bt.*DM_getRegValData()* equals *validationData* and bt.*DM_getRegSettings()* equals *settings*.

   e) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – data and data specification not compatible*..

9.2.3 DM_getRegTrnData Method

Purpose

Return the DM_MiningData value representing the training data of the DM_RegBldTask value.

Definition

```
CREATE METHOD DM_getRegTrnData()
  RETURNS DM_MiningData
  FOR DM_RegBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegTrnData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for training, then the null value.

   b) Otherwise, a value of type *DM_MiningData* representing the data to be used for training.

9.2.4 DM_getRegValData Method

Purpose

Return the DM_MiningData value representing the validation data of the DM_RegBldTask value.

Definition

```
CREATE METHOD DM_getRegValData()
   RETURNS DM_MiningData
   FOR DM_RegBldTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRegValData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for validation, then the null value.

   b) Otherwise, a value of type *DM_MiningData* representing the data to be used for validation.

9.2.5 DM_getRegSettings Method

Purpose

Return the DM_RegSettings value representing the settings of the DM_RegBldTask value.

Definition

```
CREATE METHOD DM_getRegSettings()
  RETURNS DM_RegSettings
  FOR DM_RegBldTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegSettings()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_RegSettings* value, then the null value.

   b) Otherwise, a value of type *DM_RegSettings* representing the settings to be used for the training phase.

9.2.6 DM_buildRegModel Method

Purpose

Return the result of a training run of a regression model as DM_RegressionModel value given the information specified in SELF.

Definition

```
CREATE METHOD DM_buildRegModel()
   RETURNS DM_RegressionModel
   FOR DM_RegBldTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_buildRegModel()* is determined as follows:

   Case:

   a) If the training run of this mining function is successful, then an implementation-dependent value of type *DM_RegressionModel* representing the result of the training run as well as the settings, training and validation data (if any) that were used.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model computation failed*.

9.3 DM_RegressionModel Type and Routines

9.3.1 DM_RegressionModel Type

Purpose

The DM_RegressionModel type represents a regression model.

Definition

```
CREATE TYPE DM_RegressionModel
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_impRegModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RegressionModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_expRegModel()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_applyRegModel
  (inputRow DM_ApplicationData)
  RETURNS DM_RegResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_testRegModel(inputData DM_MiningData)
  RETURNS DM_RegTestResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRegRSquared()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_isValidated ()
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,
```



```
METHOD DM_getRankQuality ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getPredAccuracy ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getReliability ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getMdlQuality ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getRMSE ()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getGainsChart ()
   RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_getRegTask()
   RETURNS DM_RegBldTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getRegMdlSpec()
   RETURNS DM_LogicalDataSpec
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_getFields ()
   RETURNS TABLE (COLNAME VARCAHR(DM_MaxFieldNameLength),
                  FIELDNAME VARCAHR(DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

```
METHOD DM_getRegTarget()
   RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegressionModel*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The *DM_RegressionModel* type provides for public use:

   a) a method *DM_impRegModel(CHARACTER LARGE OBJECT(DM_MaxContentLength))*,

   b) a method *DM_expRegModel(),*

   c) a method *DM_applyRegModel(DM_ApplicationData)*,

   d) a method *DM_testRegModel(DM_MiningData),*

   e) a method *DM_getRegRSquared(),*

   f) a method *DM_isValidated(),*

   g) a method *DM_getRankQuality(),*

   h) a method *DM_getAccuracy(),*

   i) a method *DM_getReliability(),*

   j) a method *DM_getMdlQuality(),*

   k) a method *DM_getRMSE(),*

   l) a method *DM_getGainsChart(),*

   m) a method *DM_getRegTask(),*

   n) a method *DM_getRegMdlSpec(),*

   o) a method *DM_getFields(),*

   p) a method *DM_getRegTarget()*.

9.3.2 DM_impRegModel Method

Purpose

Return the DM_RegressionModel value determined by the given input string.

Definition

```
CREATE STATIC METHOD DM_impRegModel
          (inputModel CHARACTER LARGE OBJECT(DM_MaxContentLength))
  RETURNS DM_RegressionModel
  FOR DM_RegressionModel
  BEGIN
  --
  --!! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegressionModel*.

Description

1) The method *DM_impRegModel* takes the following input parameter:

   a) a *CHARACTER LARGE OBJECT* value *inputModel*.

2) The result of the invocation *DM_impRegModel*(*inputModel*) is determined as follows:

   Case:

   a) If *inputModel* contains an XML document defined by the PMML schema with exactly one RegressionModel element, GeneralRegressionModel element, SupportVectorMachineModel element or NeuralNetwork element with numerical target, then a value of type *DM_RegressionModel* representing the given PMML model.

   b) If *inputModel* contains an XML document defined by the PMML schema with more than one RegressionModel element, GeneralRegressionModel element, SupportVectorMachineModel element or NeuralNetwork element with numerical target, then a value of type *DM_RegressionModel* representing the first of the models.

   c) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid import format*.

9.3.3 DM_expRegModel Method

Purpose

Return a CHARACTER LARGE OBJECT value representing the regression model contained in SELF.

Definition

```
CREATE METHOD DM_expRegModel()
  RETURNS CHARACTER LARGE OBJECT(DM_MaxContentLength)
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegressionModel*.

Description

1) The result of the invocation *DM_expRegModel()* is a *CHARACTER LARGE OBJECT* value that contains an XML document conforming to the PMML schema with exactly one RegressionModel element, GeneralRegressionModel element, SupportVectorMachinElement or NeuralNetwork element with numerical target, representing the regression model contained in *SELF*.

9.3.4 DM_applyRegModel Method

Purpose

Return the DM_RegResult value representing the result of applying the regression model contained in SELF to a given DM_ApplicationData value.

Definition

```
CREATE METHOD DM_applyRegModel(inputRow DM_ApplicationData)
  RETURNS DM_RegResult
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_applyRegModel* takes the following input parameter:

   a) a *DM_ApplicationData* value *inputRow*.

2) The result of the invocation *DM_applyRegModel(inputRow)* is the *DM_RegResult* value representing the result of an application of *SELF* to *inputRow*.

3) Note that field values missing in *inputRow* are substituted by null values. Field values in *inputRow* for fields not contained in *SELF.DM_getRegDataSpec()* are discarded.

9.3.5 DM_testRegModel Method

Purpose

Return the DM_RegTestResult value representing the result of testing the regression model contained in SELF, using a given DM_MiningData value.

Definition

```
CREATE METHOD DM_testRegModel(inputData DM_MiningData)
  RETURNS DM_RegTestResult
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_testRegModel* takes the following input parameter:

   a) a *DM_MiningData* value *inputData*.

2) The result of the invocation *DM_testRegModel(inputData)* is determined as follows: :

   Case:

   a) If *SELF.DM_getRegDataSpec().DM_isCompatible(inputData.genLogDataSpec())* returns TRUE, then the result is the *DM_RegTestResult* value representing the result of testing of *SELF* with *inputData* as test data.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible.*

9.3.6 DM_getRegRSquared method

Purpose

Return the squared Pearson correlation coefficient of the regression model computed during the training phase using the validation data specified in the DM_RegBldTask value used to generate the model.

Definition

```
CREATE METHOD DM_getRegRSquared()
   RETURNS DOUBLE PRECISION
   FOR DM_RegressionModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRegRSquared()* is determined as follows:

   Case:

   a) If the model does not contain the squared Pearson correlation, then the null value.

   b) If *SELF.DM_isValidated() = 0* (zero), then the squared Pearson correlation coefficient based on an implementation-dependent subset of the training data used for validation purposes.

   c) Otherwise, it is the squared Pearson correlation coefficient of the regression model based on the validation data.

9.3.7 DM_isValidated Method

Purpose

Indicates whether validation data have been used during the training phase so that quality measures are based on validation rather than training data.

Definition

```
CREATE METHOD DM_isValidated ()
  RETURNS INTEGER
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_isValidated()* is determined as follows:

Case:

a) 1 (one), if validation data have been used during training of the input model.

b) Otherwise, 0 (zero).

9.3.8 DM_getRankQuality Method

Purpose

Returns a value between -1 and 1 measuring the ranking quality of the regression model. This method returns a value which compares the order of the data records when sorted by their predicted values with a random order of the same data records. 1 indicates an optimal model, by which the records are ordered exactly in the order of their actual values. Positive values indicate that the model is better than random. 0 is returned for models that work no better than random ordering. Negative values indicate that the ordering is worse than random. -1 means that the order is inverse to the actual values.

This value is based on validation data if possible, otherwise on training data.

Definition

```
CREATE METHOD DM_getRankQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getRankQuality()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain ranking quality information, then the null value.

b) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between -1 and 1 (one), the ranking quality of the model, based on an implementation-dependent subset of the training data.

c) Otherwise, a double precision number between -1 and 1 (one), the ranking quality of the model, based on validation data.

9.3.9 DM_getPredAccuracy Method

Purpose

Returns a value between 0 and 1 indicating the accuracy of the model's predictions. 1 means that all the predicted values are equal to the actual values. A value close to 0 means that predictions are very far off. This value is based on validation data if possible, otherwise on training data.

Definition

```
CREATE METHOD DM_getPredAccuracy ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getPredAccuracy()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain predictive accuracy information, then the null value.

b) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between 0 (zero) and 1 (one), the predictive accuracy of the model, based on an implementation-dependent subset of the training data.

c) Otherwise, a double precision number between 0 (zero) and 1 (one), the predictive accuracy of the model, based on validation data.

9.3.10 DM_getReliability Method

Purpose

Returns a value between 0 and 1, which is a measure of the model's estimated capability to correctly predict unknown data, compared to its prediction capability on the training data. Hence, values considerably smaller than 1 indicate that the model may be overtrained. A value of 1 means that the error on the test data is not larger than that on the training data.

Definition

```
CREATE METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getReliability()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain reliability information, especially if *SELF.DM_isValidated()* = 0 (zero), then the null value.

b) 1.0, if the root mean squared error (rmse) for validation data is smaller than that for training data.

c) Otherwise, rmse (training data) / rmse (validation data).

9.3.11 DM_getMdlQuality Method

Purpose

Returns a value between 0 and 1 indicating the overall quality of the regression model. A value of 1 is returned for a perfect model. 0 indicates that the model is no better than guessing.This value is based on validation data if possible, otherwise on training data.

It is assumed that the overall quality value takes the individual quality values into account, i.e. ranking quality, accuracy and reliability.

Note that the quality value depends not only on the model itself but also on the validation or test data used. Even for a high quality value, it can only be assumed that model application produces high quality results, if the application data is similar to the validation data used.

Definition

```
CREATE METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getMdlQuality()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain quality information, then the null value.

b) If *SELF.DM_isValidated()* = 0 (zero), then a double precision number between 0 (zero) and 1 (one), the overall quality of the model, based on an implementation-dependent subset of the training data.

c) Otherwise, a double precision number between 0 (zero) and 1 (one) is returned, the overall quality of the model, based on validation data.

9.3.12 DM_getRMSE Method

Purpose

Returns a non-negative value giving the root mean squared error of the regression model. This value is based on validation data if possible, otherwise on training data.

Note that the value depends not only on the model itself but also on the validation or test data used. Even for a low value, it can only be assumed that model application produces high quality results, if the application data is similar to the validation data used.

Definition

```
CREATE METHOD DM_getRMSE ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getRMSE()* is determined as follows:

Case:

a) If *SELF.DM_content* does not contain information about the root mean squared error, then the null value.

b) If *SELF.DM_isValidated()* = 0 (zero), then a non-negative double precision number, the root mean squared error of the model, based on an implementation-dependent subset of the training data.

c) Otherwise, a non-negative double precision number is returned, the root mean squared error of the model, based on validation data.

9.3.13 DM_getGainsChart method

Purpose

Returns a table containing gains chart information for SELF.

Definition

```
CREATE METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  FOR DM_RegressionModel
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getGainsChart()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain gains chart information, then the null value.

   b) If *SELF.DM_isValidated()* = 0 (zero), then a table containing gains chart information for *SELF*, based on an implementation-dependent subset of the training data.

   c) Otherwise, a table containing gains chart information for *SELF*, based on validation data.

9.3.14 DM_getRegTask method

Purpose

Return the DM_RegBldTask value used to create this model.

Definition

```
CREATE METHOD DM_getRegTask()
   RETURNS DM_RegBldTask
   FOR DM_RegressionModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRegTask()* is determined as follows:

   a) If *SELF* does not contain the *DM_RegBldTask* value that was used to compute the model, then the null value.

   b) Otherwise, it is the *DM_RegBldTask* value that was used to compute the model.

9.3.15 DM_getRegMdlSpec method

Purpose

Return the DM_LogicalDataSpec value specifying the set of fields necessary for an application or test of this model.

Definition

```
CREATE METHOD DM_getRegMdlSpec ()
  RETURNS DM_LogicalDataSpec
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegMdlSpec()* is the *DM_LogicalDataSpec* value representing the set of fields needed for an application of this model.

9.3.16 DM_getFields method

Purpose

Return a table of fields contained in SELF. For each field, its alias name, its mining type and its importance for SELF are given.

Definition

```
METHOD DM_getFields ()
   RETURNS TABLE (COLNAME VARCAHR(DM_MaxFieldNameLength),
                  FIELDNAME VARCAHR(DM_MaxAliasNameLength),
                  MININGTYPE INTEGER, IMPORTANCE REAL)
   FOR DM_RegressionModel
   BEGIN
   ---
   --- !! See Description
   ---
   END
```

Definitional Rules

1) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getFields()* is a table containing a row for each input field in the model used for model building.

9.3.17 DM_getRegTarget method

Purpose

Return the name of the target field used when training the regression model.

Definition

```
CREATE METHOD DM_getRegTarget()
  RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getRegTarget()* is the name of the target field used when training the regression model.

9.4 DM_RegTestTask Type and Routines

9.4.1 DM_RegTestTask Type

Purpose

The type DM_RegTestTask is a representation of all the information that constitutes a regression test task, in particular it contains the mining model and the data mining data for input.

Definition

```
CREATE TYPE DM_RegTestTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defRegTestask
          (model DM_RegressionModel, data DM_MiningData)
  RETURNS DM_RegTestTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegTestMdl()
  RETURNS DM_RegressionModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_getRegTestData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL,

METHOD DM_testRegMdl()
  RETURNS DM_RegTestResult
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a *DM_RegTestTask*.

Description

1) The type *DM_RegTestTask* provides for public use:
   a) a method *DM_defRegTestTask(DM_RegressionModel, DM_MiningData),*
   b) a method *DM_getRegTestMdl(),*
   c) a method *DM_getRegTestData(),*
   d) a method *DM_testRegMdl().*

9.4.2 DM_defRegTestTask Method

Purpose

Return the DM_RegTestTask value containing the given DM_RegressionModel value and the given DM_MiningData value.

Definition

```
CREATE STATIC METHOD DM_defRegTestTask
          (model DM_RegressionModel, data DM_MiningData)
  RETURNS DM_RegTestTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The method *DM_defRegTestTask* takes the following input parameters:

   a) a *DM_RegressionModel* value *model*.

   b) a *DM_MiningData* value *data*,

2) The result of the invocation *DM_defRegTestTask(model, data)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *data* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *model.DM_getRegMdlSpec().DM_isCompatible(data.genDataSpec())* returns FALSE, then an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible*.

   d) Otherwise, the *DM_RegTestTask* value containing the given arguments.

9.4.3 DM_getRegTestData Method

Purpose

Return the DM_MiningData value for input contained in SELF.

Definition

```
CREATE METHOD DM_getRegTestData()
   RETURNS DM_MiningData
   FOR DM_RegTestTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_getRegTestData()* is determined as follows:.

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value containing the input.

9.4.4 DM_getRegTestMdl Method

Purpose

Return the DM_RegressionModel value contained in the regression test task.

Definition

```
CREATE METHOD DM_getRegTestMdl()
  RETURNS DM_RegressionModel
  FOR DM_RegTestTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegTestMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_RegressionModel* value, then the null value.

   b) Otherwise, it is the *DM_RegressionModel* value contained in *SELF*.

9.4.5 DM_testRegMdl Method

Purpose

Test the regression model in the DM_RegTestTask value using the table determined by the input data in the DM_RegTestTask value.

Definition

```
CREATE METHOD DM_testRegMdl()
   RETURNS DM_RegTestResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of the invocation *DM_testRegMdl()* is determined as follows:

   Case:

   a) If the test run is successful, then a value of type *DM_RegTestResult* holding the outcome of the test.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model test failed*.

9.5 DM_RegTestResult Type and Routines

9.5.1 DM_RegTestResult Type

Purpose

The DM_RegTestResult type is the description of the result of a test run of a regression model.

Definition

```
CREATE TYPE DM_RegTestResult
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_getPredError()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRankQuality ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getPredAccuracy ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getRMSE ()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
```


Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegTestResult*.

Description

1) The *DM_RegTestResult* type provides for public use:

   a) a method *DM_getPredError(),*

   b) a method *DM_getRankQuality(),*

   c) a method *DM_getAccuracy(),*

   d) a method *DM_getReliability(),*

   e) a method *DM_getMdlQuality(),*

   f) A method *DM_getRMSE(),*

   g) a method *DM_getGainsChart().*

9.5.2 DM_getPredError Method

Purpose

Returns the prediction error value contained in the DM_RegTestResult value.

Definition

```
CREATE METHOD DM_getPredError()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getPredError()* is the *DOUBLE PRECISION* value representing the aggregated error computed during the test of a prediction model.

9.5.3 DM_getRankQuality Method

Purpose

Returns a value between -1 and 1 indicating the ranking quality of the regression model on which the test result is based. This method returns a value which compares the order of the data records when sorted by their predicted values with a random order of the same data. 1 is assigned to an optimal model, by which the records are ordered exactly in the order of their actual values. Positive values indicate that the model is better than random. 0 is assigned to models that work no better than random ordering. Negative values indicate that the ordering is worse than random. -1 means that the order is inverse to the actual values.

This value is based on test data.

Definition

```
CREATE METHOD DM_getRankQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getRankQuality()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain ranking quality information, then the null value.

   b) Otherwise, a *DOUBLE PRECISION* number between -1 and 1 (one) is returned, the ranking quality of the model on which the test result is based. This value is based on test data.

9.5.4 DM_getPredAccuracy Method

Purpose

Returns a value between 0 and 1 indicating the predictive accuracy of the regression model on which the test result is based. 1 indicates that all the values are exactly predicted. A value close to 0 means that predictions are very far off.

This value is based test data.

Definition

```
CREATE METHOD DM_getPredAccuracy ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getPredAccuracy()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain predictive accuracy information, then the null value.

   b) Otherwise, a double precision number between 0 (zero) and 1 (one), the predictive accuracy.

9.5.5 DM_getReliability Method

Purpose

Returns a value, which is a measure of the estimated capability to correctly predict unknown data, compared to the prediction capability on the training data. Hence, values considerably smaller than 1 indicate that the model may be overtrained. A value of 1 means that the error on the test data is not larger than that on the training data.

Definition

```
CREATE METHOD DM_getReliability ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getReliability()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain reliability information, then the null value.

   b) 1.0, if the error rate for the test data is smaller than that for the training data.

   c) Otherwise, rmse (training data) / rmse (test data).

9.5.6 DM_getMdlQuality

Purpose

Returns a value between 0 and 1 indicating the overall quality of the regression model on which the test result is based. A value of 1 is returned for a perfect model. 0 indicates that the model is no better than guessing. This value is based on test data.

It is assumed that the overall quality value takes the individual quality values into account, i.e. ranking quality, accuracy and reliability.

Note that the quality value depends not only on the model itself but also on test data used. Even for a high quality value, it can only be assumed that model application produces high quality results, if the application data is similar to the test data used.

Definition

```
CREATE METHOD DM_getMdlQuality ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getMdlQuality()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain quality information, then the null value.

   b) Otherwise, a *DOUBLE PRECISION* number between 0 (zero) and 1 (one), the overall quality of the model.

9.5.7 DM_getRMSE Method

Purpose

Returns a non-negative value giving the root mean squared error of the regression test result.

Note that the value depends not only on the model itself but also on the test data used. Even for a low value, it can only be assumed that model application produces high quality results, if the application data is similar to the test data used.

Definition

```
CREATE METHOD DM_getRMSE ()
  RETURNS DOUBLE PRECISION
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getRMSE()* is determined as follows:

   Case:

   a) If *SELF.DM_content* does not contain information about the root mean squared error, then the null value.

   b) Otherwise, a non-negative *DOUBLE PRECISION* number is returned, the root mean squared error of the test result.

9.5.8 DM_getGainsChart Method

Purpose

Returns a table containing gains chart information for SELF.

Definition

```
CREATE METHOD DM_getGainsChart ()
  RETURNS TABLE (ROWCOUNT BIGINT, SUMACTUAL DOUBLE, THRESHOLD DOUBLE)
  FOR DM_RegTestResult
  BEGIN
  ---
  --- !! See Description
  ---
  END
```

Description

1) The result of the invocation *DM_getGainsChart()* is determined as follows:

   Case:

   a) A table containing gains chart information for *SELF*, if present in the model.

   b) Otherwise, the null value.

9.6 DM_RegApplTask Type and Routines

9.6.1 DM_RegApplTask Type

Purpose

The type DM_RegApplTask is a representation of all the information that constitutes a regression application task, in particular it contains the mining model and the data mining data for input and output.

Definition

```
CREATE TYPE DM_RegApplTask
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

STATIC METHOD DM_defRegApplTask
          (model DM_RegressionModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RegApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegMdl()
  RETURNS DM_RegressionModel
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegApplData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegOutpData()
  RETURNS DM_MiningData
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_getRegResFld()
  RETURNS CHARACTER VARYING (DM_MaxFieldAliasLength)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD DM_applyRegMdl()
  RETURNS DM_RegApplTask
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length of the *DM_content* of a *DM_RegApplTask*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for the alias name of a field.

Description

1) The type *DM_RegApplTask* provides for public use:

   a) a method *DM_defRegApplTask(DM_RegressionModel, DM_MiningData, DM_MiningData, CHARACTER VARYING),*

   b) a method *DM_getRegMdl(),*

   c) a method *DM_getRegApplData(),*

   d) a method *DM_getRegOutpData(),*

   e) a method *DM_getRegResFld(),*

   f) a method *DM_applyRegMdl().*

9.6.2 DM_defRegApplTask Method

Purpose

Return the DM_RegApplTask value containing the given DM_RegressionModel value, the given DM_MiningData values for input and output, and the given result field value.

Definition

```
CREATE STATIC METHOD DM_defRegApplTask
          (model DM_RegressionModel, inputData DM_MiningData,
           outputData DM_MiningData,
           resultField CHARACTER VARYING(DM_MaxFieldAliasLength))
  RETURNS DM_RegApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The method *DM_defRegApplTask* takes the following input parameters:

   a) a *DM_RegressionModel* value *model*.

   b) a *DM_MiningData* value *inputData*,

   c) a *DM_MiningData* value *outputData*,

   d) a *CHARACTER VARYING* value *resultField*,

2) The result of the invocation *DM_defRegApplTask(model, inputData, outputData, resultField)* is determined as follows:

   Case:

   a) If *model* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null model*.

   b) If *inputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null input data*.

   c) If *outputData* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – null output data*.

   d) If *resultField* is not the alias of a field of type *DM_RegResult* of *outputData*, then an exception condition is raised: *SQL/MM Data Mining exception – invalid result field*.

   e) If *model.DM_getRegMdlSpec().DM_isCompatible(inputData.genDataSpec())* returns FALSE, then an exception condition is raised: *SQL/MM Data Mining exception – data and data specification of model not compatible*.

   f) Otherwise, the *DM_RegApplTask* value containing the given arguments.

9.6.3 DM_getRegMdl Method

Purpose

Return the DM_RegressionModel value representing the regression model contained in the regression application task.

Definition

```
CREATE METHOD DM_getRegMdl()
  RETURNS DM_RegressionModel
  FOR DM_RegApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegMdl()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_RegressionModel* value, then the null value.

   b) Otherwise, it is the *DM_RegressionModel* value contained in *SELF*.

9.6.4 DM_getRegApplData Method

Purpose

Return the DM_MiningData value for input contained in SELF.

Definition

```
CREATE METHOD DM_getRegApplData()
  RETURNS DM_MiningData
  FOR DM_RegApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegApplData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for input, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for input contained in *SELF.*

9.6.5 DM_getRegOutpData Method

Purpose

Return the DM_MiningData value for output contained in SELF.

Definition

```
CREATE METHOD DM_getRegOutpData()
  RETURNS DM_MiningData
  FOR DM_RegApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getRegOutpData()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *DM_MiningData* value for output, then the null value.

   b) Otherwise, it is the *DM_MiningData* value for output contained in *SELF*.

9.6.6 DM_getRegResFld method

Purpose

Return the CHARACTER VARYING value for the result field contained in SELF.

Definition

```
CREATE METHOD DM_getRegResFld()
   RETURNS CHARACTER VARYING(DM_MaxFieldAliasLength)
   FOR DM_RegApplTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

1) The result of the invocation *DM_getRegResFld()* is determined as follows:

   Case:

   a) If *SELF* does not contain a valid representation of a *CHARACTER VARYING* value for the result field, then the null value.

   b) Otherwise, it is the *CHARACTER VARYING* value for the result field contained in *SELF*.

9.6.7 DM_applyRegMdl method

Purpose

Apply the regression model in the DM_RegApplTask value to each row of the table determined by the input data in the DM_RegApplTask value. The results of the application are written to the table determined by the output data. In addition, any number of columns from the input table can be copied to the output table.

Definition

```
CREATE METHOD DM_applyRegMdl()
  RETURNS DM_RegApplTask
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_applyRegMdl()* is determined as follows:

   Case:

   a) If the application run is successful, then *SELF*.

   b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – model application failed*.

2) In a successful application run, for each row of the table determined by the input data mining data contained in *SELF* one row is inserted into the table determined by the output data mining data contained in *SELF*. The column values are determined as follows:

   a) The value for the column, corresponding to the field of *SELF.DM_getRegOutpData()* whose alias name is *SELF.DM_getRegResFld()*, is the result of the application of the regression model to the input row. This is a value of type *DM_RegResult*.

   b) For each field of *SELF.DM_getRegOutpData()*, whose alias name is the same as an alias name of a field of *SELF.DM_getRegApplData()*, the value of the corresponding output column is the same as the value of the corresponding column of the input row.

   c) Otherwise, the null value.

9.7 DM_RegResult Type and Routines

9.7.1 DM_RegResult Type

Purpose

The DM_RegResult type is description of the result of an application run of a regression model.

Definition

```
CREATE TYPE DM_RegResult
  AS (
         DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
  )
  INSTANTIABLE
  NOT FINAL

METHOD DM_getPredValue()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_RegResult*.

Description

1) The *DM_RegResult* type provides for public use:

   a) a method *DM_getPredValue()*.

9.7.2 DM_getPredValue Method

Purpose

Returns the predicted value contained in the DM_RegResult value.

Definition

```
CREATE METHOD DM_getPredValue()
  RETURNS DOUBLE PRECISION
  FOR DM_RegResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getPredValue()* is is determined as follows:

   Case:

   a) If the result does not contain a predicted value, then the null value.

   b) Otherwise, a *DOUBLE PRECISION* value representing the predicted value contained in *SELF*.

10 Status Codes

The character string value returned in an SQLSTATE parameter comprises a 2-character class value followed by a 3-character subclass value. The class value for each condition and the subclass value or values for each class value are specified in Table 9 below.

The "Category" column has the following meanings: "S" means that the class value given corresponds to successful completion and is a completion condition; "W" means that the class value given corresponds to a successful completion but with a warning and is a completion condition; "N" means that the class value corresponds to a no-data situation and is a completion condition; "X" means that the class value given corresponds to an exception condition.

For a successful completion code but with a warning, the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class value for *warning* (defined in Subclause 23.1, "SQLSTATE", in part 2 of ISO/IEC 9075) and the third character of the SQLSTATE is 'H'.

Category	Condition	Class	Subcondition	Subclass
X	SQL/MM Data Mining exception	‡	alias already in use	F01
X	SQL/MM Data Mining exception	‡	data and data specification of model not compatible	F02
X	SQL/MM Data Mining exception	‡	field already defined	F03
X	SQL/MM Data Mining exception	‡	field not categorical	F04
X	SQL/MM Data Mining exception	‡	field not defined in data specification	F05
X	SQL/MM Data Mining exception	‡	field not numerical	F06
X	SQL/MM Data Mining exception	‡	invalid table name	F07
X	SQL/MM Data Mining exception	‡	invalid field name	F08
X	SQL/MM Data Mining exception	‡	invalid import format	F09
X	SQL/MM Data Mining exception	‡	invalid input data	F10
X	SQL/MM Data Mining exception	‡	mining field position out of range	F11
X	SQL/MM Data Mining exception	‡	model computation failed	F12
X	SQL/MM Data Mining exception	‡	no logical data specification defined	F13
X	SQL/MM Data Mining exception	‡	null settings	F14
X	SQL/MM Data Mining exception	‡	null input data	F15
X	SQL/MM Data Mining exception	‡	parameter out of range	F16
X	SQL/MM Data Mining exception	‡	invalid application input data format	F17
X	SQL/MM Data Mining exception	‡	null model	F18
X	SQL/MM Data Mining exception	‡	null output data	F19
X	SQL/MM Data Mining exception	‡	invalid result field	F20
X	SQL/MM Data Mining exception	‡	model application failed	F21
X	SQL/MM Data Mining exception	‡	model test failed	F22
X	SQL/MM Data Mining exception	‡	invalid settings format	F23
X	SQL/MM Data Mining exception	‡	data and data specification not compatible	F24
X	SQL/MM Data Mining exception	‡	sequence model cannot be applied to item sets	F25

Table 9 — SQLSTATE class and subclass values

‡ If the routine is implemented as an SQL-invoked routine, then the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class for *SQL routine exception* (defined in Subclause 23.1, "SQLSTATE", in part 2 of ISO/IEC 9075).

Otherwise, the routine is implemented as an external routine and the first two characters of the SQLSTATE are equal to the SQLSTATE condition code class for *external routine exception* (defined in Subclause 23.1, "SQLSTATE", in part 2 of ISO/IEC 9075).

11 Conformance

11.1 Requirements for conformance

11.1.1 Support for user-defined types and functions

A conforming implementation supports a user-defined type by:

1) providing a user-defined type with the name as specified by this part of ISO/IEC 13249, and
2) providing the functions for that user-defined type, either as the specified methods or as regular functions, where the signature of the function shall be identical to the <method selection> determined by Subclause <method invocation> of part 2 of ISO/IEC 9075:2003 and Subclause <static method invocation> of part 2 of ISO/IEC 9075:2003 respectively.

A conforming implementation does not need to support any of the attributes of user-defined types introduced by this part of ISO 13249.

11.1.2 Support for data mining phases

A conforming implementation shall state the type of conformance to be one of

1) Conformance of the application phase
2) Full conformance (training, testing and application phase)

If full conformance is stated, user-defined types shall be completely supported, with the exceptions stated below for particular data mining techniques.

If conformance of the application phase is stated, only a subset of the user-defined types and functions needs to be supported as specified in the remainder of clause 11 (Conformance).

11.1.3 Support for basic user-defined types

A conforming implementation shall support the following user-defined types:

1) DM_LogicalDataSpec

   For conformance of the application phase only the following methods need to be supported:

   a) DM_getNumFields,
   b) DM_getFldName,
   c) DM_getFldType,
   d) DM_isCompatible.

2) DM_MiningData

11.1.4 Support for data mining techniques

A conforming implementation shall support at least one of the groups of public user-defined types and routines specified in Subclause 11.2, “Claims of conformance”. They correspond to the following data mining techniques:

a) data mining association rule discovery,

b) data mining sequence rule discovery,

c) data mining clustering,

d) data mining classification,

e) data mining regression.

11.2 Claims of conformance

Claims of conformance to this part of ISO/IEC 13249 shall state:

1) If full conformance is claimed, the following groups of public user-defined types are supported for each of the data mining techniques:

   a) data mining association rule discovery:
   DM_RuleBldTask, DM_RuleSettings (except methods DM_setSequence and DM_getSequence), DM_RuleModel (except methods DM_getNumSequences, DM_getSequences, DM_getSeqDetails, DM_getSeqRules), DM_RuleFilter, DM_RuleApplTask.

   b) data mining rule discovery (association and sequence rules):
   DM_RuleBldTask, DM_RuleSettings, DM_RuleModel, DM_RuleFilter, DM_RuleApplTask.

   c) data mining clustering:
   DM_ApplicationData, DM_ClusBldTask, DM_ClusSettings, DM_ClusteringModel, DM_ClusResult, DM_ClusApplTask.

   d) data mining classification:
   DM_ApplicationData, DM_ClasBldTask, DM_ClasSettings, DM_ClasModel, DM_ClasResult, DM_ClasTestResult, DM_ClasTestTask, DM_ClasApplTask.

   e) data mining regression:
   DM_ApplicationData, DM_RegBldTask, DM_RegSettings, DM_RegressionModel, DM_RegResult, DM_RegTestResult, DM_RegTestTask, DM_RegApplTask.

2) If conformance of the application phase is claimed, the following groups of public user-defined types with the specified methods are supported:

   a) DM_RuleModel with
      - DM_impRuleModel,
      - DM_applyRuleModel,

   DM_RuleApplTask with all methods.

   b) DM_ApplicationData with all methods,
   DM_ClusteringModel with
      - DM_impClusModel,
      - DM_applyClusModel,
      - DM_getClusMdlSpec,

   DM_ClusResult with all methods,
   DM_ClusApplTask with all methods.

   c) DM_ApplicationData with all methods,
   DM_ClasModel with
      - DM_impClasModel,
      - DM_applyClasModel,
      - DM_getClasMdlSpec,
      - DM_getClasTarget,

   DM_ClusResult with all methods,
   DM_ClusApplTask with all methods.

d) DM_ApplicationData with all methods,
DM_RegModel with
- DM_impRegModel,
- DM_applyRegModel,
- DM_getRegMdlSpec,
- DM_getRegTarget,

DM_RegResult with all methods
DM_RegApplTask with all methods.

3) The definitions for all elements and actions that this part of ISO/IEC 13249 specified as implementation-defined.

Blank page

Annex A
(informative)

Implementation-defined Elements

This Annex references those features that are identified in the body of this part of ISO/IEC 13249 as implementation-defined.

The term implementation-defined is used to identify characteristics that may differ between implementations, but that shall be defined for each particular implementation.

A.1 Implementation-defined Meta-variables

1) *DM_MaxContentLength* is the implementation-defined maximum number of characters of the *DM_content* attribute of the introduced user-defined types. It is also the maximum number of characters for import or export representations of values of type *DM_ApplicationData, DM_RuleSettings, DM_RuleModel, DM_ClusSettings, DM_ClusteringModel, DM_ClasSettings, DM_ClasModel, DM_RegSettings,* and *DM_RegModel*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

3) *DM_MaxTableNameLength* is the implementation-defined maximum number of characters for the name of a table in method *DM_defMiningData*.

4) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a data field.

5) *DM_MaxItemLength* is the implementation-defined maximum number of characters for a name of an item used by association or sequence rule discovery. Such items may occur as entries in the HEAD or HEADNAME columns of tables returned by method *DM_getRules* as well as the ITEM or ITEMNAME columns of tables returned by *DM_getRuleBodies, DM_applyRuleModel*, *DM_getItemSets,* and *DM_getItemConstrs*. They also occur as item parameters for the method *DM_setItemConstr*.

6) *DM_MaxBodyLength* is the implementation-defined maximum number of characters for an entry in the BODYTEXT column of tables returned by method *DM_getRules*. It is also the maximum number of characters for an entry in the SEQTEXT column of tables returned by method *DM_getSequences*, in the ITEMSETTEXT column of tables returned by method *DM_getSeqDetails*, as well as columns BODYSEQTEXT and HEADSEQTEXT of tables returned by method *DM_getSeqRules.*

7) *DM_MaxItemsetLength* is the implementation-defined maximum number of characters for a value of an item set.

8) *DM_MaxClusterNameLength* is the implementation-defined maximum number of characters of a cluster name. Cluster names are returned in the NAME column of the table returned by method *DM_getClusters*.

9) *DM_MaxClassLabelValueLength* is the implementation-defined maximum number of characters for a value of a class label. Class label parameters are used by methods *DM_setClasCost, DM_getRankQuality,* and *DM_getPredAccuracy*. They are returned by methods *DM_getClasCost, DM_getClass,* and *DM_getPredClass*.

A.2 Other Implementation-defined Elements

1) The default type of a field in a value of type DM_LogicalDataSpec is implementation-defined.

2) The mapping of datetime values from input tables to numbers to be used by data mining algorithms is implementation-defined.

3) The cluster names in the NAME column of a table returned by method *DM_getClusters* is implementation-defined.

4) The textual description of rule bodies in the BODYTEXT column of a table returned by method *DM_getRules* is implementation-defined.

5) The textual description of sequences in the SEQTEXT column of a table returned by method *DM_getSequences* is implementation-defined.

6) The textual description of item sets in the ITEMSETTEXT column of a table returned by method *DM_getSeqDetails* is implementation-defined.

7) The textual description of sequences in the BODYSEQTEXT and HEADSEQTEXT columns of a table returned by method *DM_getSeqRules* is implementation-defined.

Annex B
(informative)

Implementation-dependent elements

This Annex references those places where this part of ISO/IEC 13249 states explicitly that the actions of a conforming implementation are implementation-dependent.

The term implementation-dependent is used to identify characteristics that may differ between implementations, but that are not necessarily specified for any particular implementation.

B.1 Implementation-dependent elements

1) The *position* of a field in a value of type *DM_LogicalDataSpec* is implementation-dependent.

2) The *position* of a class label value in a value of type *DM_ClasTestResult* is implementation-dependent.

3) The import and export representation of data mining settings, i.e. a CHARACTER LARGE OBJECT representation of values of type *DM_RuleSettings, DM_ClusSettings, DM_ClasSettings,* and *DM_RegSettings* is implementation-dependent. But for any one implementation, the import and export formats for each data mining function shall be the same.

4) The data mining models of type *DM_RuleModel, DM_ClusteringModel, DM_ClasModel,* and *DM_RegModel* generated by the corresponding build methods *DM_buildRuleModel, DM_buildClusModel, DM_buildClasModel,* and *DM_buildRegModel* are implementation-dependent.

5) The values in the column BODYTEXT generated by the method *DM_getRules* is an implementation-dependent textual description of rule bodies; they are provided for print-out purposes.

6) If the value of *validationData* in a build task of type *DM_ClasBldTas*k or *DM_RegBldTask* is the null value, then an implementation-dependent subset of the training data is used for internal validation.

Blank page

Annex C
(informative)

C.1 A Scenario Using the Classification Technique

This informative annex presents an example technique on how the user-defined types of part 6 of ISO 13249 might be used in a data warehouse environment. The given example uses again classification as the example case.

The following should be the application scenario:

1) Given a customer table CT of an insurance company with columns C1, C2, …, C9 and a column r containing the risk class of a customer (e.g. 'low', 'medium', 'high').

2) The objective is to compute a classification model such that the risk for a new customer can be predicted.

3) It should be possible to re-compute the classification model from time to time to see whether the model changes over time.

4) It shall be possible to test a classification model with a given test table TT also containing columns C1, C2, …, C9 and r.

As indicated in **Figure 5 — Overview of user-defined types for the training phase** on page 20, the first steps are to define the values of the user-defined types needed to store the information for a training run, i.e. to create a DM_ClasBldTask value. Due to requirement 3) the DM_ClasBldTask value has to be stored to be able run the data mining training several times. For this purpose the standard assumes to have a table for mining tasks MT with columns ID of type INTEGER and TASK of type DM_ClasBldTask. The following list enumerates the steps to define a DM_ClasBldTask:

1) Create a DM_MiningData value using the static method DM_defMiningData.

2) Create a DM_LogicalDataSpec value using the method DM_genDataSpec of the DM_MiningData value.

3) Create a DM_ClasSettings value using the default constructor and assign the DM_LogDataSpec value.

4) Declare the column named ‘r’ as the target field using the DM_setClasTarget method.

5) Create a DM_ClasBldTask value using the DM_defClasBldTask method.

6) Store the newly created DM_ClasBldTask value in table MT.

All the steps described above can be expressed as a single SQL statement:

```
WITH MyData AS (
   DM_MiningData::DM_defMiningData('CT')
)
INSERT INTO MT (ID, TASK)
VALUES (
   1,
   DM_ClasBldTask::DM_defClasBldTask(
                           MyData, NULL,
                           (
                                 (new DM_ClasSettings())

   .DM_useClasDataSpec(MyData.DM_genDataSpec())
                           ).DM_clasSetTarget('r')
   )
)
```

Now that the DM_ClasBldTask value is generated and stored in the MT table, the classification training can be initiated and the classification model is computed. Since the model shall be used in later application and test runs, it is stored in a table MM having two columns ID of type integer and MODEL of type DM_ClasModel:

```
INSERT INTO MM (ID, MODEL)
VALUES (
   1,
   MyTask.DM_buildClasModel()
)
```

A simple test of the model can be done using the data used to train the model:

```
SELECT MODEL.DM_testClasModel(DM_MiningData::DM_defMiningData('CT'))
FROM MM
WHERE ID = 1
```

Again the table used to train the model is used in the following example for an application of the model:

```
WITH MyModel AS (
   SELECT MODEL FROM MM
   WHERE ID = 1
)
SELECT C1, C2, C3, C4, C5, C6, C7, C8, C9,
   MyModel.applyModel(
                           DM_ApplicationData::DM_impApplData(
                                 '<C1>' + C1 + '</C1>' +
                                 '<C2>' + C2 + '</C2>' +
                                 '<C3>' + C3 + '</C3>' +
                                 '<C4>' + C4 + '</C4>' +
                                 '<C5>' + C5 + '</C5>' +
                                 '<C6>' + C6 + '</C6>' +
                                 '<C7>' + C7 + '</C7>' +
                                 '<C8>' + C8 + '</C8>' +
                                 '<C9>' + C9 + '</C9>'
                           )
   )
FROM CT
```

Index

An index entry appearing in **boldface** indicates a range of pages where a user-defined type is defined in this part of ISO/IEC 13249. All other index entries appear in roman type.

An index page number appearing in **boldface** indicates a page or range of pages where the attribute, routine, or user-defined type is specified. An index page number appearing in *italics* indicates a page where the word, phrase, attribute, routine, or user-defined type is defined. An index page number appearing in roman type indicate a page where the word, phrase, attribute, routine, or type was used.

–F–

–G–

–L–

–M–

–P–

–Q–

–R–

–S–

–T–

–V–