ISO/IEC JTC 1/SC 32 N _____

Date: 2000-07-27

ISO/IEC 13249-6:200x (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: Working Draft
Document subtype: Not applicable
Document stage: (00) Preliminary
Document language: E

Blank page

Contents Page

Figures Page

Tables 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 3.

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 of 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 paten 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 B of this part of ISO/IEC 13249 are for information only.

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 Model Types”, defines the user-defined types and associated routines for data mining models.

6) Clause 6, “Data Mining Settings Types”, defines the user-defined types and associated routines for data mining settings

7) Clause 7, “Data Mining Application Result Types”, defines the user-defined types and associated routines for the results of the application of a data mining model.

8) Clause 8, “Data Mining Test Result Types”, defines the user-defined types and associated routines for the results of a test of a data mining model.

9) Clause 9, “Data Mining Data Types”, defines the user-defined types and associated routines for the input data definitions of data mining.

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 list 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.

In the text of this part of ISO/IEC 13249, Clauses begin a new odd-numbered page, and in Clause 5, "Data Model Types", through Clause 9, "Data Mining Data Types", subclauses begin a new page. Any resulting blank space is not significant.

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 aspects of data mining data such as data mining models, data mining settings, and data mining 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 9075-1:1999, Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).

ISO/IEC 9075-2:1999, Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).

ISO/IEC 9075-4:1999, Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM).

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

2.2 Publicly-available specifications

World Wide Web Consortium (W3C), *Extensible Markup Language (XML 1.0),*
http://www.w3.org/TR/ 1998/REC-xml-19980210

Object Management Group (OMG), *XML Metadata Interchange (XMI)*,
ftp://ftp.omg.org/pub/docs/ad/98-10-05.pdf

Data Mining Group (DMG), *Predictive Model Markup Language (PMML)*,
http://www.dmg.org/public/techreports/pmml-1_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 ISO/IEC 13249-1.

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
active field
An active field is a field that is used during the mining run to compute a model or evaluate or predict a single row. The active fields are defined in the data mining mapping.

3.1.2.2
application phase
Only used for clustering, classification and regression at the moment. During the application of a data mining model a row is evaluated against a model and one or more values are computed: the identification of the cluster for clustering, the predicted value for classification or regression.

3.1.2.3
association rules
Given a set of rows with a set of fields out of which one is a grouping field (e.g. an identifier of a purchase transaction). An association rule identifies combinations of field values which appear frequently (according to a specified threshold), i.e. there is a significant number of groups containing the combination of the values.
Example of application: store layout.

3.1.2.4
categorical field type
A field being of categorical field type supports only the <equal operator> as a comparison operator. There is no defined order and no arithmetic operations are supported.

3.1.2.5
class label
The class label is the target field used in classification to create a prediction model, whose application to additional data without a class label allows it to predict a value for this class label. The class label must have a categorical field type.

3.1.2.6
classification
Given a set of rows with a set of fields and a special categorical target field, called *class label*, classification computes a prediction model such that the class label can be predicted by using the model and a set of field values without the class label. Classification optimizes the model such that a class label can be predicted with a minimal number of field values.
Example of application: insurance risk prediction.

3.1.2.7
classification error
The classification error is the percentage of wrongly predicted values among the total number of values predicted during the validation phase of the classification technique. This error is returned at the end of the training phase.

3.1.2.8
classification model
The result of classification.

3.1.2.9
cluster
A cluster is a set of rows with common characteristics (see clustering).

3.1.2.10
clustering
Given a set of rows with a set of fields, clustering is the process of discovering sets of rows with common characteristics - the *clusters*. Rows are possibly homogeneous inside a cluster, and possibly heterogeneous between two clusters. Clustering characterizes each cluster by field values and ranks the fields such that the most distinguishing fields come first.
Example of application: customer mailings.

3.1.2.11
cost rate
The cost rate is a parameter specific to the classification settings. It defines the maximal classification error tolerated during the validation phase. If the classification error is greater than the cost rate, the model is considered not good enough and the training phase continues to refine this model.

3.1.2.12
data mining
Data mining is the process of discovering hidden, previously unknown and usable information from a large amount of data. The data is analyzed without any expectation on the result. Data mining delivers knowledge that can be used for a better understanding of the data.

3.1.2.13
data mining data type
Data mining data type represents an abstraction of a source data. It defines the location of the data source, e.g. a table or a view, where fields (column names) can be found. Additionally, a data mining data type is able to map the field names to alias names. These alias names will be compared before the mining run to the field names used in the specified data mining mapping.

3.1.2.14
data mining model
A data mining model is the result of the mining run (training phase) of a data mining technique over a given set of data. The model can be understood as a structure containing the information discovered in the data during the mining run. This structure contains the usable information that will be displayed to the end-user. Note that the 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).

3.1.2.15
data mining mapping
A data mining mapping defines the field names and field types that will be used during the data mining run. It is part of the settings of each data mining technique. Note that these fields will be compared before the mining run to the field names or aliases defined in the data mining data type.

3.1.2.16
data mining settings
Data mining settings specify the use of mining fields during the mining run, as well as some constraints for each data mining technique. The mining fields are defined in a mining mapping. Other parameters dependend on the data mining technique and do also belong to the data mining settings.

3.1.2.17
data mining technique
Four types of data mining techniques are distinguished: discovering association rules, clustering/segmentation, classification and regression.

3.1.2.18
data mining test result
A data mining test result is the statistical result returned at the end of the testing phase. It contains an evaluation of the correctly and incorrectly predicted values during this phase. This result can be considered as a quality criterion of the data mining model used for the testing phase.

3.1.2.19
error weight
The error weight is a parameter specific to the classification settings. It defines a weight that has to be applied to a wrong prediction during the validation phase. The error weight depends on the wrongly predicted value and the real value of the target field in the tested row.

3.1.2.20
input mapping
See data mining mapping.

3.1.2.21
mining field type
A data mining field is either categorical or numeric. The field type defines the way the field values will be compared and managed during the mining run.

3.1.2.22
mining task
A mining task is an abstraction for the complete data needed to compute a mining model. It contains the data mining settings and the data mining data value used to compute the model.

3.1.2.23
numeric field type
A field being of numeric field type supports all operations of <numeric value expression>s.

3.1.2.24
predicted value
The result of a prediction using a data mining model. The value can either be a categorical or numeric value.

3.1.2.25
regression
Regression is very similar to classification except for the type of the target field. Rather than predicting a class label, regression is predicting a numeric value. Hence, the predicted value might not be identical with any value contained in the data used to build the model.
Example of application: customer ranking.

3.1.2.26
regression model
The result of regression.

3.1.2.27
segmentation
see clustering.

3.1.2.28
support
Support is a quantity criterion of an association rule. The support of the association rule X+Y=>Z is the number of purchase transactions that contains X, Y and Z.

3.1.2.29
r-square
The r-square value is the maximum allowed squared Pearson correlation coefficient for a training run. This r-square value can be tolerated in the regression model for the error on verification data.

3.1.2.30
target field
The target field is the special field used in classification and regression whose value will be predicted. In classification, the target field is called a class label and has a categorical field type. In regression, the target field has a numeric field type.

3.1.2.31
testing phase
Only used for classification and regression. The testing phase reads a set of rows containing values for the special field, and evaluates each row in the application phase. Then, the predicted value is compared to the actual value in the special field.

3.1.2.32
training data
Data used as input for the training phase.

3.1.2.33
training phase
Common to all data mining techniques, this is the phase in which the data mining model is computed.

3.1.2.34
validation phase
The validation phase is part of the training phase for classification and regression techniques. It uses the calculated prediction model and another set of rows to test these rows against the model, as described in the testing phase.

3.2 Notations

3.2.1 Notations provided in Part 1

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

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:

- “imp” for “import”,
- “def” for “define”,
- “NO” for “number of”,
- “Clas” for “classification”,
- “Clus” for “clustering”,
- “Reg” for “regression”,
- “Pred” for “predicted”,
- “Src” for source,
- “Fld” for “field”,

- “Appl” for “application”,
- “Map“ for “mapping“.

3.3 Conventions

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

Blank page

4 Concepts

4.1 Data mining techniques

Data mining is an emerging field of technology which develops techniques to find “previously unknown information in large amounts of data”. Four techniques have been commonly accepted so far and are widely used in applications:

1) The search for association rules:
   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: store layout.

2) Clustering / Segmentation:
   Given a set of rows with a set of fields. Find sets of rows with common characteristics - the so-called clusters. Characterize each cluster by field values and rank the fields such that the most distinguishing fields come first.
   Example application: customer mailings.

3) Classification
   Given a set of rows with a set of fields and a special field the so-called class label. Compute a classification model such that the class label can be predicted by using the model and a set of field values without the class label. Optimise 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 patterns

Three major phases can be distinguished in data mining: the training 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

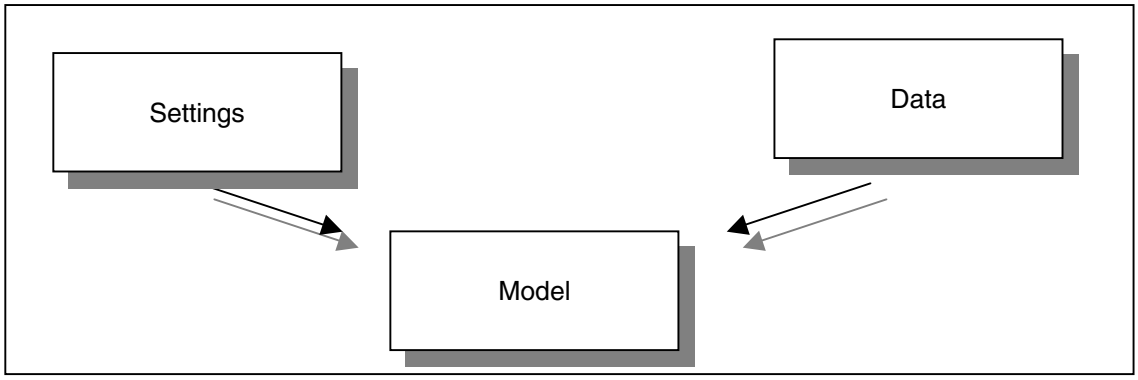


Figure 1: The training phase

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

Essentially, the data mining needs some *settings* and a set of 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 Application phase

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

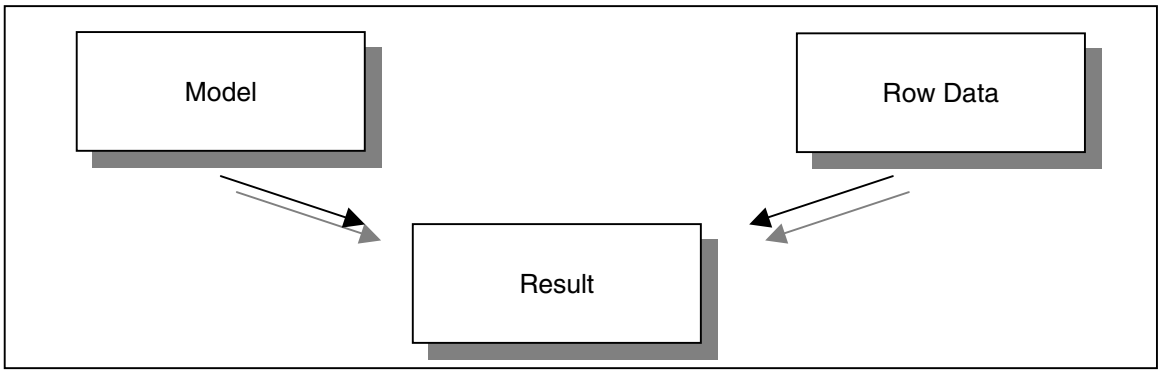


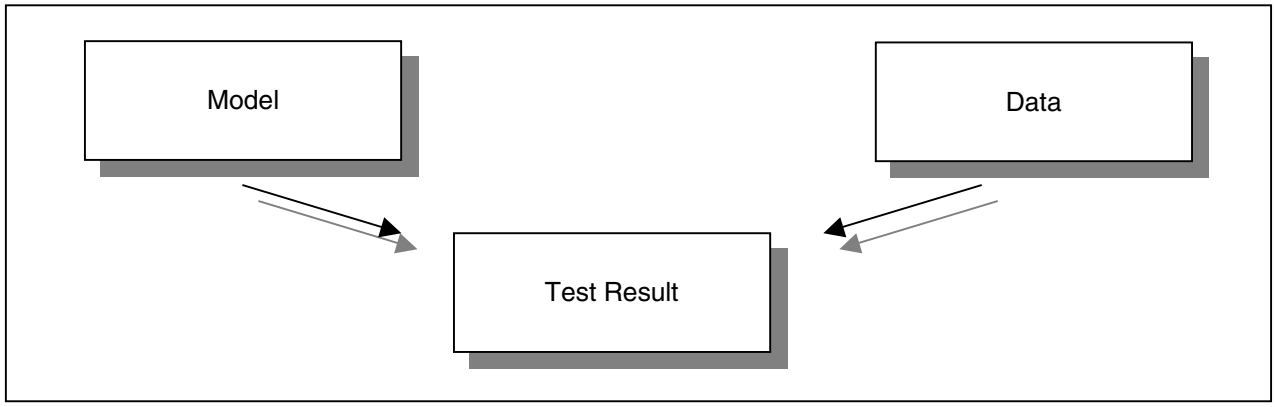
Figure 2: The application phase

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

4.2.3 Test 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 rows with the special training field is read and for each row the application of the mining model is invoked. Then, the predicted value is compared to the actual value in the special field. When all rows are read, a statistical result about the false predictions is computed and returned as the test result.

Figure 3: 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 be able to work with relatively small parts that can be put together in many different ways. 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 entities 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

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.

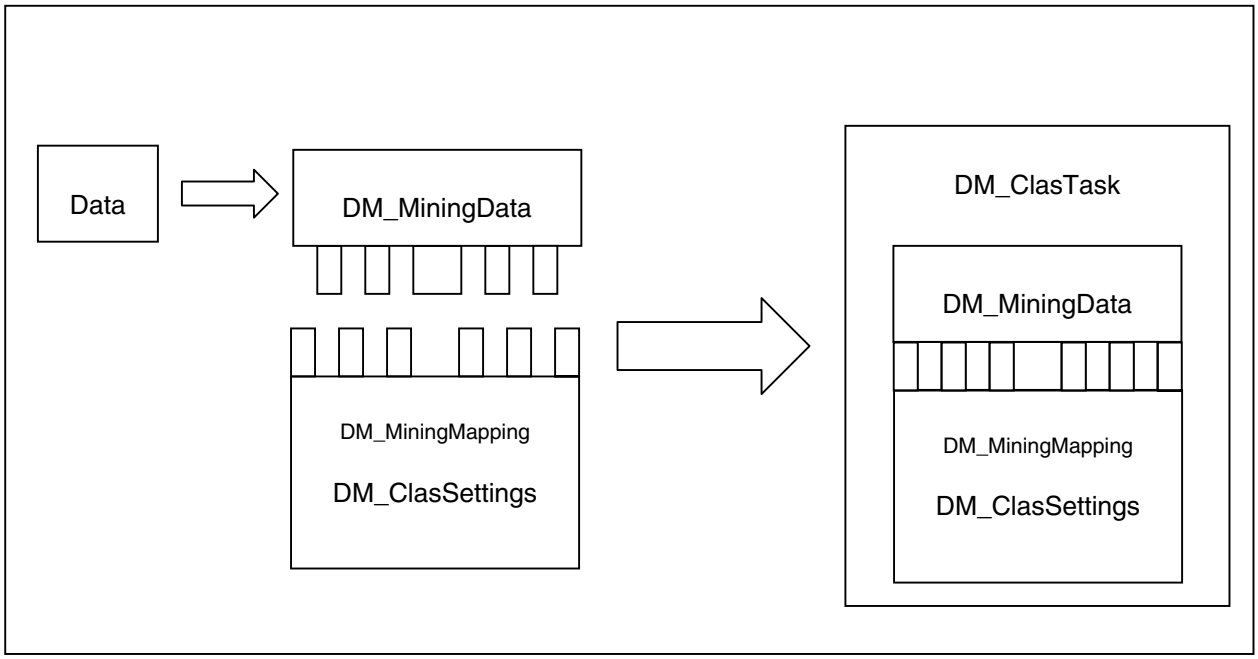


Figure 4: Overview of user-defined types for definitions

Figure 4 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 mining mapping. In a next step, a representation of real data (the DM_MiningData) is assigned which completes the definition of a DM_ClasTask object. However, one can also start by defining a DM_MiningData value. The mining mapping can simply be derived from this mining data value and can then be used to define a mining settings. Finally, the settings with the mapping and the representation of the data are connected in the classification task object.

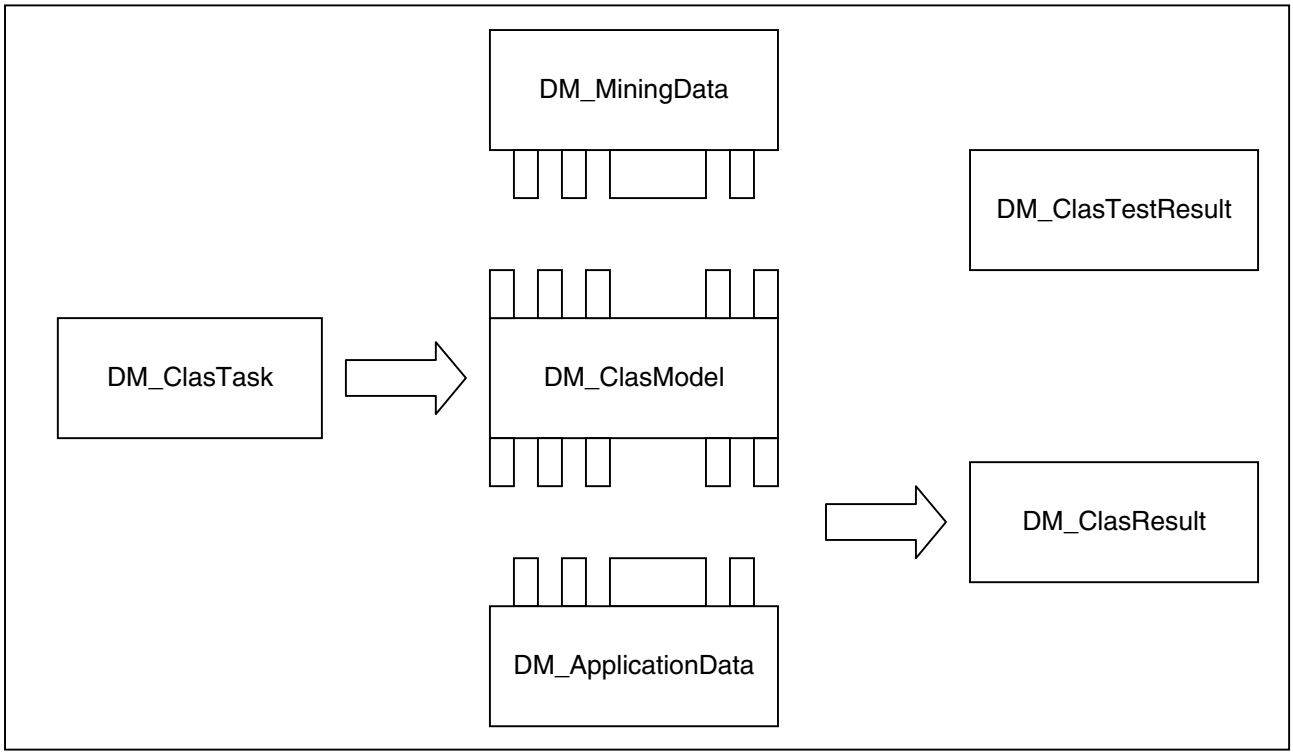


Figure 5: Overview of user-defined types for training, test and application

The user-defined types introduced in figure 4 were introduced to allow for the storage and flexible combinations of definitions used for data mining. The UDTs shown in figure 5 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 data mining mapping (not necessarily identical to the data mining mapping of the data mining task). Objects of type mining data or application data must match this mapping 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.

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

The user-defined types for the data mining data, the data mining mapping, and the data mining application data represent information which is not mining technique specific. These types provide the basic infra-structure for the mining technique specific types.

1) The DM_MiningData user-defined type is an abstraction for real data contained in tables or views. However, it just stores the metadata to access the real data sources and any other information necessary to make the real data accessible for a later data mining training or test run. This might include internal transformations used to clean or pre-process data in order to increase the quality data mining results.

2) The DM_MiningMapping 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 data mining filed type. The data mining field type defines how a 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 must be mapped onto the mining mapping of the mining model. The names of the values are used to implement that mapping.

4.3.4 User-defined types that are related to mining techniques

The types introduced in subclause 4.3.3 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_ClasTask type are intended to provide all information actually needed to start a data mining training run and to compute a data mining model. In a warehouse environment, this type might be used to store fully defined 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.

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 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 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 ISO/IEC 9075-2. 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.

4.4 A scenario using the classification technique

This subclause presents an example 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 must 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 4, 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_ClasTask value. Due to requirement 3) the DM_ClasTask 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_ClasTask. The following list enumerates the steps to define a DM_ClasTask:

1) Create a DM_MiningData value using the static method DM_defMiningData.

2) Create a DM_MiningMapping value using the method DM_genMiningMap of the DM_MiningData value.

3) Create a DM_ClasSettings value using the default constructor and assign the DM_MiningMapping value as the mapping to use.

4) Declare the column named ‘r’ as the predicted field using the DM_clasSetTarget method.

5) Create a DM_ClasTask value using the DM_defClasTask method.

6) Store the newly created DM_ClasTask 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_ClasTask::DM_defClasTask(
      MyData, NULL,
      (
         (new DM_ClasSettings())
         .DM_clasUseMapping(MyData.DM_genMiningMap())
      ).DM_clasSetTarget('r')
   )
)
```

Now that the DM_ClasTask 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
```

5 Data Mining Model Types

The types in this family provide for the storage and retrieval of data mining model values.

5.1 DM_RuleModel Type and Routines

5.1.1 DM_RuleModel Type

Purpose

The *DM_RuleModel* type represents models which are the result of the search for association rules.

Definition

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

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

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

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

METHOD DM_getRuleTask()
   RETURNS DM_RuleTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

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

Description

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

   a) a method DM_impRuleModel(CHARACTER LARGE OBJECT(DM_MaxContentLength)),

   b) a method DM_expRuleModel(),

c) a method DM_getNORules(),

d) a method DM_getRuleTask().

5.1.2 DM_impRuleModel Method

Purpose

Return a specified value of type *DM_RuleModel*.

Definition

```
CREATE STATIC METHOD DM_impRuleModel
   (input 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_MiningModel*.

Description

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

   a) a CHARACTER LARGE OBJECT value input.

2) The result of the invocation *DM_impRuleModel(CHARACTER LARGE OBJECT)* is determined as follows:

   Case:

   a) If *input* is a proper representation of a *DM_RuleModel*, then a new value of type *DM_RuleModel*.

   b) Otherwise, the null value.

5.1.3 DM_expRuleModel Method

Purpose

Return the CHARACTER LARGE OBJECT representation of the association rules 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

The result of the invocation *DM_expRuleModel()* is a CHARACTER LARGE OBJECT representing the association rule model contained in SELF.

5.1.4 DM_getNORules Method

Purpose

Returns the number of rules contained in the *DM_content* of an value of *DM_RuleModel*.

Definition

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

Description

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

   Case:

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

   b) Otherwise, the number of rules contained in *DM_content*.

5.1.5 DM_getRuleTask method

Purpose

Return the DM_RuleTask contained in the DM_RuleTask.

Definition

```
CREATE METHOD DM_getRuleTask()
   RETURNS DM_RuleTask
   FOR DM_RuleModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) If SELF does not contain the information about the DM_RuleTask value which was used to compute SELF, then the null value.

   b) Otherwise, it is the DM_RuleTask contained in SELF which was used to compute the model.

5.2 DM_ClusteringModel Type and Routines

5.2.1 DM_ClusteringModel Type

Purpose

The *DM_ClusteringModel* type represents models which are the result of a segmentation.

Definition

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

STATIC METHOD DM_impClusModel
   (input 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,

METHOD DM_clusGetTask()
   RETUNRS DM_ClusTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

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

METHOD DM_clusModelMap()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_applyClusModel
   (input DM_ApplicationData)
   RETURNS DM_ClusResult
   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*.

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_clusGetTask(),*

   d) a method *DM_getNOClusters(),*

   e) a method *DM_clusModelMap(),*

   f) a method *DM_applyClusModel(input DM_ApplicationData).*

5.2.2 DM_impClusModel Method

Purpose

Return the *DM_ClusteringModel* determined by the given string.

Definition

```
CREATE STATIC METHOD DM_impClusModel
   (input 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_MiningModel*.

Description

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

   a) a CHARACTER LARGE OBJECT value *input*.

2) The result of the invocation *DM_impClusModel(CHARACTER LARGE OBJECT*) is determined as follows:

   Case:

   a) If *input* is a proper representation of a *DM_ClusteringModel*, then a value of type *DM_ClusteringModel*.

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

5.2.3 DM_expClusModel Method

Purpose

Return a string representing the clustering model of the *DM_ClusteringModel*.

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_MiningModel*.

Description

1) The result of the invocation *DM_expClusModel()* is a CHARACTER LARGE OBJECT representing the clustering model contained in SELF.

5.2.4 DM_getNOClusters Method

Purpose

Returns the number of clusters contained in the *DM_content* of a value of type *DM_ClusteringModel*.

Definition

```
CREATE METHOD DM_getNOClusters()
   RETURNS INTEGER
   FOR DM_ClusteringModel
   BEGIN
      --
      -- !! See Description
      --
   END
```

Description

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

   Case:

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

   b) Otherwise, the number of clusters contained in *DM_content*.

5.2.5 DM_applyClusModel Method

Purpose

Return the DM_ClusResult representing the result of applying the clustering model contained in SELF to the given *DM_ApplicationData*.

Definition

```
CREATE METHOD DM_applyClusModel(input 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 *input*.

2) The result of the invocation *DM_applyClusModel(input)* is determined as follows:

   Case:

   a) If the set of fields of *input* is not compatible with the mining mapping of the clustering model, then an exception is raised: *SQL/MM Data Mining exception –data and mapping not compatible*.

   b) Otherwise, it is the *DM_ClusResult* representing the result of an application of the clustering model to *input*.

5.2.6 DM_clusGetTask method

Purpose

Return the DM_ClusTask value used to create this model.

Definition

```
CREATE METHOD DM_clusGetTask()
   RETURNS DM_ClusTask
   FOR DM_ClusteringModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) If SELF does not contain the information about the *DM_ClusTask* value which was used to create the model, then the null value.

   b) Otherwise, it is the *DM_ClusTask* used to compute the model.

5.2.7 DM_clusModelMap method

Purpose

Return the *DM_MiningMapping* representing the set of fields needed for an application of the *DM_ClusteringModel*.

Definition

```
CREATE METHOD DM_clusModelMap ()
  RETURNS DM_MiningMapping
  FOR DM_ClusteringModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation of *DM_clusModelMap ()* is the *DM_MiningMapping* representing the set of fields needed for an application of this model.

5.3 DM_RegressionModel Type and Routines

5.3.1 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
   (input 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,

METHOD DM_regGetTask()
   RETUNRS DM_RegTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_regGetRSquared()
   RETURNS DOUBLE
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_regModelMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_applyRegModel
   (input DM_ApplicationData)
   RETURNS DM_RegResult
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_testRegModel
   (input DM_MiningData)
   RETURNS DM_RegTestResult
   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_RegressionModel*.

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_regGetTask(),*

   d) a method *DM_regGetRSquared(),*

   e) a method *DM_regModelMapping(),*

   f) a method *DM_applyRegModel(input DM_ApplicationData)*,

   g) a method *DM_testRegModel(input DM_MiningData).*

5.3.2 DM_impRegModel Method

Purpose

Return the *DM_RegressionModel* determined by the given input string.

Definition

```
CREATE STATIC METHOD DM_impRegModel
   (input 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_MiningModel*.

Description

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

   a) a CHARACTER LARGE OBJECT value *input*.

2) The result of the invocation *DM_impRegModel(CHARACTER LARGE OBJECT*) is determined as follows:

   Case:

   a) If input is a proper representation of a *DM_RegressionModel*, then a corresponding *DM_RegressionModel*.

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

5.3.3 DM_expRegModel Method

Purpose

Return a CHARACTER LARGE OBJECT 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_MiningModel*.

Description

1) The result of the invocation *DM_expRegModel()* is a CHARACTER LARGE OBJECT representing the regression model contained in SELF.

5.3.4 DM_applyRegModel Method

Purpose

Return the DM_RegResult representing the result of applying the regression model contained in SELF to a given *DM_ApplicationData*.

Definition

```
CREATE METHOD DM_applyRegModel(input 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 *input*.

2) The result of the invocation *DM_applyRegModel(input)* is determined as follows: :

   Case:

   a) If the set of fields of *input* is not compatible with the mining mapping of the regression model, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   b) Otherwise, it is a value of *DM_RegResult* representing the result of an application of the regression model to *input*.

5.3.5 DM_testRegModel Method

Purpose

Return the DM_RegTestResult representing the result of testing the regression model contained in SELF, using a given *DM_MiningData*.

Definition

```
CREATE METHOD DM_testRegModel(input 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 *input*.

2) The result of the invocation *DM_testRegModel(input)* is determined as follows: :

   Case:

   a) If the mining mapping of *input* is not compatible with the mining mapping of the regression model, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   b) Otherwise, it is the *DM_RegTestResult* representing the result of testing SELF with *input* as test data.

5.3.6 DM_regGetRSquared 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_RegTask value used to generate the model.

Definition

```
CREATE METHOD DM_regGetRSquared()
   RETURNS DOUBLE PRECISION
   FOR DM_RegressionModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) If the null value was specified as the validation data for the *DM_RegTask* value used to compute the model, then the squared Pearson correlation coefficient computed on an implementation-dependent subset of the training data used for validation purposes.

   b) Otherwise, it is the squared Pearson correlation coefficient of the regression model computed on the validation data.

5.3.7 DM_regTask method

Purpose

Return the DM_RegTask value used to create this model.

Definition

```
CREATE METHOD DM_regTask()
  RETURNS DM_RegTask
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

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

   a) If SELF does not contain the information about the *DM_RegTask* used to compute the model, then the null value.

   b) Otherwise, it is the DM_RegTask used to compute the model.

5.3.8 DM_regModelMapping method

Purpose

Return the *DM_MiningMapping* value specifying the set of fields necessary for an application or test of this model.

Definition

```
CREATE METHOD DM_regModelMapping ()
  RETURNS DM_regModelMapping
  FOR DM_RegressionModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_regModelMapping ()* is the DM_MiningMapping specifying the set of fields necessary for an application or test of this model.

5.4 DM_ClasModel Type and Routines

5.4.1 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
   (input 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,

METHOD DM_clasGetTask()
   RETURNS DM_ClasTask
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_clasCostRate()
   RETURNS DOUBLE
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_clasModelMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_applyClasModel
   (input DM_ApplicationData)
   RETURNS DM_ClasResult
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_testClasModel
   (input DM_MiningData)
   RETURNS DM_ClasTestResult
   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_ClasModel*.

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_clasGetTask(),*

    d) a method *DM_clasCostRate(),*

    e) a method *DM_clasModelMapping(),*

    f) a method *DM_applyClasModel(input DM_ApplicationData)*,

    g) a method *DM_testClasModel(input DM_MiningData).*

5.4.2 DM_impClasModel Method

Purpose

Return a specified value of type *DM_ClasModel*.

Definition

```
CREATE STATIC METHOD DM_impClasModel
   (input 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_MiningModel*.

Description

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

   a) a CHARACTER LARGE OBJECT value *input*.

2) The result of the invocation *DM_impClasModel(CHARACTER LARGE OBJECT*) is determined as follows:

   Case:

   a) If input is a proper representation of a *DM_ClasModel*, then a value of type *DM_ClasModel*.

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

5.4.3 DM_expClasModel Method

Purpose

Return a CHARACTER LARGE OBJECT representing the classification model contained in the *DM_content* value of the *DM_ClasModel*.

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_MiningModel*.

Description

1) The result of the invocation *DM_expClasModel()* is a CHARACTER LARGE OBJECT representing the classification model contained in SELF.

5.4.4 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(input 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 *input*.

2) The result of the invocation *DM_applyClasModel(input)* is determined as follows: :

   Case:

   a) If the set of fields of *input* is not compatible with the mining mapping of the classification model, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   b) Otherwise, it is the *DM_ClasResult* representing the result of an application of the classification model to *input*.

5.4.5 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(input 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 *input*.

2) The result of the invocation *DM_testClasModel(input)* is determined as follows: :

   Case:

   a) If the mining mapping of *input* is not compatible with the mining mapping of the classification model, then an exception is raised: *SQL/MM Data Mining exception – data and mapping incompatible*.

   b) Otherwise, it is the *DM_ClasTestResult* representing the result of testing the classification model using *input*.

5.4.6 DM_clasCostRate method

Purpose

Return the classification cost rate of the classification model computed during the training phase using the validation data specified in the DM_ClasTask value used to generate the model.

Definition

```
CREATE METHOD DM_clasCostRate()
  RETURNS DOUBLE
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

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

   a) If the validation data of SELF is the null value, then the cost rate on an implementation-dependent subset of the training data used for validation purposes.

   b) Otherwise, it is the cost rate of the classification model contained in SELF computed using the the validation data.

5.4.7 DM_*c*lasGetTask method

Purpose

Return the DM_ClasTask value used to create this model.

Definition

```
CREATE METHOD DM_clasGetTask()
   RETURNS DM_ClasTask
   FOR DM_ClasModel
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) If SELF does not contain the information about the DM_ClasTask value which was used to create the model, then the null value.

   b) Otherwise, it is the DM_ClasTask used to compute the model.

5.4.8 DM_*c*lasModelMapping method

Purpose

Return *the DM_MiningMapping* value specifying the set of fields for an application or test of this model.

Definition

```
CREATE METHOD DM_clasModelMapping ()
  RETURNS DM_clasModelMapping
  FOR DM_ClasModel
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_clasModelMapping ()* is a value of type DM_MiningMapping contained in SELF. This DM_MiningMapping contains the set of fields for an application or test of this model.

6 Data Mining Settings Types

6.1 DM_ClasSettings Type and Routines

6.1.1 DM_ClasSettings Type

Purpose

The *DM_ClasSettings* type is the description for the settings which 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

METHOD DM_clasSetCostRate
   (costRate DOUBLE PRECISION)
   RETURNS DM_ClasSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_clasUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_ClasSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_clasGetMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_clasSetTarget
   (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS DM_ClasSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_clasGetTarget()
   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*.

Description

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

   a) a method *DM_clasSetCostRate(DOUBLE PRECISION*),

   b) a method *DM_clasUseMapping(DM_MiningMapping)*,

   c) a method *DM_clasGetMapping()*,

   d) a method *DM_clasSetTarget(CHARACTER VARYING)*,

   e) a method *DM_clasGetTarget()*.

6.1.2 DM_clasSetCostRate Method

Purpose

Return a value of type *DM_ClasSettings* by specifying an error percentage for a training run. The *cost rate* value specifies the percentage of incorrect predictions that can be tolerated in a classification model on validation data.

Definition

```
CREATE METHOD DM_clasSetCostRate
   (costRate DOUBLE PRECISION)
   RETURNS DM_ClasSettings
   FOR DM_ClasSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DOUBLE PRECISION value *costRate*.

2) The result of an invocation of *DM_clasSetCostRate(DOUBLE PRECISION)* is determined as follows:

   Case:

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

   b) Otherwise, it is the *DM_ClasSettings* containing *costRate* as tolerated percentage of wrong predictions.

6.1.3 DM_clasUse*Mapping* Method

Purpose

Specify a DM_Mining*Mapping* for the DM_ClasSettings. The DM_Mining*Mapping* determines valid DM_MiningData which can be processed in a training run.

Definition

```
CREATE METHOD DM_clasUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_ClasSettings
   FOR DM_ClasSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DM_MiningMapping value *miningMapping*.

2) The result of an invocation of *DM_clasUseMapping(DM_MiningMapping)* is determined as follows:

   Case:

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

   b) If *miningMapping* contains a field with a name equal to the name of the target field but the field type is not categorical, then an exception is raised: *SQL/MM Data Mining exception - field not categorical*.

   c) Otherwise, it is a value of type *DM_ClasSetting*s containing *miningMapping* as the mining mapping of SELF.

6.1.4 DM_clasGetMapping Method

Purpose

Returns the mining mapping defined for a classification settings.

Definition

```
CREATE METHOD DM_clasGetMapping()
   RETURNS DM_MiningMapping
   FOR DM_ClasSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of an invocation of *DM_clasGetMapping()* is determined as follows:

   Case:

   a) If a mining mapping is defined for the classifications settings, then the value of that mining mapping.

   b) Otherwise, it is the null value.

6.1.5 DM_clasSetTarget Method

Purpose

Specify the target (class label) field for a *DM_ClasSetting*s.

Definition

```
CREATE METHOD DM_clasSetTarget
   (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 length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *targetField*.

2) The result of an invocation of *DM_clasSetTarget(CHARACTER VARYING)* is determined as follows:

   Case:

   a) If *targetField* is the null value, then a value of type *DM_ClasSettings* not containing a target field.

   b) If no mining mapping is contained in SELF, then an exception is raised: *SQL/MM Data Mining exception - no mining mapping defined.*

   c) If *targetField* is not equal to the name of any field contained in the mining mapping of SELF, then an exception is raised: *SQL/MM Data Mining exception – field not defined in mapping*.

   d) If a field with a name equal to *targetField* is contained in the mining mapping of SELF and the mining type of that field is defined but not categorical, then an exception is raised: *SQL/MM Data Mining exception - field not categorical*.

   e) Otherwise, it is the *DM_ClasSettings* containing the field named *targetField* as the target (class label) field.

6.1.6 DM_clasGetTarget Method

Purpose

Returns the target field name defined for a classification settings.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

1) The result of an invocation of *DM_clasTarget ()* is determined as follows:

   Case:

   a) The name of the field which is the target field in SELF.

   b) Otherwise, the null value.

6.2 DM_ClusSettings Type and Routines

6.2.1 DM_ClusSettings Type

Purpose

The *DM_ClusSettings* type is the description for the settings which are used to generate a segmentation model.

Definition

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

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

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

METHOD DM_clusUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_ClusSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_clusGetMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

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

Description

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

   a) a method DM_getMaxNOClus(),

   b) a method DM_setMaxNOClus(INTEGER),

   c) a method DM_clusUseMapping(DM_MiningMapping),

d) a method DM_clusGetMapping().

6.2.2 DM_setMaxNOClus Method

Purpose

Specifies the maximum number of clusters for a *DM_ClusSettings*.

Definition

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

Description

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

   a) an INTEGER value *maxNOClusters*.

2) The result of an invocation of *DM_ setMaxNOClus (INTEGER)* is determined as follows:

   Case:

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

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

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

6.2.3 DM_getMaxNOClus Method

Purpose

Return the maximum number of clusters specified in SELF.

Definition

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

Description

1) The result of an invocation of *DM_getMaxNOClus()* is determined as follows:

   a) If the maximum number of clusters was set in SELF in a previous invocation of DM_setMaxNOClusters, then the current value determining the maximum number of clusters.

   b) Otherwise, it is the null value.

6.2.4 DM_clusUseMapping Method

Purpose

Specify a DM_MiningMapping for the DM_ClusSettings. The DM_MiningMapping determines valid DM_MiningData values which can be processed in a training run.

Definition

```
CREATE METHOD DM_clusUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_ClusSettings
   FOR DM_ClusSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DM_MiningMapping value *miningMapping*.

2) The result of an invocation of *DM_clusUseMapping(DM_MiningMapping)* is determined as follows:

   Case:

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

   b) Otherwise, it is the *DM_ClusSetting*s containing *miningMapping* as the data mining mapping.

6.2.5 DM_clusGetMapping Method

Purpose

Returns the mining mapping defined for a clustering settings.

Definition

```
CREATE METHOD DM_clusGetMapping()
   RETURNS DM_MiningMapping
   FOR DM_ClusSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of an invocation of *DM_clusGetMapping()* is determined as follows:

   Case:

   a) If a mining mapping is defined for the clustering settings, then this value of type *DM_MiningMapping*.

   b) Otherwise, the null value.

6.3 DM_RegSettings Type and Routines

6.3.1 DM_RegSettings Type

Purpose

The *DM_RegSettings* type is the description of the settings which 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

METHOD DM_regSetRSquared
   (Rsquared DOUBLE PRECISION)
   RETURNS DM_RegSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_regUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_RegSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_regGetMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_regSetTarget
   (targetField CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS DM_RegSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_regGetTarget()
   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*.

Description

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

   a) a method DM_regSetRSquared(DOUBLE PRECISION),

   b) a method DM_regUseMapping(DM_MiningMapping),

   c) a method DM_regGetMapping(),

   d) a method DM_regSetTarget(CHARACTER VARYING),

   e) a method DM_regGetTarget().

6.3.2 DM_regSetRSquared Method

Purpose

Return a *DM_RegSettings* with the specified maximum tolerated squared Pearson correlation coefficient for a training run.

Definition

```
CREATE METHOD DM_regSetRSquared
   (rsquared DOUBLE PRECISION)
   RETURNS DM_RegSettings
   FOR DM_RegSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DOUBLE PRECISION value *rsquared*.

2) The result of an invocation of *DM_regSetRSquared(DOUBLE PRECISION)* is determined as follows:

   Case:

   a) If *rsquared* is negative, or greater than 1, then an exception is raised: *SQL/MM Data Mining exception – parameter out of range*.

   b) Otherwise, a value of type *DM_RegSettings* containing *rsquared* as the maximum tolerated r-squared value.

6.3.3 DM_regUseMapping Method

Purpose

Specify a DM_MiningMapping for the DM_RegSettings. The DM_MiningMapping determines valid DM_MiningData values which can be processed in a training run.

Definition

```
CREATE METHOD DM_regUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_RegSettings
   FOR DM_RegSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DM_MiningMapping value *miningMapping*.

2) The result of an invocation of *DM_regUseMapping(DM_MiningMapping)* is determined as follows:

   Case:

   a) If *miningMapping* is the NULL value, then SELF.

   b) Otherwise, it is the *DM_RegSetting*s containing *miningMapping* as the data mining mapping.

6.3.4 DM_regGetMapping Method

Purpose

Returns the mining mapping defined for a regression settings.

Definition

```
CREATE METHOD DM_regGetMapping()
  RETURNS DM_MiningMapping
  FOR DM_RegSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of an invocation of *DM_regGetMapping()* is determined as follows:

   Case:

   a) If a mining mapping is defined for the regression setting, then the value of that mapping

   b) Otherwise, the NULL value.

6.3.5 DM_regSetTarget Method

Purpose

Specify a target (predicted) field for the *DM_RegSettings*.

Definition

```
CREATE METHOD DM_regSetTarget
   (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 length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *targetField*.

2) The result of an invocation of *DM_regSetTarget (CHARACTER VARYING)* is determined as follows:

   Case:

   a) If *targetField* is the null value, then a value of type *DM_RegSettings* not containing a target field.

   b) If no mining mapping is contained in SELF, then an exception is raised: *SQL/MM Data Mining exception - no mining mapping defined.*

   c) If *targetField* is not equal to the name of any field contained in the mining scheam of SELF, then an exception is raised: *SQL/MM Data Mining exception - field not defined in mapping*.

   d) If a field with name equal to *targetField* is contained in the mining mapping of SELF and the mining type of that field is defined but not numeric, then an exception is raised: *SQL/MM Data Mining exception - field not numeric*.

   e) Otherwise, it is the DM_RegSettings containing the field named *targetField* as the target field.

6.3.6 DM_regGetTarget Method

Purpose

Returns the target field name defined for a regression settings.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

1) The result of an invocation of *DM_regGetTarget ()* is determined as follows:

   Case:

   a) If a target field has been defined and subsequently *DM_setRegTarget* has not been called with null input, then the name of this target field.

   b) Otherwise, the null value.

6.4 DM_RuleSettings Type and Routines

6.4.1 DM_RuleSettings Type

Purpose

The *DM_RuleSettings* type is the description of the settings which are used to generate an association rule model.

Definition

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

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_ruleUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_RuleSettings
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD DM_ruleGetMapping()
   RETURNS DM_MiningMapping
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

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

METHOD DM_ruleGetGroup()
   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*.

Description

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

   a) a method setMinSupport(DOUBLE PRECISION),

   b) a method getMinSupport(),

   c) a method DM_ruleUseMapping(DM_MiningMapping),

   d) a method DM_ruleGetMapping(),

   e) a method DM_ruleSetGroup(CHARACTER VARYING)

   f) a method DM_ruleGetGroup().

6.4.2 DM_setMinSupport Method

Pupose

Sets the minimum support in a value of a settings object for association rules.

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 of *DM_setMinSupport(DOUBLE PRECISION)* is determined as follows:

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

6.4.3 DM_getMinSupport Method

Purpose

Returns the minimum support which was specified as a parameter for the search for association rules.

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 *DM_setMinSupport(DOUBLE PRECISION)* was called before, then the value which was specified as *support* in the last call of *DM_setMinSupport(DOUBLE PRECISION)*.

   b) Otherwise, the implementation defined default minimum support.

6.4.4 DM_ruleUseMapping Method

Purpose

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

Definition

```
CREATE METHOD DM_ruleUseMapping
   (miningMapping DM_MiningMapping)
   RETURNS DM_RuleSettings
   FOR DM_RuleSettings
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a DM_MiningMapping value *miningMapping*.

2) The result of an invocation of *DM_ruleUseMapping(DM_MiningMapping)* is determined as follows:

   Case:

   a) If *miningMapping* is the NULL value, then SELF.

   b) Otherwise, it is the *DM_RuleSetting*s containing *miningMapping* as the data mining mapping.

6.4.5 DM_ruleGetMapping Method

Purpose

Returns the mining mapping defined for a rule settings.

Definition

```
CREATE METHOD DM_ruleGetMapping()
  RETURNS DM_MiningMapping
  FOR DM_RuleSettings
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of an invocation of *DM_ruleGetMapping()* is determined as follows:

   Case:

   a) If a mining mapping is defined for the rule setting, then the value of that mapping

   b) Otherwise, the NULL value.

6.4.6 DM_ruleSetGroup Method

Purpose

Specifies the field used to identify the group (e.g. purchase transaction) for mining association rules.

Definition

```
CREATE METHOD DM_ruleSetGroup()
   (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 length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *groupField*.

2) The result of an invocation of *DM_ruleSetGroup (CHARACTER VARYING)* is determined as follows:

   Case:

   a) If *groupField* is the null value, then a value of type *DM_RuleSettings* not containing a group field.

   b) If no mining mapping is contained in SELF, then an exception is raised: *SQL/MM Data Mining exception - no mining mapping defined.*

   c) If no field with name *groupField* is contained in the mining mapping of SELF, then an exception is raised: *SQL/MM Data Mining exception - field not defined in mapping*.

   d) If a field with name *groupField* is contained in the mining mapping of SELF and the mining type of that field is defined but not categorical, then an exception is raised: *SQL/MM Data Mining exception - field not categorical*.

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

6.4.7 DM_ruleGetGroup Method

Purpose

Returns the group field name defined for a rule settings.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

1) The result of an invocation of *DM_ruleGetGroup ()* is determined as follows:

   Case:

   a) If a group field has been defined and subsequently *DM_ruleSetGroup* has not been called with null input, then the name of this group field.

   b) Otherwise, it is the null value.

6.5 DM_ClasTask Type and Routines

6.5.1 DM_ClasTask Type

Purpose

The type DM_ClasTask is an abstraction of all the information which 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_ClasTask
   AS (
      DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
   )
   INSTANTIABLE
   NOT FINAL

STATIC METHOD DM_defClasTask
   (train_data DM_MiningData,
    validation_data DM_MiningData,
    settings DM_ClasSettings)
    RETURNS DM_ClasTask
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT,

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

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

METHOD DM_clasGetSettings()
   RETURNS DM_ClasSettings
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_buildClasModel()
   RETURNS DM_ClasModel
   LANGUAGE SQL
   NON 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_ClasTask*

Description

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

   a) a method *DM_defClasTask(DM_MiningData, DM_MiningData, DM_ClasSettings)*,

   b) a method *DM_clasTrainData()*,

   c) a method *DM_clasValiData()*,

   d) a method *DM_clasSettings()*,

   e) a method *DM_buildClasModel()*.

6.5.2 DM_defClasTask Method

Purpose

Return a specified value of type DM_ClasTask.

Definition

```
CREATE STATIC METHOD DM_defClasTask
   (train_data DM_MiningData,
    validation_data DM_MiningData,
    settings DM_ClasSettings)
   RETURNS DM_ClasTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a *DM_MiningData* value *train_data*,

   b) a *DM_MiningData* value *validation_data*,

   c) a *DM_ClasSettings* value *settings*.

2) The result of the invocation of *DM_defClasModel(DM_MiningData, DM_MiningData, DM_ClasSettings)* is determined as follows:

   Case:

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

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

   c) If *train_data* or *validation_data* is incompatible with the mining mapping in *settings*, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   d) If *validation_data* is the null value, then a value of type *DM_ClasTask* determined by the given parameters where no validation data are defined. An implementation-dependent subset of the training data is used for internal validation in this case.

   e) Otherwise, it is the *DM_ClasTask* determined by the given parameters.

6.5.3 DM_clasTrainData Method

Purpose

Return the value of type *DM_MiningData* representing the training data for the classification task.

Definition

```
CREATE METHOD DM_clasTrainData()
   RETURNS DM_MiningData
   FOR DM_ClasTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

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

   Case:

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

   b) Otherwise, it is the *DM_MiningData* representing the data to be used for training.

6.5.4 DM_clasValiData Method

Purpose

Return the value of type *DM_MiningData* representing the validation data for the classification task.

Definition

```
CREATE METHOD DM_clasValiData()
   RETURNS DM_MiningData
   FOR DM_ClasTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_clasValiData()* 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* representing the data to be used for validation.

6.5.5 DM_*c*lasGetSettings Method

Purpose

Return the value of type *DM_ClasSettings* representing the settings of the classification task.

Definition

```
CREATE METHOD DM_clasGetSettings()
   RETURNS DM_ClasSettings
   FOR DM_ClasTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_clasGetSettings()* 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* representing the settings to be used for the training phase.

6.5.6 DM_buildClasModel Method

Purpose

Return a specified value of type *DM_ClasModel*.

Definition

```
CREATE STATIC METHOD DM_buildClasModel()
  RETURNS DM_ClasModel
  FOR DM_ClasTask
  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 a value of type *DM_ClasModel* .

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

6.6 DM_RuleTask type and Routines

6.6.1 DM_RuleTask Type

Purpose

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

Definition

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

STATIC METHOD DM_defRuleTask
   (train_data DM_MiningData,
    settings DM_RuleSettings)
    RETURNS DM_RuleTask
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

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


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

METHOD DM_buildRuleModel()
   RETURNS DM_RuleModel
   LANGUAGE SQL
   NON DETERMINISTIC
   CONTAINS SQL
```

Definitional Rules

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

Description

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

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

   b) a method *DM_ruleTrainData()*,

   c) a method *DM_getRuleSettings()*,

   d) a method DM_buildRuleModel().

6.6.2 DM_defRuleTask Method

Purpose

Given a DM_MiningData, train_data, and a DM_RuleSettings, settings, return the corresponding DM_RuleTask.

Definition

```
CREATE STATIC METHOD DM_defRuleTask
   (train_data DM_MiningData,
    settings DM_RuleSettings)
   RETURNS DM_RuleTask
   FOR DM_RuleTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a *DM_MiningData* value *train_data*,

   b) a *DM_RuleSettings* value *settings*.

2) The result of the invocation *DM_defRuleTaskl(DM_MiningData, DM_RuleSettings)* returns an implementation-dependent value of DM_RuleTask determined by train_data and settings.

6.6.3 DM_ruleTrainData Method

Purpose

Return the *DM_MiningData* representing the training data for the given DM_RuleTask.

Definition

```
CREATE METHOD DM_ruleTrainData()
   RETURNS DM_MiningData
   FOR DM_RuleTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

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

   Case:

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

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

6.6.4 DM_ruleSettings Method

Purpose

Return the value of type *DM_RuleSettings* representing the settings of the association rules search.

Definition

```
CREATE METHOD DM_ruleSettings()
   RETURNS DM_RuleSettings
   FOR DM_RuleTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_ruleSettings()* 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* representing the settings to be used for the training phase.

6.6.5 DM_buildRuleModel Method

Purpose

Return a specified value of type *DM_RuleModel*.

Definition

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

Description

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

   Case:

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

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

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

6.7 DM_RegTask type and Routines

6.7.1 DM_RegTask Type

Purpose

The type DM_RegTask 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_RegTask
   AS (
      DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
   )
   INSTANTIABLE
   NOT FINAL

STATIC METHOD DM_defRegTask
   (train_data DM_MiningData,
    validation_data DM_MiningData,
    settings DM_RegSettings)
    RETURNS DM_RegTask
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT,

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

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

METHOD DM_regGetSettings()
   RETURNS DM_RegSettings
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD DM_buildRegModel()
   RETURNS DM_RegressionModel
   LANGUAGE SQL
   NON 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_RegTask*

Description

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

   a) a method *DM_defRegTask(DM_MiningData, DM_MiningData, DM_RegSettings)*,

   b) a method *DM_regTrainData()*,

   c) a method *DM_regValiData()*,

   d) a method *DM_regGetSettings()*,

   e) a method *DM_buildRegModel()*.

6.7.2 DM_defRegTask Method

Purpose

Return the DM_RegTask determined by the given input parameters.

Definition

```
CREATE STATIC METHOD DM_defRegTask
   (train_data DM_MiningData,
    validation_data DM_MiningData,
    settings DM_RegSettings)
   RETURNS DM_RegTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a *DM_MiningData* value *train_data*,

   b) a *DM_MiningData* value *validation_data*,

   c) a *DM_RegSettings* value *settings*.

2) The result of the invocation of *DM_defRegTask(DM_MiningData, DM_MiningData, DM_RegSettings)* is determined as follows:

   Case:

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

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

   c) If *train_data* or *validation_data* are incompatible (see [miningData]) with the mining mapping in *settings*, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   d) If *validation_data* is the null value, then a value of type DM_regTask determined by the given parameters where no validation data are defined. It is implementation-dependent which subset of the training data is used for internal validation in this case.

   e) Otherwise, a value of type *DM_RegTask* determined by the given parameters.

6.7.3 DM_regTrainData Method

Purpose

Return the *DM_MiningData* representing the training data of the *DM_RegTask*.

Definition

```
CREATE METHOD DM_regTrainData()
   RETURNS DM_MiningData
   FOR DM_RegTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

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

   Case:

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

   b) Otherwise, a value of type *DM_MiningData* representing the data to be used for training.

6.7.4 DM_regValiData Method

Purpose

Return the *DM_MiningData* representing the validation data of the *DM_RegTask*.

Definition

```
CREATE METHOD DM_regValiData()
   RETURNS DM_MiningData
   FOR DM_RegTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_regValiData()* 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.

6.7.5 DM_regGetSettings Method

Purpose

Return the *DM_RegSettings* representing the settings of *DM_RegTask*.

Definition

```
CREATE METHOD DM_regGetSettings()
   RETURNS DM_RegSettings
   FOR DM_RegTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_regGetSettings()* 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.

6.7.6 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_RegTask
  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 a 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 is raised: SQL/MM Data Mining exception – model computation failed.

6.8 DM_ClusTask type and Routines

6.8.1 DM_ClusTask Type

Purpose

The type DM_ClusTask 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_ClusTask
   AS (
      DM_content CHARACTER LARGE OBJECT(DM_MaxContentLength)
   )
   INSTANTIABLE
   NOT FINAL

STATIC METHOD DM_defClusTask
   (input_data DM_MiningData,
    settings DM_ClusSettings)
    RETURNS DM_ClusTask
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT,

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

METHOD DM_clusGetSettings()
   RETURNS DM_ClusSettings
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL,

METHOD DM_buildClusModel()
   RETURNS DM_ClusteringModel
   LANGUAGE SQL
   NON DETERMINISTIC
   CONTAINS SQL.
```

Definitional Rules

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

Description

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

   a) a method *DM_defClusTask(DM_MiningData, DM_ClusSettings)*,

   b) a method *DM_clusTrainData()*,

   c) a method *DM_clusGetSettings()*,

   d) a method DM_buildClusModel().

6.8.2 DM_defClusTask Method

Purpose

Return the DM_ClusTask determined by both the given DM_MiningData and the given DM_ClusSettings.

Definition

```
CREATE STATIC METHOD DM_defClusTask
   (input_data DM_MiningData,
    settings DM_ClusSettings)
   RETURNS DM_ClusTask
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

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

   a) a *DM_MiningData* value *input_data*,

   b) a *DM_ClusSettings* value *settings*.

2) The result of the invocation of *DM_defClusModel(DM_MiningData, DM_ClusSettings)* is determined as follows:

   Case:

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

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

   c) If *input_data* is incompatible with the mining *mapping* in *settings*, then an exception is raised: *SQL/MM Data Mining exception – data and mapping not compatible*.

   d) Otherwise, the *DM_ClusTask* determined by the given parameters.

6.8.3 DM_clusTrainData Method

Purpose

Return the *DM_MiningData* representing the training data for the clustering task.

Definition

```
CREATE METHOD DM_clusTrainData()
   RETURNS DM_MiningData
   FOR DM_ClusTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation of *DM_clusTrainData()* 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* representing the input data to be used for training.

6.8.4 DM_clusGetSettings Method

Purpose

Return the *DM_ClusGetSettings* representing the settings of the clustering task.

Definition

```
CREATE METHOD DM_clusGetSettings()
   RETURNS DM_ClusSettings
   FOR DM_ClusTask
   BEGIN
   --
   --!! See Description
   --
   END
```

Description

1) The result of the invocation *DM_clusGetSettings()* 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* representing the settings to be used for training.

6.8.5 DM_buildClusModel Method

Purpose

Return the *DM_ClusModel* representing the result of a clustering training run given SELF.

Definition

```
CREATE METHOD DM_buildClusModel()
   RETURNS DM_ClusteringModel
   FOR DM_ClusTask
   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 a 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 is raised: SQL/MM Data Mining exception – model computation failed.

Blank page

7 Data Mining Application Result Types

7.1 DM_ClusResult Type and Routines

7.1.1 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
   RETURNS NULL ON NULL INPUT,

METHOD DM_getQuality()
   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_ClusResult*.

Description

1) The DM_ClusResult type provides for public use:

   a) a method DM_getClusterID(),

   b) a method DM_getQuality().

7.1.2 DM_getClusterID Method

Purpose

Returns the cluster identification number contained in the DM_ClusResult

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.

7.1.3 DM_getQuality Method

Purpose

Returns the quality value contained in the DM_ClusResult.

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.

7.2 DM_ClasResult Type and Routines

7.2.1 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_MaxClassLabelLength)
   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_MaxClassLabelLength* is the implementation-defined maximum length for a class label.

Description

1) The DM_ClasResult type provides for public use:

   a) a method DM_getPredClass(),

   b) a method DM_getConfidence().

7.2.2 DM_getPredClass Method

Purpose

Returns the predicted class contained in the DM_ClasResult

Definition

```
CREATE METHOD DM_getPredClass()
   RETURNS CHARACTER VARYING(DM_MaxClassLabelLength)
   FOR DM_ClasResult
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxClassLabelLength* is the implementation-defined maximum length for a class label.

Description

1) The result of the invocation *DM_getPredClass()* is a CHARACTER VARYING representing the class label contained in SELF.

7.2.3 DM_getConfidence Method

Purpose

Returns the confidence value contained in the DM_ClasResult.

Definition

```
CREATE METHOD DM_getConfidence()
  RETURNS DOUBLE PRECISION
  FOR DM_ClasResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

1) The result of the invocation *DM_getPredClass()* is the DOUBLE PRECISION value representing the confidence value computed for the prediction of the class label contained in SELF.

7.3 DM_RegResult Type and Routines

7.3.1 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().

7.3.2 DM_getPredValue Method

Purpose

Returns the predicted value contained in the DM_RegResult

Definition

```
CREATE METHOD DM_getPredValue()
  RETURNS DOUBLE PRECISION
  FOR DM_RegResult
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

The result of the invocation *DM_getPredValue()* is a DOUBLE PRECISION value representing the predicted value contained in SELF.

8 Data Mining Test Result Types

8.1 DM_ClasTestResult Type and Routines

8.1.1 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
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

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

Description

1) The DM_ClasTestResult type provides for public use:

   a) a method DM_getClasError().

8.1.2 DM_getClasError Method

Purpose

Returns the classification error value contained in the DM_ClasTestResult.

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 number of false classifications computed during a test of a classification model.

8.2 DM_RegTestResult Type and Routines

8.2.1 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
```

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().

8.2.2 DM_getPredError Method

Purpose

Returns the prediction error value contained in the DM_RegTestResult.

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 Data Mining Data Types

9.1 DM_MiningMapping Type and Routines

9.1.1 DM_MiningMapping Type

Purpose

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

Definition

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

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

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

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

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

METHOD DM_setFieldType
   (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength)
    miningType  SMALLINT)
   RETURNS DM_MiningMapping
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,
```

```
METHOD DM_getFieldType
   (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS SMALLINT
   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_MiningMapping*.

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum length of a mining field name.

Description

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

   a) a method DM_addMappingField(CHARACTER VARYING),

   b) a method DM_remMappingField(CHARACTER VARYING),

   c) a method DM_getNOFields(),

   d) a method DM_getFieldName(INTEGER),

   e) a method DM_setFieldType(CHARACTER VARYING, SMALLINT),

   f) a method DM_getFieldType(CHARACTER VARYING).

2) The following values for mining types are defined:

Table 1 - Values for mining types

DM_Categorical	0
DM_Numeric	1

9.1.2 DM_addMappingField Method

Purpose

Add a field with the specified name to the *DM_MiningMapping*.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *fieldName*.

2) The result of an invocation of *DM_addMappingField(CHARACTER VARYING)* 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 is raised: *SQL/MM Data Mining exception - field already defined*.

   c) Otherwise, it is a value of type *DM_MiningMapping* 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.

9.1.3 DM_rem*Mapping*Field Method

Purpose

Remove the mining field with the specified name from the *DM_MiningMapping*.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *fieldName*.

2) The result of an invocation of *DM_remMappingField(CHARACTER VARYING)* 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 is raised: *SQL/MM Data Mining exception - field not defined in mapping*.

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

9.1.4 DM_getNOFields Method

Purpose

Returns the number of fields of the *DM_MiningMapping*.

Definition

```
CREATE METHOD DM_getNOFields()
  RETURNS INTEGER
  FOR DM_MiningMapping
  BEGIN
  --
  -- !! See Description
  --
  END
```

Description

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

9.1.5 DM_getFieldName Method

Purpose

Returns the name of the data mining field of the *DM_MiningMapping* at the specified position.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

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

   a) an INTEGER value *position*.

2) The result of an invocation of *DM_getFieldName(INTEGER)* is determined as follows:

   Case:

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

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

9.1.6 DM_setFieldType Method

Purpose

Sets the data mining type of the field with the specified name contained in the *DM_MiningMapping*.

Definition

```
CREATE METHOD DM_setFieldType
   (fieldName CHARACTER VARYING(DM_MaxFieldAliasLength)
    miningType SMALLINT)
   RETURNS DM_MiningMapping
   FOR DM_MiningMapping
   BEGIN
   --
   -- !! See Description
   --
   END
```

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *fieldName*,

   b) a SMALLINT value *miningType*.

2) The result of an invocation of *DM_setFieldType(CHARACTER VARYING, SMALLINT)* 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 is raised: *SQL/MM Data Mining exception - field not defined in mapping*.

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

   d) Otherwise, the mining 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 1 for mining type values).

9.1.7 DM_getFieldType Method

Purpose

Returns the type of the data mining field with the specified name of the *DM_MiningMapping*.

Definition

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

Definitional Rules

1) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

Description

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

   a) a CHARACTER VARYING value *fieldName*.

2) The result of an invocation of *DM_getFieldType(CHARACTER VARYING)* is determined as follows:

   Case:

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

   b) If no mining type has been set for the field named *fieldName*, then the null value.

   c) Otherwise, it is the data mining type of the field with name *fieldName* contained in the set of fields in SELF. For defined values for the data mining type see table 1.

9.2 DM_MiningData Type and Routines

9.2.1 DM_MiningData Type

Purpose

DM_MiningData is an abstraction of the input data for data mining training and test functions.

Definition

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

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

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

METHOD DM_genMiningMap()
   RETURNS DM_MiningMapping
   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_MaxDataSourceNameLength* is the implementation-defined maximum length for the name of a data source.

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

4) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for the alias name of a field.

Description

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

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

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

c) a method *DM_genMiningMap()*.

9.2.2 DM_defMiningData Method

Purpose

Return *DM_MiningData* corresponding with the data source determined by the given name.

Definition

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

Definitional Rules

1) *DM_MaxDataSourceNameLength* is the implementation-defined maximum length for the name of a data source.

Description

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

   a) a CHARACTER VARYING value *dataSourceName*.

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

   a) If *dataSourceName* identifies a valid data source, then a value of *DM_MiningData* determined by the identified data source. Valid data sources include base tables and views. Each column of the data source of type character, character varying, datetime, or numeric is mapped onto a field of 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 data source.

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

9.2.3 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),
    alias CHARACTER VARYING(DM_MaxFieldAliasLength))
   RETURNS DM_MiningData
   FOR DM_MiningData
   BEGIN
    --
    -- !! See Description
    --
   END
```

Definitional Rules

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

2) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for the alias name of a field.

Description

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

   a) a CHARACTER VARYING value *dataField*,

   b) a CHARACTER VARYING value *alias*.

2) The result of the invocation of DM_defFldAlias is determined as follows:

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

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

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

   d) If *dataField* is a valid name of a field of the *DM_MiningData* value and *alias* is the null value, then the alias of that field is set to the value of *dataField.*

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

9.2.4 DM_genMiningMap Method

Purpose

Generates value of type *DM_MiningMapping*.

Definition

```
CREATE METHOD DM_genMiningMap()
   RETURNS DM_MiningMapping
   BEGIN
   --
   -- !! See Description
   --
   END
```

Description

1) The result of an invocation of *DM_genMiningMap()* is a value of type *DM_MiningMapping* 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_MiningMapping* value. Each field in the *DM_MiningMapping* has an associated name which is identical to the alias name of the corresponding field in SELF.

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

   Case:

   a) If the field in *DM_MiningData* is of numeric type, then the mining type is set to DM_Numeric (see table 1 in subclause 9.1.1).

   b) Otherwise, the mining type is set to DM_Categorical (see table 1 in subclause 9.1.1).

9.3 DM_ApplicationData Type and Routines

9.3.1 DM_ApplicationData Type

Purpose

The *DM_ApplicationData* type is the description of the data which 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_impApplData
   (input 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

Editor's Note 6-1
To be done.

9.3.2 DM_impApplData Method

Editor's Note 6-2
To be done.

10 Conformance

10.1 Requirements for conformance

A conforming implementation shall support one of the mandatory groups of public user-defined types and routines given by this part of ISO/IEC 13249 and may in addition support some or all of the optional user-defined types and routines.

10.2 Claims of conformance

Claims of conformance to this part of ISO/IEC 13249 shall state:

Editor's Note 6-3

Claims of conformance are to be supplied.
In particular, it should be stated that an implementation does not necessarily need to implement this part of ISO/IEC 13049 with user-defined structured types.

11 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 2.

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 22.1, "SQLSTATE" in ISO/IEC 9075-2) and the third character of the SQLSTATE is 'H'.

Table 2 — SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
X	SQL/MM Data Mining exception	‡	alias already in use	F01
X	SQL/MM Data Mining exception	‡	data and mapping 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 mapping	F05
X	SQL/MM Data Mining exception	‡	field not numeric	F06
X	SQL/MM Data Mining exception	‡	invalid data source 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 mining mapping defined	F13
X	SQL/MM Data Mining exception	‡	null error weight	F14
X	SQL/MM Data Mining exception	‡	null parameter	F15
X	SQL/MM Data Mining exception	‡	null settings	F16
X	SQL/MM Data Mining exception	‡	null training data	F17
X	SQL/MM Data Mining exception	‡	parameter out of range	F18

‡ 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 22.1, "SQLSTATE" in ISO/IEC 9075-2).

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 22.1, "SQLSTATE" in ISO/IEC 9075-2).

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 length of the *DM_content* attribute of the introduced user-defined types.

2) *DM_MaxClassLabelLength* is the implementation-defined maximum length for a class label.

3) *DM_MaxDataSourceNameLength* is the implementation-defined maximum length for the name of a data source.

4) *DM_MaxFieldNameLength* is the implementation-defined maximum length for the name of a field.

5) *DM_MaxFieldAliasLength* is the implementation-defined maximum length for strings representing a field name within a mining mapping.

6) *DM_MaxFieldValueLength* is the implementation-defined maximum length for the parameters *correctValue* and *predictedValue* of method *DM_addErrorWeight*.

Blank page

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.

Editor's Note 6-4

To be done.

Blank page

Bibliography

Blank page

Index

Index entries 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.

Index page numbers appearing in **boldface** indicates a page or range of pages where the attribute, routine, or user-defined type is specified. Index page numbers appearing in *italics* indicates a page where the word, phrase, attribute, routine, or user-defined type is defined. Index page numbers appearing in roman type indicate a page where the word, phrase, attribute, routine, or type was used.

Editor's Note 6-5

To be done.