
**Information technology — Database
languages — SQL multimedia and
application packages —**

**Part 6:
Data mining**

*Technologies de l'information — Langages de bases de données —
Multimédia SQL et paquetages d'application —*

Partie 6: Exploration de données



Reference number
ISO/IEC 13249-6:2006(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2006

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2006

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents	Page
Foreword	ix
Introduction	x
1 Scope	1
2 Normative references	2
2.1 International standards	2
2.2 Publicly-available specifications	2
3 Terms, definitions, notations and conventions	3
3.1 Terms and definitions.....	3
3.2 Notations.....	10
3.2.1 Notations provided in ISO/IEC 13249-1.....	10
3.2.2 Notations provided in this part of ISO/IEC 13249.....	10
3.3 Conventions	11
4 Concepts	12
4.1 Data mining techniques.....	12
4.2 Computational phases.....	12
4.2.1 Training phase.....	12
4.2.2 Model introspection phase.....	13
4.2.3 Application phase	15
4.2.4 Testing phase	16
4.3 Mapping on user-defined types.....	17
4.3.1 Target application	17
4.3.2 The relationships of the user-defined types	17
4.3.3 User-defined types that are not related to mining techniques.....	19
4.3.4 User-defined types that are related to mining techniques	19
5 Data Mining Data Types	22
5.1 DM_LogicalDataSpec Type and Routines	22
5.1.1 DM_LogicalDataSpec Type	22
5.1.2 DM_addDataSpecFld Method	24
5.1.3 DM_remDataSpecFld Method	25
5.1.4 DM_getNumFields Method	26
5.1.5 DM_getFldName Method	27
5.1.6 DM_setFldType Method.....	28
5.1.7 DM_getFldType Method.....	30
5.1.8 DM_isCompatible Method	31
5.2 DM_MiningData Type and Routines.....	32
5.2.1 DM_MiningData Type.....	32
5.2.2 DM_defMiningData Method	34
5.2.3 DM_setFldAlias Method.....	35
5.2.4 DM_genDataSpec Method	36
5.3 DM_ApplicationData Type and Routines.....	37
5.3.1 DM_ApplicationData Type.....	37
5.3.2 DM_genDataSpec Method	38
5.3.3 DM_impApplData Method.....	39
6 Association Rules	40

6.1	DM_RuleSettings Type and Routines	40
6.1.1	DM_RuleSettings Type	40
6.1.2	DM_impRuleSettings Method	43
6.1.3	DM_expRuleSettings Method	44
6.1.4	DM_setRuleFilter Method	45
6.1.5	DM_getRuleFilter Method	46
6.1.6	DM_setMinSupport Method	47
6.1.7	DM_getMinSupport Method	48
6.1.8	DM_setMinConf Method	49
6.1.9	DM_getMinConf Method	50
6.1.10	DM_setMaxLength Method	51
6.1.11	DM_getMaxLength Method	52
6.1.12	DM_useRuleDataSpec Method	53
6.1.13	DM_getRuleDataSpec Method	54
6.1.14	DM_setGroup Method	55
6.1.15	DM_getGroup Method	56
6.1.16	DM_setSequence Method	57
6.1.17	DM_getSequence Method	58
6.2	DM_RuleBldTask type and Routines	59
6.2.1	DM_RuleBldTask Type	59
6.2.2	DM_defRuleBldTask Method	60
6.2.3	DM_getRuleTrnData Method	61
6.2.4	DM_getRuleSettings Method	62
6.2.5	DM_buildRuleModel Method	63
6.3	DM_RuleModel Type and Routines	64
6.3.1	DM_RuleModel Type	64
6.3.2	DM_impRuleModel Method	67
6.3.3	DM_getRuleMdlType Method	68
6.3.4	DM_expRuleModel Method	69
6.3.5	DM_applyRuleModel Method	70
6.3.6	DM_getNumItemSets Method	71
6.3.7	DM_getNumSequences Method	72
6.3.8	DM_getNumRules Method	73
6.3.9	DM_filterRules Method	74
6.3.10	DM_getNumTransacts Method	75
6.3.11	DM_getItemSets Method	76
6.3.12	DM_getRules Method	77
6.3.13	DM_getRuleBodies Method	78
6.3.14	DM_getSequences Method	79
6.3.15	DM_getSeqDetails Method	80
6.3.16	DM_getSeqRules Method	81
6.3.17	DM_getRuleBldTask method	83
6.4	DM_RuleFilter Type and Routines	84
6.4.1	DM_RuleFilter Type	84
6.4.2	DM_addRangeConstr Method	86
6.4.3	DM_remRangeConstr Method	88
6.4.4	DM_getRangeConstrs Method	89
6.4.5	DM_setMaxNumRules Method	90
6.4.6	DM_getMaxNumRules Method	91
6.4.7	DM_addCountConstr Method	92
6.4.8	DM_remCountConstr Method	93
6.4.9	DM_getCountConstrs Method	94
6.4.10	DM_setItemConstr Method	95
6.4.11	DM_getItemConstrs Method	97
6.5	DM_RuleApplTask Type and Routines	98
6.5.1	DM_RuleApplTask Type	98
6.5.2	DM_defRuleApplTask Method	100
6.5.3	DM_getRuleMdl Method	102

6.5.4	DM_getRuleAppIData Method	103
6.5.5	DM_getRuleOutpData Method	104
6.5.6	DM_getRuleGroupFld method	105
6.5.7	DM_applyRuleMdl method	106
7	Clustering	107
7.1	DM_ClusSettings Type and Routines	107
7.1.1	DM_ClusSettings Type	107
7.1.2	DM_impClusSettings Method	109
7.1.3	DM_expClusSettings Method	110
7.1.4	DM_setMaxNumClus Method	111
7.1.5	DM_getMaxNumClus Method	112
7.1.6	DM_setFldWeight Method	113
7.1.7	DM_getFldWeight Method	114
7.1.8	DM_setFldSimScale Method	115
7.1.9	DM_getFldSimScale Method	116
7.1.10	DM_useClusDataSpec Method	117
7.1.11	DM_getClusDataSpec Method	118
7.2	DM_ClusBldTask type and Routines	119
7.2.1	DM_ClusBldTask Type	119
7.2.2	DM_defClusBldTask Method	120
7.2.3	DM_getClusTrnData Method	121
7.2.4	DM_getClusSettings Method	122
7.2.5	DM_buildClusModel Method	123
7.3	DM_ClusteringModel Type and Routines	124
7.3.1	DM_ClusteringModel Type	124
7.3.2	DM_impClusModel Method	126
7.3.3	DM_expClusModel Method	127
7.3.4	DM_getNumClusters Method	128
7.3.5	DM_getClusters Method	129
7.3.6	DM_getMdlQuality Method	130
7.3.7	DM_applyClusModel Method	131
7.3.8	DM_getClusTask method	132
7.3.9	DM_getClusMdlSpec method	133
7.3.10	DM_getFields	134
7.4	DM_ClusAppITask Type and Routines	135
7.4.1	DM_ClusAppITask Type	135
7.4.2	DM_defClusAppITask Method	137
7.4.3	DM_getClusMdl Method	138
7.4.4	DM_getClusAppIData Method	139
7.4.5	DM_getClusOutpData Method	140
7.4.6	DM_getClusResFld method	141
7.4.7	DM_applyClusMdl method	142
7.5	DM_ClusResult Type and Routines	143
7.5.1	DM_ClusResult Type	143
7.5.2	DM_getClusterID Method	144
7.5.3	DM_getQuality Method	145
8	Classification	146
8.1	DM_ClasSettings Type and Routines	146
8.1.1	DM_ClasSettings Type	146
8.1.2	DM_impClasSettings Method	149
8.1.3	DM_expClasSettings Method	150
8.1.4	DM_setClasCostRate Method	151
8.1.5	DM_getClasCostRate Method	152
8.1.6	DM_setClasCost Method	153
8.1.7	DM_getClasCost Method	154
8.1.8	DM_useClasDataSpec Method	155

8.1.9	DM_getClasDataSpec Method	156
8.1.10	DM_setClasTarget Method.....	157
8.1.11	DM_getClasTarget Method.....	158
8.2	DM_ClasBldTask Type and Routines.....	159
8.2.1	DM_ClasBldTask Type	159
8.2.2	DM_defClasBldTask Method.....	161
8.2.3	DM_getClasTrnData Method	162
8.2.4	DM_getClasValData Method	163
8.2.5	DM_getClasSettings Method	164
8.2.6	DM_buildClasModel Method	165
8.3	DM_ClasModel Type and Routines	166
8.3.1	DM_ClasModel Type	166
8.3.2	DM_impClasModel Method	169
8.3.3	DM_expClasModel Method	170
8.3.4	DM_applyClasModel Method	171
8.3.5	DM_testClasModel Method	172
8.3.6	DM_getClasCostRate Method	173
8.3.7	DM_isValidated Method.....	174
8.3.8	DM_getRankQuality method	175
8.3.9	DM_getPredAccuracy Method	176
8.3.10	DM_getReliability Method	177
8.3.11	DM_getMdlQuality Method.....	178
8.3.12	DM_getGainsChart method.....	179
8.3.13	DM_getClasTask method	180
8.3.14	DM_getClasMdlSpec method.....	181
8.3.15	DM_getFields.....	182
8.3.16	DM_getClasTarget method	183
8.4	DM_ClasTestTask Type and Routines	184
8.4.1	DM_ClasTestTask Type	184
8.4.2	DM_defClasTestTask Method	185
8.4.3	DM_getClasTestData Method	186
8.4.4	DM_getClasTestMdl Method	187
8.4.5	DM_testClasMdl Method	188
8.5	DM_ClasTestResult Type and Routines	189
8.5.1	DM_ClasTestResult Type	189
8.5.2	DM_getClasError Method.....	191
8.5.3	DM_getRankQuality Method	192
8.5.4	DM_getPredAccuracy Method	193
8.5.5	DM_getReliability Method	194
8.5.6	DM_getMdlQuality Method.....	195
8.5.7	DM_getGainsChart Method.....	196
8.5.8	DM_getNumClasses Method.....	197
8.5.9	DM_getClass Method.....	198
8.5.10	DM_getClasPreds Method.....	199
8.6	DM_ClasApplTask Type and Routines	200
8.6.1	DM_ClasApplTask Type	200
8.6.2	DM_defClasApplTask Method	202
8.6.3	DM_getClasMdl Method	203
8.6.4	DM_getClasApplData Method.....	204
8.6.5	DM_getClasOutpData Method	205
8.6.6	DM_getClasResFld method	206
8.6.7	DM_applyClasMdl method	207
8.7	DM_ClasResult Type and Routines.....	208
8.7.1	DM_ClasResult Type.....	208
8.7.2	DM_getPredClass Method.....	209
8.7.3	DM_getConfidence Method.....	210

9	Regression.....	211
9.1	DM_RegSettings Type and Routines	211
9.1.1	DM_RegSettings Type	211
9.1.2	DM_impRegSettings Method	213
9.1.3	DM_expRegSettings Method	214
9.1.4	DM_setRegRSquared Method.....	215
9.1.5	DM_getRegRSquared Method	216
9.1.6	DM_useRegDataSpec Method	217
9.1.7	DM_getRegDataSpec Method	218
9.1.8	DM_setRegTarget Method.....	219
9.1.9	DM_getRegTarget Method	220
9.2	DM_RegBldTask type and Routines	221
9.2.1	DM_RegBldTask Type	221
9.2.2	DM_defRegBldTask Method	223
9.2.3	DM_getRegTrnData Method	224
9.2.4	DM_getRegValData Method	225
9.2.5	DM_getRegSettings Method	226
9.2.6	DM_buildRegModel Method.....	227
9.3	DM_RegressionModel Type and Routines	228
9.3.1	DM_RegressionModel Type	228
9.3.2	DM_impRegModel Method	231
9.3.3	DM_expRegModel Method	232
9.3.4	DM_applyRegModel Method	233
9.3.5	DM_testRegModel Method	234
9.3.6	DM_getRegRSquared method	235
9.3.7	DM_isValidated Method.....	236
9.3.8	DM_getRankQuality Method	237
9.3.9	DM_getPredAccuracy Method	238
9.3.10	DM_getReliability Method	239
9.3.11	DM_getMdlQuality Method	240
9.3.12	DM_getRMSE Method.....	241
9.3.13	DM_getGainsChart method.....	242
9.3.14	DM_getRegTask method	243
9.3.15	DM_getRegMdlSpec method	244
9.3.16	DM_getFields method.....	245
9.3.17	DM_getRegTarget method	246
9.4	DM_RegTestTask Type and Routines.....	247
9.4.1	DM_RegTestTask Type.....	247
9.4.2	DM_defRegTestTask Method	248
9.4.3	DM_getRegTestData Method	249
9.4.4	DM_getRegTestMdl Method	250
9.4.5	DM_testRegMdl Method	251
9.5	DM_RegTestResult Type and Routines	252
9.5.1	DM_RegTestResult Type	252
9.5.2	DM_getPredError Method.....	254
9.5.3	DM_getRankQuality Method	255
9.5.4	DM_getPredAccuracy Method	256
9.5.5	DM_getReliability Method	257
9.5.6	DM_getMdlQuality	258
9.5.7	DM_getRMSE Method.....	259
9.5.8	DM_getGainsChart Method	260
9.6	DM_RegApplTask Type and Routines	261
9.6.1	DM_RegApplTask Type	261
9.6.2	DM_defRegApplTask Method	263
9.6.3	DM_getRegMdl Method	264
9.6.4	DM_getRegApplData Method.....	265
9.6.5	DM_getRegOutpData Method	266

9.6.6	DM_getRegResFld method	267
9.6.7	DM_applyRegMdl method	268
9.7	DM_RegResult Type and Routines	269
9.7.1	DM_RegResult Type	269
9.7.2	DM_getPredValue Method.....	270
10	Status Codes	271
11	Conformance	273
11.1	Requirements for conformance.....	273
11.1.1	Support for user-defined types and functions.....	273
11.1.2	Support for data mining phases	273
11.1.3	Support for basic user-defined types	273
11.1.4	Support for data mining techniques	273
11.2	Claims of conformance	274
Annex A	(informative) Implementation-defined Elements	276
Annex B	(informative) Implementation-dependent elements	278
Annex C	(informative) A Scenario Using the Classification Technique.....	279
Index		281

Figures	Page
Figure 1 — The training phase	13
Figure 2 — Gains chart.....	15
Figure 3 — The application phase.....	15
Figure 4 — The testing phase.....	17
Figure 5 — Overview of user-defined types for the training phase.....	17
Figure 6 — Overview of user-defined types for training, testing and application	18
Figure 7 — Overview of user-defined types for bulk application	19

Tables	Page
Table 1 — Model interpretation	13
Table 2 — Gains chart information	14
Table 3 — Items inferred from an association rule model.....	16
Table 4 — Groups of items inferred from an association rule model	16
Table 5 — Values for field types.....	28
Table 6 — Values for operators	86
Table 7 — Values for restriction criteria	87
Table 8 — Values for constraint types.....	95
Table 9 — SQLSTATE class and subclass values.....	272

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 technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

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

This second edition cancels and replaces the first edition (ISO/IEC 13249-6:2002), which has been technically revised.

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*
- *Part 7: History*

Introduction

The purpose of this ISO/IEC 13249 is to define multimedia and application specific types and their associated routines using the user-defined features in ISO/IEC 9075:2003.

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

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

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.

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

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

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

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

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

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

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

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

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

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.

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

Annex C, "A scenario using the classification technique", is an informative annex. It presents explanatory material on how to use this part of ISO/IEC 13249.

Information technology — Database languages — SQL multimedia and application packages —

Part 6: Data mining

1 Scope

ISO/IEC 13249 defines a number of packages of generic data types common to various kinds of data used in multimedia and application areas, to enable that data to be stored and manipulated in an SQL database.

This part of ISO/IEC 13249

- a) introduces the data mining part of ISO/IEC 13249,
- b) gives the references necessary for this part of ISO/IEC 13249,
- c) defines notations and conventions specific to this part of ISO/IEC 13249,
- d) defines concepts specific to this part of ISO/IEC 13249,
- e) defines data mining user-defined types and their associated routines.

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

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

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

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 International Standards

ISO/IEC 13249-1:2006, *Information technology — Database languages — SQL multimedia and application packages — Part 1: Framework*

2.2 Publicly-available specifications

Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

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

3 Terms, definitions, notations and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions in ISO/IEC 13249-1 and the following apply.

3.1.1

application phase

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

3.1.2

association rule

Relationship between two sets of items, which appear frequently (according to a specified threshold), i.e. there is a significant number of groups containing the combination of the values, describing how the presence of one of the sets (the rule head) in a group depends on the presence of the other set (the rule body).

Note: In this document, the term “rule” is sometimes used as a synonym for “association rule” where the meaning is unambiguous from the context.

3.1.3

association rule body

The set of conditions in an association rule. The body of the association rule “Z, if X and Y” is {X, Y}.

Note: In this document, the term “rule body” is sometimes used as a synonym for “association rule body” where the meaning is unambiguous from the context.

3.1.4

association rule confidence

The percentage of transactions that contain all items of the rule out of those transactions that contain the rule body.

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

3.1.5

association rule filter

A specification defining how to compute a subset of a given set of association rules.

Note: In this document, the terms “rule filter” or “filter” are sometimes used as synonyms for “association rule filter” where the meaning is unambiguous from the context.

3.1.6

association rule head

The implication of an association rule. The head of the association rule “Z, if X and Y” is Z.

Note: In this document, the terms “rule head” and “head” are sometimes used as synonyms for “association rule head” where the meaning is unambiguous from the context.

3.1.7

association rule item constraint

A restriction of the contents of an association rule, which can either mandate or bar the presence of a particular item in an association rule.

Note 1: A restriction may apply to the complete rule or only to its body or its head.

Note 2: In this document, the term “item constraint” is sometimes used as a synonym for “association rule item constraint” where the meaning is unambiguous from the context.

3.1.8

association rule lift

A property of an association rule measuring how much more likely the rule is to occur when the rule body is present than statistically expected.

Note: In this document, the terms “rule lift” and “lift” are sometimes used as synonyms for “association rule lift” where the meaning is unambiguous from the context.

3.1.9

association rule model

The result of data mining association rule discovery, which is a set of association rules.

Note: In this document, the term “rule model” is sometimes used as a synonym for “association rule model” where the meaning is unambiguous from the context.

3.1.10

association rule support

The percentage of transactions that contain all items of an association rule.

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

3.1.11

categorical field type

A field type that supports only the <equal operator> as a comparison operator.

Note: There is no defined order, and no arithmetic operations are supported.

3.1.12

class label

Target field used in data mining classification to create a classification model, whose application to data without a class label allows it to predict a value for this class label.

Note: The type of a class label is a categorical field type.

3.1.13

classification cost rate

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

3.1.14

classification error

Percentage of wrongly predicted values among the total number of values predicted during the validation phase of the classification technique.

Note: This error is returned at the end of the test phase.

3.1.15

classification model

The result of data mining classification.

3.1.16

cluster homogeneity

The property of a cluster describing how similar (on a scale between 0 and 1) the records in a cluster are.

Note: This is a measure of the degree of similarity between the records of a cluster according to the criteria for similarity between records.

3.1.17

clustering model

The result of data mining clustering.

3.1.18**data mining application task**

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

3.1.19**data mining association rule discovery**

A data mining technique that searches for common occurrences of different items in data.

Note 1: In this document, the terms “association rule discovery” or “rule discovery” are sometimes used as synonyms for “data mining association rule discovery” where the meaning is unambiguous from the context.

Note 2: Example of usage: store layout.

3.1.20**data mining classification**

Data mining technique that computes a classification model such that the class label value can be predicted by using the model and input field values without the class label value.

Note 1: In this document, the term “classification” is sometimes used as a synonym for “data mining classification” where the meaning is unambiguous from the context.

Note 2: Example of usage: insurance risk prediction.

3.1.21**data mining cluster**

A set of input rows with common characteristics (see data mining clustering).

Note: In this document, the term “cluster” is sometimes used as a synonym for “data mining cluster” where the meaning is unambiguous from the context.

3.1.22**data mining clustering**

A data mining technique that discovers sets of input rows with common characteristics - the *data mining clusters*, so that rows are as homogeneous as possible inside a data mining cluster and as heterogeneous as possible between two data mining clusters.

Note 1: In this document, the term “clustering” is sometimes used as a synonym for “data mining clustering” where the meaning is unambiguous from the context.

Note 2: Example of usage: customer mailings.

3.1.23**data mining data**

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

3.1.24**data mining data type**

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

3.1.25**data mining model**

The result of a data mining run of a data mining technique over a given set of data, which contains the usable correlations discovered in the data.

Note 1: The data mining model can also be used alone as a substitute of the original large amount of data to associate, classify or predict the behavior of additional data (in application and testing phases) .

Note 2: In this document, the terms “mining model” and “model” are sometimes used as synonyms for “data mining model” where the meaning is unambiguous from the context.

3.1.26**data mining regression**

Data mining technique similar to data mining classification except for the numerical type of the target field, which computes a regression model allowing to predict a numerical value.

Note 1: The predicted value might not be identical with any value contained in the data used to build the model.

Note 2: In this document, the term “regression” is sometimes used as a synonym for “data mining regression” where the meaning is unambiguous from the context.

Note 3: Example of usage: customer ranking.

3.1.27

data mining run

Computation of a data mining model using data mining settings.

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

3.1.28

data mining segmentation

synonym for data mining clustering.

Note: Data mining clustering is the term used in this part of ISO/IEC 13249.

3.1.29

data mining sequence

Combination of field values that appears frequently in the same order in the input data (according to a specified threshold), i.e. there is a significant number of groups containing the combination of the values.

Note: In this document, the term “sequence” is sometimes used as a synonym for “data mining sequence” where the meaning is unambiguous from the context.

3.1.30

data mining sequence rule

Combination of field values that appears frequently in the input data (according to a specified threshold), where two data mining sequences are related such that the occurrence of the first influences the frequency of occurrence of the second.

Note: In this document, the terms “sequence rule” or “rule” are sometimes used as synonyms for “data mining sequence rule” where the meaning is unambiguous from the context.

3.1.31

data mining sequence rule discovery

A data mining technique similar to association rule discovery that searches for repeated joint occurrences of items in data and correlations between these items.

Note 1: In this document, the terms “sequence rule discovery” and “sequence discovery” are sometimes used as synonyms for “data mining sequence rule discovery” where the meaning is unambiguous from the context.

Note 2: Example of usage: warranty and quality management.

3.1.32

data mining settings

Settings used in the data mining run that specify the logical data specification and parameters depending on the data mining technique.

Note: In this document, the terms “mining settings” and “settings” are sometimes used as synonyms for “data mining settings” where the meaning is unambiguous from the context.

3.1.33

data mining task

Abstraction for the complete data needed to compute, test or apply a data mining model, containing the data mining settings and the data mining data type.

Note: In this document, the terms “mining task” and “task” are sometimes used as synonyms for “data mining task” where the meaning is unambiguous from the context.

3.1.34

data mining technique

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

3.1.35**data mining test result**

Statistical result returned at the end of the testing phase that contains an evaluation of the correctly and incorrectly predicted values in the test data.

Note 1: This result can be considered a quality criterion of the data mining model.

Note 2: In this document, the term “test result” is sometimes used as a synonym for “data mining test result” where the meaning is unambiguous from the context.

3.1.36**data mining test task**

Abstraction for the complete data needed to test a data mining model, containing values of the data mining model and the data mining data used for the operation.

Note: In this document, the term “test task” is sometimes used as a synonym for “data mining test task” where the meaning is unambiguous from the context.

3.1.37**data warehouse**

A collection of data for decision support.

3.1.38**export**

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

3.1.39**gains chart**

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

3.1.40**import**

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

3.1.41**input field**

Field of the row type of an input row.

3.1.42**input field importance**

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

3.1.43**input row**

A row used as input for a data mining run or an application of a data mining model.

Note: The expected row type for a data mining run is determined by the logical data specification of the settings value or—in case of a test run—by the logical data specification and the target field of the data mining model. It is determined by the logical data specification of the data mining model.

3.1.44**logical data specification**

Part of the data mining settings that lists field names and field types to be used during the data mining run.

Note: These fields will be compared before the data mining run to the field names or aliases defined in the data mining data type.

3.1.45**logical field**

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

3.1.46

logical field type

Type of a logical field, either categorical or numerical, that defines the way the data mining field values will be compared and managed during the data mining run.

Note: In this document, the term “field type” is sometimes used as a shorthand for “logical field type” where the meaning is unambiguous from the context.

3.1.47

model introspection phase

Phase, common to all data mining techniques, in which the data mining model is scrutinized with the goal to understand the correlations in the data that were uncovered during the training phase.

3.1.48

numerical field type

A field type that supports all operations of <numerical value expressions>.

3.1.49

predicted value

The value of the target field predicted using a data mining classification or regression model.

Note: The predicted value can either be a categorical or numerical value.

3.1.50

ranking quality

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

3.1.51

regression model

The result of data mining regression.

3.1.52

root mean squared error

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

3.1.53

rule model

An association rule model or a sequence rule model.

3.1.54

sequence rule model

The result of data mining sequence rule discovery.

Note: In this document, the term “rule model” is sometimes used as a synonym for “sequence rule model” where the meaning is unambiguous from the context.

3.1.55

similarity scale

A scale for a numerical field used to compute similarities between record values for data mining clustering.

Note: The similarity scale is used to map the absolute difference between two real values to a similarity. It is 50% when the values differ by the similarity scale.

3.1.56

source table

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

3.1.57

target field

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

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

3.1.58**testing phase**

Phase that reads a set of input rows containing values for the target field, evaluates each input row as in the application phase, and compares the predicted value to the actual value in the target field.

Note: The testing phase is only used for data mining classification and regression. Results of the testing phase are for example used by a data mining user or a data mining application to determine whether a model can safely be applied based on its quality.

3.1.59**training data**

Data used as input for the training phase.

3.1.60**training phase**

Phase, common to all data mining techniques, in which the data mining model is computed.

3.1.61**validation phase**

Part of the training phase for data mining classification and regression techniques, which uses the calculated data mining model and another set of input rows to test these input rows against the data mining model, as described in the testing phase.

Note: Results of the validation phase are for example used to decide when the algorithm should terminate.

3.2 Notations

3.2.1 Notations provided in ISO/IEC 13249-1

For the purposes of this document, the notations given in ISO/IEC 13249-1 apply.

3.2.2 Notations provided in this part of ISO/IEC 13249

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

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

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

3.3 Conventions

For the purposes of this document, the conventions given in ISO/IEC 13249-1 apply.

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

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

- 1) Method names have the form

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

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

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

`DM_set<technique><property>`.

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

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

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

4 Concepts

4.1 Data mining techniques

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

1) Rule Discovery

a) Association rule discovery:

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

Example application: planning of store layout.

b) Sequence rule discovery:

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

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

2) Clustering:

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

Example application: customer mailings.

3) Classification

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

Example application: insurance risk prediction.

4) Regression

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

Example application: customer ranking.

4.2 Computational phases

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

4.2.1 Training phase

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

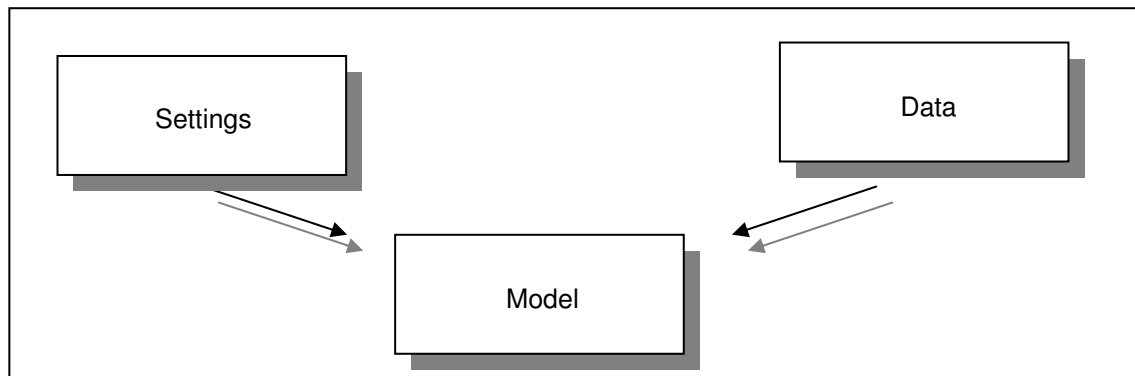


Figure 1 — The training phase

Essentially, data mining needs some *settings* and a set of input rows to compute a data mining model, as shown in Figure 1. However, it is important to note that usually several passes over the input data are needed before a model can be generated. Furthermore, most data mining techniques need to be able to identify fields to assign data mining related information like data mining types and special handling of a field as in case of predictive techniques.

4.2.2 Model introspection phase

The model introspection phase is common to all data mining techniques. It is the phase in which the *data mining model* is scrutinized with the goal to understand the correlations in the data that were uncovered during the training phase. This is done for two purposes, model interpretation and model evaluation:

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

Column Name:	ID	HEAD NAME	HEAD	BODYID	LENGTH	BODY TEXT	SUPP ORT	CONFID ENCE	LIFT
Column Type:	INTE-GER	CHAR-ACTER VARYING	CHAR-ACTER VARYING	INTEGER	INTEGER	CHAR-ACTER VARYING	REAL	REAL	REAL
...	1	scarf	11307	1	3	jeans, hat	4.2	65.0	3.8
example rows:	2	scarf	11307	17	4	shoes, coat, hat	5.8	82.0	0.7
	3	jeans	9891	3	2	trousers	3.0	10.0	0.6
...	4	jeans	9891	17	4	shoes, coat, hat	10.5	55.0	2.0

Table 1 — Model interpretation

Table 1 shows an example of model interpretation, which can be generated from an association rule model using the table function `DM_getRules`.

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

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

As an example for model evaluation consider the problem of determining the risk of a customer canceling an insurance policy. In a database of 100,000 customers, 10,000 actually left the company. A classification model is computed from these data to rank the customers by their propensity to leave. In order to evaluate the quality of the model a gains chart is created. A gains chart shows the ranking quality of a predictive model, i.e. its capability to correctly order records based on the predicted value of their target fields. The method DM_getGainsChart computes a table, such as that shown in Table 2, from a classification model:

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

Table 2 — Gains chart information

In the example in Table 2, from the 23,750 most likely to leave (as expected by the model) 4,951 actually left. If 31,250 customers are considered, 6,408 of them leave, etc. The threshold column holds the minimum confidence of belonging to this group.

Typically, these figures are presented as a chart, such as that shown in Figure 2.

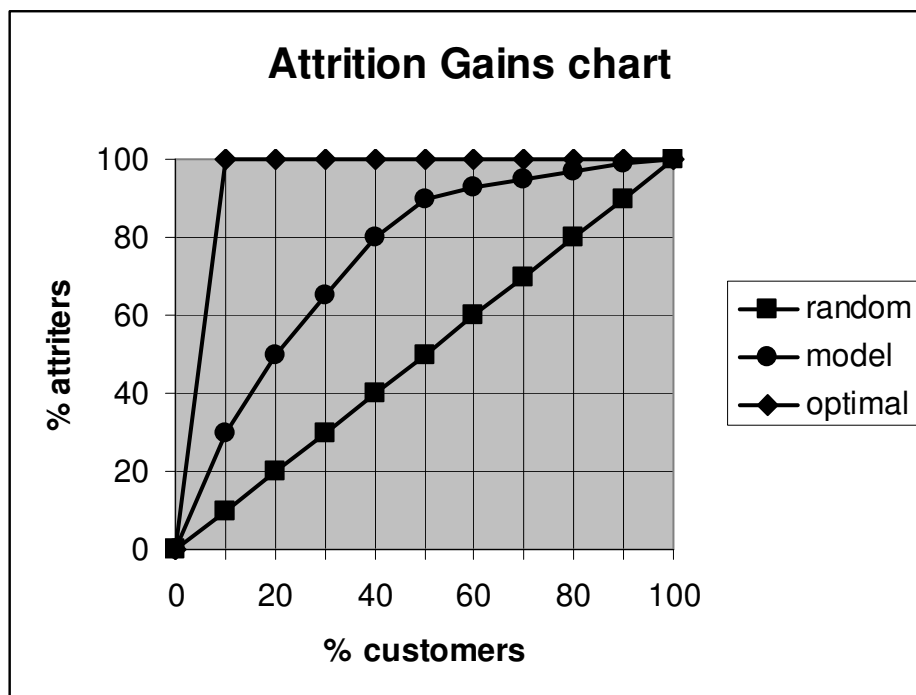


Figure 2 — Gains chart

The chart in Figure 2 shows, for example, that in a group of about 25% of the customers selected by the model, there are likely to be about half of the overall attriters (the customers who have left the company). In contrast, random selection would only deliver a quarter, and with perfect knowledge one would identify all of them in just 10% of the customers.

4.2.3 Application phase

During the application of a model, input rows are evaluated against a model and one or more values are computed, as shown in general terms in Figure 3. For instance, applying a clustering model to a customer input row would assign a cluster id and a quality value to the customer input row. In case of classification a class label and a confidence value is computed for the input row.

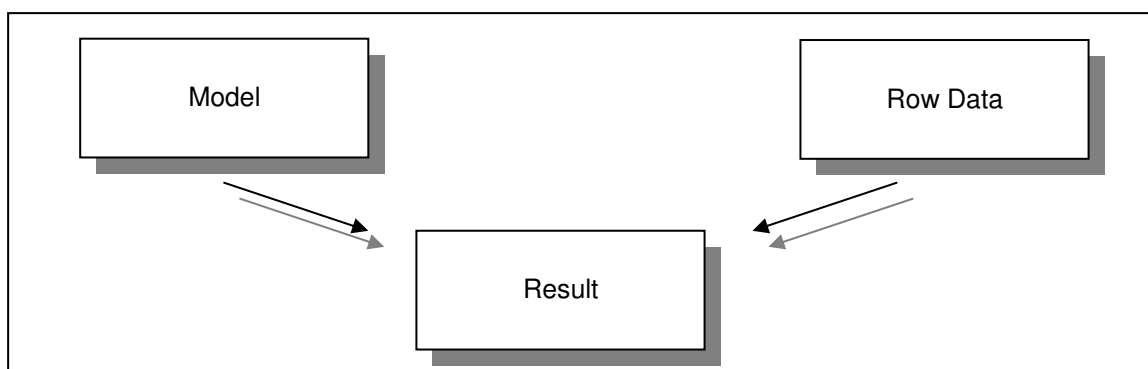


Figure 3 — The application phase

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

The application of association rule models is different from that of the other data mining functions. The model is applied to an item set. As output, a table is generated holding all the items that can be inferred from the input set using one or more rules in the model. In that way any inferred items can be delivered together with additional properties. In particular these are the support of the inferred item and its confidence. The output table has a layout such as shown in Table 3.

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

Table 3 — Items inferred from an association rule model

In the case of bulk application of rule models, using DM_RuleApplTask, the input sets are defined as in association rule training by defining a group column. For each of the sets defined in that way, items may be inferred, which are then grouped together using a group column as well. The output is a table such as that shown in Table 4.

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

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

4.2.4 Testing phase

Only the techniques for classification and regression have a testing phase. To test means to check the quality of the prediction using the data mining model. So, during the testing phase, a set of input rows

with the target field is read and for each input row the application of the mining model is invoked, as shown in general terms in Figure 4. Then, the predicted value is compared to the actual value in the target field. When all input rows are read, a statistical result about the false predictions is computed and returned as the test result.

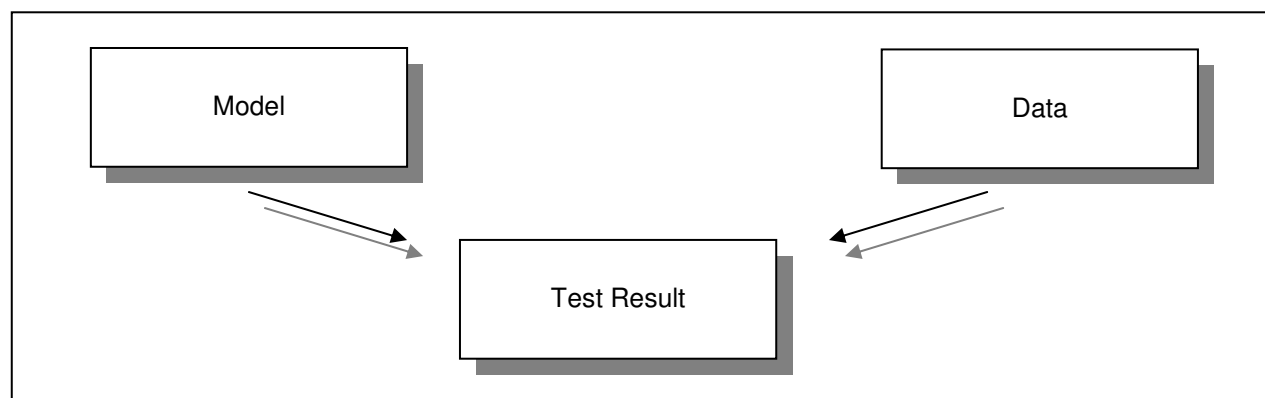


Figure 4 — The testing phase

4.3 Mapping on user-defined types

4.3.1 Target application

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

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

4.3.2 The relationships of the user-defined types

Before the user-defined types of this part of ISO/IEC 13249 are introduced, it is necessary to describe how the types are related.

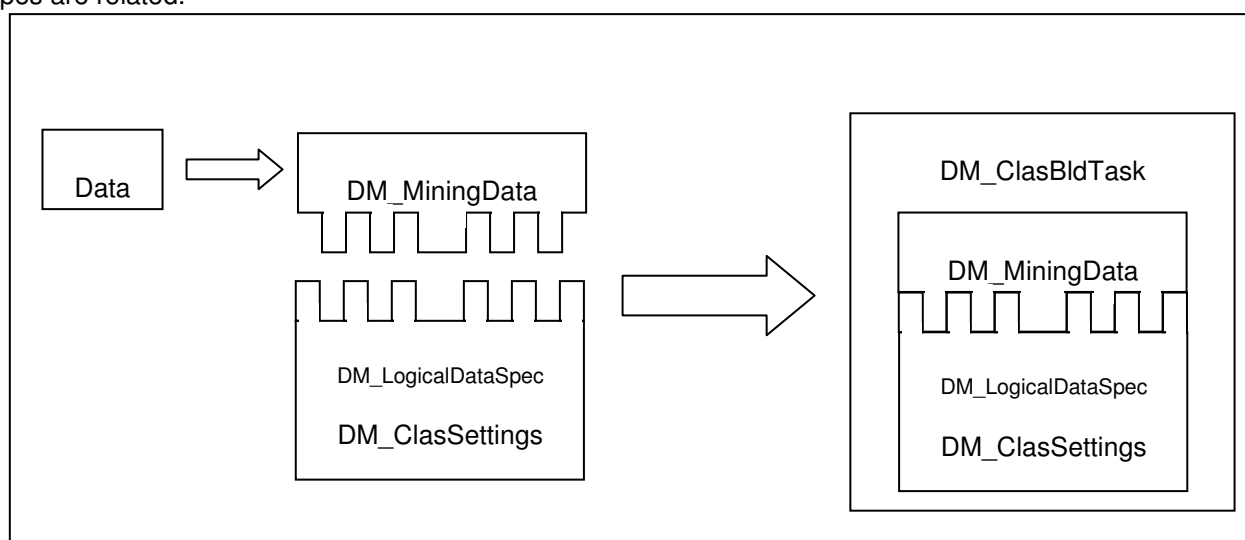


Figure 5 — Overview of user-defined types for the training phase

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

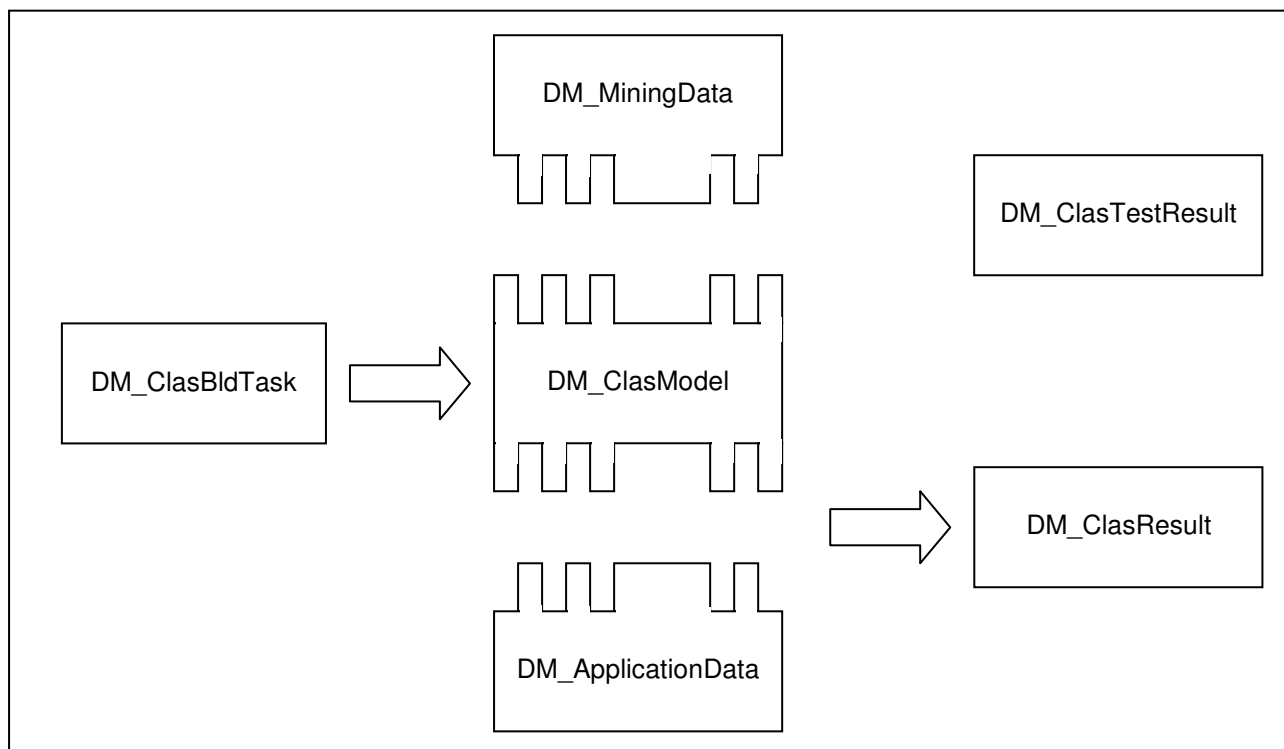


Figure 6 — Overview of user-defined types for training, testing and application

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

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

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

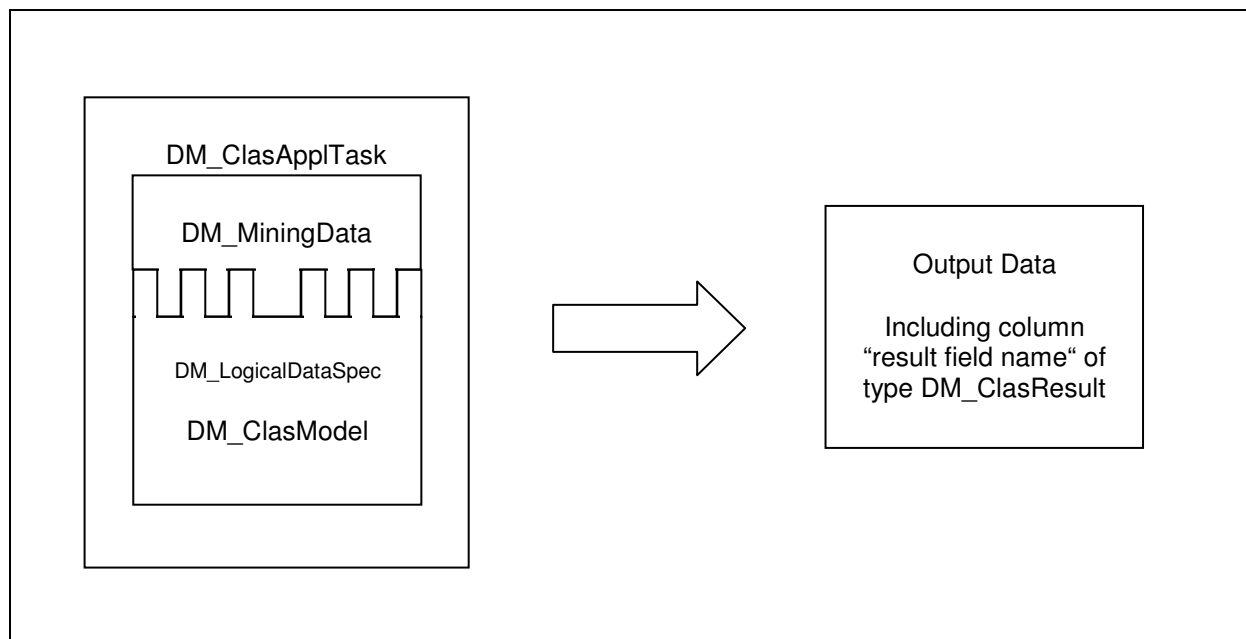


Figure 7 — Overview of user-defined types for bulk application

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

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

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

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

- 1) The DM_MiningData user-defined type is an abstraction for real data contained in tables or views. A value of this type represents the metadata to access the real tables for a later data mining training, test or application run. This might include internal transformations used to clean or pre-process data in order to increase the quality of data mining models or data mining test results.
- 2) The DM_LogicalDataSpec user-defined type defines the input fields used by data mining training, data mining test, or data mining application runs. It allows the specification of data mining field related information. The most important information of this category is the association of a logical field type. The logical field type defines how an input field is handled by the data mining techniques. For instance, a field can be declared as categorical. This tells the data mining algorithms that there is no order defined for the values of this type and that by default only values contained in the input data are also valid values.
- 3) The DM_ApplicationData user-defined type is introduced as a container for data used to apply a data mining model. Basically, it is an abstraction for a set of values with associated names representing a single row of input data. This type is needed because the input values to apply a data mining model shall be mapped onto the logical data specification of the mining model.

4.3.4 User-defined types that are related to mining techniques

The types introduced in Subclause 4.3.3, “User-defined types that are not related to mining techniques”, do not provide any methods to compute any mining related information. These types can only be used to define the metadata needed for later data mining functions. The following categories of user-defined

types are used for data mining processing. For each mining technique, there is at least one type in each of the following categories:

- 1) The mining task types like the `DM_ClasBldTask` type are intended to provide all information actually needed to start a data mining training, test or application run to compute, test or apply a data mining model. In a warehouse environment, this type might be used to store fully defined data mining runs such that the actual computation can be scheduled.
- 2) The mining model type (e.g. `DM_ClasModel`) is defined as an abstraction for an actual data mining model. It provides methods to access the properties of the model as well as methods to apply or to test the model. Values of the mining model type can only be generated by either using the build method of a corresponding mining task value or by importing the model from somewhere else.

SQL/MM data mining requires support for data mining models in PMML format (see subclause 2.2 Publicly-available specifications on page 2). This format is based on XML and widely used for the exchange of data mining models. The internal representation of data mining models is implementation-dependent, but it must be possible to export a model to a PMML format and to import a model from a PMML format

- 3) The data mining test result type (e.g. `DM_ClasTestResult`) is introduced to hold the result information of a test run computed for a data mining model. A separate type was introduced because a test result might consist of an arbitrary number of values resulting from the invocation of a method having the characteristics of an aggregate function. That means that a data mining test run reads a set of input rows, applies a previously computed model and aggregates information about the quality of the prediction. The data mining test result represents the aggregate after the last input row was read.
- 4) The data mining application result type (e.g. `DM_ClasResult`) is also introduced because the application of a data mining model might also return multiple values. The computational phase of the data mining application mode is pretty simple in the sense that a value of `DM_ApplicationData` is used in the apply method of a data mining model to compute a value of type application result. The methods of the application result value can then be used to query the details.

- 5) The data mining settings types (e.g. DM_ClusSettings) are introduced to hold all the information necessary to specify a data mining run. In particular, the settings types support the following parameters:
- a) DM_ClasSettings:
The purpose of classification is to predict the class label value of previously unseen data of which this value is unknown. From a mathematical standpoint, the number of wrong predictions it leads to determines the quality of a classification model. The *classification cost rate* defines the maximal classification error tolerated during the validation phase, above which the model is considered unsatisfactory. In practice, however, some wrong predictions will cause large damage, while others can be tolerated more easily. To take such knowledge into account, the specific *cost* for the wrong prediction of one class label value instead of another can be specified.
 - b) DM_ClusSettings:
The purpose of clustering is the discovery of sets of input rows with common characteristics - the *clusters*. A basic operation in clustering is the computation of distance or similarity between pairs of records. This is determined by comparing their individual field values.
It is not always useful to assign the same importance to all fields. In order to let the user define the relative importance of the fields, *field weights* are often used to define the relative importance of the fields.
For the comparison of numerical field values, functions are used such as the absolute difference. This may produce poor results if the variances are largely different between different fields. A *similarity scale* is a value to allow the algorithm to compensate for this. If two values are one similarity scale apart, they should have medium similarity, i.e. their similarity becomes 0.5 in a range from 0 to 1.
 - c) DM_RegSettings:
Regression is a data mining technique similar to classification except for the numerical type of the target field. It computes a regression model allowing to predict numerical values. In order to specify the maximum allowed squared Pearson correlation coefficient for the training phase, the parameter *r-squared* is used.
 - d) DM_RuleSettings:
The association rules function discovers combinations of field values that appear frequently. The *minimum support* parameter specifies to compute only rules that occur in a sufficiently large percentage of the cases. *Minimum confidence* allows only rules that hold with a given minimal probability, and the *maximum rule length* puts a restriction on the number of values that are allowed in a single rule.
- 6) The rule filter type DM_RuleFilter holds all the information necessary to specify a subset of association or sequence rules. Using this type allows the user to select a specific subset of rules from a rule model of type DM_RuleModel. Alternatively, values of type DM_RuleFilter can be added to rule settings. As a consequence, only rules that satisfy the filter conditions are kept during the training phase.

5 Data Mining Data Types

5.1 DM_LogicalDataSpec Type and Routines

5.1.1 DM_LogicalDataSpec Type

Purpose

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

Definition

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

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

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

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

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

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

```

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

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

```

Definitional Rules

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

Description

- 1) The *DM_LogicalDataSpec* type provides for public use:
 - a) a method *DM_addDataSpecFld*(*CHARACTER VARYING*),
 - b) a method *DM_remDataSpecFld*(*CHARACTER VARYING*),
 - c) a method *DM_getNumFields*(),
 - d) a method *DM_getFldName*(*INTEGER*),
 - e) a method *DM_setFldType*(*CHARACTER VARYING*, *INTEGER*),
 - f) a method *DM_getFldType*(*CHARACTER VARYING*),
 - g) a method *DM_isCompatible*(*DM_LogicalDataSpec*).

5.1.2 DM_addDataSpecFld Method

Purpose

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

Definition

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

Definitional Rules

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

Description

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

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

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

Case:

- a) If *fieldName* is the null value, then *SELF* is returned.
 - b) If a field having a name equal to *fieldName* is contained in the set of fields of *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field already defined*.
 - c) Otherwise, it is a value of type *DM_LogicalDataSpec* consisting of the set of fields contained in *SELF* followed by a new field with *fieldName* as its name. The type of the field is set to the implementation-defined default type. The position of the field is set to an implementation-dependent value.

5.1.3 DM_remDataSpecFld Method

Purpose

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

Definition

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

Definitional Rules

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

Description

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

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

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

Case:

- a) If *fieldName* is the null value, then *SELF* is returned.
 - b) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.
 - c) Otherwise, it is the *DM_LogicalDataSpec* value derived from *SELF* by deleting the field with a name equal to *fieldName* from the set of fields contained in *SELF*. The positions of the fields in the returned value are implementation-dependent.

5.1.4 DM_getNumFields Method

Purpose

Returns the number of fields of the DM_LogicalDataSpec value.

Definition

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

Description

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

5.1.5 DM_getFldName Method

Purpose

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

Definition

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

Definitional Rules

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

Description

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

- a) an *INTEGER* value *position*.

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

Case:

- a) If *position* is greater than 0 (zero) and less than or equal to the result of a call of *DM_getNumFields()*, then the name of the mining field with the associated number *position*.
 - b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – parameter out of range*.

5.1.6 DM_setFldType Method

Purpose

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

Definition

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

Definitional Rules

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

Description

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

- a) a *CHARACTER VARYING* value *fieldName*,
- b) an *INTEGER* value *miningType*.

- 2) The following values for field types are defined:

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

Table 5 — Values for field types

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

Case:

- a) If *fieldName* is the null value, then *SELF* is returned.
- b) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.
- c) If *miningType* is the null value, then the field type of the field with name *fieldName* of the set of fields of *SELF* is reset to the implementation-defined default.
- d) If *miningType* is not a value defined in Table 5 — Values for field types, then an exception condition is raised: *SQL/MM Data Mining Exception – parameter out of range*.

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

5.1.7 DM_getFldType Method

Purpose

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

Definition

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

Definitional Rules

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

Description

- 1) The method *DM_getFldType* takes the following input parameter:
 - a) a *CHARACTER VARYING* value *fieldName*.
- 2) The result of the invocation *DM_getFldType(fieldName)* is determined as follows:

Case:

- a) If *fieldName* is not equal to the name of any field contained in *SELF*, then an exception condition is raised: *SQL/MM Data Mining exception – field not defined in data specification*.
- b) If *SELF* is equal to *SELF.DM_setFldType(fieldName, v)* for some SMALLINT value *v* representing a field type, then *v*. For defined values for the field type see Table 5 — Values for field types on page 28.
- c) Otherwise, the null value.

5.1.8 DM_isCompatible Method

Purpose

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

Definition

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

Description

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

a) a *DM_LogicalDataSpec* value *dataSpec*.

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

- a) 1 (one) if *dataSpec* is compatible with *SELF*. This is the case if for each field F in *SELF*, there is a field G in *dataSpec* with an identical name. The type of G shall be *DM_Numerical* whenever the type of F is *DM_Numerical*.
- b) 0 (zero) otherwise.

5.2 DM_MiningData Type and Routines

5.2.1 DM_MiningData Type

Purpose

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

Definition

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

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

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

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

Definitional Rules

- 1) *DM_MaxContentLength* is the implementation-defined maximum length for the *DM_content* of a *DM_MiningData*.
- 2) *DM_MaxTableNameLength* is the implementation-defined maximum number of characters for the name of a table.
- 3) *DM_MaxFieldNameLength* is the implementation-defined maximum number of characters for the name of a field.
- 4) *DM_MaxFieldAliasLength* is the implementation-defined maximum number of characters for strings representing a field name within a logical data specification.

Description

- 1) The *DM_MiningData* type provides for public use:
 - a) a method *DM_defMiningData*(*CHARACTER VARYING*),
 - b) a method *DM_setFldAlias*(*CHARACTER VARYING*, *CHARACTER VARYING*),

- c) a method *DM_genDataSpec()*.

5.2.2 DM_defMiningData Method

Purpose

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

Definition

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

Definitional Rules

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

Description

- 1) The method *DM_defMiningData* takes the following input parameter:
 - a) a *CHARACTER VARYING* value *tableName*.
- 2) The result of the invocation *DM_defMiningData(tableName)* is determined as follows:
 - a) If *tableName* identifies a valid table, then a value of *DM_MiningData* determined by the identified table. For each column of the table of type character, character varying, datetime, or numerical a field description is created in the *DM_MiningData* value. The field has an associated name and an associated alias name, which are both identical to the name of the column in the table.
 - b) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid table name*.

5.2.3 DM_setFldAlias Method

Purpose

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

Definition

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

Definitional Rules

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

Description

- 1) The method *DM_setFldAlias* takes the following input parameters:
 - a) a *CHARACTER VARYING* value *dataField*,
 - b) a *CHARACTER VARYING* value *aliasName*.
- 2) The result of the invocation *DM_setFldAlias(dataField, aliasName)* is determined as follows:
 - a) If *dataField* is the null value, then *SELF*.
 - b) If *aliasName* is not the null value and there is a field contained in *SELF* with an alias name equal to *aliasName*, then an exception condition is raised: *SQL/MM Data Mining exception – alias already in use*.
 - c) If *dataField* is a valid name of a field in *SELF* and *aliasName* is not the null value, then the alias of that field is set to the value of *aliasName*.
 - d) If *dataField* is a valid name of a field in *SELF* and it is not the alias of any other field in *SELF*, and *aliasName* is the null value, then the alias of that field is set to the value of *dataField*.
 - e) If *dataField* is the alias of some other field in *SELF* and *aliasName* is the null value, then an exception condition is raised: *SQL/MM Data Mining exception – alias already in use*.
 - f) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid field name*.

5.2.4 DM_genDataSpec Method

Purpose

Generates value of type DM_genDataSpec.

Definition

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

Description

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

Case:

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

5.3 DM_ApplicationData Type and Routines

5.3.1 DM_ApplicationData Type

Purpose

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

Definition

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

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

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

Definitional Rules

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

Description

- 1) The *DM_ApplicationData* type provides for public use:
 - a) a method *DM_genDataSpec()*,
 - b) a method *DM_impApplData(CHARACTER LARGE OBJECT(DM_MaxContentLength))*.

5.3.2 DM_genDataSpec Method

Purpose

Return the DM_LogicalDataSpec corresponding to SELF.

Definition

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

Description

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

5.3.3 DM_impApplData Method

Purpose

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

Definition

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

Definitional Rules

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

Description

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

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

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

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

Case:

- a) if *inputRow* is a valid representation of application input data, then a value of *DM_ApplicationData* corresponding to the input data.
- 4) Otherwise, an exception condition is raised: *SQL/MM Data Mining exception – invalid application input data format*.

6 Association Rules

6.1 DM_RuleSettings Type and Routines

6.1.1 DM_RuleSettings Type

Purpose

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

Definition

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

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

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

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

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

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

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