Documentation of the MiningMart Meta Model (M⁴)
Martin Scholz, Timm Euler
Dortmund, January 13, 2003
1 What this document is about

This document describes the database tables that are used to store the MiningMart Meta Model (M^4). During the development of the MiningMart system, M^4 has changed to some extend with respect to the previous document about it, the MiningMart Deliverable 7. For future reference therefore this document should be used.

Section 2 documents the actual M^4 tables. Section 3 describes a few special tables which are used to enable the MiningMart system to control the administration of the metadata. For all tables, their columns are listed together with their datatype. Primary keys are underlined. Tables are listed alphabetically in each section.

2 The M4 Schema

This section contains the tables that are used to store the metamodel. Note that the different objects of the metamodel (such as concepts or columns), called M4 objects, are uniquely referenced by a global ID (an integer). This global ID also serves as the primary key in the tables that store these objects. This means that there is a constraint on all the primary keys of the tables listed here which cannot be enforced by database-internal mechanisms.

2.1 BASEATTRIB_T

A BaseAttribute is an atomic feature of a Concept. It represents a Column in the database on the conceptual level.

- **BA_ID** datatype: integer, no missing values
  Unique M4 Id.

- **BA_NAME** datatype: string (length: 100), no missing values
  Name of this BaseAttribute.

- **BA_CONDTID** datatype: integer, no missing values
  Conceptual datatype of this BaseAttribute. Foreign link to table CONDT T (section 2.15).

- **BA_RELEVANCE** datatype: string (length: 5)
  One of 'YES' or 'NO'. Not used.

- **BA_ATTRIBTYPE** datatype: string (length: 15), no missing values
  One of 'BASE', 'DB' or 'MINING'. 'DB' is only for BaseAttributes whose Columns existed in the database from the start. 'MINING' is for those that are created by the MiningMart compiler. 'BASE' is not used.
• **BA.MCFID** datatype: integer
  If this BaseAttribute is part of a MultiColumnFeature, this field is used for the foreign link to table MCFEATURE_T (section 2.20).

• **BA.HIDE** datatype: string (length: 5)
  One of 'YES' or 'NO'. Should be 'NO'.

• **BA.VALID** datatype: string (length: 5)
  One of 'YES' or 'NO'. Replace ThisStringByAnExplanationForAllTheValid-Fields.

### 2.2 **BA.COLUMN.T**

This table links BaseAttributes and Columns. Although it allows an n:m relation, each Column has only one BaseAttribute.

• **BAC.ID** datatype: integer, no missing values
  Unique table Id.

• **BAC.BAID** datatype: integer, no missing values
  Foreign link to table BASEATTRIB.T (section 2.1).

• **BAC.COLID** datatype: integer, no missing values
  Foreign link to table COLUMN.T (section 2.9).

### 2.3 **BA(CONCEPT.T**

This table links BaseAttributes and Concepts. Although it allows an n:m relation, each BaseAttribute has only one Concept.

• **BC.ID** datatype: integer, no missing values
  Unique table Id.

• **BC.BAID** datatype: integer, no missing values
  Foreign link to table BASEATTRIB.T (section 2.1).

• **BC.CONID** datatype: integer, no missing values
  Foreign link to table CONCEPT.T (section 2.13).

### 2.4 **CASE.T**

This table holds the general information about a MiningMart Case.

• **CA.ID** datatype: integer, no missing values
  Unique M4 Id.
• **CA_NAME** datatype: string (length: 100), no missing values
  Name of this Case.

• **CA_MODE** datatype: string (length: 15), no missing values
  One of 'DESIGN', 'TEST' or 'FINAL'.

• **CA_POPULATION** datatype: integer
  Not used.

• **CA_OUTPUT** datatype: integer
  Not used.

• **CA_VALID** datatype: string (length: 5)
  One of 'YES' or 'NO'. ReplaceThisStringByAnExplanationForAllTheValid-Fields.

### 2.5 CASEATTRIB_T

• **CAA_ID** datatype: integer, no missing values
  Unique table Id.

• **CAA_CAID** datatype: integer, no missing values
  Foreign link to table CASE_T (section 2.4).

• **CAA_OBJID** datatype: integer, no missing values
  The Id of any M4 object.

• **CAA_OBJTYPE** datatype: string (length: 20), no missing values
  The type of the M4 object. Allowed types are 'CON' for Concepts, 'REL' for Relations, 'V' for Values, 'BA' for BaseAttributes, and 'MCF' for MultiColumnFeatures.

### 2.6 CASEINPUT_T

• **CAL_ID** datatype: integer, no missing values
  Unique table Id.

• **CAL_CAID** datatype: integer, no missing values
  Foreign link to table CASE_T (section 2.4).

• **CAL_OBJID** datatype: integer
  The Id of any M4 object.

• **CAL_OBJTYPE** datatype: string (length: 20), no missing values
  The type of the M4 object. Allowed types are 'CON' for Concepts, 'REL' for Relations, 'V' for Values, 'BA' for BaseAttributes, and 'MCF' for MultiColumnFeatures.
2.7 CHAIN_T
A Chain is a (sub-)sequence of Steps that is part of a Case.

- **CH_ID** datatype: integer, no missing values
  Unique M4 Id.

- **CH_CASEID** datatype: integer, no missing values
  Foreign link to table CASE_T (section 2.4).

- **CH_NAME** datatype: string (length: 100) no missing values
  Name of this Chain.

- **CH_DESCRIP** datatype: string (length: 400)
  Free text that describes the function of this Chain in its Case in abstract terms.

2.8 COL_DATATYPE_T
This table stores the MiningMart datatypes for Columns (relational datatypes).

- **COLDT_ID** datatype: integer, no missing values
  Unique M4 Id.

- **COLDT_NAME** datatype: string (length: 100), no missing values
  Name of this relational datatype.

2.9 COLUMN_T
Entries in this table represent a Column.

- **COL_ID** datatype: integer, no missing values
  Unique M4 Id.

- **COL_NAME** datatype: string (length: 100), no missing values
  Name of this Column.

- **COL_CSID** datatype: integer
  Foreign link to table COLUMNSET_T (section 2.10).

- **COL_COLDT_ID** datatype: integer, no missing values
  Foreign link to table COL_DATATYPE_T (section 2.8).

- **COL_SQL** datatype: string (length: 4000)
  SQL definition for this Column. It is used if this Column was constructed by a Feature Construction Operator. It defines how the entries in this Column are computed from other Columns.
2.10 COLUMNSET_T

Entries in this table represent a database table, a view, a snapshot or a materialized view.

- **CS_ID** datatype: integer, no missing values
  Unique M4 Id.

- **CS_SCHEMA** datatype: string (length: 100), no missing values
  Name of the database schema in which the table or view represented by this ColumnSet lives.

- **CS_NAME** datatype: string (length: 100), no missing values
  Name of this ColumnSet (equals the name of the table or view represented).

- **CS_FILE** datatype: string (length: 500)
  Not used.

- **CS_USER** datatype: string (length: 500)
  Not used.

- **CS_CONNECT** datatype: string (length: 500)
  Not used.

- **CS_TYPE** datatype: string (length: 5), no missing values
  One of 'T', 'V', 'SN' or 'MV' for table, view, snapshot and materialized view, respectively.

- **CS_SQL** datatype: string (length: 4000)
  SQL definition of this ColumnSet. For views, their definition is stored here. For tables, this field may be NULL or may contain the same entry as CS_NAME.

- **CS_CONID** datatype: integer
  Foreign link to table CONCEPT_T (section 2.13).

- **CS_MSBREANCH** datatype: string (length: 1000)
  A string that is controlled by the MiningMart compiler. It contains information about how this ColumnSet was created if Multistep-Operators are present in the chain of steps. It should not be manipulated manually.
2.11 COLSTATIST1_T
This table stores statistical information about columns. It contains at most one entry per column object. Further statistics about a column can be found in the table COLSTATIST2_T (section 2.12). Note that the MiningMart compiler fills this table only when necessary or when explicitly demanded. Several fields only make sense for certain relational datatypes (numeric or ordinal).

- **COLST1_ID** datatype: integer, no missing values
  Unique table Id.

- **COLST1_COLID** datatype: integer, no missing values
  Foreign link to table COLUMN_T (section 2.9).

- **COLST1_UNIQUE** datatype: integer
  Stores the number of unique (distinct) entries in the referenced column.

- **COLST1_MISSING** datatype: integer
  Stores the number of missing values (NULL entries) in the referenced column.

- **COLST1_MIN** datatype: string (length: 100)
  Contains the smallest value in the referenced column (if a minimum exists/is defined).

- **COLST1_MAX** datatype: string (length: 100)
  Contains the biggest value in the referenced column (if a maximum exists/is defined).

- **COLST1_Avg** datatype: number (precision: 20,5)
  Contains the average of the values in the referenced column (if an average is defined).

- **COLST1_STDDEV** datatype: number (precision: 20,5)
  Contains the standard deviation of the values in the referenced column.

- **COLST1_VARIANCE** datatype: number (precision: 38,5)
  Contains the variance of the values in the referenced column.

- **COLST1_MEDIAN** datatype: string (length: 100)
  Contains the median value of the referenced column.

- **COLST1_MODAL** datatype: string (length: 100)
  Contains the modal value of the referenced column.
2.12 COLSTATIST2_T

This table stores information about the distribution of values in columns. For non-numeric columns, it contains one entry per value of the column. For numeric columns, the range of values of that column is discretized in some way, and this table contains one entry per interval. Further statistics about a column can be found in the table COLSTATIST1_T (section 2.11). Note that the MiningMart compiler fills this table only when necessary or when explicitly demanded.

- **COLST2_ID** datatype: integer, no missing values
  Unique table Id.
- **COLST2_COLID** datatype: integer, no missing values
  Foreign link to table COLUMN_T (section 2.9).
- **COLST2_DISTVALUE** datatype: string (length: 100), no missing values
  For non-numeric columns, this field contains one of its values. For numeric columns, this field contains the average of the values in one of its intervals.
- **COLST2_DISTCOUNT** datatype: integer, no missing values
  Stores the number of entries with the value (or belonging to the interval) specified in COLST2_DISTVALUE.
- **COLST2_DISTMIN** datatype: number (precision: 20,5)
  For numeric columns, the smallest entry within the interval is stored here.
- **COLST2_DISTMAX** datatype: number (precision: 20,5)
  For numeric columns, the biggest entry within the interval is stored here.

2.13 CONCEPT_T

Entries in this table represent a Concept.

- **CON_ID** datatype: integer, no missing values
  Unique M4 Id.
- **CON_CAID** datatype: integer
  Foreign link to table CASE_T (section 2.4). Gives the Case in which this Concept is used.
- **CON_NAME** datatype: string (length: 100), no missing values
  Name of this Concept.
- **CON_TYPE** datatype: string (length: 10), no missing values
  Type of this Concept. Allowed values are 'DB', 'BASE' and 'MINING'. 'DB' is used for Concepts which refer to original tables in the business data.
'MINING' is used for Concepts which refer to Columnsets that the MiningMart compiler creates during execution of an operator chain. 'BASE' is not used.

- **CON_SUBCONRESTR** datatype: string (length: 4000)
  Not used.

- **CON_VALID** datatype: string (length: 5)
  One of 'YES' or 'NO'. ReplaceThisStringByAnExplanationForAllTheValid-Fields.

### 2.14 CONCEPTISA_T

This table may be used to define hierarchies of Concepts.

- **CISA_ID** datatype: integer, no missing values
  Unique table Id.

- **CISA_SUPERCONID** datatype: integer, no missing values
  Foreign link to table CONCEPT_T (section 2.13). Defines the superconcept(s) for the Concept referenced in CISA_SUBCONID.

- **CISA_SUBCONID** datatype: integer, no missing values
  Foreign link to table CONCEPT_T (section 2.13). Defines the Concept for which the superconcept(s) is/are given in CISA_SUPERCONID.

### 2.15 CON_DATATYPE_T

This table stores the MiningMart datatypes for Concepts (conceptual datatypes).

- **CONDT_ID** datatype: integer, no missing values
  Unique M4 Id.

- **CONDT_NAME** datatype: string (length: 100), no missing values
  Name of this conceptual datatype.

### 2.16 CSSTATIST_T

This table stores statistical information about Columnsets. It contains atmost one entry per columnset object. Note that the MiningMart compiler fills this table only when necessary or when explicitly demanded.

- **CSST_ID** datatype: integer, no missing values
  Unique table Id.

- **CSST_CSID** datatype: integer, no missing values
  Foreign link to table COLUMNSET_T (section 2.10).
• **CSST\_ALL** datatype: integer
  Stores the number of rows in the referenced Columnset.

• **CSST\_ORD** datatype: integer
  Stores the number of Columns with datatype NUMBER in the referenced Columnset. This refers to table `COL\_DATATYPE\_T` (section 2.8).

• **CSST\_NOM** datatype: integer
  Stores the number of Columns with datatype STRING in the referenced Columnset. This refers to table `COL\_DATATYPE\_T` (section 2.8).

• **CSST\_TIME** datatype: integer
  Stores the number of Columns with datatype DATE in the referenced Columnset. This refers to table `COL\_DATATYPE\_T` (section 2.8).

2.17 **DOCU\_T**

This table may be used to store any textual information about any M4 object.

• **DOC\_ID** datatype: integer, no missing values
  Unique table Id.

• **DOC\_OBJID** datatype: integer, no missing values
  Reference to the table Id of any of the tables described in this section (section 2).

• **DOC\_OBJTYPE** datatype: string (length: 20), no missing values
  Specifies the table which the field DOC\_OBJID refers to. Allowed are all prefixes of the table Id fields.

• **DOC\_TEXT** datatype: string (length: 4000), no missing values
  Textual description for the referenced M4 object.

2.18 **KEYHEAD\_T**

This table and the table `KEYMEMBER\_T` (section 2.19) are used to model a Key relationship between Columns. This table stores the information about the involved Columnsets for one Key relationship.

• **KH\_ID** datatype: integer, no missing values
  Unique M4 Id.

• **KH\_NAME** datatype: string (length: 100), no missing values
  Name of this Key relationship.
• **KH_PKCSID** datatype: integer  
  Foreign link to table **COLUMNSET_T** (section 2.10). The Columnset that contains the primary key Column is referenced here.

• **KH_FKCSID** datatype: integer  
  Foreign link to table **COLUMNSET_T** (section 2.10). The Columnset that contains the foreign key Column is referenced here.

### 2.19 **KEYMEMBER_T**

This table and the table **KEYHEAD_T** (section 2.18) are used to model a Key relationship between Columns. This table stores the information about the involved Columns for one Key relationship.

• **KM_ID** datatype: integer, no missing values  
  Unique M4 Id.

• **KM_KHID** datatype: integer, no missing values  
  Foreign link to table **KEYHEAD_T** (section 2.18).

• **KM_PKCOLID** datatype: integer  
  Foreign link to table **COLUMNSET_T** (section 2.9). Identifies the primary key Column of this Key relationship.

• **KM_FKCOLID** datatype: integer  
  Foreign link to table **COLUMNSET_T** (section 2.9). Identifies the foreign key Column of this Key relationship.

• **KM_POS** datatype: integer, no missing values  
  Not used.

• **KM_FKTYPE** datatype: string (length: 10)  
  Not used.

### 2.20 **MCFEATURE_T**

A MultiColumnFeature is represented in this table. MultiColumnFeatures bundle BaseAttributes so that several Columns can be seen as one feature of a Concept.

• **MCF_ID** datatype: integer, no missing values  
  Unique M4 Id.

• **MCF_NAME** datatype: string (length: 100), no missing values  
  Name of this MultiColumnFeature.
• MCF_CONID datatype: integer
  Foreign link to table CONCEPT_T (section 2.13). Identifies the Concept that this MultiColumnFeature belongs to.

• MCF_VALID datatype: string (length: 5)
  One of 'YES' or 'NO'. ReplaceThisStringByAnExplanationForAllTheValid-Fields.

2.21 OP_ASSERT_T

This table is used to store parts of the static information about an operator, namely its assertions. Please refer to the MiningMart technical report TR18.1 "Representing Constraints, Conditions and Assertions in M4".

• ASSERT_ID datatype: integer, no missing values
  Unique table Id.

• ASSERT_OPID datatype: integer, no missing values
  Foreign link to table OPERATOR_T.

• ASSERT_TYPE datatype: string (length: 10) no missing values
  Type of this assertion (see technical report).

• ASSERT_OBJ1 datatype: string (length: 100) no missing values
  Object 1 for this assertion (see technical report).

• ASSERT_OBJ2 datatype: string (length: 100)
  Object 2 for this assertion (see technical report).

• ASSERT_DOCU datatype: string (length: 400)
  Free text to explain this assertion.

• ASSERT_SQL datatype: string (length: 1000)
  An SQL string for this assertion (see technical report).

2.22 OP_COND_T

This table is used to store parts of the static information about an operator, namely its conditions. Please refer to the MiningMart technical report TR18.1 "Representing Constraints, Conditions and Assertions in M4".

• COND_ID datatype: integer, no missing values
  Unique table Id.

• COND_OPID datatype: integer, no missing values
  Foreign link to table OPERATOR_T.
• **COND_TYPE** datatype: string (length: 10) no missing values
  Type of this condition (see technical report).

• **COND_OBJ1** datatype: string (length: 100) no missing values
  Object 1 for this condition (see technical report).

• **COND_OBJ2** datatype: string (length: 100)
  Object 2 for this condition (see technical report).

• **COND_DCU** datatype: string (length: 400)
  Free text to explain this condition.

• **COND_SQL** datatype: string (length: 1000)
  An SQL string for this condition (see technical report).

### 2.23 OP CONSTR.T

This table is used to store parts of the static information about an operator, namely its constraints. Please refer to the MiningMart technical report TR18.1 “Representing Constraints, Conditions and Assertions in M4”.

• **CONSTR_ID** datatype: integer, no missing values
  Unique table Id.

• **CONSTR_OPI** datatype: integer, no missing values
  Foreign link to table OPERATOR.T.

• **CONSTR_TYPE** datatype: string (length: 10) no missing values
  Type of this constraint (see technical report).

• **CONSTR_OBJ1** datatype: string (length: 100) no missing values
  Object 1 for this constraint (see technical report).

• **CONSTR_OBJ2** datatype: string (length: 100)
  Object 2 for this constraint (see technical report).

• **CONSTR_DCU** datatype: string (length: 400)
  Free text to explain this constraint.

• **CONSTR_SQL** datatype: string (length: 1000)
  An SQL string for this constraint (see technical report).
2.24 OPERATOR_T
This table stores static information about the MiningMart operators.

- **OP_ID** datatype: integer, no missing values
  Unique M4 Id.

- **OP_NAME** datatype: string (length: 1000), no missing values
  Name of this operator. Note that the name must correspond exactly, respecting case, to the name of the Java class that implements this operator (see also MiningMart technical report TR12.4 “How to implement M4 operators”).

- **OP_LOOP** datatype: string (length: 5)
  One of 'YES' or 'NO'. Indicates whether this operator is loopable, which means that may be applied several times (ie in several loops) in one step, while some of its parameters are different for each loop.

- **OP_MULTI** datatype: string (length: 5)
  One of 'YES' or 'NO'. Indicates whether this operator is multistepable, which means that it can have more than one output Columnset.

- **OP_MANUAL** datatype: string (length: 5)
  One of 'YES' or 'NO'. Indicates whether this operator is manual, which means that it does not use an external algorithm.

- **OP_REALIZE** datatype: string (length: 100)
  Not used.

2.25 OP_PARAMS_T
This table is used to store parts of the static information about an operator, namely its parameters. Please refer to the MiningMart technical report TR18.1 “Representing Constraints, Conditions and Assertions in M4” (there section 2).

- **PARAM_ID** datatype: integer, no missing values
  Unique table Id.

- **OP_ID** datatype: integer, no missing values
  Foreign link to table OPERATOR_T.

- **MINARG** datatype: integer, no missing values
  Minimum number of times that this parameter may be specified for a given step. May be 0.

- **MAXARG** datatype: integer
  Maximum number of times that this parameter may be specified for a given step. NULL means no restriction.
• NAME datatype: string (length: 100), no missing values
  Name of this parameter.
• IO datatype: string (length: 5), no missing values
  One of 'IN' or 'OUT', depending on whether this parameter is an input
  or output parameter.
• TYPE datatype: string (length: 5), no missing values
  Type of this parameter. One of 'BA' (BaseAttribute), 'MCF' (Multi-
  ColumnFeature), 'FEA' (Feature, ie BaseAttribute or MultiColumnFea-
  ture), 'CON' (Concept), 'V' (Value), 'REL' (Relation) or 'FUNC' (Func-
  tion). See the technical report.
• DOCU datatype: string (length: 400)
  Free textual description for this parameter.

2.26 OP_TYPE_T
This table may be used to store information about the type hierarchy of opera-

tors.
• OPT_ID datatype: integer, no missing values
  Foreign link to table OPERATOR_T (section 2.24).
• OPT_TYPE datatype: string (length: 100), no missing values
  Type of this operator, for example 'MissingValues' or 'FeatureConstruc-
  tion' or 'FeatureSelection' etc.

2.27 PARAMETER_T
This table stores the input and output parameters for each step. The number,

names and types of parameters are statically stored for each operator in table

OP_PARAMS_T (section 2.25). This table contains the parameters for a con-

crete instance of an operator in a case chain. The MiningMart compiler uses the

information in OP_PARAMS_T to find all parameters for an operator in this

table.
• PAR_ID datatype: integer, no missing values
  Unique M4 Id.
• PAR_NAME datatype: string (length: 100), no missing values
  Name of this parameter.
• PAR_OBJID datatype: integer
  Link to one of the tables BASEATTRIB_T, MCFEATURE_T, VALUE_T,
  CONCEPT_T or RELATION_T. The Id of an M4 object in one of these
  tables is entered here.
- **PAR_OBJTYPE** datatype: string (length: 20), no missing values
  
  One of 'BA' (BaseAttribute), 'MCF' (MultiColumnFeature), 'CON' (Concept), 'V' (Value), or 'REL' (Relation).

- **PAR_OPID** datatype: integer, no missing values
  
  Foreign link to table OPERATOR_T (section 2.24). Specifies the operator this parameter belongs to.

- **PAR_TYPE** datatype: string (length: 10), no missing values
  
  One of 'IN' for input parameters or 'OUT' for output parameters.

- **PAR_NR** datatype: integer, (3) no missing values
  
  The argument position of this parameter.

- **PAR_STID** datatype: integer, no missing values
  
  Foreign link to table STEP_T (section 2.32). Specifies the step for which this parameter is valid (there may be several operators of the same type in a chain).

- **PAR_STLOOPNR** datatype: integer, (5)
  
  If this operator is loopable, the loop number for which this parameter is valid is entered here. For parameters that are valid for all loops (typically the input concept), either 0 or NULL can be entered here. For the others a loop number, starting with 1, must be entered. For non-loopable operators this field must be NULL.

### 2.28 **PROJECTION_T**

This table may be used to store projection relations between concepts. Concept A is a projection of Concept B iff its features are a subset of B’s features, but it contains the same (number of) rows.

- **PRO_ID** datatype: integer, no missing values
  
  Unique table Id.

- **PRO_FROMCONID** datatype: integer, no missing values
  
  Concept B (referring to above explanation). Foreign link to table CONCEPT_T (section 2.13).

- **PRO_TOCONID** datatype: integer, no missing values
  
  Concept A (referring to above explanation). Foreign link to table CONCEPT_T (section 2.13).
2.29 RELATION_T

This table is used to store relationships between Concepts. Relations are M4 objects. Both one-to-many (1:m) relations and many-to-many (n:m) relations are modelled with this table. The table stores the conceptual level as well as the relational level.

- **REL_ID** datatype: integer, no missing values
  Unique M4 Id.

- **REL_NAME** datatype: string (length: 100), no missing values
  Name of this Relation.

- **REL_FROMCONID** datatype: integer
  Foreign link to table CONCEPT_T (section 2.13). For a 1:m Relation, the Concept with the foreign key is stored here. For an n:m Relation, any of the two Concepts is stored here.

- **REL_TOCONID** datatype: integer
  Foreign link to table CONCEPT_T (section 2.13). For a 1:m Relation, the Concept with the primary key is stored here. For an n:m Relation, the other of the two Concepts is stored here.

- **REL_FROMKID** datatype: integer
  Foreign link to table KEYHEAD_T (section 2.18). For a 1:n Relation, the Keyhead modelling the foreign-key-link is stored here. For an n:m Relation, the Keyhead that models the link from the cross table to the Concept stored in field REL_FROMCONID is stored here.

- **REL_TOKID** datatype: integer
  Foreign link to table KEYHEAD_T (section 2.18). For a 1:n Relation, this field is NULL. For an n:m Relation, the Keyhead that models the link from the cross table to the Concept stored in field REL_TOCONID is stored here.

- **REL_CSID** datatype: integer
  Foreign link to table COLUMNSET_T (section 2.10). For a 1:n Relation, this field is NULL. For an n:m Relation, the Columnset that represents the cross table of this Relation is stored here.

- **REL_SUBRELRESTR** datatype: string (length: 4000)
  Not used.

- **REL_VALID** datatype: string (length: 5)
  One of 'YES' or 'NO'. Replace This String By An Explanation For All The Valid Fields.
2.30 RELATIONISA_T

This table may be used to store information about a hierarchy of Relations.

- **RISA_ID** datatype: integer, no missing values
  Unique table Id.
- **RISASUPERRELID** datatype: integer, no missing values
  Foreign link to table RELATION_T (section 2.29). Defines the superrelation.
- **RISASUBRELID** datatype: integer, no missing values
  Foreign link to table RELATION_T (section 2.29). Defines the subrelation.

2.31 ROLERESTRICTION_T

Not used.

- **RR_ID** datatype: integer, no missing values
- **RR_NAME** datatype: string (length: 100), no missing values
- **RR_RELID** datatype: integer
- **RRFROMCONID** datatype: integer
- **RRTOCONID** datatype: integer
- **RR_MIN** datatype: integer
- **RR_MAX** datatype: integer

2.32 STEP_T

This table holds the information about a Step in a Case. In particular it refers to the operator and to the case of each step.

- **ST_ID** datatype: integer, no missing values
  Unique M4 Id.
- **ST_NAME** datatype: string (length: 100), no missing values
  Name of this step.
- **ST_CAID** datatype: integer
  Foreign link to table CASE_T (section 2.4). Shows the case that this step belongs to.
- **ST_NR** datatype: integer, (5)
  Position of this step within its Chain. Not used by the MiningMart compiler.
• ST.OPID datatype: integer
  Foreign link to table OPERATOR.T (section 2.24). Stores the operator that is applied in this step.

• ST.LOOPNR datatype: integer, (5)
  Number of loops this operator must be applied in. For compatibility reasons, this field must contain 0 or NULL if the operator in this step is not applied in loops, ie it is only applied once. If there are loops, this field contains the number of loops which is at least 2. Thus the number 1 must never be entered here.

• ST_MULTISTEPCOND datatype: string (length: 1000)
  Not used.

• ST.CHID datatype: integer
  Foreign link to table CHAIN.T (section 2.7). Stores the chain to which this step belongs.

• ST.VALID datatype: string (length: 5)
  One of 'YES' or 'NO'. ReplaceThisStringByAnExplanationForAllTheValid-Fields.

2.33  STEPSEQUENCE_T

• STS.ID datatype: integer, no missing values
  Unique table Id.

• STS_STID datatype: integer, no missing values
  Foreign link to table STEP.T (section 2.32). Stores the step for which successors are entered in this table. Several entries for the same step are possible. A step without successors does not have to be entered here.

• STS_SUCCESSORSTID datatype: integer
  Foreign link to table STEP.T (section 2.32). Gives one successor step for the step in the field STS_STID. Must be NULL if there is no successor step.

2.34  VALUE.T

This table stores simple values that are needed as input for operators. For example, if an operator has a real value parameter, the actual number can be stored in this table as a Value object which can then be referenced from the parameter table PARAMETER.T.

• V.ID datatype: integer, no missing values
  Unique M4 Id.
• V_CONDTID datatype: integer, no missing values
  Conceptual datatype of this Value. Foreign link to table CON_DATA_TYPE_T (section 2.15).

• V_NAME datatype: string (length: 100)
  Name of this value object. Can be NULL as it only serves mnemotechnical purposes.

• V_VALUE datatype: string (length: 4000), no missing values
  The actual value, stored as a string.

3 Special Purpose M4 Runtime Tables

This section lists some tables that are used by the MiningMart system for control and administration purposes. The two trash tables DBTrash_T and M4Trash_T are used by the MiningMart compiler to store an index to the metadata and other data that it creates; this data is deleted before a step is compiled again. The table HCI_Coord_T is used by the HCI to store its own information about arbitrary M4 objects. Finally, the table M4Access_T is used to ensure that atmost one user works on a given case at the same time.

3.1 DBTRASH_T

Used by the M4 Compiler to store references to tables, views and functions that it creates.

• OBJTYPE datatype: datatype: string (length: length: 5), no missing values
  One of 'T', 'V' or 'F', for types table, view or function. Specifies the type of this database object.

• OBJNAME datatype: datatype: string (length: length: 100), no missing values
  Database name of this table, view or function.

• STEPID datatype: integer, no missing values
  Foreign link to table STEP_T (section 2.32). Stores the step during which this database object was created.

• SCHEMANAME datatype: datatype: string (length: length: 200)
  Name of the database schema where this database object lives.
3.2 **HCI_COORD_T**

This table is used by the HCI (the graphical user interface) to store information about graphical ordering of objects on the screen.

- **OBJ_ID** datatype: integer, no missing values
  M4 Id of the object for which information is stored here.

- **OBJ_NAME** datatype: string (length: 100), no missing values
  Name of this M4 object.

- **OBJ_TYPE** datatype: string (length: 25), no missing values
  One of 'CON' for concepts, 'ST' for steps or 'REL' for relations.

- **CONTEXT_ID** datatype: integer, no missing values
  M4 Id of a context object.

- **CONTEXT_NAME** datatype: string (length: 100), no missing values
  Name of the context object.

- **CONTEXT_TYPE** datatype: string (length: 25), no missing values
  Type of the context object. One of 'CON' for concepts or 'CH' for chains.

- **X** datatype: integer, no missing values
  X coordinate of this object.

- **Y** datatype: integer, no missing values
  Y coordinate of this object.

3.3 **M4_ACCESS_T**

This table is used to lock an M4 object, usually a Case, if one user is working on it.

- **OBJECT_ID** datatype: string (length: 100), no missing values
  M4 Id of the object to be locked.

- **OBJECT_TYPE** datatype: string (length: 25), no missing values
  Type of the object.

- **CLIENT_NAME** datatype: string (length: 50), no missing values
  Name of the object.

- **ACCESS_TYPE** datatype: string (length: 10), no missing values
  One of 'READ', 'WRITE'.
3.4 **M4TRASH_T**

Used by the M4 Compiler to store references to M4 objects that it creates.

- **M4ID** datatype: integer, no missing values
  M4 Id of the object which is referenced.

- **M4TABLE** datatype: string (length: 50), no missing values
  Name of the M4 table (“COLUMNSET_T”, “COLUMN_T” etc.) in which the object that is referenced is stored.

- **STEPID** datatype: integer, no missing values
  Foreign link to table STEP_T (section 2.32). Stores the step during which this M4 object was created.