Enabling End-User Datawarehouse Mining
Contract No. IST-1999-11993
Deliverable No. D7a

Description of the M4-Relational Metadata-Schema within the Database

Regina Zücker[1]

Swiss Life, CC/ITRD
Information Technology Research & Development
CH-8022 Zürich, Switzerland
regina.zuecker@swisslife.ch
http://research.swisslife.ch

June 22, 2001

Abstract

In this part a of the deliverable, the relational dataschema for storing all persistent data of the M4-MetaModel is described. At this milestone a first version of the user interface and the MD-Compiler (MetaData-Compiler) and the statistics use this dataschema.

Not the full functionality is implemented, like it is described in Deliverable 8&9. Following is missing: (1) loopable and multistepable operators, (2) execution of a whole operator chain and therefore also parallel execution of operators, (3) underlying ontology of concepts, especially sorting output concepts into the ontology. The necessary tables or attributes for this missing functionality are marked with (not used in this version). That means that some implementation ideas already exist and will be realized in the next workpackage WP7+.

For a better understanding of which attribute of the here described data schema stores which information of the M4-MetaModel of Deliverable 8&9 we have put a reference behind every attribute in brackets. The first reference is the class name, the second the attribute name or association name of M4.

Chapter 1

Relational Metadata-Schema

All tables described in this chapter are active used, so far by the user interface, the MD-Compiler and the statistics. These software components use the tables for reading and writing.

1.1 Common

Every table has an ID-attribute. To fill this attribute a sequence has to be used. Only one sequence (ALL_SQ) is available for all tables. That ensures an unique object-id over the whole metadata-schema. The sequence has to be used with ALL_SQ.NEXTVAL.

In the M4-MetaModel a reference exists from the conceptual to the implementational level and vice versa. In this M4-Relational Metadata-Schema only the reference from the conceptual to the implementational level exist, e.g. only from a Concept to the corresponding ColumnSet.

1.2 Table CASE_T

- CA_ID
  Unique table-id
- CA_NAME (Case - name)
  Name of the case.
- CA_MODE (Case - case mode)
  A case can have several application modes. Depending on the mode, the compiler will have different functionality. Following modes are allowed: DESIGN, TEST, FINAL.

  During DESIGN-mode the compiler will always recompile all operator steps until the next learning operator occurs. After applying this learning operator, the compiler will write the results as metadata into the metadata-schema and then stop applying the next operator. So the case designer can check the result after every operator. Also during DESIGN-mode the compiler will always start a learning operator with a data sample.

During TEST-mode the compiler will... (its not clear yet, if this mode is needed.)

During FINAL-mode the compiler will recompile the whole preprocessing chain without stop. So the case user has no possibility to interact with the system during this mode. If the preprocessing chain has a learning operator in between, the compiler will first start this learning operator with a data sample, write the results as metadata in the schema and then compile the according manual operator without stop.

- CA_POPULATION (Case - population)
  Is the object for the actual mining task. It must be an element of the objects defined as case input. (Not used in this version.)

- CA_OUTPUT (Case - caseOutput)
  Is an object of type CONCEPT which is the result of the actual mining task. (Not used in this version.)

1.3 Table CASEINPUT_T

(Not used in this version.)

- CAI_ID
  Unique table-id.

- CAI_CAID
  Id-reference to table CASE_T. Defines to which case this input attribute belongs to. Several input attributes can be defined.

- CAI_OBJID (Case - caseInput)
  Id-reference to any object of type CONCEPT, RELATION, BASEATTRIBUTE, MULTICOLUMNFEATURE or VALUE which defines the case input.

- CAI_OBJTYPE (Case - caseInput)
  Defines the actual type of the object of CAI_OBJID. Allowed values are CON (for Concept), REL (for Relation), BA (for BaseAttribute), MCF (for MultiColumnFeature) and V (for Value).

1.4 Table CASEATTRIB_T

(Not used in this version.)

- CAA_ID
  Unique table-id.
- CAA_CAID
  Id-reference to table CASE_T. Defines to which case this case attribute belongs to. Several case attributes can be defined.
- CAA_OBJID (Case - targetAttributes)
  Id-reference to any object of type BASEATTRIBUTE which belongs to the defined output concept in CA_OUTPUT and defines a target attribute of the actual mining task.
- CAA_OBJTYPE (Case - targetAttributes)
  Defines the actual type of the object of CAA_OBJID. Allowed value in this version is only BA (for BaseAttribute).

1.5 Table STEP_T

- ST_ID
  Unique table-id.
- ST_CAID (Case - listOfSteps, Step - belongsToCase)
  Id-reference to table CASE_T. Defines to which case this step belongs to. Several steps can be defined.
- ST_NR
  Defines the application order of a case during sequentiell execution. During parallel execution this order together with the STEPSEQUENCE will give the actual application order.
- ST_OPID (Step - embedsOperator)
  Id-reference to table OPERATOR_T. Defines the operator which belongs to this step. Only one operator is possible for one step.
- ST_LOOPNR (Step - iterationSet (only indirect reference))
  If the step is applied as loop, this attribute defines the order of the loop-sequence. During a loop the same operator is applied several times on different input sets (which are defined in PARAMETER_T). This order must match with the order of PAR_STLOOPNR.
- ST_MULTISTEPCOND (MultiStep - iterationCondition)
  If the step is applied as multistep, this attribute defines the condition.

During a multistep the same operator is applied on the several output-concepts of the previous operator. The condition defines, how and which attributes of these output concepts have to be transformed.

1.6 Table STEPSEQUENCE_T

(Not used in this version.)

- STS_ID
  Unique table-id.
- STS_STID
  Id-reference to table STEP_T. Defines to which step this sequence-definition belongs to. Several definitions can exist. Inserts within this table are only relevant during parallel execution. Then the application order is defined by predecessors and sucessors of a step.
- STS_PREDECESSOR (Step - predecessor)
  Id-reference to table STEP_T. Defines one predecessor of a step.
- STS_SUCESSOR (Step - successor)
  Id-reference to table STEP_T. Defines one sucessor of a step.

1.7 Table OPERATOR_T

- OP_ID
  Unique table-id.
- OP_NAME
  Name of the operator. In case of a feature-construction-operator, the actual name of the transformation function (existing database function!) must be written here.
- OP_OPNID (reference to all subclasses of Operator)
  Id-reference to table OP_NAME_T. Defines the name of the actual operator, which has to be applied.
- OP_LOOP (Operator - loopable)
  Flag, if the operator is loopable. Allowed values are YES, NO.
- OP_MULTI (Operator - loopable)
  Flag, if the operator is multistepable. Allowed values are YES, NO.
- OP_MANUAL (Operator - manual)
  Flag, if the operator is a manual or learning operator. Allowed values are YES (for manual), NO (for learning).

- OP_REALIZE (Operator - realizes)
  Attribute for execution information. (Not used in this version.)

1.8 Table OPCONSTRAINT_T

(Not used in this version.)

- OPCON_ID
  Unique table-id.
- OPCON_OPID
  Id-reference to table OPERATOR_T. Defines one or more constraints for an operator.
- OPCON_TYPE (indirect reference to associations of Operator)
  Defines the type of the operator constraint. Allowed values are ASS (for assertion), COND (for condition), CONSTR (for constraint).
- OPCON_VALUE (indirect reference to associations of Operator)
  Defines the actual value for the condition or constraint or assertion.

1.9 Table PARAMETER_T

- PAR_ID
  Unique table-id.
- PAR_NAME
  Name of the parameter.
- PAR_OBJID (Operator - input, Operator - output (indirect reference))
  Id-reference to any object of type CONCEPT, RELATION, BASEATTRIBUTE, MULTICOLUMNFEATURE or VALUE which defines a parameter for an operator or loopable operator. Several parameters can be defined.
- PAR_OBJTYPE (Operator - input, Operator - output (indirect reference))
  Defines the actual type of the object of PAR_OBJID. Allowed values are CON (for Concept), REL (for Relation), BA (for BaseAttribute), MCF (for MultiColumnFeature) and V (for Value).
- PAR_OPID
  Id-reference to OPERATOR_T. Defines the operator to which this parameter belongs to.

- PAR_TYPE (Operator - input, Operator - output (indirect reference))
  Defines if the parameter is of type input or output. Allowed values are IN (for input) and OUT (for output).
- PAR_NR
  Defines the order for this parameter within all parameters. In case of a feature-construction-operator the first parameter for the transformation function start with PAR_NR 3. All following input parameters also have to be input parameters of this transformation function.
- PAR_STID (Operator - loopable)
  Id-reference to STEP_T. Defines to which step this parameter belongs to. This attribute can only be filled, if a loop-step is defined. Then the attribute PAR_OPID has no relevance.
- PAR_STLOOPNR (Step - iterationSet)
  Number of input-set of a loop-step to which this parameter belongs to. This number must correspond with ST_LOOPNR.

1.10 Table VALUE_T

- V_ID
  Unique table-id.
- V_CONDTID (Value - domainDataType)
  Id-reference to table CON_DATATYPE_T. Defines the datatype on the conceptual level.
- V_VALUE
  Defines the value of this datatype. In case of a row-selection-operator the condition can be defined by the case designer as value. Then the case designer must use conceptual names instead of implementational names (e.g. 'PARTNER_AGE = 50' instead of 'PART = 50'). The compiler replaces these names by the implementational names.

1.11 Table CONCEPT_T

- CON_ID
  Unique table-id.
- CON_NAME (Concept - name)
  Name of the concept.
- CON_TYPE
  Type of the concept when considering the underlying concept-ontology.

Allowed values are BASE (for base level), DB (for database level), MINING (for mining level).

All concepts of type BASE have no corresponding column set (no insert in CON_CSID). During a case-transfer, all concepts of this type have to be transferred.

All concepts of type DB correspond to a column set which is a basic datatable, e.g. of a data warehouse. The connection between concept and column set has to be done by the case administrator. During a case-transfer, all concepts of this type have to be transferred.

All concepts of type MINING also have a corresponding column set. But this column set is generated by the MD-Compiler during case mode DESIGN or FINAL. During case-transfer concepts of this type won't be transferred but re-generated at the new location.

- CON_CSID (Concept - correspondsToColumnSet)
  Id-reference to COLUMNSET_T. Defines the corresponding column set at implementation level.

- CON_SUBCONRESTR (Concept - subConceptRestriction)
  Defines a sub-concept-restriction. (Not used in this version.)

1.12 Table CONCEPTISA_T

(Not used in this version.) Only operators of type row selection need an insert here.

- CISA_ID
  Unique table-id.

- CISA_SUPERCONID (Concept - isA)
  Id-reference to CONCEPT_T. Defines the super-concept of a concept. Only all super-concepts have to be defined.

- CISA_SUBCONID (Concept - isA)
  Id-reference to CONCEPT_T. Defines the actual concept (output concept of row selection) for which the super-concept has to be defined.

1.13 Table PROJECTION_T

(Not used in this version.) Only operators of type feature construction and multi-relational feature construction need an insert here.

- PRO_ID
  Unique table-id.

- PRO_FROMCONID (Concept - fromConcept)
  Id-reference to CONCEPT_T. Defines the from-concept-relation of a concept of type projection.

- PRO_TOCONID (Concept - toConcept)
  Id-reference to CONCEPT_T. Defines the actual concept or projection concept (output concept of feature construction or multi-relational feature construction) for which the from-concept has to be defined.

1.14 Table RELATION_T

- REL_ID
  Unique table-id.

- REL_NAME (Relationship - name)
  Name of the relation.

- REL_FROMCONID (Relationship - fromConcept)
  Id-reference to table CONCEPT_T. Defines the from-concept-direction of this relation. For a n:m relation, REL_FROMCONID and REL_TOCONID have to be filled. For a 1:m relation only one of these attributes is necessary.

- REL_TOCONID (Relationship - toConcept)
  Id-reference to table CONCEPT_T. Defines the to-concept-direction of this relation.

- REL_FROMKID (Relationship - correspondsToForeignKey)
  Id-reference to table KEYHEAD_T. Defines the from-key-direction at implementation level of this relation. For a n:m relation, REL_FROMKID, REL_TOKID and REL_CSID have to be filled. For a 1:m relation, only one key-reference is necessary. The defined REL_FROMCONID and REL_FROMKID must correspond at implementation level in such a way that for the corresponding column set of REL_FROMCONID a primary key is defined to which the defined foreign key of REL_FROMKID belongs to.

- REL_TOKID (Relationship - correspondsToForeignKey)
  Id-reference to table KEYHEAD_T. Defines the to-key-direction of this relation.

- REL_CSID (Relationship - correspondsToColumnSet)
  Id-reference to table COLUMNSET_T. Defines the corresponding column set at implementation level. It can only be filled for a n:m relation.

- REL_SUBRELRESTR (Relationship - subRelationshipRestriction)
  Defines a sub-relation-restriction. (Not used in this version.)

1.15 Table MM_RELATIONISA_T

(Not used in this version.)

- RISA_ID
  Unique table-id.
- RISA_SUPERRELID (Relationship - isA)
  Id-reference to RELATION_T. Defines the super-relation of a relation. Only all super-relations have to be defined.
- RISA_SUBRELID (Relationship - isA)
  Id-reference to RELATION_T. Defines the actual relation for which the super-relation has to be defined.

1.16 Table MCFEATURE_T

- MCF_ID
  Unique table-id.
- MCF_NAME (FeatureAttribute - name)
  Name of the multi-column feature.
- MCF_CONID
  Id-reference to CONCEPT_T. Defines the concept to which this multi-column feature belongs to.

1.17 Table BASEATTRIB_T

- BA_ID
  Unique table-id.
- BA_NAME (FeatureAttribute - name)
  Name of the base attribute.
- BA_COLID (FeatureAttribute - correspondsToColumns)
  Id-reference to COLUMN_T. Defines the corresponding column at implementation level.
- BA_CONDTID (BaseAttribute - domainDataType)
  Id-reference to table CON_DATATYPE_T. Defines the datatype on the conceptual level.

- BA_CONID (FeatureAttribute - belongsToConcept)
  Id-reference to table CONCEPT_T. Defines the concept to which this base attribute belongs to.

- BA_RELEVANCE (FeatureAttribute - relevanceForMining)
  Flag, if the base attribute is relevant for mining. Allowed values are YES, NO.

- BA_ATTRIBTYPE (FeatureAttribute - attributeType)
  Type information about how the attribute is created. Allowed values are BASE, DB, MINING.

- BA_MCFID (BaseAttribute - isPartOfMultiColumnFeature)
  Id-reference to table MCFEATURE_T. Defines to which multi-column feature this attribute belongs to. Several definitions for one multi-column feature can exist.

1.18 Table COLUMNSET_T

- CS_ID
  Unique table-id.

- CS_SCHEMA
  Database-schema where the column set belongs to.

- CS_NAME (ColumnSet - name)
  Name of the column set. This name is the actual name of an existing database table, if the column set is of type T, SN or MV. If it is of type V, the compiler decides, if this object will be an existing view object or just a sql-string.

- CS_FILE (ColumnSet - file)
  External filename for SQL-statement creating this column set. (Not used in this version.)

- CS_USER (ColumnSet - user)
  User-name who created this column set. (Not used in this version.)

- CS_CONNECT (ColumnSet - dbConnectString)
  Connect-String to the database. (Not used in this version.)

- CS_TYPE
  Type of physical database element for this column set. Allowed values are T (for table), V (for view), SN (for snapshot), MV (for materialized view).

- CS_SQL
  SQL-Create-String, generated by the MD-Compiler. Defines the create-statement for this column set.

1.19 Table CSSTATIST_T

- CSST_ID
  Unique table-id.
- CSST_CSID (ColumnSetStatistics - forColumnSet)
  Id-reference to table COLUMNSET_T. Defines to which column set this statistic information belongs to.
- CSST_ALL (ColumnSetStatistics - allNumber)
  Number of rows this column set has.
- CSST_ORD (ColumnSetStatistics - ordinalNumber)
  Number of columns of type NUMBER this column set has.
- CSST_NOM (ColumnSetStatistics - nominalNumber)
  Number of columns of type STRING this column set has.
- CSST_TIME (ColumnSetStatistics - timeNumber)
  Number of columns of type DATE this column set has.

1.20 Table COLUMN_T

- COL_ID
  Unique table-id.
- COL_NAME (Column - name)
  Name of the column.
- COL_CSID (Column - belongsToColumnSet)
  Id-reference to table COLUMNSET_T. Defines to which column set this column belongs to.
- COL_COLDTID (Column - dataType)
  Id-reference to table COL_DATATYPE_T. Defines the datatype of this column.
- COL_SQL
  SQL-String, generated by MD-Compiler. Defines the select-statement for this column. This attribute is only filled if it is the output attribute of an operator of type feature construction or multi-relational feature construction.

1.21 Table COLSTATIST1_T

This table contains basic statistic information for a column. Only one insert exists for one column.

- COLST1_ID
  Unique table-id.
- COLST1_COLID (ColumnStatistics - forColumn)
  Id-reference to table COLUMN_T. Defines to which column this basic statistic information belongs to.
- COLST1_UNIQUE (ColumnStatistics - unique)
  Number of different values of this column within the column set.
- COLST1_MISSING (ColumnStatistics - missing)
  Number of missing values of this column within the column set.
- COLST1_MIN (ColumnStatistics - min)
  Minimum-value of this column within the column set.
- COLST1_MAX (ColumnStatistics - max)
  Maximum-value of this column within the column set.
- COLST1_AVG (ColumnStatistics - average)
  Average-value of this column within the column set.
- COLST1_STDDEV (ColumnStatistics - standardDeviation)
  Standard-deviation-value of this column within the column set.
- COLST1_VARIANCE
  Variance-value of this column within the column set.

1.22 Table COLSTATIST2_T

This table contains distribution information for a column. Several inserts can exist for one column.

- COLST2_ID
  Unique table-id.
- COLST2_COLID (ColumnStatistics - forColumn)
  Id-reference to table COLUMN_T. Defines to which column this distribution information belongs to.
- COLST2_DISTVALUE (ColumnStatistics - distributionValue)
  Defines the value of the distribution element. For columns with type NUMBER, the average value of a distribution range is written here.

- COLST2_DISTCOUNT (ColumnStatistics - distributionCount)
  Number of elements within the column set which belong to this distribution element or distribution range.
- COLST2_DISTMIN (ColumnStatistics - distributionMin)
  Defines the minimum border of a distribution range in case of a column with type NUMBER.
- COLST2_DISTMAX (ColumnStatistics - distributionMax)
  Defines the maximum border of a distribution range in case of a column with type NUMBER.

1.23 Table KEYHEAD_T

- KH_ID
  Unique table-id.
- KH_NAME
  Name of the key.
- KH_PKCSID (Key - isAssociatedToColumnSet)
  Id-reference to table COLUMNSET_T. Defines the reference to an existing column set used for the primary key.
- KH_FKCSID (Key - isAssociatedToColumnSet)
  Id-reference to table COLUMNSET_T. Defines the reference to an existing column set used for the foreign key.

1.24 Table KEYMEMBER_T

- KM_ID
  Unique table-id.
- KM_KHID
  Id-reference to table KEYHEAD_T. Defines the keyhead to which this keymember belongs to.
- KM_PKCOLID (Key - hasColumn)
  Id-reference to table COLUMN_T. Defines the primary key column.
- KM_FKCOLID (Key - hasColumn)
  Id-reference to table COLUMN_T. Defines the foreign key column.
- KM_POS
  Defines the position of this keymember within the keyhead.

1.25 Table DOCU_T

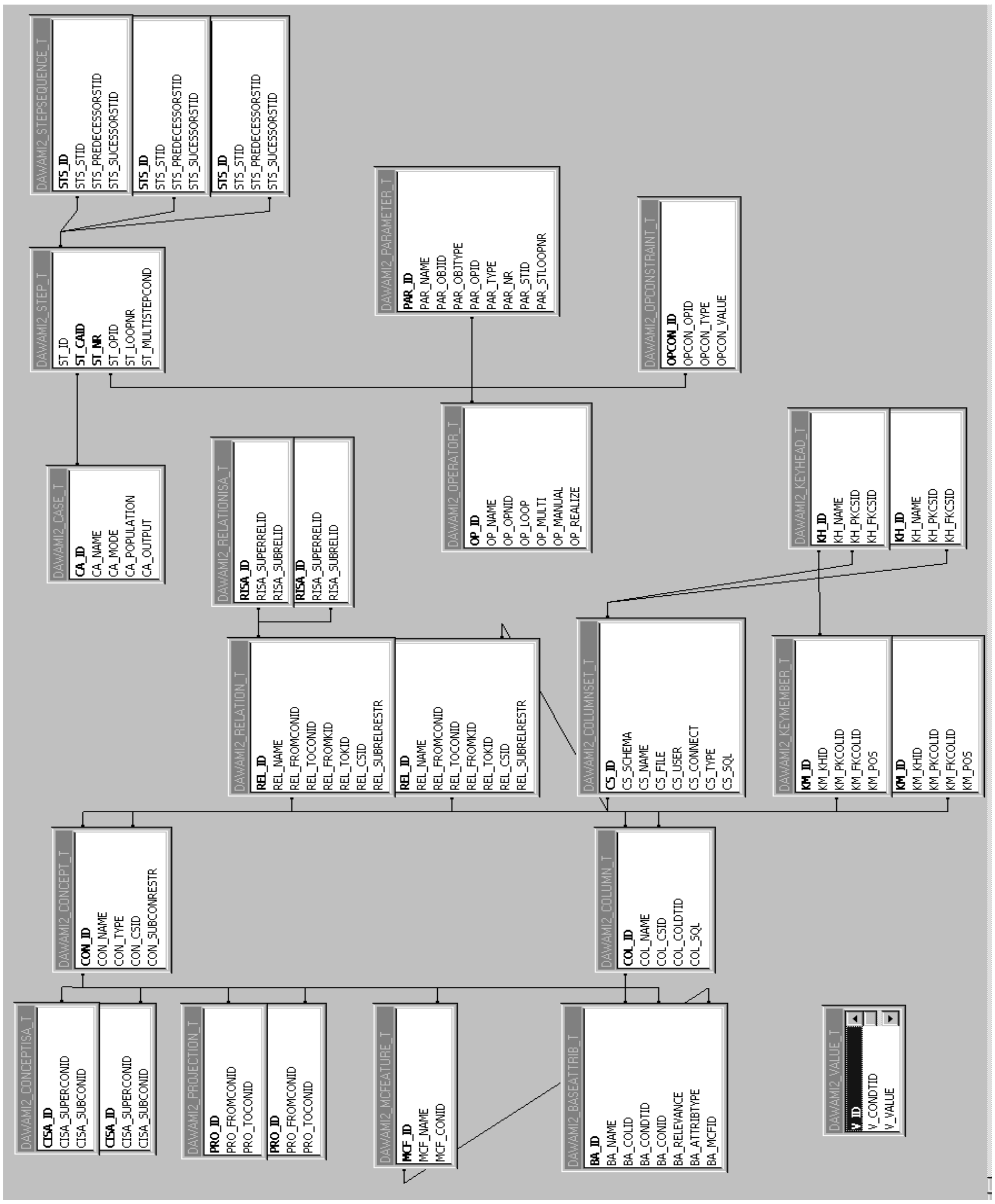
- DOC_ID
  Unique table-id.

- DOC_OBJID
  Id-reference to any object of type, e.g CASE, OPERATOR, PARAMETER, CONCEPT to which this documentation belongs to.

- DOC_OBJTYPE
  Defines the actual type of the object DOC_OBJID. Allowed are all prefixes of table-id's from all metadata tables.

- DOC_TEXT
  Actual text description.

1.26 Picture of the M4-Relational Metadata-Schema



Chapter 2

System-Tables for the Metadata-Schema

All tables described in this chapter are only used for reading. They content static information about system functionality.

2.1 Table CON_DATATYPE_T

- CONDT_ID
  Unique table-id.
- CONDT_NAME
  Name of datatype on the conceptual level, e.g. 'CATEGORIAL', 'KEYATTRIB' or 'TIMEGROUP'. To convert a concept-datatype to a implementation-datatype the database function COLUMN_DATATYPE is used.

2.2 Table COL_DATATYPE_T

- COLDT_ID
  Unique table-id.
- COLDT_NAME
  Name of datatype on the implementational level, 'NUMBER', 'STRING', 'DATE' and 'KEY'.

2.3 Table OP_TYPE_T

- OPT_ID
  Unique table-id.

- OPT_TYPE
  Name of the abstract operator type of the M4-MetaModel, e.g. 'FEATURE_CONSTRUCTION' or 'FEATURE_SELECTION'.

2.4 Table OP_NAME_T

- OPN_ID
  Unique table-id.
- OPN_TYPE
  Name of the concrete operator, for which executable code exists, e.g. 'RANDOM_SAMPLING' or 'SELECT_BY_QUERY'.

2.5 Table S_CASEMODE_T

- NAME
  Name of the case-mode, e.g. 'DESIGN', 'TEST' or 'FINAL'.

2.6 Table S_OBJTYPE_T

- NAME
  Name of the object type when referencing an object indirectly, for example at PAR_OBJTYPE. Values of this table are up to now 'CON', 'REL', 'BA', 'V' and 'MCF'.

2.7 Table S_YESNO_T

- NAME
  Boolean flag values 'YES' and 'NO.

2.8 Table S_INOUT_T

- NAME
  Values 'IN' and 'OUT'.

2.9 Table S_ONTOLOGY_T

- NAME
  Name of the ontology level for a concept, e.g. 'BASE', 'DB' or 'MINING'.

2.10 Table S_CSTYPE_T

- NAME
  Name of the columnset-type. This information is used by the MD-Compiler to create the corresponding database object. Values are up to now 'T' (for table), 'SN' (for snapshot), 'V' (for view) and 'MV' (for materialized view).