Conceptual Data Model : Concept, Formal Bases, and Implementation Issues*

Manuk G. Manukyan[0000−0002−8578−3440]

Yerevan State University, Yerevan 0025, Armenia,
mgm@ysu.am

Abstract. The investigation subjects of this paper are the implementation issues of a conceptual data model. An approach to implement the concept of the considered conceptual data model is proposed. The discussed conceptual data model concept is based on the behavioral and data definition symbols to model conceptual entities. The result of formalization of these symbols is content dictionaries. An important feature of the considered conceptual data model is its extensibility property. The extension is achieved by introducing new symbols into the conceptual data model. Thus, the result of extension of the considered conceptual data model is reduced to create new content dictionaries to support new concepts. The extensibility property provides the ability to conceptually model arbitrary data sources. The conceptual data model is defined as the union of all content dictionaries.

Keywords: Conceptual Data Model, Hierarchical Relation, Hierarchical Relations Algebra, Data Integration, Extensible Data Model, OPENMath.

1 Introduction

The emergence of a new paradigm in science and various applications of information technology is related to issues of big data handling. Big data is a field that treats of ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software. The concept of big data is relatively new and is based on the following widely spread notions: *data volume*, *velocity*, *variety*, *veracity* and *value* [1]. This concept involves the growing role of data in all areas of human activity beginning with research and ending with innovative developments in business. In this connection, the creation of new information technology is expected in which data becomes dominant for new approaches to conceptualization, organization, and implementation of systems to solve problems that were previously considered extremely hard or, in some cases, impossible to solve. In this context, the issues of conceptual modeling of big data become relevant.

* This work was supported by the RA MES State Committee of Science, in the frames of the research project No. 21T-1B326.

Conceptual data modeling has been the subject of intense research since the late 1970s. Prerequisites for such research is that database systems usually have limited knowledge about the *meaning* of the data stored in them [2]. In fact, they allow to manipulate data of certain simple types. Any more complex interpretation is left to the user. In this context, the issues of formal knowledge representation in the form of a set of concepts of some subject domain and relations between them are topical. Such representations are used for reasoning about entities of the subject domains, as well as for the domains description. Thus, conceptualization of subject domain assumes accessing and managing the data in terms of the conceptual entities. Some arguments in favor of creating a conceptual data model with extensibilty property are discussed below.

The new data management paradigms, as well as the many directions in which modern database systems are evolving, leads to the idea of developing a conceptual data model with extensibility property. In particular, the data integration concept and temporal databases concept are examples of very important directions in which modern database systems are evolving. Analysis of existing approaches to data integration can be found in [3]. Basically, these works are devoted to the problems of integrating homogeneous data sources (in the case of big data the data sources basically are heterogenous). Typically, an extended relational or object data model was used as the target data model. Details of temporal databases and current trends in their development can be found in [4].

The polyglot persistence approach is an another argument to develop a conceptual data model with extensibility property. This is explained by the fact that different databases are designed to solve different problems. Using a single database engine for all of the requirements usually leads to non-performant solutions; storing transactional data, caching session information, traversing the graph of customers and the products their friends bought are essientially different problems. Even in the RDBMS space, the requirements of an OLAP and OLTP systems are very different nonetheless, they are often forced into the same schema (for more details see [5]).

In this paper the formal bases of implementation of a conceptual data model concept are considered. The discussed conceptual data model was proposed in [6]. An important feature of the considered conceptual data model is its extensibility property. The extensibility property provides the ability to conceptually model arbitrary data sources. An informal example of extending the conceptual data model to support the concept of data integration is introduced. Also, the formal bases of the considered conceptual data model are discussed.

The paper is organized as follows: A conceptual data model and its formal bases are considered in Sections 2. An approach to implement the concept of the considered conceptual data model is proposed in Section 3. Related work is presented in Section 4. The conclusion is provided in Section 5.

2 Formal Bases

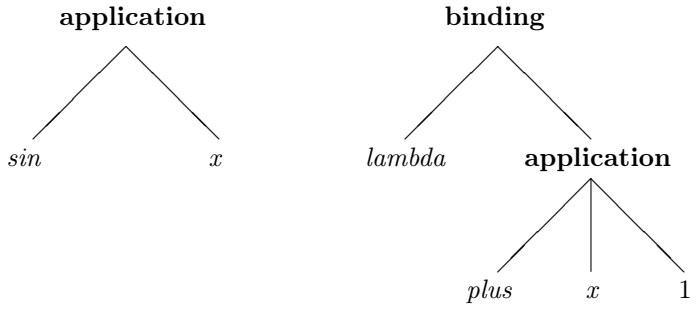
In this section we will briefly analyze formal bases of a conceptual data model. A more detailed analysis of these formalisms can be found in [6, 7].

2.1 OPENMath Objects

OPENMath is an extensible formalism for representing mathematical objects so that software packages can exchange these objects without losing semantic content. The possibility of the extensibility of the considered formalism is explained by the fact that the mathematical notation is constantly evolving. Moreover, mathematics and its applications are a growing field of knowledge, new ideas and notations appear constantly. Formally, an OPENMath object is a labeled tree whose leaves are basic OOPENMath objects. Examples of basic OPENMath objects are: Integer, Symbol and Variable. The compound objects are defined in terms of *binding* and *application* of the λ-calculus [8]. The following recursive rules for constructing compound OPENMath objects are proposed:

- Basic OPENMath objects are OPENMath objects.
- If $A_1, A_2, ..., A_n$ $(n \geq 1)$ are OPENMath objects, then *application*$(A_1,\ A_2, ..., A_n)$ is an OPENMath *application object*.
- If $S_1, S_2, ..., S_n$ are OPENMath symbols, and $A, A_1, A_2, ..., A_n$ $(n \geq 1)$ are OPENMath objects, then *attribution*$(A,\ S_1\ A_1,\ S_2\ A_2,\ ...,\ S_n\ A_n)$ is an OPENMath *attribution object* and A is the object *stripped* of *attributions*.
- If B and C are OPENMath objects, and $v_1, v_2, ..., v_n$ $(n \geq 0)$ are OPENMath variables or attributed variables, then *binding*$(B, v_1, v_2, ..., v_n,\ C)$ is an OPENMath *binding object*.

OPENMath objects have the expressive power to cover all areas of computational mathematics. The OPENMath application and binding objects for $sin(x)$ and $\lambda x.x + 1$ in tree-like notation are presented below:



2.2 Hierarchical Relations

In the frame of the considered conceptual data model (as in the relational data model), a single concept is used to model subject domains, namely, hierarchical relation. Below we introduce the definitions of the hierarchical relation schema

and the hierarchical relation. These definitions can be considered as strengthening the definitions of the relation schema and relation of the relational databases.

Definition 1. *A hierarchical relation schema X is an attribution object and is interpreted by a finite set of attribution objects $\{A_1, A_2, \ldots, A_n\}$. Corresponding to each attribution object A_i is a set D_i (a finite, non-empty set), $1 \leq i \leq n$, called the domain of A_i.*

Definition 2. *Let $D = D_1 \cup D_2 \cup \ldots \cup D_n$. A hierarchical relation x on hierarchical relation schema X is a finite set of mappings $\{t_1, t_2, \ldots, t_k\}$ from X to D with the restriction that for each mapping $t \in x$, $t[A_i]$ must be in D_i, $1 \leq i \leq n$. The mappings are called hierarchical tuples or simply tuples.*

A hierarchical relation is an instance of a hierarchical relation schema. The considered concept of hierarchical relations allows to model as XML as well as JSON data.

Definition 3. *A key of a hierarchical relation x on hierarchical relation schema X is a minimal subset K of X such that for any distinct tuples $t_1, t_2 \in x$, $t_1[K] \neq t_2[K]$.*

Finally, a database schema S is a finite set of schemas of the hierarchical relations. A database d on database schema S is a collection of hierarchical relations $\{x_1, x_2, \ldots, x_n\}$ such that for each schema of the hierarchical relation schema $s \in S$ there is a hierarchical relation $x \in d$ such that x is a hierarchical relation with schema s that satisfies every constraint defined in s.

3 Conceptual Data Model Concept Implementation

In this section we will consider the concept to model conceptual entities and its implementation issues.

3.1 Conceptual Schema

The conceptual schema is an instance of conceptual data model and is intended for formal knowledge representation in the form of a set of concepts of some subject domain and relations between them. Such representations are used for reasoning about entities of the subject domains, as well as for the domains description. A conceptual schema is defined as a set of OPENMath attribution objects. A distinguishing feature of the conceptual level is its stratification of the local and global levels to model the conceptual entities. On the local level, homogeneous representation of heterogeneous data sources is provided. The global level is intended to define derived conceptual entities. We consider the following formalisms to support conceptual entities:

- content dictionaries to define concepts (symbols) of conceptual data model (for example, type constructors, algebraic operations, etc.);

- signature files to formalize signatures of conceptual data model symbols to check the semantic validity of their representations.

A content dictionary (CD) which contains representation of symbols of the conceptual data model contains two types of information: one which is common to all CDs, and one which is restricted to a particular symbol definition. Definition of a new symbol includes name and description of the symbol, and also some optional information about this symbol. Below an example of a symbol definition is considered:

<CDDefinition>
<Name> *add* < /Name>
<Description> A n-ary commutative function addition < /Description>
<CMP> $x + y = y + x$ < /CMP>
< /CDDefinition>

The above used XML elements have obvious interpretations. Only note, that the element "CMP" contains the commented mathematical property of the defined arithmetic function symbol. Specific information pertaining to the symbol like the signature is defined in additional files associated with CDs. CDs contain just one part of the information that can be associated with a symbol in order to stepwise define its meaning and its functionality. Signature files are used to formalize formats of conceptual data model symbols. The considering symbols can be divided into two groups: data definition symbols and behavioral symbols. Functional notation is used below to formalize the symbols of the conceptual data model. CDs of the conceptual data model are based on these formal definitions.

3.2 Behavioral Symbols

To define the behavior of entities of the conceptual level, an algebra of hierarchical relations and its formal semantics was developed [6]. The proposed symbols are analog to the relational algebra operations. Here the functional notation is used to define the semantics of these symbols. Detailed definitions of the semantics of the considered operations can be found in [6]. Let r be the set of all hierarchical relations expressible within conceptual data model.

To support n-ary associative operations union, we introduced the symbol *union*. The symbol *union* is used to denote the n-ary union of sets (hierarchical relations). It takes sets as arguments, and denotes the set that contains all the tuples that occur in any of them:

$$union : r^{*assoc} \rightarrow r$$

To support operations minus, we introduced the symbol *minus*. The symbol *minus* is used to denote the difference of sets (hierarchical relations). It takes two sets as arguments, and returns a set that is the difference between two sets:

$$minus : r \times r \rightarrow r$$

To support n-ary associative operation joining, we introduced the symbol *join*. The symbol *join* is used to denote the n-ary join of sets. It takes sets as arguments, denotes a set of tuples, and is interpreted analogously to the operation natural join of the relational algebra in general case (joins of many relations):

$$join : r^{*assoc} \to r$$

To support a filtering operation, we introduced the symbol σ. This symbol is used to denote a select operation on the set. It takes a set and a predicate as arguments, and denotes the set which contains all the tuples for which the predicate is satisfied:

$$\sigma : \{r \to \{p : \{tuple\} \to boolean\}\} \to r$$

Here p is a predicate which is applied to *tuple*.

To support a projection operation, we introduced the symbol π. This symbol is used to denote a unary operation on the set. It takes a set and a list of *attribution* object names as argument, denotes a set of tuples, and is interpreted analogously to the operation *project* of the relational algebra:

$$\pi : r[name^*] \to r$$

Here *name* denotes the name of an attribution object.

For processing data, aggregating functions play a significant role. We introduced the *min*, *max*, *count*, *sum* and *avg* symbols to support the corresponding aggregate functions of the relational algebra. Let $f \in \{min, max, avg, sum, count\}$, then

$$f : r[name] \to numericalvalue \mid string$$

Often, we need to consider the tuples of a hierarchical relation in groups. For this purpose, we introduced a grouping symbol γ. This symbol is used to denote a unary operation on the set. It takes a set, a list of *attribution* object names and aggregate functions as arguments, denotes a set of tuples, and is interpreted analogously to the operation *grouping* of the relational algebra:

$$\gamma : r[name^*(, f : (tuple[name^*])^* \to numericalvalue \mid string)^*] \to r$$

3.3 Data Definition Symbols

To model the hierarchical relation concept we introduce the symbols *sequence* and *choice* which have analogous semantics as *sequence* and *choice* elements in the XML Schema language. The arguments of these functions are typed attribution objects, and the return value is a typed attribution object[1]. Let R be the set of all hierarchical relations schemas in the frame of conceptual data model and $f \in \{sequence, choice\}$, then

[1] The attribution construct is used to define conceptual entities.

$$f : R^* \to R$$

We introduced the symbol *card* for modeling a cardinality number concept:

$$card \in \{?, *, +\}$$

To model constraint concept of databases a *constraints* symbol is introduced, which value is an attribution object.

We introduced the symbol *key* to model the hierarchical relation key's concept, which value is a set of attribution objects names.

To support the built-in data types concept of the XML Schema language the corresponding symbols were introduced (for instance, *integer*, *string*, etc.).

The above considered symbols form the CD for the conceptual data model which is defined in the previous section. Our concept to creating conceptual data model assumes that this model must be extensible. The extensibility concept of the considered conceptual data model coincides with the analogous concept of OPENMath. In other words, extension of the conceptual data model is reduced to defining new symbols and formalyzing these symbols using the CD mechanism.

3.4 An informal Example of Extending the Conceptual Data Model

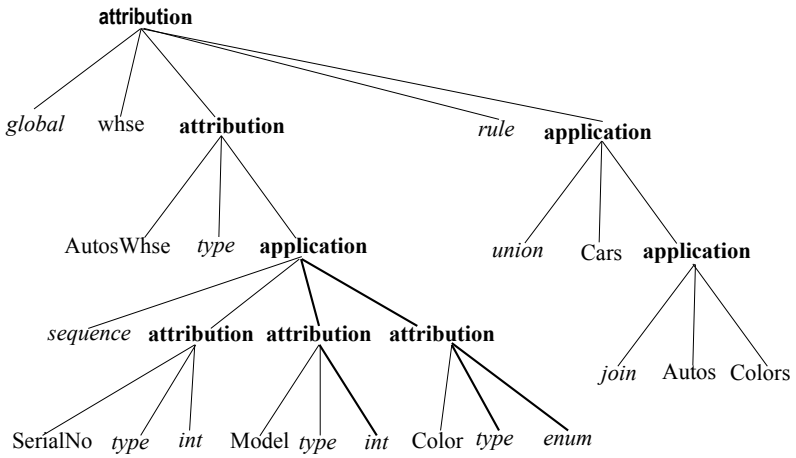
As mentioned above, the conceptual schema is stratified into two levels: local and global. To support this concept, the conceptual data model has been extended with *local* and *global* symbols. On the local level homogeneous representation of heterogeneous data sources is provided. In other words, data sources are represented by a set of hierarchical relations. Hierarchical relations of the local level are result data extraction from data sources. To support the data extraction concept (extract, transform, and load) the following symbols *etl*, *source* and *location* are introduced. These symbols are applying at the local conceptual level and have obvious semantics. Below, a formal example of local level attribution object is produced:

attribution(*local*, *source attribution*(Name, *type* applicationObject),
etl applicationObject, *location* LValue)

We have expanded the conceptual data model to support the data integration concept with the following data definition symbols: *med*, *whse*, *cube* and *rule* (they are nullary functions). These symbols applying on the global conceptual level. The values of the *whse*, *med* and *cube* symbols are attribution objects (the data warehouse, mediator, and data cube schemas, respectively) and the value of the *rule* symbol is an application object (an algebraic program) by means of which a mapping from data sources into data warehouse, mediator and data cube are defined. To define the entities of the global level the following construction is proposed:

attribution(*global*, *mwc attribution*(Name, *type* applicationObject),
rule algebraicProgram), where $mwc \in \{med, whse, cube\}$

The result of such extension of the conceptual data model is a new CD to model the data integration concept[2]. Below, an example of a data warehouse in tree-like notation for an automobile company database is adduced [10] which is an instance of the conceptual data model with orientation to data integration. Suppose for simplicity that there are only two dealers in the Aardvark system and they respectively use the schemas:
Cars = {serialNo, Model, Color}, and
Autos = {serialNo, Model}
Colors = {serialNo, Color}
It is assumed that a data warehouse is created with the schema
AutosWhse = {serialNo, Model, Color}



Examples of formalization of symbols by CD of the conceptual data model is given in Appendix A.

3.5 Signature Files for Conceptual Data Model

As is mentioned above, to check semantic validity of symbols representations we associate extra information with CDs, namely signature files. A signature file contains the definitions of all symbol signatures of the considered CD. We use Small Type System [9] to formalize the concept of signatures. Below the definition of the signature of the above considered symbol *add* is provided (see examples of the conceptual data model concept signature in Appendix B):

<Signature name = "add">

[2] Here we did not consider all the symbols that are used to support the data integration concept.

```
    <OMOB>
      <OMA>
        <OMS name = "mapsto" cd = "sts"/ >
        <OMA>
          <OMS name = "nassoc" cd = "sts"/ >
          <OMV name = "AbelianSemiGroup" cd = "sts"/ >
        < /OMA>
        <OMV name = "AbelianSemiGroup" cd = "sts"/ >
      < /OMA>
    < /OMOB>
< /Signature>
```

The above considered symbols *mapsto* and *nassoc* were defined in the OPENMath. The symbol *mapsto* represents the construction of a function type. The first n-1 children denote the types of the arguments, the last denotes the return type. The symbol *nassoc* constructs a child of *mapsto* which denotes an arbitrary number of copies of the argument of *nassoc*. The operator is associative on these arguments which means that repeated uses may be flattened/unflattened.

4 Related Work

In this Section we will consider popular notations for describing database designs, namely the E/R, UML and XML models [10]. Furthermore, a brief discussion of the data integration approach based on the conceptual data model is provided, along with known data integration approaches.

The most common conceptual data model is the E/R model in which two concepts (*entity set* and *relationship*) are used to model the subject domain. By means of this model, it is impossible to define the behavior of the conceptual entities. In addition, this model does not support means of extension.

UML offers much the same capabilities as the E/R model, with the exception of multiway relationships. Basic construction for modeling the subject domain is *class*. In contrast to the E/R model, it provides the ability to define the behavior of conceptual entities. There are three common extensibility mechanisms that are defined by the UML: stereotypes, tagged values, and constraints. The extensibility mechanisms allows to extend the UML by adding new building blocks, creating new properties, and specifying new semantics in order to make the language suitable for specific problem domain.

In the XML data model, basic construction to model the subject domain is *element*. The XML data model, like the E/R model, does not provide means to define the behavior of the conceptual entities. In contrast to the above considered data models, the XML data model supports means of extension. Extension is reduced to the creation of a new DTD. Due to this property, the use of the XML data model as the conceptual data model is preferred.

In [11], the methods and tools for equivalent data model mapping construction is proposed. The canonical data model is expanded axiomatically. In works

[12–16], relational data sources are considered in the frame of the traditional approach of the data integration as well as in the frame of the paradigm of ontology-based data access and integration. In [17] an analysis to use machine learning techniques to solve different problems to integrate unstructured and semistructured data is provided.

In our case, a single concept (*attribution*) is used to model the conceptual entities. Like the XML data model, the considered conceptual data model is extensible. As UML, it provides the ability to define the behavior of the conceptual entities. The extensibility property is an argument to use this conceptual data model as a canonical data model for integrating arbitrary data sources. We used the considered conceptual data model as a canonical data model for supporting the data integration concept [6]. Modeling means of the conceptual level are insensitive to the extension of the canonical data model. The conceptual level means are sufficient to model the data integration concepts proposed in the above considered works.

5 Conclusions

The investigation subjects of this paper are the implementation issues of a conceptual data model. The result of our research is an approach to implement the concept of the considered conceptual data model. A distinguishing feature of the discussed conceptual data model is its stratification into local and global levels to model the conceptual entities. On the local level, homogeneous representation of heterogeneous data sources as basic conceptual entities is provided. The global level is intended to define derived conceptual entities. A single concept is used to model subject domains, namely, hierarchical relation. The conceptual data model concept is based on the behavioral and data definition symbols to model conceptual entities. The considered behavioral symbols are analogues to relational algebra operations. The data definition symbols are used to support hierarchical relations and as a rule are nullary functions. The result of formalization of these symbols is CDs. An important feature of the considered conceptual data model is its extensibility property. The extension is achieved by introducing new symbols into the conceptual data model. Thus, the result of extension of the considered conceptual data model is reduced to create new CDs to support new concepts. The extensibility property provides the ability to conceptually model arbitrary data sources. A non-formal example of extending the conceptual data model is considered. The outcome of such extension is a new CD to support the data integration concept. The conceptual data model is defined as the union of all CDs. A computionally complete language is used for data modeling in the frame of the considered conceptual data model. As a development of this work, it is planned to construct an extension of the considered conceptual data model to support big data integration concept.

References

1. S. Sharma, U. S. Tim, J. Wong, S. Gadia, S Sharma.: A Brief Review on Leading Big Data Models. Data Science Journal **13**, pp. 138-157 (2014)
2. C. J. Date.: An Introduction to Database Systems. 8th edn. Addison-Wesley, USA (2004)
3. B. Golshan, A. Halevy, G. Mihaila, W. Tan.: Data integration: After the teenage years. In: Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems, pp. 101-106, (2017)
4. J. Gamper, M. Ceccarello, A. DignÃűs. WhatâĂŹs New in Temporal Databases?. In: Chiusano, S., Cerquitelli, T., Wrembel, R. (eds) Advances in Databases and Information Systems. LNCS, Springer, vol 13389, pp. 45-58 (2022)
5. P. J. Sadalage and M. Fowler.: NoSQL Distilled. 2nd edn. Addison-Wesley, USA (2013)
6. M. G. Manukyan.: On an Extensible Conceptual Data Model by a Non-formal Example. In: , et al. New Trends in Database and Information Systems. Communications in Computer and Information Science, Springer, vol 1652, pp. 3-13 (2022).
7. M. Dewar.: OpenMath: An Overview. ACM SIGSAM Bulletin **34**(2), pp. 2-5 (2000)
8. J. R. Hindley, J. P. Seldin.: Introduction to Combinators and λ-Calculus. Cambridge University Press, Great Britain (1986)
9. J. H. Davenport.: A small openmath type system. ACM SIGSAM Bulletin **34**(2), pp. 16-21 (2000)
10. H. Garcia-Molina, J. Ullman, J. Widom.: Database Systems: The Complete Book. 2nd edn. Prentice Hall, USA (2009)
11. L. A. Kalinichenko.: Methods and tools for equivalent data model mapping construction. In: F. Bancilhon, et al. (eds.) Advances in Database Technology-EDBT'90, Springer, pp. 92-119, Italy (1990).
12. D. Calvanese, G.De. Giacomo, D. Lembo, M. Lenzerini, R. Rosati.: Tractable reasoning and efficient query answering in description logics: The $DL-Lite$ family. JAR **39**(3), pp. 385-429 (2007)
13. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati: Ontology-based data access and integration. In: L. Liu, M. T. Özsu (eds.) Encyclopedia of Database Systems, Springer, pp. 1-7, USA (2017).
14. D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi.: Query Processing under GLAV Mappings for Relational and Graph Databases. In: Proceedings of the VLDB Endowment, **6**(2), pp. 61-72 (2012)
15. D. Calvanese, D. Lembo, G. De Giacomo, M. Lenzerini, A. Poggi, M. Rodriguez - Muro, R. Rosati, M. Ruzzi, D. F. Savo.: The mastro system for ontology-based data access. Semantic Web **2**(1), pp. 43-53 (2011)
16. M. Console, M. Lenzerini.: Data quality in ontology-based data access: The case of consistency. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2014 **28**(1), pp. 1020-1026 (2014)
17. X. L. Dong and T. Rekatsinas.: Data integration and machine learning: A natural synergy. In: Proceedings of the VLDB Endowment, **11**(12), pp. 2094-2097 (2018). https://doi.org/10.14778/3229863.3229876

A The *cdm* Content Dictionary File

<CD>

<CDName> cdm < /CDName>
<Description>
This CD defines the symbols of conceptual data model.
< /Description>

<CDDefinition>
<Name> R < /Name>
<Description>
This symbol represents the set of hierarchical relations schemas.
< /Description>
<CMP>$x \in R \Leftrightarrow attribution(x, type\ A, S_1\ A_1, S_2\ A_2, ..., S_k\ A_k), k \geq 0$< /CMP>
< /CDDefinition>

<CDDefinition>
<Name> $card$ < /Name>
<Description>
This symbol represents the cardinality number.
< /Description>
<CMP> $card \in \{?, *, +\}$ < /CMP>
< /CDDefinition>

<CDDefinition>
<Name> $sequence$< /Name>
<Description>
This symbol represents the sequence of the attribution objects.
< /Description>
<CMP> $R^* \rightarrow R$ < /CMP>
< /CDDefinition>

<CDDefinition>
<Name> r < /Name>
<Description>
This symbol represents the set of hierarchical relations.
< /Description>
<CMP> $R \rightarrow r$ < /CMP>
< /CDDefinition>

<CDDefinition>
<Name> $union$ < /Name>
<Description>
An n-ary associative union operation.
< /Description>
<CMP> $r^{*assoc} \rightarrow r$ < /CMP>
< /CDDefinition>

<CDDefinition>
<Name> $minus$ < /Name>
<Description>
A difference operation.
< /Description>
<CMP> $r \times r \rightarrow r$ < /CMP>
< /CDDefinition>

...
< /CD>

B The *cdm* Signature File

<CDSignatures type = "sts" cd = "*cdm*">

<! − − Definition of signature of the R symbol. This symbol defines a finite set of all hierarchical relations schemas in the frame of conceptual data model.− − >
<Signature name = "R">
<OMOB>
<OMV name = "Set"/ >
< /OMOB>
< /Signature>

<! − − Definition of signature of the *card* symbol. The value of this symbol is the cardinality number.− − >[3]
<Signature name = "card">
<OMOB>
<OMS name = "*attribution*" cd = "sts"/ >
< /OMOB>
< /Signature>

<! − − Definition of signature of the *sequence* symbol. This symbol defines the sequence of the attribution objects. − − >[4]
<Signature name = "sequence">
<OMOB>
<OMA>
<OMS name = "mapsto" cd = "sts"/ >
<OMA>
<OMS name = "nary" cd = "sts"/ >
<OMS name = "R" cd = "*cdm*"/ >
< /OMA>
<OMV name = "List" / >
< /OMA>
< /OMOB>
< /Signature>

<! − − Definition of signature of the r symbol. This symbol defines a finite set of all hierarchical relations expressible in the frame of conceptual data model.− − >
<Signature name = "r">
<OMOB>
<OMV name = "*Set*"/ >
< /OMOB>
< /Signature>

<!− − Definition of signature of the *union* function. An n-ary commutative function union.− − >
<Signature name = "*union*">
<OMOB>

[3] An *attribution* object consists of pairs of keys and values. The use of the symbol *attribution* in a signature indicates that the symbol is to be used as a key.
[4] The symbol *nary* constructs a child of *mapsto* which denotes an arbitrary number of copies of the argument of *nary*.

```
    <OMA>
      <OMS name = "mapsto" cd = "sts"/ >
      <OMA>
        <OMS name = "nassoc" cd = "sts"/ >
        <OMS name = "r" cd = "cdm"/ >
      < /OMA>
      <OMS name = "r" cd = "cdm"/ >
    < /OMA>
 < /OMOB>
< /Signature>

<! − − Definition of signature of the minus function. The value of this function is
the difference between two sets.−− >
<Signature name = "minus">
 <OMOB>
    <OMA>
      <OMS name = "mapsto" cd = "sts"/ >
        <OMS name = "r" cd = "cdm"/ >
        <OMS name = "r" cd = "cdm"/ >
        <OMS name = "r" cd = "cdm"/ >
    < /OMA>
 < /OMOB>
< /Signature>

...
< /CDSignatures>
```