# An Ontology Approach to Data Integration

Manuk Manukyan

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

**Abstract.** In the frame of an XML-oriented data model an ontology approach to data integration is considered. Three kinds mechanisms are used to formalize data integration concept: reasoning rules, content dictionaries and semantical constraints. We introduced the concept an integrable data as ontology object which has been formalized within so-called algebra of integrable data. The offered ontology is presented by reasoning rules which are based on the proposed mathematical model. Mappings from source data models into ontology are defined by means of algebraic programs. To support the conceptual entities an algorithm to generate mappings from relational data sources into ontology is developed.

**Keywords:** Data Integration, Data Warehouse, Mediator, Data Cube, Ontology, Reasoning Rules, Metamodel, XML, OPENMath.

## 1 Introduction

In this paper we will consider an approach to ontology-based data integration. Ontology in informatics is understood as a 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 [15]. Various metamodels have been developed to define ontology [15,22]. We consider an XML- oriented data model which is a result of strengthening the XML data model by means of the OPENMath concept [10] as a metametamodel to support an ontology approach to data integration. OPENMath is a formalism to represent mathematical concepts with their semantics and is implemented as an XML application. We have certain experience in OPENMath usage in our research to support databases with ontological dependencies [18].

The considered XML - oriented data model was developed in the frame of our research devoted to the problems of heterogeneous databases integration (for instance, see [16,17,20,21]). Within these works, an approach to virtual and materialized integration of data has been developed. In [16] the existence issues of reversible mapping of an arbitrary source data model into a target one (canonical model) were considered. The considered approach in [16] is based on L. A. Kalinichenko's method of commutative mapping of data models, who was one of the pioneers in the area of justifiable data models mapping for heterogeneous databases integration [14]. According to this method, the mapping of an arbitrary resource data model into the canonical model is reversible, if diagram of schemas mapping and diagram of operators mapping are commutative. To verify the principle of commutative mapping of data models, the concept of data model is formalized by means of the AMN formalism [1]. Finally, in the frame of our research to heterogeneous databases integration a new dynamic indexing structure for multidimensional data was developed to support data materialized integration [20]. The problems of supporting OLAP-queries were considered in [17, 21]. In this work, based on our previous investigations in the area of heterogeneous databases integration, an ontology approach to data integration has been proposed.

The paper is organized as follows: the formal bases of an ontology definition metamodel are considered briefly in Section 2. An algebra of integrable data and intepreatation of these data in the frame of the metamodel are discussed in Section 3. An ontology-based data integration concept and its mathematical model, and also formalization by metamodel are proposed in Section 4. Some problems of mappings generation from data sources into ontology are considered in Section 5. Related work is presented in Section 6. The conclusion is provided in Section 7.

## 2 A Formalism for Defining Metamodels

We consider the XML - oriented data model as a best solution to use as metamodels construction formalism after addressing some of the shortcomings. Choosing the XML data model as a metamodel construction formalism is explained with the fact that this model is some compromise between conventional and semi-structured DMs because in contrast to:

- semi-structured DM, the concept of database schema in the sense of conventional DMs is supported;
- conventional DMs hard schemas, there is possibility to define more flexible database schemas.

The weakness of XML data model is the absence of data types concept in conventional sense. To eliminate this shortcoming and to support ontological dependencies on the XML data model level, we expand the XML data model by means of the OPENMath concept. The result of such extension is a data model which coincides with XML data model and which was strengthened with computational and ontological constructs of OPENMath.

### 2.1 The OPENMath Concept

This section is based on works [17, 18]. OPENMath is an extensible formalism which allows to strengthen the XML data model with ontological and computational constructions and is oriented to represent semantic information on the mathematical objects. OPENMath objects are such representations of mathematical objects which assume an XML interpretation. Formally, an OPENMath object is a labeled tree whose leaves are basic OPENMath objects. The compound objects are defined in terms of *binding* and *application* of the $\lambda$-calculus [13]. A type system is built from basic types and certain recursive rules whereby compound types are built from simpler types. The basic types consist of the conventional atomic types (for example, *integer*, *string*, *boolean*, etc.). To build compound types the following type constructors are used:

• *Attribution.* If $v$ is a basic object variable and $t$ is a typed object, then **attribution**($v$, *type* $t$) is typed object. It denotes a variable with type $t$.

• *Abstraction.* If $v$ is a basic object variable and $t$, $A$ are typed objects, then **binding**(*lambda*, **attribution**($v$, *type* $t$), $A$) is a typed object.

• *Application.* If $F$ and $A$ are typed objects, then **application**($F, A$) is a typed object.

### 2.2 Semantic Level

OPENMath is implemented as an XML application. Its syntax is defined by syntactical rules of XML, its grammar is partially defined by its own DTD. Only syntactical validity of the OPENMath objects representation can be provided on the DTD level. To check semantics, in addition to general rules inherited by XML applications, the considered application defines new syntactical rules. This is achieved by means of introduction of *signature files* concept (semantical constraints), in which these rules are defined. Signature files contain the signatures of basic concepts defined in some content dictionary and are used to check the semantic validity of their representations. A content dictionary is the most important component of OPENMath concept on preservation of mathematical information.

## 3 Algebra of Integrable Data

In the frame of our approach to ontology-based data integration we are introducing the concept of integrable data as an entity of the conceptual level [19].

### 3.1 Formalization of Integrable Data

**Definition 1** *An integrable data 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 $D = D_1 \cup D_2 \cup \ldots \cup D_n$. An integrable data $x$ on integrable data schema $X$ is a finite set of mappings $\{e_1, e_2, \ldots, e_k\}$ from $X$ to $D$ with the restriction that for each mapping $e \in x$, $e[A_i]$ must be in $D_i$, $1 \leq i \leq n$. The mappings are called elements.*

**Definition 3** *A key of integrable data $x$ is a minimal subset $K$ of $X$ such that for any distinct elements $e_1, e_2 \in x$, $e_1[K] \neq e_2[K]$.*

We introduce a symbol $d$ to denote the set of all integrable data. It is assumed that the schema of each integrable data is a subset of the set of all attribution objects.

**Interpretation of Integrable Data** In the frame of the metamodel, the following interpretation of integrable data schema $X$ is proposed:

$attribution(X,\ type\ A)$, or

$attribution(X,\ type\ A,\ S_1\ A_1,\ S_2\ A_2, ...,\ S_k\ A_k), k \geq 1$

Here $type, S_1, S_2, ..., S_k$ are OPENMath symbols, and $X, A, A_1, A_2, ..., A_k$ are OPENMath objects. In this representation, $X$ is the name of the atttribution object, and $A$ represents its type (basic or compound). A compound type by a type constructor is defined, for example:

$application(sequence, A_1, A_2, ..., A_l), l \geq 1$

The symbol *sequence* is defined analogously as in XML Schema language, and $A_1, A_2, ..., A_l$ are attribution objects. An integrable data $x$ with schema $X$ is interpreted by means of a finite set of nested XML-elements. An instance $x$ of $X$ is constructed according to the following rules:

1. $attribution(X, type$ basic type$) \Rightarrow < x >$ value $< /x >$
2. $attribution(X, type\ application(typeOp,\ A_1, A_2, ..., A_m)) \Rightarrow$

   $< x >$ compound value $< /x >$

   Compound value is constructed according to type constructor,

   where $\forall i(\ 1 \leq i \leq m),\ A_i = attribution(X_i, type$ basic type$)$, or

   $A_i = attribution(X_i,\ type\ application(typeOp,\ B_1, B_2, ..., B_n))$,

   where $B_i$ are attribution objects, $1 \leq i \leq n$.
3. Repeat the above steps in all possible ways until no more conversion of the content of any nested XML-element in $x$.

Below an XML Schema element definition and its equivalent representation in the frame of the metamodel (see Fig. 1.) is considered:

```
< xs : element name = "MovieStar">
  < xs : complexType >
    < xs : sequence >
      < xs : element name = "Name" type = "xs : string">
      < xs : element name = "Birthday" type = "xs : date">
      < xs : element name = "Address" maxOccur = "2">
        < xs : complexType >
          < xs : sequence >
            < xs : element name = "Street" type = "xs : string">
            < xs : element name = "City" type = "xs : string">
          < /xs : sequence >
        < /xs : complexType >
      < xs : element >
    < /xs : sequence >
  < /xs : complexType >
< /xs : element >
```

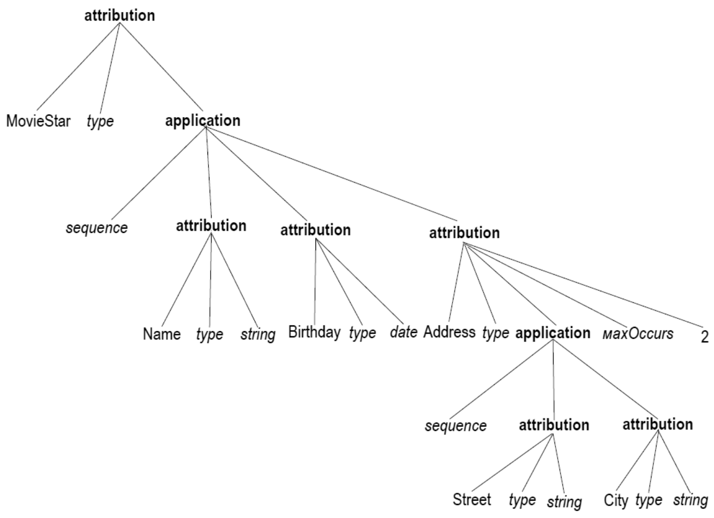

**Fig. 1.** From XML Schema to labeled tree: Transformation example.

### 3.2 Operations

Virtual and materialized integration of data assumes introduction of special operations, such as filtering, joining, aggregating, etc. The proposed operations are analogs of the corresponding operations in realtional algebra. Formal definitions of these operations were offered in [19].

## 4 Data Integration Ontology

Our approach to ontology-based data integration concept assumes formalizing the mediator, data warehouse and data cube concepts by an XML DTD. Formalization result of these concepts are so-called ontology reasoning rules. These rules will be interpreted by means of algebraic programs. Supporting reasoning rules assumes developing new content dictionaries to assign informal and formal semantics of data integration ontology basic concepts (for instance, integrable data, integrable data schema, algebraic operations, data types of the XML Schema). Formalization of basic concepts is achieved by applying the concept of OPENMath content dictionaries. Also we should formalize signatures of these basic concepts for checking semantic validity of its representations. In this case, we will be using the OPENMath signature files concept to formalize signatures of basic concepts.

### 4.1 Mathematical Model

In our case research object is the area of ontology-based data integration. We should formalize entities of this subject domain and define relationships between these mathematical objects. The proposed formalization will be the mathematical basis for constructing the reasoning rules. We differentiate three kind of reasoning rules: mediator rule, data warehouse rule and data cube rule. A formalization of the considered subject domain was proposed in [19].

## 4.2 Basic Concepts Formalization

The basic concepts such as integrable data, integrable data schema, algebraic operations, are mathematical concepts. Thus, it is natural to use the OPENMath content dictionaries to formalize these concepts. The content dictionaries are used to define semantical information on the basic concepts of data integration. A content dictionary which contains representation of basic concepts of the subject domain contains two types of information: one which is common to all content dictionaries, and one which is restricted to a particular basic concept definition. Definition of a new basic concept includes the name and description of the basic concept, and also some optional information about this concept. To support basic concepts of data integration and the type system of XML Schema, two content dictionaries have been developed. Below an example of a basic concept definition is considered:

<CDDefinition>
    <Name> X < /Name>
    <Description>
    To support the concept of integrable data schema we introduce
    the symbol $X$. Below we are using the *Attribution* symbol which has
    been defined in the OPENMath.
    < /Description>
    <CMP> $X : \mathit{Attribution}^* \to \{\mathit{Attribution}\}$ < /CMP>
< /CDDefinition>

The above used XML elements have obvious interpretations. Only note, that the element "CMP" contains the commented mathematical property of the defined algebraic concept. The detailed description of the *dic* content dictionary is given in Appendix A (analogously the *xts* content dictionary for modeling the type system concept of the XML Schema is defined). Content dictionaries contain just one part of the information that can be associated with a basic concept in order to stepwise define its meaning and its functionality. Specific information pertaining to the basic concepts like the signatures is defined separately in the so-called signature files.

## 4.3 Semantical Constraints

As is mentioned above, to check the semantic validity of the basic concepts representations, we associate extra information with content dictionaries in the form of signature files. A signature file contains the definitions of all basic concepts signatures of some content dictionary which is associated with this file. We use Small Type System [9] to formalize the basic concept signatures. Below the definition of the signature of the above considered symbol $X$ is provided (see the more detailed description of the *dic* signature in Appendix B):

```
<Signature name = “X”>
  <OMOB>
    <OMA>
      <OMS name = ”mapsto” cd = ”sts”/ >
      <OMA>
        <OMS name = ”nary” cd = ”sts”/ >
        <OMS name = ”attribution” cd = ”sts”/ >
      < /OMA>
      <OMS name = ”attribution” cd = ”sts”/ >
    < /OMA>
  < /OMOB>
< /Signature>
```

In the considered definition the symbols *mapsto* and *nary* 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 *nary* constructs a child of *mapsto* which denotes an arbitrary number of copies of the argument of *nary*.

## 4.4 Ontology as an XML Application

Based on the above discussed formalisms, an XML application to support conceptual schemas during ontology-based data integration is developed. Namely, an XML DTD was constructed based on the proposed mathematical relationships. The proposed XML DTD is an instance of the above considered metamodel which is an advanced XML data model and we use it as an ontology definition language. In contrast to OWL which is a description logic based ontology language for semantic Web, the considered metamodel is oriented to data integration and is based on an algebra of integrable data. Data integration concept is formalized by means of XML elements. Content of these elements are based on the OPENMath *attribution* and/or *application* concepts. Thus, it is possible to use a computationally complete language to support conceptual entities (integrable data). By means of XML elements reasoning rules (namely, mediator rule, data warehouse rule and data cube rule) were modeled based on the proposed mathematical relationshoips. Mappings from data sources into ontology are defined within reasoning rules and are modeled by means of algebraic programs. The ontology concept assumes a conceptual representation of the subject domain. Within this concept, accessing and managing the data is provided by notions of the conceptual representation. In connection with this, a problem arises to generate a query over source data based on the conceptual representation of the subject domain. In the next section, we will consider an algorithm to generate a query to relational source data. In the frame of the considered approach to ontology-based data integration, the conceptual schema is defined as an instance of the proposed XML DTD. In Appendix C an XML DTD for modeling the reasoning rules is presented.

## 5 Mappings Generation

Our concept to ontology-based data integration assumes constructing a mapping from arbitrary source data model into ontology. For proving the reversibility of the mapping from arbitrary source data model into ontology, it is sufficient to prove the existence of a reversible mapping from source data model into the extended metamodel. Thus, the extended metamodel acts as the target data model when applying the method of commutative mapping of data models of L. A. kalinichenko. A mapping generation from source data model into ontology assumes generating a query over data source and transforming these data into the ontology objects (set of integrable data). In the frame of this paper we consider some issues when constructing the mapping from relational source data into ontology. For proving the reversibility of mapping from the relational data model into the extended metamodel, these models have been formalized by means of AMN formalism and a mapping has been built from the relational data model to the extended metamodel. Finally, based on the B-technology, it has been proven that the mapping from the relational data model into the extended metamodel is reversible. Below, an algorithm to generate a query to extract data from relational data sources is proposed. The proposed algorithm is based on the in-order method of the tree traversal.

**Algorithm:** Relational Data Extractor.
**Input:** Global Schema (a labeled tree).
**Output:** SQL-query which is represented by string (further as a resulting string).
**Method:** Tree traversal. The following steps can be applied recursively to any *application* node. Initially, the resulting string is empty.

**1.** The considered operation is:

**a) union:** The generated query is defined as union of subqueries. The $i$-th subquery is constructed based on the $i$-th argument of this operation.

**b) minus:** The generated query is defined as difference of subqueries. The $i$-th subquery is constructed based on the $i$-th argument of this operation.

**c) join:** The following query is generated:
**select** resulting attributes
**from** subquery (is defined as join expression). The $i$-th operand is constructed based on the $i$-th argument of this operation.

**d) $\sigma$:** The following query is generated:
**select** *
**from** subquery (is constructed based on the first argument of this operation)
**where** predicate is constructed based on the second argument of this operation.

**e) $\pi$:** The following query is generated:
**select** resulting attributes
**from** subquery (is constructed based on the first argument of this

operation)

**f)** $\gamma$**:** The following query is generated:
**select** resulting attributes and aggregate function (s)
**from** subquery (is constructed based on the first argument of this operation)
**group by** list of grouping attributes (is constructed based on the second argument of this operation).

**2.** If in the step **1**, the argument on which the subquery is based is a leaf node, then:

**a)** In the case of the union/minus operation, the following query is generated:
**select***
**from** leaf_name

**b)** In the case of the operations **join**, $\sigma$, $\pi$ and $\gamma$, subquery construction is defined as leaf_name.

**3.** The generated query is substituted in the corresponding position of the resulting string based on the semantic of the algebraic program. If all the *application* nodes were considered, then return resulting string as the answer.

**4.** If in the step **1**, the argument on which the subquery is based is a non-leaf node (a subtree), then apply this algorithm recursively to this subtree.

We are using the proposed algorithm to support the data warehouse and data cube concepts. In the mediator case, this algorithm should be modified. The detailed discussion of problems to support queries over the mediator is beyond the topic of this paper.

## 6 Related Work

Since the end of the last century, the problems of creating "ontologically based" data access systems have been the subject of investigations in the field of databases and information systems. In [7] the role of ontologies in data integration is discussed. Particularly, the different aspects to use ontology in data integration are considered (such as in metadata representation, global conceptualization, high-level querying, declarative mediation, and mapping support). Ontology definition metamodels (for instance RDF, OWL, etc.) have been developed [22]. There are the following variants of data integration based on the ontology concept [7,11]: **Single ontology approach** relies on a single global ontology that provides a uniform interface to the user. **Multiple ontology approach** assumes defining for each data source its own local ontology and semantic mapping between these local ontologies. **Hybrid-ontology approach** combines the two preceding approaches. An example of this approach is the work [8]. A significant contribution to the theory and practice of data integration was made by the research group of M. Lenzerini (for instance, see [2–6]). Their investigations were carried out 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 these investigations only relational data as source data are considered. To define ontology as well as mappings between ontology and data sources, the description logic is used. Finally, in the frame of these investigations, a formal approach to data quality is proposed. Namely, one of the most important dimensions (consistency) of data quality is considered. The following papers [25, 26] can be considered as some development of the works of M. Lenzerini group. As a query language on the ontology level SPARQL is proposed. SPARQL-program is translated into efficient (federated) SQL-program over data sources based on the proposed optimisation techniques. In [8] a layered framework for the integration of heterogeneous networked data sources (e.g., relational, XML, or RDF) is proposed. Within this approach, a global ontology is used to mediate among the schemas of the data sources. A query is expressed in terms of one of the data sources or of the global ontology and is then translated into subqueries on the other data sources using mappings based on a common vocabulary. In [23] a system to automatically generate direct mappings between relational databases and given target ontologies is developed. The considered system is based on an intermediate internal graph representation that allows the representation of both factual knowledge and heuristically observed patterns from the input. A more detailed analysis of approaches to data integration in traditional sense and ontology-based data integration can be found in [12, 15, 24].

## 7 Conclusions

In the frame of an approach to ontology-based data integration, an ontology definition metamodel is proposed. The considered metamodel is oriented to XML data model which has been extended by the OPENMath concept. In the result of such extension, the XML data model has been strengthened with ontological and computational constructions. We introduced the concept of an integrable data as an ontology object which has been formalized within so-called integrable data algebra developed by us. The considered algebraic operations are analogs of the correpronding operations in relational algebra. The interpretation of the integrable data concept in the frame of the metamodel has been proposed. A mathematical model of the suggested concept of ontology-based data integration is constructed. The concepts of the integrable data algebra have been formalized by mechanisms of content dictionary and signature files (semantical constraints) of the OPENMath. The proposed ontology for data integration concept is presented by reasoning rules which are based on the considered mathematical model. The offered XML DTD is an instance of the considered metamodel. Thus, the ontology-based data integration concept formalization result is an ontological modeling language which is defined as an XML application. Mappings from source data models into ontology are defined by means of algebraic programs. To support the conceptual entities, an algorithm to generate mappings from relational data sources into ontology is developed. The output of this algorithm is an SQL-program by means of which we can extract data from relational sources to support the concepts of data warehouse and data cube. It is essential, that the metamodel is extensible, which allows to integrate arbitrary data models by using a computationally complete language.

**Acknowledgments.** This work was supported by the RA MES State Committee of Science, in the frames of the research project No. 18T-1B341.

## References

1. Abrial, J.R.: The B-Book-Assigning programs to meaning. Cambridge University Press, Great Britain (1996)
2. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The $DL-Lite$ family. JAR **39**(3), 385–429 (2007)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Ontology-based data access and integration. In: Encyclopedia of Database Systems. pp. 1–7 (2017)
4. Calvanese, D., Lembo, D., Giacomo, G.D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semantic Web **2**(1), 43–53 (2011)
5. Calvanese, D., andM. Lenzerini, G.D.G., Vardi, M.Y.: Query processing under GLAV mappings for relational and graph databases. In: PVLDB. pp. 61–72 (2012)
6. Console, M., Lenzerini, M.: Data quality in ontology-based data access: The case of consistency. In: Twenty-Eighth AAAI Conference on Artificial Intelligence. pp. 1020–1026. AAAI 2014 (2014)
7. Cruz, I.F., Xiao, H.: The role of ontologies in data integration. International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications **14**(4), 1–18 (2005)
8. Cruz, I.F., Xiao, H.: Ontology driven data integration in heterogeneous networks. In: Studies in Computational Intelligence. vol. 168, pp. 75–98. Springer (2009)
9. Davenport, J.H.: A small openmath type system. ACM SIGSAM Bulletin **34**(2), 16–21 (2000)
10. Drawar, M.: OpenMath: An overview. ACM SIGSAM Bulletin **34**(2), 2–5 (2000)
11. Ekaputra, F.J., Sabou, M., Serral, E., Kiesling, E., Biffl, S.: Ontology-based integration in multi-disciplinary engineering environments: A review. Open Joyrnal of Information Systems **4**(1), 1–26 (2017)
12. Golshan, B., Halevy, A., Mihaila, G., Tan, W.: Data integration: After the teenage years. In: PODS'17. pp. 101–106 (2017)
13. Hindley, J.R., Seldin, J.P.: Introduction to Combinators and $\lambda$-Calculus. Cambridge University Press, Great Britain (1986)
14. Kalinichenko, L.A.: Methods and tools for equivalent data model mapping construction. In: Advances in Database Technology-EDBT'90. pp. 92–119. Italy, Springer (March 1990)

15. Kalinichenko, L.A.: Effective support of databases with ontological dependencies: Relational languages instead of description logics. Programmirovanie **38**(6), 315–326 (2012)
16. Manukyan, M.G.: Canonical model: Construction principles. In: iiWAS2014. pp. 320–329. Vietnam, ACM (December 2014)
17. Manukyan, M.G.: On an approach to data integration: Concept, formal foundations and data model. In: CEUR-WS. vol. 2022, pp. 206–213 (2017)
18. Manukyan, M.G.: On an ontological modeling language by a non-formal example. In: CEUR-WS. vol. 2277, pp. 41–48 (2018)
19. Manukyan, M.G.: Ontology-based data integration. In: CEUR-WS. vol. 2523, pp. 117–128 (2019)
20. Manukyan, M.G., Georgyan, G.R.: A dynamic indexing scheme for multidimensional data. Modern Information Technologies and IT-Education **14**(1), 111–125 (2018)
21. Manukyan, M.G., Gevorgyan, G.R.: Canonical data model for data warehouse. In: Communications in Computer and Information Science. vol. 637, pp. 72–79 (2016)
22. OMG: Ontology definition metamodel. In: OMG Specification (2014)
23. Pinkel, C., Binnig, C., Jimenez-Ruiz, E., Kharlamov, E., Nikolov, A., Schwarte, A., Heupel, C., Kraska, T.: IncMap: A journey towards ontology-based integration. In: BTW 2017. pp. 145–164. Lecture Notes in Informatics (2017)
24. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: IJCAI-18. pp. 5511–5519 (2018)
25. Xiao, G., Hovland, D., Bilidas, D., Rezk, M., Giese, M., Calvanese, D.: Efficient ontology-based data integration with canonical IRIs. In: ESWC 2018. pp. 697–713 (2018)
26. Xiao, G., Kontchakov, R., Cogrel, B., Calvanese, D., Botoeva, E.: Efficient handling of SPARQL OPTIONAL for OBDA. In: ISWC 2018. pp. 354–373. Springer (2018)

## A The *dic* Content Dictionary File

```
<CD>
<CDName> dic < /CDName>
<Description>
This CD defines the symbols of Algebra of Integrable Data.
< /Description>
<CDDefinition>
    <Name> x < /Name>
    <Description>
    To support the concept of integrable data we introduce the symbol
    x.
    < /Description>
    <CMP> x : X → adom* < /CMP>
< /CDDefinition>
<CDDefinition>
```

    <Name> adom < /Name>
    <Description>
    To support the concept of a domain of attribution object we introduce the symbol *adom*. Below we are using the *Set* symbol which has been defined in the OPENMath and denotes a set.
    < /Description>
    <CMP> $adom : Set^* \to Set$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name> d < /Name>
    <Description>
    We introduced the symbol $d$ to denote the set of all integrable data.
    < /Description>
    <CMP> $d : x^* \to Set$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name> union < /Name>
    <Description>
    An n-ary associative union operation.
    < /Description>
    <CMP> $union : x^{*assoc} \to d$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name> $\sigma$ < /Name>
    <Description>
    This symbol denoting the filtering operation.
    < /Description>
    <CMP> $\sigma : \{x \to \{p : \{element\} \to boolean\}\} \to d$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name> element < /Name>
    <Description> This symbol denoting the element. < /Description>
    <CMP> $element : X \to D$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name> $\pi$ < /Name>
    <Description>
    This symbol denoting the projection operation.
    < /Description>
    <CMP> $\pi : x[name^*] \to d$ < /CMP>
< /CDDefinition>

<CDDefinition>
    <Name>$f$ < /Name>
    <Description>
    This symbol denoting the aggregate function : $f \in \{count, sum, avg\}$.
    < /Description>
    <CMP> $f : x[name] \to numericalvalue$ < /CMP>
< /CDDefinition>

```
<CDDefinition>
    <Name>γ < /Name>
    <Description>
    This symbol denoting the grouping operation.
    < /Description>
    <CMP> $\gamma : x[name^*, (f : (element[name^*])^* \rightarrow$
                  $numericalvalue)^*] \rightarrow d$
    < /CMP>
< /CDDefinition>
< /CD>
```

## B The *dic* Signature File

```
<CDSignatures type = "sts" cd = "dic">
   ...
<! − − Definition of signature of the x symbol. −− >
<Signature name = "x">
    <OMOB>
       <OMA>
         <OMS name = "mapsto" cd = "sts"/ >
         <OMS name = "X" cd = "dic"/ >
         <OMA>
           <OMS name = "nary" cd = "sts"/ >
           <OMS name = "adom" cd = "dic"/ >
         < /OMA>
       < /OMA>
    < /OMOB>
< /Signature>
<! − − Definition of signature of the d symbol. −− >
<Signature name = "d">
    <OMOB>
       <OMA>
         <OMS name = "mapsto" cd = "sts"/ >
         <OMA>
           <OMS name = "nary" cd = "sts"/ >
           <OMS name = "x" cd = "dic"/ >
         < /OMA>
         <OMS name = "Set" cd = "sts"/ >
       < /OMA>
    < /OMOB>
< /Signature>
<! − − Definition of signature of the union function. −− >
<Signature name = "Union">
    <OMOB>
       <OMA>
         <OMS name = "mapsto" cd = "sts"/ >
         <OMA>
           <OMS name = "nassoc" cd = "sts"/ >
           <OMS name = "d" cd = "dic"/ >
```

```
      < /OMA>
      <OMS name = ”d” cd = ”dic”/ >
    < /OMA>
  < /OMOB>
< /Signature>
<! − − Definition of signature of the filtering function. −− >
<Signature name = “σ”>
  <OMOB>
    <OMA>
      <OMS name = ”mapsto” cd = ”sts”/ >
      <OMS name = ”d” cd = ”dic”/ >
      <OMA>
        <OMS name = ”mapsto” cd = ”sts”/ >
        <OMS name = ”element” cd = ”dic”/ >
        <OMS name = ”boolean” cd = ”xts”/ >
      < /OMA>
      <OMS name = ”d” cd = ”dic”/ >
    < /OMA>
  < /OMOB>
< /Signature>
...
< /CDSignatures>
```

## C An XML DTD for Modeling the Reasoning Rules

```
<! − − include dtd for extended OPENManth objects −− >
<!ELEMENT dir (source+, (med | whse | cube))>
<!ELEMENT med (msch)+>
<!ELEMENT msch (sch, wrapper)>
<!ELEMENT sch (OMATTR)>
<!ELEMENT wrapper (OMA)>
<!ELEMENT whse (wsch, extractor)>
<!ELEMENT wsch (OMATTR)>
<!ELEMENT extractor (OMA)>
<!ELEMENT cube (ssch, mview)>
<!ELEMENT ssch (OMATTR)+ >
<!ELEMENT mview (view+, granularity+) >
<!ELEMENT view (OMA)>
<!ELEMENT granularity (partition)+ >
<!ELEMENT partition EMPTY>
<!ELEMENT source (OMATTR)+>
<!ATTLIST source name CDATA #REQUIRED>
<!ATTLIST granularity name CDATA #REQUIRED>
<!ATTLIST partition name CDATA #REQUIRED>
<!ATTLIST view name CDATA #REQUIRED>
```