Ontology-based Data Integration

Manuk Manukyan

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

Abstract. The data integration concept formalization issues have been considered within an XML-oriented data model. An ontology for data integration concept is proposed. Three kinds mechanisms are used to formalize the data integration concept: content dictionary, signature file and reasoning file (collections of reasoning rules). The reasoning rules are based on an algebra of integrable data and formalized by an XML DTD. The data translation mechanisms are non-sensitive to extension of the considered algebra. It is important that the considered data model is extensible and we use a computationally complete language to support the data integration concept.

Keywords: Data Integration, Data Warehouse, Mediator, Data Cube, Ontological Modeling, XML, OPENMath.

1 Introduction

We have published a number of papers that are devoted to investigation of data integration problems (for instance, see [12, 13, 15, 16]). Within of these works an approach to virtual and materialized integration of data has been developed. In [12] we considered the existence issues of reversible mapping of an arbitrary source data model into a target data model. The considered approach in [12] is based on the method of commutative mapping of data models of L. A. Kalinichenko [10]. According to this method, each data model is defined by syntax and semantics of two languages, data definition language (DDL) and data manipulation language (DML). The main principle of mapping of an arbitrary resource data model into the target one could be reached under the condition, that the diagram of DDL (schemas) mapping and the diagram of DML (operators) mapping are commutative. A new dynamic indexing structure for multidimensional data has been developed in [15] to support data materialized integration. The problems to support OLAP-queries are considered in [13, 16].

In this paper we will consider an approach to ontology-based data integration. An ontology is a formal, explicit specification of a conceptualization of a shared knowledge domain. In other words, ontologies offer means to represent high level concepts, their properties, and their interrelationships. Such representations are used for reasoning about entities of the subject domains, as well as for the domains description. In the frame of our approach to ontology-based data integration we have developed an XML-oriented data model by strengthening the

XML data model by means of the OPENMath concept [5]. OPENMath is a standard to represent mathematical concepts with their semantics on the Web. Usage of OPENMath concept allows to extend the XML language with computational and ontological constructs. We have certain experience in OPENMath usage in our research in this context (for instance see [14]. The proposed ontology is based on the OPENMath formalism and the so-called algebra of integrated data which also has been developed by us. Three kinds of mechanisms of the OPENMath are used to formalize the data integration concept: content dictionary, signature file and reasoning file (collections of reasoning rules). The reasoning rules are based on the algebra of integrable data and formalized by an XML DTD. It is essential that the considered data model is extensible and we use a computationally complete language to support the data integration concept.

The paper is organized as follows: the formal bases of the data integration concept formalization are considered briefly in Section 2. An algebra of integrable data and an ontology for data integration concept are proposed in Section 3 and Section 4 correspondingly. Related work is presented in Section 5. The conclusion is provided in Section 6.

2 Formal Bases

In this section we will briefly consider the OPENMath concept. Namely, the formalism and the constructions on which this concept is based. OPENMath is an extensible formalism and we use it to formalize the ontology-based data integration concept. This Section is based on the following works [13, 14].

2.1 The OPENMath Concept

OpenMath is a standard for representation of the mathematical objects, allowing them to be exchanged between computer programs, stored in databases, or published on the Web. The considered formalism is oriented to represent semantic information and is not intended to be used directly for presentation. Any mathematical concept or fact is an example of mathematical object. OpenMath objects are such representation of mathematical objects which assumes 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 λ-calculus [9]. The type system is built on the basis of types that are defined by themselves and certain recursive rules, whereby the compound types are built from simpler types. The basis consists 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 typed object.

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

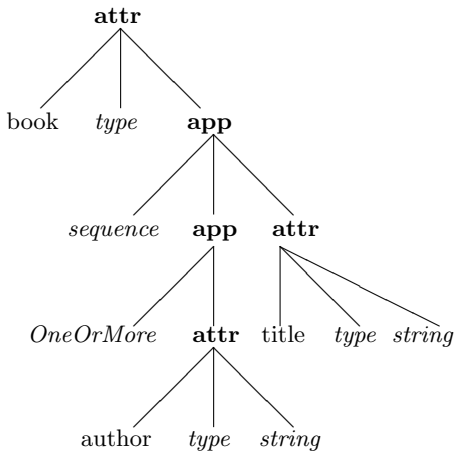

Fig. 1. An example of compound object

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, 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. In other words, content dictionaries are used to assign formal and informal semantics to all symbols (concepts) used in the OPENMath objects. A content dictionary is a collection of related symbols, encoded in XML format and fixing the "meaning" of concepts independently of the application.

2.2 Data Integration Model

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. In the frame of this model

we proposed a minor extension of OPENMath to support the built-in data types concept of the XML Schema. Namely, to model the constants of built-in data types of the XML Schema the corresponding basic objects were introduced. In the context of the considered data model we consider three kinds of mechanisms to formalize the data integration concept:

• content dictionaries to define basic concepts (integrable data, operations, types, etc.);

• signature files to define signatures of basic concepts to check the semantic validity of their representations;

• reasoning file to define knowledge in the frame of data integration concept. Defining a concept in terms of known ones we introduce a new concept (knowledge) in this area. Thus, the considered file is collections of reasoning rules, which are defining the new concepts in terms of known ones.

Extension Principle. Our concept to data integration assumes that the data integration model must be extensible. The extension of the data integration model is formed during consideration of each new data model by adding new concept(s) to its DDL to define logical data dependencies of the source model in terms of the target model if necessary. Thus, the data integration model extension assumes defining new symbols. The extension result must be equivalent to the source data model. For applying a *symbol* on the data integration model level the following rule is proposed:
Concept ← *symbol* ContextDefinition.

For example, to support the concepts of *key* of relational data model, we have expanded the data integration model with the symbol *key*. Let us consider a relational schema example: S={Snumber, Sname, Status, City}. The equivalent definition of this schema by means of extended data integration model is considered below:

S ← *attribution*(S, *type* TypeContext, *constraint* ConstraintContext)

TypeContext ← *application*(*sequence*, ApplicationContext)

ApplicationContext ← *attribution*(Snumber, *type int*),
attribution(Sname, *type string*), *attribution*(Status,
type int), *attribution*(City, *type string*)

ConstraintContext ← *attribution*(ConstraintName, *key* Snumber)

It is essential that we use a computationally complete language to define the context [11]. As a result of such approach usage of new symbols in the DDL does not lead to any changes in DDL parser. According to this approach, the data integration model is synthesized as a union of extensions. A schema of the integrated databases is an instance of the XML DTD for modeling reasoning rules.

3 Algebra of Integrable Data

In the frame of the data integration concept we differentiate one kind of data – integrable data.

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 $\{A_1, A_2, ..., A_n\}$. 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 $D = D_1 \cup D_2 \cup ... \cup D_n$. 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.

3.2 Operations

Virtual and materialization integration of data assumes introduction of special operations, such as filtering, joining, aggregating, etc. The proposed operations are similar to the realtional algebra operations.

To support n-ary associative operations union and joining, we introduced the symbols *union* and *join* correspondingly. The symbol *union* is used to denote the n-ary union of sets (integrable data). It takes sets as arguments, and denotes the set that contains all the elements that occur in any of them: $union : x^{*assoc} \rightarrow d$.

The symbol *join* is used to denote the n-ary join of sets. It takes sets as arguments, denotes a set of elements, and is interpreted analogously to the operation natural join of the relational algebra in general case (joins of many relations): $join : x^{*assoc} \rightarrow d$.

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 elements for which the predicate is satisfied:

$\sigma : \{x \rightarrow \{p : \{element\} \rightarrow boolean\}\} \rightarrow d.$

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

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 elements, and is interpreted analogously to the operation *project* of the relational algebra:

$\pi : x[name^*] \rightarrow d.$

Here *name* denotes the name of an attribution object and is defined as follows:

$name : \{Attribution\} \rightarrow string.$

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

$f : x[name] \rightarrow numericalvalue.$

Often, we needs to consider the elements of an integrable data 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 elements, and is interpreted analogously to the operation *grouping* of the relational algebra:

$\gamma : x[name^*, (f : (element[name^*])^* \rightarrow numericalvalue)^*] \rightarrow d.$

4 An Ontology for Data Integration Concept

Formalization of the data integration concept assumes developing new content dictionaries to model the algebra of integrable data and data types concept of the XML Schema. Also we should define signatures of the introduced symbols (basic concepts) and reasoning rules of the data integration concept.

4.1 The *dic* Content Dictionary File

A content dictionary which contains representation of basic concepts of the data integration concept 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 name and description of the basic concept, and also some optional information about this concept (analogously the *xts* content dictionary for modeling the type system concept of the XML Schema is defined). 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> $X : Attribution^* \rightarrow \{Attribution\}$ < /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. Specific information pertaining to the basic concept like the signature and the defining of a concept in terms of known ones is defined in additional files associated with content dictionaries. 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. Signature files and files of reasoning are used to formalize the different aspects of the data integration concept. Namely, to formalize the basic concepts formats, and to define reasoning rules to formalize knowledge in this area.

4.2 The *dic* Signature File

As is mentioned above, to check semantic validity of the basic concepts representations we associate extra information with content dictionaries, namely signature files. A signature file contains the definitions of all the basic concept signatures of the considered content dictionary. We use Small Type System [4] to formalize the basic concept signatures. Below the definition of the signature of the above considered symbol X is provided :

```
<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>
```

The above considered 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*. The operator is associative on these arguments which means that repeated uses may be flattened/unflattened.

4.3 Reasoning File

We propose an XML DTD to define reasoning rules to support an ontology for data integration concept. The proposed ontology is based on the OPENMath formalism and the algebra of integrable data. As mentioned above, within our approach to ontology-based data integration we consider issues of virtual as well

as materialized data integration. Therefore we should formalize the concepts of this subject area such as integrable data, mediator, data warehouse, data cube, etc.

Let the symbols *msch* and *wrapper* correspondingly denote the set of all mediator schemas and the set of all subsets of the wrappers which are defined on source data schemas to support the mediator concept, and let the symbol *med* denote the set of all mediators, then

$med \subseteq msch \times wrapper.$

The *msch* symbol is based on the OPENMath *attribution* concept. By means of this concept we can model source data schemas. The *wrapper* symbol is based on the OPENMath *application* concept and is presented by an algebraic program of the integrable data.

Let the symbols *wsch* and *extractor* correspondingly denote the set of all data warehouse schemas and the set of all subsets of the extractors which are defined on source data schemas to support the data warehouse concept, and let the symbol *whse* denote the set of all data warehouses, then

$wshe \subseteq wsch \times extractor.$

The symbols *wsch* and *extractor* are interpreted analougsly as in the mediator case.

Materialized integration of data assumes the creation of data warehouses. Our approach to create data warehouses is mainly oriented to support data cubes. Using data warehousing technologies in OLAP applications is very important [7]. Firstly, the data warehouse is a necessary tool to organize and centralize corporate information in order to support OLAP queries (source data are often distributed in heterogeneous sources). Secondly, significant is the fact that OLAP queries, which are very complex in nature and involve large amounts of data, require too much time to perform in a traditional transaction processing environment.

In typical OLAP applications, some collection of data called *fact table* which represent events or objects of interest are used [7]. Usually, fact table contains several attributes representing dimensions, and one or more dependent attributes that represent properties for the point as a whole. The creation of the data cube requires generation of the power set (set of all subset) of the aggregation attributes. To implement the formal data cube concept in literature the CUBE operator is considered [8]. In addition to the CUBE operator in [8] the operator ROLLUP is produced as a special variety of the CUBE operator which produces the additional aggregated information only if they aggregate over a tail of the sequence of grouping attributes. In this context, it is assumed that all independent attributes are grouping attributes. For some dimensions there are many degrees of granularity that could be chosen for a grouping on that dimension. When the number of choices for grouping along each dimension grows, it becomes noneffective to store the results of aggregating based on all the subsets of groupings.

Thus, it becomes reasonable to introduce materialized views. A materialized view is the result of some query which is stored in the database, and which does not contain all aggregated values. The materialized view is interpreted by the OPENMath *application* concept.

Let the symbols *ssch* and *mview* correspondingly denote the set of all fact table schemas which are defined on source data schemas and the set of all materialized views to support the data cube concept, and let the symbol *cube* denote the set of all data cubes in this context, then

$$cube \subseteq ssch \times mview.$$

As we noted above, the reasoning rules to support ontology for the data integration concept are based on the OPENMath formalism and the algebra of integrable data. Let the symbol *source* denote the set of all integrable data schemas and let the symbol *dir* denote the set of all data integration rules, then

$$dir \subseteq source \times (med \cup whse \cup cube).$$

In Appendix A an XML DTD for modeling the reasoning rules of the data integration concept is presented. Below, an example of a mediator for an automobile company database is adduced [7] which is an instance of the XML DTD of the data integration concept. It is assumed that the mediator with schema AutosMed = {SerialNo, Model, Color} integrates two relational sources: Cars = {SerialNo, Model, Color} and Autos = {SerialNo, Model}, Colors = {SerialNo, Color}.

```
<dir>
<! − − Source schemas definitions −− >
 <med>
  <msch>
  AutosMed: schema for mediator is defined
  < /msch>
  <wrapper>
   <OMA>
    <OMS name="union" cd="dic"/ >
    <OMV name="cars"/ >
    <OMA>
     <OMS name="join" cd="dic"/ >
     <OMV name="Autos"/ >
     <OMV name="Colors"/ >
    < /OMA>
   < /OMA>
  < /wrapper>
 < /med>
< /dir>
```

5 Related Work

Using ontologies to support the data integration concept, it is explained that they provide an explicit and machine- understandable conceptualization of a subject domain. The use of ontologies for data integration is discussed in [3]. Examples of ontological modeling languages are XML Schema, RDFS, OWL, etc. There are the following variants of data integration based on the ontologies [19]:

• *Single-ontology.* All source schemas are directly related to a shared global ontology that provide a uniform interface to the user [2]. Single-ontology approach assumes that data sources are semantically close. SIMS [1] is a system which is based on such approach.

• *Multiple-ontology.* Each data source is defined independently using a local ontology. Such approach assumes developing a formalism for defining the inter-ontology mappings. The OBSERVER system [17] is based on this approach.

• *Hybrid-ontology.* This approach combines the two preceding approaches. In other words, for each source schema a local ontology is built alongside a global shared ontology. [2] is an example of this approach.

An important challenge in data integration is the construction of mappings from the source data models into the target one. As rules in well-known works, the mapping from the source models into the target one is constructed semi-automatically. The exception is the work [18] in which the mappings between relational databases and ontologies are generated automatically. Our approach to data integration allows to automatically generate mappings from arbitrary data models into the target one. A more detailed analysis of approaches to ontology-based data integration can be found in [6, 19].

6 Conclusions

In the frame of a data integration model, the data integration concept was formalized. The proposed ontology is based on the OPENMath formalism and the so-called algebra of integrated data which has also been developed by us. The considered data integration model is oriented to XML and is distinguished by its computational capabilities. Such data integration model is an extension of XML by means of the OPENMath concept. In the result of such extension, the XML data model has been strengthened with ontological and computional constructions. Three kinds of mechanisms of the OPENMath are used to formalize the data integration concept: content dictionary, signature file and reasoning file. By these mechanisms we formalize the different aspects of the data integration concept. Namely, we formalize basic concepts (integrable data and operations on it), their signatures and reasoning rules (to model the data integration concepts). The reasoning rules are based on the algebra of integrable data and are formalized by an XML DTD. If necessary, we can extend the algebra of integrable data by adding new algebraic operations. It should be noted that the different

mechanisms of data translation (wrapper, extractor) are non-sensitive to such extension. It is essential that the data integration model is extensible and we use a computationally complete language to support the data integration concept. Finally, based on the proposed ontology, algorithms can be developed to generate the schemas of integrable data and transformers from high level concepts which are represented as an instance of the XML DTD.

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. Arens, Y., Knoblock, C.A., and Hsu, C.: Query processing in the sims information mediator. In *ARPI 1996*. AAAI Press, 61–99 (1996).
2. Cruz, I.F. and Xiao, H.: Using a layered approach for interoperability on the semantic web. In *WISE 2003*, 221–232 (2003).
3. Cruz, I.F. and 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).
4. Davenport, J.H.: A small openmath type system. *ACM SIGSAM Bulletin* **34** (2), 16–21 (2000).
5. Drawar, M.: Openmath: An overview. *ACM SIGSAM Bulletin* **34** (2), 2–5 (2000).
6. Ekaputra, F.J., Sabou, M., Serral, E., Kiesling, E., and Biffl, S.: Ontology-based integration in multi-disciplinary engineering environments: A review. *Open Journal of Information Systems* **4** (1), 1–26 (2017).
7. Garcia-Molina, H., Ullman, J., and Widom, J.: *Database Systems: The Complete Book*. Prentice Hall, USA, 2009.
8. Gray, J., Bosworth, A., Layman, A., and Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-tab. In *ICDE*, 152–159. USA, 1996.
9. Hindley, J.R. and Seldin, J.P.: *Introduction to Combinators and λ-Calculus*. Cambridge University Pressl, Great Britain, 1986.
10. Kalinichenko, L.A.: Methods and tools for equivalent data model mapping construction. In *Advances in Database Technology-EDBT'90*, 92–119. Italy, Springer, March 1990.
11. Manukyan, M.G.: Extensible data model. In *Advances in Databases and Information Systems*, 42–57. Finland, 2008.
12. Manukyan, M.G.: Canonical model: Construction principles. In *iiWAS2014*, 320–329. Vietnam, ACM, December 2014.
13. Manukyan, M.G.: On an approach to data integration: Concept, formal foundations and data model. In *CEUR-WS* **2022**, 206–213 (2017).
14. Manukyan, M.G.: On an ontological modeling language by a non-formal example. In *CEUR-WS* **2277**, 41–48 (2018).
15. Manukyan, M.G. and Georgyan, G.R.: A dynamic indexing scheme for multidimensional data. *Modern Information Technologies and IT-Education* **14** (1), 111–125 (2018).
16. Manukyan, M.G. and Georgyan, G.R.: Canonical data model for data warehouse. In *Communications in Computer and Information Science* **637**, 72–79 (2016).

17. Mena, E., Kashyap, V., Sheth, A.P., and Illarramendi, A.: Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *CoopIS 1996*, 14–25 (1996).
18. Pinkel, C., Binnig, C., Jimenez-Ruiz, E., Kharlamov, E., Nikolov, A., Schwarte, A., Heupel, C., and Kraska, T.: Incmap: A journey towards ontology-based integration. In *BTW 2017*, 145–164. Lecture Notes in Informatics, 2017.
19. Wache, H., Vogele, T., Visser, U., Stuckensmidht, H., Schuster, G., Neumann, H., and Hubner, S.: Ontology-based integration of information-a survey of existing approaches. In *CEUR-WS*, 1–10 (2001).

APPENDIX A. An XML DTD for Modeling the Reasoning Rules of the Data Integration Concept

<! − − *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)*>*s*
<!*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*>