REFERENTIAL INTEGRITY MODEL FOR XML DATA INTEGRATED FROM HETEROGENEOUS DATABASES SYSTEMS

Using the power of XML for consistent data integration

Mauri Ferrandin
UNERJ - Centro Universitário de Jaraguá do Sul - SC, Brazil

Murilo S. de Camargo
UnB – Universidade de Brasília, Brasília - DF, Brazil

Keywords: Distributed Heterogeneous Databases Systems Integration, XML

Abstract: This article presents a proposal for maintenance of the referential integrity in data integrated from relational heterogeneous databases stored in XML materialized views. The central idea is the creation of a repository of rules that will have to be observed to if carrying through any operation of update in the mediating layer of a system for integration of heterogeneous relational sources of data to guarantee that the updates carried through in the data stored in this layer can be propagated to the relational databases that are part of the system integrated without causing problem of referential integrity in the same ones. This proposal has as main objective to specify a mechanism capable to guarantee that the data after exported from the relational heterogeneous databases in a mediating layer continue respecting the same integrity which these were submitted in the origin databases.

1 INTRODUCTION

A Database Management System (DBMS) is capable to solve problems of data access and management of big data amount in one system, but many problems appears when two or more platforms need to work in an integrated way. The majority of these problems are consequences of semantic heterogeneity – same data duplicated and represented in different ways in the database schemas (Hull, 1997).

The major idea of this paper consists is the specification of a system capable of guarantee the referential integrity of data integrated from relational heterogeneous sources through the XML (Extended Markup Language) materialized views, keeping the same integrity that the data was submitted in the relational databases. For that, will be proposed a mechanism to define the referential integrity for data stored in the integrated view, making possible a synchronization of the data integrated back to the databases from which it was integrated.

The proposed system doesn't care about questions and problems with data updates in the distributed databases; the major focus of this work is the maintenance of the referential integrity for exported data.

There are several researches and proposals in the databases and semi-structured data that are correlated to this work, among then, we can enumerate the proposals of MIX (Zhu, 2000), SilkRoute (Fernandez et al, 1999), Argos (Quan et al., 2001), Heros (Castro, 1998), Jupter (Murphy e Grimson, 1995) and others.

The work contains four main sessions: the section 2 describes the interoperability between heterogeneous data provided by XML; the section 3 describes the maintenance of integrity in XML data; the section 4 describes the specification of integrity maintenance mechanism for the XML materialized views.

Ferrandin M. and S. de Camargo M. (2004).
REFERENTIAL INTEGRITY MODEL FOR XML DATA INTEGRATED FROM HETEROGENEOUS DATABASES SYSTEMS - Using the power of XML for consistent data integration.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 15-20
DOI: 10.5220/0002643800150020

2 INTEROPERABILITY OF DBMS SYSTEMS AND XML

With the specification of XML standard, people around the world begun to use it for data exchange because it makes possible and easy to represent self described data through metadata, making possible the representation of some data that couldn't be represented in the relational model, model used by the most databases systems. With XML standard is possible to create views (materialised or not) of data stored in one DBMS and use this view for many ends.

In this scenario that researches about databases are being developed is clear the need of models for data integration between relational databases and the Web. The convergence of the solutions involving XML and semi-structured data technologies will create a new combined technology for the Web (Abiteboul et al., 2000).

The technologies related to XML are being incorporated like native functions in the most important databases systems. Databases that don't implement this native support need the implementation of an intermediary layer called middleware with the objective of guarantee the interoperability.

A middleware is a software component that exploits the codified knowledge about sets or subsets of data for create a layer with useful data for the application layer (Wiederhold, 1992). An application that need to access data stored in heterogeneous databases will need an intermediary software layer for intermediate the data exchanges between her (the application) and the data sources, this mediation layer will have functions like abstraction, combination and explication of data.

The presence of a middleware indicates the existence of architecture based on three layers showed in the Figure 1.

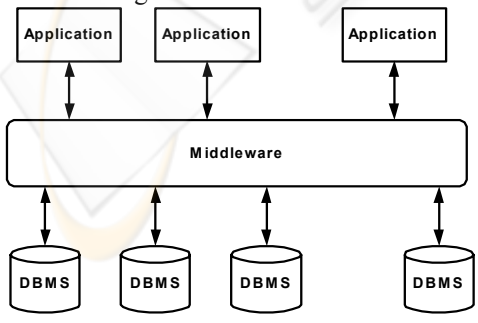

Figure 1: Basic architecture of a data integration system with middleware.

The superior layer is composed by the applications, the intermediary layer is composed by multiples mediators managed by a specialist in knowledge domains and the number of mediators depends of how much the heterogeneous data are needed by applications in the application layer and the databases that composes the data layer where the data are stored.

A mediator have a job of provide data for the applications, acting like a intelligent interface to solve problems of different data representation and abstraction in the different data sources, it works with active rules and contains knowledge structures to guide the data transformations.

2.1 Representing Relational Data with XML

A relational DBMS is represented by a schema, for example:

```
r1(a,b,c) and r2(d,e)
```

where r1 and r2 are the relations name, a,b,c are columns of the relation r1 and d,e are columns of the relation r2.

Let's consider an example of a relational database composed by two relations r1 and r2 described by a notation:

$$\{r1 : i_1,\ r2 : i_2\}$$

where i_1 and i_2 represent the data stored in the respective relations that can be represented by a set of rows (tuples) like:

```
{r1 : {tuple{a : a1, b : b1, c : c1},
       {tuple{a : a2, b : b2, c : c2},
},
{r2 : {tuple{d : d1, e : e1},
       {tuple{d : d2, e : e2},
       {tuple{d : d3, e : e3}
}
```

These relations can also be represented as tables as showed bellow:

r1:

a	b	c
a1	b1	c1
a2	b2	c2

r2:

d	e
c1	e1
d2	e2

Abiteboul (Abiteboul et al., 2000) says that we can represent the data as a tree and in various ways as we can see in the Figure 2 and Figure 3.

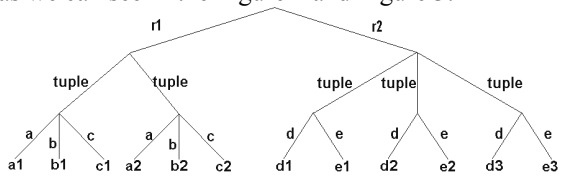

Figure 2: Representation of relational data as a tree (1)

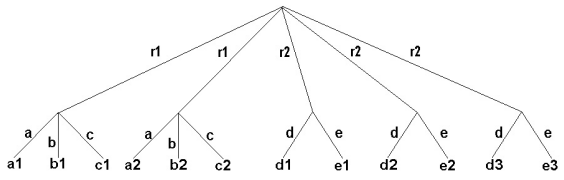

Figure 3: Representation of relational data as a tree (2)

Every tree represented in the pictures Figure 2 and Figure 3 can also be represented with XML:

```
<?xml version="1.0"?>
<db>
    <r1>
        <tuple>
            <a> a1 </a>
            <b> b1 </b>
            <c> c1 </c>
        </tuple>
        <tuple>
            <a> a2 </a>
            <b> b2 </b>
            <c> c2 </c>
        </tuple>
    </r1>
    <r2>
        <tuple>
            <d> d1 </d>
            <e> e1 </e>
        </tuple>
        <tuple>
            <d> d2 </d>
            <e> e2 </e>
        </tuple>
        <tuple>
            <d> d3 </d>
            <e> e3 </e>
        </tuple>
    </r2>
</bd>
```

In this way, we conclude that we can represent the data stored in a relational database with XML in many ways. The structure of this XML document will be determined by who (user/application/developer) will eventually uses it.

3 REFERENTIAL INTEGRITY WITH XML DATA

The questions related to the maintenance of integrity in XML data is receiving great attention by researchers communities around the world with intuit of create ways and standards to define and validate integrity rules with XML documents, like primary keys and foreign key rules in a database, for example. Various models were proposed to define and verify the integrity of XML documents using the standards of the XML specification, like the Data Type Definition (DTD) and the XML Schema, and others (Buneman et al., 2001).

The biggest problem on specify integrity constraints for XML documents is how to determine, for example, that one element present in one part of the tree is a primary key and may appear only once in a determined context, or how to determine that a attribute of this element is a foreign key referenced to a existence of other element with the same value of this attribute in other part of the document. To solve this problem, is necessary use a standard to make possible the representation of the path between root and one element in a XML document, and then represent a element and in what level of the document it is.

For (Buneman et al., 2001), keys have a fundamental importance in a database providing a efficient way for data integrity maintenance, establishing relationships between the entities and lookup data, having a great importance in the validation of data making possible to keep the data according to a model defined to represent then based on the real world. Many models were proposed for maintenance of keys using the XML standard through the DTD and XML Schema, but major part of them was inefficient in many cases. The XML document doesn't need to have its definition (through DTD or XML Schema), so it's very useful to create a mechanism for creation of integrity constraints independent of his type.

For to surpass these limitations, authors proposed that mechanisms of key definition for XML documents:

- should be defined through one or more path expressions making possible determine precisely on node in a XML tree;
- should be defined for a relatively set of nodes;
- doesn't have any dependencies of any mechanism used for to specify the document structure like DTD or XML Schema;

(Chen et ali., 2002) proposed one notation for to define keys with the following syntax:

$$(Q, (Q', \{P_1, \ldots, P_P\}))$$

where Q, Q', and $P_1,...,P_P$ are path expressions. Q is called context path, Q' target path and $P_1,...,P_P$ are the keys paths. The idea is that the context path Q identifies the set of context nodes where, for which node *n* the integrity constraint must be respected in the set of nodes accessible through the target path Q'.

For example, a key KS_1 defined by :

$$KS_1 = (/, (./university, \{./name\}))$$

indicates that in a context path "/" (the root of the document) one university is identified by a sub element denominated "name". Other example, the key :

KS_2=(/*university*,(.//*employee*, {./@*employeeId*}))

indicates that in some level of a document identified by a element university (context path), one employee node present in any sub level (indicated by ".//") of the document in the context path is uniquely identified by a attribute ("@") of the element employee named "employeeId". Lets take another example of integrity constraint defined by :

KS_3= (/, (.//*employee*, {./*name*, /*phonenumber*}))

that indicates that in the context path "/" (or better, entire document), one element "employee" present in any sub level of the XML tree is identified uniquely by a element named "name" and one element named "phonenumber".

This notation seems to be a little bit complex, but it makes possible the definition of integrity rules about XML documents. There are several researches for implementation of the algorithms and mechanisms for validate this kind of rules over XML documents.

4 PROPOSED MODEL

Too many questions should be considered to propose a integration of data proceeded from relational heterogeneous databases. Thus, the proposed model is focused in the maintenance of integrity constraints for XML data integrated from heterogeneous RDBMS, providing a mechanism capable of to keep the integrity of the database of origin to the data integrated through XML views. This mechanism makes possible to serialize the data back to his originally database from where it was integrated, with no integrity violation.

4.1 General view of the proposed system

Usually, a system that provides data integration from two or more relational sources is composed by a lot of software modules. Those modules have a main objective the data integration and to provide the integrated data to the application layer, in a transparent way.

Commonly, a general architecture proposed to an integration framework specifies a set of specialised wrappers for the different data sources that are capable of export the data from a relational schema to a XML view. The set of wrappers will generate a set of views that will be integrated, generating another view (global view) storing the data from all databases components of the system. This makes possible applications/developers to access this data in a transparent way.

This work proposes the inclusion of a software module responsible for the maintenance of constraint integrity rules in a global view (Figure 4) composed by integrated data making possible the serialization of the updates made to the global view to de local views and then to the original data sources without violation of referential integrity.

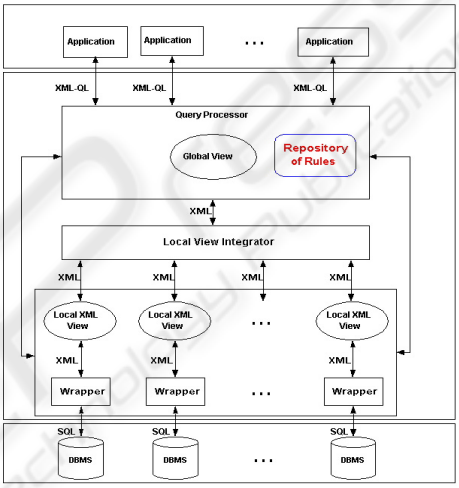

Figure 4: Architecture proposed for data integration.

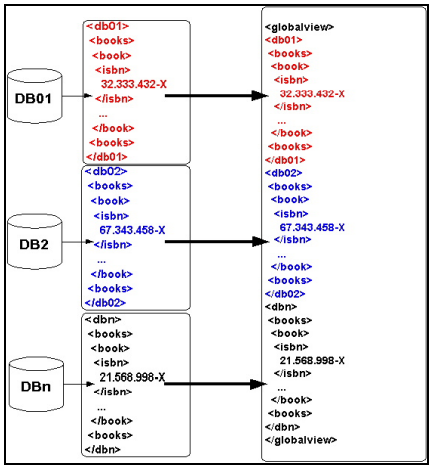

Figure 5: How is composed the global view.

In order to store the integrity rules for the global view, a repository will be created and these rules will be defined using the XML standard. In a integration framework the presence of a human

specialist will be necessary for the definition of the integrity rules, this expert will define what rules presents in the databases will be preserved in the global view too according with the application. A minimal integrity will be necessary because rules like primary keys and foreign keys are indispensable for synchronise the data back to the origin.

The picture Figure 5 show us how the data exported from the databases will be organised with n local views compounding a global xml view. The root element of the global view was called <globalview>, and every database integrated is identified by a node <dbxx> that will be stored like a sub element of the root element.

4.2 Repository of integrity rules

The integrity rules of the global view are defined according to the integrity rules of the databases from which the data was exported to the local views. For storing these rules a XML based notation will be used, maintaining in this way, a standard mechanism for query and data exchange between the modules of the integration framework.

The following example shows an example of the one repository of integrity rules stored with XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<integrityrules>
  <globalview.db01.books.book type="pk">
    <fields>
      <field>isbn</field>
    </fields>
  </globalview.db01.books.book>
  <globalview.db02.bookpublishers type="pk">
    <fields>
      <field>isbn</field>
      <field>publish_id</field>
    </fields>
  </globalview.db02.bookpublishers>
   . . .
</integrityrules>
```

One rule is represented by one tag that indicates the context path of the global view where this rule will be observed and it's type is represented through a attribute of this tag that can be "pk" (for a primary key rule) or "fk" (for a foreign key rule).

The tag that represents a rule will have one or n sub tags to represent the fields that compose the key in that context.

Let's analyse this rule:

```
. . .
<globalview.db01.books.book type="pk">
  <fields>
    <field>isbn</field>
  </fields>
</globalview.db01.books.book>
. . .
```

Basically, the described rule represents a primary key rule meaning that in the context <globalview.db01.book> we will never insert two sub elements called isbn with the same values.

The following cod represents a set of invalid values in a global view according to the primary key rules defined before for store the books data :

```
<?xml version="1.0" encoding="UTF-8"?>
<globalview>
 <db01>
  <books>
   <book>
    <isbn>83.326.1461-X</isbn>
    <title>Teaching XML to Yourself</title>
   </book>
   <book>
    <isbn>83.326.1461-X</isbn>
    <title>Teaching XML to Yourself</title>
   </book>
  </books>
 </db01>
 ...
</globalview>
```

A foreign key rule indicates, in fact, that one object only can be stored in a context if there is a related object in other context of the document. So, the foreign key rules need one more attribute to represent the context where the related object must exists. In the following example we show a definition of a foreign key rule:

```
<?xml version="1.0" encoding="UTF-8"?>
<integrityrules>
 <globalview.db01.books.book type="fk">
  <fields>
    <field>publisher_id</field>
  </fields>
  <relatedcontext>
   <globalview.db01.publishers.plublisher>
    <fields>
     <field>publisher_id</field>
    </fields>
   </globalview.db01.publishers.plublisher>
  </relatedcontext>
 </globalview.db01.books.book>
</integrityrules>
```

As we can see, the foreign key specification is composed by a tag indicating the context element and a sub tag indicating the related context. In practice, this rule means that an element representing the book publisher in the books context must have a related element in the publishers context.

Now, we can generalize and define other integrity rules and other features presents in the databases, like triggers, stored procedures or others in the repository of rules :

```
<?xml version="1.0" encoding="UTF-8"?>
<integrityrules>
```

```
 <globalview.db01.books type="trigger">
  <onnewrecord>
    <execute>
     dbms_functions.updaterelatedcontext()
    </execute>
  </onnewrecord >
  <ondeleterecord>
    <execute>
     dbms_functions.deleterelatedcontext()
    </execute>
  </ondeleterecord >
 </globalview.db01.books>
</integrityrules>
```

5 CONCLUSION

Keep the integrity constraints of data exported from heterogeneous databases systems is a hard job, hardest than to create integrity rules for validation of a simple XML document, due to the fact that every database that will compose the system has his own schema with his integrity rules defined.

The major deficiencies of the proposed system are:

- the necessity of a human expert to start up the system, defining the initial integrity for the system and if a one database schema is changed, this specialist will have to do the work again;
- a complexity acquired by the system if we consider a context where the most part of queries executed by the system are for updating the data in the global view, overcharging the system for the data synchronisation with the distributed databases.

Compared to other models and strategies for data integration and constraint rules validation for the integrated data, the proposed model bring as major contribution to the context of data integration from heterogeneous sources a solution to the problematic of maintenance of the same referential integrity in the integrated data that this was respecting in the database from which it came from, with the objective of to guarantee that this that the updates in a global view will not cause integrity violations if propagated to de databases. The proposed system was specified using open standards with intuit of keep it independent from hardware or software platform, and the XML is very important for this job, for this reason, we used XML for specify the integrity rules.

As suggestions for future works we enumerate, (i) a specification of a system capable of to extract automatically the integrity rules from databases and generate the repository, (ii) a implementation of a XML API (as SAX, DOM etc) capable of validate the rules like these proposed in this work against XML views, (iii) extent this idea to object oriented databases.

REFERENCES

ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. Data on the *Web* : From relations to semistructured data and XML. Morgan Kaufmann Publishers, San Francisco, CA, 2000.

BUNEMAN, Peter; DAVIDSON, Susan; et al. Reasoning about Keys for XML (Technical Report). International Workshop on Database Programming Languages (DBPL), 2001. Disponível em http://db.cis.upenn.edu/DL/absreltr.ps.gz

CASTRO, C. E. P. S.. Integração de Legacy Systems a Sistemas de Bancos de Dados Heterogêneos. Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Jul. 1998.

CHEN, Yi; DAVIDSON, Susan; ZHENG, Yifeng. Validating Constraints in XML. Department of Computer and Information Science, University of Pennsylvania, 2002. Disponível em http://db.cis.upenn.edu/DL/validate.ps.

FERNANDEZ, M.; TAN, W. C.; SUCIU, D. SilkRoute: Trading between Relations and XML. University of Pennsylvania: [s.n.], 1999. Disponível em: < http://db.cis.upenn.edu/RXL/papers/sr.html>. Acesso em: 22 mar 2000.Technical Report.

HULL, R. Managing semantic heterogeneity in databases : A theoretical perspective. In: ICDT, 1997. [s.n.], 1997.

MURPHY, J. & GRIMSON, J.. Multidatabase Interoperability in the Jupiter System. Information and Software Technology. Vol 37, N. 9, 1995.

QUAN, L.; CHEN, L.; RUNDENSTEINER, E. A. Argos: Efficient refresh in an XQL-based *Web* caching system. Lecture Notes in Computer Science, [S.l.], v.1997.

WIEDERHOLD, G. Mediators in the architecture of future information systems. Computer Magazine of the Computer Group News of the IEEE Computer Group Society, [S.l.], 1992.

ZHU, Y. et al. Materializing *Web* data for OLAP and DSS. In: *WEB*-AGE INFORMATION MANAGEMENT, 2000. [s.n.], 2000. p.201–214.