Distributed database system in interaction with XML, an identification problem

Gabriel Dragomir
Department of Computer Science
Technical University of Cluj-Napoca
George Baritiu St. 26-28 Cluj-Napoca
Romania
Gabriel.Dragomir@astral.ro

Iosif Ignat
Department of Computer Science
Technical University of Cluj-Napoca
George Baritiu St. 26-28 Cluj-Napoca
Romania
Iosif.Ignat@cs.utcluj.ro

Abstract – **In last years XML has been accepted as the new standard for data representation and exchange on the Internet (EDI). This paper takes in discussion the situation where two or more enterprises, like a bank and a business, have common customers and there are frequent transactions between (by example direct debit). Each of them is supposed to have it's own customer database (distributed). The databases in general are heterogeneous but have the capabilities to treat XML documents (input and output). The identification problem appears when the same real-life object has different identifiers in different databases. We propose the usage of synonyms for candidate attributes of identifiers and focus on XML capabilities to express multivalued values for an attribute element. The system has the ability to maintain such relation between synonyms and primary keys. It has the capacity to learn new rules from facts also stored in XML document which is the means of exchange.**

I. INTRODUCTION

A. An example helping us in describing the identification problem

In last years Electronic Data Interchange (EDI) had found new opportunities.

Data integration [11] is made sharing resources or by sending data using networking and inter-process communication mechanisms. Example of first is today's Enterprise Resource Planning (ERP) packages e.g. the customer service application understands the schema and relationships for its own tables as well as the tables for sales and financial information.

Integration through data exchange creates a loosely coupled relationship between applications. This approach implies significant custom code to perform mappings between the two loosely coupled schemas.

For simplify the presentation, we take two generic enterprises named "Businesses" (B1 and B2 in fig. 1). Each of them has its own applications and database. There is no presumption as regards database schema. The two businesses have to cooperate in administer to their (common) customers.

An example is with a bank and a regular services company. A company customer instructs his bank to collect varying amounts from his account, as long as the customer has been given advance notice of the collection amounts and dates (direct debit).

The most common model for DB (DataBase) is the relational model. An enterprise may have a hierarchical organization and DB should be distributed over many sites. The rule thereupon values for keys are assigned in organization's DB is pure subjective. Suppose a new client which makes a demand. This client must be recorded in *Customer* table where the object will get an *idcust*. The problem is to discern if a client is really a new client or it is an old. Here, existence of candidate key such as personal identification number or a combination of forename, surname and other characteristic that guarantee uniqueness (like initial letter of parent's forename) is very important.

In such a situation where the database is distributed the system has to search every location (that means overload of communication) to assure uniqueness of the identifier. In a system with active database capabilities it is more convenient to state some rules that will govern trigger's logic as in [6]. A rule could be: "Customers are allocated according to location of their address". This rule implies that there exists *Address* relation and it is fragmented over locations. Each location has a correspondence table with all address localities allocated to that location. At this rate trigger's choice between fragments of *Customer* table will be straightforward.

There are many possibilities to implement such a distributed database. One way is with pure distribution where each node in the system is a fragment and where global schema is obtained at the external level (by example partitioned views). Replication is another solution [5], but the two could be combined.

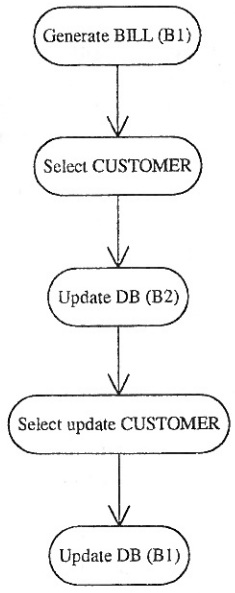

Fig. 1. Activity diagram for the example taken into discussion[1]

[1] The stages deal with collection of customers although singular was used for nouns.

Of course there may be put the question "what importance is the distribution of the database" in the context of electronic data interchange. If we treat the problem as a whole we must look at the entire data flow, starting with the event of billing and for optimization purposes may be important to know.

Let take the stages from activity diagram in fig. 1 shortly into discussion. After a bill is generated, that customer is a member of the set candidate to "Select CUSTOMER" process. Communication between B1 and B2 may be chosen depend on a common protocol. It is unlikely that the two systems could be directly connected like a homogenous one. More credible is that B1 and B2 are like two black boxes and they agree data structure for EDI.

So, the result of the process "Select CUSTOMER" is of that agreed data structure. Similarly is the result of "Select update CUSTOMER" with the difference that data source is B2 database.

The processes that update databases: "Update DB (B1)" and "Update DB (B2)" have as input the common data structure and as output the updated database. Each process must know some rules about conveyed protocol. By example, the names of attributes that describe values of billed services and are used for account transaction and on the other side the names of attributes that mention checked services. Each business should have the correspondence C between entities in data source DS and entities from data interchanged DI.

A question that arises from is how much must know each of them about the other. The aim is to reduce the amount of knowledge and make system scalable. When B3 comes in, no farther need for upgrade system's components is desirable.

There are similarities between data integration systems wherein views take an important role like Local-As-View (LAV) respectively Global-As-View (GAV) [9] and data exchange systems.

We will take the formalisms from [7] where a data exchange setting consists of a source schema S, a target schema T, a set Σ_{st} of source-to-target dependencies, and a set Σ_t of target dependencies. Given a finite source instance I, the data exchange problem associated with the setting is to find a finite target instance J such that $\langle I, J\rangle$ satisfies Σ_{st} and J satisfies Σ_t. The set of source-to-target dependencies are of form:

$$\forall x(\phi_S(x) \rightarrow \exists y \psi_T(x,y)),$$

where $\phi_S(x)$ is a conjunction of atomic formulas over S
$\psi_T(x,y)$ is a conjunction of atomic formulas over T
free variable x is a vector of variables $x_1, x_2, \ldots, x_k$
k is the set of relations in S
free variable y is a vector of variables $y_1, y_2, \ldots, y_l$
l is the set of relations in T.

A target dependency from Σ_t is either touple-generating dependency (tgd):

$$\forall x(\phi_T(x) \rightarrow \exists y \psi_T(x,y)),$$

or an equality-generating dependency (egd):

$$\forall x(\phi_T(x) \rightarrow (x_1 = x_2)),$$

where x_1, x_2 are among variables in x.

Let O be a real-life object and R_1 respectively R_2 the representations of O in databases DB_1 (with schema S) and DB_2 (with schema T). Identification problem appears in the context of correspondence C when an instance of O in DI must match an instance of O in DS. That is to found for each O starting with R_1 the set Σ_{st} so the resulting R_2 is a representation of O.

B. XML and the Internet

XML documents are the best choice B1 and B2 could communicate with. Below we enumerate some of the properties that relieve it.

Extensible Markup Language (XML) [12] is a meta-markup language derived from SGML (Standard Generalized Markup Language). It provides a framework for data interchange via Internet instead of permanent data storage. When the database is distributed it acts like an external global schema (fig. 2).

The template files, XML-Data Reduced (XDR) schema files (MS SQL Server) or Document Access Definition (DAD) files (DB2, IBM)[2], and Extensible Stylesheet Language (XSL) files could accompany XML documents. The XDR is flexible and overcomes some of the limitations of the Document Type Definition (DTD), which also describes the document structure. Unlike DTDs, XDR schemas describe the structure of the document using the same syntax as the XML document. It can be used an XDR schema with annotations that describe the mapping to the database to query the database and return the results in the form of an XML document.

It is not necessary that XML document carry along a schema Σ.

```
<Customer>
        <Surname/>
        <Forename/>
        <Address>
                <Locality/>
                <Street/>
                <Number/>
                <Postal_code/>
        </Address>
        <Bill>
                <Bill_date/>
                <Total_amount/>
                <Service>
                        <Name/>
                        <Value/>
                        <Check_date/>
                </Service>
        </Bill>
</Customer>
```

Fig. 2. Sample of XML document

[2] Oracle XML DB is available in Oracle9i release 2 and XML schema is integrated into database

There are many ways in which XML documents could be manipulated. There is possible that one own business DBMS (Data Base Management System) be able to do this manipulation or in other case a special, external, module (e.g. with HTML and java script) could be used for this purpose.

ICE (Information and Content Exchange) [8] treats with syndicators and subscribers. The delivery of message (XML document) is done by push or pull.

The capacity of self-describing of a XML document facilitates reduction the amount of knowledge a business must know before it can communicate with other. However there must be a correspondence (C) between XML document and its internal database schema. For process automation purpose, XDR schema for document has to be created.

When a business receives XML document, a process that exploits hierarchical structure of document nodes (DOM -Document Object Model) is initiated. This process will generate a derived document whereat an "element type" (E) is tied up to a correspondent database table and an "attribute type" (A) is tied up to equivalent column. "PCDATA" type ({S}) is a text node useful in specification additional information.

The way nodes could be identified in hierarchical structure of an XML document (keys for XML) was studied in [1, 3].

An XML document is a tree

$$T = (V, lab, ele, att, val, r),$$

where V is a set of nodes;

lab is a mapping $V \rightarrow E \cup A \cup \{S\}$ which assigns a label to each node v in V;

ele and *att* are partial mappings that define the edge relation of T for any v in V. If $lab(v) \in E$ then ele(v) is a sequence of elements and text nodes in Vand att(v) is a set of attributes in V. For each v' in ele(v) or att(v), v' is called a child of v and there is a edge between v and v'. If v is an attribute or text node, then ele(v) and att(v) are undefined;

val is a partial mapping that assigns a string to each attribute and text node, for element type node val(v) is undefined;

r is the unique and distinguished root node.

The keys for XML documents are expressed in matter of absolute or relative paths.

Depends on the semantic of database transaction derived attributes may exists, like collector account debit which is the sum of services who's values are lower than credit of customer's account.

There is no standard method to translate XML document in DB schena [11], there are few techniques: visual mapping tools, xPointer and DOM XML documents suffer from impedance mismatches with relational technologies. There is not a 1:1 mapping between a hicrarchy and relational operators.

The problem of transformations from XML data to relations has been studied inter alia in [4] where an algorithm for the reformulation of client XQueries in XML publishing is found (chase and backchase algorithm).

II. THE ABILITY TO TREAT SYNONYMS

However there is not absolute that same real-world person (or in general real-world object) have the same representation in the two databases of above example.

In database applications a good programming issue is that of attribute to domain restriction. The best thing is to prohibit bondless values of key constituents and to restrict access only to values in acceptable domain. An *acceptable domain* has to be defined for each key (primary and candidate) constituent attribute. At application level, presentation layer, a user could only pick out from a list whose data source is *acceptable domain*. The presence of synonyms is most valuable in queries and can simplify identification for referential integrity purpose. For an *acceptable domain* existence of synonyms means doubly space but allow for the goal, the programmer must look at it.

Let AC be *acceptable domain* for a key constituent attribute and ac a value of the domain. For every ac there exist zero or many syac values from SYAC (synonyms) domain.

AC and SYAC are treated like completed entities [10] so they are not fragmented.

Any SQl specification

$$attr = ac$$

is to be replaced with the specification

$$attr \text{ IN } LSYAC, \qquad (1)$$

where LSYAC is

$$\{ac\} \cup \{\text{select Syac from ASYAC where original value is ac}\}.$$

ASYAC is a dual (Oac, Syac) that convey relation above.

Exteriorly there is a view with n values for each internally value, where n minimum is 1 (the original value, ac). In other words there is a one-to-many relationship between internal and external representation of an object identifier constituent.

With this presumption all the queries are sound although when cardinality of LSYAC is one, a query is equivalent with a completed query.

AC is updated in manner of facts. If no correspondence exists then a rule based system decides the instance of ASYAC to be added. The same thing keeps vice versa, when an instance has to be removed.

A special case is when two (maybe different) objects from two different databases are being put in relation. In assumption ASYAC exists in the two databases, two situations appear:

- The first situation is when the two objects represent the same real-world object. The oac may differ in the two databases. In this case each of them considers the other like a synonym. Therefore the interest is on LSYAC which includes oac.
- The second situation allow for one-to-many object relationship between the two databases (by example

one person from bank database which pays his own bills and bills of his child) and requires one more information, this relationship.

In the context of data the two businesses have to interchange, LSYAC for each candidate key constituent attribute has to be included in message.

When the candidate key is composed, then the values for constituent attributes form a subset of Cartesian product:

$$LSYAC_1 \times LSYAC_2 \times \ldots \times LSYAC_k$$

where k is the number of constituent attributes.

In fact there is a multivalued dependency [2]:

$$Pkey \rightarrow\rightarrow Syac_i,\ i=1..k,$$

where Pkey represents primary key and is considered an internal database assigned attribute. In PJ/NF, that means existence of k relations $CKEY_i(Pkey, Syac_i)$.

III. THE ARCHITECTURE OF THE SYSTEM

In the case of distributed databases, communication could be in term of global schema, or in term of fragments. In definition of what distributed database means, there are items like location transparency. The communication process could be simplified by selecting the fragment whom operation applies to.

Depends on fragmentation predicates, it can be state some rules for generate adecvate XML documents and so to reduce document's dimmension.

The system's architecture is presented in fig. 3.

SYN Converter is a component which main role is transcription of (1). Depends on XML facility of local DBMS if there exists or not a special component to interpret XML documents to database schema.

The synonyms of a candidate key constituent could exist in XML document as multivalued attribute or as distinct elements with proper attributes. SYN Converter among other things will find pkey from the join:

$$\bowtie\ (CKEY_1, CKEY_2, \ldots, CKEY_k)$$

and at one way will construct a well-formed XML document.

In sample from fig. 2 an instance of customer id candidates look like below:

1. multivalued attribute

```
<Surname Sur1 Sur2 />
<Forename For1 For2 For3/>
```

2. distinct element for id candidate attribute

```
<Surname>
  <Value1 Sur1/>
  <Value2 Sur2/>
</Surname>
<Forename>
  <Value1 For1/>
  <Value2 For2/>
  <Value3 For3/>
</Forename>
```

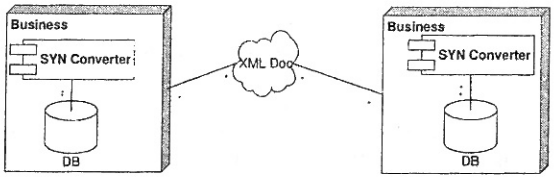

Fig. 3 The architecture of the system with two businesses

The correspondence C, an XML document (also a materialized global view), must conform to a set of constraints of XML data model. Such a constraint is

$$\forall x,n,v_1,v_2\ [attr(x,n,v_1) \wedge attr(x,n,v_2) \rightarrow v_1=v_2].$$

That means, in the situation of distinct element for id candidate attribute in relational schema, there exist a set of XML attributes with same cardinality as the corresponding set LSYAC. In other words SYN Converter will assign distinct name for each attribute resulting from values in LSYAC. Whereas in first alternative entire set LSYAC becomes a multivalued attribute.

If primary keys should be stored in XML document then the document has to be updated each time a new bill is generated and obsolete data have to be removed and it became an alternative storage of common customers. We consider as long as effort is large it is better to generate a new document each time a business needs.

Algorithm to perform tasks of SYN Converter, anytime global view has to be materialized, starts for each instance of leading relation with getting the appropriate set of $Syac_i$.

$$\pi_{Syaci}(\sigma_{Pkey=pkey}(CKEY_i))$$

where π and σ are algebraic operators projection and selection.

This algorithm supposes that locally exists a schema Σ^L that maps relational schema to global view represented by XML document.

At the other side, the transformation means that for each element in XML document with multivalued attributes (or with children attributes of id candidates), Cartesian product of the sets of multivalued attributes is stored in a temporarily (universal) table. Next, it scans while a row is found in the target relation (table or view, when distributed database is reconstructed with partitioned views) corresponding to element type node parent of attribute type node and pkey is retrieved. If no one is found then an inference process begins from a set of auxiliary information (in the worst case human conducted).

Auxiliary information is captured from special nodes of PCDATA type, where they are written from assertions existing in DB.

If still exist orphan instances in temporarily table after auxiliary information is operated, then there must be found another inductive dependency consequent on instances of other relation present as descendants elements in XML document starting with current element.

In our example such of dependency may be a functional dependency Address → Customer (in supposition address represents an alternative to customer identification, a customer has a unique address) which is retrieved from the

set $\sum_t$ of target dependencies.

The discussion may continue then with that entity which is not the leading one. That is Address element may have at its turn multivalued attributes.

In case DBMS do not have XML facilities, a special component will convert DOM specifications in database specifications and vv.

For optimization purposes if location from addresses of customers overlaps the location of distributed databases, XML document should include a selection of customers upon location. In this case the communication will be between two points representing fragments.

Another possible optimization derives from activity diagram in fig. 1. At final stage operation will be with initial database. At precedent stage the materialized view is a subset of initial view:

$$\forall c \in UC \rightarrow c \in C$$

where c is an instance of customer
UC is the set of updated customers
C is initial set of customers

This is why the proposition that primary keys should not be present in XML document is not sustained. This proposition shall be rewritten as:

"Whenever there exists a cycle in data exchange, primary keys must accompany. The document is regenerated at the beginning of a new cycle."

A characteristic of the system is the capacity to enrich the knowledge about synonyms. When UC XML document is constructed, the set i for each id candidate is enriched from actual $CKEY_i^T$. At destination (initial source) in update process $CKEY_i^S$ is synchronized with $CKEY_i^T$. This task is opposite with local optimization but in perspective is gainful. The next time this business communicates with another the chance to a mismatch is reduced.

IV CONCLUSIONS AND FUTURE WORK

In this paper we have studied a problem that emerges from business-to-business data exchange. It is a common fact that same thing not to have the same representation (in sense of values of instance) in two different business databases. In our proposed system there exist a component namely SYN that manages the multivalued dependency between primary key and id candidate components.

As data exchange support XML was chosen. The XML document acts like a global (sound) view.

Our method is a subclass of universal data exchange solution [7].

The system can be optimized when interacts with a distributed database in case of fragment predicates contain common atoms like location = 'value'. The construction of XML document should follow the same rule of partitioned views in distributed system.

An important characteristic of the system is the capacity to enrich synonyms dictionary when the cycle is closed.

For now on we want to implement the system so that it could deal with heterogeneous data sources. We would like to show the results in another paper.

V. REFERENCES

[1] P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, "Reasoning about keys for XML," *DBLP*, 2001

[2] C. J. Date, R. Fagin, "Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases," ACM Transactions on Database Systems 17(3), 1992, pp. 465-476

[3] S. Davidson, W. Fan, C. Hara, J. Qin, "Propagating XML Constraints to Relations," *ICDE*, 2003

[4] A. Deutsch, V. Tannen, "Reformulation of XML Queries and Constraints," International Conference on Database Theory (ICDT), 2003

[5] G. Dragomir, "Techniques for replicated data management," *ACAM*, vol. 9, no. 1-2, 2000, pp. 30-40

[6] G. Dragomir, "Identification in a Distributed Database System," *in Proceedings of the 2003 SINTES11 vol2*, pp. 323-327.

[7] R. Fagin, P. G. Kolaitis, "Data Exchange: Getting to the Core," ACM Symposium on Principles of Database Systems, 2003, pp. 90-101

[8] C. F. Goldfarb, "*The XML Handbook*", Prentice Hall PTR, Upper Saddle River, NJ: 2001, p. 300.

[9] M. Lenzerini. "Data integration: A Theoretical Perspective," in PODS, 2002, pp. 233-246

[10] A. Lelutiu, "High-level design using Data Models for Education Information Systems," SSGRR, 2002

[11] J. P. Morgenthal, B. La Forge, "Enterprise Application Integration with XML and Java," Prentice Hall PTR, NJ07458, 2001, pp. 91

[12] J. Widom, "Data Management for XML:Research Directions," *IEEE Data Engineering Bulletin*, Special Issue on XML, 22(3), Sep. 1999, pp. 44-52.