3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)

Proceedings of Engineering & Technology (PET)
pp. 185-191

Extending the use of RuleML to store metadata and database semantics

Atia M. Albhbah
Faculty of Information Technology
Alasmarya University
Zliten, Libya
E-mail: a.albabah@yahoo.com

Mohamed A. Mgheder
Faculty of Information Technology
University of Tripoli
Tripoli, Libya
E-mail: m.mgheder@uot.edu.ly

Mick Ridley
School of Electrical Engineering and Computer science
University of Bradford
Bradford, UK
E-mail: M.J.Ridley@Bradford.ac.uk

***Abstract*— Shifting legacy data held in stand-alone systems to be used in Web application systems can be expensive and time consuming. RuleML can be used to represent RDBMS data by storing database metadata in an external format for some design tools. Just as XML Schema, which uses elements and attributes to express the semantics of XML data, but XML Schema does not have active elements in principle; RuleML is used as a representation for RDBMS metadata too. This paper proposes the use of RuleML format to implement more semantics for Web forms and demonstrate how this RuleML based approach can provide support for greater semantics using the example of advanced domain support even when this is not a feature supported by a specific RDBMS.**

***Keywords*— Rulebase, RuleML, XML, Metadata.**

I. INTRODUCTION

Domains are useful for abstracting common fields between tables into a single location for maintenance. For example, an email address column may be used in several tables, all with the same properties. This allows us to define a domain and use that rather than setting up each table's constraints individually. The benefits of domains are many [1] for example:

- A constraint placed on a domain ensures that all columns and variables intended to hold values in a range or format can hold only the intended values. For example, a data type can ensure that all credit card numbers typed into the database contain the correct number of digits.
- To make the applications and the database structure easy to understand.

Database logic is found in multiple places in RDBMSs, for example type information in create table statements and create domain statements; therefore it will be helpful if we can get all rules/logic in one format and place. In addition if we can provide a more independent format that can help transfers from one RDBMS to another of both metadata and data itself.

Not all RDBMSs fully support advanced SQL features such as create domain. Even if they do they may or may not support further features such as constraints within create domain or composite type. We illustrate this with a typical create table statement from a system that doesn't support domains as in Figure 1 below:

```
create table person (
        id serial,
        name char (25),
        building_no char (5),
        street char (20),
        town char (20),
        postcode char (25));
```

Figure 1 Person table creation without composite type

PostgreSQL now supports the creation of more structure in create table statements as illustrated below:

- Create structured type as example in Figure 2 below of creating address table as type of composite attributes [2]. We create an address-structured type via the route of creating a table. In most advanced RDBMSs table creation is equivalent to type creation [3]:

ISSN: 2356-5608

```
create table address(
        building_no char (5),
        street char (20),
        town char (20),
        postcode char (25));
```

Figure 1 Address table (and type) information

- Create a table that uses the address table as in the example below in figure 3. This shows how the address table can be used in another table as a type for the address column [2]:

```
create table person (
        id serial,
        name char (25),
        address address);
```

Figure 2 Person table creation using composite type

Figure 1 and Figure 3 representations may be seen as equivalent in that they both store the same data but arguably the form using the address type has greater semantics and would be preferable if this feature is supported.

Our aim is to provide rich representations in RuleML for the table information that can be used to create the richest table structure in any RDBMS, which also support the development of semantically richer forms.

II. PREVIOUS RELATED WORK

We discuss a number of approaches using metadata which have some limitations in terms of separation of logic and application.

A. Developing Web Entry Forms Based on Metadata

Elbibas et al. in [4] proposed an approach to develop and maintain HTML forms based on metadata extracted from a database table. The authors have used Java Database Connectivity (JDBC) [5] for accessing different databases. It included metadata features. Their proposed approach generates dynamic HTML forms, which have been generated and validated automatically. As the HTML is generated automatically on the fly, i.e., dynamic HTML, changes that are made to the database are reflected once the data is accessed again. Java and metadata were used to show help messages to the user to validate the input data. The set of rules of this scheme is embedded in the application code where it is difficult to locate and change their logic. In addition the set of rules does not support the manipulation of semantics of database metadata in some cases. So it is possible to develop domain specific rules to support the generic rules, as an example to deal with column names.

B. Using Database Metadata and its Semantics to Generate Automatic and Dynamic Web Entry Forms.

Elsheh et al. in [6] proposed a model which aims to generate dynamic Web entry forms based on metadata extracted from system tables. They used the Java servlet class to convert the extracted metadata via JDBC into an XML document. A set of rules has been developed and applied to database metadata which is used to map each column to specific user interface controls. In addition, the XML document is transformed into an XHTML document, using XSLT stylesheet, which is returned back to the user as Web entry form. Although XML used it differs from our approach which is using RuleML. This approach has the same problems encountered by [4] where the set of rules of this scheme is embedded in the application code where it is difficult to locate and change their logic. In addition the set of rules does not support the manipulation of semantics of database metadata in some cases. So it is possible to develop domain specific rules to support the generic rules, as example to deal with columns name.

C. Automatic Generation of Web User Interfaces in PHP using Database Metadata

Mgheder et al. in [7] suggested an approach that uses metadata stored in system tables in databases (columns name, type, size etc.) to develop generic user interface elements. They used PHP as the server script and the database abstraction library ADOdb to achieve their goal. The metadata is extracted from the database by using the ADOdb metadata methods. This metadata information combined with a developed set of rules is used to automatically map each column in the database table to a specific user interface control. The proposed model uses a set of rules which are extracted from the database to build the Web form; these rules are again built within the application code, where it is not easy to maintain them.

D. Approaches to Implementing Active Semantics with XML Schema

XML Schema uses elements and attributes to express semantics of XML data, but XML Schema does not have active elements. Bernauer et al. in [8] proposed an approach which implemented an Active XML Schema with XML Schema that defines active behaviour to enrich XML documents. Active XML Schema specifies active behaviour by using Event-Condition-Action rules, which automatically performs an action as reaction if a given condition applied. They do not use RuleML.

E. RuleML as Rulebase

RuleML provides a format for what is claimed [9] to be a natural form for human reasoning and behaviour, that is if-then-rules. However the individual rules need to be developed into a Rulebase. In a different domain to ours Schmidberger et al. in [9] have mentioned that there is no established standard rule format for industrial plant information reasoning available. They described an approach

which implements rulebase engineering of automation systems. The system was created especially for the automatic instantiation of Asset Management Functionalities and the automatic creation of interlocking control code. They have used a rule format based on a combination of RuleML and MathML elements in the logic part. Thus, in the context of rulebase automation of plant engineering tasks there will be a need for common description of such rules in a format, which is understood by humans and can be interpreted by a computer.

III. Application Development

This section introduces a mechanism, which aims to design a framework, using an XML format, to save database table's metadata in an external format using RuleML in order to support the creation of tables using domains as attribute types, and composite attributes, which consist of groups of values from more than one domain. This can be used with RuleML rulebases in order to generate automatic and dynamic Web forms. The proposed framework consists of several processes as shown in Figure 4. The following objectives are intended to be achieved.

- Store table's metadata in XML files. These files uses XML tags to describe the tables and it's columns information as:

<Rulebase><table><name> </name>
<column><name> </name>
<type> </type>
<size> </size>
<isnull> </isnull>
<unique> </unique>
<key> </key>
</column>
</table></Rulebase>

Each column is represented in a single XML node, and the empty tags could be included.

- Create database tables using the stored metadata for new database or recreate the existed database tables. To create the new tables a PHP script is used which reads the structure of the table stored in XML files. This script then creates the SQL script which actually creates the table in RDBMS.
- Apply Rulebase in conjunction with the metadata of each column stored in XML file to map each column to the correct Web entry control element.
- Generate Web form element.

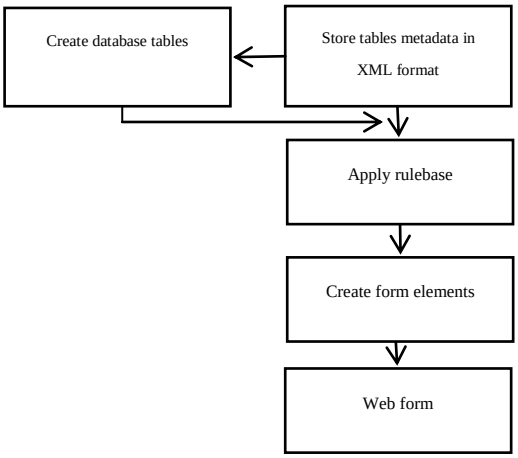


Figure 4 Framework mechanism

To illustrate this mechanism and investigate if there are any difficulties in implementing it, the following sections introduce an example of the implementation of this approach.

A. Table's metadata in XML files for table creation

A database schema is represented in RuleML file. This RuleML information uses XML tags to describe the tables, columns, and rows. It is used for modelling database information, so the previous structure of composite attributes or domains could be represented in XML tags as in Figure 5 and Figure 6 below:

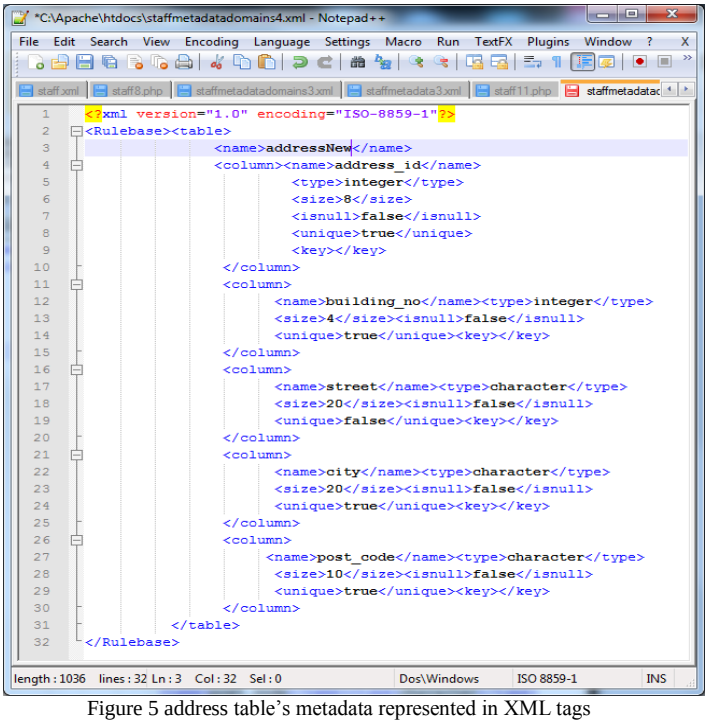


Figure 5 address table's metadata represented in XML tags

Figure 6 staff table's metadata represented in XML tags

B. Database tables creation

To create new tables a PHP script is used to read the structure of the table stored in XML files as in Figure 5, Figure 6. This script then creates the SQL script as shown in Figure 7, which actually creates the tables in the RDBMS.

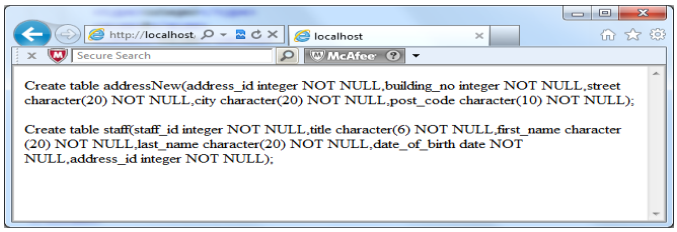


Figure 7 SQL script created dynamically using table's metadata stored in XML files

As a result of the created SQL script the tables originally specified in the XML file will be created as below:

```
CREATE TABLE addressnew (
address_id integer NOT NULL,
 building_no integer NOT NULL,
street character(20) NOT NULL,
city character(20) NOT NULL,
post_code character(10) NOT NULL);

CREATE TABLE staff (
staff_id integer NOT NULL,
title character(6) NOT NULL,
first_name character(20) NOT NULL,
last_name character(20) NOT NULL,
date_of_birth date NOT NULL,
address_id integer NOT NULL);
```

C. Existing table's metadata stored as XML format

In this section we address how to store a table's metadata in an XML format, particularly for systems that do not support domains and composite attributes. Database metadata can be represented in a XML file, this XML file uses XML tags to describe the tables and columns metadata, it is for modeling database information, so the metadata is stored into XML format.

1. Staff table metadata stored in XML format

The database metadata is stored in a XML format in separate files, as the example used in the prototype implementation the staff table metadata stored in XML file as shown in Figure 8. The XML file includes all the required information to (re) create the table in an RDBMSs whether it support domains or not. The tags organized to specify each column's metadata in separate column tags. As shown in Figure 8 below the table staff consists of 8 columns the last two columns are created using domains, each column refers to a separate domain as below:

```
<column>
    <name>address</name>
    <type>domain</type>
</column>
<column>
    <name>Branch</name>
    <type>domain</type>
</column>
```

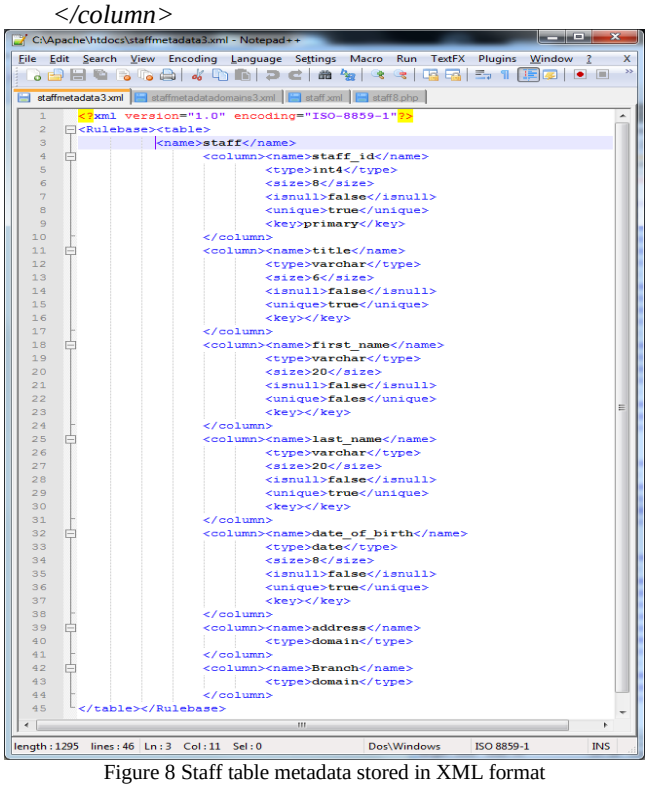
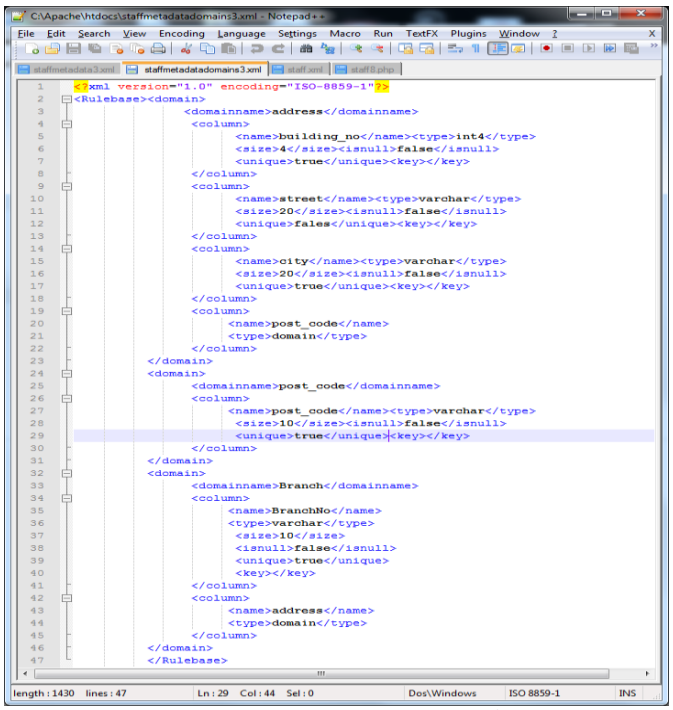


Figure 8 Staff table metadata stored in XML format

2. Domain tables metadata in XML format

The database domain's metadata and the structure of the composite attributes are stored in XML format as shown in Figure 9. Each domain in the previous XML file shown in Figure 8 is connected with the XML domains file shown in Figure 9. A domain can be used inside another one as shown in the address that contains a postcode column which is itself a domain. The structure of the post code column is also included in the domains file.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rulebase><domain>
        <domainname>address</domainname>
        <column>
            <name>building_no</name><type>int4</type>
            <size>4</size><isnull>false</isnull>
            <unique>true</unique><key></key>
        </column>
        <column>
            <name>street</name><type>varchar</type>
            <size>20</size><isnull>false</isnull>
            <unique>fales</unique><key></key>
        </column>
        <column>
            <name>city</name><type>varchar</type>
            <size>20</size><isnull>false</isnull>
            <unique>true</unique><key></key>
        </column>
        <column>
            <name>post_code</name>
            <type>domain</type>
        </column>
    </domain>
    <domain>
        <domainname>post_code</domainname>
        <column>
            <name>post_code</name><type>varchar</type>
            <size>10</size><isnull>false</isnull>
            <unique>true</unique><key></key>
        </column>
    </domain>
    <domain>
        <domainname>Branch</domainname>
        <column>
            <name>BranchNo</name>
            <type>varchar</type>
            <size>10</size>
            <isnull>false</isnull>
            <unique>true</unique>
            <key></key>
        </column>
        <column>
            <name>address</name>
            <type>domain</type>
        </column>
    </domain>
    </Rulebase>
```

Figure 9 Domain tables metadata in XML format

3. Generate the Web forms

We now demonstrate the use of the XML metadata format to generate the Web forms. By using the stored metadata files in conjunction with the RuleML Rulebase, PHP script is written to loop through all the metadata for each column in every table and uses the RuleML rulebase to map each column to a Web form element, and the required fields label generated from the <isnull>false</isnull> on the fly.

From the generated Web form which is shown in Figure 10 we can see how the composite columns' attributes are generated by using the domain address table and how the domain table can be used many times. Figure 10 shows the result of using address domain table twice within one form, the first one is to generate the staff's address elements and the second one is to generate the branch address elements using the same domain. Additionally within each address the post code is itself another domain.

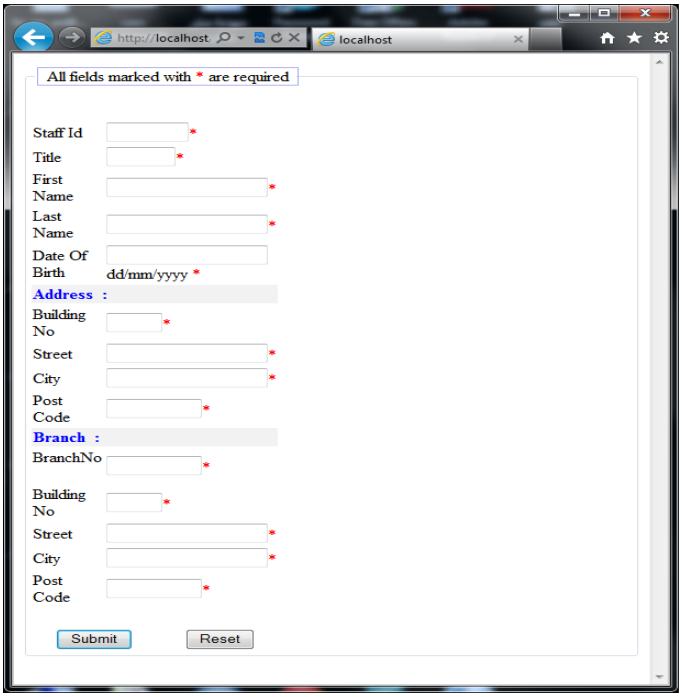


Figure 10 User interface form generated automatically using metadata stored as XML format and rulebase as RuleML format.

IV. Conclusion

XML Schema uses XML elements and attributes to express the structure of XML data, which may be comparable to RDBMS data, but XML Schema does not do everything. It can be used to express some limitations of data such as possible ranges of values and characteristics such as uniqueness. It does not have active elements which would allow us to express more behavior; however these can be found in an XML format in RuleML's Event-Condition-Action like elements.

To overcome some RDBMSs limitations RuleML is used to represent RDBMS data by storing database metadata in an external format, so it is also a way to overcome the differences between RDBMSs in areas such as whether they support domains and composites. Thus we propose a way to give a single syntax that can then map them to structures supported by a particular RDBMS and we test this by producing the same result for the Web form.

As a result a Web form for user interface is generated dynamically that corresponds to the database being used and at the same time maximizes the use of semantics in metadata or elsewhere. RuleML can also drive automatic javaScript validation code as well as form layout.

So XML Schema alone is not sufficient but by using a RuleML format we can go one stage further to implement more semantics for both database structures themselves and the Web forms built dynamically to access them.

REFERENCES

[1] Using domains. Available: http://dcx.sybase.com/1200/en/dbusage/domains-integrit.html. [Accessed: 12 Dec. 2015].

[2] PostgreSQL extended or object relational features. Available: http://www.comp.brad.ac.uk/~postgres/postgreSQL/worksheet5.html. [Accessed: 12 Dec. 2015].

[3] PostgreSQL: The world's most advanced open source database. Available: http://www.postgresql.org/. [Accessed: 12 Dec. 2015].

[4] A. Elbibas and M. J. Ridley, "Developing Web Entry Forms Based on METADATA," Presented at International Workshop on Web Quality in conjunction with ICWE 04- International Conference on Web Engineering 2004. Available: http://www.pst.informatik.uni-muenchen.de

[5] Understanding JDBC Metadata. Available: http://www.databaseskill.com/1323105/. [Accessed: 9 Aug. 2015].

[6] M. M. Elsheh and M. J. Ridley, "Using Database Metadata and its semantics to Generate Automatic and Dynamic Web Entry Forms," in Proceedings of WCECS 2007 World Congress on Engineering and Computer Science 2007, pp. 654-658.

[7] M. A. Mgheder and M. J. Ridley, "Automatic Generation of Web User Interfaces in PHP Using Database Metadata," in Proceedings of Internet and Web Applications and Services, 2008. ICIW '08. Third International Conference on, 2008, pp. 426-430.

[8] M. Bernauer, G. Kappel, and G. Kramler, "Approaches to implementing active semantics with XML schema," in Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on, 2003, pp. 559-565.

[9] T. Schmidberger and A. Fay, "A rule format for industrial plant information reasoning," in Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, 2007, pp. 360-367.