Volume 6 • Issue 2 • July / August 2015

INTERNATIONAL JOURNAL OF

DATA ENGINEERING (IJDE)

ISSN : 2180-1274
Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF DATA ENGINEERING (IJDE)

VOLUME 6, ISSUE 2, 2015

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180-1274

International Journal of Data Engineering is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJDE Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF DATA ENGINEERING (IJDE)

Book: Volume 6, Issue 2, July / August 2015
Publishing Date: 31-08-2015
ISSN (Online): 2180-1274

IJDE Journal is a part of CSC Publishers
http://www.cscjournals.org

© IJDE Journal
Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2015

EDITORIAL PREFACE

This is *Second* Issue of Volume *Six* of the International Journal of Data Engineering (IJDE). IJDE is an International refereed journal for publication of current research in Data Engineering technologies. IJDE publishes research papers dealing primarily with the technological aspects of Data Engineering in new and emerging technologies. Publications of IJDE are beneficial for researchers, academics, scholars, advanced students, practitioners, and those seeking an update on current experience, state of the art research theories and future prospects in relation to computer science in general but specific to computer security studies. Some important topics cover by IJDE is Annotation and Data Curation, Data Engineering, Data Mining and Knowledge Discovery, Query Processing in Databases and Semantic Web etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 6, 2015, IJDE appear with more focused issues. Besides normal publications, IJDE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

This journal publishes new dissertations and state of the art research to target its readership that not only includes researchers, industrialists and scientist but also advanced students and practitioners. The aim of IJDE is to publish research which is not only technically proficient, but contains innovation or information for our international readers. In order to position IJDE as one of the top International journal in Data Engineering, a group of highly valuable and senior International scholars are serving its Editorial Board who ensures that each issue must publish qualitative research articles from International research communities relevant to Data Engineering fields.

IJDE editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJDE. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJDE provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts..

Editorial Board Members
International Journal of Data Engineering (IJDE)

EDITORIAL BOARD

Dr. Andrey Balmin
IBM Almaden Research Center
United States of America

Dr. Rishi R. Sinha
Microsoft Corporation
United States of America

Dr. Qiong Luo
Hong Kong University of Science and Technology
China

Dr. Thanaa M. Ghanem
University of St. Thomas
United States of America

Dr. Ravi Ramamurthy
Microsoft Research
United States of America

Dr. David DeHaan
Sybase
Canada

Dr. Theodore Dalamagas
IMIS, Athens
Greece

Dr Moustafa Hammad
Google, Inc.
United States of America

TABLE OF CONTENTS

Volume 6, Issue 2, July / August 2015

Catalog-based Conversion from Relational Database into XML Schema (XSD)

Husam Ahmed Al Hamad[1, 2] *hhamad@qu.edu.sa, hushamad@yahoo.com*
[1]*Department of Information Technology,*
College of Computer, Qassim University,
Qassim, Saudi Arabia
[2]*Mobile Computing Department,*
Computer Science and Informatics College, Amman Arab University,
Amman, Jordan

Abstract

Where we are in the age of information revolution, exchange information, and transport data effectively among various sectors of government, commercial, service and industrial, etc., the uses of a new databases model to support this trend has become very important because inability of traditional databases models to support it. eXtensible Markup Language (XML) considers a new standard model for data interchange through internet and mobiles devices networks, it has become a common language to exchange and share the data of traditional models in easy and inexpensive ways. In this research, we propose a new technique to convert the relational database contents and schema into XML schema (XSD- XML Schema Definition), the main idea of the technique is extracting relational database catalog using Structured Query Language (SQL). We follow three steps to complete the conversion process. First, extracting relation instance (actual content) and schema catalog using SQL query language, which consider the required information to implement XML document and its schema. Second, transform the actual content into XML document tree. The idea of this step is converting table columns of the relations (tables) into the elements of XML document. Third, transform schema catalog into XML schema for describing the structure of the XML document. To do so, we transform datatype of the elements and the variant data constrains such as data length, not null, check and default, moreover define primary foreign keys and the referential integrity between the tables. Overall results of the technique are very promise while the technique is very clear and does not require complex procedures that could adversely effect on the results accuracy. We performed many experiments and report their elapsed CPU times.

Keywords: Conversion Relational Schema into XML Schema, Transformation Schema, XSD Schema, XML Schema.

1. INTRODUCTION

XML is one such innovative usage of relational database prompted by increasing the usage of organizations database applications and its related need of managing frequent storage and retrieval of not-very structured data in document format. XML database becomes an important database structure in presenting storing, and exchanging data through internet and mobile systems. It makes the message self-documenting by presenting of the elements.

A schema does not need expert to understand the meaning of the text. The format of XML document is not rigid; easily we can add an additional information using the elements. In addition, we can ignore any information or element. In other words, the ability to recognize and ignore unexpected elements allows the format of the data to evolve over time, without invalidating existing applications. Similarly, the ability to have multiple occurrences of the same element makes it easy to represent multivalued attributes. Likewise, XML allows nested structures, and a wide variety of tools are available to assist in XML processing, including programming language to create and to read XML data, browser software, and database tools [1].

XML is designed to describe data using tags (elements) with focus on what data is, by store it in plain text format, this makes it much easier to share data between different applications and read by different incompatible applications. XML allows expressing information in ways that match better for business. XML allows us to model information systems in natural and intuitive way. It brings a number of powerful capabilities to information modeling such as heterogeneity, extensibility, and flexibility. For these reasons, XML becomes a standard data format widely used in these organizations and a common language for data transmission over the internet. Numerous of languages such as Document Type Definition (DTD) is also used for restructuring the XML documents [2, 3]. Many papers used and focused on DTD or XML tree to expand the elements, attributes and data by extend the notion of Functional Dependency (FD) and compare the values of leaf nodes in a specified context of its corresponding XML tree to form an integrated XML tree [3]. eXtensible Stylesheet Language Transformations (XSLT) can be used for creating a mediate architecture of XML schemas [4, 5] as well.

This research develops a technique to convert a relational database schema catalog into XML schema (XSD) database. The catalog of relational schema contains schema structure of the database such as tables name, relationship, keys, constraints, etc. We use SQL for extracting the database schema catalog and relation instance (the actual content of the database at a particular point in time). We propose extracting relational schema catalog for representing the conceptual schema of XML model, this style considers simple and contains the required schema data and content. In our approach, after extracting actual content and schema catalog, we transform actual content into XML document tree to represent the elements values, we also transform catalog schema into XML schema document tree (XSD) to represent the schema design and constraints; Figure 1 illustrates the architecture of converting a relational database into an XML document.

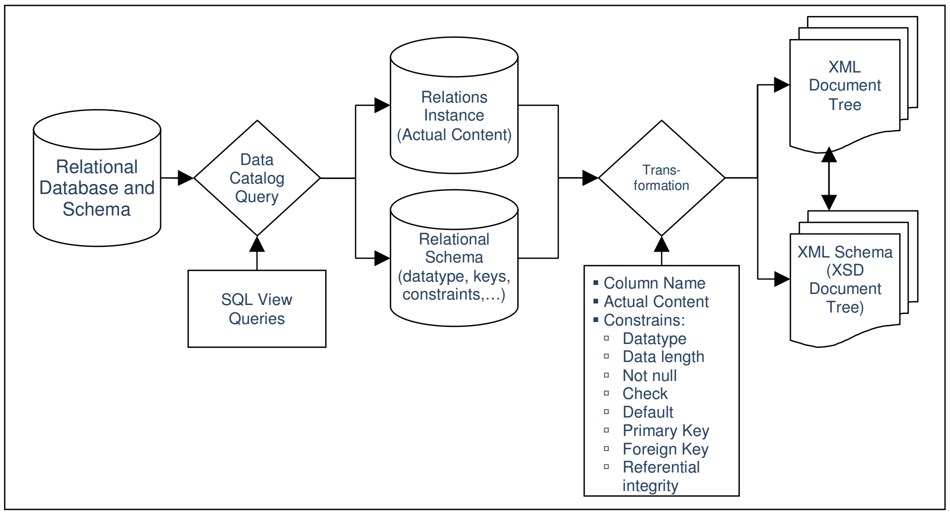

FIGURE 1: Architecture of converting a relational database into an XML document.

Database catalog contains the relations names (tables names), attributes names (table columns names), and constraints of the database schema such as data type, primary and foreign keys, referential integrity, check, default, etc. Relation instance contains actual content of each table columns names. We transform the extracted information into two documents XML document tree and XML schema document tree. We use three SQL view queries statements to extract the required information of the catalog. We transform table name, table columns names, and actual content of all tables into XML document tree, XML document contains elements, sub-elements, and attributes if needed, we mapped table name as root element of the document, mapped table columns names as elements of the document, and finally mapped the actual content as values of the document elements. Likewise, we transform the data type, primary and foreign keys, referential integrity, check, default and the other restrictions and constraints of the relational schema catalog into XSD document tree, we used the variant XSD elements to represent the restrictions and constraints of the relational schema, as we will see later.

2. RELATED WORKS

XML data is self-describing, where XML tags describe the data itself, it is suitable for interpreting the data and programing. This means that a program receiving an XML document can interpret it in multiple way, filter the document based upon its content and restructure it to suit the application's needs [6]. Ye Feng et al. [7] maps XML-DTD to Relational Schema by using Absolute Data Group (ADG) technique, then optimize the ADG convert it to relational schema. Chunyan Wang et al. [8] addresses both catalog-based and legacy relational databases; it uses catalog for applying the reverse engineering approach to extract the ER (Extended Entity Relationship) model from legacy relational databases. Then, the technique converts the ER to XML schema. Teng Lv et al. [9] converts schema from relational schemas to XML-DTDs and preserve the semantics implied by functional dependencies and keys of relational schemas.

Yan-Feng Zhang et al. [10] converts XML Schema to UML diagram, the integration process includes three steps: clustering of concepts, unification of concepts, restructuring of relationships, and provided a global conceptual model for users. Tzvetkov et al. [11] connects XML with relational databases and converted data both ways -from XML-schema to relational database schema and from relational database schema to XML-schema. Sungchul Hong et al. [12] converts XML to relational data model and relational data model to XML, they used a virtual collaboration system to store the in a single XML file. Ye Feng [13] converts XML-DTD to Relational Schema, the algorithm accesses elements, attributes, and relationship of elements, it creates the DTD graph to express the elements, attributes and semantic constraints of XML DTD, then optimizes the DTD graph, and converts DTD to relational schema.

Fong [14] uses XML-based technique for integration between relational schema and XML schema. The technique consists four different types: (1) functional dependency; (2) multi-valued dependency; (3) join dependency; and (4) M: N cardinality. Fong [15] translates a relational schema to an XML schema, the mechanism applies the Indirect Schema Translation Method for translating Extended Entity Relationship (EER) to an XML Schema (XSD) Graph. Then mapping XSD Graph into the XSD as an XML logical schema. VXMLR [16] is a visual based XML document management system, it is parses XML document into a DOM tree and extracts the DTD of the document then map the document tree into a relational table.

3. TECHNIQUE DETAILS

An XML Schema (XSD) describes the structure of an XML document. We choose XSD because it is much more powerful than DTDs. XML Schema is extensible, because it is written in XML format. It describes relationship among elements and data type of each element. It defines structure and content as well as semantics that can be described in an XSD document.

The proposed method in this research introduces a new technique for converting a relational database catalog into XML database and XML schema. The first step extracts relation instance and schema catalog of the database, relation instance contains the actual content of the relations while schema catalog contains all features and components of relational database schema such as table columns, datatype, keys, constraints, etc.,. In this process, we use SQL query language to extract the required information. The second step maps relation instance of the database into XML document tree and determines the table columns. The third step maps schema catalog into XSD document and define the datatype, keys, references and all other constraints.

For clarifying the idea, we use a simple Registration Application database. In this application, a student register several courses via register, and a course registered by one or several students via register. In addition, a semester contains several courses via register, and a course offered in one or several semesters via register relation. Figure 2 illustrates database schema of the registration application example. Figure 3 illustrates the relation instance (actual content) of each relation. The following processes are outlines of the research methodology.

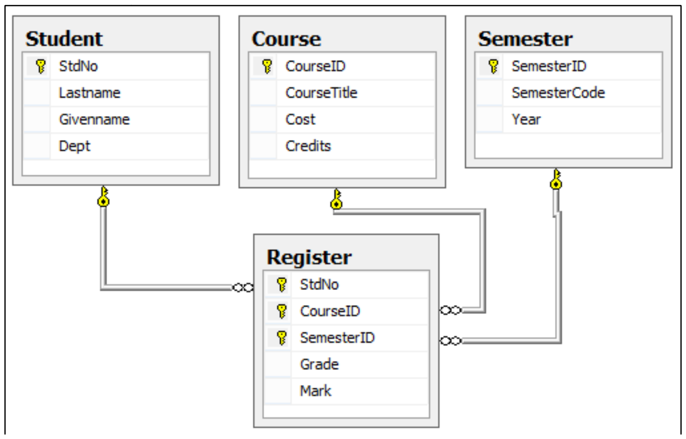

FIGURE 2: Database schema of a simple Registration Application.

Student Relation

StdNo	Lastname	Givenname	Dept
S0001	Marwan	Khaled	CS
S0201	Mosa	Ahmad Majd	IT
S0211	Alnasser	Ameen	CS
S0421	Mohammad	Omar Riyadh	IT
S0711	Alsaleh	AbdulazizSaleh	CS

Course Relation

CourseID	CourseTitle	Cost	Credits
CSC152	C Programming	8200.5	4
CSC153	Object Oriented Programming	8480	4
IT125	Database	6435.5	4
IT224	Java Programming	7680	3
IT326	Data Mining	5340	3

Semester Relation

SemesterID	SemesterCode	Year
SM01	1	2012
SM02	2	2012
SM03	1	2013
SM04	2	2013

Register Relation

StdNo	CourseID	SemesterID	Grade	Mark
S0001	CSC152	SM02	A	92
S0001	IT125	SM01	A+	96
S0201	CSC153	SM01	NULL	NULL
S0201	IT125	SM02	B+	87
S0201	IT224	SM01	A	92
S0211	CSC153	SM01	NULL	NULL
S0421	IT125	SM02	B	84
S0421	IT224	SM02	F	57
S0711	CSC152	SM01	F	55
S0711	CSC152	SM03	C+	77

FIGURE 3: Relation instance (actual content) of all tables.

The three major steps transform the actual content and database schema of relational database into XML and XML schema (XSD) document are:

Step 1: Extracting Schema Catalog and Relation Instance

In general, data structures format of relational schema is different from XML and XSD documents, XML and XSD represent hierarchical tree structure; their implementation is based on root elements. The sub-elements under the root must be relevant to the root element. XML root element represents the table's name of relational schema; XML sub-elements represent the table columns of the relational schema. Two phases for extracting the required information. First, extracting actual content that contains table's name and domains of the table columns, which represent elements and attributes in an XML document. Second, extracting schema catalog that contains table's constraints such as primary and foreign keys, cardinality constraint, datatype name, length, check, and other constraints, these constraints will transform and included into XML schema.

Different SQL query statements return information of instance (actual data) and schema (database catalog). For example, we use a simple SQL statement to extract the actual content of the tables "`select * from table_name`". Likewise, in SQL Server, we use SQL view query statement "`INFORMATION_SCHEMA.COLUMNS`" to extract tables' names, tables' columns names, datatype, null constraint, and maximum length, Table 1 illustrates the output of this view for the "Registration Application" database.

TABLE_NAME	**COLUMN_NAME**	**DATA_TYPE**	**IS_NULL-ABLE**	**CHAR-ACTER_MAX-IMUM_LENGTH**
Student	StdNo	char	NO	5
Student	Lastname	varchar	NO	25
Student	Givenname	varchar	NO	50
Student	Dept	char	YES	4
Course	CourseID	char	NO	8
Course	CourseTitle	varchar	NO	50
Course	Cost	decimal	YES	NULL
Course	Credits	int	YES	NULL
Semester	SemesterID	char	NO	5
Semester	SemesterCode	int	YES	NULL
Semester	Year	int	YES	NULL
Register	StdNo	char	NO	5
Register	CourseID	char	NO	8
Register	SemesterID	char	NO	5
Register	Grade	char	YES	2
Register	Mark	decimal	YES	NULL

TABLE 1: Output of the view for the "Registration Application" "`INFORMATION_SCHEMA.COLUMNS`".

In addition, we use SQL views statement "`information_schema.TABLE_CONSTRAINTS`", "`INFORMATION_SCHEMA.KEY_COLUMN_USAGE`", and "`sp_fkeys table`" to extract the relationship between the tables, the relationships contain Primary and Foreign keys of each table, as well as Check and Unique constrains and their details. Table 2 illustrates the output of the views "`information..CONSTRAINTS`", Table 3 illustrates the output of foreign key constraint "`sp_fkeys`" for the "Registration Application" database.

TABLE NAME	**CONSTRAINT_TYPE**	**COLUMN_NAME**	**CONSTRAINT_Details**
Student	PRIMARY KEY	StdNo	
Course	PRIMARY KEY	CourseID	
Course	UNIQUE	CourseTitle	
Course	CHECK	Cost	cost>=0
Course	CHECK	Credits	Credits between 0 and 200) **Default 2**
Semester	PRIMARY KEY	SemesterID	
Semester	CHECK	SemesterCode	SemesterCode between 1 and 4
Semester	CHECK	Year	Year between 2000 and 9999
Register	PRIMARY KEY	StdNo, CourseID,	

TABLE NAME	CONSTRAINT_TYPE	COLUMN_NAME	CONSTRAINT_Details
		SemesterID	
Register	FOREIGN KEY	StdNo	
Register	FOREIGN KEY	CourseID	
Register	FOREIGN KEY	SemesterID	
Register	CHECK	Mark	Mark between 0.00 and 100.00

TABLE 2: Output of the views "information..CONSTRAINTS".

PKTABLE_NAME	PKCOLUMN_NAME	FKTABLE_NAME	FKCOLUMN_NAME
Student	StdNo	Register	StdNo
Course	CourseID	Register	CourseID
Semester	SemesterID	Register	SemesterID

TABLE 3: Output of the view "sp_fkeys".

Step 2: Mapping Relation Instance into XML document
Structure of XML document contains a root element, elements and attributers; root element represents "the parent" of all other elements, the elements in the XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree. All elements can have sub-elements (child elements). All elements can have text content and attributes. In order to create a XML document tree including the extracted information of the first step, we should identify the root element, branches elements, and their attributes.

We transform the actual content of the relational schema into XML document tree, which process by mapping the table name with root element of the XML document, and defining XML namespace, schema namespace, and location of the schema. Thereafter, we transform all columns of the table into branches elements of the XML document tree, as well as the actual content of each branch element. Thus, XML document contains only elements names and their actual content without consider any constraint, keys, datatype, or references, which will be defined in the XSD schema.

For more clarification, we transform "Student" table to illustrate mapping of root element of the XML document. We convert columns names of the table to illustrate mapping of branch element of the XML document.

List 1 shows transformation of "Student" table into XML document root. List 2 shows transform of "Register" table into XML document root that contains a composite primary key. Likewise, we transform other tables "Course", "Semester".

LIST 1: Transform "Student" table into XML tree.

```
<?xml version="1.0"?>
<Student
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.qu.eud.com student.xsd ">
 <tuple>
  <StdNo>S0001</StdNo>
  <Lastname>Marwan</Lastname>
  <Givenname>Khaled</Givenname>
  <Dept>CS</Dept>
 </tuple>
 <tuple>
  <StdNo>S0201</StdNo>
  <Lastname>Mosa</Lastname>
  <Givenname>Ahmad Majd</Givenname>
  <Dept>IT</Dept>
 </tuple>
...
</Student>
```

LIST 2: Transform "`Register`" table into XML tree.

```
<?xml version="1.0"?>
<Register
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.qu.eud.com register.xsd ">
 <tuple>
  <StdNo>S0001</StdNo>
  <CourseID>CSC152</CourseID>
  <SemesterID>SM02</SemesterID>
  <Grade>A</Grade>
  <Mark>92</Mark>
 </tuple>
 <tuple>
  <StdNo>S0001</StdNo>
  <CourseID>IT125</CourseID>
  <SemesterID>SM01</SemesterID>
  <Grade>A+</Grade>
  <Mark>96</Mark>
 </tuple>
...
</Register>
```

Step 3: Mapping Schema Catalog into XML Schema (XSD) Document
We integrate constraints and restrictions of the relational schema database source into the XML schema tree; we transform schema catalog that contains the constraints of the relational schema into XSD document hierarchical tree. When XML document parser reads XSD document, it creates a document object first, and then complete the whole XML document from this point. Using XSD, the technique transforms every element node not only the structure of roots and branches relationship, but also their actuarial values as well.

We represent the tables in XML Schemas by complex elements, and the table columns as sub-elements. Each table owns several keys such as primary, foreign and composite keys. In XML document, each key specifies as element (tag), where the keys used in XML query such as XPath or XQuery to specify a node of the element. The name of the table is equal to the name of the root element. The names of the table columns are equal to the names of the sub-elements and attributes that make up the complex element. For example, XML schema in List 3 could represent a table named "`Student`". The elements "`StdNo`" to "`Dept`" represent the table columns that make up the table. "`StdNo`" column is defined as a primary key; we use `<sequence>` indicator to specify the child elements must appear in a specific order. Conversely, `<all>` indicator specifies the child elements can appear in any order, which is not used for updating issues. In the same example, we use the sub-element tag `<xs:element name="StdNo"/>` to define the column name "`StdNo`", then add the sub-element tag `<xs:simpleType>` to define the restrictions of the table column, the "`simpleType`" element specifies the constraints and information about the values of attributes or text-only elements. List 3 to List 6 contain the complete mapping of the relational schema "Registration Application" database for the tables "`Student`", "`Course`", "`Semester`", and "`Register`". For more clarification of XML schema document mapping, we split up the schema transformation into three parts as below.

1. Transform Datatype of The Elements and Attributes:
XML Schema supports many data types such as string, decimal, integer, Boolean, date, time, etc. For Example, the attribute type "`string`" refers to a simple type that is built-in to all XML Schemas. Each element or attribute of XML document has a data type defined as mentioned before by the XML schema, data type considers restrictions on the element's or attribute's content. Fox example, in "`Student`" table, "`StdNo`" data type must be "`string`", we use "`base`" attribute in "`restriction`" element to represent the datatype in the XML schema by add the sub-

element tag restriction `<xs:restriction base="xs:string">` branch of `<xs:simpleType>`, which is branch of `<xs:element name="StdNo"/>`. For example, if the element type is "`xs:date`" and the content a string like "Java Programming", the element will not validate. XML schemas allow to add an own restrictions for the elements and attributes.

2. Transform the Constraints: data length, not null, check and default:

We define different restrictions in XML schema to represent null, length values, and check constraints. In the extracted catalog of the relational schema.

If the column has an exact length, this length should not less than or greater than a specific amount. In this case, we use "`length`" element restriction to limit the exact length constraint of the element in XSD schema. For example, in "`Student`" table, "`StdNo`" column value must be exactly eight characters, we use "`value`" attribute in "`length`" restriction element to represent this constraint in XML schema, we add the sub-element tag restriction `<xs:length value="5"/>` as a branch of `<xs:restriction base="xs:string">`.

If there are minimum and/or maximum characters of the column domain. In this case, we use "`minLength`" and "`maxLength`" elements restrictions to define min and max lengths of the element in XSD schema. For example, in "`Student`" table, "`Lastname`" column value must be less than 25 characters, we use "`value`" attribute in "`maxlength`" restriction element to represent this constraint, we add the sub-element tag restriction `<xs:maxLength value="25"/>` as a branch of `<xs:restriction base="xs:string">`.

If the column domain is not null, such as primary key of the tables or any column defined as required in the relational schema. In this case, we use "`use`" element restriction in XSD schema. For example, in "`Student`" table, all of "`StdNo`", "`Lastname`" and "`Givenname`" columns must not null, we use "`value`" attribute in "`use`" element restriction to make the element is required (not null), we add the sub-element tag restriction `<xs:use value="required"/>` as a branch of `<xs:restriction base="xs:string">`.

In XSD schema, there are another method to define null value, the method uses "`minOccurs`" element restriction to mention that the column should not be null (required). "`minOccurs`" element restriction specifies the minimum number of times an element can occur, whereas the elements with the "`minOccurs="0"`" means each element cad appear zero (null) or one time. Anyway, we do not use this method because it is more complex than using "`use`" element restriction.

If there are other constraints in relational schema that use Check clause in the SQL, such as, the value of the column should equal, less than or greater than something. In this case, we use "`minInclusive`" and "`maxInclusive`" elements restrictions in XSD schema. For example, in "`Course`" table, "`Credits`" column should between 0 and 20 (greater than or equal 0), we use "`value`" attribute in "`minInclusive`" and "`maxInclusive`" elements restrictions to define this restriction, we add the sub-elements tag restrictions `<xs:minInclusive value="0"/>` and `<xs:maxInclusive value="200"/>` as a branch of `<xs:restriction base="xs:integer">`.

If the column domain contains a "`default`" value constraint with any amount, such as, the default value of the column is two. In this case, we use "`default`" element restriction in XSD schema. For example, in "`Course`" table, the default value of "`Credits`" column is 2, we use "`value`" attribute in "`default`" element restriction to define this restriction, we add the sub-elements tag restrictions `<xs: default value="2"/>` as a branch of `<xs:restriction base="xs:integer">`.

3. Transform primary key, foreign key and unique

In the process, as mentioned before, relational schema catalog recovers key constraints in the relational schema database. In this respect, we identify the primary key, composite key and foreign key of each table in the database and classify their referential integrity in terms of data constraints. Transform key constraints of the XML schema tree is very important while it allows better inquiry through XML documents.

We define primary keys for each table using "`key`" element, the key in XSD schema is allows unique, non-nullable, "`name`" attribute specifies the unique name of the key in the schema, we can specify an optional XSD attribute "`PrimaryKey`" for more declaration. For example, in "`Student`" table, "`StdNo`" column is the primary key of the table, we use "`key`" element to define the key of the table, and add "`name`" attribute to specify the name of that key in the schema `<xs:key name="StudentPK" PrimaryKey="true">`. We also use "`XPath`" attribute in "`selector`" sub-element to specify an Xpath expression selects reference of the key that belong to (the parent), the full element tag is `<xs:selector xpath=".//Student"/>`. We also use "`XPath`" attribute in "`field`" sub-element to specify an Xpath expression determines the table column that the key belong to, the full element tag is `<xs:field xpath="StdNo"/>`

The same idea for the composite primary key, the difference in this case is add "`field`" sub-element in the same number of column the composite attribute. For example, In "`Register`" table, the composite primary key is splitting up to three columns "`StdNo`", "`CourseID`", and "`SemesterID`". We define the "`key`" element `<xs:key name="RegisterPK" PrimaryKey="true">` and the reference of the key using `<xs:selector xpath=".//Register"/>`. We add three "`field`" sub-element with their attributes "`xpath`" to determine the primary key, for "`StdNo`" the sub-element is `<xs:field xpath="StdNo"/>`, for "`CourseID`" the sub-element is `<xs:field xpath="CourseID"/>`, for "`SemesterID`" the sub-element is `<xs:field xpath="SemesterID"/>`. The three elements should specify the same parent before enforcing the composite primary key.

In the some way, we specify the unique constraints, unique can allow null values, whereas primary key constraints do not allow null values. For example, in "`Course`" table, "`CourseTitle`" column is unique column, we use "`unique`" element to specify the unique constraint, we add "`name`" attribute to specify the name of unique element in the schema `<xs:unique name="CourseUnique">`. We also use "`XPath`" attribute in "`selector`" sub-element to specify an Xpath expression selects reference of the unique element belong to (the parent), the full element tag is `<xs:selector xpath=".//Course"/>`. We also use "`XPath`" attribute in "`field`" sub-element to specify an Xpath expression determines the table column that the key belong to, the full element tag is `<xs:field xpath="CourseTitle"/>`

To define the referential integrity between the tables, we define an element as a foreign key in a table associates a primary key in another table. To do so, we use "`keyref`" constraint to define the referential integrity; we use "`selector`" and "`field`" elements for the same purpose in the definition of primary key. For example, In "`Register`" table, there are composite foreign key contains three columns, these columns associate three tables "`Student`", "`Course`", and "`Semester`". We define attributes "`name`" and "`refer`" in the "`keyref`" element, "`name`" attribute specify the name of that foreign key in the schema, since "refer" attribute references to its primary key `<xs:keyref name="Regester_Student" refer="StudentPK">`. We use "`XPath`" attribute in "`selector`" sub-element to specify an Xpath expression selects reference of the foreign key element that belong to (the parent) `<xs:selector xpath=".//Register"/>`. We also use "`XPath`" attribute in "`field`" sub-element to specify an Xpath expression determines the table column that the key belong to, the full element tag is `<xs:field xpath="StdNo"/` , we define the rest parts of the composite foreign keys "`CourseID`" and "`SemesterID`" in the same way. List 6 illustrates transform "`Registrar`" table into XSD document tree that contains a composite foreign key assassinated with three other tables.

List 3 to List 6 contain the complete mapping of the relational schema for the tables "`Student`", "`Course`", "`Semester`", and "`Register`".

List 3: Transform "`Student`" table into XSD document tree.

```
<xs:element name="UniversityDB">
 <xs:complexType>
  <xs:element name="Student">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="StdNo" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:length value="5"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Lastname" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string" >
        <xs:maxLength value="25"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Givenname" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:maxLength value="50"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Dept"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:maxLength value="4"/>
       </xs:restriction>
      </xs:simpleType>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:complexType>
 <xs:key name="StudentPK" PrimaryKey="true">
  <xs:selector xpath=".//Student"/>
   <xs:field xpath="StdNo"/>
 </xs:key>
</xs:element>
```

LIST 4: Transform "`Course`" table into XSD document tree.

```
<xs:element name="UniversityDB">
 <xs:complexType>
  <xs:element name="Course">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="CourseID" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:length value="8"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="CourseTitle" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string" >
        <xs:maxLength value="50"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Cost"/>
      <xs:simpleType>
       <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Credits" default="2" />
      <xs:simpleType>
       <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="200"/>
       </xs:restriction>
      </xs:simpleType>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:complexType>
 <xs:key name="CoursePK" PrimaryKey="true">
  <xs:selector xpath=".//Course"/>
   <xs:field xpath="CourseID"/>
 </xs:key>
<xs:Unique name="CourseUnique">
  <xs:selector xpath=".//Course"/>
   <xs:field xpath="CourseTitle"/>
 </xs:Unique>
</xs:element>
```

LIST 5: Transform "`Semester`" table into XSD document tree.

```
<xs:element name="UniversityDB">
 <xs:complexType>
  <xs:element name="Semester">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="SemesterID" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:length value="5"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="SemesterCode" />
      <xs:simpleType>
       <xs:restriction base="xs:integer" >
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="4"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Year"/>
      <xs:simpleType>
       <xs:restriction base="xs:integer">
        <xs:minInclusive value="2000"/>
        <xs:maxInclusive value="9999"/>
       </xs:restriction>
      </xs:simpleType>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:complexType>
 <xs:key name="SemesterPK" PrimaryKey="true">
  <xs:selector xpath=".//Semester"/>
   <xs:field xpath="SemesterID"/>
 </xs:key>
</xs:element>
```

LIST 6: Transform "`Register`" table into XSD document tree.

```
<xs:element name="UniversityDB">
 <xs:complexType>
  <xs:element name="Register">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="StdNo" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:length value="5"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="CourseID" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
```

```
        <xs:length value="8"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="SemesterID" use="required"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:length value="5"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Grade"/>
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:maxLength value="2"/>
       </xs:restriction>
      </xs:simpleType>
     <xs:element name="Mark"/>
      <xs:simpleType>
       <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0.00"/>
        <xs:maxInclusive value="100.00"/>
       </xs:restriction>
      </xs:simpleType>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:complexType>
 <xs:key name="RegisterPK" PrimaryKey="true">
  <xs:selector xpath=".//Register"/>
  <xs:field xpath="StdNo"/>
  <xs:field xpath="CourseID"/>
  <xs:field xpath="SemesterID"/>
 </xs:key>
 <xs:keyref name="Regester_Student " refer="StudentPK">
  <xs:selector xpath=".//Register" />
  <xs:field xpath="StdNo" />
 </xs:keyref>
 <xs:keyref name=" Regester_Course" refer="CoursePK">
  <xs:selector xpath=".//Register" />
  <xs:field xpath="CourseID" />
 </xs:keyref>
 <xs:keyref name=" Regester_Semester" refer="SemesterPK">
  <xs:selector xpath=".//Register" />
  <xs:field xpath="SemesterID" />
 </xs:keyref>
</xs:element>
```

4. PERFORMANCE EVALUATION AND DISCUSSION

To access the CPU time performance of the algorithm, we performed many experiments using sets of databases samples; we used a personal computer with a Core i7-3520M CPU @ 2.90GHz, 8GB memory and running windows 7 operating system. We compared from 100 to 600 database sets, database sets from 100 to 300 are smaller than database sets from 300 to 600. Table 4 and Figure 4 illustrate comparing CPU time for generating XML document and XSD schema.

	XML Generate Time (m/s)	**XSD Generate Time (m/s)**	**Total (m/s) XML and XSD**
100 DB Sets	1,291.01	1,621.49	2,912.50
300 DB Sets	5,048.15	6,572.08	11,620.24
600 DB Sets	11,844.23	14,317.95	26,162.18

TABLE 4: CPU time of generating XML document and XSD schema.

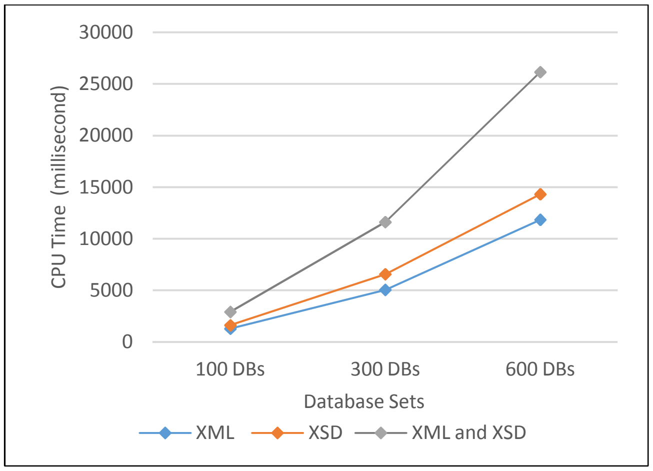

FIGURE 4: CPU time of generating XML document and XSD schema.

Many researches proposed solutions for data integration; they performed several experiments to bring together the flexibility of the XML model for data exchange and the performance of the relational model for data storage. Some solutions have focused on querying and storing XML database schema into specifically designed relational databases, while others have addressed the problem of publishing relational databases as XML documents.

Our proposed technique gives easy and generic solution for online transforming relational database schema into XML. In addition, XML represents a common language as mediate language supported by rich existing tools and standards built around the XML family.

5. CONCLUSION AND FUTURE WORK

In this research, we presented a new conversion technique has implemented based on extracting actual content and catalog of the relational database. The technique helps companies to ease and reliable converting relational databases into XML documents and schemas. The technique is feasible because it uses simple SQL queries to extract the actual content and schema of the relational database. The new XML document and schema contain all aspects of the relational database schema such as content, datatype, constrains and define referential integrity between the tables. The algorithm of the technique contains most of XSD schema toolset in order to develop reliable and costless XSD schema. The experiments demonstrated the efficiency and usefulness of the results of the technique. We calculated CPU time required for conversion process, set of database used for this purpose.

The future research of this paper is generate a fully SQL queries that will be used to update the XML and XSD schema documents and retrieve the relevant relational data from the sources and publish them according to the required XML format, the process should be automatic and dynamically support any incoming data structure. As well, work on a generic technique for data conversion from XML database into relational database.

6. ACKNOWLEDGMENT

The authors would like to acknowledge the financial support provided by the Deanship of Scientific Research at Qassim University, under research project reference number (2376/1435), entitled, "Web-Based Tool for Conversion between Relational Databases and XML".

7. REFERENCES

[1] Silberschatz, A., Korth, H. F., Sudarshan, S. "Database System Concepts", McGraw-Hill, Sixth Edition, 2010, pp. 981-1025.

[2] Arenas, M, Barcelo, P., Libkin, L., Synthesis FM. "Relational and XML Data Exchange", Vol. 2, No. 1, 2010, pp.1-112.

[3] Arenas, M. and Libkin, L. (2005) "Xml data exchange: Consistency and query answering", In Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'05), Baltimore, USA, 2005, pp.13-24.

[4] Jumaa, H., Rubel, P., Fayn, J. "An XML-based framework for automating data exchange in healthcare", e-Health Networking Applications and Services (Healthcom), 12th IEEE International Conference, 1-3 July 2010, pp.26124-269.

[5] Al Hamad, H.A. (2015) "XML-Based Data Exchange in the Heterogeneous Databases (XDEHD)", International Journal of Web & Semantic Technology (IJWesT), Vol. 6, No. 3, 2015, pp.11-24.

[6] Shanmugasundaram J., Tufte K., He G., Zhang C., DeWitt D., J. Naughton. "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp.302–314.

[7] Feng, Y and Jingsheng, X. "Mapping XML DTD to Relational Schema", Database Technology and Applications, First International Workshop, 25-26 April 2009, pp.557-560.

[8] Chunyan, W.; Lo, A.; Alhajj, R.; Barker, K. "Novel Approach for Reengineering Relational Databases into XML", Data Engineering Workshops, 21st International Conference, 05-08 April 2005, pp. 1284.

[9] Teng, L. and Ping, Y. "Schema Conversion from Relation to XML with Semantic Constraints", Fuzzy Systems and Knowledge Discovery, FSKD 2007. Fourth International Conference, 24-27 Aug. 2007, pp.619-623.

[10] Zhang, Y. and Liu, W. "Semantic integration of XML Schema", Machine Learning and Cybernetics, Proceedings. 2002 International Conference, pp. 1058- 1061.

[11] Tzvetkov, V. and Xiong W. (2005) "DBXML - Connecting XML with Relational Databases", Computer and Information Technology, CIT 2005. The Fifth International Conference, 21-23 Sept. 2005, pp.130-135.

[12] Hong, S. and Song, Y. "Efficient XML query using Relational Data Model", Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2007. Eighth ACIS International Conference, July 30 2007-Aug. 1 2007, pp.1095-1100.

[13] Feng, Y. "Converting XML DTD to Database", Intelligent Systems and Applications, ISA 2009. International Workshop, 23-24 May 2009, pp.1-4.

[14] Fong, J., Wonga, H.K., Chengb, Z. "Converting relational database into XML documents with DOM", Information and Software Technology, Vol. 45, 2003, pp.335– 355.

[15] Fong, J. and Cheung, S. 'Translating relational schema into XML schema definition with data semantic preservation and XSD graph', Information and Software Technology, Vol. 47, No. 7, 2001, pp.437-462.

[16] Zhou, A., Lu, H., Zheng, S., Liang, Y., Zhang, L., W. Ji, Tian, Z., "VXMLR: a visual XML-relational database system", Proceedings of the 27th International Conference on Very Large Data Bases, September 11-14, 2001, pp.719-720.

INSTRUCTIONS TO CONTRIBUTORS

Data Engineering refers to the use of data engineering techniques and methodologies in the design, development and assessment of computer systems for different computing platforms and application environments. With the proliferation of the different forms of data and its rich semantics, the need for sophisticated techniques has resulted an in-depth content processing, engineering analysis, indexing, learning, mining, searching, management, and retrieval of data.

International Journal of Data Engineering (IJDE) is a peer reviewed scientific journal for sharing and exchanging research and results to problems encountered in today's data engineering societies. IJDE especially encourage submissions that make efforts (1) to expose practitioners to the most recent research results, tools, and practices in data engineering topics; (2) to raise awareness in the research community of the data engineering problems that arise in practice; (3) to promote the exchange of data & information engineering technologies and experiences among researchers and practitioners; and (4) to identify new issues and directions for future research and development in the data & information engineering fields. IJDE is a peer review journal that targets researchers and practitioners working on data engineering and data management.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJDE.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 6, 2015, IJDE appears with more focused issues. Besides normal publications, IJDE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJDE LIST OF TOPICS
The realm of International Journal of Data Engineering (IJDE) extends, but not limited, to the following:

- Approximation and Uncertainty in Databases and Pro
- Autonomic Databases
- Data Engineering
- Data Engineering Algorithms
- Data Engineering for Ubiquitous Mobile Distributed
- Data Engineering Models
- Data Integration
- Data Mining and Knowledge Discovery
- Data Ontologies
- Data Privacy and Security
- Data Query Optimization in Databases
- Data Streams and Sensor Networks
- Data Warehousing
- Database Tuning
- Database User Interfaces and Information Visualiza
- Knowledge Technologies
- Metadata Management and Semantic Interoperability
- OLAP and Data Grids
- Personalized Databases
- Query Processing in Databases
- Scientific Biomedical and Other Advanced
- Semantic Web

Database

- Social Information Management
- Spatial Temporal

CALL FOR PAPERS

Volume: 6 - **Issue:** 3

i. Paper Submission: October 31, 2015 **ii. Author Notification:** November 30, 2015

iii. Issue Publication: December 2015

CONTACT INFORMATION