Integration of SQL with XML Syntax in Relational Database Systems

Behrooz Seyed-Abbassi,

Sheldon Snyder and John Stoner

- Impact on formatting of Internet data

- Rapid evolution of XML

1997- World Wide Web Consortium (W3C) adopts a **tag-based**, less complex derivative of SGML called **Extensible Markup Language (XML).**

1998- XML introduced as an applicable language for utilization with database systems using **semi-structured data and web-based systems**.

Changes in Web Technology

Impact on formatting of Internet data

resulting in

Rapid evolution of XML

&

Its incorporation in database systems

XML Paradigm

Tag-based, meta-language similar to HTML

- Represents and manipulates data elements in an organized structure with start tag and end tag -- <start> . . . </end>

Example:

<FacultyName> Nancy Fields </FacultyName>

Or

<PartName> Modem </PartName>

Tags: Should be well-formed and valid

Have beginning and end tags

Use the same case

Be properly nested

- XML or xml prefixes reserved for XML tags only
- Not a replacement for HTML

XML Document Example (FacultyInfo.xml)

```
<FACULTY>
        <FIRSTNAME>Jim</FIRSTNAME>
        <LASTNAME>Lange</LASTNAME>
        <TITLE>Associate</TITLE>
        <BIRTHDATE>12/25/1953</BIRTHDATE>
        <ADDRESS>500 Main Street</ADDRESS>
        <SALARY>50000</SALARY>
</FACULTY>
<FACULTY>
        <FIRSTNAME>Jack</FIRSTNAME>
        <LASTNAME>Amick</LASTNAME>
        <TITLE>Assistant</TITLE>
        <BIRTHDATE>1/10/1973 </BIRTHDATE>
        <ADDRESS>16 Lawn Ave.</ADDRESS>
        <SALARY>35000</SALARY>
</FACULTY>
<FACULTY>
        <FIRSTNAME>Helen</FIRSTNAME>
        <LASTNAME>Knapp</LASTNAME>
        <TITLE>Full Professor</TITLE>
        <BIRTHDATE>1/10/1965 </BIRTHDATE>
        <ADDRESS >16 May Ave.</ADDRESS>
        <SALARY>85000</SALARY>
</FACULTY>
```

XML Documents

Stored as text file using one of its major formats or templates for distribution by the Internet to different users on different sites.

- Data Type Definition (DTD)
- XML Schema Definition (XSD)

Document Type Definition (DTD)

- File with .dtd extension to describe XML elements
- Provides composition of database’s logical model
- Defines syntax rules for each XML document
- Defines valid tags

FacultyInfo is the root element
+ Occurrence of one or more times
* Occurrence of zero or more times
? Indicates optional elements
#PCDATA keyword represents the actual text data

XML Document and DTD Document Example

```
<?xml version = "1.0"?>
<!DOCTYPE Faculty INFORMATION “Facutly Info.dtd“>
<FACULTY>
<FIRSTNAME>Jim</FIRSTNAME>
<LASTNAME>Lange</LASTNAME>
<TITLE>Associate</TITLE>
<BIRTHDATE>12/25/1953</BIRTHDATE>
<ADDRESS>500 Main Street</ADDRESS>
<SALARY>50000</SALARY>
</FACULTY>
<FACULTY>
<FIRSTNAME>Jack</FIRSTNAME>
<LASTNAME>Amick</LASTNAME>
<TITLE>Assistant</TITLE>
<BIRTHDATE>1/10/1973 </BIRTHDATE>
<ADDRESS>16 Lawn Ave.</ADDRESS>
<SALARY>35000</SALARY>
</FACULTY>
<FACULTY>
<FIRSTNAME>Helen</FIRSTNAME>
<LASTNAME>Knapp</LASTNAME>
<TITLE>Full Professor</TITLE>
<BIRTHDATE>1/10/1965 </BIRTHDATE>
<ADDRESS >16 May Ave.</ADDRESS>
<SALARY>85000</SALARY>
</FACULTY>
```

```
<?xml version="1.0"?>
<!DOCTYPE Faculty [
<!ELEMENT FACULTY (FIRSTNAME, LASTNAME,
TITLE, BIRTHDATE, ADDRESS, SALARY)>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT LASTNAME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT BIRTHDATE (#PCDATA)>
<!ELEMENT ADDRESS (#PCDATA)>
<!ELEMENT SALARY (#PCDATA)>
]>
```

XML Schema

- **Advanced data definition language to describe structure of XML documents**
- **Checks for database types**
- **Validates data for out-of-range values**
- **XML Schema Definition (XSD) file uses syntax-like XML document**

Example of a simple type

<xsd: element name=“FacultyID" type = "xsd:integer"/>

XML Schema Document Example

```
<?xml version="1.0"?>
<schema xmlns:xsd="FacultyXSDnew" elementFormDefault="qualified"
attributeFormDefault="qualified">
<xsd:complexType name="Faculty">
<xsd: element name="FIRSTNAME" type="xsd:string"  maxOccurs="1" minOccurs="1" />
<xsd:element name="LASTNAME" type="xsd:string" maxOccurs="1" minOccurs="1"  />
<xsd:element name="TITLE" type="xsd:string" maxOccurs="1" minOccurs="1" />
<xsd:element name="BIRTHDATE" type="xsd:string" maxOccurs="1" minOccurs="1" />
<xsd:element name="ADDRESS" type="xsd:string" maxOccurs="1" minOccurs="1" />
<xsd:element name="SALARY" type="xsd:decimal" maxOccurs="1" minOccurs="1" />
</xsd:complexType >
</schema>
```

Project Work

Development of simple query language

similar to traditional SQL with added symbols

to query a relational database &

to convert results to XML document

XML with related attribute names

from one or multiple tables

Definition 1

Syntax of XML structure for tags and elements conforms to the basic standard format of XML with minor enhancement to capture some of the semantics of relational database schema.

Syntax
start tag name, table name, attribute name, attribute value,
end tag name

```
<start TagName   table="TableName"  name="AttributeName“  >
                 AttributeValueRiCj
</end TagName>
```

Example of Definition 1

Customer (CustNo, CustName, Street, City, PhoneNo, ...)

XML structure for Customer schemata with i and j = 1, 2, 3, ..., n

```
<CUSTNO   table="Customer" name="CustNo">
          AttributeValueRiCj
</CUSTNO>
<CUSTNAME   table="Customer" name="CustName">
          AttributeValueRiCj
</CUSTNAME>
<STREET   table="Customer" name="Street">
          AttributeValueRiCj
 </STREET>
< CITY  . . .>
```

Example

Customer (CustNo, CustName, Street, City, PhoneNo, ...)

XML structure for Customer schemata with i and j = 1, 2, 3, ..., n

```
<CUSTNO   table="Customer" name="CustNo">
            AttributeValueR1C1
</CUSTNO>
<CUSTNAME   table="Customer" name="CustName">
            AttributeValueR1C2
</CUSTNAME>
<STREET   table="Customer" name="Street">
            AttributeValueR1C3
 </STREET>
< CITY  . . .>
```

Definition 2

The default tag name for the start and end tags can be modified during conversion and set to any desirable name by the user.

Use of [**AS NewTagName**] resulting in a new name for the tags

<**start NewTagName**   table="TableName" name="AttributeName“   >
AttributeValueRiCj
</**end NewTagName**>

Example of Definition 2

Customer (CustNo, CustName, Street, City, PhoneNo, ...)

```
<CUSTNO   table="Customer" name="CustNo">
            AttributeValueR1C1
</CUSTNO>
```

CUSTNO tag name changing to new name by using [**AS Customer_Number**]

<**CUSTOMER_NUMBER** table="Customer" name="CustNo">
AttributeValueRiCj
</**CUSTOMER_NUMBER**>

Definition 3

Attribute names from one or more tables may be grouped to present a hierarchy level by using one beginning **< tag symbol and one end >** tag symbol before and after group respectively.

Syntax

<Group_tag_name, Table*T1*.AttributeName*i*, Table*T1*.AttributeName*j*...., **>**

clusters tags and elements under level tag **Group_tag_name**

Definition 3

Attribute names qualified with table name

Perhaps from different tables rather than only *T1*

Embedding of levels in levels using a similar formulation

<Group_tag_name1, Table*T1*.AttributeName*i*,..., **<Group_tag_name2**, Table*T2*.AttributeName*j*,...,**>**,**>**,

Perhaps multiple table names instead of only *T1* & *T2*

Example of Definition 3

Customer Address with its attribute names under one level using grouping syntax during conversion

<CustomerAddress, Customer.Street, Customer.City, Customer.PhoneNo,**>**

Result of XML Customer Address conversion using tuple with

Street= 123 Main St, City = San Diego, and PhoneNo= 555-1234

```
<CustomerAddress>
        <STREET table="Customer" name="Street">
                123 Main St
        </STREET>
        <CITY table="Customer" name="City">
                San Diego
        </CITY>
        <PHONENO table="Customer" name="PhoneNo">
                555-1234
        </PHONENO>
</CustomerAddress>
```

Conversion with Definitions 2 and 3

Formulation of Customer Address for more descriptive tag names instead of default tag names for each element

<CustomerAddress ,Customer.Street **AS StreetName** ,Customer.City, Customer.PhoneNo **AS PhoneNumber**,**>**

```
<CustomerAddress>
        <STREETNAME table="Customer" name="Street">
                123 Main St
        </STREETNAME>
        <CITY table="Customer" name="City">
                San Diego
        </CITY>
        <PHONENUMBER table="Customer" name="PhoneNo">
                555-1234
        </PHONENUMBER>
</CustomerAddress>
```

Definition 4

A compression technique developed to allow multiple attributes or records with the same level name and group to be compressed to one level heading with different repeating attribute names and tag elements in the XML document.

Symbols in compression

+, ?, or * after attribute names or

group of attribute names with <[+ | ? | *] ...> syntax

Example of Compression Technique

All addresses for same customers compressed and listed under one heading, instead of listed with different <CustomerAddress, ...> levels or tags by using

CustName, < **[+]** CustomerAddress, Customer.Street AS StreetName, Customer.City, Customer.PhoneNo AS PhoneNumber,>.

Technique especially beneficial when XML document results from **join** of two or more tables with multiple, equal sub-records

Methodology for Query Language Conversion

Combination of power and simplicity of SQL with an extension for XML conversion

Extended SQL language with incorporation of XML conversion

Full benefit of SQL operations on attributes, records, & tables

Generation of XML structures with designated hierarchic levels

Methodology for Query Language Conversion

Utilization of expanded DTD and compression on query language

Extended SQL syntax & prototype presented as traditional SQL

Simple SELECT Clause

To retrieve relational records for conversion to XML format

SELECT AttributeName1, [AttributeName2, ...] FROM Table1

```
<thisQuery>
<Arecord>
        <AttributeName1>AttributeValue11 </AttributeName1>
        <AttributeName2>AttributeValue21</AttributeName2>
   ...
</Arecord>
<Arecord>
        <AttributeName1>AttributeValue12 </AttributeName1>
        <AttributeName2>AttributeValue22</AttributeName2>
   ...
</Arecord>
   ...
</thisQuery>
```

i. Extension of SQL to Support Definition 1

XML structure generation with tags containing attribute name and table name in start tag and attribute name in end tag

```
SELECT Table[1, ...].AttributeName1, Table[1, ...].[AttributeName2, ... ],
        FROM Table1 [, Table 2 , ...] ...
            WHERE Theta ... [any other SQL options]
```

i. Extension of SQL to Support Definition 1

SELECT Table1.AttributeName1, Table1.AttributeName2, ... FROM Table1 [Table 2] ... WHERE …

```
<ThisQuery>
<A_Record>
        <ATTRIBUTENAME1 table="Table1" name="AttributeName1">
                AttributeValue11
        </ATTRIBUTENAME1>
        <ATTRIBUTENAME2 table="Table1" name="AttributeName2">
                AttributeValue12
        </ATTRIBUTENAME2>
</A_Record>
…
</ThisQuery>
```

ii. Extension of SQL to Support Definition 2

Default start and end tag attribute names change by user to a new name using keyword AS

Example with AS Name1 and Name2 tag values

SELECT Table1.AttributeName1 AS NAME1, Table1.AttributeName2 AS NAME2, ... FROM Table1 [Table2] ... WHERE …

```
<ThisQuery>
<A_Record>
        <NAME1 table="Table1" name="AttributeName1">
                AttributeValue11
        </NAME1>
        <NAME2 table="Table1" name="AttributeName2">
                AttributeValue12
        </NAME2>
</A_Record>
…
</ThisQuery>
```

iii. Extension of SQL to Support Definition 3

Relational attribute names from different tables grouped in different levels for generation of XML structure

Levels in SQL commands enclosed by < and >

Syntax for new tag name creation for particular level to allow desired attribute participation
<TagName, AttributeName1[, AttributeName2 , ...]**>**

Recursive utilization by embedding each level as needed using XML structure to create necessary tag levels

iii. Extension of SQL to Support Definition 3

Syntax for SQL with one TagName and generated result

SELECT Table1.AttributeName1 AS NAME1, <TagName, Table1.AttributeName2 AS NAME2, AttributeName3 ..., > FROM Table1 [,Table 2, ...] WHERE ...

```
<ThisQuery>
<A_Record>
        <NAME1 table="Table1" name="AttributeName1">
                AttributeValue11
        </NAME1>
        <TagName>
                <NAME2  table="Table1" name="AttributeName2">
                        AttributeValue12
                </NAME2>
                <ATTRIBUTENAME3 table="Table1"name="AttributeName3">
                        AttributeValue13
                </ATTRIBUTENAME3>
        </TagName>
</A_Record>
...
</ThisQuery>
```

iv. Extension of SQL to Support Definition 4

Repetition of attributes during selection process

Join operations

Compression syntax in SQL query to repeat first record attribute name(s) one time as an upper level hierarchy with all matching, multiple record attribute names associated with each occurrence

Symbols of +, ?, and * to annotate compression in XML structure to help with grouping

iv. Extension of SQL to Support Definition 4

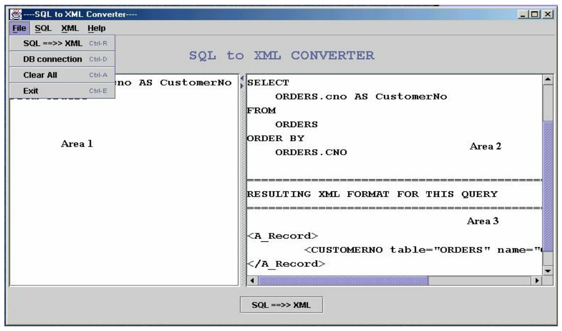
Example of repeating attribute
where AttributeValue11 = AttributeValue21

SELECT DISTINCT Table1.AttributeName1 AS NAME1, + Table1.AttributeName2 AS NAME2 FROM Table1 [,Table 2, ...] WHERE ...

```
<ThisQuery>
<A_Record>
    <NAME1 table="Table1" name="AttributeName1">
                AttributeValue11
    </NAME1>
   <NAME2  table="Table1" name="AttributeName2">
                AttributeValue12
   </NAME2>
   <NAME2  table="Table1" name="AttributeName2">
                AttributeValue22
   </NAME2>
</A_Record>
...
</ThisQuery>
```

Windows Conversion Software Package

Programmed in Windows using Java and JDBC



```
no AS CustomerNo
```

```
SELECT
    ORDERS.cno AS CustomerNo
FROM
    ORDERS
ORDER BY
    ORDERS.CNO

=========================================
RESULTING XML FORMAT FOR THIS QUERY
=========================================

<A_Record>
        <CUSTOMERNO table="ORDERS" name="
</A_Record>
```

Figure 1

Figures 2 and 3: CUSTOMER, SALESPERSON, PRODUCT and ORDERS

CUSTOMER (C)

CNO	CNAME	STREET	CITY	PHONE
C1	Lange	123 Main St	San Diego	555-1234
C2	Johnson	456 Broadway	New York	555-2345
C3	Collins	789 Penn Blvd	Philadelphia	555-4567
C4	Williams	5434 Windy Lane	Chicago	555-9876
C5	Lafavor	987 West Main St	Norman	555-0128

SALESPERSON (S)

SNO	SNAME	QUOTA	CITY
S1	Adams	3000	Dallas
S2	Smith	10000	Chicago
S3	Jones	7500	Phoenix
S4	Knapp	13000	San Diego
S5	Martin	25000	New York

PRODUCT (P)

PNO	PNAME	PRICE	AVLQTY
P1	Modem	35	85
P2	Monitor	265	45
P3	Printer	325	15
P4	CPU Board	250	20
P5	Disk Drive	200	25
P6	Tape Drive	80	20

Figure 2

ORDERS

SNO	PNO	CNO	TOTQTY
S1	P1	C5	4
S1	P3	C5	2
S1	P3	C1	1
S2	P1	C1	2
S2	P4	C4	5
S2	P2	C4	5
S2	P3	C4	5
S3	P5	C4	5
S3	P6	C4	2
S5	P2	C2	10
S5	P4	C2	10
S5	P5	C2	10
S5	P6	C2	2
S5	P5	C3	3
S5	P3	C3	3

Figure 3

Query 1: Simple Query Retrieving Two Attribute Names from Salesperson Table

SELECT s.sno, s.city FROMs

```
<?xml version="1.0"?>
<ThisQuery>
<A_Record><SNO table="s" name="sno">S1</SNO>
<CITY table="s" name="city">Dallas </CITY>
</A_Record>
<A_Record><SNO table="s" name="sno">S2</SNO>
<CITY table="s" name="city">Chicago </CITY>
</A_Record>
<A_Record><SNO table="s" name="sno">S3</SNO>
<CITY table="s" name="city">Phoenix </CITY>
</A_Record>
<A_Record><SNO table="s" name="sno">S4</SNO>
<CITY table="s" name="city">San Diego </CITY>
</A_Record>
<A_Record><SNO table="s" name="sno">S5</SNO>
<CITY table="s" name="city">New York </CITY>
</A_Record>
</ThisQuery>
```

Query 2: Retrieval of All Distinct Salesperson Names with Customer Numbers, Ordered Quantities in Regular Listing

SELECT DISTINCT s.sname as Salesperson_Name, <Customer, orders.cno as Customer_No, orders.totqty,> FROM s, orders WHERE s.sno = orders.sno

```
<?xml version="1.0"?>
<ThisQuery>
…
<A_Record>
<SALESPERSON_NAME table="s" name="sname"> Adams
</SALESPERSON_NAME>
<Customer>
<CUSTOMER_NO table="orders" name="cno">C1</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">1</TOTQTY>
</Customer>
</A_Record>
<A_Record>
<SALESPERSON_NAME table="s" name="sname"> Adams
</SALESPERSON_NAME>
```

```
<Customer>
<CUSTOMER_NO table="orders" name="cno">C5</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">2</TOTQTY>
</Customer>
</A_Record>
<A_Record>
<SALESPERSON_NAME table="s" name="sname"> Adams
</SALESPERSON_NAME>
<Customer>
<CUSTOMER_NO table="orders" name="cno">C5</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">4</TOTQTY>
</Customer>
…
</A_Record>
</ThisQuery>
```

Query 3: Retrieval of All Distinct Salesperson Names with Customer Numbers, Ordered Quantities in Compression Listing

SELECT DISTINCT s.sname as Salesperson_Name, <+ Customer, orders.cno as Customer_No, orders.totqty,> FROM s, orders WHERE s.sno = orders.sno

Query 3 same as Query 2 with compression + symbol

Iteration of salesperson name only one time for all different customers with multiple order quantities for same salesperson

SELECT DISTINCT s.sname as Salesperson_Name, <+ Customer, orders.cno as Customer_No, orders.totqty,> FROM s, orders WHERE s.sno = orders.sno

```
<?xml version="1.0"?>
<ThisQuery>
…
<A_Record>
<SALESPERSON_NAME table="s" name="sname">Adams
</SALESPERSON_NAME>
<Customer>
<CUSTOMER_NO table="orders" name="cno">C1</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">1</TOTQTY>
</Customer>
<Customer>
<CUSTOMER_NO table="orders" name="cno">C5</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">2</TOTQTY>
</Customer>
<Customer>
<CUSTOMER_NO table="orders" name="cno">C5</CUSTOMER_NO>
<TOTQTY table="orders" name="totqty">4</TOTQTY>
</Customer>
…
</A_Record>
</ThisQuery>
```

Query 4: Complex Query with Multiple Conditional Join and Different Hierarchy Levels Using Tag Names

SELECT DISTINCT s.sname as salesperson_name, <Customer, c.cname as customer_name, <CustAddress ,c.street ,c.city ,c.phone,>,>, FROM s, c, orders, p WHERE s.sno = orders.sno and c.cno = orders.cno and p.pno= orders.pno

```
<?xml version="1.0"?>
<ThisQuery>
<A_Record>
< SALESPERSON_NAME table="s" name="SNAME"> Smith   </SALESPERSON_NAME >
<Customer>
     <CUSTOMER_NAME table="c" name="CNAME"> Lange
     </CUSTOMER_NAME>
     <CustAddress>
         <STREET table="c" name="STREET"> 123 Main St
         </STREET>
         <CITY table="c" name="CITY"> San Diego </CITY>
         <PHONE table="c" name="PHONE"> 555-1234 </PHONE>
     </CustAddress>
  </Customer>
…
</A_Record>
</ThisQuery>
```

Example of Additional Commands
Query 5: Aggregate Functions on Orders Table

SELECT AVG(orders.totqty) as avgs, MIN(orders.totqty) as mins, MAX(orders.totqty) as maxs, SUM(orders.totqty) as sums, COUNT(orders.totqty) as counts FROM orders

```
<?xml version="1.0"?>
<ThisQuery>
<A_Record>
<AVGS table="AVG(ORDERS" name="TOTQTY)">4.6</AVGS>
<MINS table="MIN(ORDERS" name="TOTQTY)">1</MINS>
<MAXS table="MAX(ORDERS" name="TOTQTY)">10</MAXS>
<SUMS table="SUM(ORDERS" name="TOTQTY)">69</SUMS>
<COUNTS table="COUNT(ORDERS" name="TOTQTY)">15</COUNTS>
</A_Record>
</ThisQuery>
```

Conclusions

Simple SQL for DB-to-XML

1. Traditional SQL with embedded symbols from XML syntax for retrieving and converting to XML document are mentioned.

2. Considers techniques

- Simple task for parsing and conversion of SQL to XML
- Uses the same format with minimal addition to SQL
- Allows capturing XML syntax by providing definitions
    - Definition 1: Syntax of XML structure for tags and elements
    - Definition 2: The default tag name for the start and end tags
    - Definition 3: Attribute names under one level using grouping syntax
    - Definition 4: A compression technique
- Any additional commands and conversion

3. User can utilize an SQL command to transform the data from a table or tables to XML document with DTD or XSD format.