COS 326 Database Systems

Lecture 8

Semi-structured data and XML (2)

Chapter 31

17 August 2016

Admin matters

Essay topics & bookings

Week	Date	Day	Topic
5	16 Aug	Tues	L7: Semi-structured data & XML databases
	17 Aug	Wed	L8: Semi-structured data & ORDBs (change to study guide schedule)
	19 Aug	Fri	Practical 4: XML DB (BaseX)
6	23 Aug	Tues	Class Test 1: OODB and ORDB L9: Big data and NoSQL databases
	24 Aug	Wed	L10: NoSQL databases (MongoDB) Presentation: Essay topic 1
	26 Aug	Fri	Practical 5: XML data & ORDB (PostgreSQL)

In this lecture

Semi-structured data continued

XML, databases and SQL

XML and Databases

Storing XML in a Relational or Object-Relational database

- Four general approaches:
 - 1. store XML as *value of some attribute* within a tuple
 - 2. store XML in a *shredded* form across a number of attributes and relations
 - 3. store XML in a schema independent form
 - 4. store the XML in a *parsed form* (internal)

Approach 1: Storing XML as an Attribute

docNo	docDate	XMLdoc (whole doc or fragment)
D001	2016-05-20	<stafflist></stafflist>

- Data types
 - CLOB (character large object) used in the past
 - Native XML data type: xml (PostgreSQL) or XMLType (Oracle)
- Raw XML stored as an attribute in a table row
 - efficient to insert documents into database
 - efficient to retrieve documents
 - easy to apply full-text indexing to documents
 - updates: entire XML document is replaced (more recently an XML doc can be updated using SQL)
 - general query performance is poor due to parsing on the fly

Approach 2: Storing XML in Shredded Form

XML decomposed (shredded) into constituent elements

- data distributed over number of attributes
- in one or more relations
- easier to index values of individual elements
- need additional data relating to hierarchical nature of the XML
 - to recompose original document
 - XML updates
- Have to create appropriate database structure from schema: relational or object-relational

(reading for the student: pg 1123)

BRANCHNO	STAFFNO	NAME	FNAME	LNAME
BOO5	SL21		Adam	Eden
вооз	SG37		Eve	Eden

Approach 3: Schema-Independent Representation

e.g. create a	nodelD	nodeType
relation from the	0	Document
Document Object	1	Element
Model	2	Element
(DOM) for the xml	3	Element
document	4	Text
document	5	Element
	6	Element
	7	Text
	8	Element
<stafflist></stafflist>		
<staff branchno="B005"></staff>		
<staffno>SL21</staffno>		
<name></name>		
<fname>John</fname>	<lname>Whi</lname>	ite
<position>Manager</position>		
<dob>1-Oct-45</dob>		
<salary>30000</salary>		

)	nodeName	nodeData	parentID	rootID
	STAFFLIST			0
	STAFFLIST		0	0
	STAFF		1	0
	STAFFNO		2	0
		SL21	3	0
	NAME		2	0
	FNAME		5	0
		John	6	0
	LNAME		5	0
		White	8	0

XML and SQL

- In SQL:2003, SQL:2008 and SQL:2011
 - native XML data type: XML
 - set of XML/SQL operators for the type
 - set of XML/SQL functions
 - set of mappings from relational data to XML

Creating Table using XML Type

```
CREATE TABLE XMLStaff (
     docNo CHAR(4) PRIMARY KEY,
      docDate DATE, staffData XML | );
  INSERT INTO XMLStaff VALUES ('D001',
      DATE '2004-12-01',
      XML('<STAFF branchNo = "B005">
             <STAFFNO> SL21 </STAFFNO>
             <POSITION> Manager </POSITION>
             SELECT * FROM XMLStaff;
Output pane
 Data Output
        Explain
             Messages
                    History
                        staffdata
    docno
             docdate
    character(4)
             date
                        xml
                       <STAFF branchNo = "B005">
    D001
             2004-12-01
                        <STAFFNO>SL21</STAFFNO>
                        <POSITION>Manager</POSITION>
 1
                        <DOB>1945-10-01</DOB>
                        <SALARY>30000</salary> </staff>
```

SQL/XML Functions (reading for the student)

XMLFOREST

· generates XML value with a list of elements as children of a root item.

XMLCONCAT

concatenates a list of XML values

XMLPARSE

performs a non-validating parse of a character string to produce an XML value

XMLROOT

 creates an XML value by modifying the properties of the root item of another XML value

XMLCOMMENT

generates an XML comment

· XMLPI

generates an XML processing instruction

XMLSERIALIZE

generates a character or binary string from an XML value

XMLAGG

 aggregate function, to generate a forest of elements from a collection of elements

SQL/XML Operators (1)

CREATE TABLE Staff (staffNo CHAR(4), fName CHAR(10),

IName CHAR(10), branchNo CHAR(4));

```
INSERT INTO staff VALUES ('SL21', 'John', 'Green', 'B005'); INSERT INTO staff VALUES ('SG5', 'Susan', 'Brown', 'B003');
```

SELECT * FROM Staff;

Output p	Output pane										
Data	a Output Explai		in Messages		History						
				me aracter(10)	Iname character(10)		branchno character(4)				
1	1 SL21 2 SG5		Jo	hn	Green		в005				
2			Susan		Brown		в003				

SQL/XML Operators (2)

XMLELEMENT

 generates XML value with a single element as a child of its root item.

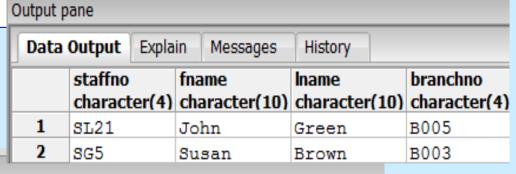
e.g. |**SEL**

Output nane

SELECT

xmlelement (NAME staffname, concat(fName, IName))

FROM Staff;

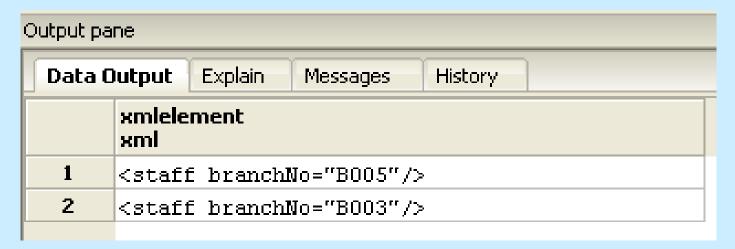


Output	Output pane									
Data	Output	Output Explain Messages History								
	xmlelement xml									
1	<stafi< th=""><th>Ename>J</th><th>ohn</th><th>Green</th><th></th></stafi<>	Ename>J	ohn	Green						
2 <staff< th=""><th>Ename>S</th><th>usan</th><th>Brown</th><th></th></staff<>		Ename>S	usan	Brown						

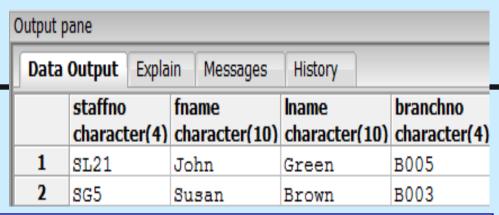
SQL/XML Operators (3)

- Element can have zero or more attributes
 - specified in the XMLATTRIBUTES sub-clause.
- e.g.
 SELECT xmlelement (NAME staff,
 xmlattributes (branchNo AS "branchNo"))

 FROM Staff;



SQL/XML Operators (4)



List all staff as XML elements with name and branch number as an attribute

```
SELECT staffNo, xmlelement
```

- (NAME staff, xmlattributes (branchNo AS "branchNo"), attrib. concat(fName,IName)
-) AS staffXMLCol FROM Staff;

Data	Output	Explai	in Messages	History			
staffno			staffxmlcol				
character(4)		er(4)	xml				
1	1 SL21		<staff bran<="" th=""><th>nchNo="E</th><th>3005">John</th><th>Green</th><th></th></staff>	nchNo="E	3005">John	Green	
2	2 SG5 <staff branc<="" th=""><th>3003">Susa</th><th>n Brown</th><th></th></staff>				3003">Susa	n Brown	

SQL/XML Mapping Functions

SQL/XML

- defines mappings from tables to XML documents
- mappings produces two types of XML documents:
 - 1. mapped table data
 - e.g. PostgreSQL: table_to_xml
 - 2. XML Schema describing the mapped table data
 - e.g. PostgreSQL: table_to_xmlschema
- maps SQL data type to closest match in XML Schema

PostgreSQL and XML data (1)

Functions for converting tables to XML (PostgreSQL 9.2 documentation – pg 247)

Queries:

CREATE TABLE numbers(num1 int, num2 int);

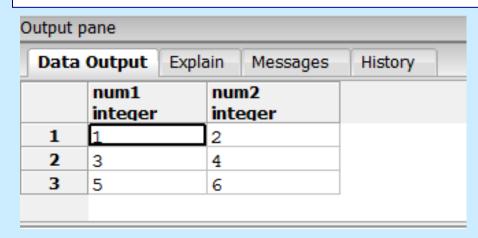
INSERT INTO NUMBERS VALUES(1,2);

INSERT INTO NUMBERS VALUES(3,4);

INSERT INTO NUMBERS VALUES(5,6);

SELECT * FROM NUMBERS;

Results



PostgreSQL and XML data (2)

Queries to generate XML from table

```
SELECT table_to_xml('Numbers', false, false,"); --OR:
SELECT query_to_xml('SELECT num1, num2 FROM Numbers', false,false,");
Output (when table_to_xml is used):
```

```
Data Output | Explain
                   Messages
                             History
     table to xml
     xml
     <numbers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <row>
       <num1>1</num1>
       <num2>2</num2>
     </row>
     <row>
       <num1>3</num1>
 1
       <num2>4</num2>
     </row>
     <row>
       <num1>5</num1>
       <num2>6</num2>
     </row>
     </numbers>
```

PostgreSQL and XML data (3)

Generate XML schema from table

Query:

```
SELECT table_to_xmlschema('Numbers', false,false,"); --OR:
SELECT query_to_xmlschema('SELECT num1, num2 FROM Numbers', false,false,");
```

Output when table_to_xmlschema is used (partial listing):

```
Data Output | Explain | Messages
                           History
    table to xmlschema
    xml
    <xsd:schema</pre>
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:simpleType name="INTEGER">
      <xsd:restriction base="xsd:int">
        <xsd:maxInclusive value="2147483647"/>
        <xsd:minInclusive value="-2147483648"/>
      </xsd:restriction>
1
    </xsd:simpleType>
    <xsd:complexType name="RowType.TestingXML.public.numbers">
      <xsd:sequence>
        <xsd:element name="num1" type="INTEGER" minOccurs="0"></xsd:element>
        <xsd:element name="num2" type="INTEGER" minOccurs="0"></xsd:element>
      </xsd:sequence>
```

RECALL: Creating Table using XML Type

```
CREATE TABLE XMLStaff (
      docNo CHAR(4) PRIMARY KEY,
       docDate DATE, staffData XML | );
  INSERT INTO XMLStaff VALUES ('D001',
       DATE '2004-12-01',
       XML('<STAFF branchNo = "B005">
               <STAFFNO>SL21</STAFFNO>
               <POSITION>Manager</POSITION>
               <DOB>1945-10-01
               <SALARY>30000</SALARY> </STAFF>') );
   SELECT * FROM XMLStaff;
Output pane
 Data Output
          Explain
               Messages
                        History
                            staffdata
     docno
               docdate
     character(4)
               date
                            xml
                            <STAFF branchNo = "B005">
     D001
               2004-12-01
                            <STAFFNO>SL21</STAFFNO>
                            <POSITION>Manager</POSITION>
  1
                            <DOB>1945-10-01</DOB>
                            <SALARY>30000</SALARY> </STAFF>
```

PostgreSQL and XML data (4)

SQL and **XPath** queries:

syntax is: xpath('xpath-expression', colname)

SELECT docNo, docDate,

xpath ('/STAFF/@branchNo',staffData) AS BranchNumber,

xpath ('/STAFF/STAFFNO/text()',staffData) AS StaffNumber,

xpath ('/STAFF/POSITION/text()',staffData) AS Position,

xpath ('/STAFF/DOB/text()',staffData) AS DOB,

xpath ('/STAFF/SALARY/text()',staffData) AS Salary

FROM XMLStaff;

Results:

Data Output Explain Messages History										
	docno		branchnumber		•	dob	salary			
	character(4)	date	xml[]	xml[]	xml[]	xml[]	xml[]			
1	D001	2004-12-01	{B005}	{SL21}	{Manager}	{1945-10-01}	{30000}			

PostgreSQL and XML data (5)

SQL, XPath queries & unnest function:

```
SELECT docNo, docDate,
  unnest(xpath ('/STAFF/@branchNo',staffData) )AS BranchNumber,
  unnest(xpath ('/STAFF/STAFFNO/text()',staffData)) AS StaffNumber,
  unnest(xpath ('/STAFF/POSITION/text()',staffData) )AS Position,
  unnest(xpath ('/STAFF/DOB/text()',staffData)) AS DOB,
  unnest(xpath ('/STAFF/SALARY/text()',staffData)) AS Salary
FROM XMLStaff;
```

Results:

	Data	Output [Explain	Messages	History				
docno characte				branchnumber staffnumb xml xml		•	dob xml	salary xml	
	1	D001	20	004-12-01	в005	SL21	Manager	1945-10-01	30000

Reading for the student

Section 31.7: XML in Oracle