

XML and Databases

Chapter 17: XML-Support in Modern SQL Databases

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Winter 2022/23

<http://www.informatik.uni-halle.de/~brass/xml22/>

Objectives

After completing this chapter, you should be able to:

- write SQL queries with SQL/XML functions that generate XML,
- compare different options of storing XML data in a relational database (including the XML data type).

Contents

Introduction

Introduction

- Since SQL:2003, the SQL standard contains a part about “XML-Related Specifications”.
- Large object-relational DBMS have XML support:
 - Oracle
 - IBM DB2
 - Microsoft SQL Server
- These systems contain support for XQuery.
- PostgreSQL basically supports SQL/XML.

It must be built with `configure --with-libxml`. It only offers XPath instead of XQuery.

Example Database

STUDENTS			
<u>SID</u>	FIRST	LAST	EMAIL
101	Ann	Smith	...
102	David	Jones	NULL
103	Paul	Miller	...
104	Maria	Brown	...

EXERCISES			
<u>CAT</u>	<u>ENO</u>	TOPIC	MAXPT
H	1	ER	10
H	2	SQL	10
M	1	SQL	14

RESULTS			
<u>SID</u>	<u>CAT</u>	<u>ENO</u>	POINTS
101	H	1	10
101	H	2	8
101	M	1	12
102	H	1	9
102	H	2	9
102	M	1	10
103	H	1	5
103	M	1	7

XML-Generating Functions (2)

- Example:

```
SELECT XMLELEMENT(NAME STUDENT,
                  XMLATTRIBUTES(SID, FIRST,
                                LAST, EMAIL))
FROM   STUDENTS
```

Note that the element name is an SQL identifier, not a string.
PostgreSQL maps identifiers to lowercase, Oracle to uppercase.
Use delimited identifiers "..." to protect case.

- This returns a table with one column of STUDENT nodes:

```
<STUDENT SID='101' FIRST='Ann' LAST='Smith'
        EMAIL='...' />
```


XML-Generating Functions (4)

- Of course, one can nest such functions:

```
SELECT XMLELEMENT(NAME "STUDENT",
                  XMLELEMENT(NAME "FIRST", FIRST),
                  XMLELEMENT(NAME "LAST", LAST))
FROM   STUDENTS
```

- This returns STUDENT nodes like the following one:

```
<STUDENT><FIRST>Ann</FIRST>
      <LAST>Smith</LAST></STUDENT>
```

XML-Generating Functions (5)

- **XMLAGG** is an aggregation function that concatenates the XDM trees for the tuple variable assignments in the group:

```
SELECT XMLELEMENT(NAME "STUDENTS",  
                  XMLAGG(XMLELEMENT(NAME "STUD",  
                                     XMLATTRIBUTES(  
                                         S.FIRST,  
                                         S.LAST))))  
FROM STUDENTS S
```

Above, we got one result row with an XML element for each student. This query generates one row with data for all students (without `GROUP BY`, there is only one group). See the next slide for the result.

XML-Generating Functions (6)

- The query on the previous slide with XMLAGG generates a single result row with the following XML document:

```
<STUDENTS>
  <STUD first="Ann" last="Smith"/>
  <STUD first="David" last="Jones"/>
  <STUD first="Paul" last="Miller"/>
  <STUD first="Maria" last="Brown"/>
</STUDENTS>
```

PostgreSQL outputs everything in one line.

XML-Generating Functions (7)

- **XMLFOREST** is an easy way to map columns to elements:

```
SELECT XMLELEMENT(NAME STUD,  
                  XMLFOREST(S.FIRST, S.LAST))  
FROM   STUDENTS S
```

`XMLATTRIBUTES` does the same for attributes. One can rename the elements with “AS” (like the output columns under `SELECT`). It is called “FOREST” because it returns a sequence of XDM trees.

- This returns a result row for each student, e.g.

```
<stud>  
  <first>Ann</first>  
  <last>Smith</last>  
</stud>
```

XML-Generating Functions (8)

- **XMLCONCAT** can be used to concatenate XDM sequences, for instance, one can generate a sequence of two elements in each result row:

```
SELECT XMLCONCAT(XMLELEMENT(NAME FIRST, S.FIRST),  
                XMLELEMENT(NAME LAST, S.LAST))  
FROM    STUDENTS S
```

XMLCONCAT works with any number of arguments. XMLCONCAT is probably seldom needed, since XMLELEMENT has this function for its content.

- This returns a result row for each student, e.g.

```
<first>Ann</first><last>Smith</last>
```

XML-Generating Functions (9)

Functions to generate other node types:

- `XMLCOMMENT('Comment')` generates `<!--Comment-->`.
- `XMLPI(NAME "xml-stylesheet",
'type="text/xsl" href="my.xsl"')`

generates the processing instruction

```
<?xml-stylesheet type="text/xsl"  
href="my.xsl"?>
```

- `XMLROOT` can be used to set the XML version and the standalone property.

Deprecated since SQL 2005, but available in Oracle, PostgreSQL.

More Examples (1)

- One can combine XMLAGG with GROUP BY, e.g. to generate students with nested homework points:

```
SELECT XMLELEMENT(NAME "STUDENT",  
                  XMLATTRIBUTES(S.LAST),  
                  XMLAGG(XMLELEMENT(NAME "RESULT",  
                                    XMLATTRIBUTES(R.ENO,R.POINTS))))  
FROM    STUDENTS S, RESULTS R  
WHERE   S.SID = R.SID  
AND     R.CAT = 'H'  
GROUP  BY S.LAST, S.SID  
ORDER  BY S.LAST;
```

More Examples (2)

- This returns three rows:
 - ```
<STUDENT last="Jones">
 <RESULT eno="2" points="9"/>
 <RESULT eno="1" points="9"/>
</STUDENT>
```
  - ```
<STUDENT last="Smith">
  <RESULT eno="2" points="8"/>
  <RESULT eno="1" points="10"/>
</STUDENT>
```
 - ```
<STUDENT last="Miller">
 <RESULT eno="1" points="5"/>
</STUDENT>
```

“Maria Brown” did not submit homeworks (or use outer join).



## More Examples (3)

- Complex XML structures can be generated by using subqueries as terms.
- These subqueries may return only one row and column (given an assignment for the tuple variables in the outer query).

It is no error if they return 0 rows, then the subquery result is a null value.  
The null value is usually treated as empty sequence in XML.

- Typically, one uses a subquery with `XMLAGG` (and no `GROUP BY`) to compute a single value that can be a sequence of many elements.

# More Examples (4)

```

SELECT
XMLELEMENT(NAME "HOMEWORK_RESULTS",
 (SELECT XMLAGG(XMLLEMENT(NAME "STUDENT",
 XMLATTRIBUTES(S.FIRST, S.LAST),
 (SELECT XMLAGG(XMLLEMENT(NAME "HW",
 XMLATTRIBUTES(R.ENO,
 R.POINTS)))
 FROM RESULTS R
 WHERE R.SID=S.SID AND R.CAT='H'))))
FROM STUDENTS S))

```

PostgreSQL permits SELECT without FROM. In Oracle, one writes FROM DUAL (DUAL is a dummy relation with exactly one row).

Note that “Maria Brown” (without submitted homeworks) appears in the output (see next slide) because S in the outer query runs over all students.

# More Examples (5)

## Query Result:

```
<HOMEWORK_RESULTS>
 <STUDENT first="Ann" last="Smith">
 <HW eno="1" points="10"/>
 <HW eno="2" points="8"/>
 </STUDENT>
 <STUDENT first="David" last="Jones">
 <HW eno="1" points="9"/>
 <HW eno="2" points="9"/>
 </STUDENT>
 <STUDENT first="Paul" last="Miller">
 <HW eno="1" points="5"/>
 </STUDENT>
 <STUDENT first="Maria" last="Brown"/>
</HOMEWORK_RESULTS>
```

# Identifier Mapping

- The SQL standard defines a mapping from SQL identifiers to XML identifiers that uses hexadecimal encoding if necessary.

Note that delimited identifiers "... " in SQL permit arbitrary character sequences.

- E.g. `xmlelement(NAME "STUDENT DATA")` gives  
`<STUDENT_x0020_DATA/>`.

The ASCII-code of the space is 32, i.e. 20 in hexadecimal notation.  
 The translation works with Unicode, therefore two bytes are used.

- Oracle does not map explicitly given identifiers.

# References

- Georg Lausen: Datenbanken — Grundlagen und XML-Technologien. Elsevier/Spektrum, 2005. Folien zu Kapile 6: SQL und XML:  
[\[http://dbis.informatik.uni-freiburg.de/content/DBBuch/Folien/kapitel06.pdf\]](http://dbis.informatik.uni-freiburg.de/content/DBBuch/Folien/kapitel06.pdf)
- Jim Melton, Stephen Buxton: Querying XML — XQuery, XPath and SQL/XML in Context. Elsevier/Morgan Kaufmann, 2006.
- ISO/IEC 9075-14:2006: Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) [\[Non-Final Version\]](#)
- Oracle Database Online Documentation 11g Release 2 (11.2): XML DB Developer's Guide.  
[\[https://docs.oracle.com/cd/E11882\\_01/appdev.112/e23094/toc.htm\]](https://docs.oracle.com/cd/E11882_01/appdev.112/e23094/toc.htm)  
[\[http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/\]](http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/)
- Microsoft SQL Server: XML Data Type and Columns.  
[\[https://docs.microsoft.com/en-us/sql/relational-databases/xml/\]](https://docs.microsoft.com/en-us/sql/relational-databases/xml/)  
[\[xml-data-type-and-columns-sql-server\]](#)
- DB2 Version 10.5 for Linux, UNIX, and Windows: pureXML Guide.  
[\[https://www-01.ibm.com/support/docview.wss?uid=swg27038855\]](https://www-01.ibm.com/support/docview.wss?uid=swg27038855)  
[\[http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en\\_US/DB2pureXML-db2xge1050.pdf\]](http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en_US/DB2pureXML-db2xge1050.pdf)
- PostgreSQL: Documentation: XML Functions  
[\[https://www.postgresql.org/docs/10/static/functions-xml.html\]](https://www.postgresql.org/docs/10/static/functions-xml.html)