SQL 2003

Standardized by ANSI/ISO; next version after SQL 1999

- Includes SQL/XML: SQL extensions for XML (other aspects of SQL 2003 are not relevant here)
- Distinct from Microsoft's SQLXML
- SQL/XML is included in products
  - By DBMS vendors, sometimes with different low-level details (MINUS versus EXCEPT)
  - DBMS-independent products

XML Type in SQL/XML

- A specialized data type for XML content; distinct from text
- Usable wherever an SQL data type is allowed: type of column, variable, tuple cell, and so on . . .
- Value rooted on the XML Root information item (described next)

XML Root Information Item: 1

Based on the XML InfoSet document information item, this can be an

- XML root (as in SQL/XML)
- XML element
- XML attribute
- XML parsed character data (text; aka PCDATA)
- XML namespace declaration
- XML processing instruction
- XML comment

And some more possibilities from the InfoSet . . .

XML Root Information Item: 2

- Unlike the XML InfoSet root (which allows exactly one child element), this allows zero or more children
  - Partial results need not be documents
- IS DOCUMENT: a predicate that checks if the argument XML value has a single root
- An XML value can be
  - NULL, as usual for SQL
  - An XML root item, including whatever it includes

SQL/XML Builtin Operators

- xmlparse(): maps a string (char, varchar, clob) to a value of type XML (stripping whitespace by default)
- xmlserialize(): maps a value of type XML to a string
- xmlconcat(): combines values into a forest
- xmlroot(): create or modify the root node of an XML value

SQL/XML Publishing Functions: 1

These are templates that go into a SELECT query; all with names that begin “xml”

- xmlelement(name ’Song’, ·)
  - Needs a value: an SQL column or expression or an attribute or an element
  - Yields a value (an element)
  - Can be nested, of course
- xmlattributes(column [AS cname], column [AS cname],. . .)
  - Creates XML attributes from the columns
  - Inserts into the surrounding XML element

SQL/XML Publishing Functions: 2

- xmlforest()
  - Creates XML elements from columns
  - Analogous to a node-set in XPath
  - Must be placed within an element; otherwise not well-formed XML
- xmlagg(): combines a collection of rows, each with a single XML value into a single forest
- xmlnamespaces()
- xmlcomment(): comment
- xmlpi(): processing instruction

SQL/XML Example: 1

```
SELECT xmlelement(Name 'Sgr',
2          xmlattributes (z.sgrId AS student_ID),
           z.sgrName)
FROM Singer z
WHERE ...
```

yields something like

```
<Sgr student_ID='s1'>
 Eagles
</Sgr>
```

SQL/XML Example: 2

```
SELECT xmlelement(Name 'Sgr',
2          xmlattributes (z.sgrId AS student_ID),
           z.sgrName,
           xmlelement(Name 'Song', 'Hotel'))
FROM Singer z
WHERE ...
```

yields something like

```
<Sgr student_ID='s1'>
  Eagles
  <Song>Hotel</Song>
4 </Sgr>
```


SQL/XML Mapping Rules

A number of low-level matters, which are conceptually trivial but complicate combining SQL and XML effectively; captured as *mapping rules*

- Lexical encodings in names and content
- Mapping datatypes in each direction, e.g., SQL date and XML Schema date
- Mapping SQL tables, schemas, catalogs to and from XML

Tool Support for SQL 2003

- Oracle 10g, IBM DB2, Sybase support it
- Apparently, Microsoft doesn't or won't [not sure]
- Oracle 9i release 2 supports similar constructs, but in proprietary syntax

Oracle 9i SQL/XML: 1

```
1 CREATE TABLE singer ( sgrId VARCHAR2(9) NOT NULL,
                        sgrName VARCHAR2(15) NOT NULL,
                        sgrInfo SYS.XMLTYPE NULL,
                        CONSTRAINT singer_key
                        PRIMARY KEY (sgrId));
```

Oracle 9i SQL/XML: 2

```
INSERT INTO singer VALUES ('Sgr-01', 'Eagles',
   SYS.XMLTYPE.createXML('<genre>rock</genre>'));

INSERT INTO singer VALUES ('Sgr-04', 'Beatles',
             SYS.XMLTYPE.createXML(
   '<trivia><convictions>freedom</convictions>
   <genre>rock</genre></trivia>'));

SELECT z.sgrName, z.sgrInfo.extract( '/genre/text()')
                                      .getClobVal()
FROM singer z;
```


Oracle 9i SQL/XML: 3

```
SELECT z.sgrName, z.sgrInfo.extract( '//genre/text()')
                                      .getClobVal()
FROM singer z
WHERE z.sgrInfo.extract(
     '//genre/text()').getStringVal() like 'r%';

SELECT z.sgrName, z.sgrInfo.extract( '/genre/text()')
                                      .getClobVal()
FROM singer z
WHERE z.sgrInfo.existsNode('//genre') = 1;
```

Oracle 9i SQL/XML: 4

```
SELECT SYS_XMLAGG(SYS_XMLGEN(z.sgrname),
    SYS.XMLGENFORMATTYPE.createformat('FooList'))
                                        .getClobVal()
FROM singer z
WHERE z.sgrId IS NOT NULL
GROUP BY z.sgrname;
```


Modern Information Systems

- Three legs of modern software systems
  - *Documents:* as in XML
  - *Tuples:* as in the information stored in relational databases
  - *Objects:* as in programming languages
- A lot of effort goes into managing translations among these at the level of programming
- But deeper challenges remain . . .

Limitations of XML

- Doesn’t represent meaning
- Doesn’t represent conceptual structure
- Enables multiple representations for the same information
  - Give an example

Transforms can be robustly specified and accurately documented only if models are known, but usually the models are not known

Directions in XML

Trends: sophisticated approaches for

- Querying and manipulating XML, e.g., XSLT and XQuery
- Sophisticated storage and access techniques in traditional relational databases
- Tools that shield programmers from low-level details
- Semantics, e.g., RDF, OWL, . . .