Lecture Notes

Course Number:    CSC 513
Instructor:       Dr. Singh

Lecture Number:   24B
Quick Look at SQL

Structured Query Language

- Data Definition Language: CREATE TABLE
- Data Manipulation Language: SELECT, INSERT, DELETE, UPDATE
- Basic paradigm for SELECT

```sql
SELECT t1.column-1, t1.column-2 ... tm.column-n
FROM table-1 t1, table-m tm
WHERE t1.column-3=t4.column-4 AND ...
```
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)
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 Built-in 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
SELECT xmlElement(Name 'Sgr',
   xmlAttributes (z.sgrld AS student-ID),
   z.sgrName)
FROM Singer z
WHERE ...

yields something like

<Sgr student-ID='s1'>
  Eagles
</Sgr>
SQL/XML Example: 2

```sql
SELECT xmlElement(Name 'Sgr',
    xmlAttributes (z.sgrId AS student-ID),
    z.sgrName,
    xmlElement(Name 'Song', 'Hotel'))
FROM Singer z
WHERE ...
```

yields something like

```xml
<Sgr student-ID='s1'>
  Eagles
  <Song>Hotel</Song>
</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
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',
SY.SYS.XMLTYPE.createXML(''<genre>rock</genre>''));

INSERT INTO singer VALUES ('Sgr-04', 'Beatles',
SY.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()').getStringVal() like 'r%';
```

```
SELECT z.sgrName, z.sgrInfo.extract('/genre/text()').getAsciiVal() like 'r%';
```

```
SELECT z.sgrName, z.sgrInfo.extract('/genre/text()').getAsciiVal() like 'r%';
```

```
FROM singer z
```

```
WHERE z.sgrInfo.existNode('/genre') = 1;
```

© Munindar P. Singh. CSC 513, Spring 2009
SELECT SYS_XMLAGG(SYS_XMLGEN(z.sgrname),
    SYS.XMLGENFORMATTYPE.createformat('FooList'))
    .getClobVal()
FROM singer z
WHERE z.sgrld 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 . . .