Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Contracts and Governance

XML Concepts and Techniques

XML Modeling and Storage

Summary and Directions
The major aspects of storing XML include

- Concepts: Data and Document Centrism
- Storage
- Mapping to relational schemas
- SQL/XML
Modern Information Systems

- Four legs of modern software systems
  - Documents: as in XML
  - Tuples: as in the information stored in relational databases
  - Objects: as in programming languages
  - Rules: as in policies

- Effort in managing translations among these for programming

- But deeper challenges remain . . .
Data-Centric View: 1

```xml
<relation name='Student'>
  <tupl>
    <attr1>V11</attr1>
    ...
    <attrn>V1n</attrn>
  </tupl>
  ...
</relation>
```

- Extract and store via mapping to DB model
- Regular, homogeneous structure
Data-Centric View: 2

- Ideally, no mixed content: an element contains text or subelements, not both
- Any mixed content would be templatic, i.e.,
  - Generated from a database via suitable transformations
  - Generated via a form that a user or an application fills out
- Order among siblings likely irrelevant (as is order among relational columns)
- Increased precision at the cost of reduced flexibility

Expensive if documents are repeatedly parsed and instantiated
Document-Centric View

- Irregular: doesn’t map well to a relation
  - Supports casual modifications, enhancing loose coupling
  - Matches heterogeneous data
- Depending on entire doc for application-specific meaning
Data- vs Document-Centric Views
Database administrator vs business process analyst

- **Data-centric**: data is the main thing
  - XML simply renders the data for transport
  - Store as data
  - Convert to/from XML as needed
  - The structure is important

- **Document-centric**: documents are the main thing
  - Documents are complex (e.g., design documents) and irregular
  - Store documents wherever
  - Use DBMS where it facilitates performing important searches
Storing Documents in Databases

- Use character large objects (CLOBs) within DB: searchable only as text
- Store paths to external files containing docs
  - Simple, but no support for integrity
- Use some structured elements for easy search as well as unstructured clob or files
- Heterogeneity complicates mappings to typed OO programming languages

Storing documents in their entirety may sometimes be necessary for external reasons, such as regulatory compliance
Database Features

- Storage: schema definition language
- Querying: query language
- Transactions: concurrency
- Recovery
Potential DBMS Types for XML: 1

- **Object-oriented**
  - Nice structure
  - Intellectual basis of many XML concepts, including schema representations and path expressions
  - Not highly popular in standalone products

- **Relational**
  - Limited structuring ability (1NF: each cell is atomic)
  - Extremely popular
  - Well optimized for flat queries
Potential DBMS Types for XML: 2

- Object relational: hybrids of above
  - Not highly popular in standalone products
- Custom XML stores or native XML databases
  - Emerging ideas: may lack core database features (e.g., recovery, ...)
  - Enable fancier content management systems
  - Leading open source products:
    - Apache CouchDB (server; XPath)
    - Berkeley DB XML (libraries; XQuery)
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Contracts and Governance

XML Concepts and Techniques

XML Modeling and Storage
  SQL/XML
  XML and Relational Databases
  XML Schema
  XML Keys
Quick Look at SQL
Structured Query Language

- Data Definition Language: CREATE TABLE
- Data Manipulation Language: SELECT, INSERT, DELETE, UPDATE
- Basic paradigm for SELECT
  ```
  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 and SQL 2008
Standardized by ANSI/ISO

- 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
XML Modeling and Storage

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 `xml_element` (Name 'Sgr',
    `xml_attributes` (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',
    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>
</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
Too 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',
  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

```sql
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;
```