Oracle XML DB

© Leonidas Fegaras
University of Texas at Arlington

- Native support for storage and processing of XML documents
  - Using XMLType
    - Unstructured CLOBs
    - Structured table objects derived from XML Schema
  - Access to XML data from SQL
  - Special XML indexing
  - Supports most W3C standards
    - XML Schema, XPath, XQuery

XMLType

- An Oracle supplied type that can be used as a column’s datatype
- XMLType column supports two storage options:
  - Unstructured Storage using Large Objects (CLOBs) or files
    - maintains accuracy of the original XML
  - Structured storage
    - Based on Document Object Model (DOM)
  - Hybrid storage
- XMLType is viewed as collection of elements and attributes
  - Elements can be
    - Complex types: consisting of elements and attributes
    - Simple types: consisting of scalar values
  - Structured Storage maps a complex type (a child element or an attribute) to SQL complex type
    - Choices: CLOBs, VARRAYS, Nested Object Tables
    - A simple type maps to a primitive SQL data type

Example

CREATE TABLE myXML (
    name VARCHAR(20),
    xml XMLType );

INSERT INTO myXML
VALUES("<student>
    <name>John Smith</name>
    <gpa>3.5</gpa>
</student>");
Structured Storage from XML Schema

- Unstructured XMLType must be parsed to DOM before processing
- Structured XMLType consists of a set of objects
  - Uses less space, allows faster accesses (native indexes, set processing), piecewise updates
  - XPaths/XQueries are translated to native database queries
- Storage structure is derived from XML Schema annotations
  - You may pass the XML Schema to the XMLType constructor
  - or get the XML Schema from the link inside the XML document

Example

- Example:
  
  CREATE TABLE orders (order_info XMLTYPE)
  XMLSCHEMA 'http://www.oracle.com/xdb/orderSchema.xsd'
  ELEMENT 'Purchase Order';

- This may create the relational schema:
  CREATE TYPE purchaseOrderItem as OBJECT(...);
  CREATE TYPE purchaseOrderItems as VARRAY(…) OF purchaseOrderItem;
  CREATE TYPE purchaseOrder as OBJECT(
    customerId number,
    customerName varchar2(2000),
    orderDate date,
    shipDate date,
    Items purchaseOrderItems );

Example (cont.)

- You can insert data:
  
  INSERT INTO orders VALUES(
    XMLType('<?xml version="1.0"?>
      <PO pon=",1">
        <PNAME>PO_1</PNAME>
        ...
    </PO>'));

- The XML data are validated against the XML Schema and then inserted into the relational tables

- Querying:
  
  SELECT order_info FROM orders
  WHERE existsNode(order_info, '/PO/PNAME') > 0;

  SELECT extract(order_info,'/PO/PNAME') FROM orders;

Example (cont.)

- Updates:
  
  UPDATE myxml R
  SET value(R) =
    updateXML(value(R),'/gradstudent/name/text()','Mary James')
  WHERE existsNode(value(R),'/gradstudent[name="Smith"]') =1;

- Convert relational data to XML
  
  SELECT SYS_XMLGEN(*)
  FROM employee