**LAB exercise XML data type**

Oracle9i Release 1 (9.0.1) (and later versions) introduced a new datatype, XMLType, to facilitate native handling of XML data in the database. The following summarizes XMLType:

* XMLType can be used in PL/SQL stored procedures as parameters, return values, and variables.
* XMLType can represent an XML document as an instance (of XMLType) in SQL.
* XMLType has built-in member functions that operate on XML content. For example, you can use XMLType functions to create, extract, and index XML data stored in Oracle9i database.
* Functionality is also available through a set of Application Program Interfaces (APIs) provided in PL/SQL and Java.

With XMLType and these capabilities, SQL developers can leverage the power of the relational database while working in the context of XML. Likewise, XML developers can leverage the power of XML standards while working in the context of a relational database.

XMLType datatype can be used as the datatype of columns in tables and views. Variables of XMLType can be used in PL/SQL stored procedures as parameters, return values, and so on. You can also use XMLType in SQL, PL/SQL, and Java (through JDBC).

When you create an XMLType column without any XML schema specification, a hidden **CLOB column is automatically created** to store the XML data. The XMLType column itself becomes a virtual column over this hidden CLOB column. It is not possible to directly access the CLOB column; however, you can set the storage characteristics for the column using the XMLType storage clause. Check the CLOB object by select \* from user\_objects.

**CREATE TABLE warehouses(**

**warehouse\_id NUMBER(4),**

**warehouse\_spec XMLTYPE,**

**warehouse\_name VARCHAR2(35),**

**location\_id NUMBER(4));**

**INSERT INTO warehouses VALUES**

**( 100, XMLType(**

**'<Warehouse whNo="100">**

**<Building>Owned</Building>**

**<Owner>J Smith</Owner>**

**</Warehouse>'), 'Tower 1 Records', 1003);**

**INSERT INTO warehouses VALUES**

**( 200, XMLType(**

**'<Warehouse whNo="200">**

**<Building>Owned</Building>**

**<Owner>J Martin</Owner>**

**</Warehouse>'), 'Tower 2 Records', 1004);**

**Select and XMLType**

Text description of extract_xml.gif follows

**extract()** on XMLType extracts the node or a set of nodes from the document identified by the **XPath** expression. The extracted nodes can be elements, attributes, or text nodes. When extracted out, all text nodes are collapsed into a single text node value. Namespace can be used to supply namespace information for prefixes in the XPath string.

The XMLType resulting from applying an XPath through **extract**() need not be a well-formed XML document but can contain a set of nodes or simple scalar data in some cases. You can use the **getStringVal**() or getNumberVal() methods on XMLType to extract this scalar data.

**The following SQL statement extracts the owners from warehouse\_spec XML:**

**SELECT w.warehouse\_spec.extract('/Warehouse/Owner/text()').getStringVal() "Owner"**

**FROM warehouses w;**

|  |
| --- |
| **Owner** |
| J Smith |
| J Martin |