

### Why Oracle XML

### XML

Extensible Markup Language      User defined tags

tags associated w/ the storage of data

xml:Nothing but the presentation of data      HTML:present of web page

syntax

```
<?xml version="1.0" encoding="UTF-8"?>
```

tags: <Product> .... </Product>

tags: case sensitive, attribute, val is not require

free tool: html-kit      tag align, indent etc

### Oracle XML DB

stand database feature      native support by XMLType

XMLPath and XQuery      XML<->relational data

File repository

XMLType limitations

performance Hit      used only when required

new API and design

### Relational data -> XML

XMLELEMENT      get one XMLType instance

XMLType.getcolval      char format of xmltype data

XMLAttributes      set attribute

XMLForest      deal with col

XMLAgg      deal wit row

XMLElement(XMLForest)      get parent tag for group of col

XMLPi      xml processing instruction

XMLComment      add comment line

### DBMS\_XMLGen

```
--avoid complex syntax of using XML function to
create xml doc
-- create object view:simplify sql
CREATE OR REPLACE VIEW VIEW_NAME OF OBJECT_TYPE
WITH OBJECT IDENTIFIER (COLUMN(s)) as SELECT
--using dbms_xmlgen to generate file and save
--directory: pre-defined & ALL CAPS
out_file utl_file.file_type;
out_file := utl_file.fopen ('MY_XML_FILES','f-
ile.xml','W');
QueryContext dbms_xmlgen.ctxhandle;
QueryContext := dbms_xmlgen.newcontext ('sql_que-
ry');
```

```
dbms_xmlgen.setrowsettag (QueryContext, 'xxx');
dbms_xmlgen.setrowtag (QueryContext, 'xxx');
```

```
my_clob := dbms_xmlgen.getxml (QueryContext);
--write that XML to our external file
utl_file.put (out_file, my_clob);
utl_file.fclose (out_file);
dbms_xmlgen.closecontext (QueryContext);
```

### xml --> relational table

external.xml file      xmltype col type

get external file into col: xmltype(bfilename('DIR','fname.xml'))

Query xml col

extract(xml\_doc,'Path')      return xml

extract(xml\_doc,'Path/text()')      return value only

existnode(xml\_doc,'path').getdtringval      same as text()

existsnode(xml\_doc, 'path')      if exists =1, else 0

dot notation      nested element

[i] to access array element

to avoid access violation error(11g):

```
select XMLSERIALIZE (CONTENT xmldocument AS CLOB INDENT
SIZE = 2) from xmldocuments
```



### XML\_TABLE

```
VARIABLE v_job VARCHAR2(10);
EXEC :v_job := 'CLERK';
SELECT xt.*
FROM xml_tab x,
     XMLTABLE('/employees/employee[job=$job]'
              PASSING x.xml_data, :v_job AS "job"
              COLUMNS
                 empno VARCHAR2(4) PATH 'empno',
                 ename VARCHAR2(10) PATH 'ename',
                 job VARCHAR2(9) PATH 'job',
                 hiredate VARCHAR2(11) PATH 'hiredate',
                 id VARCHAR2(10) PATH '@id',
              ) xt;
-- :v_job AS "job"
--The variable must be aliases using AS and double
quoted to make sure the name and case matches that
of the variable in the XPath expression.
--
SELECT x.xml_data.getClobVal()
FROM xml_tab x;
--nested XML_table
SELECT xt.*
FROM xml_tab x,
     XMLTABLE('/departments/department'
              PASSING x.xml_data
              COLUMNS
                 deptno VARCHAR2(4) PATH 'dept_no',
                 XMLTABLE('/employees/employee'
                           PASSING dd.employees
                           COLUMNS
                              empno varchar2(4) PATH 'emp_no',
                              ename varchar2(4) PATH 'emp_name'
```

### XML\_TABLE (cont)

```

)
) xt
--form xmltype in variable
PASSING xmltype(v_varchar2)
PASSING v_xml
```

### XDB

#### Resource\_view

res	xmltype
any_path	varchar2
resid	raw

#### Path\_view

path	varchar2
res	xmltype
link	xmltype
resid	raw

#### xml function

