GUIA PARA UTILIZAR LAS FUNCIONES DE ORACLE QUE PERMITEN GENERAR UN ARCHIVO XML

Introducción

Este es un documento ha sido elaborado como guía para una realizar la creación de un archivo xml utilizando las funciones de oracle

XMLElement y XMLAttributes

En el siguiente ejemplo el usuario puede utilizar XMLElement para generar un elemento de XML para cada empleado:

```
SELECT XMLElement("empleado",
last_name) FROM employees;

SELECT XMLELEMENT("Hiredate", e.hire_date) AS "Emp
Element" FROM employees e
WHERE employee_id = 1005;
```

XMLElement es una de las funciones más importantes de SQL/XML. XMLElement es un identificador de la misma manera que un nombre de la tabla o un nombre de la columna es un identificador, para este caso se utilizan las comillas dobles en el query de XMLElement. Si el usuario desea un nombre del elemento, inclúyalo dentro de comillas dobles. *Es importante entender que la función de XMLElement devuelve un valor de tipo XMLType, no una cadena de caracteres.*

El ejemplo utiliza la función XmlElement con el XMLAttributes anidados para crear elementos XML con valores de atributo para el elemento de nivel superior:

```
SELECT XMLELEMENT("Emp",
      XMLATTRIBUTES(e.employee_id AS "ID",
      e.last_name), XMLELEMENT("Dept",
      e.department_id), XMLELEMENT("Salary",
      e.salary)) AS "Emp Element"
   FROM employees e
   WHERE e.employee_id = 1002;

   Select XMLElement("Emp", XMLATTRIBUTES(e.employee_id AS "ID",
   e.last_name), XMLELEMENT("Dept", e.department_id),
   XMLElement("Salary", XMLATTRIBUTES(e.Job_ID AS "Ocupacion"),
   e.salary)) From Employees e;
```

El ejemplo siguiente utiliza una subconsulta en la XMLAttributes para recuperar información de otra tabla en los atributos de un elemento:

```
SELECT XMLELEMENT("Emp", XMLATTRIBUTES(e.employee_id,
   e.last_name), XMLELEMENT("Dept",
   XMLATTRIBUTES(e.department_id,
   (SELECT d.department_name FROM departments d
   WHERE d.department_id = e.department_id) as
   "Dept_name")), XMLELEMENT("salary", e.salary))
FROM EMPLOYEES e;

SELECT XMLELEMENT("Emp", XMLATTRIBUTES(e.employee_id, e.last_name),
```

```
                    XMLELEMENT("Dept",
XMLATTRIBUTES(e.department_id),(SELECT d.department_name FROM
departments d WHERE d.department_id = e.department_id) ),
                    XMLELEMENT("salary", e.salary))
FROM EMPLOYEES e;
```

XMLForest

XMLFOREST convierte cada uno de sus parámetros de argumento para XML y devuelve un fragmento de XML que es la concatenación de estos argumentos convertidos.

En el ejemplo siguiente, se crea un elemento Emp para un subconjunto de los empleados, con employee_id anidada, apellidos y elementos salariales como el contenido de Emp:

```
SELECT XMLELEMENT("Emp",
   XMLFOREST(e.employee_id, e.last_name,
   e.salary)) "Emp Element"
   FROM employees e;
```

XMLAgg

Es una función de agregación. A partir de una colección de fragmentos XML, devuelve un único documento agregado, similar a sum y avg.

```
SELECT XMLELEMENT("Department",
   XMLAGG(XMLELEMENT("Employee",
   e.job_id||'
   '||e.last_name) ORDER BY
   last_name))
   as "Dept_list"
   FROM employees
   e
   WHERE e.department_id = 1;
```

XMLColattval

Crea un fragmento XML, con etiqueta COLUMN y un atributo NAME, que lo iguala al nombre de la columna. Podemos cambiar el valor del atributo mediante el alias.

```
SELECT XMLELEMENT("Emp",
   XMLCOLATTVAL(e.employee_id, e.last_name, e.salary)) "Emp
   Element" FROM employees e
   WHERE employee_id = 1002;
```

XMLComment

Permite crear un comentario

```
SELECT XMLCOMMENT('OrderAnalysisComp imported,
reconfigured, disassembled')
   AS "XMLCOMMENT" FROM DUAL;
```

XMLConcat

Produce un valor XML dado dos o más expresiones de tipo XML, si alguna de las expresiones se evalúa como nulo, es ignorada.

```
SELECT XMLCONCAT(XMLELEMENT("last", e.last_name),
   XMLELEMENT("email", e.email)) AS
   "Result" FROM employees e
```

```
   WHERE e.employee_id > 202;
```

XMLQuery

La función XMLQUERY devuelve un valor XML a partir de la evaluación de una expresión XQuery utilizando los argumentos de entrada especificados como variables XQuery.

```
Select id, nombre,
      XMLQuery( 'for $i in
      /POSTBOX
      where $i/MailAddressTo/Zipcode =
      "22334" return <Details>
      <Zipcode num="{$i/MailAddressTo/Zipcode}"/>
      <CityName char="{$i/MailAddressTo/City}"/>
      </Details>'
      PASSING direccion RETURNING CONTENT)
Personas FROM person2;
```

Salida:

```
WAREHOUSE_ID Area      Big_warehouses
------------ --------- ------------------------------------------------------
          -- 1  25000
          2     50000
          3     85700 <Details><Docks></Docks><Rail>false</Rail></Details>
          4    103000 <Details><Docks num="3"></Docks><Rail>true</Rail></Details>
```

XMLTable

Transforma datos XML en formato tabla.
Evalúa una expresión XQuery o XPath “el patrón fila” Cada ítem del resultado es una fila

Los valores de los elementos/atributos se mapean a valores de columna usando expresiones XPath “el patrón columna”. El nombre y tipo de dato tienen que especificarse. Se puede asignar valores por defecto. Se puede generar una columna ORDINALITY que contiene un nº secuencial del ítem XQuery al que corresponde.

```
SELECT warehouse_name warehouse,
   warehouse2."Water",
   warehouse2."Rail" FROM warehouses,
   XMLTABLE('/Warehouse'
      PASSING
      warehouses.warehouse_spec
      COLUMNS
         "Water" varchar2(6) PATH
         '/Warehouse/WaterAccess', "Rail" varchar2(6)
         PATH '/Warehouse/RailAccess')
      warehouse2;
```

Salida:

```
WAREHOUSE                          Water  Rail
---------------------------------- ------ ------
Southlake, Texas                   Y      N
San Francisco                      Y      N
New Jersey                         N      N
Seattle, Washington                N      Y
```

Sentencias sobre la tabla EMP

Sentencias select en formato XML

```
SELECT XMLELEMENT ("EMP_ROW", XMLFOREST (EMPNO, ENAME, JOB,
HIREDATE, SAL, COMM, DEPTNO )) FILA_XML
FROM EMP;
```

Creación de una tabla que almacena datos de tipo XML

```
CREATE TABLE TABLA_XML_PRUEBA (COD NUMBER, DATOS XMLTYPE);
```

Inserción en la tabla anterior

```
INSERT INTO TABLA_XML_PRUEBA VALUES (1,
XMLTYPE('<FILA_EMP><EMPNO>123</EMPNO><APELLIDO>GARCIA</APELLIDO>
<OFICIO>FONTANERO</OFICIO><SALARIO>1300</SALARIO></FILA_EMP>'));
INSERT INTO TABLA_XML_PRUEBA VALUES (2,
XMLTYPE('<FILA_EMP><EMPNO>124</EMPNO><APELLIDO>RUIZ</APELLIDO><OFICIO>
ALBAÑIL</OFICIO><SALARIO>1100</SALARIO></FILA_EMP>'));
```

Extracción de datos mediante la sentencia XPATH

```
SELECT EXTRACTVALUE (DATOS, '/FILA_EMP/APELLIDO') FROM
TABLA_XML_PRUEBA;

SELECT EXTRACTVALUE (DATOS, '/FILA_EMP/APELLIDO') FROM
TABLA_XML_PRUEBA
WHERE EXISTSNODE (DATOS, '/FILA_EMP[EMPNO=123]')=1;
```

Devuelve la tabla en fichero XML organizado.

```
select dbms_xmlgen.getxml
('select * from emp')from dual;
```