Bases de Datos Avanzadas

Práctica no. 4:

Manipulación de documentos XML en una base de datos relacional

**Número de Práctica: 4**

**Nombre de la Práctica: Manipulación de documentos XML en una base de datos relacional**

REQUISITOS PREVIOS

Para la realización de la práctica se deberá tener instalado en la computadora el software:

* [Oracle Server 11g Express Edition (XE)](https://drive.google.com/file/d/1P9EVa3eU5Fp3v80ojSSzDFpSzoH6nMaH/view?usp=sharing)
* [Oracle SQL Developer](https://drive.google.com/file/d/1wJjee6Cw_GopctKiDW1tIZ3HSvicFe64/view?usp=sharing)
* [Script de la práctica 1](https://drive.google.com/file/d/1JA-6_UmJzs_x90jNm1AO4oVLEdR7TiRA/view?usp=sharing)

MARCO TEÓRICO

**Almacenamiento y consulta de documentos XML**

Para almacenar información en formato XML en una tabla Oracle, se tienen las opciones de crear atributos CLOB (Character Large Object) o de tipo XMLType. Aunque ambos son válidos, si se desea utilizarlos como contenedores y la gestión de información se realizará desde una aplicación externa, puede ser mejor opción crear un atributo del tipo CLOB. Los atributos CLOB guardarán contenido XML básicamente como texto plano. En cambio, si se desea trabajar con el contenido de estos atributos directamente desde la base de datos, puede resultar más cómodo hacerlo con atributos del tipo XMLType, ya que el contenido almacenado está preparado y listo para ser utilizado.

Considere las siguientes definiciones:

**CREATE TABLE** alumnosXMLType

( id number **PRIMARY KEY**,

alumno **XMLType**

);

**CREATE TABLE** alumnosClob

( id number **PRIMARY KEY**,

alumno CLOB

);

Para insertar datos en ambas tablas se usa:

**INSERT INTO** alumnosXMLType

**VALUES**(1, **XMLType**('<?xml version="1.0"?>

<ALUMNO>

<NOMBRE>Pedro</NOMBRE>

<APELLIDOS>Díaz</APELLIDOS>

<DIRECCION>

<CALLE>Jacarandas 34</CALLE>

<COLONIA>Juan Escutia</COLONIA>

<ESTADO>Veracruz</ESTADO>

</DIRECCION>

<CURSO>1</CURSO>

</ALUMNO>'));

/

**INSERT INTO** alumnosClob

**VALUES**(1, '<?xml version="1.0"?>

<ALUMNO>

<NOMBRE>Pedro</NOMBRE>

<APELLIDOS>Díaz</APELLIDOS>

<DIRECCION>

<CALLE>Jacarandas 34</CALLE>

<COLONIA>Juan Escutia</COLONIA>

<ESTADO>Veracruz</ESTADO>

</DIRECCION>

<CURSO>1</CURSO>

</ALUMNO>');

Aunque el almacenamiento es prácticamente el mismo, el uso de *XMLType* vigila la buena formación del documento XML, así como las capacidades de búsqueda y procesamiento del documento dentro del propio manejador de bases de datos.

**SELECT** a.alumno.**extract**('/ALUMNO/NOMBRE/text()')

**FROM** alumnosXMLType a

**WHERE** id=1;

Oracle proporciona una serie de funciones propias para el manejo de documentos XML:

**XMLPARSE** - Convierte una cadena de caracteres que contiene datos XML en un valor (instancia) de tipo XML.

**XMLSERIALIZE** - Obtiene una representación en string o LOB de un dato de tipo XML

**XMLELEMENT** - Crea un fragmento de un documento XML desde datos relacionales. Los nombres de los elementos se insertan en la expresión. Los valores de las columnas se convierten en valores de texto del elemento. Las instancias anidadas de XMLELEMENT se utilizan para crear documentos XML bien formados.

**XMLATTRIBUTES** - Una expresión anidada de XMLELEMENT(). Proporciona una lista de atributos para un elemento como pares de valores. Los nombres de los atributos se pueden insertar dentro de la expresión. Por defecto, los nombres de las columnas se convierten en nombres de atributos y los valores de las columnas en valores de atributos.

**XMLFOREST** - Crea fragmentos de documentos XML desde datos relacionales. Por defecto, cada nombre de columna se convierte en nombre de un elemento y cada valor de columna en el valor de texto del elemento. Las instancias anidadas de XMLELEMENT y XMLFOREST se utilizan para formar documentos XML bien formados.

**XMLCONCAT** - Produce también atributos para las columnas, pero con una diferencia importante. La función **XMLCOLATTVAL** - Produce un elemento denominado column para cada valor de columna y un atributo denominado name para cada nombre de columna. El valor de la columna es el valor de texto del elemento

**XMLAGG** - Crea un agregado de fragmentos de un documento XML desde una colección de objetos XMLType separados. SYS\_XMLAGG anida el resultado dentro del elemento raíz ROWSET, XMLAGG no hace esto.

**EXTRACT** - Retorna un fragmento de un documento XML en un formato de tipo de dato XMLType desde una expresión XPath.

**EXTRACTVALUE** - Retorna un valor escalar desde una expresión XPath

**EXISTNODE** - Si el resultado de una expresión XPath devuelve cualquier nodo, la función retorna verdadero.

**XMLQUERY** - La función XMLQuery permite construir o consultar datos XML.

**XMLTABLE** - La función XMLTABLE puede tomar datos XML como entrada y producir una tabla relacional como salida. 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.

**UPDATEXML**- Función que permite la actualización parcial de un documento almacenado como un valor XMLType. Permite realizar múltiples cambios en una sola operación. Cada cambio consiste en una expresión XPath que identifica el nodo a ser actualizado y el nuevo valor para ese nodo

**DELETEXML** – Elimina un elemento de un documento de acuerdo a la evaluación de una expresión XPath de búsqueda.

Sintaxis de la función XMLQUERY:

**XMLQUERY** (

Expresión-XQuery

[ **PASSING** { **BY REF** | **BY VALUE** } argument-list ]

[**RETURNING** { **CONTENT** | **SEQUENCE** } [{ **BY REF** | **BY VALUE** }]]

**NULL ON EMPTY** | **EMPTY ON EMPTY**)

DESARROLLO DE LA PRÁCTICA:

a) Abra una ventana de la Línea de Comandos SQL de Oracle, y escriba las siguientes sentencias:

CONNECT / AS SYSDBA;

CREATE USER bdxml IDENTIFIED BY bdxml;

GRANT CONNECT, RESOURCE TO bdxml;

GRANT CREATE ANY DIRECTORY TO bdxml;

GRANT DROP ANY DIRECTORY TO bdxml;

GRANT SELECT ON BD\_REL.EMPLEADO TO BDXML;

GRANT SELECT ON BD\_REL.PROYECTO TO BDXML;

GRANT SELECT ON BD\_REL.DEPENDIENTE TO BDXML;

GRANT SELECT ON BD\_REL.LOCALIZACION\_DPTO TO BDXML;

GRANT SELECT ON BD\_REL.PROYECTO TO BDXML;

GRANT SELECT ON BD\_REL.TRABAJA TO BDXML;

DISCONNECT;

CONNECT bdxml/bdxml;

ALTER SESSION SET NLS\_TERRITORY='MEXICO';

SET PAGESIZE 500;

SET LINESIZE 300;

SET LONG 2000;

b) Descargue los archivos *empleados.xml, empleados\_ref.xml, departamentos\_ref.xml, dependientes\_ref.xml, proyectos.xml, localizaciones\_ref.xml, y trabaja\_ref.xml* siguiendo las indicaciones del profesor, y copíelos en la carpeta **c:\repositorio** (debe crear la carpeta antes). Ejecute las siguientes sentencias, describa que realiza cada una de ellas y muestre el resultado obtenido de las marcadas en color amarillo. Pueden ser capturas de pantallas, vigilando que sean legibles de los resultados obtenidos, o el texto generado por la consola de comandos SQL o la herramienta SQL Developer.

CREATE TABLE empleados\_xml OF XMLType;

SELECT TABLE\_NAME FROM USER\_XML\_TABLES;

Imagen que contiene Texto

Descripción generada automáticamente

CREATE OR REPLACE DIRECTORY REPOSITORIO AS 'c:\repositorio';

INSERT INTO empleados\_xml VALUES (XMLType(BFILENAME('REPOSITORIO', 'empleados.xml'), NLS\_CHARSET\_ID('AL32UTF16')));

SELECT \* FROM empleados\_xml;

Texto

Descripción generada automáticamente

SELECT EXTRACT(e.OBJECT\_VALUE, '/empleados/empleado/paterno').getstringval() AS res FROM empleados\_xml e;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

SELECT EXTRACTVALUE(e.OBJECT\_VALUE, '/empleados/empleado[@id=101]/paterno') AS res FROM empleados\_xml e;

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

SELECT XMLQuery('for $e in /empleados/empleado

where $e/salario > 10000

order by $e/@id

return <empleado nombre="{$e/nombre}" apellidos="{$e/paterno} {$e/materno}"/>'

PASSING e.OBJECT\_VALUE RETURNING CONTENT) AS XML FROM empleados\_xml e;

Texto

Descripción generada automáticamente

SELECT EXTRACT(e.OBJECT\_VALUE, '/empleados/empleado[sexo="F"]/nombre').getstringval() FROM empleados\_xml e;

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

SELECT EXTRACT(e.OBJECT\_VALUE, '/empleados/empleado[sexo="F"]/nombre').getstringval() FROM empleados\_xml e WHERE existsNode(OBJECT\_VALUE, '//proy') = 1;

Texto

Descripción generada automáticamente

SELECT EXTRACT(e.OBJECT\_VALUE, '//empleado[sexo="F" and edad>="30"]/nombre').getstringval() FROM empleados\_xml e;

Imagen que contiene Texto

Descripción generada automáticamente

SELECT XMLRoot(e.OBJECT\_VALUE, VERSION '1.0', STANDALONE YES) FROM empleados\_xml e;

Texto

Descripción generada automáticamente

SELECT XMLSerialize(DOCUMENT e.OBJECT\_VALUE AS CLOB) AS XML FROM empleados\_xml e;

Texto

Descripción generada automáticamente

COLUMN RESULT FORMAT A300;

CREATE TABLE personas (

RFC CHAR(13) PRIMARY KEY,

Nombre VARCHAR(30) NOT NULL,

Apellidos VARCHAR(30) NOT NULL,

CV XMLType);

INSERT INTO personas VALUES ('PELJ900304JJ6', 'Juan', 'Pérez López', XMLType('<curriculum\_vitae><empleos><empleo><empresa>Fábricas de Cartón </empresa><periodo>junio/2011-enero/2017</periodo></empleo> <empleo><empresa>Comercializadora Internacional </empresa><periodo>marzo/2017-febrero/2020</periodo></empleo></empleos><formacion><secundaria>Heroes de Nacozari</secundaria><medio\_superior><escuela>Escuela preparatoria No.5 </escuela><comprobante>certificado</comprobante></medio\_superior></formacion></curriculum\_vitae>'));

INSERT INTO personas VALUES ('CACM871109LI8', 'Martha', 'Carbajal Carbajal', XMLType('<curriculum\_vitae><empleos><empleo><empresa>Comercializadora de Ropa del Centro </empresa><periodo>abril/2010-enero/2011</periodo></empleo> <empleo><empresa>Refacciones para Autobuses S.A.</empresa><periodo>febrero/2011-febrero/2012</periodo></empleo></empleos><formacion><secundaria>Independencia</secundaria><medio\_superior><escuela>Vocacional No. 3 </escuela><comprobante>certificado</comprobante></medio\_superior><superior><escuela>Escuela Internacional de Comercio C.V.</escuela><comprobante>titulo profesional</comprobante></superior></formacion></curriculum\_vitae>'));

INSERT INTO personas VALUES ('CALJ851211K9O', 'Juan Manuel', 'Camacho López', XMLType('<curriculum\_vitae><empleos><empleo><empresa>Banca de Desarrollo Empresarial </empresa><periodo>diciembre/2005-mayo/2009</periodo></empleo> <empleo><empresa>Asesores Fiscales S.A.</empresa><periodo>julio/2009-marzo/2012</periodo></empleo></empleos><formacion><secundaria>Marruecos</secundaria><medio\_superior><escuela>Escuela Comercial y Contable </escuela><comprobante>certificado</comprobante></medio\_superior></formacion></curriculum\_vitae>'));

INSERT INTO personas VALUES ('MEGR910508PY3', 'Rodrigo', 'Medina García', XMLType('<curriculum\_vitae><empleos><empleo><empresa>Armadora de Motores Nacionales </empresa><periodo>abril/2010-junio/2012</periodo></empleo></empleos><formacion><secundaria>Vasco de Quiroga</secundaria></formacion></curriculum\_vitae>'));

COMMIT;

SELECT \* FROM personas;

Texto

Descripción generada automáticamente

SELECT XMLColattval(p.apellidos) FROM personas p;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//empresa').getstringval() AS RESULT FROM personas p WHERE p.nombre LIKE '%Martha%';

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, EXTRACTVALUE(p.cv,'//empresa[contains(.,"S.A.")]') AS RESULT FROM personas p;

Texto

Descripción generada automáticamente con confianza media

SELECT p.nombre || ' ' || p.apellidos AS persona, EXTRACTVALUE(p.cv, '//empresa[contains(.,"S.A.")]') AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//empresa[contains(.,"S.A.")]') = 1;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//medio\_superior/escuela/text() | //superior/escuela/text()').getstringval() AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//medio\_superior/comprobante | //superior/comprobante') = 1;

Texto

Descripción generada automáticamente con confianza baja

SELECT p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//empleos/empleo/empresa/text()').getstringval() AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//empleos[count(empleo)>1]') = 1;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLQuery( 'for $i in //formacion return <salida>{$i//escuela}</salida>' PASSING p.cv RETURNING CONTENT) AS RESULT FROM personas p;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLQuery( 'for $i in //empleo return <salida>{$i/empresa}</salida>' PASSING p.cv RETURNING CONTENT) AS RESULT FROM personas p;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLElement("salida", XMLQuery( 'for $i in //empleo/empresa return <empresa>{data($i)}</empresa>' PASSING p.cv RETURNING CONTENT)) AS RESULT FROM personas p;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLElement("salida", XMLQuery( 'for $i in //empresa[contains(.,"S.A.")] return <empresa>{data($i)}</empresa>' PASSING p.cv RETURNING CONTENT)) AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//empresa[contains(.,"S.A.")]') = 1;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLElement("salida", XMLQuery( 'for $i in //medio\_superior/escuela/text() | //superior/escuela/text() return <escuela>{data($i)}</escuela>' PASSING p.cv RETURNING CONTENT)) AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//medio\_superior/escuela/text() | //superior/comprobante') = 1;

Texto

Descripción generada automáticamente

SELECT p.nombre || ' ' || p.apellidos AS persona, XMLElement("salida", XMLQuery( 'for $i in //empleos/empleo/empresa/text() return <empleo>{data($i)}</empleo>' PASSING p.cv RETURNING CONTENT)) AS RESULT FROM personas p WHERE EXISTSNODE(p.cv, '//empleos[count(empleo)>1]') = 1;

Texto

Descripción generada automáticamente con confianza media

SELECT p.rfc, p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//comprobante').getstringval() AS RESULT FROM personas p WHERE p.nombre LIKE '%Martha%';

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

UPDATE personas p SET p.cv = updateXML(p.cv, '//medio\_superior/comprobante', '<comprobante>diploma</comprobante>') WHERE p.rfc = 'CACM871109LI8';

SELECT p.rfc, p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//comprobante').getstringval() AS RESULT FROM personas p WHERE p.nombre LIKE '%Martha%';

Texto

Descripción generada automáticamente

UPDATE personas p SET p.cv = deleteXML(p.cv, '//medio\_superior/comprobante["diploma"]') WHERE p.rfc = 'CACM871109LI8';

SELECT p.rfc, p.nombre || ' ' || p.apellidos AS persona, EXTRACT(p.cv, '//comprobante').getstringval() AS RESULT FROM personas p WHERE p.nombre LIKE '%Martha%';

Imagen que contiene Diagrama

Descripción generada automáticamente

DECLARE

res BOOLEAN;

empsxmlstring VARCHAR2(4000):=

'<?xml version="1.0"?><empleados><empleado NSS="777888999" departamento="3"><nombre>Jesús</nombre><apellido>López</apellido><fecha\_nac>1973-04-04</fecha\_nac><direccion>Calle Venus 45, col. Irrigación,Ecatepec,89765</direccion><genero>M</genero><salario>50000</salario><titulo>Ingeniero</titulo><fecha\_ingreso>2000-04-05</fecha\_ingreso></empleado><empleado NSS="222333444 " Jefe="777888999" departamento="3"><nombre>Guadalupe</nombre><apellido>Oñate</apellido><fecha\_nac>1969-11-24</fecha\_nac><direccion>Av.Revolución 348, col. Fuentes, San Ignacio, 67656</direccion><genero>F</genero><salario>25000</salario><titulo>Licenciado</titulo><fecha\_ingreso>2000-04-07</fecha\_ingreso></empleado></empleados>';

<empleado NSS="444555666" Jefe="111222333" departamento="1"><nombre>Julia</nombre><apellido>Regalado</apellido><fecha\_nac>1975-07-30</fecha\_nac><direccion>Calle cielo 76, col. Independencia, Tepeji, 34256</direccion><genero>F</genero><salario>28000</salario><titulo>Ingeniero</titulo><fecha\_ingreso>2001-06-28</fecha\_ingreso></empleado><empleado NSS="555666777" Jefe="777888999" departamento="3"><nombre>Mario</nombre><apellido>Medina</apellido><fecha\_nac>1977-10-01</fecha\_nac><direccion>Av.Politécnico 650, col. Profesiones, CDMX, 09765</direccion><genero>M</genero><salario>20000</salario><titulo>Licenciado</titulo><fecha\_ingreso>2100-05-30</fecha\_ingreso></empleado> <empleado NSS="333444555" departamento="2"><nombre>Rogelio</nombre><apellido>Calzada</apellido><fecha\_nac>1965-03-25</fecha\_nac><direccion>Av.Independencia 123, col. Fuentes, San Ignacio, 56565</direccion><genero>M</genero><salario>39000</salario><titulo>Licenciado</titulo><fecha\_ingreso>2000-04-05</fecha\_ingreso></empleado><empleado NSS="666777888" Jefe="111222333" departamento="1"><nombre>Bruce</nombre><apellido>Bolaños</apellido><fecha\_nac>1963-08-07</fecha\_nac><direccion>Mar Mediterráneo 56, col. Lomas Lindas, CDMX, 09879</direccion><genero>M</genero><salario>24000</salario><titulo>Licenciado</titulo><fecha\_ingreso>2100-12-17</fecha\_ingreso></empleado>

<empleado NSS="999000111" Jefe="333444555" departamento="2"><nombre>Laura</nombre><apellido>Méndez</apellido><fecha\_nac>1982-02-20</fecha\_nac><direccion>Av.Montevideo 98, col. Ampliación Hidalgo, Texcoco, 45679</direccion><genero>F</genero><salario>18000</salario><titulo>Ingeniero</titulo><fecha\_ingreso>2100-12-20</fecha\_ingreso></empleado><empleado NSS="111222333" departamento="1"><nombre>Sandra</nombre><apellido>Guzmán</apellido><fecha\_nac>1970-11-27</fecha\_nac><direccion>Av.Siempreviva 444, col. San Mateo, Ecatepec, 89740</direccion><genero>F</genero><salario>45000</salario><titulo>Ingeniero</titulo><fecha\_ingreso>2000-04-05</fecha\_ingreso></empleado>

<empleado NSS="888999000" Jefe="333444555" departamento="2"><nombre>Guadalupe</nombre><apellido>Hidalgo</apellido><fecha\_nac>1985-03-19</fecha\_nac><direccion>Av.Miguel Hidalgo 56, col. Independencia, Tepeji, 87981</direccion><genero>F</genero><salario>27000</salario><titulo>Ingeniero</titulo><fecha\_ingreso>2001-10-11</fecha\_ingreso></empleado></empleados>';

deptsxmlstring VARCHAR2(4000):=

'<?xml version="1.0"?>

<departamentos><departamento numero="1" Jefe="111222333"><nombre>Sistemas</nombre><fecha\_inicio>2003-03-08</fecha\_inicio><fecha\_fin></fecha\_fin></departamento><departamento numero="2" Jefe="333444555"><nombre>Ventas</nombre><fecha\_inicio>2001-11-11</fecha\_inicio><fecha\_fin></fecha\_fin></departamento><departamento numero="3" Jefe="777888999"><nombre>Administración</nombre><fecha\_inicio>2000-07-03</fecha\_inicio><fecha\_fin></fecha\_fin></departamento></departamentos>';

BEGIN

res := DBMS\_XDB.createResource('/public/empleados.xml', empsxmlstring);

res := DBMS\_XDB.createResource('/public/departamentos.xml', deptsxmlstring);

END;

/

COMMIT;

Ejecuta un error en la sentencia a la hora de cargar los datos necesarios para la practica.

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]/nombre

return <empleado nombre="{$e/nombre}" departamento="{$d}"/>'

RETURNING CONTENT) AS RESULT FROM DUAL;

Texto

Descripción generada automáticamente

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

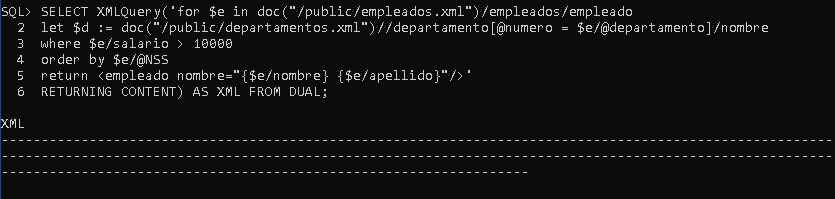
let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]/nombre

where $e/salario > 10000

order by $e/@NSS

return <empleado nombre="{$e/nombre} {$e/apellido}"/>'

RETURNING CONTENT) AS RESULT FROM DUAL;



SELECT XMLQuery('for $d in doc("/public/departamentos.xml")/departamentos/departamento/@numero

let $e := doc("/public/empleados.xml")/empleados/empleado[@departamento = $d]

where count($e) > 1

order by avg($e/salario) descending

return <departamento>{$d}<conteo>{count($e)}</conteo><promediosal>{round(avg($e/salario))}</promediosal></departamento>'

RETURNING CONTENT) AS RESULT FROM DUAL;



SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

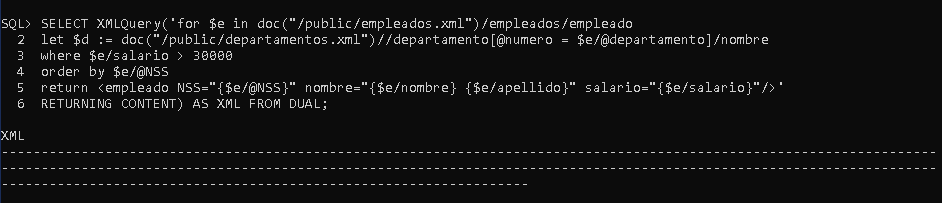
let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]/nombre

where $e/salario > 30000

order by $e/@NSS

return <empleado NSS="{$e/@NSS}" nombre="{$e/nombre} {$e/apellido}" salario="{$e/salario}"/>'

RETURNING CONTENT) AS RESULT FROM DUAL;



Todas dan resultados vacios

Texto

Descripción generada automáticamente

BEGIN

DBMS\_XDB.deleteResource('/public/empleados.xml');

DBMS\_XDB.deleteResource('/public/departamentos.xml');

END;

/

COMMIT;

DECLARE

res BOOLEAN;

BEGIN

res := DBMS\_XDB.createResource('/public/empleados.xml', BFILENAME('REPOSITORIO', 'empleados\_ref.xml'));

res := DBMS\_XDB.createResource('/public/dependientes.xml', BFILENAME('REPOSITORIO', 'dependientes\_ref.xml'));

res := DBMS\_XDB.createResource('/public/departamentos.xml', BFILENAME('REPOSITORIO', 'departamentos\_ref.xml'));

res := DBMS\_XDB.createResource('/public/localizaciones.xml', BFILENAME('REPOSITORIO', 'localizaciones\_ref.xml'));

res := DBMS\_XDB.createResource('/public/proyectos.xml', BFILENAME('REPOSITORIO', 'proyectos\_ref.xml'));

res := DBMS\_XDB.createResource('/public/trabaja.xml', BFILENAME('REPOSITORIO', 'trabaja\_ref.xml'));

END;

/

COMMIT;

SELECT e.OBJECT\_VALUE FROM empleados e;

SELECT d.OBJECT\_VALUE FROM dependientes d;

SELECT d.OBJECT\_VALUE FROM departamentos d;

SELECT l.OBJECT\_VALUE FROM localizaciones l;

SELECT p.OBJECT\_VALUE FROM proyectos p;

SELECT t.OBJECT\_VALUE FROM trabaja t;

Evaluacion de select con caracte vacio ,derivado dela linea con error

Texto

Descripción generada automáticamente

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]/nombre

return <empleado NSS="{$e/@NSS}" departamento="{$d}">{concat($e/apellido, ", ", $e/nombre)}</empleado>'

RETURNING CONTENT) AS RESULT FROM DUAL;

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]

let $p := doc("/public/proyectos.xml")/proyectos/proyecto[@departamento = $d/@numero]/nombre

return <empleado NSS="{$e/@NSS}" departamento="{$d/nombre}" proyecto="{$p}">{concat($e/apellido, ", ", $e/nombre)}</empleado>'

RETURNING CONTENT) AS RESULT FROM DUAL;

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

let $d := doc("/public/departamentos.xml")//departamento[@numero = $e/@departamento]

let $p := doc("/public/proyectos.xml")/proyectos/proyecto[@departamento = $d/@numero]

let $t := doc("/public/trabaja.xml")//trabaja[@empleado = $e/@NSS]

where $t/horas/@proyecto = $p/@numero

return <empleado NSS="{$e/@NSS}" departamento="{$d/nombre}" proyecto="{$p/nombre}" horas="{$t/horas/text()}">{concat($e/apellido, ", ", $e/nombre)}</empleado>'

RETURNING CONTENT) AS RESULT FROM DUAL;

SELECT XMLQuery('for $e in doc("/public/empleados.xml")/empleados/empleado

let $d := doc("/public/dependientes.xml")//dependiente[@empleado = $e/@NSS]

return <dependiente empleado="{concat($e/apellido, ", ", $e/nombre)}" parentesco="{$d/@parentesco}">{$d/nombre}{$d/apellido}{$d/genero}</dependiente>'

RETURNING CONTENT) AS RESULT FROM DUAL;

Selecciones vacias derivadas de la linea con error

Texto

Descripción generada automáticamente

c) Para realizar las siguientes consultas, deberá tener construidas y pobladas las tablas de la práctica 1. Refiérase al código de esa práctica y ejecute únicamente las definiciones de tablas y su contenido (sentencias de tipo CREATE TABLE, ALTER TABLE y de tipo INSERT INTO), dentro del usuario **db\_rel**. Si realizó correctamente la práctica 1, no tiene que realizar este procedimiento.

Ejecute las siguientes sentencias y muestre el resultado obtenido de las marcadas en color amarillo. Trabaje con el usuario *bdxml*

SELECT XMLELEMENT("empleado", XMLELEMENT("NSS",e.nss), XMLELEMENT("nombre",e.nombre),XMLELEMENT("apellido",e.apellido),XMLELEMENT("fecha\_nac",e.fecha\_nac),XMLELEMENT("direccion",e.direccion),XMLELEMENT("genero",e.genero),XMLELEMENT("salario",e.salario),XMLELEMENT("titulo",e.titulo),XMLELEMENT("fecha\_ingreso",e.fecha\_ingreso)) AS RESULT FROM bd\_rel.empleado e;

Texto

Descripción generada automáticamente

SELECT XMLELEMENT("empleado", XMLATTRIBUTES(e.nss AS "NSS", e.nss\_jefe as "Jefe", e.nodep as "departamento"), XMLForest(e.nss AS "NSS", e.nombre AS "nombre" , e.apellido AS "apellido", e.fecha\_nac AS "fecha\_nac", e.direccion AS "direccion", e.genero AS "genero", e.salario AS "salario", e.titulo AS "titulo", e.fecha\_ingreso AS "fecha\_ingreso")) AS RESULT FROM bd\_rel.empleado e;

Texto

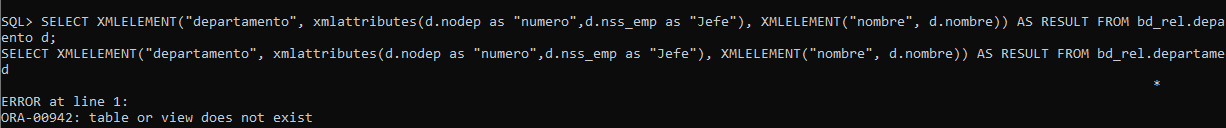
Descripción generada automáticamente

SELECT XMLELEMENT("dependiente", XMLATTRIBUTES(d.nss\_emp as "empleado", d.parentesco as "parentesco"), XMLELEMENT("nombre", d.nombre), XMLELEMENT("apellido", d.apellido), XMLELEMENT("fecha\_nac", d.fecha\_nac), XMLELEMENT("genero", d.genero)) AS RESULT FROM bd\_rel.dependiente d;

Texto

Descripción generada automáticamente

SELECT XMLELEMENT("departamento", xmlattributes(d.nodep as "numero",d.nss\_emp as "Jefe"), XMLELEMENT("nombre", d.nombre)) AS RESULT FROM bd\_rel.departamento d;



SELECT XMLELEMENT("localizacion", xmlattributes(l.nodep as "departamento"), XMLAgg(XMLELEMENT("lugar", l.loc\_dep) ORDER BY l.nodep)) AS RESULT FROM bd\_rel.localizacion\_dpto l GROUP BY l.nodep;

Texto

Descripción generada automáticamente

SELECT XMLELEMENT("proyecto", xmlattributes(p.nop as "numero", p.nodep as "departamento"), XMLELEMENT("nombre", p.nombre), XMLELEMENT("fecha\_inicio", p.fecha\_inicio),XMLELEMENT("fecha\_fin", p.fecha\_fin)) AS RESULT FROM bd\_rel.proyecto p ORDER BY p.nop;

Texto

Descripción generada automáticamente con confianza media

SELECT XMLELEMENT("trabaja", xmlattributes(t.nss as "empleado"), xmlagg(XMLELEMENT("horas", xmlattributes(t.nop as "proyecto"), t.horas) order by t.nss)) AS RESULT FROM bd\_rel.trabaja t GROUP BY t.nss;

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

SELECT XMLParse(CONTENT '<proyectos><proyecto numero="1" departamento="2"><nombre>Mejoramiento de Calidad</nombre></proyecto><proyecto numero="2" departamento="1"><nombre>Sitio Web</nombre></proyecto><proyecto numero="3" departamento="3"><nombre>Publicidad</nombre></proyecto></proyectos>') AS RESULT FROM DUAL d;

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

SELECT XMLParse(CONTENT '<proyectos><proyecto numero="1" departamento="2"><nombre>Mejoramiento de Calidad</nombre></proyecto><proyecto numero="2" departamento="1"><nombre>Sitio Web</nombre></proyecto><proyecto numero="3" departamento="3"><nombre>Publicidad</nombre></proyecto></proyecto>' WELLFORMED) AS RESULT FROM DUAL d;

Imagen que contiene Texto

Descripción generada automáticamente

SELECT XMLRoot(XMLType('<proyectos><proyecto numero="1" departamento="2"><nombre>Mejoramiento de Calidad</nombre></proyecto><proyecto numero="2" departamento="1"><nombre>Sitio Web</nombre></proyecto><proyecto numero="3" departamento="3"><nombre>Publicidad</nombre></proyecto></proyectos>'), VERSION '1.0', STANDALONE YES) AS RESULT FROM DUAL;

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

SELECT XMLRoot(xmlelement("proyecto", xmlattributes(p.nop as "numero", p.nodep as "departamento"), xmlelement("nombre", p.nombre), xmlelement("fecha\_inicio", p.fecha\_inicio), xmlelement("fecha\_fin", p.fecha\_fin)), VERSION '1.0', STANDALONE YES) AS RESULT FROM bd\_rel.proyecto p ORDER BY p.nop;

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

SELECT XMLSerialize(DOCUMENT XMLType('<proyectos><proyecto numero="1" departamento="2"><nombre>Mejoramiento de Calidad</nombre></proyecto><proyecto numero="2" departamento="1"><nombre>Sitio Web</nombre></proyecto><proyecto numero="3" departamento="3"><nombre>Publicidad</nombre></proyecto></proyectos>') AS CLOB) AS RESULT FROM DUAL;

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

SELECT XMLPI(NAME "xml-stylesheet", 'ref=”estilos.xslt”') as estilo, xmlelement("empleado", XMLELEMENT("NSS",e.nss), XMLELEMENT("nombre",e.nombre),XMLELEMENT("apellido",e.apellido),XMLELEMENT("fecha\_nac",e.fecha\_nac),XMLELEMENT("direccion", e.direccion),XMLELEMENT("genero",e.genero),XMLELEMENT("salario",e.salario),XMLELEMENT("titulo",e.titulo),XMLELEMENT("fecha\_ingreso",e.fecha\_ingreso)) AS RESULT FROM bd\_rel.empleado e;

Texto

Descripción generada automáticamente

SELECT sys\_xmlgen(XMLELEMENT("proyecto", XMLATTRIBUTES(p.nop as "numero", p.nodep as "departamento"), XMLELEMENT("nombre", p.nombre),XMLELEMENT("fecha\_inicio", p.fecha\_inicio), XMLELEMENT("fecha\_fin", p.fecha\_fin))) AS RESULT FROM bd\_rel.proyecto p ORDER BY p.nop;

Texto

Descripción generada automáticamente

Incluya una captura o el texto resultante de la ejecución de las sentencias SELECT marcadas en color resaltado amarillo. No se aceptan archivos externos de resultados fuera del contenido de este documento.

CUESTIONARIO

1.- Comente sobre la definición y uso de documentos XML aplicado sobre una base de datos relacional, con respecto a su almacenamiento y restricciones.

* Es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.
* Se utilizan bases de datos relacionales, objeto-relacionales o de objetos puras para representar documentos XML.

2.- Indique las diferencias de modelado de una base de datos relacional con respecto a una basada en XML con respecto a la conservación de la integridad referencial.

* Crean modelos lógicos en XML.
* Mapean los modelos al mecanismo de almacenamiento correspondiente.
* Las operaciones con los documentos se realizan en XML.
* Dan un mayor nivel de abstracción al programador.
* Dependencia del esquema.
* Gestionan documentos como colecciones de datos.
* No todas necesitan un esquema para almacenar documentos.
* Problemas de integridad intra-documento.
* Los esquemas se definen con DTD o con XML Schema (W3C).
* No usan un mecanismo concreto de almacenamiento físico.

CONCLUSIONES

Escriba sus conclusiones (no personales) con respecto al desarrollo de esta práctica.

El uso de herramientas como lo son el command line de oracle , nos permiten dar un enfoque directo con respect a la realizacion de bases de datos con XML , esto siendo asi por que su procesamiento y manejamiento en otras interfaces asi como sus busquedas de datos , son mas amigable con el programador y evitan que se generen una confusion al estar creando las misma.

REFERENCIAS BIBLIOGRÁFICAS (APA, IEEE, ETC.)

1.-Oracle XML DB <http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/index.html>

2.- XML DB Developer's Guide - <http://docs.oracle.com/database/121/ADXDB/toc.htm>

3.- Oracle XML Functions, <http://psoug.org/reference/xml_functions.html>

4.- Manejando XMLType en Oracle Database 11gR2 a través del componente XDB, <http://www.oracle.com/technetwork/es/articles/sql/xmltype-en-database11g-a-traves-xdb-1931103-esa.html>

5.- Trabajando XML en Oracle <https://blogdeaitor.wordpress.com/2013/02/06/trabajar-con-xml-en-oracle/>