Bases de Datos Avanzadas

Práctica no. 5:

Consultas en una base de datos orientada a documentos XML

**Número de Práctica: 5**

**Nombre de la Práctica: Consultas en una base de datos orientada a documentos XML**

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)
* [Archivos XML de los ejercicios](https://drive.google.com/file/d/1ELc5nb94cEPRZf2rrNvA1mGcVy3FX5bp/view?usp=sharing)

MARCO TEÓRICO

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

**XMLPARSE** - Convierte una cadena de caracteres que contiene datos XML en una 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:

Descargue los archivos *empleados\_ref.xml, departamentos\_ref.xml, dependientes\_ref.xml, proyectos.xml, localizaciones\_ref.xml, y trabaja\_ref.xml* que están comprimidos en el archivo *archivos\_xml.zip*, siguiendo las indicaciones del profesor, y cópielos en la carpeta **c:\repositorio** (debe crear la carpeta antes). Ejecute las siguientes sentencias, 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.

1. 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;

1. SELECT e.OBJECT\_VALUE FROM empleados e;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

Ejecute las siguientes sentencias y muestre el resultado obtenido de las marcadas en color amarillo. Trabaje con el usuario *bdxml* definido en la práctica 1.

1. CREATE TABLE EMPLEADO(

NSS CHAR(10) NOT NULL,

NOMBRE VARCHAR2(30) NOT NULL,

APELLIDO VARCHAR2(45),

FECHA\_NAC DATE NOT NULL,

DIRECCION VARCHAR2(120),

GENERO CHAR(1) NOT NULL,

SALARIO NUMBER(7,2),

NSS\_JEFE CHAR(10),

NODEP NUMBER(\*,0),

TITULO VARCHAR2(20) NOT NULL,

FECHA\_INGRESO DATE

);

1. CREATE TABLE DEPARTAMENTO(

NODEP NUMBER(3,0),

NOMBRE VARCHAR2(30) NOT NULL,

NSS\_EMP CHAR(10) NOT NULL

);

CREATE TABLE LOCALIZACION\_DPTO(

NODEP NUMBER NOT NULL,

LOC\_DEP VARCHAR2(40)

);

1. CREATE TABLE DEPENDIENTE(

NOMBRE VARCHAR2(40),

APELLIDO VARCHAR2(45),

FECHA\_NAC DATE NOT NULL,

PARENTESCO VARCHAR2(20) NOT NULL,

NSS\_EMP CHAR(10) NOT NULL,

GENERO CHAR(1) NOT NULL

);

1. CREATE TABLE PROYECTO(

NOP NUMBER NOT NULL,

NOMBRE VARCHAR2(80),

FECHA\_INICIO DATE,

FECHA\_FIN DATE,

NODEP NUMBER,

CONSTRAINT PROY\_PK PRIMARY KEY (NOP),

CONSTRAINT PROY\_UK UNIQUE (NOMBRE)

);

1. CREATE TABLE TRABAJA(

NSS CHAR(10) NOT NULL,

NOP NUMBER NOT NULL,

HORAS NUMBER NOT NULL,

CONSTRAINT TRAB\_PK PRIMARY KEY (NSS, NOP)

);

1. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (777888999,'Jesús','López', TO\_DATE('04/04/1973','dd/mm/yyyy'),'Calle Venus 45, col. Irrigación, Ecatepec, 89765','M',50000,NULL,3,'Ingeniero', TO\_DATE('05/04/2000','dd/mm/yyyy'));
2. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (222333444,'Guadalupe','Oñate', TO\_DATE('24/11/1969','dd/mm/yyyy'),'Av.Revolución 348, col. Fuentes, San Ignacio, 67656','F',25000,777888999,3,'Licenciado', TO\_DATE('07/04/2000','dd/mm/yyyy'));
3. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (444555666,'Julia','Regalado', TO\_DATE('30/07/1975','dd/mm/yyyy'),'Calle cielo 76, col. Independencia, Tepeji, 34256','F',28000,111222333,1,'Ingeniero', TO\_DATE('28/06/2001','dd/mm/yyyy'));
4. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (555666777,'Mario','Medina', TO\_DATE('01/10/1977','dd/mm/yyyy'),'Av.Politécnico 650, col. Profesiones, CDMX, 09765','M',20000,777888999,3,'Licenciado', TO\_DATE('30/05/2000','dd/mm/yyyy'));
5. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (333444555,'Rogelio','Calzada', TO\_DATE('25/03/1965','dd/mm/yyyy'),'Av.Independencia 123, col. Fuentes, San Ignacio, 56565','M',39000,NULL,2,'Licenciado', TO\_DATE('05/04/2000','dd/mm/yyyy'));
6. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (666777888,'Bruce','Bolaños', TO\_DATE('07/08/1963','dd/mm/yyyy'),'Mar Mediterráneo 56, col. Lomas Lindas, CDMX, 09879','M',24000,111222333,1,'Licenciado', TO\_DATE('17/12/2000','dd/mm/yyyy'));
7. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (999000111,'Laura','Méndez', TO\_DATE('20/02/1982','dd/mm/yyyy'),'Av.Montevideo 98, col. Ampliación Hidalgo, Texcoco, 45679','F',18000,333444555,2,'Ingeniero', TO\_DATE('20/12/2000','dd/mm/yyyy'));
8. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (111222333,'Sandra','Guzmán', TO\_DATE('27/11/1970','dd/mm/yyyy'),'Av.Siempreviva 444, col. San Mateo, Ecatepec, 89740','F',45000,NULL,1,'Ingeniero', TO\_DATE('20/12/2000','dd/mm/yyyy'));
9. INSERT INTO EMPLEADO (NSS,NOMBRE,APELLIDO,FECHA\_NAC,DIRECCION,GENERO,SALARIO,NSS\_JEFE,NODEP,TITULO,FECHA\_INGRESO) VALUES (888999000,'Guadalupe','Hidalgo', TO\_DATE('19/03/1985','dd/mm/yyyy'),'Av.Miguel Hidalgo 56, col. Independencia, Tepeji, 87981','F',27000,333444555,2,'Ingeniero', TO\_DATE('11/10/2001','dd/mm/yyyy'));
10. INSERT INTO departamento (NODEP,NOMBRE,NSS\_EMP) VALUES (1,'Sistemas', 111222333);
11. INSERT INTO departamento (NODEP,NOMBRE,NSS\_EMP) VALUES (2,'Ventas', 333444555);
12. INSERT INTO departamento (NODEP,NOMBRE,NSS\_EMP) VALUES (3,'Administración', 777888999);
13. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Gustavo','Guzmán',TO\_DATE('07/12/1999','dd/mm/yyyy'),'hijo',111222333,'M');
14. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Karen','Oñate',TO\_DATE('09/11/1995','dd/mm/yyyy'),'hija',222333444,'F');
15. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Julieta','Castillo',TO\_DATE('09/11/1951','dd/mm/yyyy'),'madre',333444555,'F');
16. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Lucia','Calzada',TO\_DATE('09/11/1995','dd/mm/yyyy'),'hija',333444555,'F');
17. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Ernesto','Medina',TO\_DATE('28/10/1945','dd/mm/yyyy'),'padre',555666777,'M');
18. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Carlos','López',TO\_DATE('12/04/2005','dd/mm/yyyy'),'hijo',777888999,'M');
19. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Karla','Hidalgo',TO\_DATE('26/08/2009','dd/mm/yyyy'),'hija',888999000,'F');
20. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Sergio','Méndez',TO\_DATE('17/12/1992','dd/mm/yyyy'),'hijo',999000111,'M');
21. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Felipe','Bolaños',TO\_DATE('26/08/2009','dd/mm/yyyy'),'nieto',666777888,'M');
22. INSERT INTO dependiente(NOMBRE,APELLIDO,FECHA\_NAC,PARENTESCO,NSS\_EMP,GENERO) VALUES ('Rodrigo','Calzada',TO\_DATE('21/07/1997','dd/mm/yyyy'),'hijo',333444555,'M');
23. INSERT INTO proyecto(NOMBRE,NOP,NODEP,FECHA\_INICIO,FECHA\_FIN) VALUES ('Sitio Web',2,1,TO\_DATE('18/02/2010','dd/mm/YYYY'), TO\_DATE('12/05/2011','dd/mm/YYYY'));
24. INSERT INTO proyecto(NOMBRE,NOP,NODEP,FECHA\_INICIO,FECHA\_FIN) VALUES ('Mejoramiento de Calidad',1,2,TO\_DATE('21/06/2011','dd/mm/YYYY'), TO\_DATE('10/01/2012','dd/mm/YYYY'));
25. INSERT INTO proyecto(NOMBRE,NOP,NODEP,FECHA\_INICIO,FECHA\_FIN) VALUES ('Publicidad',3,3,TO\_DATE('06/09/2011','dd/mm/YYYY'), TO\_DATE('29/08/2012','dd/mm/YYYY'));
26. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (222333444,3,8);
27. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (555666777,2,16);
28. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (888999000,2,8);
29. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (555666777,1,10);
30. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (666777888,1,2);
31. INSERT INTO trabaja(NSS,NOP,HORAS) VALUES (888999000,1,8);
32. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('Monterrey',3);
33. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('Guadalupe',3);
34. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('Juárez',3);
35. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('DF',1);
36. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('DF',2);
37. INSERT INTO localizacion\_dpto(LOC\_DEP,NODEP) VALUES ('Pachuca',2);
38. 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 empleado e;

Texto

Descripción generada automáticamente

1. 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 empleado e;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. 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 dependiente d;

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

1. 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 proyecto p ORDER BY p.nop;

Texto

Descripción generada automáticamente

1. 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 trabaja t GROUP BY t.nss;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. 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>') AS RESULT FROM DUAL d;

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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 proyecto p ORDER BY p.nop;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. 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;

Texto

Descripción generada automáticamente

1. 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 empleado e;

Texto

Descripción generada automáticamente

1. 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 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 las opciones que presenta Oracle para la creación de documentos XML en la base de datos.

Oracle ofrece varias opciones para trabajar con documentos XML en la base de datos. Una opción es almacenar información en formato XML en una tabla Oracle utilizando atributos **CLOB (Character Large Object)** o **XMLType.**

Los atributos CLOB guardarán el XML básicamente como texto plano, mientras que los atributos del tipo XMLType están preparados y listos para ser utilizados directamente desde la base de datos.

2.-¿Qué tecnologías emplea Oracle para la consultas de datos dentro de un documento XML almacenado en la base de datos?

XMLTYPE: Nos ayuda para el manejo de datos XML, puede tener distintas funciones para extraer nodos y grafmetos XML. También se pueden crear columnas de XMLType e insertar documentos XML.

XMLTABLE: Nos ayuda a proyectar columnas sobre nuestros datos XML, esto hace posible las consutas directamente desde SQL como si se tratara de datos relacioneales.

XMLELEMENT: Nos ayuda a devolver un XMLType. Se compone de dos argumentos, el primero es el nombre de la eqtiqueta, el siguiente es el valor y este podría ser una cadera, numero o fecha.

XMLATTRIBUTES: Nos ayuda a convertir los datos de una columna en atributos del elemento padre

CONCLUSIONES

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

Las bases de datos orientadas a documentos en XML son un tipo de base de datos diseñada para almacenar y gestionar información basada en documentos. La información basada en documentos suele almacenarse en un formato fácil de leer y comprender, como un archivo de texto o un archivo XML. Estos datos pueden ser interrogados, exportados y serializados . Las bases de datos XML están generalmente asociadas con las bases de datos documentales.

Basicamente las BDA orientadas a documentos en XML ofrecen una forma más facil de leer para almacenar y poder gestionar info basada en docs.

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/>