# INSTITUTO POLITÉCNICO NACIONALhttp://4.bp.blogspot.com/-uP_hXyaqILI/ToD3k8H-2kI/AAAAAAAAAzM/1eLxyzPmXJM/s1600/ipn%2Blogo2.jpghttp://www.sepi.escom.ipn.mx/WPS/WCM/CONNECT/7A176E804EAA2A3E83949B9BEB1E1CB/ESCOM2BB5.JPG?MOD=AJPERES&CACHEID=7a176e804eaa2a3e83949b9beb1e1cba

# ESCUELA SUPERIOR DE CÓMPUTO

# ACADEMIA DE INGENIERÍA DE SOFTWARE

Bases de Datos No Relacionales

Práctica no. 2:

Consultas en una base de datos orientada a documentos XML

Alumno: Hernández Pérez Diego Francisco

Grupo: 6CV2

**Número de Práctica: 2**

**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=drive_link)

MARCO TEÓRICO

La función XMLQuery permite consultar datos XML en sentencias SQL. Toma una expresión XQuery como un literal de cadena, un elemento de contexto opcional y otras variables de enlace y devuelve el resultado de evaluar la expresión XQuery utilizando estos valores de entrada.

XQuery 1.0 y XPath 2.0 comparten el mismo modelo de datos y admiten las mismas funciones y operadores.

XQuery utiliza funciones para extraer datos de documentos XML.

La función doc(URL) se usa para abrir archivos contenidos en alguna ruta de almacenamiento local o en la web.

XQuery usa expresiones de ruta para navegar a través de elementos en un documento XML.

doc("books.xml")/bookstore/book/title

XQuery usa predicados para limitar los datos extraídos de los documentos XML.

doc("books.xml")/bookstore/book[price < 30]

FLWOR es un acrónimo de "For, Let, Where, Order by, Return".

* **For** - selecciona una secuencia de nodos
* **Let** - une una secuencia a una variable
* **Where** - filtra los nodos
* **Order by** - ordena los nodos
* **Return** : qué devolver (se evalúa una vez por cada nodo)

for $x in doc("books.xml")/bookstore/book  
where $x/price > 30  
order by $x/title  
return $x/title

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**)

Oracle proporciona una serie de funciones propias para la generación de XML a partir de consultas SQL a una base de datos:

**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 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.

**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.

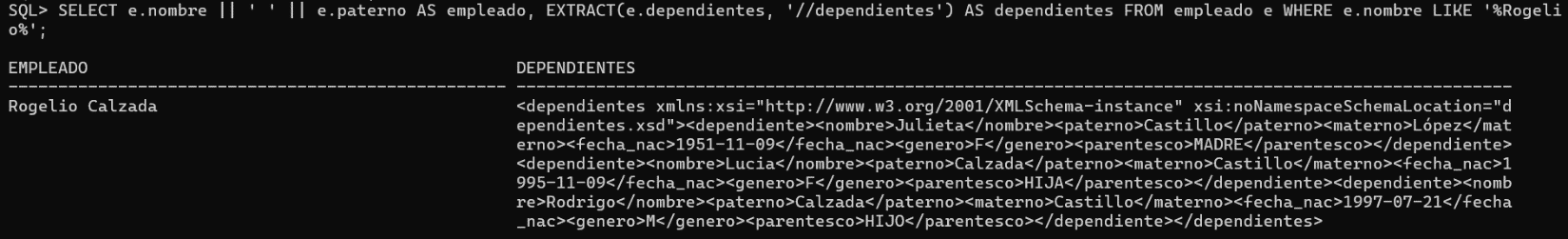
**XMLFOREST** - Crea fragmentos de documentos XML desde datos relacionales. Por defecto, cada nombre de columna se convierte en nombre de un elemento y cada valor de columna en el valor de texto del elemento. Las instancias anidadas de XMLELEMENT y XMLFOREST se utilizan para formar documentos XML bien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

**XMLAGG** - Crea un agregado de fragmentos de un documento XML desde una colección de objetos XMLType separados.

DESARROLLO DE LA PRÁCTICA:

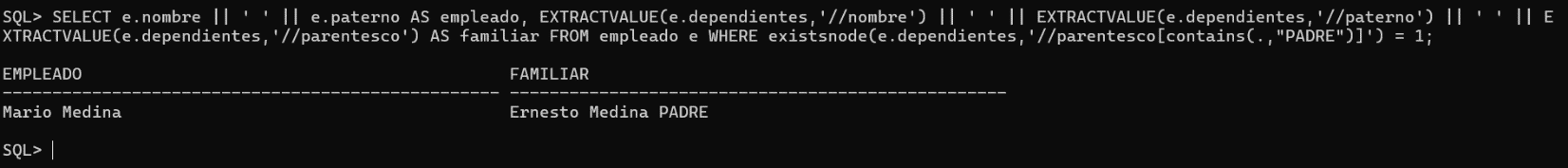
La ejecución de las sentencias de esta práctica depende de los objetos creados en la práctica anterior (número 4), por lo que es necesario verificar que se tengan las tablas y archivos XML necesarios.

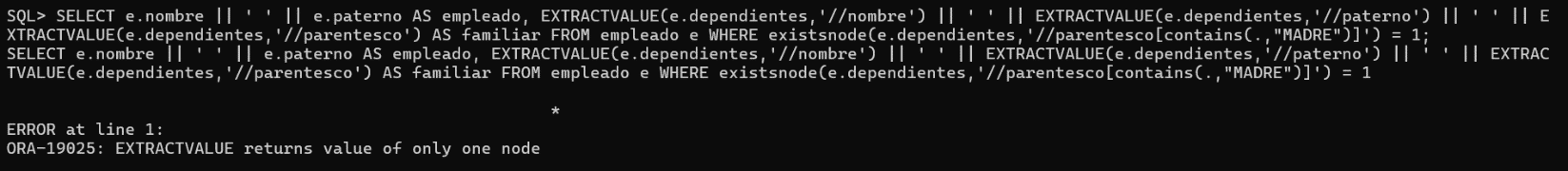
a) Abra una ventana de la Línea de Comandos SQL de Oracle, y escriba las siguientes sentencias. Incluya una captura del resultado obtenido de aquellas instrucciones resaltadas en color amarillo:

1. **CONNECT** REL\_XML/REL\_XML;
2. **ALTER** **SESSION** **SET** NLS\_TERRITORY='MEXICO';
3. **SET** PAGESIZE 500;
4. **SET** LINESIZE 300;
5. **SET** **LONG** 2000;
6. **COLUMN** empleado FORMAT A50;
7. **COLUMN** familiar FORMAT A50;
8. **COLUMN** dependientes FORMAT A100;
9. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACT**(e.dependientes, '//dependientes') **AS** dependientes **FROM** empleado e **WHERE** e.nombre **LIKE** '%Rogelio%';
10. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACT**(e.dependientes,'//nombre') || **EXTRACT**(e.dependientes,'//paterno') **AS** familiar, **EXTRACT**(e.dependientes,'//parentesco') **AS** parentesco **FROM** empleado e;

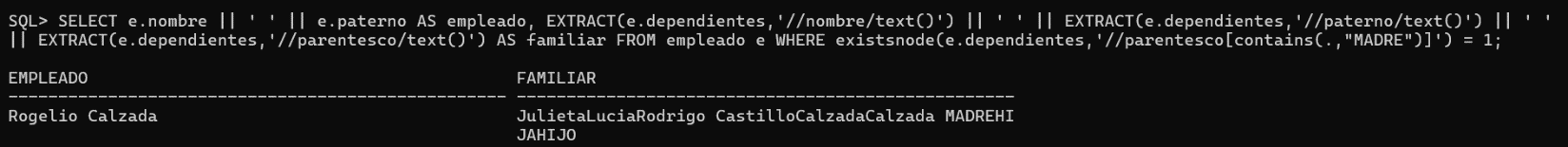
Texto

Descripción generada automáticamente

1. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACTVALUE**(e.dependientes,'//nombre') || ' ' || **EXTRACTVALUE**(e.dependientes,'//paterno') || ' ' || **EXTRACTVALUE**(e.dependientes,'//parentesco') **AS** familiar **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"PADRE")]') = 1;
2. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACTVALUE**(e.dependientes,'//nombre') || ' ' || **EXTRACTVALUE**(e.dependientes,'//paterno') || ' ' || **EXTRACTVALUE**(e.dependientes,'//parentesco') **AS** familiar **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")]') = 1;



1. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACT**(e.dependientes,'//nombre/text()') || ' ' || **EXTRACT**(e.dependientes,'//paterno/text()') || ' ' || **EXTRACT**(e.dependientes,'//parentesco/text()') **AS** familiar **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")]') = 1;

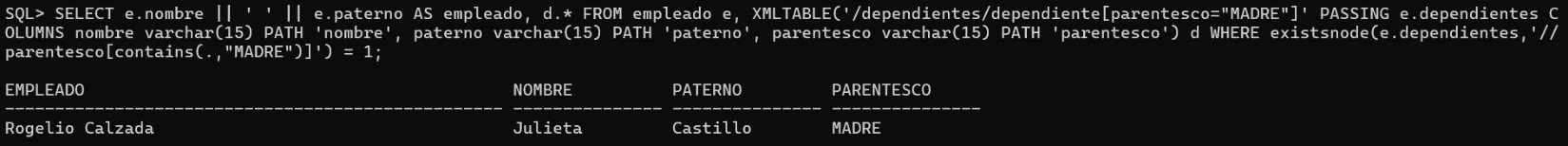
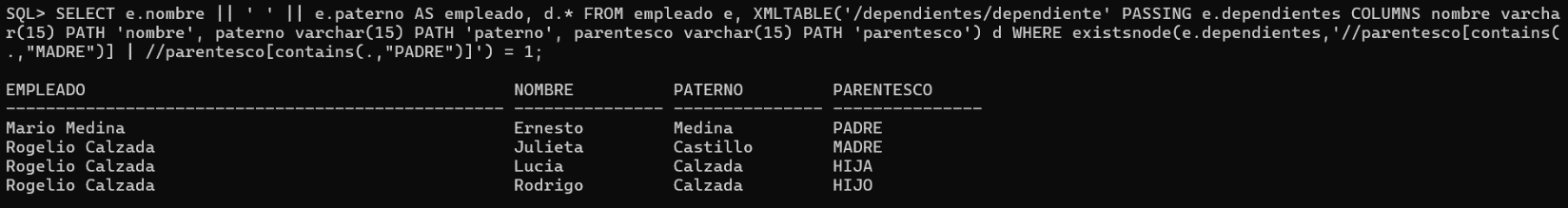


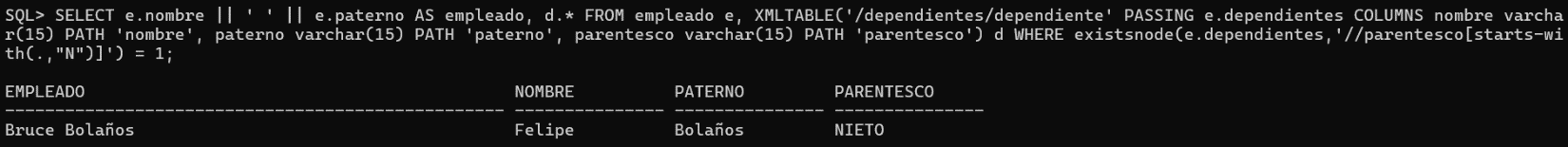
1. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, **EXTRACT**(e.dependientes,'//nombre') || **EXTRACT**(e.dependientes,'//paterno') || **EXTRACT**(e.dependientes,'//parentesco') **AS** familiar **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")]') = 1;

Texto

Descripción generada automáticamente

1. **Captura de pantalla de un celular

   Descripción generada automáticamenteSELECT** e.nombre || ' ' || e.paterno **AS** empleado, d.\* **FROM** empleado e, **XMLTABLE**('/dependientes/dependiente' PASSING e.dependientes COLUMNS nombre **varchar**(15) **PATH** 'nombre', paterno **varchar**(15) **PATH** 'paterno', parentesco **varchar**(15) **PATH** 'parentesco') d **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")]') = 1;
2. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, d.\* **FROM** empleado e, **XMLTABLE**('/dependientes/dependiente[parentesco="MADRE"]' PASSING e.dependientes COLUMNS nombre **varchar**(15) **PATH** 'nombre', paterno **varchar**(15) **PATH** 'paterno', parentesco **varchar**(15) **PATH** 'parentesco') d **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")]') = 1;
3. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, d.\* **FROM** empleado e, **XMLTABLE**('/dependientes/dependiente' PASSING e.dependientes COLUMNS nombre **varchar**(15) **PATH** 'nombre', paterno **varchar**(15) **PATH** 'paterno', parentesco **varchar**(15) **PATH** 'parentesco') d **WHERE** **existsnode**(e.dependientes,'//parentesco[contains(.,"MADRE")] | //parentesco[contains(.,"PADRE")]') = 1;
4. **Texto

   Descripción generada automáticamenteSELECT** e.nombre || ' ' || e.paterno **AS** empleado, d.\* **FROM** empleado e, **XMLTABLE**('/dependientes/dependiente' PASSING e.dependientes COLUMNS nombre **varchar**(15) **PATH** 'nombre', paterno **varchar**(15) **PATH** 'paterno', parentesco **varchar**(15) **PATH** 'parentesco') d **WHERE** **existsnode**(e.dependientes,'//parentesco[substring(.,1,3)="HIJ"]') = 1;
5. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado, d.\* **FROM** empleado e, **XMLTABLE**('/dependientes/dependiente' PASSING e.dependientes COLUMNS nombre **varchar**(15) **PATH** 'nombre', paterno **varchar**(15) **PATH** 'paterno', parentesco **varchar**(15) **PATH** 'parentesco') d **WHERE** **existsnode**(e.dependientes,'//parentesco[starts-with(.,"N")]') = 1;
6. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'/dependientes[count(.) = 1]') = 1;

Texto

Descripción generada automáticamente

1. **SELECT** e.nombre || ' ' || e.paterno **AS** empleado **FROM** empleado e **WHERE** **existsnode**(e.dependientes,'/dependientes[count(dependiente) > 1]') = 1;

Texto

Descripción generada automáticamente con confianza media

1. **UPDATE** empleado e **SET** e.dependientes = xmltype('<dependientes><dependiente><nombre>Juan</nombre><paterno>Lara</paterno><materno>Regalado</materno><fecha\_nac>2000-01-01</fecha\_nac><parentesco>HIJO</parentesco><genero>M</genero></dependiente></dependientes>') **WHERE** e.nss = 444555666;
2. **SELECT** e.dependientes **FROM** empleado e **WHERE** e.nss = 444555666;

Texto

Descripción generada automáticamente

1. **UPDATE** empleado e **SET** e.dependientes = **updateXML**(e.dependientes, '/dependientes/dependiente[1]/nombre', '<nombre>José</nombre>') **WHERE** e.nss = 444555666;
2. **SELECT** e.dependientes **FROM** empleado e **WHERE** e.nss = 444555666;

**Texto

Descripción generada automáticamente**

1. **UPDATE** empleado e **SET** e.dependientes = **updateXML**(e.dependientes, '/dependientes/dependiente[1]/nombre', '<nombre>Juan</nombre>','/dependientes/dependiente[1]/fecha\_nac', '<fecha\_nac>2000-05-24</fecha\_nac>') **WHERE** e.nss = 444555666;
2. **SELECT** e.dependientes **FROM** empleado e **WHERE** e.nss = 444555666;

Texto

Descripción generada automáticamente

1. **UPDATE** empleado e **SET** e.dependientes = **deleteXML**(e.dependientes, '/dependientes/dependiente[1]') **WHERE** e.nss = 444555666;
2. **SELECT** e.dependientes **FROM** empleado e **WHERE** e.nss = 444555666;

Texto

Descripción generada automáticamente

1. **SELECT** \* **FROM** oficinas;

Texto

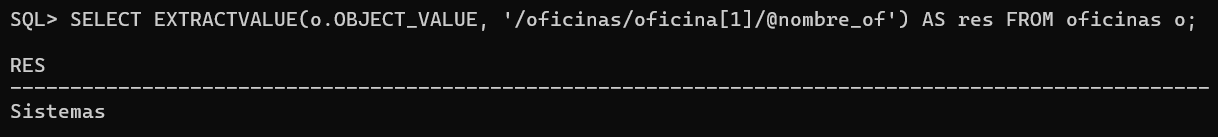
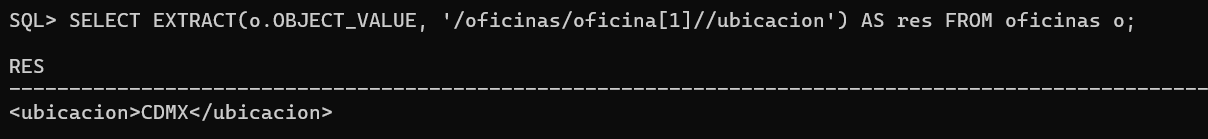
Descripción generada automáticamente

1. **COLUMN** res FORMAT A100;
2. **SELECT** **XMLROOT**(o.OBJECT\_VALUE, VERSION '1.0') **FROM** oficinas o;

Texto

Descripción generada automáticamente

1. Texto

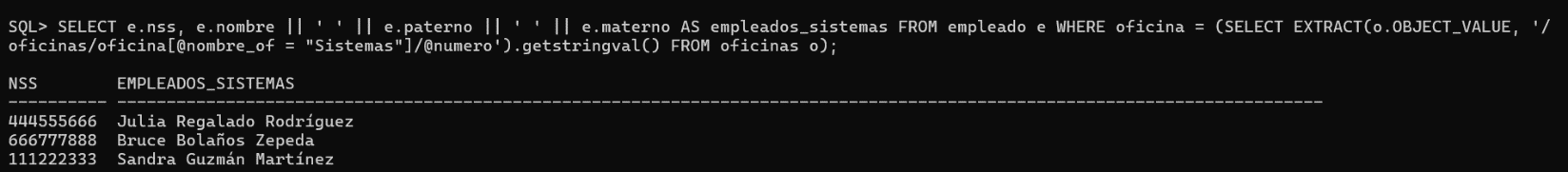
   Descripción generada automáticamente**SELECT** **EXTRACT**(o.OBJECT\_VALUE, '/oficinas/oficina[1]') **AS** res **FROM** oficinas o;
2. **SELECT** **EXTRACTVALUE**(o.OBJECT\_VALUE, '/oficinas/oficina[1]/@nombre\_of') **AS** res **FROM** oficinas o;
3. **SELECT** **EXTRACT**(o.OBJECT\_VALUE, '/oficinas/oficina[1]//ubicacion') **AS** res **FROM** oficinas o;
4. Texto

   Descripción generada automáticamente**SELECT** **EXTRACT**(o.OBJECT\_VALUE, '/oficinas/oficina/ubicaciones[ubicacion = "CDMX" or ubicacion = "Monterrey"]') **AS** res **FROM** oficinas o;
5. **SELECT** **XMLSERIALIZE**(DOCUMENT o.OBJECT\_VALUE **AS** **CLOB**) **AS** res **FROM** oficinas o;

Texto

Descripción generada automáticamente

1. **SELECT** e.nss, e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** empleados\_sistemas **FROM** empleado e **WHERE** oficina = (**SELECT** **EXTRACT**(o.OBJECT\_VALUE, '/oficinas/oficina[@nombre\_of = "Sistemas"]/@numero').getstringval() **FROM** oficinas o);

****

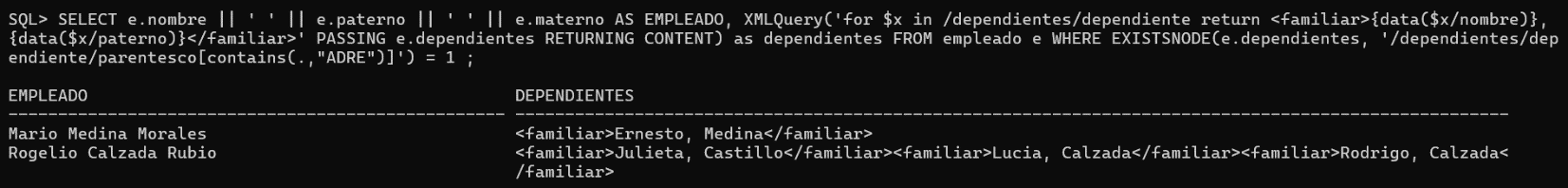
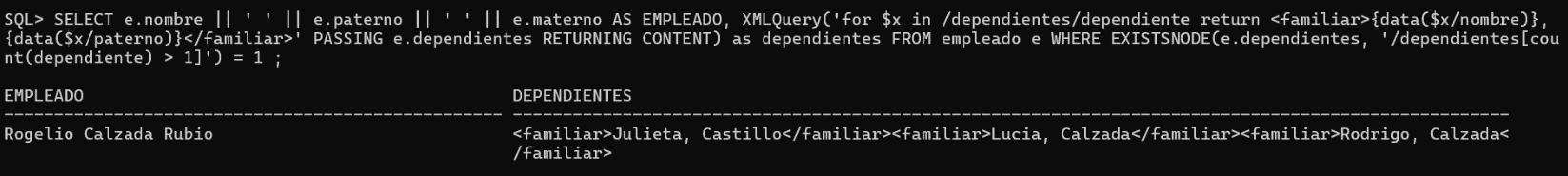
1. **Imagen que contiene Interfaz de usuario gráfica

   Descripción generada automáticamenteSELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in /dependientes/dependiente/nombre return <familiar>{$x}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e;
2. **SELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in /dependientes/dependiente/nombre return <familiar>{$x}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e **WHERE** **EXISTSNODE**(e.dependientes, '/dependientes/dependiente[parentesco="HIJO"] | /dependientes/dependiente[parentesco="HIJA"]') = 1 ;

Imagen que contiene Texto

Descripción generada automáticamente

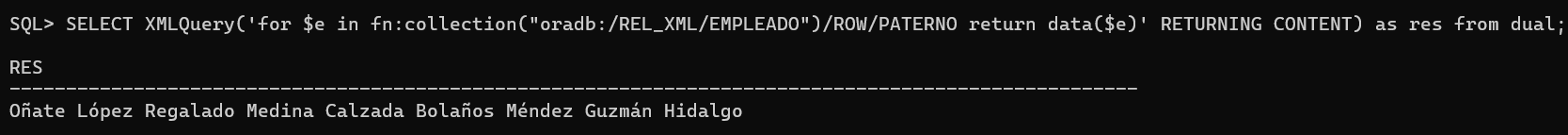
1. **Texto

   Descripción generada automáticamenteSELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in /dependientes/dependiente return <familiar>{$x/nombre}{$x/paterno}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e **WHERE** **EXISTSNODE**(e.dependientes, '/dependientes/dependiente/parentesco[contains(.,"ADRE")]') = 1 ;
2. **SELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in /dependientes/dependiente return <familiar>{data($x/nombre)}, {data($x/paterno)}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e **WHERE** **EXISTSNODE**(e.dependientes, '/dependientes/dependiente/parentesco[contains(.,"ADRE")]') = 1 ;
3. **SELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in /dependientes/dependiente return <familiar>{data($x/nombre)}, {data($x/paterno)}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e **WHERE** **EXISTSNODE**(e.dependientes, '/dependientes[count(dependiente) > 1]') = 1 ;
4. **SELECT** e.nombre || ' ' || e.paterno || ' ' || e.materno **AS** EMPLEADO, **XMLQuery**('for $x in //dependiente return <familiar>{data($x)}</familiar>' PASSING e.dependientes RETURNING CONTENT) as dependientes **FROM** empleado e **WHERE** **EXISTSNODE**(e.dependientes, '//dependiente[genero = "F"]') = 1 ;

Texto

Descripción generada automáticamente

1. Texto

   Descripción generada automáticamente**SELECT** **XMLQuery**('for $e in fn:collection("oradb:/REL\_XML/EMPLEADO") return $e' RETURNING CONTENT) **as** res **from** dual;
2. **SELECT** **XMLQuery**('for $e in fn:collection("oradb:/REL\_XML/EMPLEADO")/ROW/PATERNO return data($e)' RETURNING CONTENT) **as** res **from** dual;
3. **SELECT** **XMLQuery**('for $e in fn:collection("oradb:/REL\_XML/EMPLEADO")/ROW[PATERNO="Medina"] return $e' RETURNING CONTENT) **as** res **from** dual;

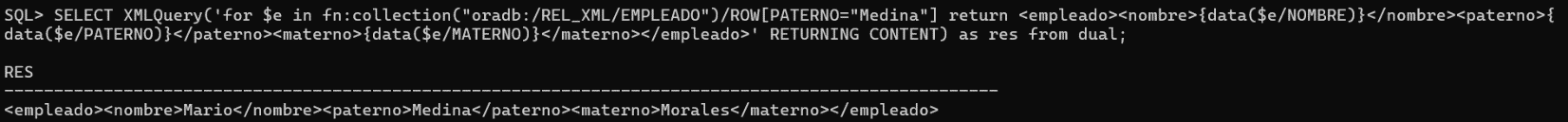
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('for $e in fn:collection("oradb:/REL\_XML/EMPLEADO") where $e/ROW/PATERNO="Medina" return $e' RETURNING CONTENT) **as** res **from** dual;

Texto

Descripción generada automáticamente con confianza baja

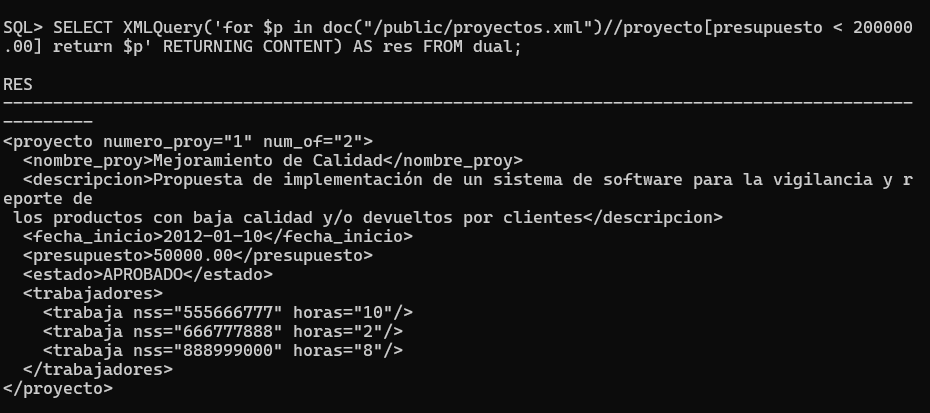
1. **SELECT** **XMLQuery**('for $e in fn:collection("oradb:/REL\_XML/EMPLEADO")/ROW[PATERNO="Medina"] return <empleado><nombre>{data($e/NOMBRE)}</nombre><paterno>{data($e/PATERNO)}</paterno><materno>{data($e/MATERNO)}</materno></empleado>' RETURNING CONTENT) **as** res **from** dual;
2. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto return $p' RETURNING CONTENT) **AS** res **FROM** dual;

Texto

Descripción generada automáticamente

1. Texto

   Descripción generada automáticamente**SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto[@num\_of="1"] return $p' RETURNING CONTENT) **AS** res **FROM** dual;
2. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto[presupuesto < 200000.00] return $p' RETURNING CONTENT) **AS** res **FROM** dual;

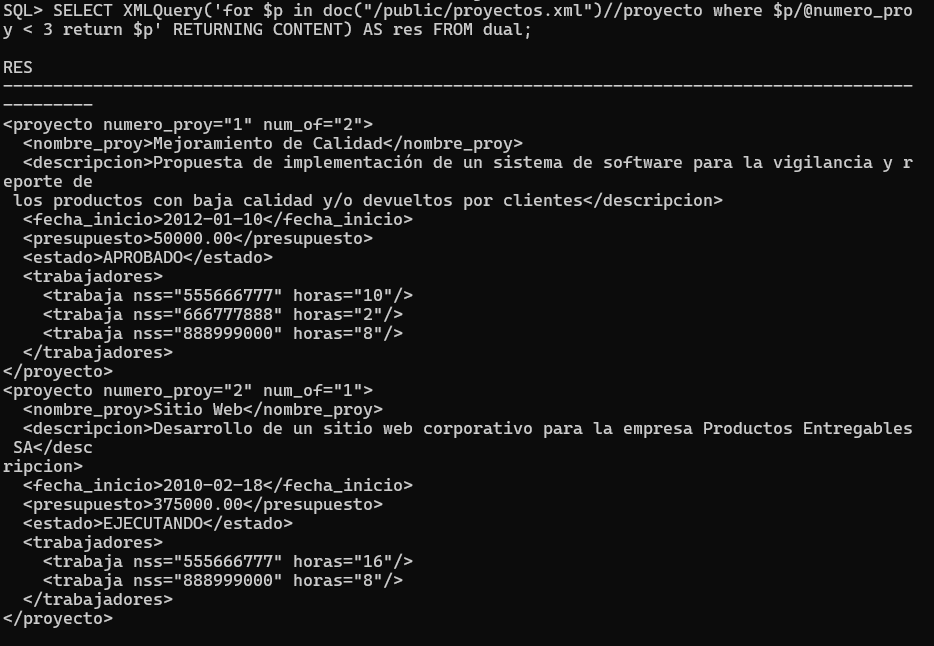


1. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto/trabajadores/trabaja[@nss="222333444"] return $p/@horas' RETURNING CONTENT) **AS** res **FROM** dual;

Escala de tiempo

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto where $p/@numero\_proy < 3 return $p' RETURNING CONTENT) **AS** res **FROM** dual;



1. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto where $p/presupuesto > 100000.00 and $p/presupuesto < 300000.00 return $p' RETURNING CONTENT) **AS** res **FROM** dual;

Texto

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto where $p/descripcion[contains(.,"CDMX")] return $p' RETURNING CONTENT) **AS** res **FROM** dual;

Texto

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('for $p in doc("/public/proyectos.xml")//proyecto[presupuesto < 200000.00] let $o := fn:collection("oradb:/REL\_XML/OFICINAS")//oficina[@numero=$p/@num\_of] return <resultado>{$p/nombre\_proy}{$p/descripcion}{$o/ubicaciones}</resultado>' RETURNING CONTENT) **AS** res **FROM** dual;

Texto

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('<resultado>{for $p in doc("/public/proyectos.xml")//proyecto[@numero\_proy=1] return <proyecto>{$p/nombre\_proy}<empleados>{for $e in fn:collection("oradb:/REL\_XML/EMPLEADO")/ROW[NSS = fn:concat($p/trabajadores/trabaja/@nss, " ")] return $e}</empleados></proyecto>}</resultado>' RETURNING CONTENT) **as** res **from** dual;

Texto

Descripción generada automáticamente

1. **SELECT** **XMLQuery**('<resultado>{for $p in doc("/public/proyectos.xml")//proyecto[@numero\_proy=1] let $o := fn:collection("oradb:/REL\_XML/OFICINAS")//oficina[@numero=$p/@num\_of] return <proyecto>{$p/nombre\_proy}{$p/descripcion}{$p/presupuesto}{$o}<empleados>{for $e in fn:collection("oradb:/REL\_XML/EMPLEADO")/ROW[NSS = fn:concat($p/trabajadores/trabaja/@nss, " ")] return $e}</empleados></proyecto>}</resultado>' RETURNING CONTENT) **as** res **from** dual;

Texto

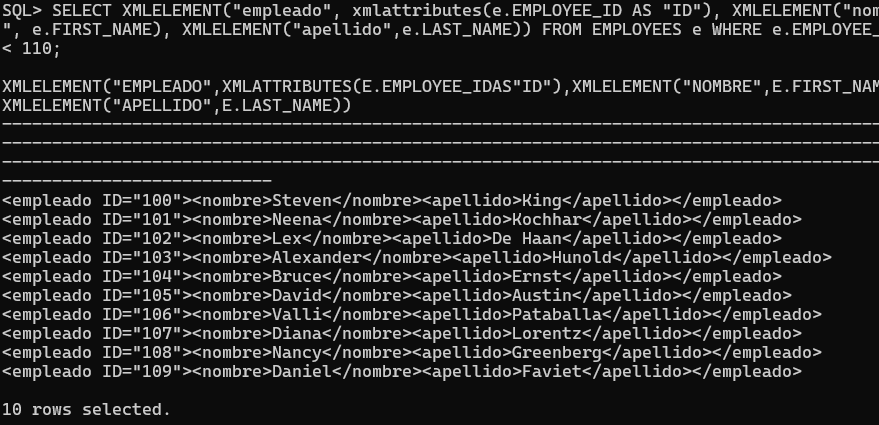
Descripción generada automáticamente

b) Ejecute las siguientes sentencias y muestre el resultado obtenido de las marcadas en color amarillo. Trabaje con el usuario *HR* proporcionado en la instalación por defecto de Oracle*:*

1. **DISCONNECT**;
2. **CONNECT** / **AS** SYSDBA;
3. **ALTER** **USER** HR ACCOUNT UNLOCK;
4. **ALTER** **USER** HR **IDENTIFIED** **BY** hr;
5. **DISCONNECT**;
6. **CONNECT** hr/hr;
7. **SELECT** XMLELEMENT("empleado", XMLELEMENT("nombre", e.FIRST\_NAME), XMLELEMENT("apellido",e.LAST\_NAME)) **FROM** EMPLOYEES e **WHERE** e.EMPLOYEE\_ID < 110;

Texto

Descripción generada automáticamente

1. **SELECT** XMLELEMENT("empleado", xmlattributes(e.EMPLOYEE\_ID **AS** "ID"), XMLELEMENT("nombre", e.FIRST\_NAME), XMLELEMENT("apellido",e.LAST\_NAME)) **FROM** EMPLOYEES e **WHERE** e.EMPLOYEE\_ID < 110;
2. **SELECT** XMLELEMENT("empleado", xmlattributes(e.EMPLOYEE\_ID **AS** "ID"), XMLELEMENT("nombre", e.FIRST\_NAME),XMLELEMENT("apellido",e.LAST\_NAME),XMLELEMENT("personales",XMLELEMENT("correoE",e.email),XMLELEMENT("telefono",e.PHONE\_NUMBER))) **FROM** EMPLOYEES e **WHERE** e.EMPLOYEE\_ID < 110;
3. **SELECT** XMLELEMENT("personales", xmlattributes(e.employee\_id **AS** "ID"), **XMLFOREST**(e.first\_name **as** "nombre", e.last\_name **AS** "apellido", e.EMAIL **AS** "correoE", e.PHONE\_NUMBER **AS** "telefono")) **FROM** employees e **WHERE** e.EMPLOYEE\_ID < 110;

Texto

Descripción generada automáticamente

1. Imagen en blanco y negro

   Descripción generada automáticamente con confianza baja**SELECT** XMLELEMENT("personales", xmlattributes(e.employee\_id **AS** "ID"), **XMLFOREST**(e.first\_name **as** "nombre", e.last\_name **AS** "apellido", e.EMAIL **AS** "correoE", e.PHONE\_NUMBER **AS** "telefono"), XMLELEMENT("trabajo", **XMLFOREST**(d.DEPARTMENT\_ID **AS** "numero", d.DEPARTMENT\_NAME **AS** "departamento", e2.FIRST\_NAME || ' ' || e2.LAST\_NAME **AS** "JEFE"))) **FROM** EMPLOYEES e, DEPARTMENTS d, EMPLOYEES e2 **WHERE** e.DEPARTMENT\_ID = d.DEPARTMENT\_ID **AND** d.MANAGER\_ID = e2.EMPLOYEE\_ID **AND** e.EMPLOYEE\_ID < 110;
2. **SELECT** XMLELEMENT("departamento", XMLATTRIBUTES(e.DEPARTMENT\_ID **AS** "numero\_departamento"), **XMLAGG**(XMLELEMENT("empleado", e.FIRST\_NAME || ' ' || e.LAST\_NAME) **ORDER** **BY** e.LAST\_NAME)) **FROM** EMPLOYEES e **GROUP** **BY** e.DEPARTMENT\_ID;



1. **SELECT** **XMLPARSE**(CONTENT '<EMPLEADOS><empleado><nombre>Steven</nombre>apellido>King</apellido></empleado><empleado><nombre>Neena</nombre><apellido>Kochhar</apellido></empleado><empleado><nombre>Lex</nombre><apellido>De Haan</apellido></empleado><empleado><nombre>Alexander</nombre><apellido>Hunold</apellido></empleado><empleado><nombre>Bruce</nombre><apellido>Ernst</apellido></empleado><empleado><nombre>David</nombre><apellido>Austin</apellido></empleado><empleado><nombre>Valli</nombre><apellido>Pataballa</apellido></empleado><empleado><nombre>Diana</nombre><apellido>Lorentz</apellido></empleado><empleado><nombre>Nancy</nombre><apellido>Greenberg</apellido></empleado><empleado><nombre>Daniel</nombre><apellido>Faviet</apellido></empleado></empleados>') **AS** parsing **FROM** dual;

Texto

Descripción generada automáticamente

1. **SELECT** **XMLPARSE**(CONTENT '<empleados><empleado><nombre>Steven<apellido>King</nombre></apellido></empleado><empleado><nombre>Neena</nombre><apellido>Kochhar</apellido></empleado><empleado><nombre>Lex</nombre><apellido>De Haan</apellido></empleado><empleado><nombre>Alexander</nombre><apellido>Hunold</apellido></empleado><empleado><nombre>Bruce</nombre><apellido>Ernst</apellido></empleado><empleado><nombre>David</nombre><apellido>Austin</apellido></empleado><empleado><nombre>Valli</nombre><apellido>Pataballa</apellido></empleado><empleado><nombre>Diana</nombre><apellido>Lorentz</apellido></empleado><empleado><nombre>Nancy</nombre><apellido>Greenberg</apellido></empleado><empleado><nombre>Daniel</nombre><apellido>Faviet</apellido></empleado></empleados>') **AS** parsing **FROM** dual;

Texto

Descripción generada automáticamente

1. **Texto

   Descripción generada automáticamenteSELECT** **XMLROOT**(XMLTYPE('<empleados><empleado><nombre>Steven</nombre>apellido>King</apellido></empleado><empleado><nombre>Neena</nombre><apellido>Kochhar</apellido></empleado><empleado><nombre>Lex</nombre><apellido>De Haan</apellido></empleado><empleado><nombre>Alexander</nombre><apellido>Hunold</apellido></empleado><empleado><nombre>Bruce</nombre><apellido>Ernst</apellido></empleado><empleado><nombre>David</nombre><apellido>Austin</apellido></empleado><empleado><nombre>Valli</nombre><apellido>Pataballa</apellido></empleado><empleado><nombre>Diana</nombre><apellido>Lorentz</apellido></empleado><empleado><nombre>Nancy</nombre><apellido>Greenberg</apellido></empleado><empleado><nombre>Daniel</nombre><apellido>Faviet</apellido></empleado></empleados>'), VERSION '1.0', STANDALONE YES) **AS** salida **FROM** dual;
2. **SELECT** **XMLSERIALIZE**(DOCUMENT XMLTYPE('<empleados><empleado><nombre>Steven</nombre><apellido>King</apellido></empleado><empleado><nombre>Neena</nombre><apellido>Kochhar</apellido></empleado><empleado><nombre>Lex</nombre><apellido>De Haan</apellido></empleado><empleado><nombre>Alexander</nombre><apellido>Hunold</apellido></empleado><empleado><nombre>Bruce</nombre><apellido>Ernst</apellido></empleado><empleado><nombre>David</nombre><apellido>Austin</apellido></empleado><empleado><nombre>Valli</nombre><apellido>Pataballa</apellido></empleado><empleado><nombre>Diana</nombre><apellido>Lorentz</apellido></empleado><empleado><nombre>Nancy</nombre><apellido>Greenberg</apellido></empleado><empleado><nombre>Daniel</nombre><apellido>Faviet</apellido></empleado></empleados>')) **AS** salida **FROM** dual;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Texto

   Descripción generada automáticamenteSELECT** **XMLROOT**(XMLELEMENT("empleados",XMLELEMENT("empleado", XMLELEMENT("nombre", e.FIRST\_NAME), XMLELEMENT("apellido",e.LAST\_NAME))), VERSION '1.0', STANDALONE YES) **FROM** EMPLOYEES e **WHERE** e.EMPLOYEE\_ID < 110;
2. **SELECT** **SYS\_XMLGEN**(XMLELEMENT("empleados",XMLELEMENT("empleado", XMLELEMENT("nombre", e.FIRST\_NAME), XMLELEMENT("apellido",e.LAST\_NAME)))) **FROM** EMPLOYEES e **WHERE** e.EMPLOYEE\_ID < 110;

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 la creación de documentos XML en la base de datos. Estas opciones incluyen:

SQL/XML: Oracle proporciona soporte integrado para la generación de documentos XML a partir de consultas SQL estándar. Con SQL/XML, los usuarios pueden utilizar funciones XML específicas para generar documentos XML directamente desde las consultas SQL.

PL/SQL: Los desarrolladores pueden utilizar PL/SQL (lenguaje de programación procedural de Oracle) para generar documentos XML de manera programática. Oracle proporciona una serie de paquetes y funciones PL/SQL que facilitan la manipulación y creación de documentos XML dentro de la base de datos.

XML DB: Oracle XML DB es una característica que permite el almacenamiento nativo de documentos XML en la base de datos Oracle. Con XML DB, los usuarios pueden almacenar documentos XML directamente en la base de datos y manipularlos utilizando consultas SQL estándar y lenguaje XQuery.

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

Oracle emplea varias tecnologías para ejecutar consultas dentro de un documento XML almacenado en la base de datos:

-SQL/XML: Oracle permite realizar consultas XPath dentro de documentos XML almacenados en la base de datos utilizando extensiones SQL/XML. Esto permite a los usuarios recuperar y manipular datos dentro de documentos XML utilizando consultas SQL estándar.

- XQuery: Oracle proporciona soporte para XQuery, un lenguaje de consulta diseñado específicamente para trabajar con datos XML. Con XQuery, los usuarios pueden realizar consultas avanzadas dentro de documentos XML almacenados en la base de datos, extrayendo información de manera eficiente y flexible.

-XML DB: Oracle XML DB proporciona capacidades integradas para el almacenamiento y la consulta de documentos XML en la base de datos. Los usuarios pueden utilizar consultas SQL estándar para recuperar y manipular datos dentro de documentos XML almacenados en la base de datos, aprovechando las características de XML DB para un rendimiento optimizado.

CONCLUSIONES

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

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/>

6.- Querying Table or View Data using XQuery <https://docs.oracle.com/cd/E11882_01/appdev.112/e23094/xdb_xquery.htm#ADXDB5129>