Alumno: Hernandez Perez Diego Francisco

Bases de Datos No Relacionales

Práctica no. 1:

Definición y creación de una base de datos orientada a documentos XML

**Número de Práctica: 1**

**Nombre de la Práctica: Definición y creación de 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)

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;

**Validación XML dentro de la base de datos**

Además de saber si un documento XML en particular está bien formado, a menudo se necesita saber si se ajusta a un esquema XML dado, es decir, si es válido con respecto a ese esquema XML. En el proceso de descomposición de datos XML para almacenarlos de forma relacional, Oracle XML DB realiza automáticamente una validación parcial para garantizar que la estructura del documento XML se ajuste a las definiciones de tipos de datos SQL que se derivaron del esquema XML.

Se pueden usar los siguientes mecanismos para realizar una validación completa y manipular el estado de validación registrado de los documentos XML:

* Función ***XMLIsValid***. Devuelve 1si se determina que el documento es válido; 0 si se determina que el documento no es válido o no se puede determinar la validez del documento.
* Método ***SchemaValidate()***: ejecuta el proceso de validación. Establece el estado de validación en 1 si se determina que el documento es válido . (De lo contrario, el estado permanece en 0).
* Método ***isSchemaValidated()*** devuelve el estado de validación registrado de una instancia XMLType.
* Método ***setSchemaValidated()*** establece (registra) el estado de validación de una instancia XMLType.

Para forzar la validación completa, se puede usar cualquiera de los siguientes mecanismos:

* Restricción CHECK de nivel de tabla
* Disparador PL/SQL BEFORE INSERT

Ambos enfoques garantizan que solo se puedan almacenar en la tabla de tipo XMLType documentos XML válidos.

La ventaja de una restricción TABLE CHECK es que es fácil de codificar. La desventaja es que se basa en la función Oracle SQL XML ***isValid***, por lo que solo puede indicar si el documento XML es válido o no. Si un documento XML no es válido, no se puede proporcionar ninguna información sobre por qué no es válido.

Un disparador BEFORE INSERT requiere un poco más de código. Valida el documento XML invocando el método ***schemaValidate()***. La ventaja de usar ***schemaValidate()*** es que la excepción generada proporciona información adicional sobre el problema del documento de instancia. También permite intentar una acción correctiva cuando se encuentra un documento no válido.

ALTER TABLE purchaseorder ADD CONSTRAINT validate\_purchaseorder **CHECK (XMLIsValid(OBJECT\_VALUE) = 1);**

CREATE OR REPLACE TRIGGER validate\_purchaseorder

BEFORE INSERT ON purchaseorder

FOR EACH ROW

BEGIN

IF (:new.OBJECT\_VALUE IS NOT NULL) THEN                  :new.**OBJECT\_VALUE.schemavalidate**();

END IF;

END; /

Antes de que Oracle XML DB pueda utilizar un esquema XML, debe registrarse. Luego se puede usar para crear tablas y columnas XMLType y para validar documentos XML. Si el registro del esquema falla, la base de datos se restaura al estado que tenía antes del intento de registro.

Se requieren dos elementos para registrar un esquema XML con Oracle XML DB:

* El documento de esquema XML
* Una cadena que se puede utilizar como un identificador único para el esquema XML, después de que se registre en la base de datos de Oracle

BEGIN

DBMS\_XMLSCHEMA.registerSchema(

SCHEMAURL => 'http://xmlns.oracle.com/xdb/documentation/purchaseOrder.xsd',

SCHEMADOC => bfilename('XMLDIR','purchaseOrder.xsd'),

LOCAL => **TRUE**,

GENTYPES => TRUE,

GENTABLES => FALSE,

CSID => nls\_charset\_id('AL32UTF8'));

END; /

BEGIN DBMS\_XMLSCHEMA.deleteSchema(

SCHEMAURL => 'purchaseOrder.xsd', **DELETE\_OPTION => DBMS\_XMLSCHEMA.DELETE\_CASCADE\_FORCE**);

END; /

SELECT OWNER, LOCAL, SCHEMA\_URL FROM DBA\_XML\_SCHEMAS;

CREATE TABLE purchaseorder\_as\_table **OF XMLType** XMLSCHEMA "purchaseOrder.xsd" ELEMENT "PurchaseOrder";

CREATE TABLE purchaseorder\_as\_column (id NUMBER, xml\_document **XMLType**) XMLTYPE COLUMN xml\_document ELEMENT purchaseOrder.xsd#PurchaseOrder;

DESARROLLO DE LA PRÁCTICA:

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

1. **CONNECT** / **AS** SYSDBA;
2. **CREATE** **USER** REL\_XML **IDENTIFIED** **BY** REL\_XML;
3. **GRANT** **CONNECT**, RESOURCE **TO** REL\_XML;
4. **CREATE** DIRECTORY REPOSITORIO **AS** 'C:\REPOSITORIO';
5. **GRANT** **READ**, **WRITE** **ON** DIRECTORY REPOSITORIO **TO** REL\_XML;
6. **DISCONNECT**;
7. **CONNECT** REL\_XML/REL\_XML;
8. **ALTER** **SESSION** **SET** NLS\_TERRITORY='MEXICO';
9. **SET** PAGESIZE 500;
10. **SET** LINESIZE 300;
11. **SET** **LONG** 2000;
12. **CREATE** **TABLE** EMPLEADO (

nss **CHAR**(10) **NOT** **NULL**,

nombre **VARCHAR2**(30) **NOT** **NULL**,

paterno **VARCHAR2**(45) **NOT** **NULL**,

materno **VARCHAR2**(45) **NOT** **NULL**,

fecha\_nac **DATE** **NOT** **NULL**,

calle **VARCHAR**(40) **NOT** **NULL**,

numero **NUMBER**(4),

colonia **VARCHAR**(40) **NOT** **NULL**,

ciudad **VARCHAR**(30) **NOT** **NULL**,

codpos **CHAR**(5),

genero **CHAR**(1) **NOT** **NULL**,

salario **NUMBER**(7,2) **DEFAULT** 2000.00,

jefe **CHAR**(10),

cargo **VARCHAR**(50) **NOT** **NULL**,

oficina **NUMBER**(5,0),

titulo **VARCHAR2**(20),

fecha\_contratacion **DATE** **NOT** **NULL**,

dependientes XMLTYPE,

**CONSTRAINT** EMP\_GENERO\_CHK **CHECK** (**UPPER**(genero) ='M' **OR** **UPPER**(genero) ='F'),

**CONSTRAINT** EMP\_SAL\_CHK **CHECK** (salario **BETWEEN** 2500.0 **AND** 50000.0),

**CONSTRAINT** EMP\_NSS\_PK **PRIMARY** **KEY** (nss));

1. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** (' 222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado',**TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
2. **SELECT** **extract**(dependientes, '/dependientes/dependiente') **FROM** empleado;

Texto

Descripción generada automáticamente

1. Texto

   Descripción generada automáticamente con confianza media**SELECT** **extractvalue**(dependientes, '/dependientes/dependiente/nombre') **FROM** empleado;
2. **DELETE** **FROM** empleado **WHERE** NSS= ' 222333444';
3. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE(**TO\_CLOB**('<dependientes><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>')));
4. **SELECT** dependientes **FROM** empleado;

Texto

Descripción generada automáticamente

1. **SELECT** **extract**(dependientes, '/dependientes/dependiente') **FROM** empleado;

**Texto

Descripción generada automáticamente**

1. **SELECT** **extractvalue**(dependientes, '/dependientes/dependiente/nombre') **FROM** empleado;

**Texto

Descripción generada automáticamente**

1. **DELETE** **FROM** empleado **WHERE** NSS= '222333444';
2. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),**XMLPARSE**(DOCUMENT '<dependientes><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
3. **SELECT** dependientes **FROM** empleado;

Texto

Descripción generada automáticamente

1. **SELECT** **extract**(dependientes, '/dependientes/dependiente') **FROM** empleado;

Texto

Descripción generada automáticamente

1. **SELECT** **extractvalue**(dependientes, '/dependientes/dependiente/nombre') **FROM** empleado;

**Texto

Descripción generada automáticamente con confianza media**

1. **DELETE** **FROM** empleado **WHERE** NSS= '222333444';

-- guardar el siguiente código XML Schema en la ruta c:\repositorio\dependientes.xsd

1. <?xml version="1.0" encoding="UTF-8" ?><xsd:**schema** xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"><xsd:**element** name="dependientes"><xsd:complexType><xsd:**sequence**><xsd:**element** name="dependiente" maxOccurs="unbounded"><xsd:complexType><xsd:**sequence**><xsd:**element** name="nombre" maxOccurs="1" **type**="xsd:string"/><xsd:**element** name="paterno" maxOccurs="1" **type**="xsd:string"/><xsd:**element** name="materno" maxOccurs="1" **type**="xsd:string"/><xsd:**element** name="fecha\_nac" **type**="xsd:date" maxOccurs="1"/><xsd:**element** name="genero" maxOccurs="1"><xsd:simpleType><xsd:restriction base="xsd:string"><xsd:enumeration value="F"/><xsd:enumeration value="M"/></xsd:restriction></xsd:simpleType></xsd:**element**><xsd:**element** name="parentesco" maxOccurs="1"><xsd:simpleType><xsd:restriction base="xsd:string"><xsd:enumeration value="MADRE"/><xsd:enumeration value="PADRE"/><xsd:enumeration value="HIJA"/><xsd:enumeration value="HIJO"/><xsd:enumeration value="NIETO"/><xsd:enumeration value="NIETA"/></xsd:restriction></xsd:simpleType></xsd:**element**></xsd:**sequence**></xsd:complexType></xsd:**element**></xsd:**sequence**></xsd:complexType></xsd:**element**></xsd:**schema**>
2. **BEGIN**

DBMS\_XMLSCHEMA.registerSchema(SCHEMAURL => 'dependientes.xsd',SCHEMADOC => **BFILENAME**('REPOSITORIO','dependientes.xsd'),**LOCAL**=>**TRUE**,GENTYPES=>**FALSE**,GENTABLES=>**FALSE**);**END**;

/

1. Texto

   Descripción generada automáticamente**SELECT** SCHEMA\_URL, **LOCAL**, **XMLSERIALIZE**(DOCUMENT **SCHEMA** **AS** **VARCHAR**(4000)) **FROM** USER\_XML\_SCHEMAS;
2. **DROP** **TABLE** empleado;
3. **CREATE** **TABLE** empleado (

nss **CHAR**(10) **NOT** **NULL**,

nombre **VARCHAR2**(30) **NOT** **NULL**,

paterno **VARCHAR2**(45) **NOT** **NULL**,

materno **VARCHAR2**(45) **NOT** **NULL**,

fecha\_nac **DATE** **NOT** **NULL**,

calle **VARCHAR**(40) **NOT** **NULL**,

numero **NUMBER**(4),

colonia **VARCHAR**(40) **NOT** **NULL**,

ciudad **VARCHAR**(30) **NOT** **NULL**,

codpos **CHAR**(5),

genero **CHAR**(1) **NOT** **NULL**,

salario **NUMBER**(7,2) **DEFAULT** 2000.00,

jefe **CHAR**(10),

cargo **VARCHAR**(50) **NOT** **NULL**,

oficina **NUMBER**(5,0),

titulo **VARCHAR2**(20),

fecha\_contratacion **DATE** **NOT** **NULL**,

dependientes XMLTYPE,

**CONSTRAINT** EMP\_GENERO\_CHK **CHECK** (**UPPER**(genero) ='M' **OR** **UPPER**(genero) ='F'),

**CONSTRAINT** EMP\_SAL\_CHK **CHECK** (salario **BETWEEN** 2500.0 **AND** 50000.0),

**CONSTRAINT** EMP\_NSS\_PK **PRIMARY** **KEY** (nss),

**CONSTRAINT** EMP\_EMP\_FK **FOREIGN** **KEY** (jefe) **REFERENCES** empleado **(**nss**) DISABLE**)

XMLTYPE **COLUMN** dependientes XMLSCHEMA "dependientes.xsd" **ELEMENT** "dependientes";

1. **CREATE** **OR** **REPLACE** **TRIGGER** valida\_dependientes

**BEFORE** **INSERT** **ON** empleado

**FOR** **EACH** **ROW**

**BEGIN**

**IF**(**:NEW**.dependientes **IS** **NOT** **NULL**) **THEN**

**:NEW**.dependientes.schemavalidate();

**END** **IF**;

**END**;

/

1. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<**DEPENDIENTES**><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
2. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes><dependiente><nombre>Karen<**nombre**><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
3. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero></dependiente></dependientes>'));
4. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero></dependiente></dependientes>'));
5. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('222333444','Guadalupe','Oñate','Martínez', **TO\_DATE**('24/11/1969','dd/mm/yyyy'),'Av.Revolución',348,'Fuentes','San Ignacio','67656','F',25000, '777888999','Oficinista',3,'Licenciado', **TO\_DATE**('07/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Karen</nombre><paterno>Oñate</paterno><materno>Rodríguez</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
6. **SELECT** dependientes **FROM** empleado;

Texto

Descripción generada automáticamente

1. **SELECT** xmlisvalid(dependientes) **FROM** empleado e;

Texto

Descripción generada automáticamente

1. **SELECT** e.dependientes.isSchemaValid('dependientes.xsd','dependientes') **FROM** empleado e;

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

1. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('777888999','Jesús','López','López', **TO\_DATE**('04/04/1973','dd/mm/yyyy'),'Calle Venus', 45, 'Irrigación', 'Ecatepec', '89765','M',50000,**NULL**,'Administrador Técnico',3,'Ingeniero', **TO\_DATE**('05/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Carlos</nombre><paterno>López</paterno><materno>Pérez</materno><fecha\_nac>2005-04-12</fecha\_nac><genero>M</genero><parentesco>HIJO</parentesco></dependiente></dependientes>'));
2. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('444555666','Julia','Regalado','Rodríguez', **TO\_DATE**('30/07/1975','dd/mm/yyyy'),'Calle cielo',76,'Independencia','Tepeji','34256','F',28000, '111222333','Desarrollador',1,'Ingeniero', **TO\_DATE**('28/06/2001','dd/mm/yyyy'),**NULL**);
3. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('555666777','Mario','Medina','Morales', **TO\_DATE**('01/10/1977','dd/mm/yyyy'),'Av.Politécnico', 650, 'Profesiones', 'CDMX', '09765','M',20000, '777888999','Documentador',3,'Licenciado', **TO\_DATE**('30/05/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Ernesto</nombre><paterno>Medina</paterno><materno>Medina</materno><fecha\_nac>1945-10-28</fecha\_nac><genero>M</genero><parentesco>PADRE</parentesco></dependiente></dependientes>'));
4. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('333444555','Rogelio','Calzada','Rubio', **TO\_DATE**('25/03/1965','dd/mm/yyyy'),'Av.Independencia', 123, 'Fuentes', 'San Ignacio', '56565','M',39000,**NULL**,'Gerente general',2,'Licenciado', **TO\_DATE**('05/04/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Julieta</nombre><paterno>Castillo</paterno><materno>López</materno><fecha\_nac>1951-11-09</fecha\_nac><genero>F</genero><parentesco>MADRE</parentesco></dependiente><dependiente><nombre>Lucia</nombre><paterno>Calzada</paterno><materno>Castillo</materno><fecha\_nac>1995-11-09</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente><dependiente><nombre>Rodrigo</nombre><paterno>Calzada</paterno><materno>Castillo</materno><fecha\_nac>1997-07-21</fecha\_nac><genero>M</genero><parentesco>HIJO</parentesco></dependiente></dependientes>'));
5. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('666777888','Bruce','Bolaños','Zepeda', **TO\_DATE**('07/08/1963','dd/mm/yyyy'),'Mar Mediterráneo',56,'LomasLindas','CDMX','09879','M',24000, '111222333','Oficinista',1,'Licenciado', **TO\_DATE**('17/12/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Felipe</nombre><paterno>Bolaños</paterno><materno>Hernández</materno><fecha\_nac>2009-08-26</fecha\_nac><genero>M</genero><parentesco>NIETO</parentesco></dependiente></dependientes>'));
6. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('999000111','Laura','Méndez','López', **TO\_DATE**('20/02/1982','dd/mm/yyyy'),'Av.Montevideo',98, 'Ampliación Hidalgo','Texcoco','45679','F',18000, '333444555','Desarrollador',2,'Ingeniero', **TO\_DATE**('20/12/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Sergio</nombre><paterno>Méndez</paterno><materno>Carrillo</materno><fecha\_nac>1992-12-17</fecha\_nac><genero>M</genero><parentesco>HIJO</parentesco></dependiente></dependientes>'));
7. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('111222333','Sandra','Guzmán','Martínez', **TO\_DATE**('27/11/1970','dd/mm/yyyy'),'Av.Siempreviva', 444, 'San Mateo', 'Ecatepec', '89740','F',45000,**NULL**,'Director de Operaciones',1,'Ingeniero', **TO\_DATE**('20/12/2000','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Gustavo</nombre><paterno>Guzmán</paterno><materno>Ramírez</materno><fecha\_nac>1999-12-07</fecha\_nac><genero>M</genero><parentesco>HIJO</parentesco></dependiente></dependientes>'));
8. **INSERT** **INTO** empleado (nss,nombre,paterno,materno,fecha\_nac,calle,numero,colonia,ciudad,codpos,genero,salario,jefe,cargo,oficina,titulo,fecha\_contratacion,dependientes) **VALUES** ('888999000','Guadalupe','Hidalgo','Pérez', **TO\_DATE**('19/03/1985','dd/mm/yyyy'),'Av.Miguel Hidalgo', 56, 'Independencia', 'Tepeji', '87981','F',27000, '333444555','Desarrollador',2,'Ingeniero', **TO\_DATE**('11/10/2001','dd/mm/yyyy'),XMLTYPE('<dependientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="dependientes.xsd"><dependiente><nombre>Karla</nombre><paterno>Hidalgo</paterno><materno>Serrano</materno><fecha\_nac>2009-08-26</fecha\_nac><genero>F</genero><parentesco>HIJA</parentesco></dependiente></dependientes>'));
9. **SELECT** dependientes **FROM** empleado;

**Texto

Descripción generada automáticamente**

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. **SELECT** xmlisvalid(dependientes) **FROM** empleado e;

Texto

Descripción generada automáticamente

1. **SELECT** e.dependientes.isSchemaValid('dependientes.xsd','dependientes') **FROM** empleado e;

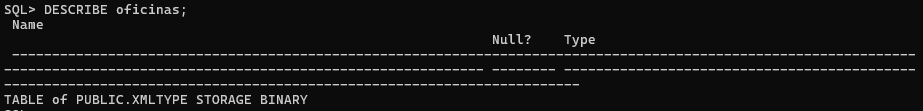
Texto

Descripción generada automáticamente

1. **ALTER** **TABLE** EMPLEADO ENABLE **CONSTRAINT** EMP\_EMP\_FK;

-- guardar el siguiente documento XML en la carpeta c:\repositorio\oficinas.xml

1. <oficinas><oficina numero="1" nombre\_of="Sistemas" jefe\_of="111222333"><ubicaciones><ubicacion>CDMX</ubicacion></ubicaciones></oficina><oficina numero="2" nombre\_of="Ventas" jefe\_of="333444555"><ubicaciones><ubicacion>CDMX</ubicacion><ubicacion>Pachuca</ubicacion></ubicaciones></oficina><oficina numero="3" nombre\_of="Administración" jefe\_of="777888999"><ubicaciones><ubicacion>Monterrey</ubicacion><ubicacion>Guadalupe</ubicacion><ubicacion>Juárez</ubicacion></ubicaciones></oficina></oficinas>
2. **CREATE** **TABLE** oficinas **OF** XMLtype;
3. **DESCRIBE** oficinas;



1. **SELECT** TABLE\_NAME **FROM** USER\_XML\_TABLES;



1. **INSERT** **INTO** oficinas **VALUES** (XMLType(**BFILENAME**('REPOSITORIO', 'oficinas.xml'), **NLS\_CHARSET\_ID**('AL32UTF16')));
2. **SELECT** \* **FROM** oficinas;

Texto

Descripción generada automáticamente

1. **SELECT** **xmlroot**(o.OBJECT\_VALUE, VERSION '1.0', STANDALONE YES) **FROM** oficinas o;

Texto

Descripción generada automáticamente

1. **DECLARE**

proyectos **VARCHAR2**(4000) := '<proyectos>';

res BOOLEAN;

**BEGIN**

proyectos := proyectos || '<proyecto numero\_proy="1" num\_of="2"><nombre\_proy>Mejoramiento de Calidad</nombre\_proy><descripcion>Propuesta de implementación de un sistema de software para la vigilancia y reporte de los productos con baja calidad y/o devueltos por clientes</descripcion><fecha\_inicio>2012-01-10</fecha\_inicio><presupuesto>50000.00</presupuesto><estado>APROBADO</estado><trabajadores><trabaja nss="555666777" horas="10" /><trabaja nss="666777888" horas="2" /><trabaja nss="888999000" horas="8" /></trabajadores></proyecto>';

proyectos := proyectos || '<proyecto numero\_proy="2" num\_of="1"><nombre\_proy>Sitio Web</nombre\_proy><descripcion>Desarrollo de un sitio web corporativo para la empresa Productos Entregables SA</descripcion><fecha\_inicio>2010-02-18</fecha\_inicio><presupuesto>375000.00</presupuesto><estado>EJECUTANDO</estado><trabajadores><trabaja nss="555666777" horas="16" /><trabaja nss="888999000" horas="8" /></trabajadores></proyecto>';

proyectos := proyectos || '<proyecto numero\_proy="3" num\_of="3"><nombre\_proy>Publicidad</nombre\_proy><descripcion>Sistema de control de la ubicación de espectaculares publicitarios en diversas zonas de la CDMX</descripcion><fecha\_inicio>2011-09-06</fecha\_inicio><fecha\_entrega>29/08/2012-08-29</fecha\_entrega><presupuesto>298000.00</presupuesto><estado>ENTREGADO</estado><trabajadores><trabaja nss="222333444" horas="8" /></trabajadores></proyecto>';

proyectos := proyectos || '</proyectos>';

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

**END**;

1. **SELECT** any\_path **FROM** resource\_view **WHERE** under\_path(res, '/public') = 1;

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **SELECT** **PATH** **FROM** path\_view **WHERE** under\_path(res, '/public') = 1;

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja**

1. **SELECT** **xmlquery**('for $x in doc("/public/proyectos.xml") return $x' RETURNING CONTENT) **FROM** dual;

**Texto

Descripción generada automáticamente**

1. **--BEGIN**

--DBMS\_XMLSCHEMA.DELETESCHEMA(SCHEMAURL => 'dependientes.xsd');

**--END**;

--/

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.

XMLse puede utilizar para almacenar datos en un formato semi-estructurado dentro de una base de datos relacional. Esto permite almacenar datos que no se ajustan perfectamente a un modelo relacional típico, como datos complejos o datos con una estructura variable.

XML puede definir restricciones que aseguren que ciertos elementos o atributos estén presentes en un documento XML. Por ejemplo, un documento XML que representa información de clientes puede requerir que el nombre y la dirección sean obligatorios para todos los elementos de cliente, asi como también puede ayudar a definir restricciones que aseguren que los datos almacenados en un documento XML sigan un formato específico. Por ejemplo, un documento XML que almacena fechas puede requerir que todas las fechas estén en un formato específico (por ejemplo, YYYY-MM-DD).

Es importante tener en cuenta que mientras XML ofrece flexibilidad para almacenar datos semi-estructurados en una base de datos relacional, también puede introducir complejidad adicional en la gestión y consulta de datos. Por lo tanto, es crucial diseñar cuidadosamente la estructura XML y considerar cómo se utilizarán los datos en la base de datos relacional.

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.

En un modelo de base de datos relacional:

Las relaciones se establecen utilizando claves primarias y claves foráneas en las tablas. Por ejemplo, en una base de datos de empleados, puede haber una tabla "Empleados" con una clave primaria "ID\_Empleado", y otra tabla "Departamentos" con una clave primaria "ID\_Departamento", y la tabla "Empleados" puede tener una columna "ID\_Departamento" que actúa como una clave foránea que hace referencia a la tabla "Departamentos".

La base de datos relacional garantiza la integridad referencial mediante el uso de restricciones de clave foránea (foreign key constraints), que aseguran que cualquier valor en la columna de clave foránea debe existir en la columna de clave primaria de la tabla relacionada.

Mientras que en un modelado en basae de datos XML:

Las relaciones pueden representarse mediante la inclusión de referencias a otros elementos dentro de un documento XML. Por ejemplo, un documento XML que representa una empresa puede tener elementos "empleado" que contienen información sobre cada empleado, y dentro de cada elemento "empleado", puede haber referencias a elementos "departamento" que contienen información sobre el departamento al que pertenece cada empleado.

CONCLUSIONES

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

la comparación entre el modelado de una base de datos relacional y una basada en XML en términos de conservación de la integridad referencial resalta diferencias significativas en los enfoques utilizados. Mientras que las bases de datos relacionales tienen mecanismos integrados, como restricciones de clave foránea, para garantizar la integridad referencial de manera inherente, las bases de datos XML requieren un enfoque más personalizado y basado en la lógica de la aplicación para mantener la coherencia de los datos.

La práctica de aplicar diversas consultas XML schema puede ser una estrategia efectiva para gestionar y consultar datos almacenados en formato XML. Estas consultas permiten definir la estructura y las restricciones de los documentos XML de manera formal, lo que facilita la validación de los datos y la manipulación de la información de acuerdo con reglas predefinidas. Además, el uso de consultas XML schema puede mejorar la interoperabilidad de los sistemas al estandarizar la representación y el intercambio de datos en formato XML.

La elección entre una base de datos relacional y una basada en XML depende de las necesidades específicas del proyecto y de la naturaleza de los datos a almacenar. Ambas tienen sus propias ventajas y desventajas en términos de modelado y conservación de la integridad referencial, y la selección adecuada debe considerar cuidadosamente los requisitos del sistema y las capacidades de cada enfoque. La práctica de aplicar consultas XML schema puede ser una herramienta valiosa en el contexto de la gestión de datos XML, proporcionando una manera estructurada y estandarizada de definir y manipular la información.

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/>