Bases de Datos Avanzadas

Práctica no. 2:

Manejo de un modelo orientado a objetos en una base de datos relacional (definición de tipos de datos estructurados y asociaciones)

**Número de Práctica: 2**

**Nombre de la Práctica: Manejo de un modelo orientado a objetos en una base de datos relacional (definición de tipos de datos estructurados y asociaciones)**

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://www.oracle.com/database/sqldeveloper/technologies/download/)

MARCO TEÓRICO

**Creación de tipos**

Los objetos están formados por:

* Una especificación o interfaz pública que consta de la declaración de los atributos y, opcionalmente, de la definición de métodos.
* Un cuerpo (parte privada) dónde se implementan los métodos definidos en la especificación. Si un objeto no tiene métodos definidos, no es necesario definir su cuerpo (BODY).

Especificación:

**CREATE** or **REPLACE TYPE** nombreObjeto **AS OBJECT** (

atributo TIPO,

...,

**MEMBER** **FUNCTION** nombreFuncion **RETURN** tipo,

**MEMBER** **FUNCTION** nombreFuncion2(nomVar tipo) **RETURN** tipo,

**MEMBER** **PROCEDURE** nombreProcedimiento

);

Cuerpo:

**CREATE OR REPLACE** **TYPE BODY** nombreObjeto **AS**

**MEMBER** **FUNCTION** nombreFuncion

**RETURN** Tipo

IS

BEGIN

...

END nombreFuncion;

**MEMBER** **FUNCTION** nombreFuncion2(nomVar tipo)

**RETURN** Tipo

IS

BEGIN

...

END nombreFuncion2;

**MEMBER PROCEDURE** nombreProcedimiento

IS BEGIN

...

END nombreProcedimiento;

END;

/

Ejemplo de definición de objetos con funciones y cuerpo:

**CREATE OR REPLACE TYPE** persona **AS OBJECT** (

idpersona NUMBER,

dni VARCHAR2(9),

nombre VARCHAR2(15),

apellidos VARCHAR2(30),

fecha\_nac DATE,

**MEMBER FUNCTION** muestraEdad **RETURN** NUMBER

);

Cuerpo:

**CREATE OR REPLACE TYPE BODY** persona **AS**

**MEMBER FUNCTION** muestraEdad

**RETURN** NUMBER

IS

BEGIN

RETURN TO\_CHAR(SYSDATE, 'YYYY')- TO\_CHAR(fecha\_nac, 'YYYY');

END muestraEdad;

END;

/

Para poder trabajar con datos, es imprescindible crear una tabla derivada de un tipo o una tabla con columnas basada en tipos. Sobre un tipo no se pueden realizar acciones de insert, update, etc. Las tablas se utilizan como si de una tabla normal se tratara, dónde cada una de sus filas es una fila-objeto.

**CREATE TABLE** cliente **OF** persona\_obj(

idpersona **PRIMARY KEY**

);

Inserción de registros

**INSERT INTO** clientes **VALUES** (1, '123456789', 'Juan', 'López', TO\_DATE('01-01-2000','dd-mm-YYYY');

Creación de tablas sin identidad de objeto

**CREATE TABLE** clientes(

cliente persona\_obj,

direccion direccion\_obj,

fecha\_alta DATE

);

Inserción de registros

**INSERT INTO** cliente **VALUES** (

persona\_obj(1, '123456789', 'Juan', 'López', TO\_DATE('01-01-2000','dd-mm-YYYY')),

direccion\_obj('av. principal S/N', '08000', 'Naucalpan', 'EDOMEX'), SYSDATE);

Ver tipos creados

**SELECT** \* **FROM** USER\_TYPES;

**Constructores**

Cada tipo tiene un constructor que es una función especial definida automáticamente por el sistema con el mismo nombre que el tipo. El constructor se utiliza para inicializar y devolver una instancia de ese tipo. Oracle genera un constructor por omisión para cada tipo. Los parámetros del constructor deben coincidir con los atributos del tipo, esto es, los parámetros y los atributos se declaran en el mismo orden y tienen el mismo nombre y tipo. PL/SQL nunca invoca al constructor implícitamente, por lo que el usuario debe invocarlo explícitamente

**Herencia**

Para permitir la herencia entre tipos se debe especificar la opción NOT FINAL en la creación del tipo para que otros tipos puedan heredar de él (valor por defecto). Si se desea que un tipo no pueda heredar, se debe especificar la opción FINAL. Lo mismo se puede hacer para los procedimientos o funciones miembro.

**CREATE TYPE** nombreSuperObj **AS OBJECT**(

nombreAtr tipo,

... ,

**FINAL MEMBER FUNCTION** nombreFuncion1 **RETURN** NUMBER,

**MEMBER FUNCTION** nombreFuncion2 **RETURN** VARCHAR2

) NOT FINAL;

Para especificar que un objeto heredará de un objeto padre, se debe especificar la cláusula UNDER. Si además se quiere sobrescribir un método se debe añadir la cláusula OVERRIDING antes de MEMBER (tanto en la especificación como en el cuerpo del objeto):

**CREATE TYPE** nombreSubObj **UNDER** nombreSuperObj(

nombreAtr TIPO,

... ,

**OVERRIDING MEMBER FUNCTION** nombreFuncion2 **RETURN** VARCHAR2

) FINAL;

Ejemplo

**CREATE TYPE** empleado\_obj **AS OBJECT**(

id\_empl NUMBER,

sueldo\_base NUMBER,

comision NUMBER,

nombre VARCHAR2(10),

apellidos VARCHAR2(25),

**MEMBER FUNCTION** calc\_sueldo\_final **RETURN** NUMBER

) NOT FINAL;

/

**CREATE OR REPLACE TYPE BODY** empleado\_obj **AS**

**MEMBER FUNCTION** calc\_sueldo\_final

**RETURN** NUMBER

IS

BEGIN

**RETURN** sueldo\_base+sueldo\_base\*comision;

END calc\_sueldo\_final;

END;

/

**CREATE TYPE** ingeniero\_obj **UNDER** empleado\_obj(

num\_proyectos NUMBER,

fecha\_titulacion DATE

) **FINAL**;

/

**CREATE TYPE** arquitecto\_obj **UNDER** empleado\_obj(

extras NUMBER,

obras\_realizadas NUMBER,

**OVERRIDING MEMBER FUNCTION** calc\_sueldo\_final **RETURN** NUMBER

) FINAL;

/

**CREATE OR REPLACE TYPE BODY** arquitecto\_obj **AS**

**OVERRIDING MEMBER FUNCTION** calc\_sueldo\_final

RETURN NUMBER

IS

BEGIN

**RETURN** sueldo\_base+sueldo\_base\*comision+extras;

END calc\_sueldo\_final;

END;

/

**CREATE TABLE** empleados **OF** empleado\_obj;

**CREATE TABLE** ingenieros **OF** ingeniero\_obj;

**CREATE TABLE** arquitectos **OF** arquitecto\_obj;

Ejemplo de inserción de registros en tipo heredado

**INSERT INTO** arquitectos (id\_empl, sueldo\_base, comision, nombre, apellidos, extras, obras\_realizadas) **VALUES** (1, 1500, 0.2, 'Juan', 'López', 500, 12);

**INSERT INTO** ingenieros (id\_empl, sueldo\_base, comision, nombre, apellidos, num\_proyectos, fecha\_titulacion) **VALUES** (2, 1500, 0.2, 'Pedro', 'Pérez', 3, TO\_DATE('07-09-2008','dd-mm-YYYY'));

**INSERT INTO** empleados (id\_empl, sueldo\_base, comision, nombre, apellidos) **VALUES** (3, 1500, 0.2, 'María', 'Martínez');

**SELECT** e.nombre, e.sueldo\_base, e.calc\_sueldo\_final() **FROM** empleados e;

**SELECT** i.nombre, i.sueldo\_base, i.calc\_sueldo\_final() **FROM** ingenieros i;

**SELECT** a.nombre, a.sueldo\_base, a.calc\_sueldo\_final() **FROM** arquitectos a;

**Tipos Colección VARRAY**

Son un conjunto de elementos ordenados y del mismo tipo. Se debe definir el número de posiciones que tendrá así como su tipo. No se debe sobrepasar el número de posiciones asignadas en su declaración.

**CREATE TYPE** nombreArray **AS VARRAY**(NºPosiciones) **OF** tipo;

/

Ejemplo:

**CREATE TYPE** lista\_compra **AS VARRAY**(20) **OF** VARCHAR2(10);

/

**CREATE OR REPLACE TYPE** compra\_obj **AS OBJECT**(

idcompra NUMBER,

fecha DATE,

lista lista\_compra

);

/

**CREATE TABLE** compra **OF** compra\_obj(

idcompra **PRIMARY KEY**,

fecha default SYSDATE

);

Actualización (UPDATE) de VARRAY

Las actualizaciones en los VARRAY deben ser atómicas, es decir, se puede modificar el VARRAY totalmente pero no parcialmente.

Ejemplo:

**UPDATE** compra **SET** lista = lista\_compra('item1', item2') **WHERE** idcompra = 42;

**Tipos Colección NESTED TABLES (Tablas anidadas)**

**CREATE OR REPLACE TYPE** nombreTablaObjeto **AS OBJECT**(

atributo1 tipo1,

atributo2 tipo2(n),

...

);

/

**CREATE TYPE** nombreColeccionObjeto **AS TABLE OF** nombreTablaObjeto;

/

**CREATE TABLE** nombreTabla(

atributo1 TIPO1

atributo2 TIPO2,

tablaAnidada nombreColeccionObjeto)

**NESTED TABLE** tablaAnidada **STORE AS** nombre\_tabla;

Ejemplo:

**CREATE OR REPLACE TYPE** persona\_obj **AS OBJECT**(

nombre VARCHAR2(10),

apellidos VARCHAR2(20),

fnacimiento DATE

);

/

**CREATE TYPE** personaNT **AS TABLE OF** persona\_obj;

/

**CREATE TABLE** empleados(

idempleado NUMBER PRIMARY KEY,

fecha\_alta DATE DEFAULT SYSDATE,

persona personaNT

) **NESTED TABLE** persona **STORE AS** personas\_tab;

Inserción:

**INSERT INTO** empleados **VALUES** (1, SYSDATE, personaNT(

persona\_obj('Nombre1', 'Apellidos1', 'fecha1'),

persona\_obj('Nombre2', 'Apellidos2', 'fecha2'),

persona\_obj('Nombre3', 'Apellidos3', 'fecha3'))

);

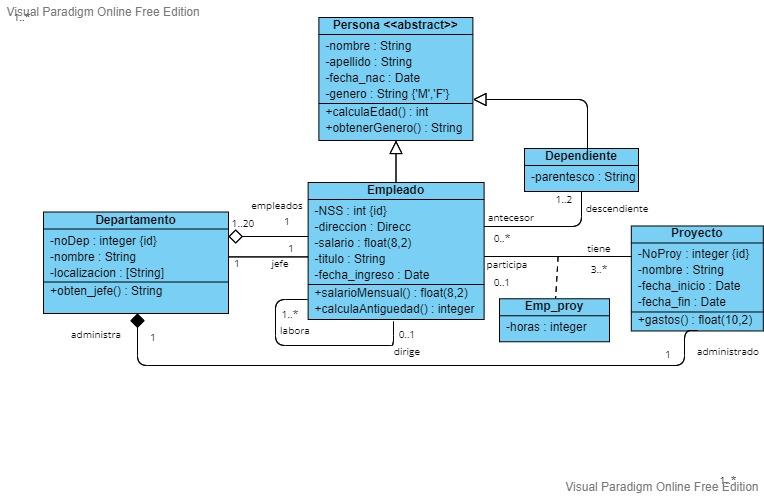
Consulta:

Para trabajar con las tablas anidadas, se debe informar que el atributo persona es, en realidad, una tabla. De esta forma se puede trabajar con su contenido.

**SELECT** e.fecha\_alta, p.nombre **FROM** empleados e, **TABLE**(e.persona) p;

DESARROLLO DE LA PRÁCTICA:

A partir del siguiente diagrama, se trata de trasladar al modelo objeto-relacional en Oracle. Se considera que:

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

1. connect / as sysdba;
2. create user obj identified by obj;
3. grant connect, resource to obj;
4. grant create session to obj;
5. disconnect;
6. connect obj/obj;
7. set pagesize 200;
8. set linesize 200;
9. ALTER SESSION SET NLS\_TERRITORY= 'MEXICO';
10. CREATE OR REPLACE TYPE tPersona AS OBJECT

(nombre VARCHAR(30),

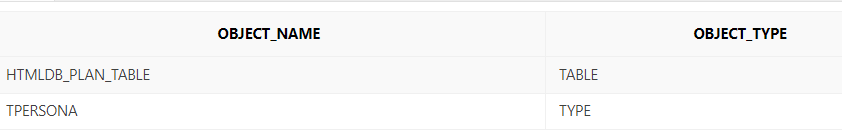
apellido VARCHAR(45),

fecha\_nac DATE,

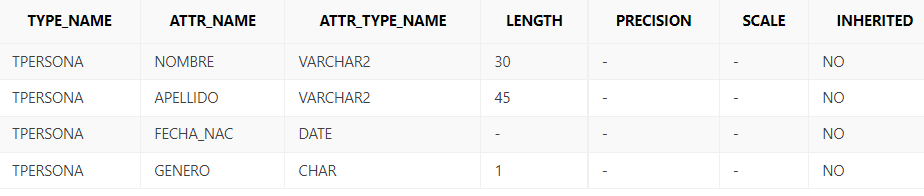
genero CHAR(1)

)NOT FINAL NOT INSTANTIABLE;

1. SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS;



1. SELECT TYPE\_NAME, ATTR\_NAME, ATTR\_TYPE\_NAME, LENGTH, PRECISION, SCALE, INHERITED FROM USER\_TYPE\_ATTRS;



1. CREATE OR REPLACE TYPE tEmpleado UNDER tPersona

(NSS VARCHAR(10),

salario NUMBER(7,2),

titulo VARCHAR(20)

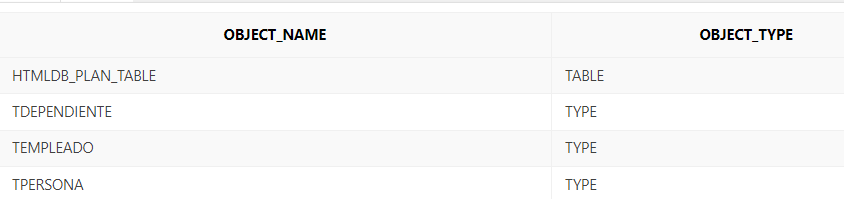
);

1. CREATE OR REPLACE TYPE tDependiente UNDER tPersona

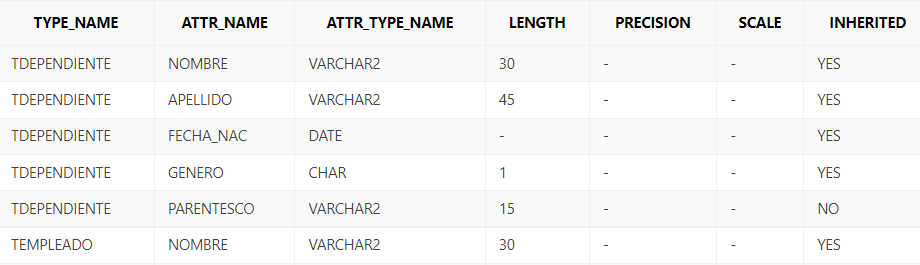
(parentesco VARCHAR(15)

);

1. SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS;



1. SELECT TYPE\_NAME, ATTR\_NAME, ATTR\_TYPE\_NAME, LENGTH, PRECISION, SCALE, INHERITED FROM USER\_TYPE\_ATTRS;



1. DROP TYPE tPersona FORCE;
2. DROP TYPE tEmpleado FORCE;
3. DROP TYPE tDependiente FORCE;
4. CREATE OR REPLACE TYPE tPersona FORCE AS OBJECT

(nombre VARCHAR(30),

apellido VARCHAR(45),

fecha\_nac DATE,

genero CHAR(1),

CONSTRUCTOR FUNCTION tPersona(SELF IN OUT NOCOPY tPersona,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR) RETURN SELF AS RESULT

)NOT FINAL NOT INSTANTIABLE;

1. CREATE OR REPLACE TYPE BODY tPersona AS

CONSTRUCTOR FUNCTION tPersona (SELF IN OUT NOCOPY tPersona,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR) RETURN SELF AS RESULT

IS

BEGIN

SELF.nombre := nombre;

SELF.apellido := apellido;

SELF.fecha\_nac := TO\_DATE(REPLACE(fecha\_nac,'-','/'),'dd/mm/YYYY');

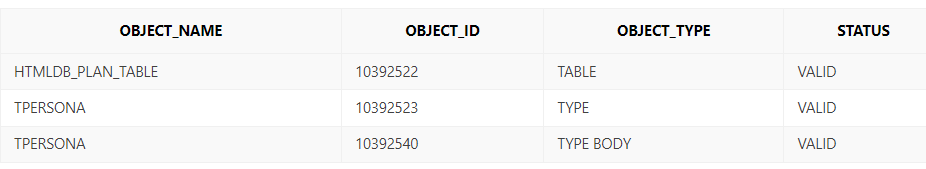
SELF.genero := UPPER(genero);

END;

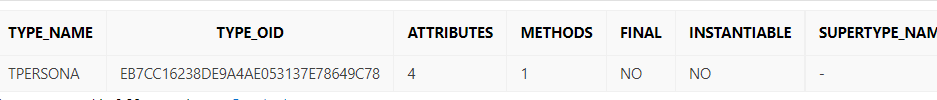
END;

/

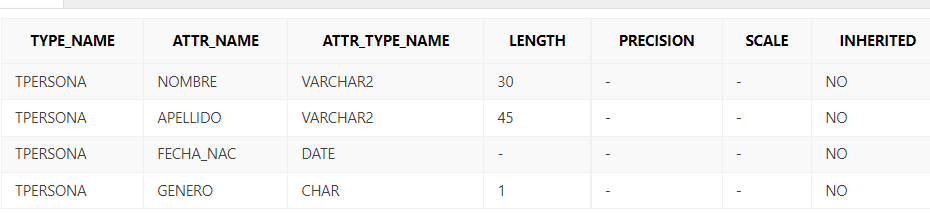
1. SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS;



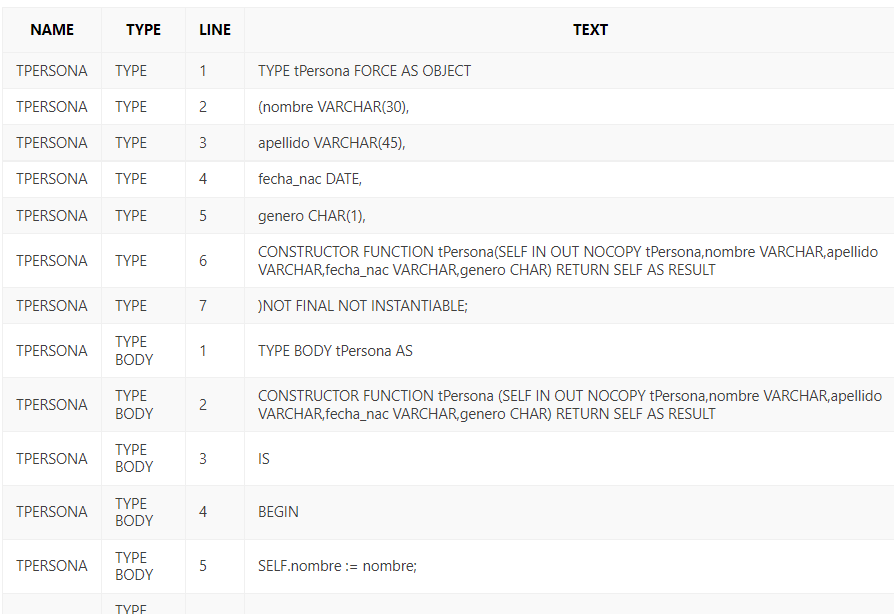
1. SELECT TYPE\_NAME, TYPE\_OID, ATTRIBUTES, METHODS, FINAL, INSTANTIABLE, SUPERTYPE\_NAME FROM USER\_TYPES;



1. SELECT TYPE\_NAME, ATTR\_NAME, ATTR\_TYPE\_NAME, LENGTH, PRECISION, SCALE, INHERITED FROM USER\_TYPE\_ATTRS;



1. SELECT NAME, TYPE, LINE, TEXT FROM USER\_SOURCE;



1. CREATE OR REPLACE TYPE tEmpleado FORCE UNDER tPersona

(NSS INTEGER,

salario NUMBER(7,2),

titulo VARCHAR(20),

fecha\_ingreso DATE,

CONSTRUCTOR FUNCTION tEmpleado(SELF IN OUT NOCOPY tEmpleado,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR, NSS VARCHAR, salario NUMBER, titulo VARCHAR, fecha\_ingreso DATE) RETURN SELF AS RESULT);

1. CREATE OR REPLACE TYPE BODY tEmpleado AS

CONSTRUCTOR FUNCTION tEmpleado (SELF IN OUT NOCOPY tEmpleado,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR, NSS VARCHAR, salario NUMBER, titulo VARCHAR, fecha\_ingreso DATE) RETURN SELF AS RESULT

IS

BEGIN

SELF.NSS := NSS;

SELF.salario := salario;

SELF.titulo := titulo;

SELF.fecha\_ingreso := TO\_DATE(REPLACE(fecha\_ingreso,'-','/'),'dd/mm/YYYY');

SELF.nombre := nombre;

SELF.apellido := apellido;

SELF.fecha\_nac := TO\_DATE(REPLACE(fecha\_nac,'-','/'),'dd/mm/YYYY');

SELF.genero := UPPER(genero);

END;

END;

/

1. CREATE OR REPLACE TYPE tDependiente FORCE UNDER tPersona

(parentesco VARCHAR(15),

CONSTRUCTOR FUNCTION tDependiente (SELF IN OUT NOCOPY tDependiente,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR, parentesco VARCHAR) RETURN SELF AS RESULT

);

1. CREATE OR REPLACE TYPE BODY tDependiente AS

CONSTRUCTOR FUNCTION tDependiente (SELF IN OUT NOCOPY tDependiente,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR, parentesco VARCHAR) RETURN SELF AS RESULT

IS

BEGIN

SELF.parentesco := UPPER(parentesco);

SELF.nombre := nombre;

SELF.apellido := apellido;

SELF.fecha\_nac := TO\_DATE(REPLACE(fecha\_nac,'-','/'),'dd/mm/YYYY');

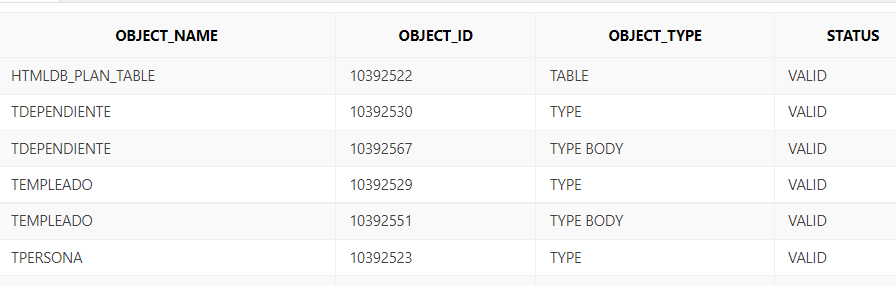
SELF.genero := UPPER(genero);

END;

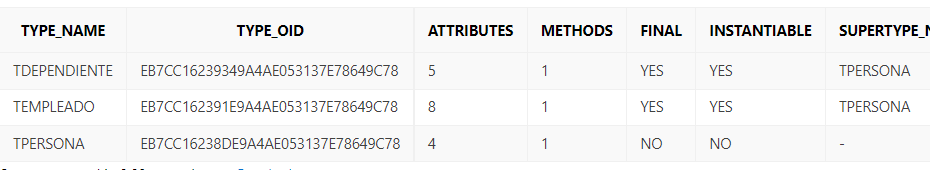
END;

/

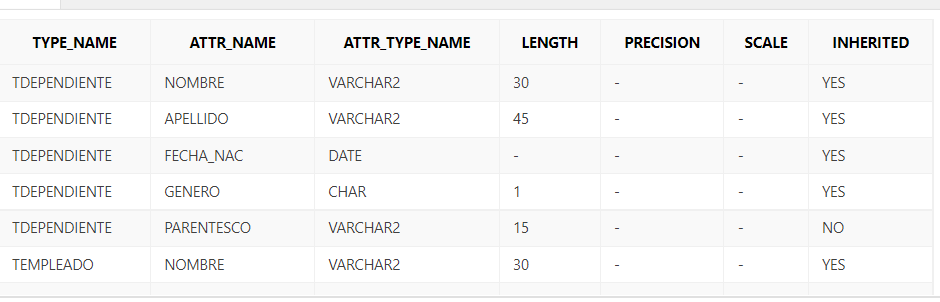
1. SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS;



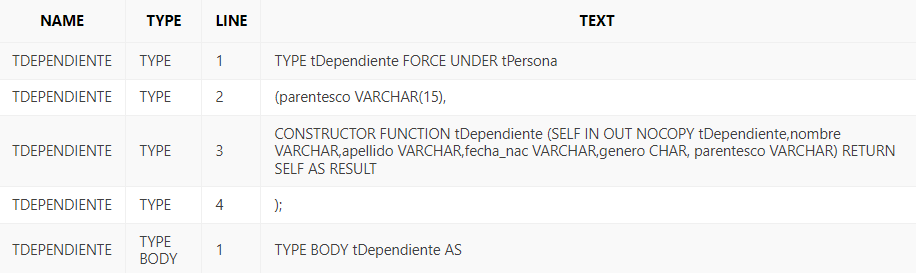
1. SELECT TYPE\_NAME, TYPE\_OID, ATTRIBUTES, METHODS, FINAL, INSTANTIABLE, SUPERTYPE\_NAME FROM USER\_TYPES;



1. SELECT TYPE\_NAME, ATTR\_NAME, ATTR\_TYPE\_NAME, LENGTH, PRECISION, SCALE, INHERITED FROM USER\_TYPE\_ATTRS;



1. SELECT NAME, TYPE, LINE, TEXT FROM USER\_SOURCE;



1. ALTER TYPE tPersona ADD MEMBER FUNCTION calcula\_edad RETURN NUMBER CASCADE;
2. ALTER TYPE tPersona ADD MEMBER FUNCTION obtener\_genero RETURN VARCHAR CASCADE;
3. CREATE OR REPLACE TYPE BODY tPersona AS

CONSTRUCTOR FUNCTION tPersona (SELF IN OUT NOCOPY tPersona,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR) RETURN SELF AS RESULT

IS

BEGIN

SELF.nombre := nombre;

SELF.apellido := apellido;

SELF.fecha\_nac := TO\_DATE(REPLACE(fecha\_nac,'-','/'),'dd/mm/YYYY');

SELF.genero := UPPER(genero);

END;

MEMBER FUNCTION calcula\_edad RETURN NUMBER IS

BEGIN

RETURN EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM SELF.fecha\_nac);

END;

MEMBER FUNCTION obtener\_genero RETURN VARCHAR IS

gen VARCHAR(50) := 'F';

BEGIN

IF SELF.genero = 'M' THEN

gen := 'Masculino';

ELSE

gen := 'Femenino' ;

END IF;

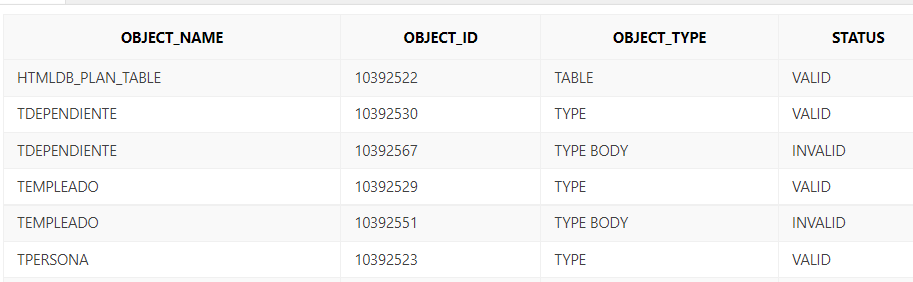
RETURN gen;

END;

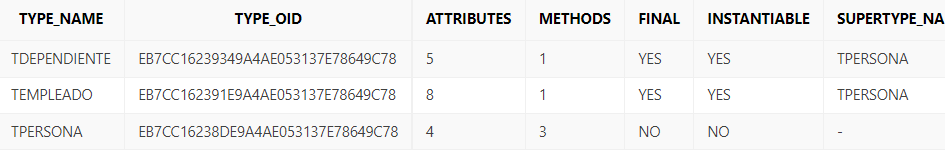
END;

/

1. SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS;



1. SELECT TYPE\_NAME, TYPE\_OID, ATTRIBUTES, METHODS, FINAL, INSTANTIABLE, SUPERTYPE\_NAME FROM USER\_TYPES;



1. ALTER TYPE tEmpleado ADD OVERRIDING MEMBER FUNCTION calcula\_edad RETURN NUMBER CASCADE;
2. ALTER TYPE tEmpleado ADD MEMBER FUNCTION salarioMensual RETURN NUMBER CASCADE;
3. ALTER TYPE tEmpleado ADD MEMBER FUNCTION calcula\_antiguedad RETURN NUMBER CASCADE;
4. CREATE OR REPLACE TYPE BODY tEmpleado AS

CONSTRUCTOR FUNCTION tEmpleado (SELF IN OUT NOCOPY tEmpleado,nombre VARCHAR,apellido VARCHAR,fecha\_nac VARCHAR,genero CHAR, NSS VARCHAR, salario NUMBER, titulo VARCHAR, fecha\_ingreso DATE) RETURN SELF AS RESULT

IS

BEGIN

SELF.NSS := NSS;

SELF.salario := salario;

SELF.titulo := titulo;

SELF.fecha\_ingreso := TO\_DATE(REPLACE(fecha\_ingreso,'-','/'),'dd/mm/YYYY');

SELF.nombre := nombre;

SELF.apellido := apellido;

SELF.fecha\_nac := TO\_DATE(REPLACE(fecha\_nac,'-','/'),'dd/mm/YYYY');

SELF.genero := UPPER(genero);

END;

OVERRIDING MEMBER FUNCTION calcula\_edad RETURN NUMBER IS

BEGIN

RETURN TRUNC((SYSDATE - SELF.fecha\_nac)/365);

END;

MEMBER FUNCTION salarioMensual RETURN NUMBER IS

BEGIN

RETURN SELF.salario;

END;

MEMBER FUNCTION calcula\_antiguedad RETURN NUMBER IS

BEGIN

RETURN EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM SELF.fecha\_ingreso);

END;

END;

/

1. CREATE OR REPLACE TYPE tDomicilio AS OBJECT

(calle VARCHAR(40),

numero NUMBER(4),

colonia VARCHAR(40),

ciudad VARCHAR(30),

cp CHAR(5)

);

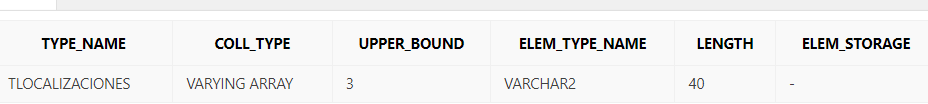
1. ALTER TYPE tEmpleado ADD ATTRIBUTE domicilio TDomicilio CASCADE;
2. CREATE OR REPLACE TYPE tDepartamento AS OBJECT

(nodep NUMBER(4),

nombre VARCHAR(30)

);

1. CREATE OR REPLACE TYPE tLocalizaciones AS VARRAY(3) OF VARCHAR(40);
2. ALTER TYPE tdepartamento ADD ATTRIBUTE localizacion tLocalizaciones;
3. SELECT TYPE\_NAME, COLL\_TYPE, UPPER\_BOUND, ELEM\_TYPE\_NAME, LENGTH, ELEM\_STORAGE FROM USER\_COLL\_TYPES;



1. ALTER TYPE tDepartamento ADD MEMBER FUNCTION lugares RETURN VARCHAR CASCADE;
2. CREATE OR REPLACE TYPE BODY tDepartamento IS

MEMBER FUNCTION lugares RETURN VARCHAR IS

salida VARCHAR(100) := '';

BEGIN

FOR i IN 1..SELF.localizacion.COUNT LOOP

salida := salida || 'ciudad ' || i || ': ' || SELF.localizacion(i) || ' ';

END LOOP;

RETURN salida;

END;

END;

/

1. CREATE OR REPLACE TYPE ref\_templeado AS OBJECT

(horas INTEGER,

refemp REF templeado);

1. CREATE OR REPLACE TYPE tNTemp\_proy AS TABLE OF ref\_templeado;
2. CREATE OR REPLACE TYPE tProyecto AS OBJECT

(noproy NUMBER(5),

nombre VARCHAR(50),

fecha\_inicio DATE,

fecha\_fin DATE,

empleados tNTemp\_proy

);

1. ALTER TYPE templeado ADD ATTRIBUTE jefe REF templeado CASCADE;
2. ALTER TYPE templeado ADD ATTRIBUTE dept REF tdepartamento CASCADE;
3. ALTER TYPE tdepartamento ADD ATTRIBUTE jefe REF templeado CASCADE;
4. ALTER TYPE tproyecto ADD ATTRIBUTE dept REF tdepartamento CASCADE;
5. CREATE OR REPLACE TYPE tPadres AS VARRAY(2) OF REF templeado;
6. ALTER TYPE tDependiente ADD ATTRIBUTE padres tPadres CASCADE;
7. ALTER TYPE tDepartamento ADD MEMBER FUNCTION obtener\_jefe RETURN VARCHAR CASCADE;
8. CREATE OR REPLACE TYPE BODY tDepartamento IS

MEMBER FUNCTION lugares RETURN VARCHAR IS

salida VARCHAR(100) := '';

BEGIN

FOR i IN 1..SELF.localizacion.COUNT LOOP

salida := salida || 'ciudad ' || i || ': ' || SELF.localizacion(i) || ' ';

END LOOP;

RETURN salida;

END;

MEMBER FUNCTION obtener\_jefe RETURN VARCHAR IS

jefe\_ref templeado;

BEGIN

UTL\_REF.SELECT\_OBJECT(SELF.jefe, jefe\_ref);

RETURN jefe\_ref.nss || ': ' || jefe\_ref.apellido || ', ' || jefe\_ref.nombre;

END;

END;

/

1. ALTER TYPE tProyecto ADD MEMBER FUNCTION gastos RETURN NUMBER CASCADE;
2. CREATE OR REPLACE TYPE BODY tProyecto IS

MEMBER FUNCTION gastos RETURN NUMBER IS

i NUMBER;

emp\_ref templeado;

total NUMBER := 0;

BEGIN

FOR i IN 1..SELF.empleados.COUNT LOOP

UTL\_REF.SELECT\_OBJECT(empleados(i).refemp, emp\_ref);

total := total + emp\_ref.salario;

END LOOP;

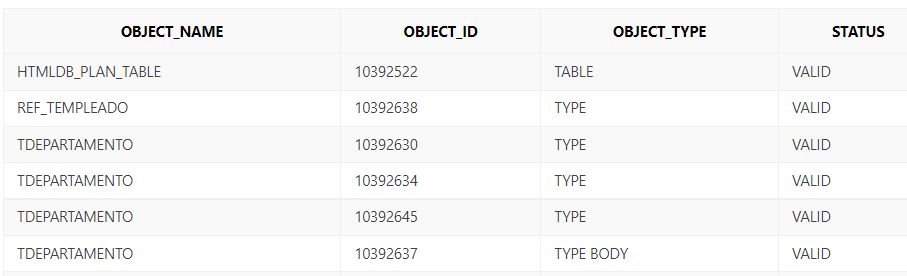
RETURN total;

END;

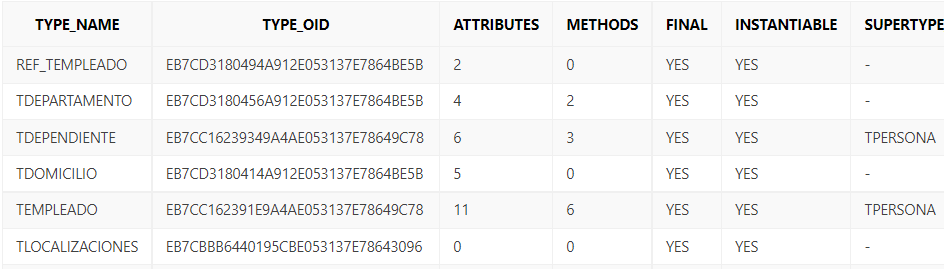
END;

/

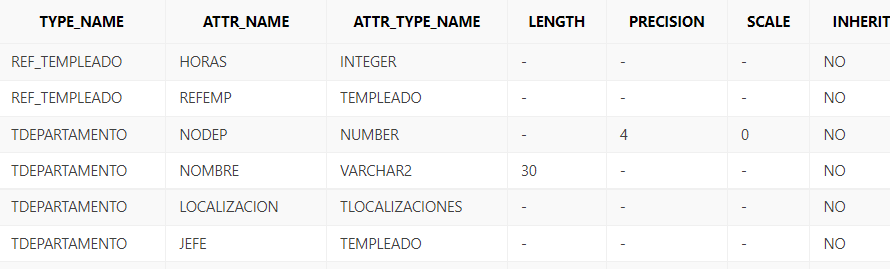
1. SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS;



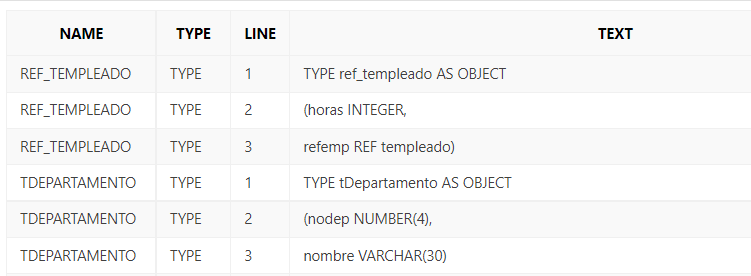
1. SELECT TYPE\_NAME, TYPE\_OID, ATTRIBUTES, METHODS, FINAL, INSTANTIABLE, SUPERTYPE\_NAME FROM USER\_TYPES;



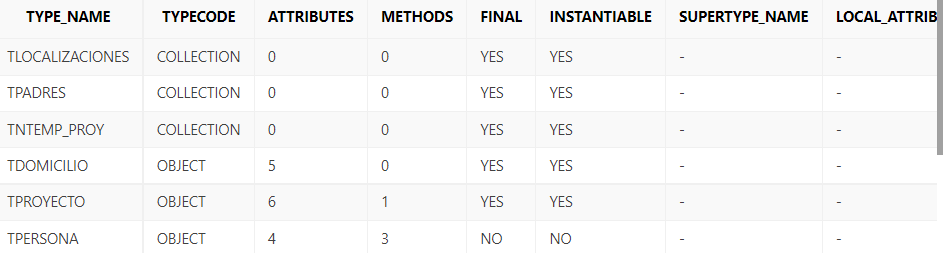
1. SELECT TYPE\_NAME, ATTR\_NAME, ATTR\_TYPE\_NAME, LENGTH, PRECISION, SCALE, INHERITED FROM USER\_TYPE\_ATTRS;



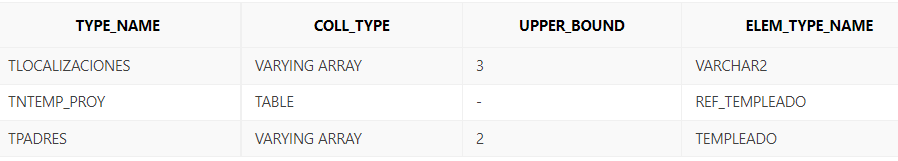
1. SELECT NAME, TYPE, LINE, TEXT FROM USER\_SOURCE;



1. SELECT TYPE\_NAME, TYPECODE, ATTRIBUTES, METHODS, FINAL, INSTANTIABLE, SUPERTYPE\_NAME, LOCAL\_ATTRIBUTES, LOCAL\_METHODS FROM USER\_TYPES ORDER BY TYPECODE;



1. SELECT TYPE\_NAME, COLL\_TYPE, UPPER\_BOUND, ELEM\_TYPE\_NAME FROM USER\_COLL\_TYPES;



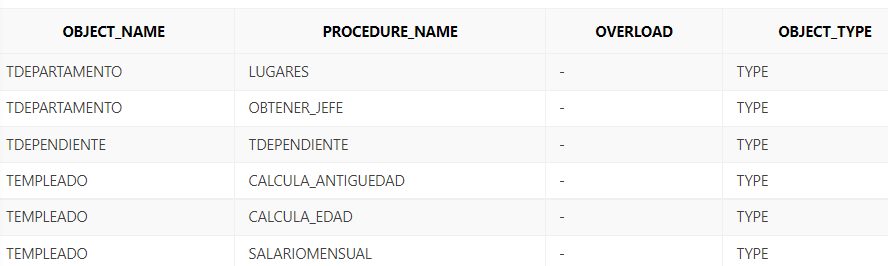
1. SELECT TABLE\_NAME, TABLE\_TYPE\_NAME, PARENT\_TABLE\_NAME, PARENT\_TABLE\_COLUMN FROM USER\_NESTED\_TABLES;



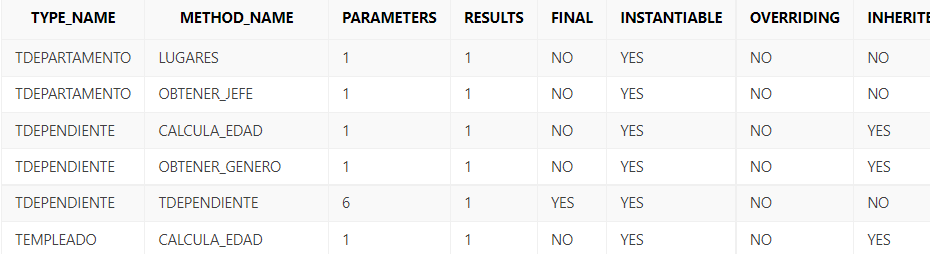
1. SELECT TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE, DATA\_TYPE\_MOD, DATA\_LENGTH, NULLABLE FROM USER\_NESTED\_TABLE\_COLS;



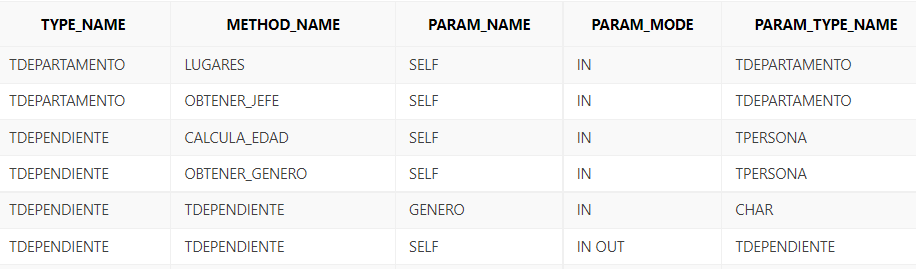
1. SELECT OBJECT\_NAME, PROCEDURE\_NAME, OVERLOAD, OBJECT\_TYPE FROM USER\_PROCEDURES;



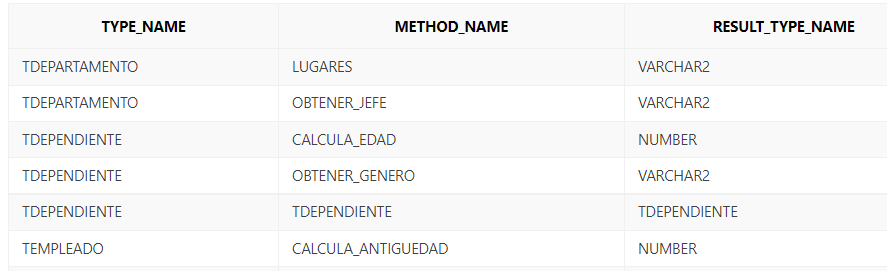
1. SELECT TYPE\_NAME, METHOD\_NAME, PARAMETERS, RESULTS, FINAL, INSTANTIABLE, OVERRIDING, INHERITED FROM USER\_TYPE\_METHODS ORDER BY TYPE\_NAME;



1. SELECT TYPE\_NAME, METHOD\_NAME, PARAM\_NAME, PARAM\_MODE, PARAM\_TYPE\_NAME FROM USER\_METHOD\_PARAMS ORDER BY TYPE\_NAME, METHOD\_NAME;



1. SELECT TYPE\_NAME, METHOD\_NAME,RESULT\_TYPE\_NAME FROM USER\_METHOD\_RESULTS ORDER BY TYPE\_NAME, METHOD\_NAME;



Incluya las capturas de pantalla (legibles y en buena resolución) con los resultados obtenidos en las lineas de código indicadas en resaltado amarillo y guarde el documento. Resuelva el cuestionario y escriba sus conclusiones. Envíe el documento para su revisión en tiempo y forma. No se aceptan archivos fuera del reporte de la práctica, ni archivos alojados en sitios externos.

CUESTIONARIO

1.- Explique claramente cómo Oracle implementa, mediante sentencias SQL, los siguientes conceptos de Orientación a Objetos en el modelo relacional:

a) UDT (Tipos definidos por el usuario)

El término UDT se usa indistintamente con los tipos de objetos de Oracle y los tipos de datos abstractos (ADT).

El nombre del UDT de Oracle distingue entre mayúsculas y minúsculas y debe tener el formato *schema\_name.type\_name*.

Pasos para invocar el metodo UDT

CommandTyp como CommandType.StoredProcedure

CommandTextcomo"*type\_name.procedure\_name*"

el comando usando cualquiera de los Executemétodos en el OracleCommandobjeto.

CREATE TABLE dbo.Points

(ID int IDENTITY(1,1) PRIMARY KEY, PointValue Point)

b) Herencia

CREATE TYPE tipoPersona AS OBJECT

(DNI VARCHAR(9),

nombre VARCHAR(25),

fecha\_nac DATE,

direccion VARCHAR(1000),

MEMBER FUNCTION edad RETURN NUMBER)

NOT INSTANTIABLE NOT FINAL;

CREATE TYPE tipoEstudiante UNDER

tipoPersona

(estudios VARCHAR(50));

CREATE TYPE tipoEmpleado UNDER

tipoPersona

(empresa VARCHAR(50));

c) Composiciones

2.- ¿En qué situaciones es necesario emplear los modificadores NOT INSTANTIABLE, NOT FINAL y OVERRIDING?

NOT INSTANTIABLE : Para cuando no queremos crear ninguna función constructora para el objeto

NOT FINAL : Para cuando posiblemente queramos definir en un futuro subclases instanceables

OVERRIDING : Para ejecutar una intromision dentro de los procesos de una manera predefinida

CONCLUSIONES

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

El uso de la creacion de los tipos para la generacion de plantillas y la definición de una base de datos nos ayuda a generar con mayo rapidas las tablas y las selecciones necesarias para la definición de los mismo .

En cuanto a las herencia al poder utilizar los tipos como preplantillas, el unicamente poder heredar todo con mayor rapidez , no ayuda agilizar la creacion de tablas y consultas.

REFERENCIAS BIBLIOGRÁFICAS (APA, IEEE, ETC.)

1.- [Database Object-Relational Developer's Guide - Contents (oracle.com)](https://docs.oracle.com/cd/E11882_01/appdev.112/e11822/toc.htm)

2.- Piattini et al. (2006): Tecnología y Diseño de Bases de Datos.

3.- Connolly y Begg (2005): Sistemas de Bases de Datos.

4.- Jeffrey A. Hoffer, Mary B. Prescott, Heikki Topi. Modern Database Management, Ninth Edition.

Pearson/Prentice, Estados Unidos 2009, págs. 690

5.- Raghu Ramakrishnan, Johannes Gehrke. Sistemas de Gestión de Bases de Datos, Tercera Edición. McGrawHill/Interamericana de España, Madrid España 2007, págs. 654

6.- Ramez Elmasri, Shamkant B. Navathe. Fundamentos de Sistemas de Bases de Datos, Quinta Edición. Pearson/Addison Wesley, Madrid España 2007, págs. 988