BASES DE DATOS B UF4

Bases de datos objetos relacionales

Introducción

- Las bases de datos pueden modelar o ir modificando todas estas características, mientras que las bases de datos objeto-relacionales son como una extensión del modelo relacional que hemos visto. No utilizan algunas reglas que antes sí llevábamos a cabo, pero añaden otra nueva solución a la hora de modelar la información.

- La mayor diferencia que existe entre los dos modelos es la existencia de tipos de objetos. Los tipos se crean mediante sentencias ODL (Object Definition Lenguage) y van a ser la base del desarrollo de las bases de datos objeto- relacionales.

Tipos de objetos

- Un tipo de objeto representa una entidad del mundo real que consta de las siguientes partes:
  - Un nombre que permite identificar el tipo objeto
  - Unos atributos que caracterizan el objeto
  - Unos métodos que definen las operaciones sobre los datos de ese tipo escritos en PL/SQL.

Características

- La principal característica que debemos destacar de las bases de datos objetos-relacionales es que combinan el modelo relacional, evolucionan con la incorporación de conceptos del modelo orientado a objetos.

- En el modelo objeto- relacional:
  - Cada registro de una tabla se considera un objeto.
  - Y la definición de la tabla, su clase.

- Este modelo tiene capacidad para gestionar tipos de datos complejos, lo cual, contradice algunas de las restricciones establecidas por el modelo relacional.

Encapsulamiento, Herencia y Polimorfismo

- Conocemos por **encapsulamiento** al mecanismo que vamos a seguir para agrupar los atributos y métodos dentro de un nuevo concepto que denominamos clase.

- La **herencia** es el mecanismo por el cual una clase derivada va a heredar los atributos de otra.

- Se utiliza el **polimorfismo** cuando una clase derivada debe verse como la clase principal.

Definición de tipos de Objeto.

- Para poder crear tipos de objetos se debe hacer uso de la sentencia CREATE TYPE. Está compuesta por los siguientes elementos:
  - Para identificar el tipo de objetos se utiliza un nombre.
  - Unos atributos que pueden ser de un tipo de datos básico o de un tipo definido por el usuario, que representan la estructura y los valores de los datos de ese tipo.
  - Unos métodos que son procedimientos o funciones. Se declaran con la cláusula MEMBER

CÓDIGO:

```
-- Ejemplo
CREATE OR REPLACE TYPE persona AS OBJECT (
   nombre VARCHAR2(30);
   teléfono VARCHAR2(20)
);
```

Definición de métodos

- Un método miembro de un tipo de objeto debe cumplir las siguientes características:
  - No puede insertar, actualizar o borrar las tablas de la base de datos
  - No se puede ejecutar en paralelo o remotamente si va a acceder a los valores de la una variable dentro de un módulo
  - No puede modificar una variable de un módulo excepto si se invoca desde una cláusula SELECT, VALUES o SET
  - No puede invocar a otro módulo o subprograma que rompa alguna de las reglas anteriores.

Definición de métodos

- La directiva PRAGMA_REFERENCES se utiliza para forzar las reglas anteriores.
- Donde restricciones puede ser cualquiera de las siguientes o incluso una combinación de ellas:
  - WINDS: evita que el método pueda modificar las tablas de la base de datos.
  - RNDS: evita que el método pueda leer las tablas de la base de datos.
  - WNPS: evita que el método modifique variables del paquete PL/SQL.
  - RNPS: evita que el método pueda leer las variables del paquete PL/SQL.

CÓDIGO:

```
-- Ejemplo Definición de clase
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,
   PRAGMA RESTRICT_REFERENCES (muestraEdad, WNDS)
);
```

CÓDIGO:

```
-- Ejemplo Cuerpo de clase
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;
```

Ejemplo Objeto “Persona”

Creación de un objeto

CÓDIGO:

```
-- Ejemplo
SET SERVEROUTPUT ON;
DECLARE
   Trabajador1 Persona;
BEGIN
   Trabajador1:= NEW Persona(1, '123456', 'Alberto', 'Olivia',
                                          '22/12/1989');
   dbms_output.put_line('El empleado '|| Trabajador1.nombre
                          ||' ha sido creado');
   dbms_output.put_line('Edad del empleado: '
                          ||Trabajador1.muestraEdad());
END;
```

Colecciones

- Para poder contar con atributos multievaluados (1:N) es necesario que los organicemos en una estructura de datos (array).
- Estas colecciones de datos las podemos generar de un tipo de dato (Number, Varchar2, etc…) o sobre un tipo de objeto.

CÓDIGO:

```
-- Ejemplo
-- Lista de un tipo primitivo varchar2
CREATE TYPE colec_nombres AS VARRAY(10) OF VARCHAR2 (20);
-- Lista de un tipo de usuario: Persona
CREATE TYPE colec_personas AS VARRAY(10) OF Persona;
```

Ejemplos de colecciones

- Estas colecciones de datos las podemos utilizar en la definición de una clase (objeto) o como definición de un campo de una tabla.

CÓDIGO:

```
-- Ejemplo
CREATE OR REPLACE TYPE Departamento AS OBJECT(
        NombreDept VARCHAR2(100),
        Empleados colec_personas
);
```

CÓDIGO:

```
-- Ejemplo
CREATE TABLE Empleados (
        NombreDept VARCHAR2(50),
        Datos_Empleado Persona
);
```

Herencia

- Una de las principales ventajas de la programación orientada a objetos (POO) es la herencia.
- Se pueden crear superclases abstractas para que, en adelante, se puedan crear subclases más específicas que hereden atributos y métodos de las superclases.
- El supertipo define los atributos o métodos que van a compartir con los subtipos, todos los objetos que hereden de él

Herencia

- La cláusula NOT FINAL hace referencia a que este objeto no es el último; es decir, cuando creemos subtipos de objetos que cuelguen de este, tendremos que declarar el objeto con esta cláusula. En el caso de no poner la cláusula al crear este objeto, cuando queramos colgar otros objetos de este nos devolverá error, aunque nos compilara el objeto.

CÓDIGO:

```
-- Ejemplo

CREATE TYPE tipo_person AS OBJECT (
   nombre VARCHAR2(25),
   fecha_nac DATE,
   MEMBER FUNCTION edad RETURN NUMBER,
   MEMBER FUNCTION printme RETURN VARCHAR2
) NOT FINAL;
```

Herencia

Para crear un subtipo utilizamos la cláusula UNDER.

CÓDIGO:

```
-- Ejemplo

CREATE TYPE tipo_empleado UNDER tipo_person (
      sueldo NUMBER,
      MEMBER FUNCTION paga RETURN NUMBER,

     OVERRIDING MEMBER FUNCTION printme RETURN VARCHAR2);
```

Identificadores, referencias

- El tipo REF o referencia es un contenedor de un identificador de objeto (object identifier – OID). **Es un puntero a un objeto.**

CÓDIGO:

```
-- Ejemplo
CREATE TYPE autor AS OBJECT (
        id INTEGER,
        nombre VARCHAR(100),
        direccion VARCHAR(255)
);
CREATE TYPE libro AS OBJECT (
        id INTEGER,
        nombre VARCHAR(200),
        escritor REF autor
);
```

BASES DE DATOS B UF4

Bases de datos objetos relacionales