Laboratorio #9

• Bases de Datos Orientadas a Objetos

ORACLE tiene la capacidad de manejar una gran variedad de tipos de bases de datos. En esta práctica nos centraremos en las bases de datos Orientadas a Objetos dentro de ORACLE, puesto que aunque son manejables en diversos gestores, cada uno posee sus diferencias.

En una base de datos orientada a objetos la información deja de representarse únicamente en base a tablas, ya que se recurre también a objetos. Se dice que una base de datos orientada a objetos es una que incorpora los principios básicos del paradigma orientado a objetos:

- ✓ Encapsulamiento: Capacidad de un objeto para ocultar su funcionamiento interno.
- ✓ Polimorfismo: Una operación puede aplicarse a distintos tipos de objetos.
- ✓ Herencia: Los objetos pueden heredar características a otros mediante cierta jerarquía.

En primer lugar, la sintaxis general para la creación de un objeto en ORACLE es la siguiente:

```
CREATE [OR REPLACE] TYPE <type_name>
        [AUTHID {CURRENT USER | DEFINER}]
        {{AS|IS} OBJECT | UNDER <supertype_name>}
        (
        <attribute_name> DATATYPE [, attribute_name DATATYPE]...
        [{MAP | ORDER} MEMBER <function_spec,]
        [{FINAL | NOT FINAL}] MEMBER <function_spec,]
        [{INSTANTIABLE | NOT INSTANTIABLE} MEMBER <function_spec,]
        [{MEMBER | STATIC} {<subprogram_spec> | <call_spec>}
        [, {MEMBER | STATIC} {<subprogram_spec> | <call_spec>}] ...]
        ) [{FINAL | NOT FINAL}] [{INSTANTIABLE | NOT INSTANTIABLE}];

[CREATE [OR REPLACE] TYPE BODY <type_name> {IS|AS}
        {{MAP|ORDER} MEMBER <function_body>;
        | {MAP|ORDER} {subprogram_body | call_spec };}
        [{MEMBER|STATIC} {<subprogram_body> | <call_spec>};] ...
        END;]
```

Se debe notar que además de definir el objeto, es decir sus tributos y métodos, debe posteriormente definirse el cuerpo de los métodos que el objeto posee.

Un método que pertenece a un objeto en ORACLE, puede ser representado de 2 formas, ya sea como una función, o como un procedimiento almacenado, el que se escoja para cada caso dependerá de si el método retorna o no un valor. En cuanto a la sintaxis para definir cada método por si mismo, esta no presenta mayores variaciones con respecto a como se define una función o procedimiento almacenado en PL/SQL.

Almacenar objetos en una tabla

Declarar tabla

Cuando se desea aprovechar la capacidad de crear objetos para encapsular datos, y almacenarlos directamente en una tabla, simplemente debe declararse que un campo de nuestra tabla es del tipo del objeto que deseamos.

```
CREATE TABLE nombre_tabla(
        Campo1 tipo,
        Cmpo2 tipo,
        …
        CampoObjeto Tipo_Objeto
);
```

Insertar objeto dentro de tabla

Si dentro de una tabla poseemos un campo que pertenece a un tipo de objeto declarado de forma previa, al momento de realizar un INSERT en dicha tabla, el objeto debe insertarse de la misma forma como se declararía una instancia de ese objeto, es decir:

```
INSERT INTO nombre_tabla VALUES (campo 1, campo2, … ,
CampoObjeto(‘parametro1, ‘parametro2’, … , ‘parametroN’);
```

Mostrar campos de tipo objeto

Para poder acceder a los atributos de un objeto almacenado dentro de una tabla, debe realizarse mediante la notación: Objeto.atributo, esto aplica de igual forma para los métodos que dicho objeto pueda poseer.

Creación de tablas a partir de objetos

Otra vía para almacenar objetos dentro de tablas es que la tabla sea creada en base a un objeto, es decir los campos de la tabla sean los atributos de un objeto declarado de forma previa. La sintaxis para realizarlo es la siguiente:

```
CREATE TABLE OF Nombre_Objeto [ (
        NuevoCampo tipo,
        NuevoCampo2 tipo,
        …
) ];
```

Herencia

Como se mencionó una característica de la programación orientada a objetos es la herencia. Esta es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, en otras palabras, dado un tipo padre del que heredan uno o varios subtipos hijo, estos extienden su funcionalidad agregando sus propios atributos, además de poseer los mismos de su padre.

Es importante mencionar que cuando se crea un subtipo a partir de un tipo, el subtipo hereda todos los atributos y los métodos del tipo padre. Cualquier cambio en los atributos o métodos del tipo padre se reflejan automáticamente en el subtipo.

Cuando se desea que exista una jerarquía de herencia entre distintos objetos, el objeto padre debe ser declarado como NOT FINAL, debe tenerse en cuenta que este es el valor por defecto para cada objeto que se crea.

Sobreescritura de métodos

Como es sabido, la programación orientada a objetos permite que clases hijo que heredan de otras clases padre puedan modificar los parámetros recibidos o la lógica de sus propios métodos, con respecto a la estructura que estos tenían en la clase padre, a esto se le conoce como sobreescritura de métodos.

En ORACLE, como se mencionó, los subtipos derivados de otros heredan los métodos del tipo original, pero en caso que desee modificarse, primero debe declararse dentro de la definición del nuevo subtipo, esto se hace con el atributo OVERRIDING, esto indica que dentro del subtipo, este nuevo método se sobrescribirá al del tipo padre.

```
CREATE OR REPLACE TYPE Nombre_Objeto UNDER Objeto_Padre
(
   Atributos…,
   OVERRIDING MEMBER [ PROCEDURE | FUNCTION ]
nombre_metodo
);
```

Además de NOT FINAL, es importante mencionar otro atributo estrechamente relacionado, NOT INSTANTIABLE, este atributo indica que un objeto no podrá ser instanciado, sin embargo, este atributo no es exclusivo de objetos, pues también los métodos de un objeto pueden etiquetarse con este atributo.