MEDAC
Instituto Oficial de Formación Profesional

ACCESO A DATOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Bases de datos objeto relacionales

10

ÍNDICE

/ 1. Introducción y contextualización práctica 3

/ 2. Definición de base de datos objeto relacionales 4

/ 3. Características de las bases de datos objeto relacionales 4

/ 4. Tablas y tipos de objetos: definición 5

/ 5. Caso práctico 1: “Nuevos tipos” 6

/ 6. Tablas y tipos de objetos: explotación 7

/ 7. Tipos de colección: array 8

/ 8. Tipos de colección: tablas anidadas 9

/ 9. Referencias 9

/ 10. Herencia de tipos 10

/ 11. Caso práctico 2: “Herencia de tipos” 11

/ 12. Resumen y resolución del caso práctico de la unidad 12

/ 13. Bibliografía 12

/ 14. Webgrafía 12

OBJETIVOS

Conocer las características de las bases de datos objeto relacionales.

Estudiar los diferentes objetos y las novedades que ofrecen este tipo de base de datos.

/ 1. Introducción y contextualización práctica

A lo largo de esta unidad, nos enfrentaremos a un nuevo modelo de almacenamiento de información, los sistemas gestores de datos objeto relacionales. Aprenderemos las diferentes características que poseen. Además, indagaremos en sus diferentes tipos de objetos, como las colecciones, las tablas anidadas, etc.

Veremos cuál es el concepto de «referencia» en bases de datos objeto relacionales y, además, dedicaremos un apartado a la herencia de tipos.

Planteamiento del caso práctico inicial

A continuación, vamos a plantear un caso práctico a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica. Encontrarás su resolución en el apartado «Resumen y resolución del caso práctico de la unidad».

Fig. 1. Bases de datos objeto-relacionales.

Audio intro. "Acceder al atributo"
https://bit.ly/2FwmYOe

/ 2. Definición de base de datos objeto relacionales

Una base de datos objeto-relacional es una combinación de una base de datos orientada a objetos y un modelo de base de datos relacional. Esto significa que soporta objetos, clases, herencia, etc. que tendríamos en los modelos orientados a objetos y también tiene soporte para tipos de datos o estructuras tabulares, entre otras cosas, del modelo de datos relacional.

Uno de los mayores objetivos de los modelos de base de datos objeto-relacionales es acortar distancias entre las bases de datos relacionales y las practicas orientadas a objetos realizadas frecuentemente en diferentes lenguajes como C++, Java, C#, etc.

Con esta información que se ha adelantado de base, podríamos definir la **base de datos objeto-relacional** como aquella base de datos relacional que evoluciona desde dicho modelo hacia a algunas de las características del modelo de objetos, haciéndola una **base de datos híbrida.**

Uno de los gestores de bases de datos más conocidos hoy en día es Oracle. Este implementa el modelo de objeto como una extensión del modelo relacional.

Muchos lenguajes, tal y como hemos comentado anteriormente, han sabido adaptarse con extensiones y *frameworks* para poder trabajar con estas nuevas bases de datos objeto relacionales de Oracle.

El resultado es un modelo relacional de objetos que ofrece la intuición y la economía de una interfaz de objetos, al mismo tiempo que conserva su alta concurrencia y el rendimiento de una relacional.

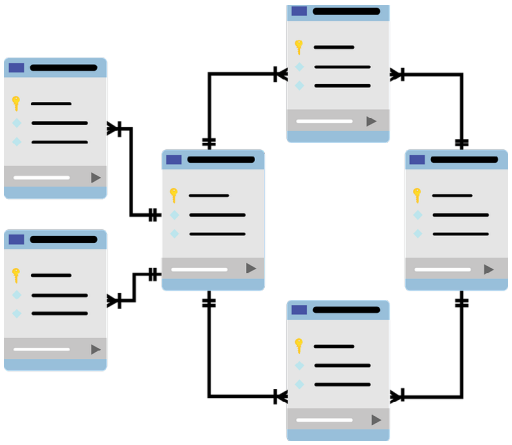

Fig. 2. Esquema relacional

/ 3. Características de las bases de datos objeto relacionales

Una de las principales características de este tipo de base de datos es que podremos crear nuevos tipos de datos, los cuales permitirán gestionar aplicaciones específicas con mucha riqueza de dominios. Estos nuevos tipos de datos pueden ser tipos compuestos, lo que nos lleva a pensar que se podrán definir, al menos, dos métodos:

- Uno para convertir de este tipo a caracteres ASCII
- Y otro que haga esta función a la inversa, desde caracteres ASCII hasta nuevos tipos de datos.

Se soportarán distintos tipos complejos como, por ejemplo:

- Registros
- Listas
- Referencias
- Pilas
- Colas
- *Arrays*

Con dicha tipología de datos, podremos crear también funciones que tengan código en diferentes lenguajes, como SQL, Java, C#, etc.

Otro de los aspectos que caracterizan a las bases de datos objeto-relacionales son:

- Dispondremos de una mayor capacidad de expresión para definir conceptos y diferentes asociaciones.
- Podremos crear también operadores asignando nombre y existencia de aquellas consultas más complejas.
- En los tipos de registro, estilo relacional, podremos usar encadenamiento y herencia.
- Podremos hacer uso de la reusabilidad, compartiendo bibliotecas de clases definidas previamente.
- Posibilidad de introducir comprobación de reglas de integridad por medio de *triggers*.

Fig. 3. Características Base de datos OR

Audio 1. "Triggers"
https://bit.ly/3mrr4Yq

/ 4. Tablas y tipos de objetos: definición

Cuando se crea un tipo de dato, realmente estamos definiendo cierto comportamiento para una agrupación de datos de nuestra aplicación. En Oracle, con la base de datos objeto-relacional, tendremos la opción de definir nuestros propios tipos de datos. Para los tipos de objetos, usaremos ***object type*** y, para los tipos de colecciones, ***collection type***. Para construir dichos tipos de usuario, deberemos usar los básicos que poseemos en el sistema.

Como ya hemos visto en unidades anteriores, un objeto representa una entidad en el mundo real y se compone de:

- **Nombre**: Con el que identificaremos el tipo de objeto
- **Atributos**: Con los que definiremos la estructura. Los atributos pueden ser de tipo creado por el usuario o básico del propio sistema.
- **Métodos**: Que pueden ser funciones o procedimientos. Los encontraremos escritos en código PL/SQL cuando están almacenados en la propia base de datos y en el lenguaje C cuando se almacenan externamente.

Diremos que la creación de un método en Oracle se realiza junto a la creación de su tipología y debe llevar siempre el tipo de compilación como, por ejemplo, PRAGMA RESTRICT_REFERENCES; de esta forma, evitamos la manipulación de los diferentes datos o de las distintas variables PL/SQL.

A continuación, se expone un fragmento de código donde crearemos un tipo de dato nuevo en el lenguaje, establecido por la base de datos Oracle:

```
CREATE TYPE address_t AS OBJECT (
street VARCHAR2(200),
city VARCHAR2(200),
prov CHAR(2),
poscode VARCHAR2(20));
```

Código 1. Creación tipo address_t.

Como podemos observar en la creación de la tabla:

- Indicaremos el nombre del tipo a definir "address_t"
- Estableceremos cuatro atributos diferentes que definen la estructura creada:
  - **Street:** Es un tipo texto VARCHAR2, con 200 caracteres máximo.
  - **City:** Otro VARCHAR2, también con 200 de extensión.
  - **Prov:** Válido para introducir 2 caracteres máximo.
  - **Poscode:** Un VARCHAR2 de 20 caracteres.

Vídeo 1. "Nulos"

https://bit.ly/2FDJzIr

/ 5. Caso práctico 1: "Nuevos tipos"

Planteamiento: Nuestro compañero del departamento de *software* tiene por delante una nueva tarea de diseño de base de datos. Se le requiere la creación de 2 nuevos objetos:

- Objeto Vehículo con los atributos: número ruedas, peso, largo.
- Objeto Coche con los atributos: tipo, marca.

Hay que tener en cuenta que el objeto Coche es un tipo de Vehículo.

Nudo: según hemos estudiado, en Oracle, con la base de datos objeto-relacional, podremos agregar nuestros propios tipos de objetos a la base de datos.

Desenlace: Inicialmente, se van a definir los 2 objetos que se requieren. Habría que reflexionar sobre qué tipología tienen dichos objetos:

```
CREATE TYPE vehiculo_t AS OBJECT (
nRuedas NUMBER,
peso NUMBER,
largo NUMBER;
```

Código 2. Definición de objetos.

En primer lugar, definiríamos el objeto vehículo teniendo en cuenta que, tanto para el número de ruedas, como el peso, y el largo, son variables del tipo básico del sistema NUMBER.

```
CREATE TYPE coche_t AS OBJECT (
tipo vehiculo_t,
marca VARCHAR2(50);
```

Código 3. Definición objeto vehículo.

En esta segunda definición, habría que caer en la cuenta que conlleva herencia, por lo tanto, la tipología de este coche es vehículo, precisamente el mismo objeto que definimos previamente.

Por último, la marca nos valdría con un Varchar de 50 caracteres.

/ 6. Tablas y tipos de objetos: explotación

Una vez se ha realizado la definición de tipos, podemos tener distintos objetivos para esos nuevos datos: podemos usarlos para definir nuevos tipos, para almacenarlos en tablas de ese tipo de datos o para definir los distintos atributos de una tabla.

Sabemos que una tabla de objetos es una clase específica de tabla que almacenará un objeto por cada fila y que, al mismo tiempo, facilita el ingreso de los atributos del mismo objeto, como si se tratara de columnas de la tabla. Por ejemplo, podríamos tener una tabla de vehículos del año actual y otra para guardar vehículos de años anteriores:

```
CREATE TABLE vehiculos_año_tab OF vehiculo_t
(numVehiculo PRIMARY KEY);

CREATE TABLE vehiculos_antiguos_tab
( anio NUMBER, vehiculo vehiculo_t);
```

Código 4. Tablas Oracle.

La diferencia entre la primera y la segunda tabla es que la primera almacena objetos con su propio ID y la segunda, realmente, no es una tabla de objetos, sino una tabla con una columna cuyo tipo de dato es un objeto.

La segunda tabla posee una columna con un tipo de datos complejo y sin identidad de objeto. Es una de las ventajas que ofrece Oracle. Aparte, Oracle nos permite definir como tabla:

- Una columna con tipología de objeto.
- Aquella que tiene el mismo número de columnas como atributos que almacena.

Fig. 4. Base de datos Oracle

/ 7. Tipos de colección: array

Para poder establecer relaciones «uno a muchos» (1:N), Oracle nos permite definir colecciones. Una colección está formada por un número no definido de elementos y todos ellos deben ser del mismo tipo. De esta forma, podemos guardar en un simple atributo un conjunto de datos en forma de *array* (Varray) o, también, tendríamos la opción de la tabla anidada.

Fig. 5. Colecciones en Oracle

EL VARRAY

Como bien sabemos, podríamos definir un *array* como una serie de elementos ordenados que son del mismo tipo.

Estos elementos llevan asociado un índice que nos sirve para saber su posición dentro del *array*.

Oracle permite que el tipo **VARRAY** sea un tipo de dato variable, pero sí se debe establecer el máximo de elementos una vez se declara dicho tipo. Podemos ver algún ejemplo:

```
CREATE TYPE numeros AS VARRAY(10)
 OF NUMBER(10);
numeros ('6', '18', '75870');
```

Código 5. VARRAY.

Tal y como se observa en el Código 5, hemos definido un nuevo tipo «números» como un VARRAY con máximo 10 elementos y con posiciones cuyo tipo serán NUMBER máximo 10 cifras.

Utilizaremos el VARRAY para:

- Definir la tipología de una columna de una tabla relacional.
- Definir la tipología de un atributo de un tipo objeto.
- Definir una variable del lenguaje PL/SQL.

Una vez declaramos este objeto VARRAY, no se reserva realmente ninguna cantidad de espacio. Si el espacio está disponible, se almacena igual que el resto de columnas, pero, si por el contrario, es superior a 4.000 bytes, se almacenará en una tabla aparte, como un dato de tipo BLOB.

Vídeo 2. "BLOB"
https://bit.ly/35Nt07E

/ 8. Tipos de colección: tablas anidadas

Tal y como vimos en la sección anterior, podemos dividir los tipos de colección en Oracle objeto-relacional en:

- VARRAYS
- Tablas anidadas

Una vez visto el concepto y cómo funcionan los VARRAY, a continuación, definiremos las tablas anidadas y veremos sus características:

- **Tablas anidadas:** Una tabla anidada es una lista de elementos no ordenados que mantienen una misma tipología. El máximo no está especificado en la definición de la tabla, y el orden de los elementos no se mantiene.

Realizaremos SELECT, INSERT, DELETE y UPDATE de la misma forma que lo hacemos con las tablas comunes, usando la expresión TABLE.

Una tabla anidada puede ser vista o interpretada como una única columna. Si la columna en una tabla anidada es un tipo de objeto de usuario, la tabla puede ser vista como una tabla multicolumna, con una columna por cada atributo del objeto usuario que fue definido.

Fig. 6. Imagen tablas anidadas.

La sintaxis para crear una tabla anidada es la siguiente:

```
CREATE TYPE nombre_tipo AS TABLE OF tabla_tipo;
```

Código 6. Sintaxis tabla anidada.

Con este código, estaremos creando una tabla anidada *tabla_tipo*, la cual contendrá objetos de tipo de usuario *nombre_tipo*.

La definición que hemos visto justo arriba no asignará espacio. Una vez definido el tipo, podremos usarlo para:

- El tipo de datos de una tabla relacional
- Un atributo de un objeto de tipo usuario
- Una variable PL/SQL, un parámetro o una función que devuelva un tipo.

/ 9. Referencias

La base de datos objeto-relacional Oracle permite que los identificadores únicos que se les asigna a los objetos de una tabla puedan ser referenciados desde los atributos de otros objetos distintos o desde la columna de una tabla.

Hablamos del tipo denominado **REF,** cuyo atributo guardará una referencia (un enlace) a un objeto de la tipología definida y genera una relación entre ambos objetos.

Este tipo de referencias se usarán para acceder a los objetos relacionados y actualizarlos, pero no es posible realizar operaciones directamente sobre las referencias. Para usar una referencia o actualizarla, usaremos **REF o NULL.**

Una vez hemos definido una columna de tipo REF, se puede acotar el alcance a los objetos que se guarden en una determinada tabla. A continuación, veremos, en el Código 7, un atributo de tipo REF y que restringe su dominio a determinada tabla.

Fig. 7. Imagen referencias o links.

```
CREATE TABLE clientes_tab OF cliente_t;

CREATE TYPE ordenes_t AS OBJECT (
ordennum NUMBER,
cliente REF clientes_t,
fechapedido DATE,
direntrega direccion_t);

CREATE TABLE ordenes_tab OF ordenes_t (
PRIMARY KEY (ordennum),
SCOPE FOR (cliente) IS clientes_tab);
```

Código 7. Referencias.

Tal y como podemos observar, al inicio se crea una tabla de tipo *cliente_t*. A continuación, se crea un tipo de objeto llamado *ordenes_t* con 4 atributos. El segundo de ellos será el objeto que será referenciado.

Por último, tenemos la creación de la tabla *ordenes_tab*, donde define una *primary key* y el segundo dato será, precisamente, esa referencia que hemos definido en el objeto anterior.

/ 10. Herencia de tipos

La herencia de tipos nos permite crear jerarquías de tipos.

Una jerarquía de tipos es una serie de niveles sucesivos de subtipos, cada vez más especializados, que derivan de un tipo de objeto ancestro común, denominado supertipo. Esto, como se puede observar, no es un concepto nuevo, ya que, en programación orientada a objetos, lo usamos muy frecuentemente, sobre todo en lenguaje Java.

Los subtipos derivados heredarán las características del tipo de objeto principal y pueden ampliar la definición de este.

Los tipos especializados pueden añadir nuevos atributos o métodos, o redefinir métodos heredados de la clase tipo padre. La jerarquía del tipo resultante facilita un nivel superior de abstracción para manejar la complejidad de un modelo de una aplicación

A continuación, mostraremos un pequeño esquema donde podremos ver la herencia entre objetos de tipo usuario. Podremos observar cómo partimos de una clase principal e iremos heredando atributos, al mismo tiempo que los hijos podrán ir aportando nuevas características a los de los objetos padre:

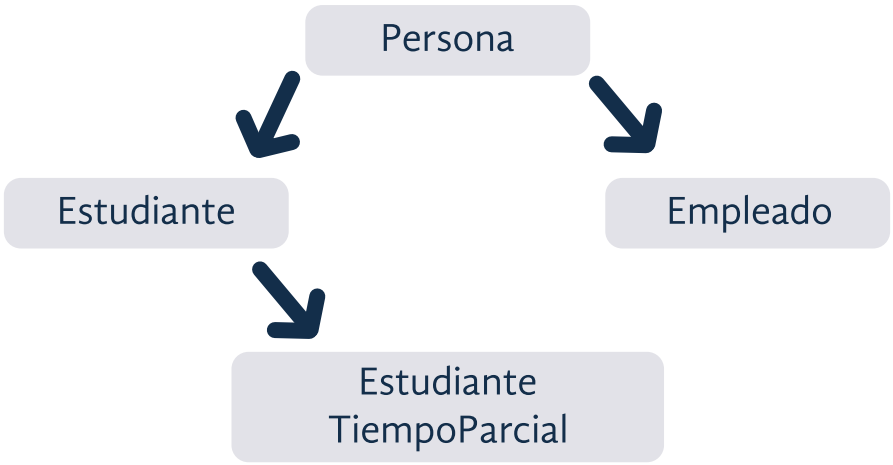

Fig. 8. Esquema herencia.

Podemos observar que el tipo «Persona» poseerá una serie de atributos que serán heredados tanto por el tipo «Estudiante» como por el tipo «Empleado».

«Estudiante» y «Empleado», a su vez, agregarán, en caminos diferentes, nuevos atributos para los tipos hijos que se puedan crear. En este caso, un hijo de «Estudiante» es «EstudianteTiempoParcial» que, finalmente, heredará atributos de su padre «Estudiante» y del padre de su padre, «Persona».

/ 11. Caso práctico 2: "Herencia de tipos"

Planteamiento: Antes del desarrollo y de la creación de varias tablas nuevas en un sistema objeto relacional Oracle, debe hacerse un análisis previo sobre las clases o tipos que se poseen y sobre su jerarquía, teniendo en cuenta los siguientes tipos:

Vehículo_t, coche_t, ciclomotor_t, cocheCarreras_t, cochePaseo_t, motocicleta_t, ciclomotorPaseo_t.

Para el análisis, se aconseja desarrollar un esquema teniendo en cuenta la herencia de dichos tipos, con la consideración que podrían heredar sus atributos de padres a hijos.

Nudo: ¿Cómo plantearías el esquema?

Desenlace: Se realiza un esquema de herencia de arriba hacia abajo, situando el tipo padre arriba.

A continuación, se muestra el esquema de tipos con herencia. La relación de tipos padre a hijos sería la siguiente:

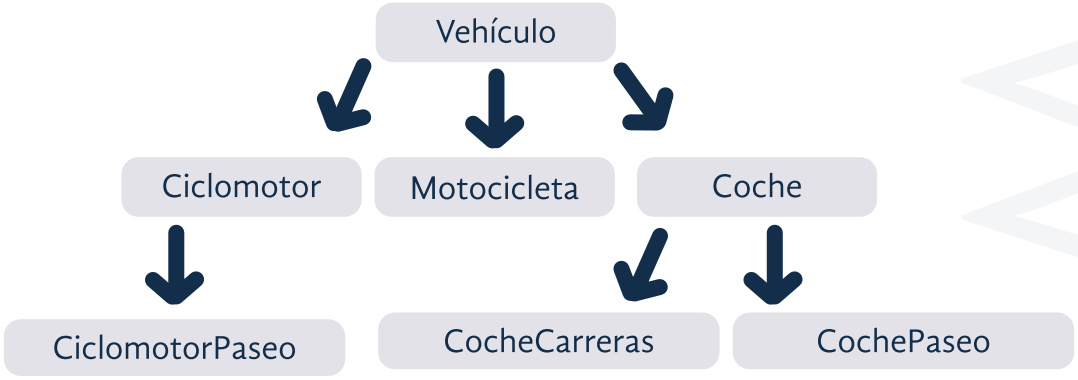

Fig. 9. Esquema herencia de tipos.

/ 12. Resumen y resolución del caso práctico de la unidad

Tras haber estudiado la unidad didáctica, podemos afirmar que tenemos una visión global del **concepto de bases de datos objeto-relacionales**. La idea de coexistir con datos tanto de tipo relacional como de tipo objeto.

En esta unidad, nos hemos centrado en el concepto de base de datos objeto-relacional con el ejemplo de la conocida **base de datos Oracle**. Hemos aprendido los diferentes tipos de objetos que podemos tener en una base de datos de este tipo, además de algunos tipos de colecciones que podemos encontrar como novedad en esta base de datos, como los **VARRAY o las tablas anidadas**.

Por último, hemos visto cómo realizar **referencias** a objetos y la posibilidad de anidar objetos con **herencia** en nuestra base de datos objeto-relacional.

Resolución del caso práctico de la unidad.

La resolución del caso práctico planteado, requiere de la creación de, básicamente, dos elementos:

- Un objeto nuevo de tipo usuario
- Creación de una tabla
- Para la creación del objeto, tendremos lo siguiente:

```
CREATE TYPE vehiculo_typ AS OBJECT (
 nombre    VARCHAR2(30),
 marca REF vehiculo_typ);
```

Código 8. Creación del objeto.

En este código, podemos ver cómo creamos un nuevo *type*, que se denominará *vehiculo_typ* y que tendrá 2 atributos, como indica el ejercicio: nombre (*Varchar*) y marca (será una referencia al propio objeto).

```
CREATE TABLE vehiculo_tabla OF vehiculo_typ;
```

Código 9. Definición tabla.

De esta forma, estaremos definiendo una tabla cuyas filas serán objetos del tipo anteriormente definido.

/ 13. Bibliografía

Bertino, E. & Martino, L. (1993). *Object-Oriented Database Systems. Concepts and Architectures.* Addison-Wesley.

/ 14. Webgrafía

https://docs.oracle.com/