6.1. Características de las bases de datos orientadas a objetos

Índice

Objetivos 3
Modelos de bases de datos 4
Bases de datos relacionales 4
Bases de datos orientadas a objetos 5
Bases de datos objeto-relacionales 7
Despedida 8
Resumen 8

Objetivos

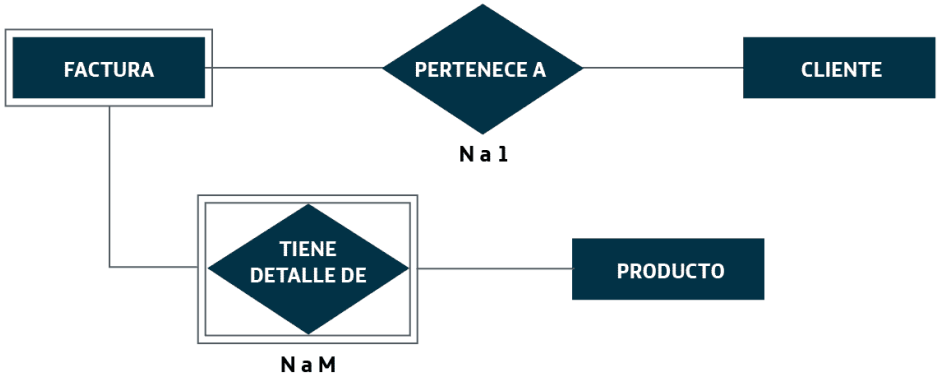
En esta lección perseguimos los siguientes objetivos:

- Conocer los fundamentos de las bases de datos orientadas a objetos.
- Distinguir entre base de datos relacional, base de datos orientada a objetos y base de datos objeto relacional.

Modelos de bases de datos

Bases de datos relacionales

Una base de datos que sigue el modelo relacional está formada por un conjunto de tablas o relaciones de datos que están interconectadas entre sí por medio de asociaciones.



Nuestra base de datos *FERRETERIA* tiene una tabla *CLIENTE* interconectada con otra tabla *FACTURA* a través del *NIF*.

Cada tabla está formada por un conjunto de filas, donde cada una de ellas almacena información sobre una determinada entidad, por ejemplo: un producto. Cada fila está compuesta por atributos cuyos valores sólo pueden ser de tipo elemental o atómico, es decir, que no podrá descomponerse en varios valores.

En las bases de datos relacionales, la información necesaria para elaborar un documento debe ser extraída normalmente de varios registros de varias tablas diferentes. Por ejemplo: para elaborar una factura, será necesario extraer los datos del cliente de la tabla *CLIENTE*, los datos factura de la tabla *FACTURA*, las ventas de la tabla *DETALLE* y las descripciones de los artículos de la tabla *PRODUCTO*. **Este detalle es lo que diferencia a las bases de datos relacionales de las bases de datos documentales,** en las que la información necesaria para confeccionar un documento está almacenada en una única unidad o registro.

En los siguientes apartados podrás comparar el modelo relacional de base de datos con el modelo orientado a objetos y el modelo objeto-relacional.

Bases de datos orientadas a objetos

Un sistema de gestión de bases de datos orientado a objetos no es más que un DBMS que soporta el paradigma de orientación a objetos.

En la primera unidad de esta módulo, aprendiste a guardar objetos en disco. En concreto, creaste un objeto *Alumno* con su colección de objetos *Calificacion* y lo guardaste en un fichero. También construiste una pequeña agenda telefónica como un archivo que almacena objetos *Contacto*. Aquello fue la base para comprender el funcionamiento de los sistemas de gestión de bases de datos orientadas a objetos.

Cuando hablamos de gestores de bases de datos orientadas a objetos, hablamos de herramientas cuya principal labor consiste en facilitar la **persistencia de objetos**, es decir, guardar colecciones de objetos en un dispositivo de almacenamiento con el fin de recuperarlos cuando sea necesario. Recuerda que para que un objeto tenga la capacidad de persistencia, debe pertenecer a una clase que implemente la interfaz *Serializable*.

Te recomendamos que repases la lección **1.2. Lectura/Escritura de objetos** de la primera Unidad Formativa de esta asignatura.

Los DBMS orientados a objetos proveen de un lenguaje basado en SQL, pero que permite la recuperación del subconjunto de objetos que cumplan un criterio dado dentro de la colección completa de objetos guardados en la base de datos. Este sublenguaje se denomina **OQL *(Object Query Language)***.

Veamos un ejemplo de consulta en SQL y su correspondencia en OQL:

1. SQL:

SELECT descripcion, precio FROM Producto
WHERE codigo = 'TOR7';

En este caso, el DBMS nos ha devuelve una fila o registro compuesto por los atributos atómicos *descripcion* y *precio*, que no podrán descomponerse en nada más.

2. OQL:

SELECT pr.descripcion, pr.precio FROM pr IN Producto
WHERE pr.codigo = 'TOR7';

En este caso, el DBMS nos devuelve un objeto de la clase *Producto*, cuya referencia es *pr* y cuyos atributos *descripcion* y *precio* pueden ser atómicos pero también podrían ser objetos que se descomponen en otros atributos (composición).

3. OQL con un atributo que se descompone:

SELECT cl.nombre, cl.direccion, cl.telefono
FROM cl IN Cliente
WHERE cl.direccion.codigoPostal = '28017';

En este otro ejemplo estamos recuperando una colección de objetos *Cliente* cuyo *codigoPostal* tenga el valor '*28017*'. Cada objeto *Cliente* está compuesto por *nombre*, *direccion* y *telefono*, pero la *direccion*, a su vez, es otro objeto compuesto por *calle*, *numero*, *piso*, *puerta*, *codigoPostal*, etc.

Tres ejemplos de ODBMS (Object Data Base Management System):

ZOPE OBJECT DATABASE

Se trata de un ODBMS (*Object Data Base Management System*) creado por Zope Corporation a finales de los 90. Permite el almacenamiento y recuperación de objetos en lenguaje Python. Incluye transacciones, historial y deshacer, concurrencia y escalabilidad.

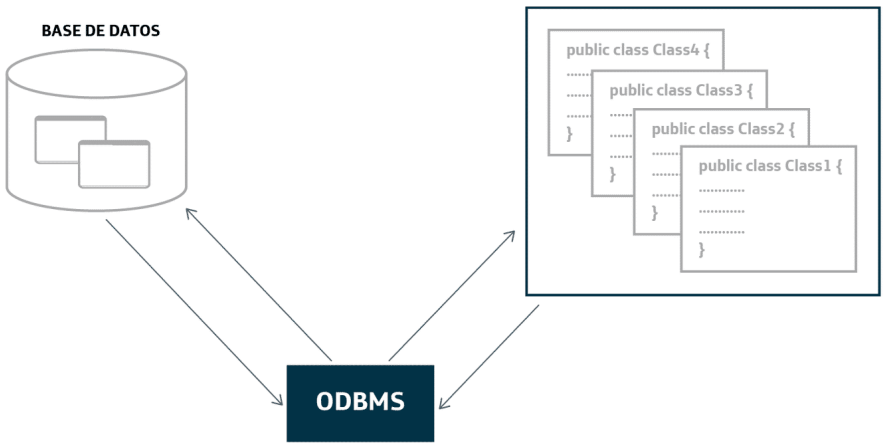
OBJECTDB

Se trata de un ODBMS muy orientado para interactuar con programas Java. Sin embargo, no dispone de API propietaria, por lo que es necesario utilizar alguna API de las estándar de Java, como, por ejemplo, JPA.

NEODATIS

Se trata de un ODBMS muy simple, escrito en Java y que cuenta con su propia API propietaria que puede ser importada en cualquier aplicación Java.

Éste podría ser el esquema de funcionamiento de un ODBMS:



La base de datos almacena objetos y el ODBMS se encarga de su administración y de suministrar los objetos requeridos por los programas que interactúan con él.

Las bases de datos orientadas a objetos pertenecen a la categoría de **bases de datos documentales,** donde toda la información necesaria para confeccionar un documento, como una factura, se encuentra disponible en una única unidad o registro; en este caso, en un único objeto, aunque luego dicho objeto nos da acceso a otros objetos dentro de su jerarquía.

Bases de datos objeto-relacionales

Los DBMS objeto-relacionales se quedan a medio camino entre el modelo relacional y el modelo orientado a objetos, utilizando lo mejor de ambos.

Las bases de datos objeto-relacionales siguen utilizando la estructura relacional basada en tablas con filas y columnas, donde cada tabla tiene asociaciones con otras tablas. La diferencia radica en que cada atributo o campo dentro de un registro o fila puede ser, bien atómico (de tipo elemental), bien un tipo de dato más complejo.

Cada atributo puede ser de uno de estos tipos:

- **Tipo de dato elemental** (*INT*, *CHAR*, *FLOAT*, etc.).
- **Tipo estructurado** formado por más atributos. Por ejemplo: dirección compuesta por calle, número, piso, etc.
- **Objetos de gran tamaño**: canciones, imágenes, vídeos, etc.

Además, los DBMS objeto-relacionales poseen características de la programación orientada a objetos, como, por ejemplo, la definición de tipos personalizados (clases) y la herencia.

Dada la declaración del tipo estructurado *Persona*:

```
CREATE TYPE Persona
 (nombre VARCHAR(20), apellidos VARCHAR(20), tlf VARCHAR(10))
```

Podemos crear otro tipo que herede de *Persona*:

```
CREATE TYPE Alumno UNDER Persona (curso VARCHAR(20));
```

El tipo estructurado *Alumno* dispone del atributo *curso*, además de los atributos *nombre*, *apellidos* y *tlf*, que ha heredado de *Persona*.

Sin duda, Oracle Database es actualmente el gestor de bases de datos objeto-relacionales más importante.

Para saber más
Consulta la entrada de la wikipedia sobre Oracle Database.
https://es.wikipedia.org/wiki/Wikipedia:Comunicado_4_julio_2018

Despedida

Resumen

Has terminado la lección, vamos a ver los puntos más importantes que hemos tratado.

- Una base de datos que sigue el **modelo relacional** está formada por un conjunto de tablas o relaciones de datos, que están interconectadas entre sí por medio de asociaciones. Cada tabla está formada por un conjunto de filas, donde cada una de ellas almacena información sobre una determinada entidad, por ejemplo, un producto. Cada fila está compuesta por atributos cuyos valores sólo pueden ser de tipo elemental o atómico, es decir, que no podrá descomponerse en varios valores.
- Un sistema de gestión de **bases de datos orientado a objetos** no es más que un DBMS que soporta el paradigma de orientación a objetos. Son herramientas cuya principal labor consiste en facilitar la persistencia de objetos, es decir, guardar colecciones de objetos en un dispositivo de almacenamiento con el fin de recuperarlos cuando sea necesario. Además, proveen de un lenguaje basado en SQL, pero que permite la recuperación del subconjunto de objetos que cumplan un criterio dado, dentro de la colección completa de objetos guardados en la base de datos. Este sublenguaje se denomina OQL *(Object Query Language).*
- Los **DBMS objeto-relacionales** se quedan en a medio camino entre el modelo relacional y el modelo orientado a objetos, utilizando lo mejor de ambos. Las bases de datos objeto-relacionales siguen utilizando la estructura relacional basada en tablas con filas y columnas, donde cada tabla tiene asociaciones con otras tablas. La diferencia radica en que cada atributo o campo dentro de un registro o fila puede ser atómico (de tipo elemental), o bien un tipo de dato más complejo.