UT3 - Conexión a Bases de Datos Relacionales

Luis Javier López López

Índice

1. Modelo Relacional vs Modelo de Objetos

Modelo Relacional BBDD

Es el modelo que usan la mayoría de sistemas de bases de datos (MySQL, PostgreSQL, Oracle, etc.).

- Datos organizados en tablas (relaciones).
- Cada tabla tiene filas (tuplas) y columnas (atributos).
- Se identifican registros con claves primarias.
- Las relaciones entre tablas se representan con claves foráneas.

```sql
CREATE TABLE Alumno (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    edad INT,
    id_curso INT,
    FOREIGN KEY (id_curso) REFERENCES Curso(id)
);
```

1. Modelo Relacional vs Modelo de Objetos

Modelo de Objetos (POO)

Es el modelo que usan los lenguajes de programación orientados a objetos (Java, C#, Python, etc.).

- Datos representados con clases y objetos.
- Cada clase define atributos (propiedades) y métodos (funciones).
- Se puede usar herencia, composición y polimorfismo.

```
class Alumno {
    private int id;
    private String nombre;
    private int edad;
    private Curso curso;  // relación con otra clase
}
```

1. Modelo Relacional vs Modelo de Objetos

Aspecto	Modelo Relacional	Modelo de Objetos
Estructura	Tablas, filas, columnas	Clases y objetos
Relaciones	Claves primarias/foráneas	Referencias, asociaciones
Herencia	No existe directamente	Sí existe (extends, superclases)
Tipos de datos	Limitados (INT, VARCHAR, DATE…)	Más ricos (objetos, listas, colecciones, enums)
Operaciones	SQL (SELECT, INSERT, UPDATE, DELETE)	Métodos de las clases

1. Modelo Relacional vs Modelo de Objetos

EL PROBLEMA

Cuando programamos en Java (modelo de objetos) y guardamos en MySQL (modelo relacional), no encajan 100% bien:

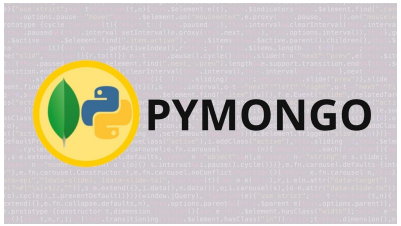
- ¿Cómo guardar un atributo que es una lista de objetos?
- ¿Cómo representar la herencia de clases en tablas?
- ¿Cómo transformar automáticamente un ResultSet en un objeto Java?

Ese choque es lo que llamamos **desfase objeto-relacional**, y de ahí nacen los conectores (JDBC) y los ORM (Hibernate, JPA) que ayudan a “traducir” entre los dos mundos.

2. Conexión de aplicaciones con BBDD

Estrategias de Conexión

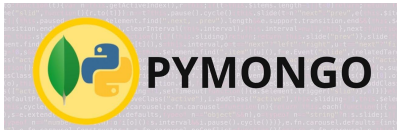
- **Conectores (JDBC, mysqlconnectorpython, pymongo)**: acceso directo, control total, más código.
- **ORM (Hibernate, JPA)**: menos código, abstracción, curva de aprendizaje.



2. Conexión de aplicaciones con BBDD

Conexión directa con conectores

- Se usan **drivers/conectores** que implementan el estándar JDBC en Java.
- El desarrollador escribe **SQL directamente** en el código.
- Tú controlas:
  - Conexión (***DriverManager.getConnection***)
  - Creación de sentencias (***Statement, PreparedStatement***)
  - Ejecución de consultas (***executeQuery, executeUpdate***)
  - Lectura de resultados (***ResultSet***)



2. Conexión de aplicaciones con BBDD

Conexión directa

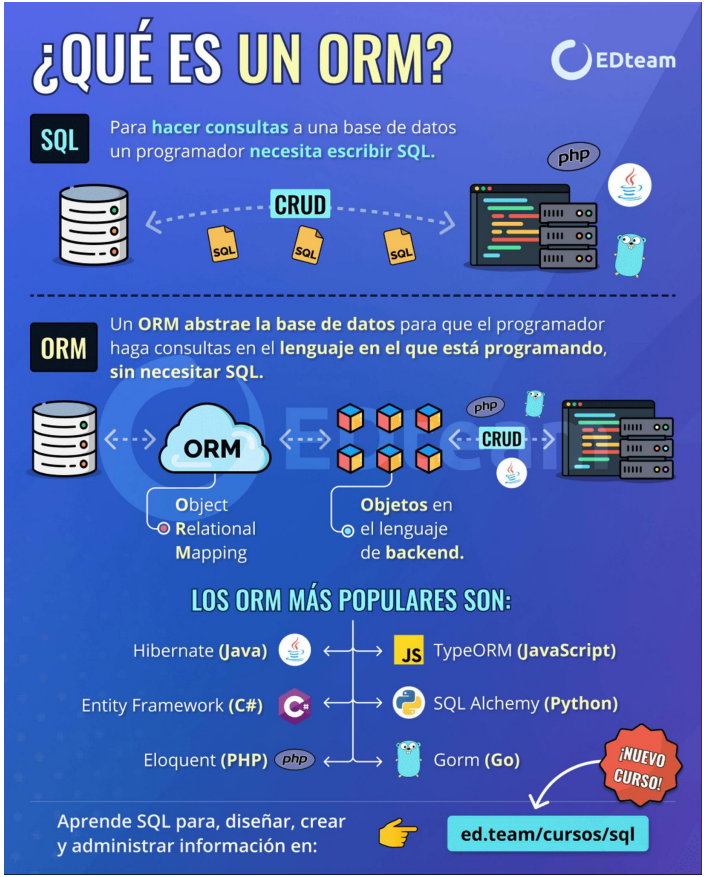
con conectores

- Control total sobre el SQL.
- Más eficiente si el programador sabe optimizar.
- Más transparente: “ves” exactamente lo que hace tu app.

- Mucho código repetitivo (abrir conexión, cerrar, tratar errores…).
- Riesgo de errores de seguridad (ej. SQL Injection si no se usan parámetros).
- Mayor dificultad en proyectos grandes.

Conexión con ORM

- En lugar de trabajar con tablas/SQL, trabajas con objetos.
- El ORM traduce objetos ↔ registros automáticamente.
- Ejemplos en Java: Hibernate, JPA, EclipseLink.



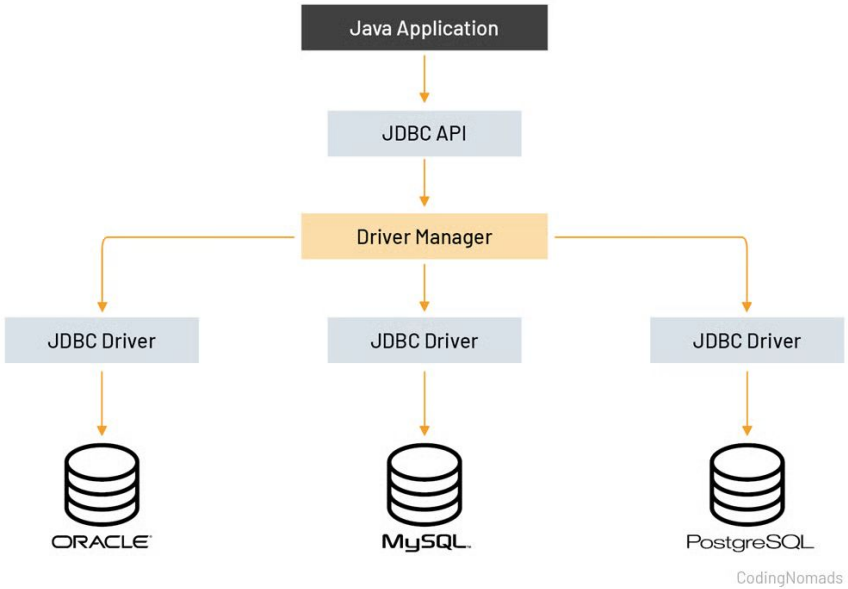
2. Conexión de aplicaciones con BBDD

Conexión ORM

- Menos código SQL “manual”.
- Más rápido de desarrollar.
- Compatible con herencia y relaciones entre clases.

- Menos control sobre las consultas (rendimiento puede ser peor).
- Curva de aprendizaje más alta.
- Debugging más complicado (traducción ORM ↔ SQL).

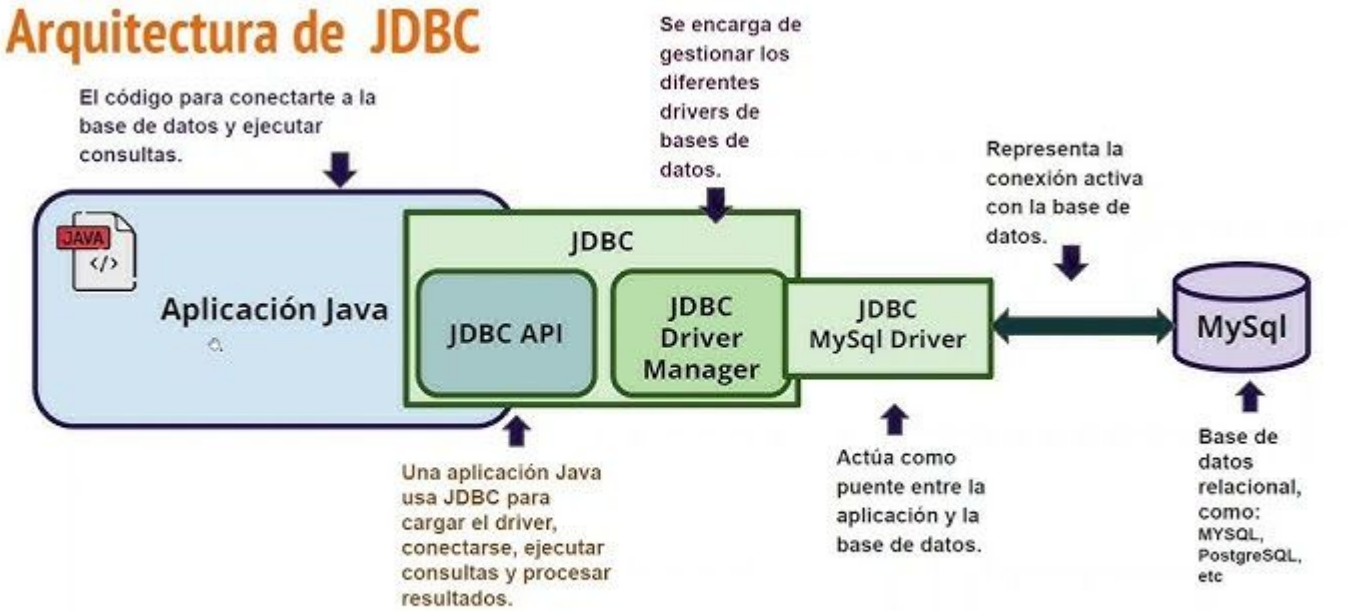
3. Uso de conectores (JDBC)



¿Qué es JDBC?

- **JDBC (*Java Database Connector*)** es una API de Java que permite a los programas **conectarse y trabajar con bases de datos relacionales.**
- Funciona mediante **drivers** que traducen las llamadas de Java a instrucciones específicas del gestor de base de datos (MySQL, PostgreSQL, Oracle, etc.).

4. Desarrollo de aplicaciones con conexión a bases de datos



ACTIVIDAD

Objetivos

- *Aprender a crear un proyecto Java con Maven.*
- *Incorporar dependencias necesarias (driver JDBC de MySQL).*
- *Programar una clase de conexión a la base de datos.*
- *Ejecutar operaciones básicas con JDBC.*
- *Reforzar la teoría de JDBC y su ciclo de uso.*

ACTIVIDAD

UT3

Práctica 1

Proyecto

Java/Maven/JDBC