JDBC

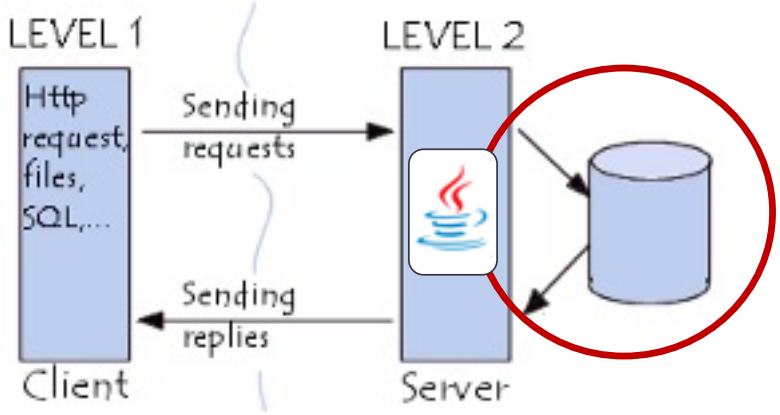
ARQUITECTURAS WEB (2023)

J. ANDRES DIAZ PACE

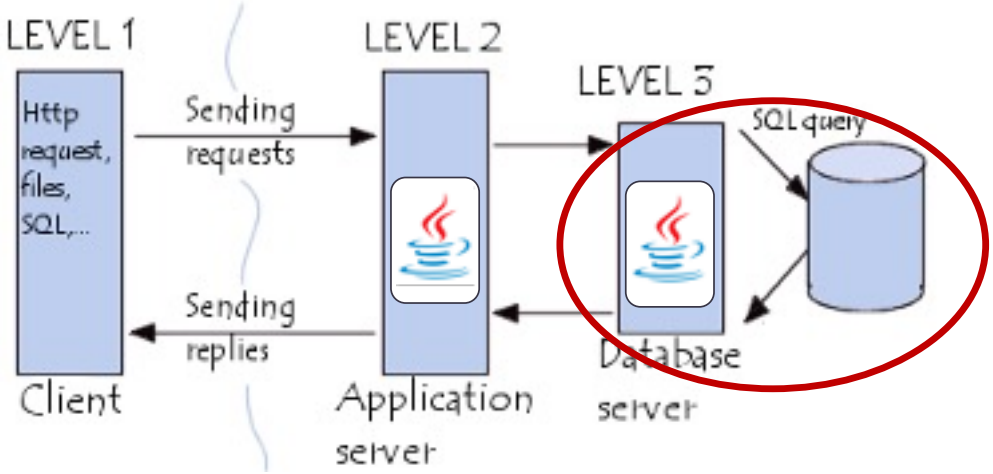
Introducción a Bases de Datos

Arquitecturas Cliente-Servidor:

- El backend a menudo se implementa en Java
- E involucra algún tipo de persistencia de datos



Arquitectura 2-tier



Arquitectura 3-tier

Repaso de Bases de Datos relacionales

- Se popularizaron comercialmente en los años 80
- Basadas en un modelo matemático para describir la estructura de los datos
- Acompañado por un **lenguaje de consulta (SQL) declarativo** y estandarizado, que es mayormente independiente de la representación de los datos

Ejemplo: Venta de auto-partes

- Consideremos un negocio que vende distintos repuestos de autos
  - Depósitos
  - Inventarios de repuestos
  - Ordenes (de clientes)
  - …

Warehouse	Warehouse Address	Part

1

Warehouse 1	123 Main Street	Transmission, Steering wheel, Brake pads, …

2

Warehouse 1	123 Main Street	Transmission
Warehouse 1	123 Main Street	Steering wheel
Warehouse 1	123 Main Street	Brake Pads

¿Cuál es el problema?

Modelado normalizado de entidades

- Aplicando 2da. y 3ra. forma normal

Warehouse Table:

Warehouse ID	Warehouse Address

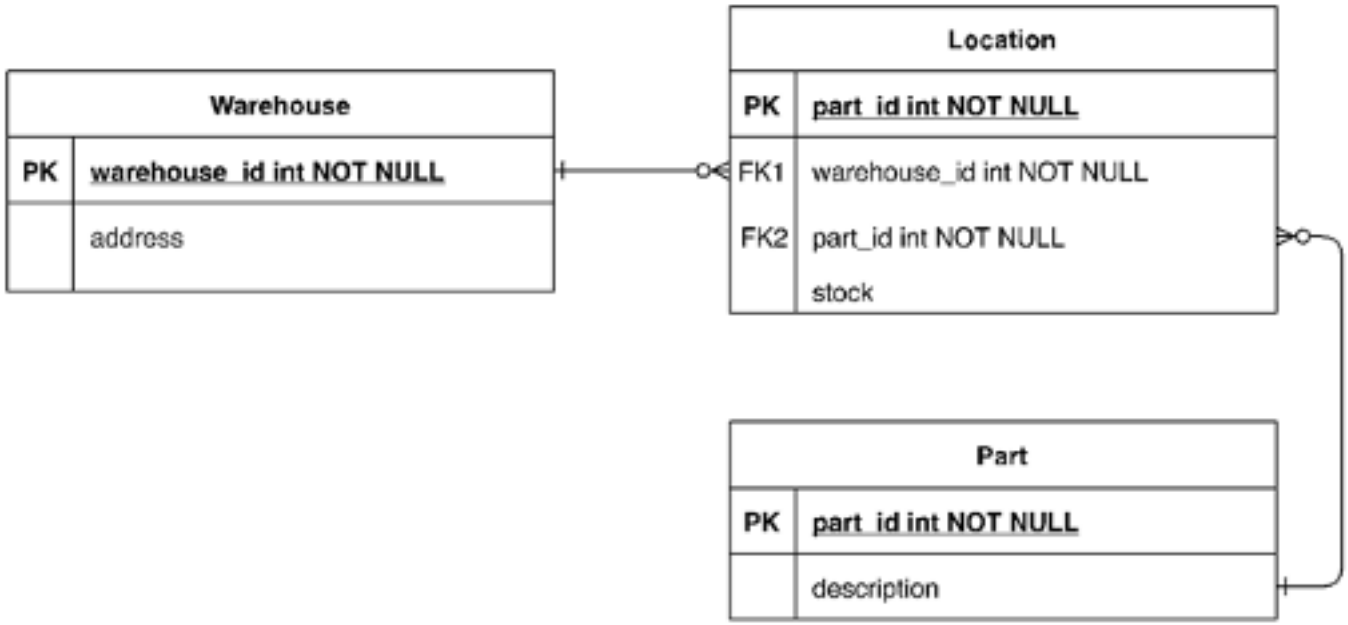
Parts Table:

Part ID	Part Description

Relations Table:

Warehouse ID	Part ID

Diagrama de Entidad-Relación (DER o ERD)



SQL

- Permite extraer datos de las tablas mediante consultas declarativas
  - Lectura, escritura, actualización y eliminación de datos
  - En algunos casos, debe combinarse información de más de una tabla

```
SELECT * FROM location WHERE stock>10;

UPDATE warehouse SET address = '...'
WHERE id = 44;

SELECT warehouse.id, warehouse.address
FROM warehouse
INNER JOIN location ON warehouse.id = location.id …;
```

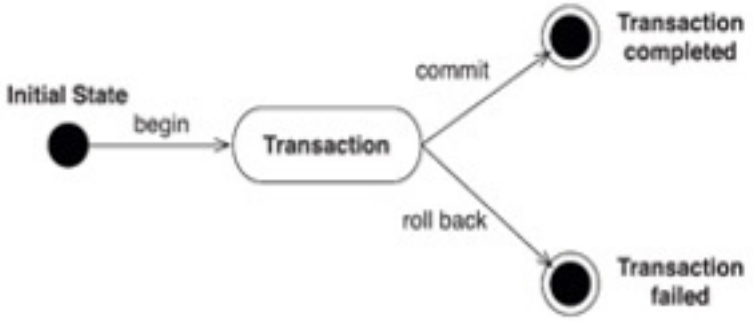
Conexión

- El medio a través del cuál se conectan servers (o clientes) a una instancia de base de datos
  - Es independiente de si están en la misma máquina o en máquinas separadas
  - El servidor envía comandos a la base de datos
  - Normalmente, la base admite un número máximo de conexiones
  - La conexión se establece con un **driver** y requiere **credenciales de autenticación**

- **Buena práctica**: abrir una conexión solo cuando es necesario, y liberarla lo antes posible

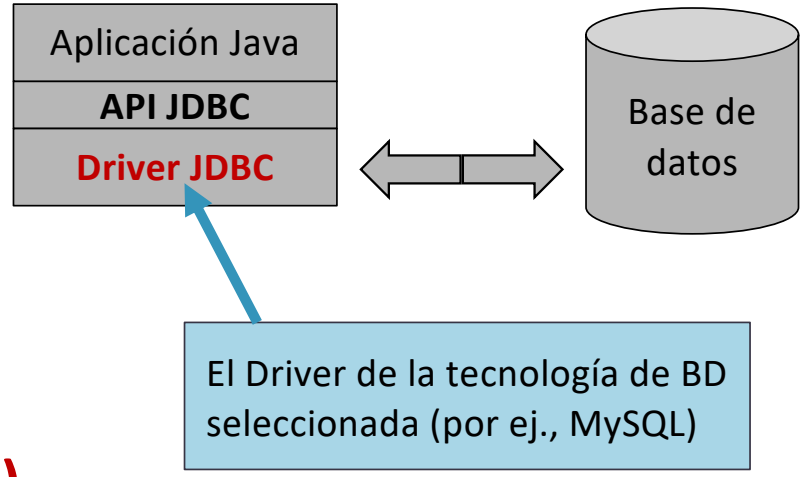
Transacciones

- Es una “unidad de trabajo” que se trata “como un todo”
- Tiene éxito sobre la base de datos, o bien falla
  (y se deben deshacer los resultados parciales)
- Propiedades **ACID**
  - Atomic: Si se hace commit de
    un cambio, esté se efectúa “completamente”
  - Consistent: El cambio solo se produce si la BD
    queda en un estado consistente
  - Isolated: Nadie ve partes de la transacción hasta que se hizo el commit
  - Durable: Una vez que el cambio se efectuó, el cambio permanece en la BD
- Una secuencia de operaciones SQL
- Depende de contar con una conexión abierta



¿Cómo lo hago desde Java?

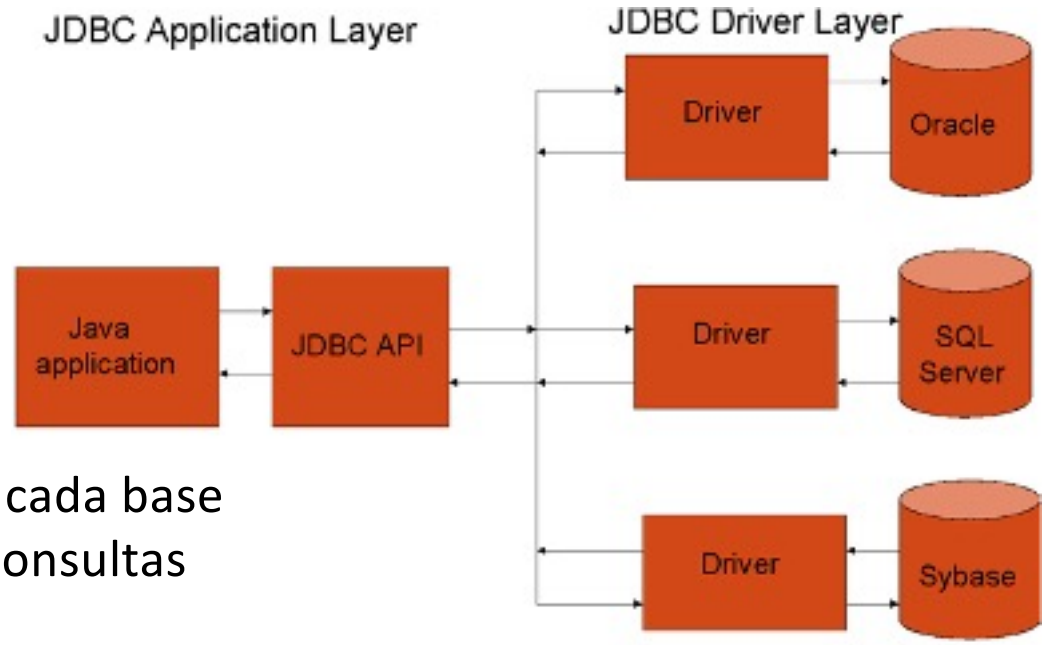
- Necesito poder establecer una conexión a la base de datos elegida



- Es necesario realizar una **traducción (o mapeo) entre el modelo de clases/objetos de mi aplicación y las tablas** (y sus relaciones) en el esquema de la base de datos

Arquitectura JDBC

- Es un conjunto de clases que permite a las aplicaciones Java comunicarse con distintas tecnologías de base de datos
  - El mecanismo de intermediación es una fachada (API) que se comunica con un driver específico para cada base
  - Permite transformar resultados de las consultas en ciertos objetos Java



- **NO es un mapeo objeto-relacional**
  - La traducción de POO a tablas queda a cargo del programador

Principales clases de JDBC

- Paquetes *java.sql y javax.sql*
- Pasos:
  1. Cargar el driver correspondiente
  2. Conectarse a la base de datos
  3. Crear y ejecutar sentencias SQL
  4. Gestionar excepciones SQL

- Algunos **inconvenientes**
  - Código Java y SQL mezclados
  - Actualizar manualmente el esquema de la base de datos
  - Dependencia de la sintaxis SQL con cada base de datos
  - Lidiar con aspectos de bajo nivel: conexiones, transacciones, etc.

- **DriverManager class** : Loads the driver for the database.
- **Driver (interface / Class)** : Represents a database driver. All JDBC driver classes must implement the Driver interface.
- **Connection interface** : Enables you to establish a connection between a Java application and a database.
- **Statement interface** : Enables you to execute SQL statements
- **ResultSet interface:** Represents the information retrieved from a database.
- **SQLException class:** Provides information about the exceptions that occur while interacting with databases.

Ejemplos en código: JDBC

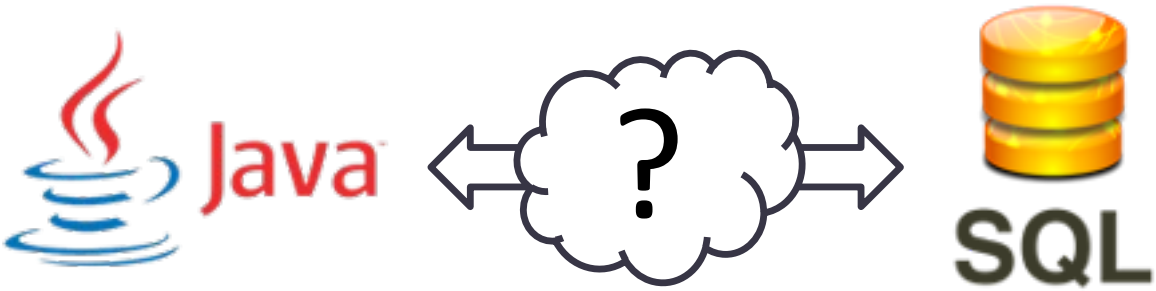
- Para MySQL, podemos también levantar una BD con una imagen Docker


```
version: "3"
services:
  db:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      - "MYSQL_ROOT_PASSWORD=password"
    ports:
      - "3306:3306"
    volumes:
      - "/Users/adiazpace/Documents/mysql:/var/lib/mysql"
```

docker-compose -f msql.yml up

¿Qué desean los desarrolladores?

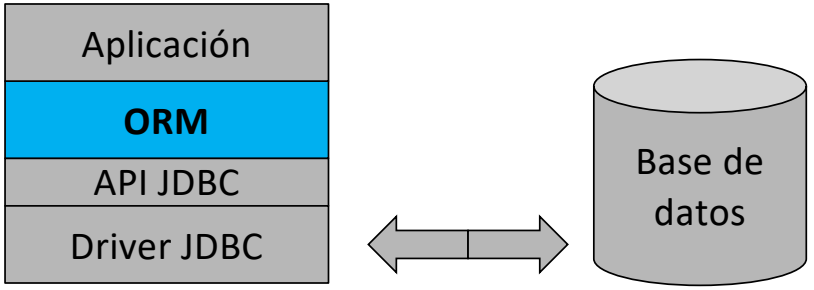
- El modelo de datos no debe restringir el modelo de objetos
- Acceso rápido a los datos, sin lidiar con aspectos de bajo nivel
- Separar código SQL de código Java
- Consultas más basadas en objetos que en SQL
- Aislar la aplicación Java de los cambios que pueda haber en los esquemas de la base de datos



Solución: Mapeo Objeto-Relacional (ORM)

- Reglas y técnicas de programación para convertir, de manera transparente, elementos entre el modelo de objetos de una aplicación y el esquema relacional de la base de datos

- El desarrollador ...
  - Trabaja con una “base de datos orientada a objetos”
  - Se desliga de los aspectos de bajo nivel y la tecnología subyacente de base de datos



Preguntas