Bases de Datos

Las bases de datos relacionales permiten organizar los datos en tablas, las cuales se relacionan mediante campos clave. Para poder realizar consultas en estas bases de datos utilizamos el lenguaje SQL.

¿Qué es una Base de Datos?

Es una representación de datos mediante tablas.

Una tabla es una serie de filas y columnas,	
cada fila es un registro	y cada columna campo

Un campo representa un dato de los elementos almacenados y cada registro es un elemento de la tabla.

La clave primaria es aquel campo que no puede repetirse en la tabla con el mismo registro.

JDBC (Java DataBase Connectivity) permite acceder a bases de datos relacionales cuando programamos con Java, utilizando SQL.

La impedancia Objeto-Relacional es el conjunto de dificultades técnicas que surgen cuando una base de datos relacional se usa en conjunto con un programa escrito con POO.

JDBC

JDBC es una API Java que hace posible ejecutar sentencias SQL, mediante un conjunto de clases e interfaces escritas en Java.

- ✓ java.sql
- ✓ javax.sql

// Drivers

Un conector o driver suele ser un fichero .jar que contiene una implementación de las interfaces, proporcionado por el fabricante de la base de datos.

⇒ Cómo GESTIONAR TABLAS EN BASE DATOS

Utilizaremos como gestor de base de datos MySQL, a través de sus líneas de comandos, con sentencias SQL.

SQL es un lenguaje no procedimental el cual indica al SGBDR (Sistema Gestor de Bases de Datos Relacional) qué queremos obtener y no cómo hacerlo, y este analiza nuestra orden y si es correcta la ejecuta.

- ✓ DDL = Definen estructuras de datos.
- ✓ DML = Operan con los datos almacenados

```
/*
* Comandos
* SQL
*/
```

```
CREATE TABLE peritos(
    num number(2),
    nombre varchar(15),
    color varchar(10)
);
```


CONEXIÓN

Para establecer una conexión con una base de datos podemos utilizar el método getConnection() de la clase DriverManager, y así recibiendo como parámetro la URL que identifica la base de datos.

La ejecución de este método devuelve un objeto Connection que representa la conexión.
Si no se encontrara el driver adecuado lanzará la excepción SQLException.

```
Class.forName
("com.mysql.
jdbc.Driver");
    //Cargar
      driver
```

Si utilizamos acceso puente JDBC-ODBC no habría que cargar controlador.

```
create database notaria;
use database notaria;

"CREATE TABLE clientes"
+ "(id INT,"
+ " PRIMARY KEY (id),"
+ " nombre VARCHAR(20),)"));
```

Controlamos las excepciones mediante el método getMessage() a través de try/catch.

CONSULTAS

1. Carga el conector
2. Establecer conexión con base de datos.
3. Envía consultas SQL.
4. Liberar los recursos.
5. Gestionar los errores.

Podemos utilizar los siguientes tipos de sentencias:

- ✓ Statement = sencillas.
- ✓ PreparedStatement = parámetros
- ✓ CallableStatement = procedimientos almacenados.

El API JDBC distingue dos tipos de consultas:

- ∅ Consultas: SELECT, método ResultSet executeQuery(St sql)
- ∅ Actualizaciones: INSERT, UPDATE, DELETE y el método executeUpdate(St sql).

Con el método next() del ResultSet() hace que dicho puntero avance al siguiente registro. El método executeQuery() devuelve un objeto, para ello usamos métodos getString() con ResultSet() para obtener valores.

El método executeUpdate() nos retornará el número de registros insertados, actualizados o eliminados.

Cerraremos la conexión y liberamos recursos con el método close().