

# Bases de Datos

# Bases de Datos

Es un conjunto de datos pertenecientes a un mismo contexto, que se almacenan sistemáticamente para ser usados de acuerdo con necesidades específicas.

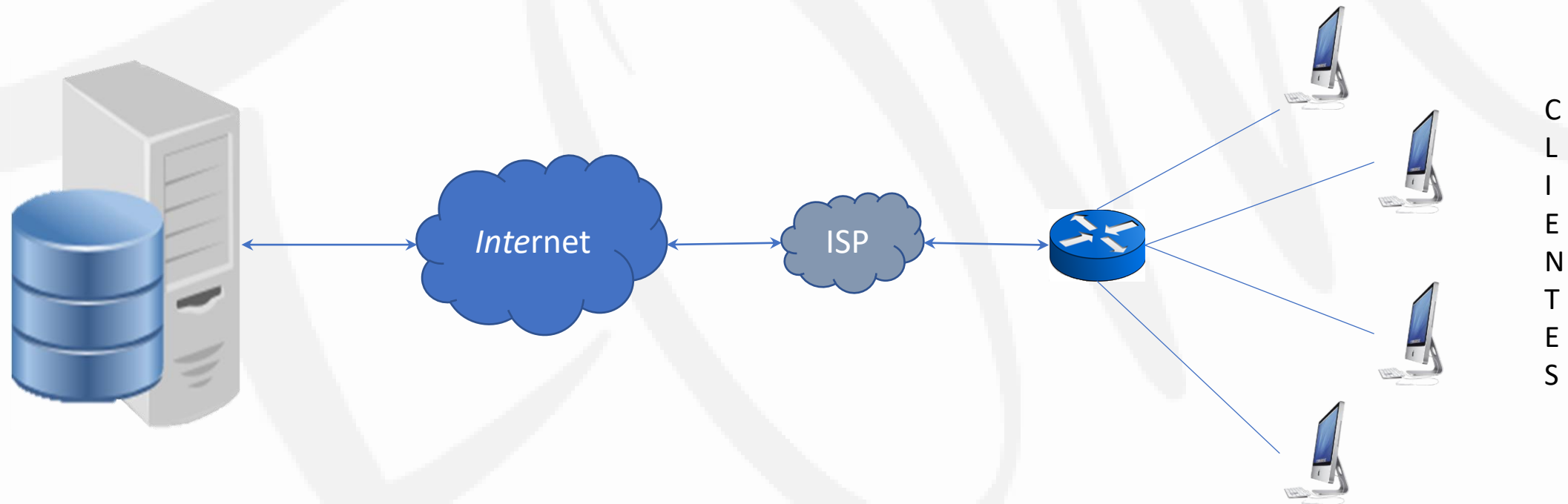
Se componen de tablas\* que almacenan:

- Datos
- Información

# Bases de Datos

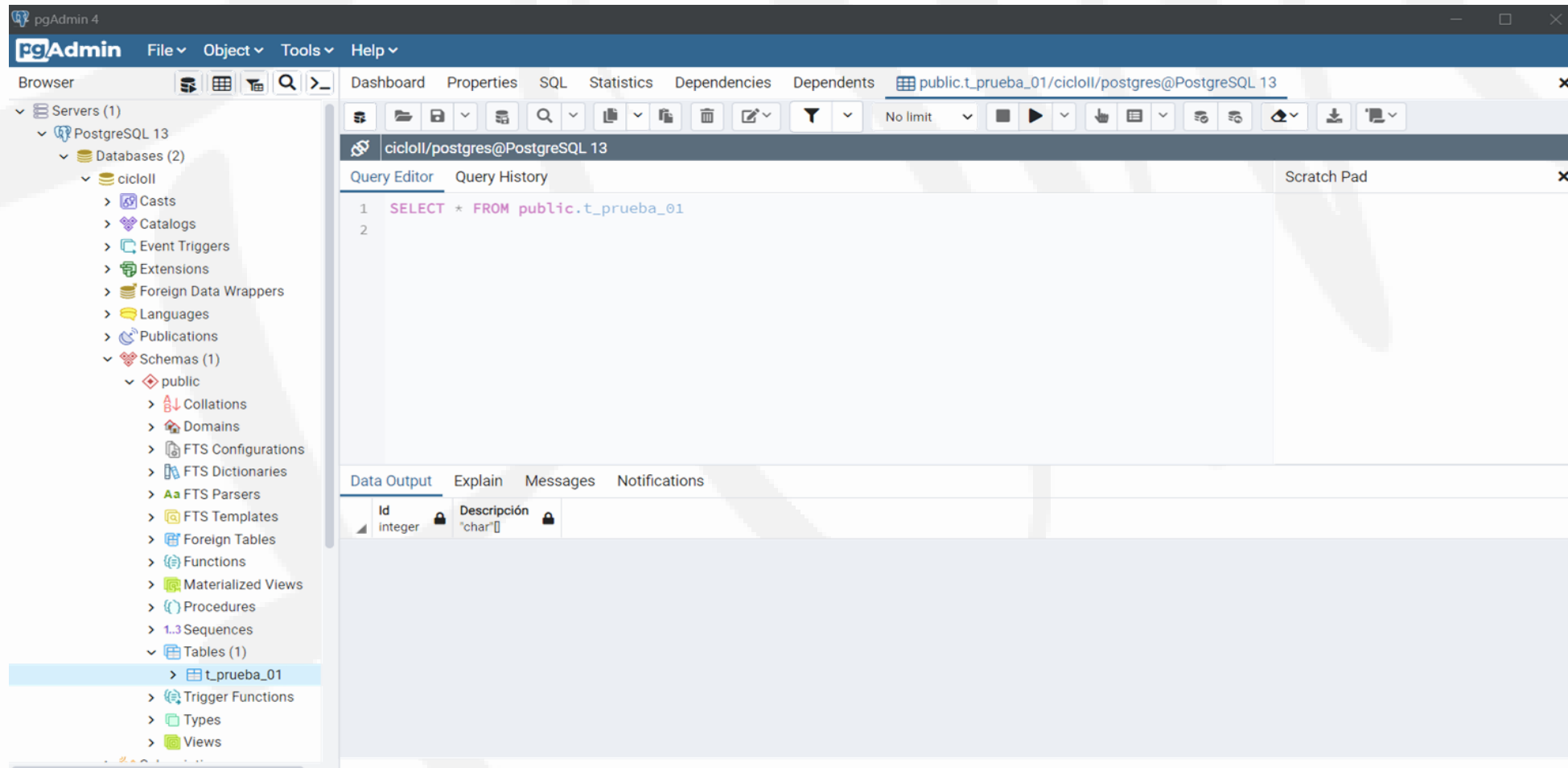
Las bases de datos se administran a través de un Sistema Gestor de Base de Datos (**SGBD - DBMS**) que consiste en un conjunto de programas que permiten el **almacenar**, **modificar** y **extraer** la información almacenada.

# Bases de Datos



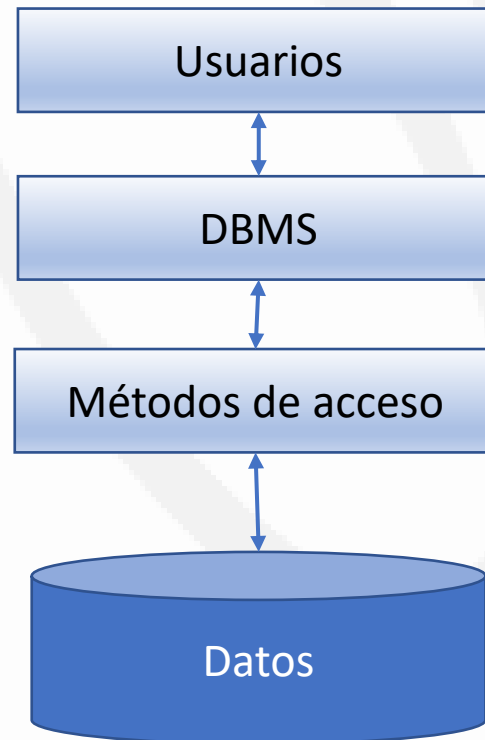
# Bases de Datos

Una **tabla** de datos, es una colección de datos organizados en filas (**tuplas**) y columnas (**atributos**).



# Bases de Datos

La forma de interactuar es:



# Bases de Datos

## Tipos de Bases de Datos:

- Centralizadas o Distribuidas
- Jerárquicas o en Red
- Relacionales o NO relacionales

# Bases de Datos

## Tipos de Base de Datos:

- Orientadas a Objetos
- Orientadas a Grafos
- Documentales



# Bases de Datos

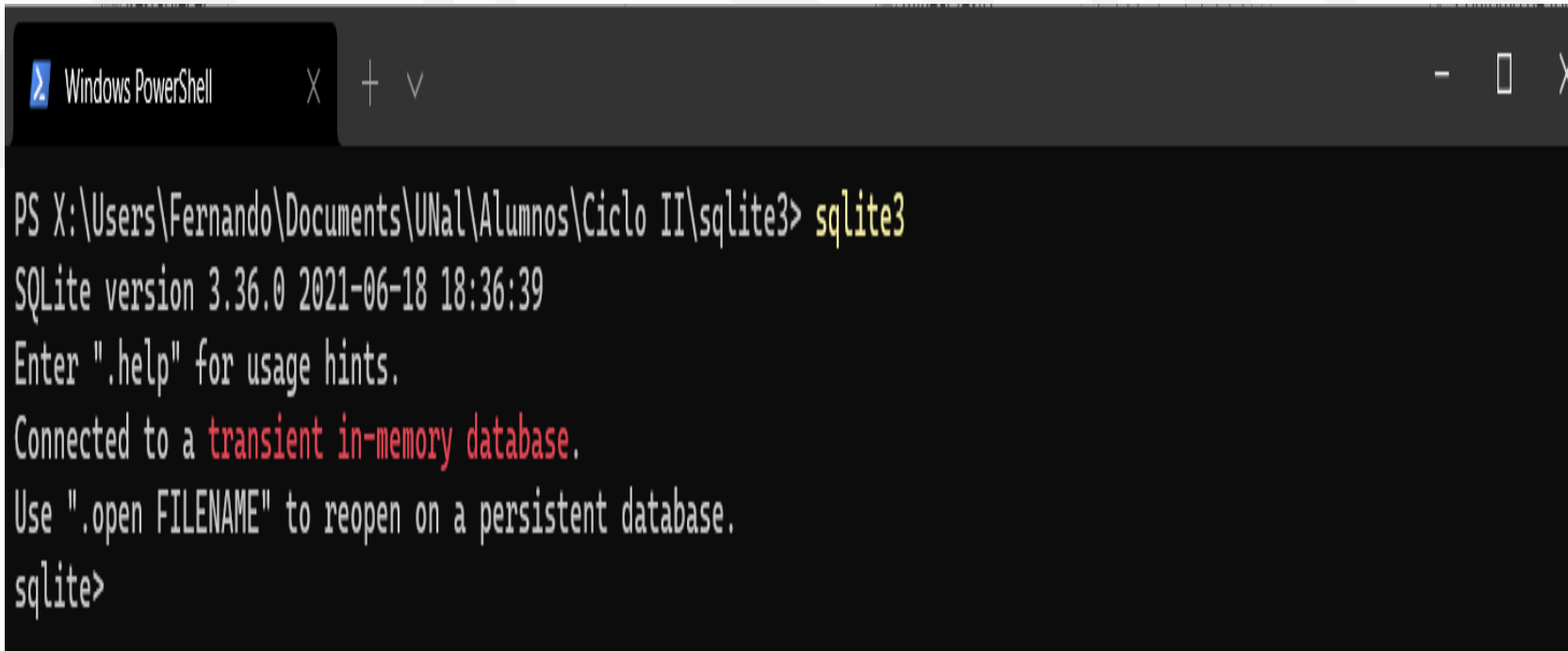
## Lenguaje de Bases de Datos:

- SQL (Structured Query Language)

 En un **lenguaje estándar** para acceder y manipular una base de datos.

# Bases de Datos

## SQL ejemplos:



```
Windows PowerShell
PS X:\Users\Fernando\Documents\UNal\Alumnos\Ciclo II\sqlite3> sqlite3
SQLite version 3.36.0 2021-06-18 18:36:39
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

# Bases de Datos

## SQL ejemplos:

- `sqlite> .databases`
  - listado de todas las bases de datos
- `sqlite > create table productos (Id INT primary key);`
- `sqlite > .tables`
  - listado de todas las tablas
- `sqlite > DROP TABLE IF EXISTS productos;`

# Bases de Datos

## SQL ejemplos:

- `sqlite > select * from productos;`
- `INSERT INTO productos  
VALUES(1,'Pantalón',8,'Corto y sin costuras');`
- `create table test(id integer primary key autoincrement,  
descripcion varchar(140));`
- `.exit`

# SQL

## 1. Cómo se crea un Schema:

```
CREATE SCHEMA libreria ;
```

## 2. Cómo se usa un Schema:

```
USE libreria ;
```

# SQL

## 1. Cómo se crea una Tabla:

```
CREATE TABLE nombre_tabla (  
  nombre_columna lista de propiedades ,  
  ...  
  nombre_columna lista de propiedades ,  
  [ referencias ]  
);
```

# SQL

Ejemplo:

```
CREATE TABLE venta (  
  vta_fatura integer AUTO_INCREMENT ,  
  vta_id_cliente integer NOT NULL ,  
  vta_fecha date NOT NULL ,  
  vta_total double ( 6 , 1 ) NULL ,  
  vta_pago ENUM ( ' Efectivo' , ' TarjetaCrédito' ,  
    ' TarjetaDébito' , ' Transferencia' , 'PSE ' ) ,  
  vta_trans varchar ( 20 ) DEFAULT " Electrónica"  
);
```

# SQL

Ejemplo:

```
CREATE TABLE libro (  
  Lib_Id integer NOT NULL primary key ,  
  Lib_Nombre varchar(60) NOT NULL ,  
  Lib_Pub smallint NULL ,  
);
```



# Consultas

## Operadores de algebra relacional básicos:

- Proyección ( $\pi$ )
- Selección ( $\sigma$ )
- Renombrar ( $\rho$ )
- Eliminar Duplicados ( $\delta$ )
- Producto cartesiano ( $\times$ )
- Join, Enlace o Reunión ( $\bowtie$ )

# Consultas

Ejemplo de Proyección:

```
SELECT libNombre FROM libro ;
```

```
SELECT libNombre , libPub FROM libro ;
```

```
SELECT autNombre , autApellido FROM autor ;
```

```
SELECT * FROM autor ;
```

# Consultas

Ejemplo de Proyección con un criterio:

```
SELECT libNombre FROM libro WHERE libId = 1003;
```

```
SELECT libNombre , libPub FROM libro WHERE libPub > 1960 ;
```

```
SELECT * FROM autor WHERE libNombre LIKE "%los%";
```

```
SELECT * FROM libro WHERE libNombre LIKE "C%" and libNombre NOT LIKE "%los%";
```

# Consultas

Ejemplo de Proyección con un criterio:

```
SELECT * FROM libro WHERE LENGHT ( libNombre ) > 8 ;
```

```
SELECT * FROM libro WHERE LOWER ( libNombre ) LIKE "maría" ;
```

```
SELECT libNombre FROM libro ORDER BY libPub DESC LIMIT 3;
```

# Consultas

```
DROP SCHEMA if exists almacen;  
CREATE SCHEMA almacen;  
USE almacen;
```

```
drop table if exists producto;  
drop table if exists vehiculo;  
drop table if exists cliente;  
drop table if exists vendedor;  
drop table if exists servicio;
```

# Consultas

```
INSERT INTO libro (Lib_Id, Lib_Nombre, Lib_Pub) VALUES (7, 'Norma', 1985);  
INSERT INTO `dbciclo2`.`libro` (`Lib_Id`, `Lib_Nombre`, `Lib_Pub`) VALUES ('2', 'Juan', '1960');  
INSERT INTO `dbciclo2`.`libro` (`Lib_Id`, `Lib_Nombre`, `Lib_Pub`) VALUES ('3', 'María', '1970');  
  
INSERT INTO libro (Lib_Id, Lib_Nombre, `Lib_Pub`) VALUES ('9', 'Petra', '1999');
```

# Consultas

```
INSERT INTO t_inventario_vehiculos (t_inventario_vehiculos_Id, t_inventario_vehiculos_tipo,  
t_inventario_vehiculos_velocidad, t_inventario_vehiculos_otro) VALUES ('ASD4401', 'PARTICULAR', '130',  
'AZUL');
```

```
INSERT INTO `dbciclo2`.`t_inventario_vehiculos` (`t_inventario_vehiculos_Id`, `t_inventario_vehiculos_tipo`,  
`t_inventario_vehiculos_velocidad`, `t_inventario_vehiculos_otro`) VALUES ('PLT099', 'PARTICULAR', '210',  
'BLANCO');
```

```
INSERT INTO `dbciclo2`.`t_inventario_vehiculos` (`t_inventario_vehiculos_Id`, `t_inventario_vehiculos_tipo`,  
`t_inventario_vehiculos_velocidad`, `t_inventario_vehiculos_otro`) VALUES ('SAB001', 'PÚBLICO', '190',  
'AMARILLO');
```

# Consultas

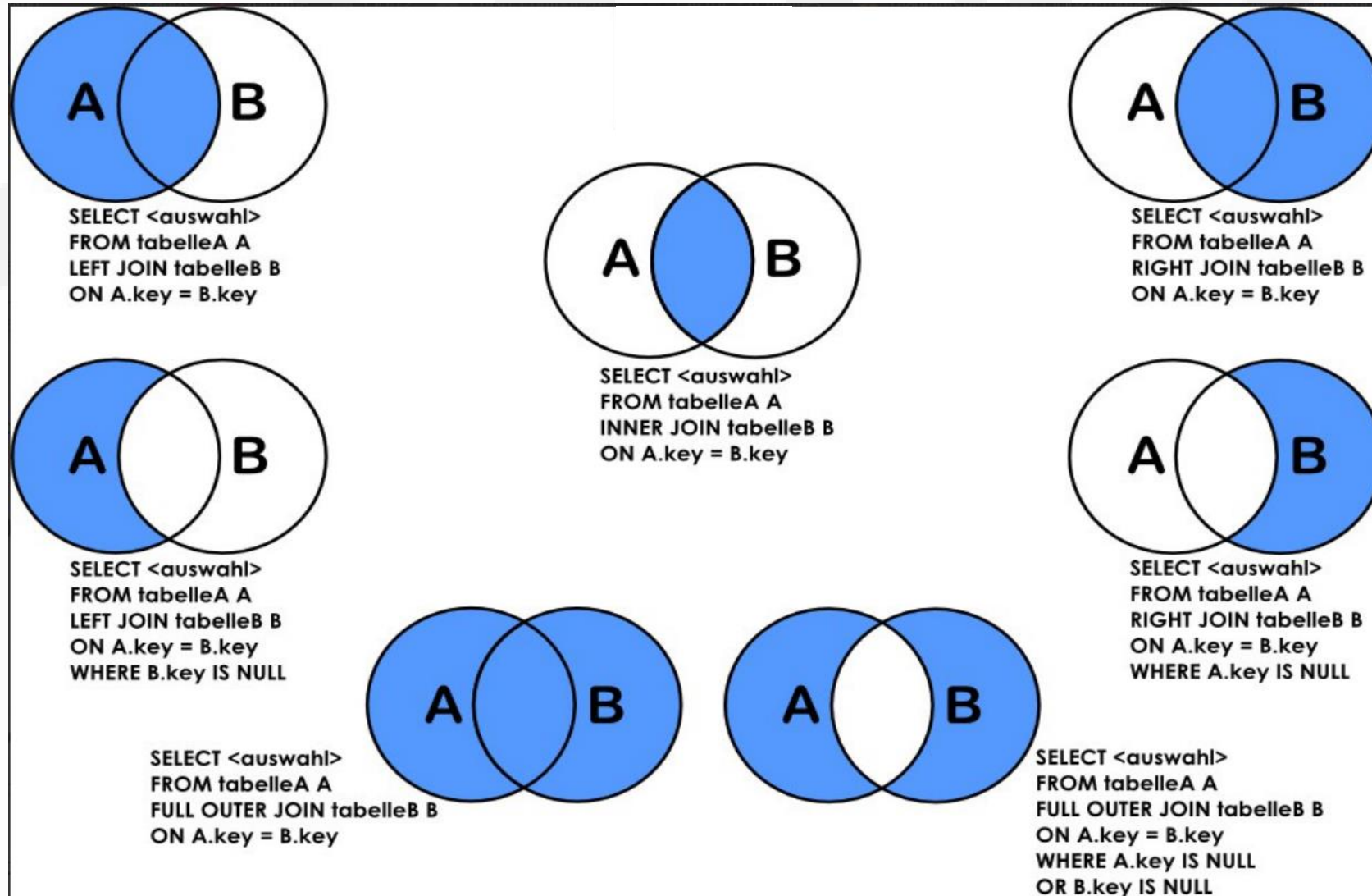
```
CREATE TABLE `dbciclo2`.`t_inventario_vehiculos` (  
  `t_inventario_vehiculos_Id` INT NOT NULL,  
  `t_inventario_vehiculos_tipo` VARCHAR(10) NULL,  
  `t_inventario_vehiculos_velocidad` SMALLINT(3) NOT NULL,  
  PRIMARY KEY (`t_inventario_vehiculos_Id`)) ENGINE =  
InnoDBDEFAULT CHARACTER SET = utf8;
```



# Teoría de Conjuntos

(Data Base)

# INNER JOIN



# INNER JOIN

Sirve para unir tablas usando la misma clave foránea

```
SELECT * FROM Tabla1 INNER JOIN Tabla2  
ON Tabla1.identificador = Tabla2.identificador;
```

# Consultas

# Consultas

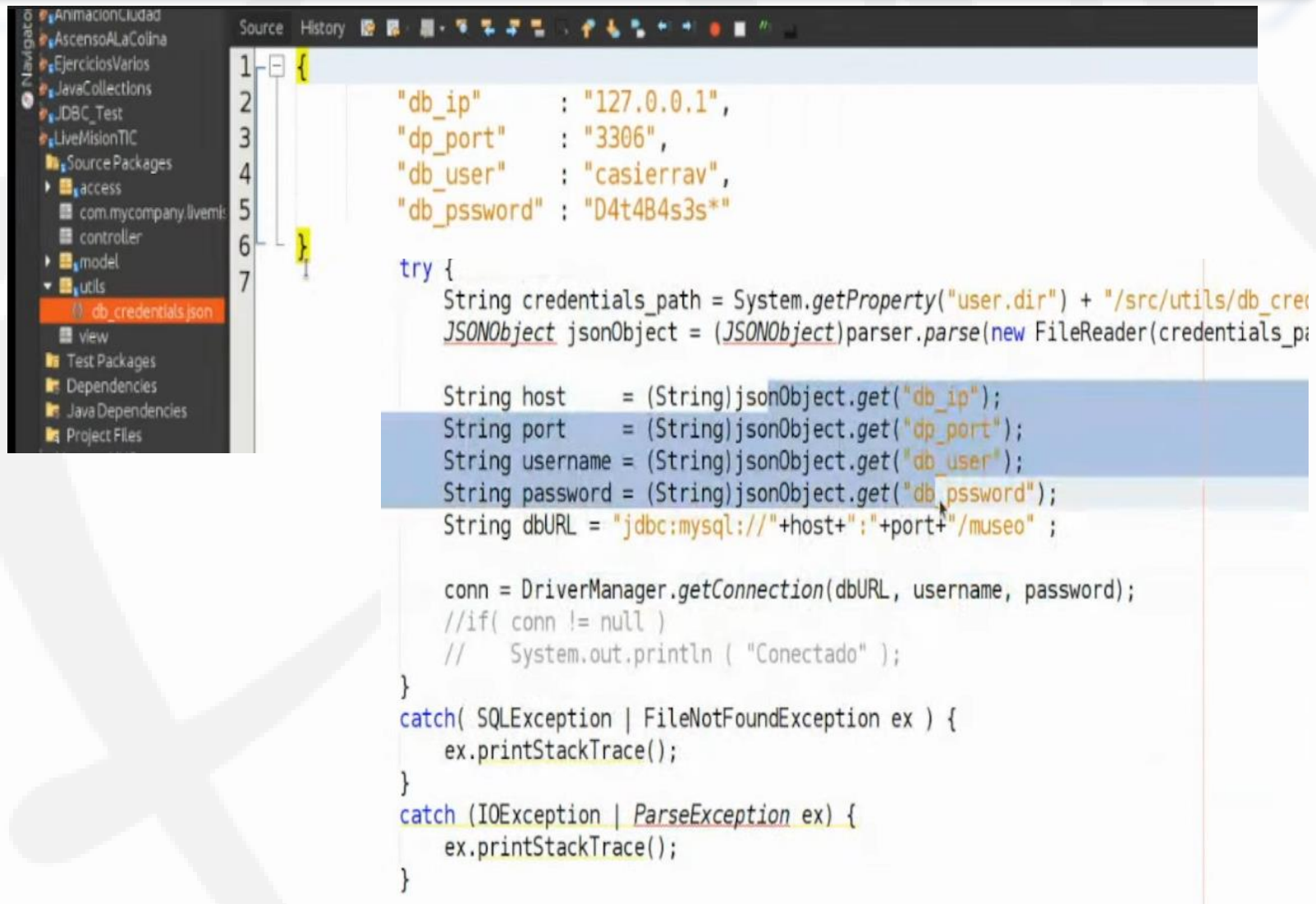
```
SELECT obra.ob_id, obra.ob_nombre, obra.ob_tipo, obra.ob_costo, museo.mu_nombre, exposicio
FROM obra
JOIN exposicion ON obra.ob_ex_id=exposicion.ex_id
JOIN presentacion ON presentacion.pre_ob_id=obra.ob_id
JOIN museo ON presentacion.pre_mu_id=museo.mu_id
WHERE obra.ob_nombre LIKE '%%'
ORDER BY obra.ob_nombre;
```

```
mysql> SELECT obra.ob_id, obra.ob_nombre, obra.ob_tipo, obra.ob_costo, museo.mu_nombre, exposicion.ex_nombre, presentacion.pre_fecha
inicial
```

```
-> FROM obra
-> JOIN exposicion ON obra.ob_ex_id=exposicion.ex_id
-> JOIN presentacion ON presentacion.pre_ob_id=obra.ob_id
-> JOIN museo ON presentacion.pre_mu_id=museo.mu_id
-> WHERE obra.ob_nombre LIKE '%%'
-> ORDER BY obra.ob_nombre;
```

ob_id	ob_nombre	ob_tipo	ob_costo	mu_nombre	ex_nombre	pre_fecha	inicial
400	Auto-retrato	pintura	100.0	Británico	Impresionismo	Ago-nov-2013	
302	Cabeza de mujer	escultura	300.0	Shangai	Cubismo	Oct-dic-2013	
201	David	escultura	700.0	Met	Renacimiento	Ene-abr-2013	
201	David	escultura	700.0	Louvre	Renacimiento	May-sept-2013	
200	Fornarina	pintura	400.0	Shangai	Renacimiento	Oct-dic-2013	
200	Fornarina	pintura	400.0	Louvre	Renacimiento	May-sept-2013	
113	Hombre vitruvio	boceto	400.0	Met	Da Vinci	Sept-2013	
113	Hombre vitruvio	boceto	400.0	Shangai	Da Vinci	Julio 2013	
113	Hombre vitruvio	boceto	400.0	Met	Da Vinci	Abr-jun-2013	

# Consultas



The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer lists various packages and files, with `db_credentials.json` selected. The code editor displays the JSON content of this file and the Java code that reads it. The JSON defines database connection parameters: `db_ip`, `dp_port`, `db_user`, and `db_pssword`. The Java code uses `System.getProperty` to find the file, `JSONObject` to parse it, and `DriverManager.getConnection` to establish a connection. It also includes exception handling for `SQLException`, `FileNotFoundException`, `IOException`, and `ParseException`.

```
1 {
2   "db_ip"      : "127.0.0.1",
3   "dp_port"    : "3306",
4   "db_user"    : "casiterrav",
5   "db_pssword" : "D4t4B4s3s"
6 }
7
try {
    String credentials_path = System.getProperty("user.dir") + "/src/utils/db_credentials.json";
    JSONObject jsonObject = (JSONObject)parser.parse(new FileReader(credentials_path));

    String host      = (String)jsonObject.get("db_ip");
    String port      = (String)jsonObject.get("dp_port");
    String username  = (String)jsonObject.get("db_user");
    String password  = (String)jsonObject.get("db_pssword");
    String dbURL     = "jdbc:mysql://" + host + ":" + port + "/museo";

    conn = DriverManager.getConnection(dbURL, username, password);
    //if( conn != null )
    //    System.out.println ( "Conectado" );
}
catch( SQLException | FileNotFoundException ex ) {
    ex.printStackTrace();
}
catch( IOException | ParseException ex ) {
    ex.printStackTrace();
}
```



# Consultas

```
private void initComponents(){
    setTitle("Museos - MVC");

    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch (Exception e) {
        e.printStackTrace();
    }

    ResultsPanel resultsPanel = new ResultsPanel();
    setContentPane(resultsPanel);
    add(new ControlsPanel(resultsPanel));

    setSize(1040, 720);

    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = getSize();
    setLocation((screenSize.width - frameSize.width) / 2,
                (screenSize.height - frameSize.height) / 2);
}
```

# Consultas

1&PARTICULAR&AAA213&120&5&AZUL

[1,PARTICULAR,AAA213,120,5,AZUL]



# Consultas