

API JDBC

Parte I

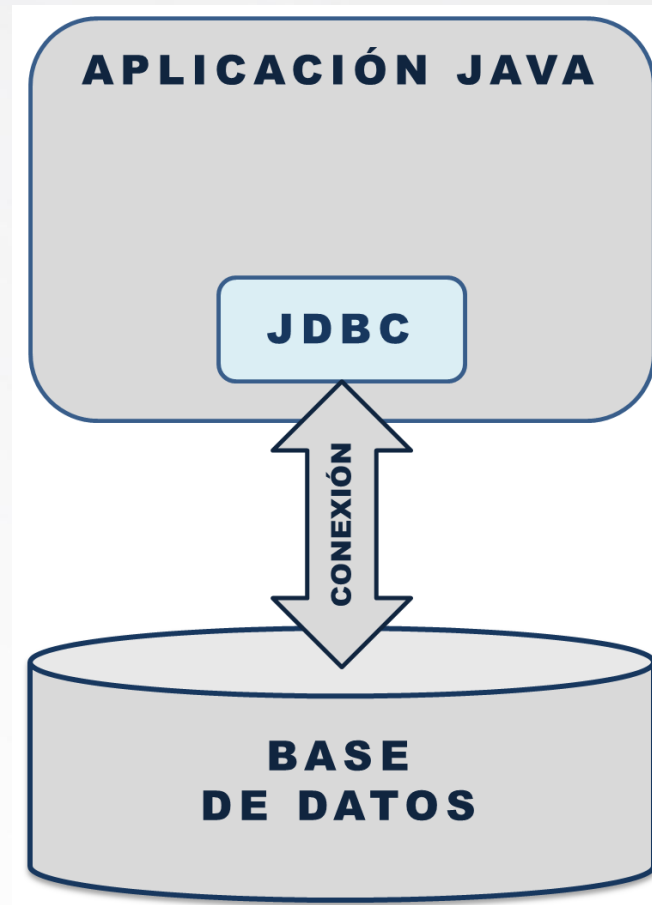
Carrera: Computación e Informática

Semestre: 2016 - I

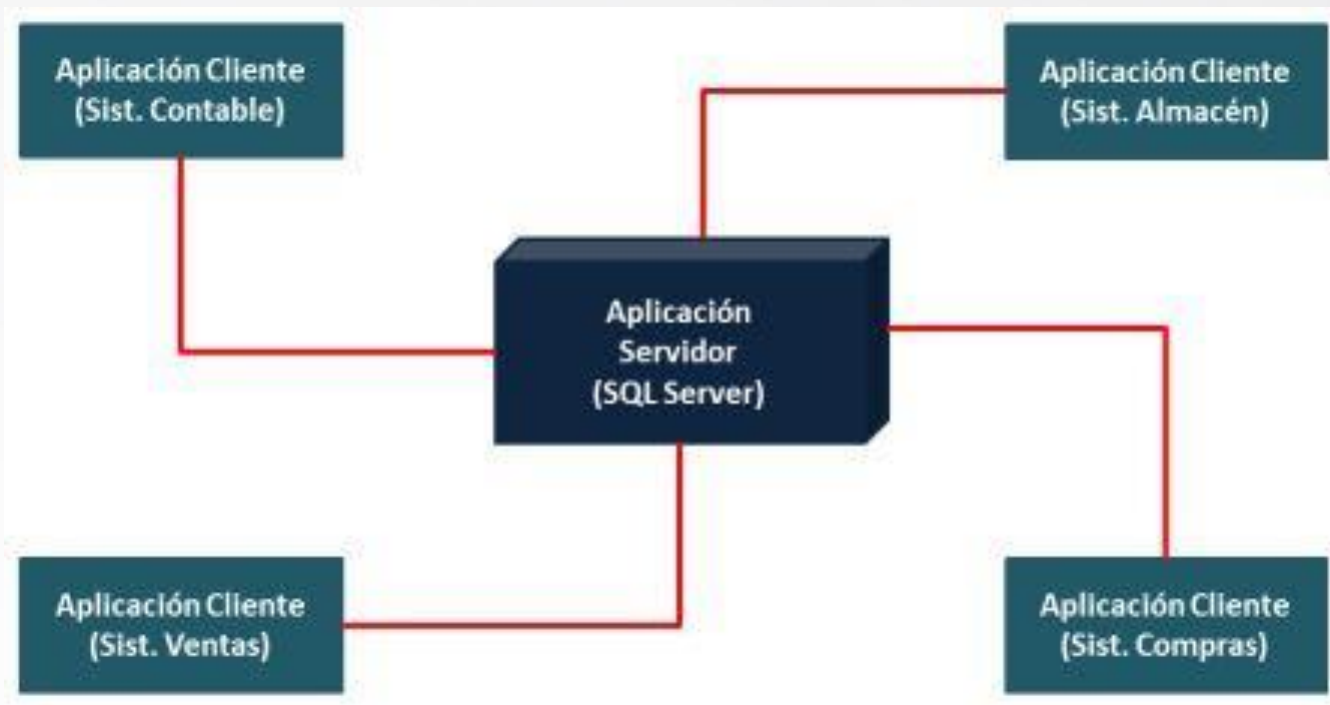
Nombre de Unidad Didáctica: Herramientas de Desarrollo de Software



Arquitectura Cliente-Servidor



Arquitectura Cliente-Servidor



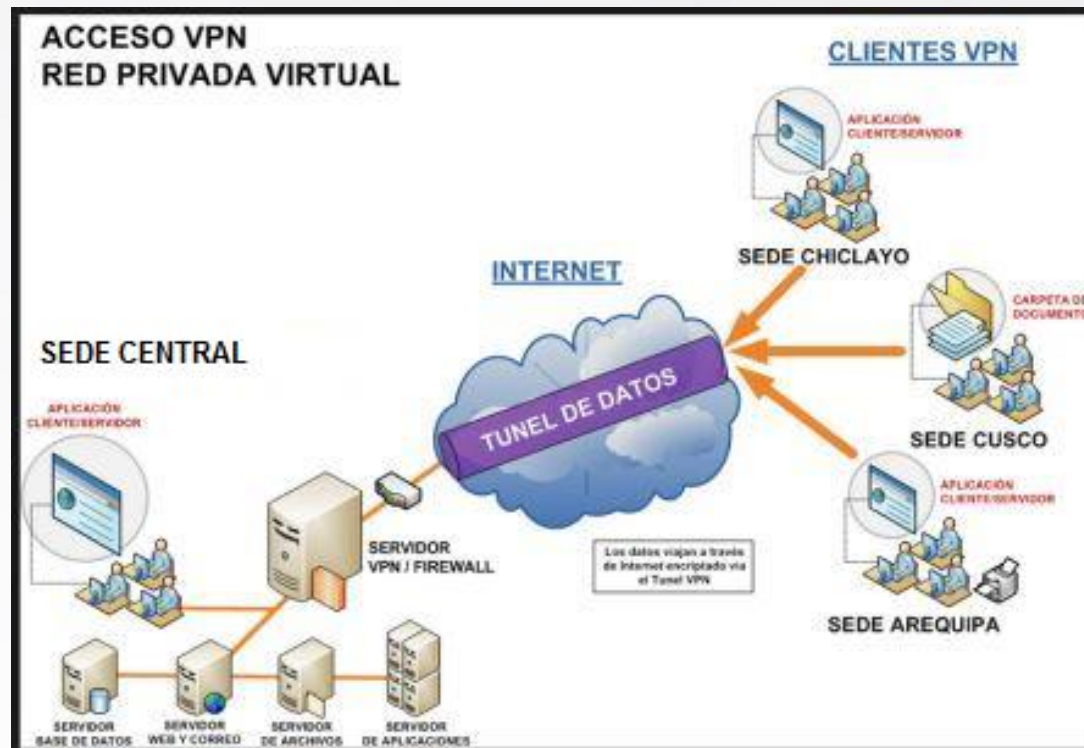
Arquitectura Cliente-Servidor

Modelo Clásico

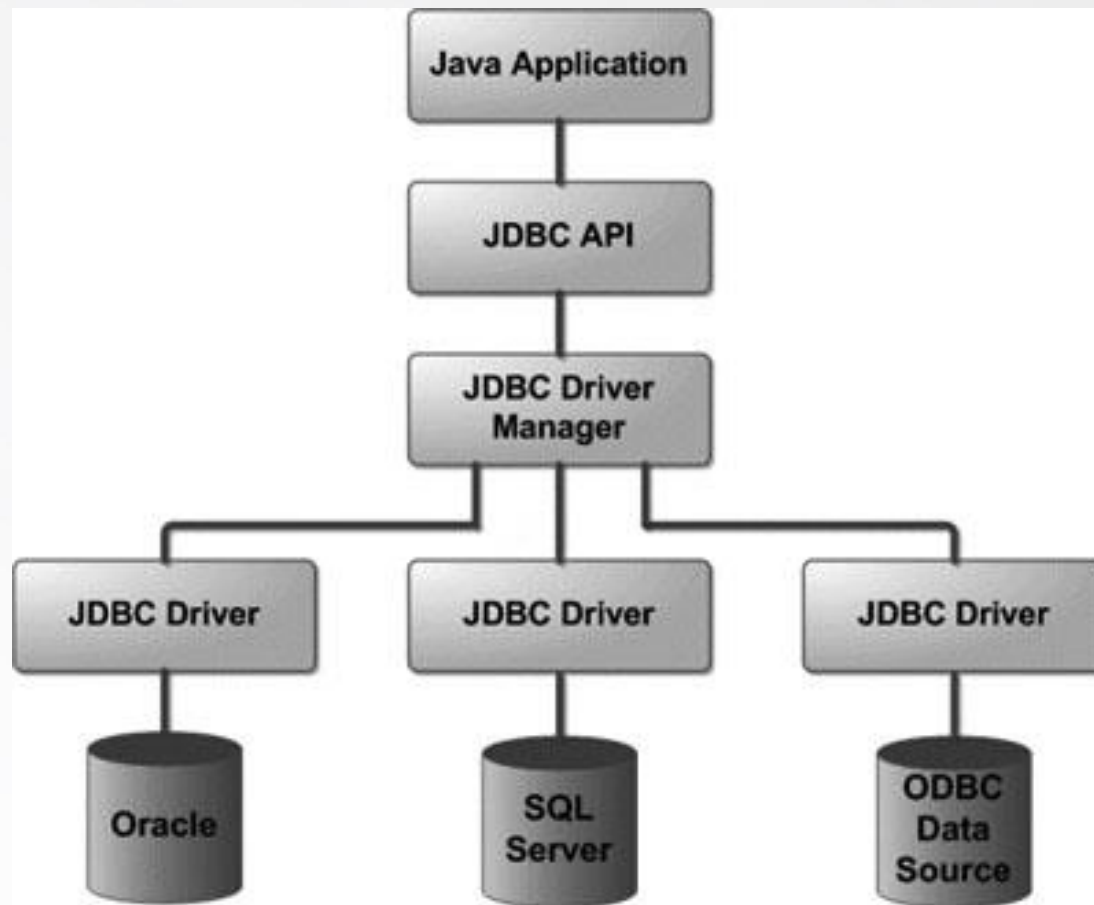


Arquitectura Cliente-Servidor

Modelo Clásico



API JDBC



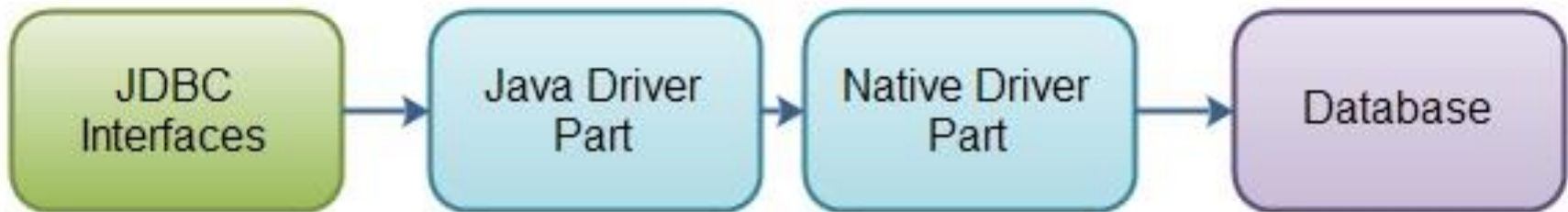
TIPOS DE DRIVER

Tipo 1: JDBC-ODBC bridge driver



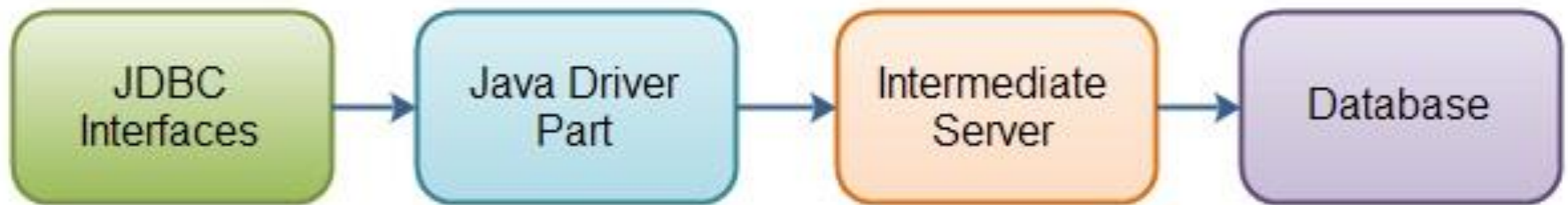
TIPOS DE DRIVER

Tipo 2: Native API/Partly Java Driver



TIPOS DE DRIVER

Tipo 3: Pure Java Driver

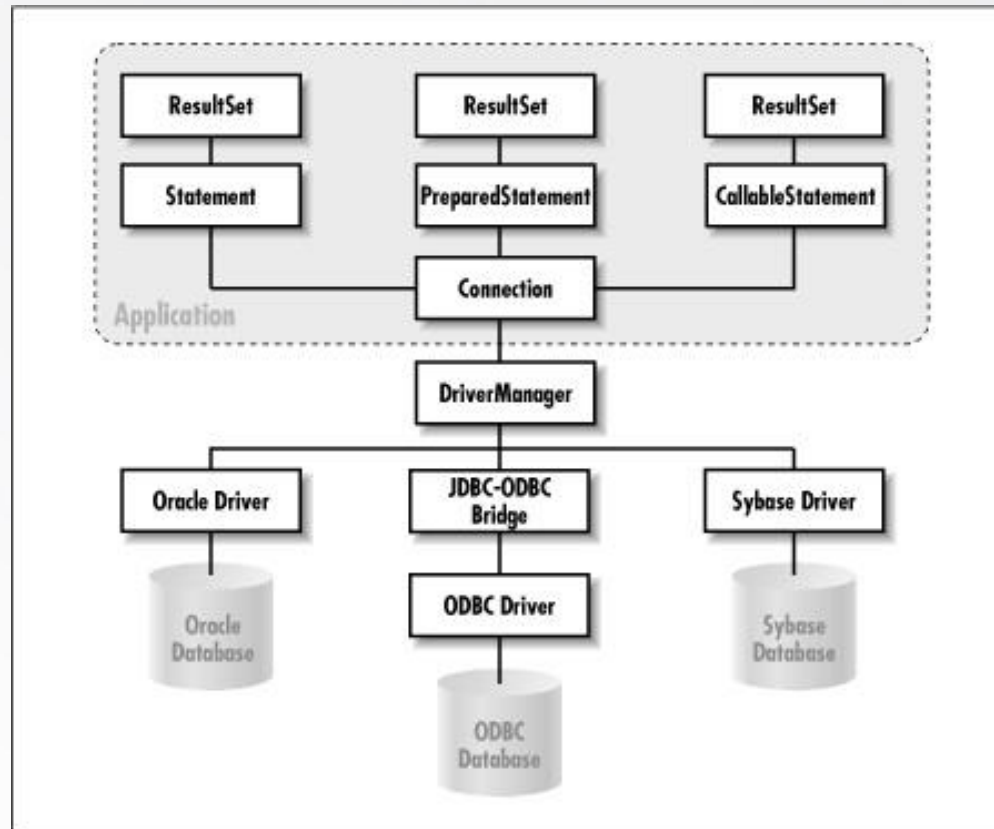


TIPOS DE DRIVER

Tipo 4: Native Protocol Java Driver

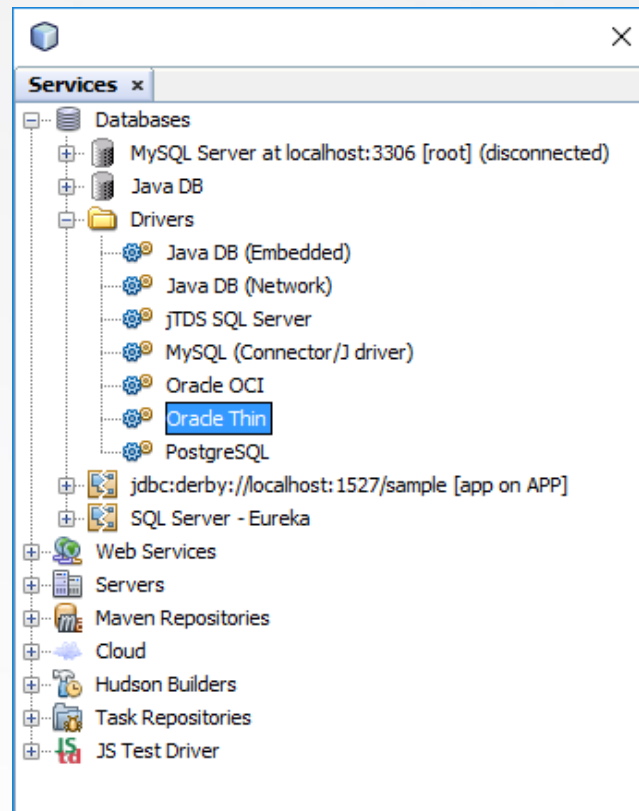


PRINCIPALES COMPONENTES



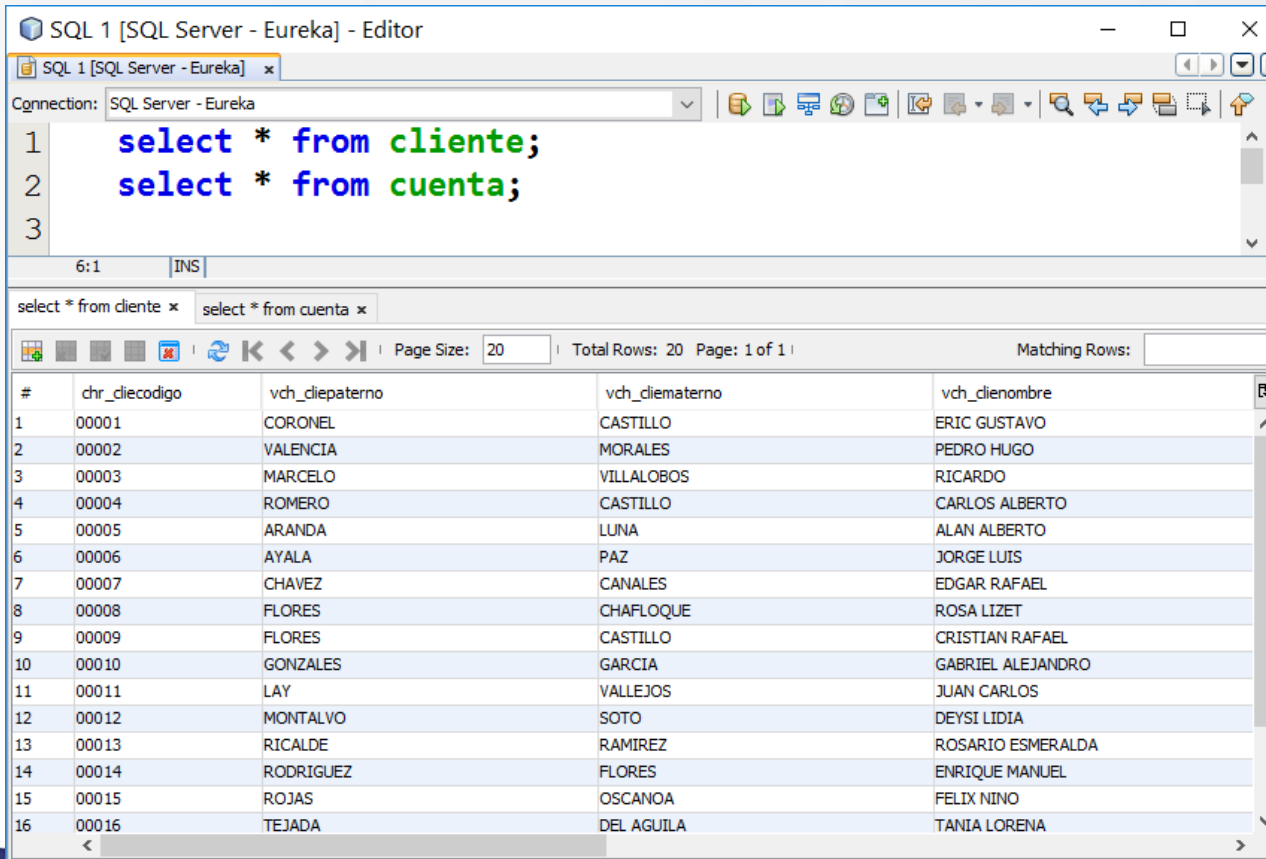
USANDO NETBEANS

Herramienta Database



USANDO NETBEANS

Ejecutando consultas



The screenshot shows the NetBeans IDE with the SQL Editor open. The editor contains the following SQL queries:

```
1 select * from cliente;  
2 select * from cuenta;  
3
```

The query execution results are displayed in a table below the editor. The table has 5 columns: #, chr_diecodigo, vch_diepaterno, vch_diematerno, and vch_dienombre. The results show 16 rows of data.

#	chr_diecodigo	vch_diepaterno	vch_diematerno	vch_dienombre
1	00001	CORONEL	CASTILLO	ERIC GUSTAVO
2	00002	VALENCIA	MORALES	PEDRO HUGO
3	00003	MARCELO	VILLALOBOS	RICARDO
4	00004	ROMERO	CASTILLO	CARLOS ALBERTO
5	00005	ARANDA	LUNA	ALAN ALBERTO
6	00006	AYALA	PAZ	JORGE LUIS
7	00007	CHAVEZ	CANALES	EDGAR RAFAEL
8	00008	FLORES	CHAFLOQUE	ROSA LIZET
9	00009	FLORES	CASTILLO	CRISTIAN RAFAEL
10	00010	GONZALES	GARCIA	GABRIEL ALEJANDRO
11	00011	LAY	VALLEJOS	JUAN CARLOS
12	00012	MONTALVO	SOTO	DEYSI LIDIA
13	00013	RICALDE	RAMIREZ	ROSARIO ESMERALDA
14	00014	RODRIGUEZ	FLORES	ENRIQUE MANUEL
15	00015	ROJAS	OSCANOA	FELIX NINO
16	00016	TEJADA	DEL AGUILA	TANIA LORENA

PROGRAMANDO LA CONEXIÓN

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public final class AccesoDB {

    // Para que no se pueda crear instancias de esta clase
    private AccesoDB() {
    }

    // Retorna un objeto de tipo Connection
    public static Connection getConnection() throws SQLException {
        Connection cn = null;
        try {
            // Paso 1: Cargar el driver a memoria
            Class.forName("oracle.jdbc.OracleDriver").newInstance();
            // Paso 2: Obtener el objeto Connection
            String url = "jdbc:oracle:thin:@localhost:1521:ORCL";
            cn = DriverManager.getConnection(url, "eureka", "admin");
        } catch (SQLException e) {
            throw e;
        } catch (ClassNotFoundException e) {
            throw new SQLException("No se encontró el driver de la base de datos.");
        } catch (Exception e) {
            throw new SQLException("No se puede establecer la conexión con la BD.");
        }
        return cn;
    }
}
```



PROGRAMANDO LA CONEXIÓN

Probando la conexión

```
package pe.egcc.eurekaapp.prueba;

import java.sql.Connection;
import pe.egcc.eurekaapp.dao.db.AccesoDB;

public class Prueba01 {

    public static void main(String[] args) {
        try {
            Connection cn;
            cn = AccesoDB.getConnection();
            System.out.println("Conexión ok.");
            cn.close();
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```



PROGRAMANDO LA CONEXIÓN

Ejecutando una consulta

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import pe.egcc.eurekaapp.dao.db.AccesoDB;

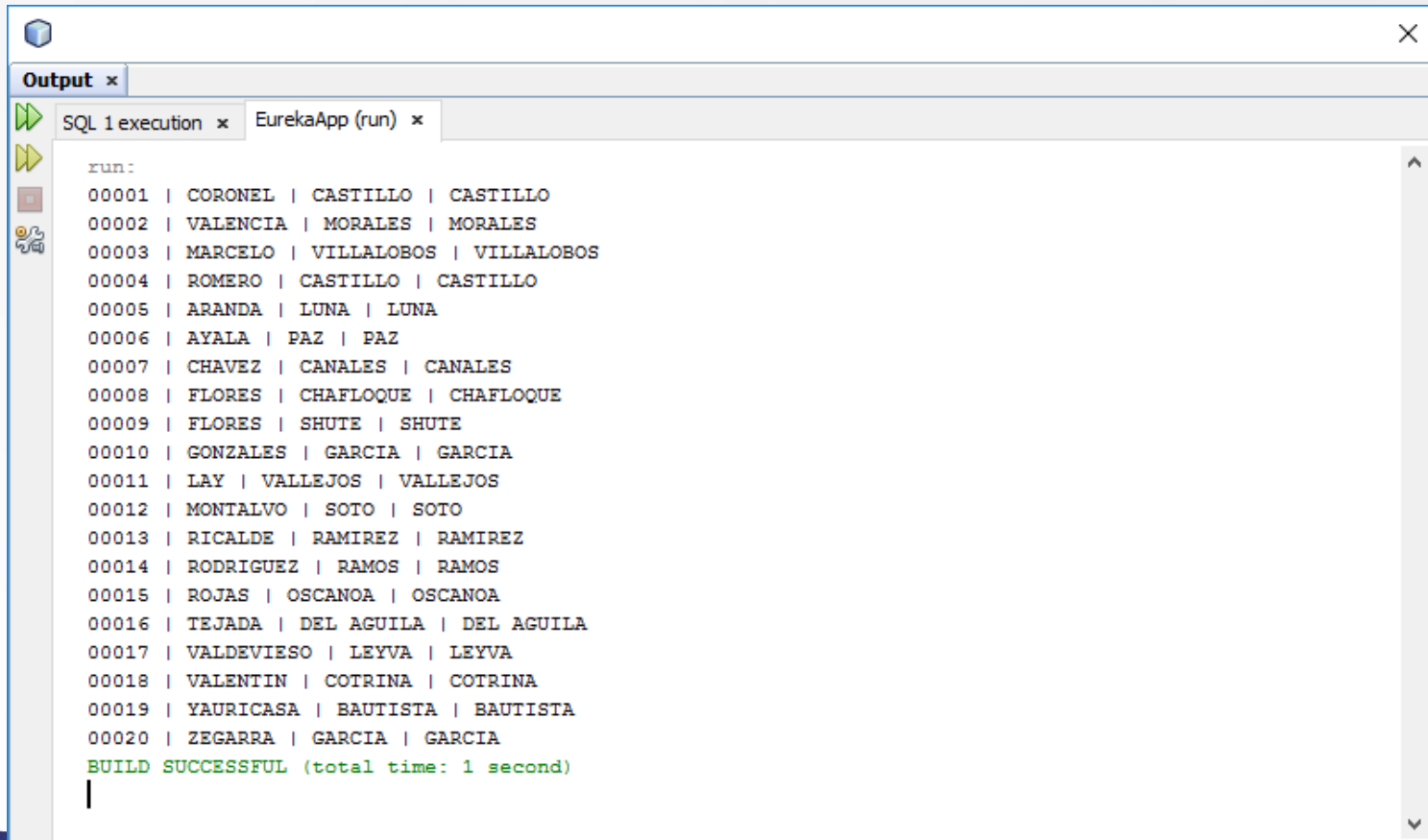
public class Prueba02 {

    public static void main(String[] args) {
        try {
            Connection cn;
            cn = AccesoDB.getConnection();
            String sql = "SELECT CHR_CLIECODIGO, VCH_CLIEPATERNO, VCH_CLIEMATERO, VCH_CLIENOMBRE FROM CLIENTE";
            Statement stm = cn.createStatement();
            ResultSet rs = stm.executeQuery(sql);
            while(rs.next()){
                System.out.println(rs.getString("CHR_CLIECODIGO") + " | " + rs.getString("VCH_CLIEPATERNO") + " | "
                    + rs.getString("VCH_CLIEMATERO") + " | " + rs.getString("VCH_CLIENOMBRE"));
            }
            rs.close();
            stm.close();
            cn.close();
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```



PROGRAMANDO LA CONEXIÓN

Ejecutando una consulta



The screenshot shows an IDE output window with two tabs: "SQL 1 execution x" and "EurekaApp (run) x". The "SQL 1 execution x" tab is active, displaying the results of a query. The output starts with "run:" followed by 20 rows of data. Each row contains an ID and three names separated by vertical bars. The data ends with a green status message: "BUILD SUCCESSFUL (total time: 1 second)".

```
run:
00001 | CORONEL | CASTILLO | CASTILLO
00002 | VALENCIA | MORALES | MORALES
00003 | MARCELO | VILLALOBOS | VILLALOBOS
00004 | ROMERO | CASTILLO | CASTILLO
00005 | ARANDA | LUNA | LUNA
00006 | AYALA | PAZ | PAZ
00007 | CHAVEZ | CANALES | CANALES
00008 | FLORES | CHAFLOQUE | CHAFLOQUE
00009 | FLORES | SHUTE | SHUTE
00010 | GONZALES | GARCIA | GARCIA
00011 | LAY | VALLEJOS | VALLEJOS
00012 | MONTALVO | SOTO | SOTO
00013 | RICALDE | RAMIREZ | RAMIREZ
00014 | RODRIGUEZ | RAMOS | RAMOS
00015 | ROJAS | OSCANOA | OSCANOA
00016 | TEJADA | DEL AGUILA | DEL AGUILA
00017 | VALDEVIESO | LEYVA | LEYVA
00018 | VALENTIN | COTRINA | COTRINA
00019 | YAURICASA | BAUTISTA | BAUTISTA
00020 | ZEGARRA | GARCIA | GARCIA
BUILD SUCCESSFUL (total time: 1 second)
```

MUCHAS GRACIAS

