



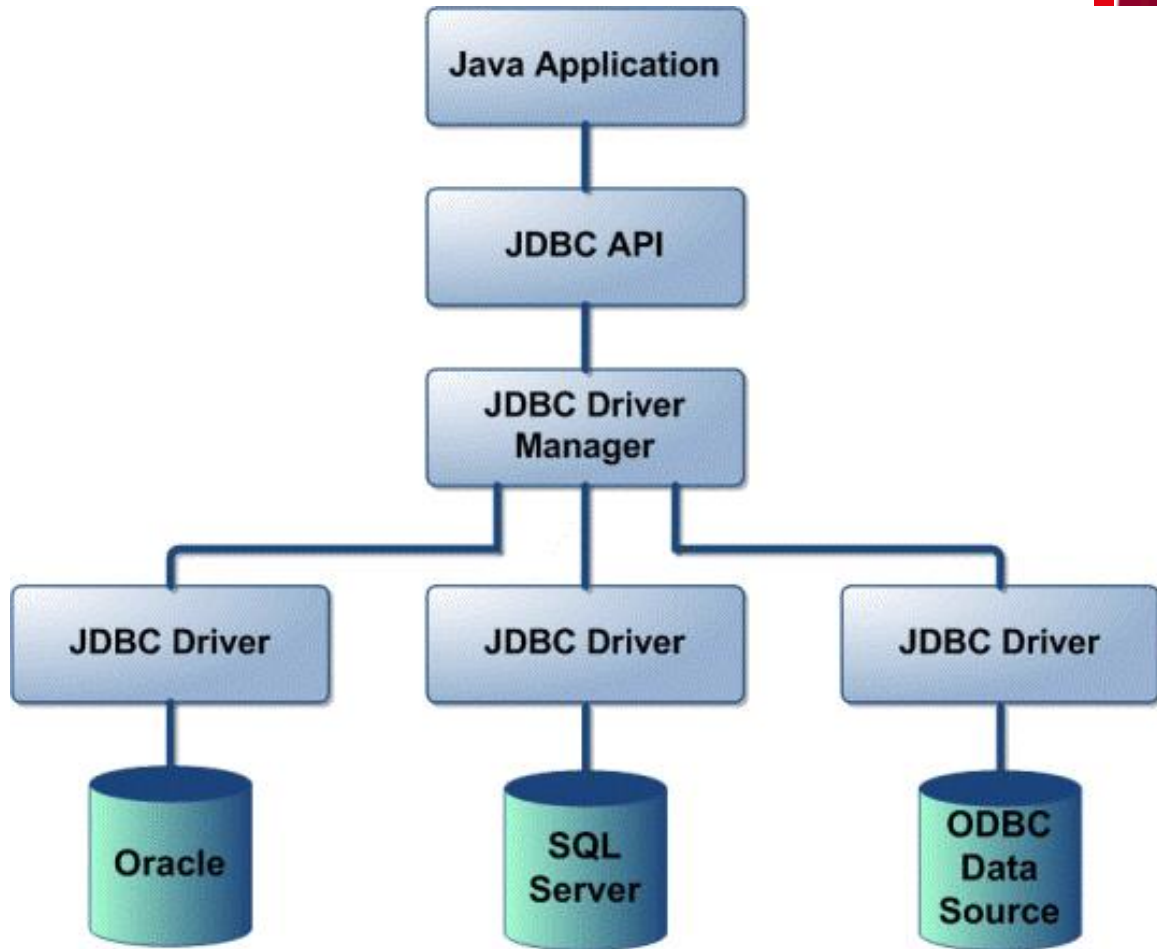
SESIÓN 12:

**Conectividad a base de datos: - Abrir y cerrar conexión. -
Lectura y escritura de datos. - Manejo de transacciones**



Que es el JDBC

Java Database Connectivity, más conocida por sus siglas **JDBC**, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.



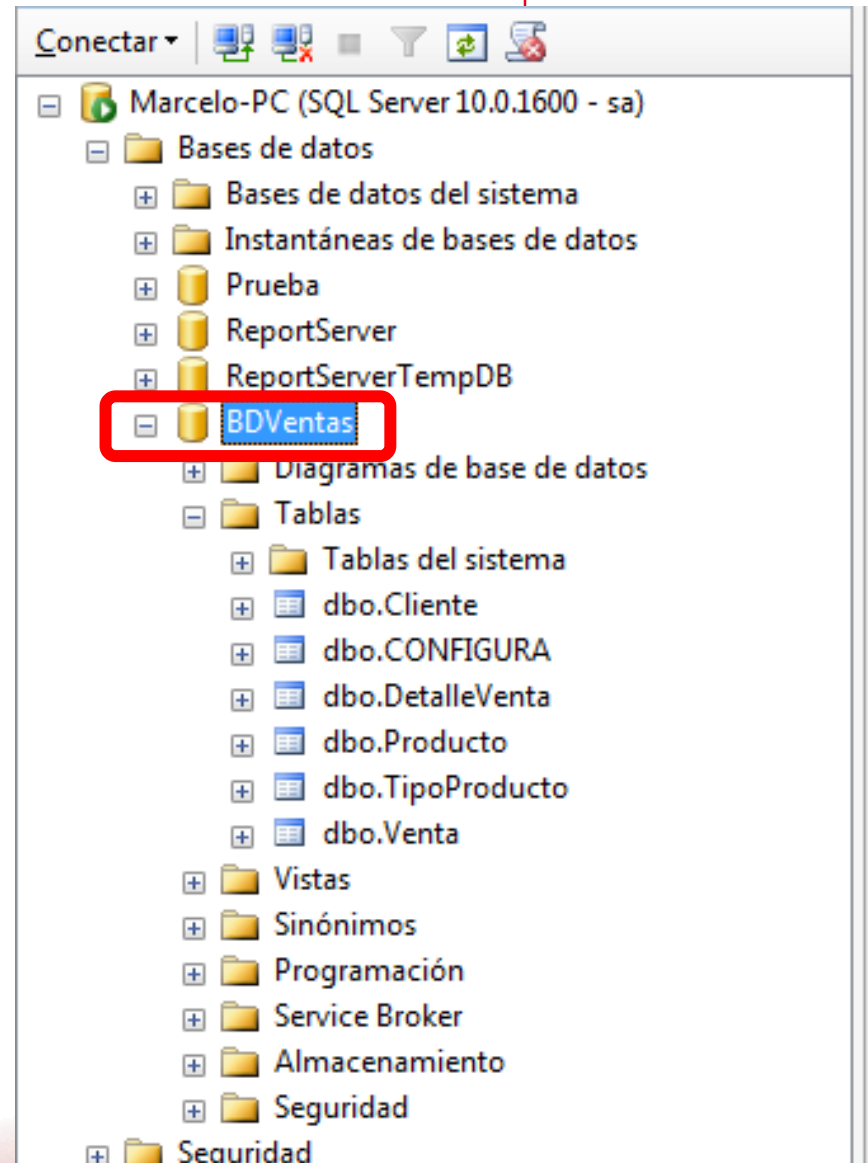


JDBC ofrece el paquete **java.sql**, en el que existen clases muy útiles para trabajar con bases de datos.

Clase	Descripción
DriverManager	Para cargar un driver
Connection	Para establecer conexiones con las bases de datos
Statement	Para crear consultas SQL y enviarlas a las BBDD
ResultSet	Para almacenar el resultado de la consulta

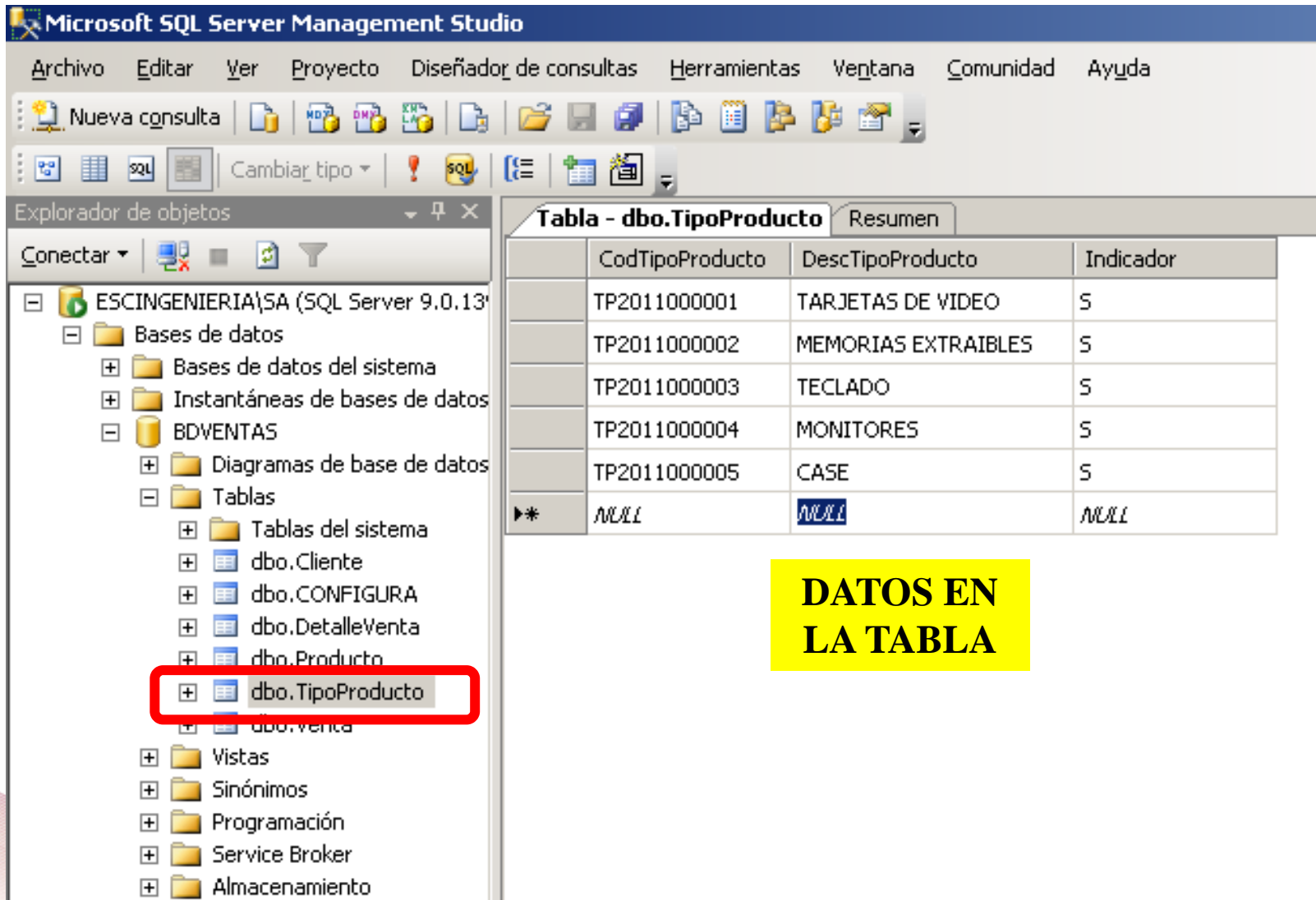
Ejemplo de conexión mediante el driver JDBC-ODBC

Vamos a establecer un conexión JDBC-ODBC para conectarnos con SQL Server 2008 con la base de datos BDVentas.



Ejemplo de conexión y consulta

Supongamos que tenemos la siguiente tabla en Microsoft SQL Server 2008



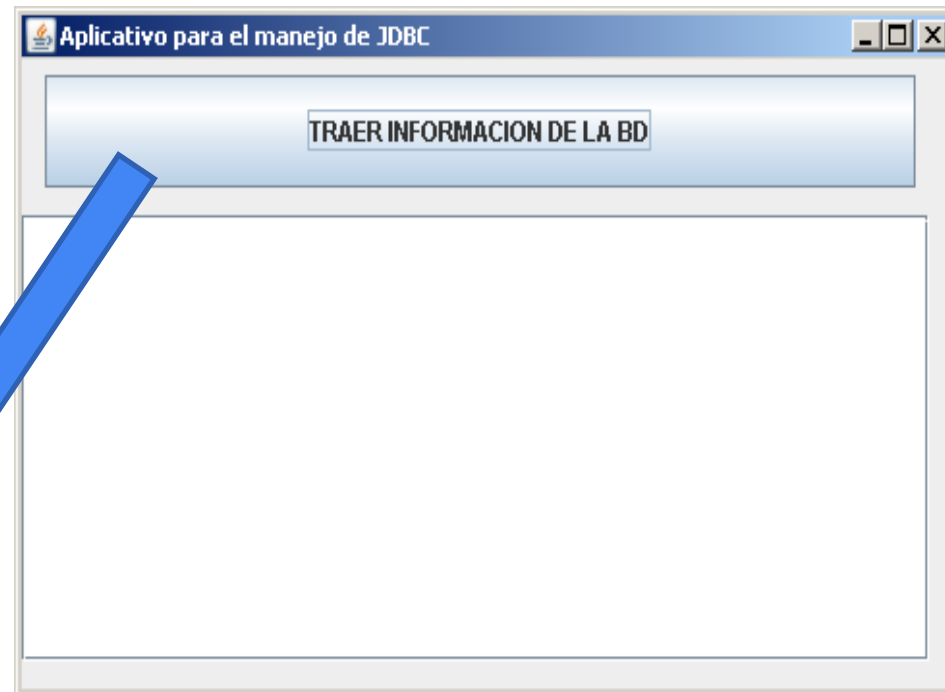
The screenshot displays the Microsoft SQL Server Management Studio interface. On the left, the 'Explorador de objetos' (Object Explorer) shows the database structure for 'ESCINGENIERIA\SA (SQL Server 9.0.13)'. The 'Tablas' (Tables) folder is expanded, and 'dbo.TipoProducto' is highlighted with a red rectangle. The main pane on the right shows the 'Tabla - dbo.TipoProducto' with the 'Resumen' (Summary) tab selected. The table structure is as follows:

	CodTipoProducto	DescTipoProducto	Indicador
	TP2011000001	TARJETAS DE VIDEO	S
	TP2011000002	MEMORIAS EXTRAIBLES	S
	TP2011000003	TECLADO	S
	TP2011000004	MONITORES	S
	TP2011000005	CASE	S
►*	NULL	NULL	NULL

**DATOS EN
LA TABLA**



Construya el siguiente aplicativo
y coloque el siguiente código
sobre el botón



```
private void jButtonTraerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    try{  
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
        Connection cn=DriverManager.getConnection("jdbc:odbc:BDVentas","sa","sa");  
        Statement st=cn.createStatement();  
        ResultSet rs=st.executeQuery("SELECT CodTipoProducto AS Codigo,"  
            + " DescTipoProducto AS Descripcion FROM TipoProducto");  
        muestraData(rs);  
        cn.close();  
    }catch(Exception ex){  
        jtxaDatos.append("La conexion fracaso..");  
        jtxaDatos.append(ex.toString());  
    }  
}
```



```
Class.forName("sun.jdbc.c  
Connection cn=DriverManag  
Statement st=cn.createSta  
ResultSet rs=st.executeQu  
+ "DeeTipoProdu  
muestraData(rs);  
cn.close();  
} catch (Exception ex) {  
    jtxaDatos.append("La co
```

```
public void muestraData(ResultSet r) throws Exception  
{  
    ResultSetMetaData rmeta=r.getMetaData();  
    int numColumnas=rmeta.getColumnCount();//cuantas columnas  
    jtxaDatos.setText("");  
    for(int i=1;i<=numColumnas;i++)  
    {  
        //obtenemos el nombre de la columna  
        jtxaDatos.append(rmeta.getColumnName(i)+"\t\t");  
    }  
    jtxaDatos.append("\n");  
    while(r.next()) // hasta el final del archivo  
    {  
        for(int j=1;j<=numColumnas;j++)  
        {  
            //Se extra el dato en la posicion j del registro  
            jtxaDatos.append(r.getString(j)+"\t\t");  
        }  
        jtxaDatos.append("\n");  
    }  
}
```


Explicación de los métodos

Los objetos de JDBC deben estar siempre dentro de una instrucción **try****catch**

Para indicar que tipo de driver vamos a usar en nuestra aplicación, ponemos la sentencia (driver a usar es el puente JDBC-ODBC)

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

El administrador de drivers (DriverManager) del JDBC, para el caso del puente JDBC-ODBC, nos pide 3 argumentos: el DSN, login y password, como se indica en la siguiente instrucción:

```
Connection cn=DriverManager.getConnection("jdbc:odbc:BDVentas","sa","sa");
```

Este nos devuelve un objeto de tipo **Connection** con el cual se apertura el acceso a la base de datos

Después de establecer la conexión, necesitamos ejecutar un instrucción SQL para traernos la información de la base de datos y para eso nos ayuda el objeto Statement, la instrucción es la siguiente

```
Statement st=cn.createStatement();
```

Una vez creado el objeto para la ejecución de la instrucción enviamos la sentencia SQL mediante el metodo **executeQuery**, como se muestra:

```
ResultSet rs=st.executeQuery("SELECT CodTipoProducto AS Codigo,"  
+ " DescTipoProducto AS Descripcion FROM TipoProducto");
```

El objeto **ResultSet** permite acceder a las filas de las consulta mediante una instrucción Select del SQL

El método **muestraData** es un método personalizado que responde a una instrucción Select. El método recibe un objeto ResultSet(rs)

```
muestraData(rs);
```

Dentro del método `muestraData` usamos un objeto de la clase **ResultSetMetaData** con la intención de solicitar información meta de la consulta, como: cuantas columnas hay y como se llama cada columna:

```
ResultSetMetaData rmeta=r.getMetaData();
int numColumnas=rmeta.getColumnCount();//cuantas columnas
jtxaDatos.setText("");
for(int i=1;i<=numColumnas;i++)
{ //obtenemos el nombre de la columna
  jtxaDatos.append(rmeta.getColumnName(i)+"\t\t");
}
```

Para leer hasta la ultima fila usamos el siguiente código:

```
while(r.next()) // hasta el final del archivo
{ for(int j=1;j<=numColumnas;j++)
  { //Se extra el dato en la posicion j del registro
    jtxaDatos.append(r.getString(j)+"\t\t");
  }
  jtxaDatos.append("\n");
}
```

Para leer el dato de una columna y retornarlo como un dato de tipo String usamos el método **getString(indice)** del objeto ResultSet.

CLASES BASICAS PARA JDBC...

Existen muchos objetos de JDBC sin embargo estos tienen vital importancia para empezar a desarrollar nuestras aplicaciones para acceder a base de datos:

Connection:

Permite la conexión a la base de datos. Origina un canal entre nuestra aplicación y la base de datos y será siempre imprescindible en una aplicación que quiere acceder a una base de datos:

Statement:

Este objeto nos permitirá ejecutar una sentencia SQL para nuestra base de datos. Por ejemplo: Select, insert, update y delete.

CLASES BASICAS PARA JDBC...

ResultSet:

Si el objeto Statement ejecuta una sentencia select del SQL, entonces este devuelve un conjunto de resultados. Este conjunto de resultados es asignado y manipulado por un objeto ResultSet.

ResultSetMetaData:

Un objeto de esta clase tiene la información meta sobre el conjunto de resultados como por ejemplo cuantas columnas tiene la consulta, nombres de las columnas, los tipos de datos que guarda cada columnas, cuantas filas, etc.



Jdbc con Oracle

```
import java.sql.*;
class dbAccess
{
    public static void main (String args []) throws SQLException
    {
        try
        {
            Class.forName ("oracle.jdbc.driver.OracleDriver");
        }
        catch (ClassNotFoundException e)
        {
            e.printStackTrace();
        }

        Connection conn = DriverManager.getConnection
            ("jdbc:oracle:oci:@ORACLEBD", "user", "passw");
            // @TNSNames_Entry, userid, password

        Statement stmt = conn.createStatement();
        ResultSet rset =
            stmt.executeQuery("select BANNER from SYS.V_$VERSION");
        while (rset.next())
            System.out.println (rset.getString(1));    // Print col 1
        stmt.close();
    }
}
```



Jdbc con MySql

```
import java.sql.*;
public class Programa {
    public static void main(String args[]){
        try {
            //Cargar clase de controlador de base de datos
            Class.forName("com.mysql.jdbc.Driver");
            //Crear el objeto de conexion a la base de datos
            Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/ejemplo?user=root&password=");
            //Crear objeto Statement para realizar queries a la base de datos
            Statement instruccion = conexion.createStatement();
            //Un objeto ResultSet, almacena los datos de resultados de una consulta
            ResultSet tabla = instruccion.executeQuery("SELECT cod , nombre FROM datos");
            System.out.println("Codigo\tNombre");
            while(tabla.next())
                System.out.println(tabla.getInt(1)+"\t"+tabla.getString(2));
        }
        catch(ClassNotFoundException e){ System.out.println(e); }
        catch(SQLException e){ System.out.println(e); }
        catch(Exception e){ System.out.println(e); }
    }
}
```



Universidad **César Vallejo**

Licenciada por Sunedu
para que puedas salir adelante