Acceso a bases de datos con Java

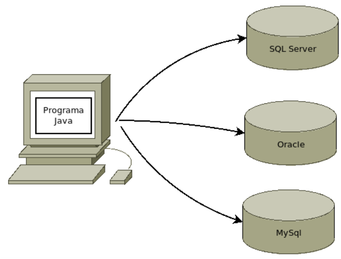
Biblioteca JDBC

Objetivo

Permitir que un programa Java pueda acceder a una
base de datos para enviarle sentencias SQL.

Distintos motores

Un objetivo principal es el de poder conectarse a distintos
motores sin modificar el código del programa.



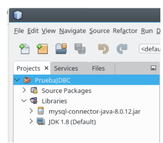
Drivers

El fabricante de cada motor de bases de datos
debe programar un conjunto de clases denominado
"Driver JDBC"

Las clases de cada driver son derivadas de clases
de la biblioteca JDBC. El programador usa sólo
las clases bases, pero se instancia la derivada correspondiente al motor seleccionado.

Los drivers son ofrecidos por cada fabricante en forma gratuita y generalmente se descargan
de internet.

Los drivers están empaquetados en archivos .jar y
para ser usados debe agregarse al proyecto una referencia de librería



Este proyecto tiene agregada en "Libraries" la referencia al driver JDBC de MySql

DriverManager y Clase Connection

Los drivers ofrecen implementaciones a clases abstractas definidas en la biblioteca estándar

Si el codigo de los programas debe instanciar esas clases derivadas, los programas quedan acoplados con un driver en particular, ya que el operador new requiere una clase concreta.

Para evitar ese acoplamiento, se dispone de la clase DriverManager, que se encarga de crear las instancias de la clase Connection

```
Connection conn = DriverManager.getConnection("cadena de conexion", "usuario", "contraseña");
```

Clase Statement

Permite enviar sentencias SQL (DML y DDL) a la base de datos conectada con un objeto Connection

Las instancias de la clase Statement no son creadas con new, sino que la clase connection las crea con el método createStatement

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

Para ejecutar sentencias de selección provee el método executeQuery(), mientras que para cualquier otra sentencia provee el método executeUpdate().

```
st.executeQuery("select * from tabla" );
st.executeUpdate("insert into tabla values (...)")
```

executeUpdate devuelve la cantidad de filas modificadas.

Clase ResultSet

Las consultas de selección devuelven un conjunto de filas que se recorre con un objeto de la clase ResultSet

Un ResultSet se encuentra en todo momento apuntando a una fila del conjunto de filas.

Para avanzar de una fila a otra se utiliza el método next(). Cuando puede avanzar, next() devuelve cuando se pasa de la última fila, next() devuelve false.

Para obtener los datos de la fila apuntada se utilizan los métodos getInt, getFloat, getString, etc.

Estos métodos reciben como parámetro el número de columna (empezando en 1) o el nombre de la columna (o su alias)

Clase PreparedStatement

Cuando una consulta SQL debe recibir diferentes valores cada vez que se la ejecuta, siempre es preferible utilizar consultas parametrizadas

Sin las consultas parametrizadas se necesita concatenar cada dato variable con porciones de código SQL.

La clase PreparedStatement es derivada de Statement y permite incluir parámetros dentro de las consultas.

```
PreparedStatement pst = conn.prepareStatement("insert into tabla values (?,?)");
pst.setString(1,"Juan");
pst.setString(2,"Perez");
pst.execute();
```

Diagrama de clases