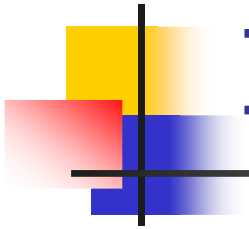
Base de Datos JDBC

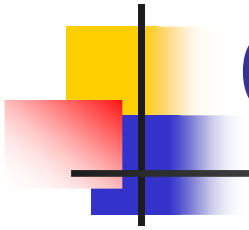
Unidad: 1
Laboratorio de Programación

Universidad Nacional de la Patagonia Austral
Unidad Académica Río Gallegos



Indice

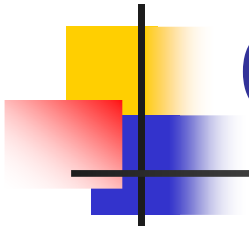
- Instalación de JDBC.
- Tipos de controladores.
- Conexión con la base de datos.
- Definir y ejecutar sentencias SQL.
- Solicitud de información. Obtención de los resultados.
- Actualización de filas. Inserción y actualización de datos.
- Sentencias preparadas.
- Navegando por los conjuntos de resultados.
- Transacciones. Autocommit. Commit y Rollback.



Qué es JDBC? (1)

- JDBC es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación 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 *[Wikipedia].*

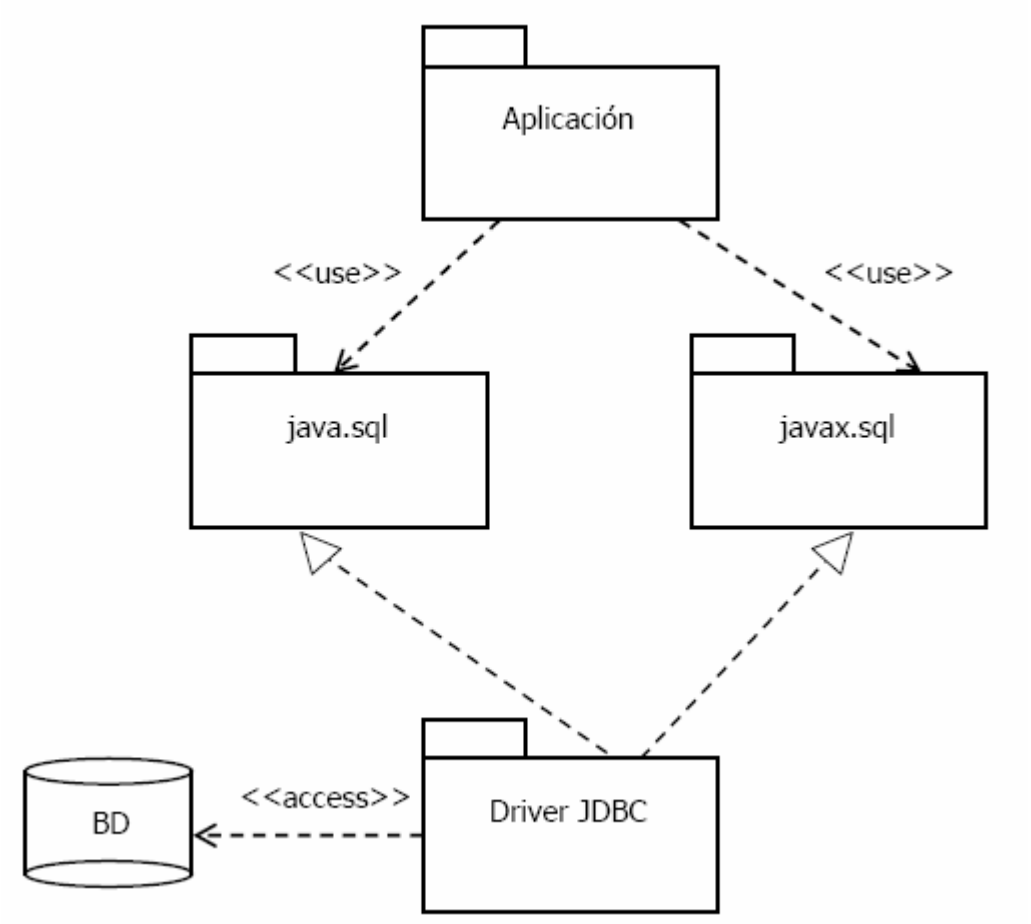
- Por tanto JDBC permite:
  - Conectar con una fuente de datos, normalmente una base de datos.
  - Enviar consultas y actualizaciones a la base de datos.
  - Procesar los resultados obtenidos de la base de datos en respuesta a una consulta



Qué es JDBC? (2)

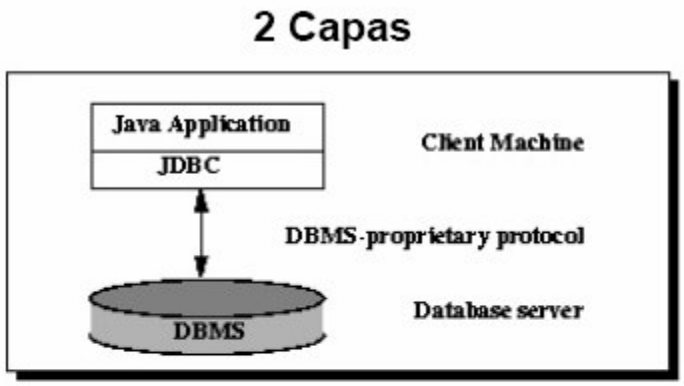
- Podemos trabajar con diferentes sistemas de gestión de bases de datos: ORACLE, ACCESS, MySQL, etc
- Sólo necesitamos una librería de conexión a esa base de datos.
- ODBC es el equivalente en Microsoft.
- Para poder conectarse a la BD y lanzar queries, es preciso tener un driver adecuado para ello
  - Un driver suele ser un fichero **.jar** que contiene una implementación de todos los interfaces del API de JDBC
  - Nuestro código nunca depende del driver, dado que siempre trabaja contra los paquetes **java.sql** y **javax.sql**

Driver JDBC

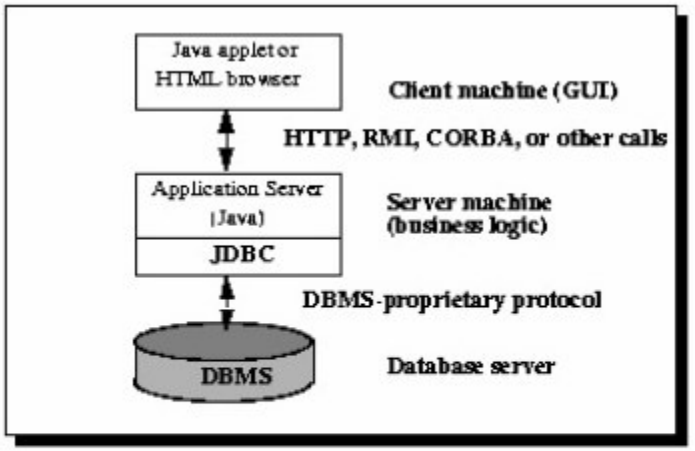


Qué es JDBC? (3)

- La arquitectura JDBC:

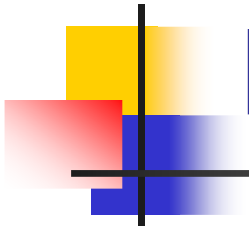


Modelo Cliente / Servidor



Modelo / Vista / Controlador.

Layers



Enlaces

- Sitio de JDBC de Sun:

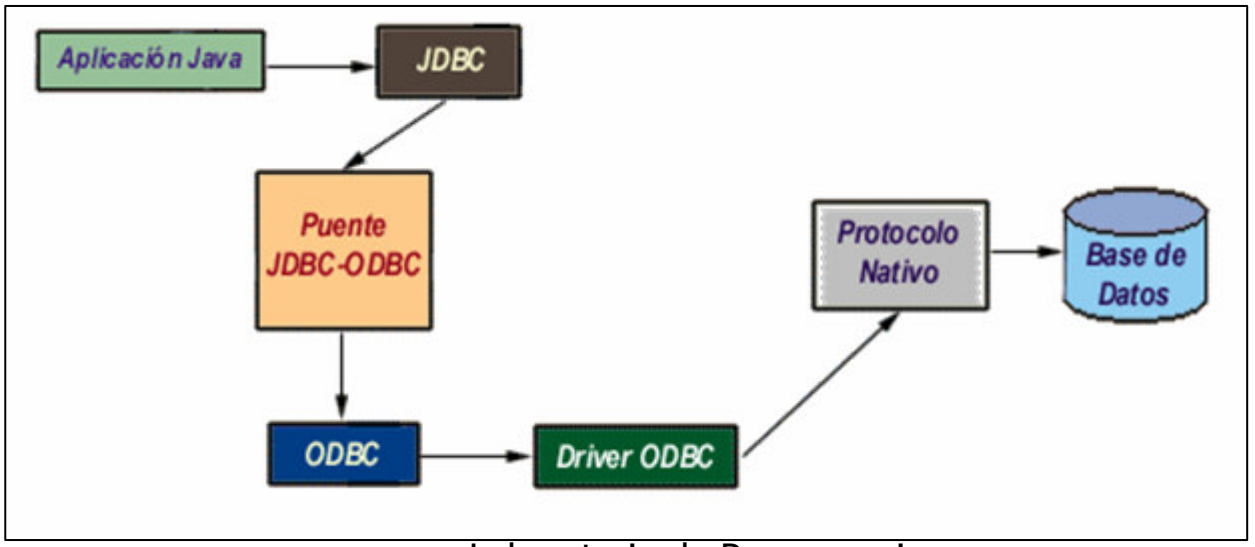
  http://java.sun.com/products/jdbc/
- Tutorial JDBC:

  http://java.sun.com/docs/books/tutorial/jdbc/
- Drivers JDBC:

  http://industry.java.sun.com/products/jdbc/drivers

Tipo de drivers (1)

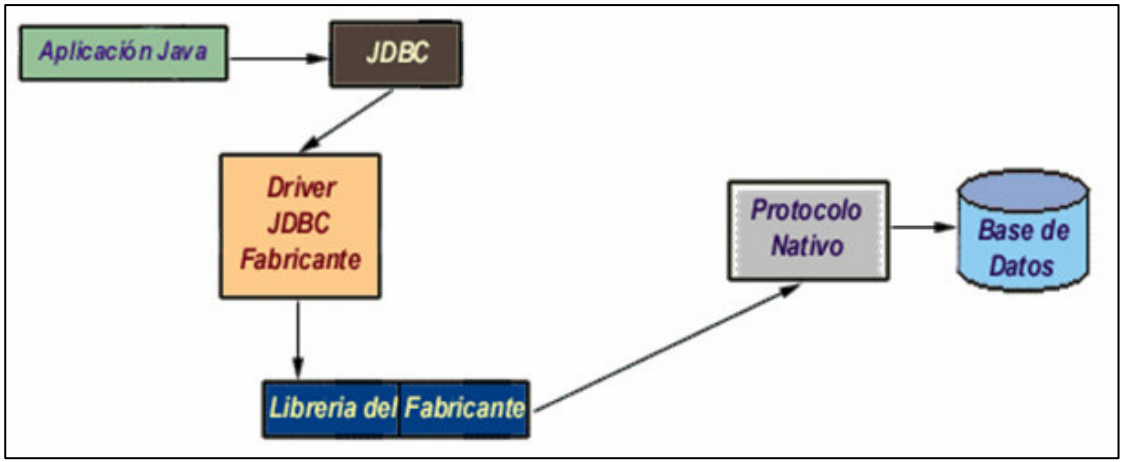
- Puente JDBC/ODBC
  - Facilidad configuración
  - Más lento



Tipo de drivers (2)

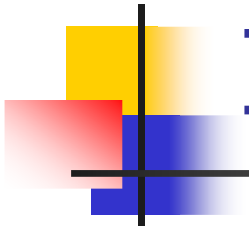
Java Binario

- No hay capa ODBC
- Necesario *Driver* del fabricante en el cliente



Otros:

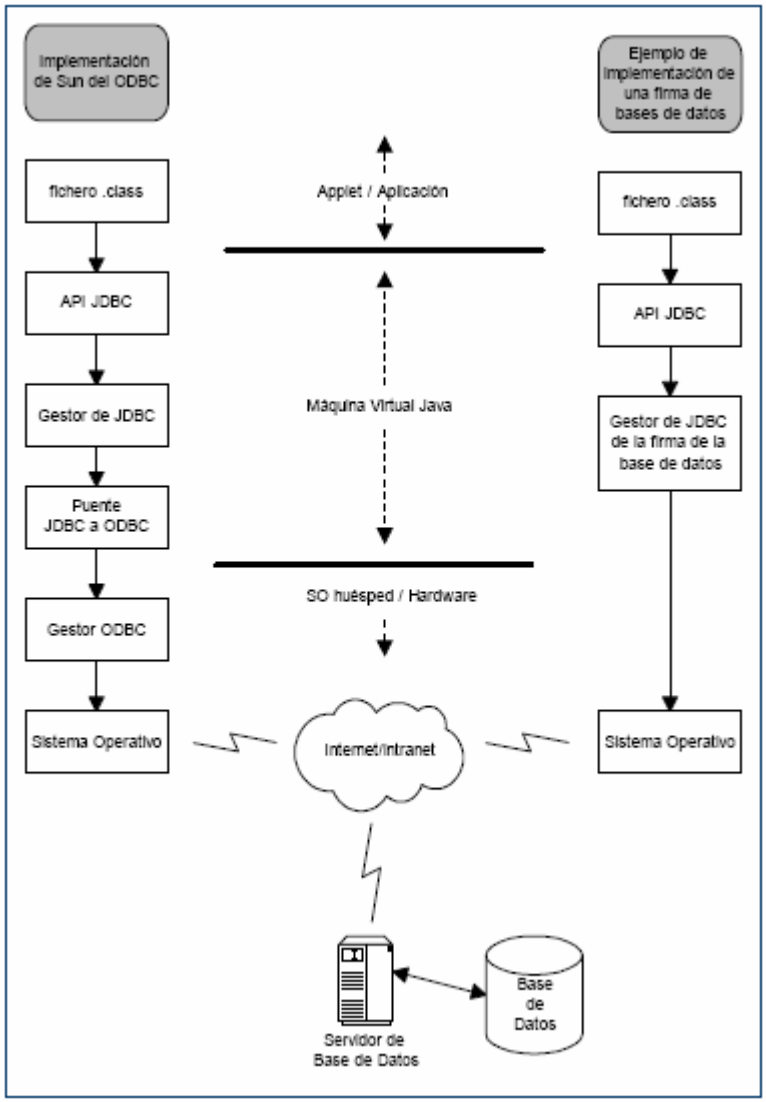
- 100% JAVA Protocolo Nativo
- 100% JAVA Protocolo Independiente



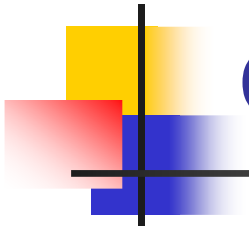
Independencia de la BD

- Idealmente, si nuestra aplicación cambia de BD, no necesitamos cambiar el código; simplemente, necesitamos otro driver
- Sin embargo, desafortunadamente las BDs relacionales usan distintos dialectos de SQL (a pesar de que en teoría es un estándar)
  - Tipos de datos: varían mucho según la BD
  - Generación de identificadores: secuencias, autonumerados, etc.

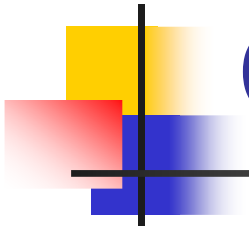
Comunicación con la BD



Qué necesitamos para trabajar con JDBC

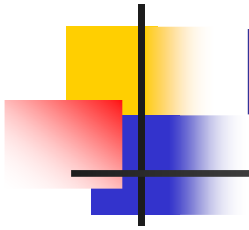


- Tener instalado Java y el driver JDBC en nuestra máquina.
  - Esto se soluciona instalando una máquina virtual:
    - Descargar el último JDK de Sun e instalar, incluye los paquetes java.sql y javax.sql.
- Instalar el Sistema Gestor de Bases de Datos (MySQL):
  - Descargar e instalar MySQL
    - http://dev.mysql.com/downloads/mysql/5.0.html
  - Descargar e instalar herramientas gráficas para MySQL
    - http://dev.mysql.com/downloads/gui-tools/5.0.html
- En función del Sistema Gestor de Bases de Datos, instalar el driver apropiado (MySQL):
  - Descargar el driver JDBC para MySQL.
    - http://dev.mysql.com/downloads/connector/j/5.0.html



Configuración Eclipse JDBC

- Instalar el driver
  - Bajar el driver de MySQL JDBC driver desde la pagina web
  - Descomprimir mysql-connector-xxx.jar
  - Añadir mysql-connector-xxx.jar al proyecto de Eclipse
    - Project → Properties → Java Build Path → Libraries → Add External JARs



Pasos de JDBC

- Siete pasos básicos en el uso de JDBC

1.- Cargar el controlador (driver)
2.- Definir el URL de la Conexión
3.- Establecer la conexión
4.- Crear un objeto Statement
5.- Ejecutar una consulta (query)
6.- Procesar los resultados
7.- Cerrar la conexión

Cargar del driver

- Para poder acceder a la base de datos…
  - En primer lugar, cargar el driver de conexión:

Facilitado por el vendedor

```
…
try {
        Class.forName ("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException e) {
        e.printStackTrace();
…
```

- O también…

```
…
try {
        Class.forName ("com.mysql.jdbc.Driver").newInstance();
} catch (ClassNotFoundException e) {
        e.printStackTrace();
…
```

Conectar el driver al SGBD

- Para establecer una conexión:
  - Connection con = DriverManager.getConnection(url, nombreUsuario,password);
    - url -> ruta de acceso a los datos.
    - nombreUsuario -> Usuario para acceder a la base de datos.
    - password -> Clave de acceso.
- Ejemplo:

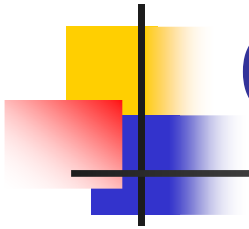
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Connection con = DriverManager.getConnection(
"jdbc:mysql://localhost/coffeebreak?user=root&password=");
…
con.close();
```

Crear la base de datos

- A partir de esta conexión creada y abierta podremos ejecutar sentencias con JDBC que son traducidas en sentencias SQL y ejecutadas en el SGBD.

- Definimos una tabla COFFEES

COF_NAME	SUP_ID	PRICE	SALES	TOTAL
Colombian	101	7.99	0	0
French_Roast	49	8.99	0	0
Espresso	150	9.99	0	0
Colombian_Decaf	101	8.99	0	0
French_Roast_Decaf	49	9.99	0	0



Creación de sentencias SQL…

- Tabla COFFEES:

```
CREATE TABLE COFFEES
(COF_NAME VARCHAR(32) NOT NULL,
SUP_ID INTEGER,
PRICE FLOAT,
SALES INTEGER,
TOTAL INTEGER,
PRIMARY KEY(COF_NAME)
)
```

- Para crear y ejecutar estas sentencias necesitamos:
  - Un objeto de tipo Statement.

```
Statement stm = con.createStatement();
stm.executeUpdate(sentenciaSQL);
stm.close()
```

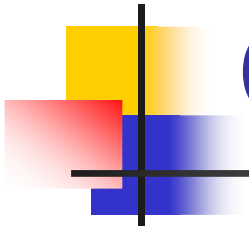
Creación de sentencias SQL…

- SentenciaSQL

```
String sqlCreacionCoffees = “CREATE TABLE COFFEES ” +
"(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +
                    "SALES INTEGER, TOTAL INTEGER … + …. ”;
Statement stm = con.createStatement();
stm.executeUpdate(sqlCreacionCoffees);
stm.close();
```

- OJO:
  - No terminar la sentencia con el carácter de terminación del SGBD determinado (por ejemplo ;).
  - El driver se encarga de poner ese carácter.
  - Así nuestro código es independiente del sistema de bd utilizado.

- Los tipos utilizados son tipos genéricos SQL y están definidos en java.sql.Types



Creación de sentencias SQL…

- Utilización de executeUpdate.
  - Utilizamos este método con sentencias SQL que modifiquen la definición de la base de datos (sentencias DDL = Data Definition language) o los datos (DML = Data Manipulation Language diferentes de SELECT):
    - Creación de tablas.
    - Eliminación de tablas.
    - Actualización de tablas.
    - Inserción, borrado y actualización de datos.
- Para sentencias que no supongan una modificación de los datos:
  - executeQuery.
  - SELECT.

Creación de sentencias SQL…

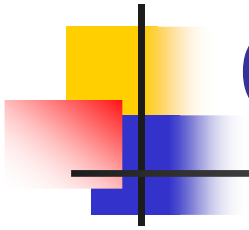
- Añadir datos a una tabla:
  - Se utiliza el objeto Statement con su método executeUpdate(SentenciaSQL).

```
stm.executeUpdate("INSERT INTO COFFEES "+
                  "VALUES ('Colombian', 101, 7.99, 0, 0)");
```

Se utilizan comillas simples

- La tabla COFFEES quedaría:

COF_NAME	SUP_ID	PRICE	SALES	TOTAL
Colombian	101	7.99	0	0



Creación de sentencias SQL…

- Recuperación de datos de las tablas. Por ejemplo:
  - SELECT COF_NAME, PRICE FROM COFFEES
- Lo primero que necesitamos es ejecutar la sentencia:
  - stm.executeQuery("SELECT COF_NAME, PRICE FROM COFFEES");
- Esta sentencia devuelve un objeto de tipo ResultSet:
  - ResultSet set = stm.executeQuery("SELECT COF_NAME … ");
- Este objeto tiene el conjunto resultante de la consulta. Paramovernos por las filas de ese conjunto resultante, utilizamos el concepto de cursor.

Creación de sentencias SQL…

COF_NAME	SUP_ID	PRICE	SALES	TOTAL
Colombian	101	7.99	0	0
French_Roast	49	8.99	0	0
Espresso	150	9.99	0	0
Colombian_Decaf	101	8.99	0	0
French_Roast_Decaf	49	9.99	0	0

- Para mover el cursor a lo largo de los registros, utilizamos el método next del objeto ResultSet.
- La primera vez que invocamos este método, el cursor se sitúa en la primera fila o primer registro, sucesivas llamadas a dicho método hacen avanzar el cursor.
- Veremos después que a partir de JDBC 2.0 podemos mover el cursor hacia delante, hacia atrás, a una fila determinada, a una relativa a la posición actual, …

Creación de sentencias SQL…

- Para acceder a cada uno de los campos de cada registro utilizamos métodos del tipo:
  - getXXXX(nombreCampo) del objeto ResultSet.
  - Por ejemplo, para acceder al campo COF_NAME, como es de tipo string, utilizamos rs.getString("COF_NAME").
  - Para acceder al precio -> rs.getFloat ("PRICE");
- Por tanto para recorrer un conjunto de datos hay 2 formas:

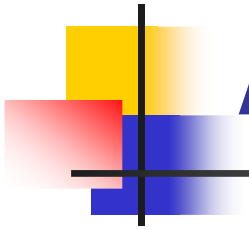
```
while (rs.next()){
        rs.getString("COF_NAME");
        rs.getFloat("Price");
        ...
}       //no case sensitive
```

```
while (rs.next()){
        rs.getString(1);
        rs.getFloat(2);
        ...
                    Más eficiente
}
```

- Observar en el API de JAVA los diferentes métodos que la clase ResultSet facilita para obtener datos de un registro.

Equivalencia de tipos

	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP	VARCHAR
getByte	x	x						
getShort	x	x						
getInt	x	x						
getLong	x	x						
getFloat	x	x						
getDouble	x	x						
getBigDecimal	x	x						
getBoolean	x	x						
getString	X	x	x	x	x	x	x	x
getBytes			X	X	x			
getDate	x	x				X		x
getTime	x	x					X	x
getTimestamp	x	x				x	x	X
getAsciiStream	x	X	x	x	x			
getUnicodeStream	x	X	x	x	x			
getBinaryStream			x	x	X			
getObject	x	x	x	x	x	x	x	x



Actualización de tablas

- Si queremos seguir modificando datos, por ejemplo actualizando tablas:
  - String updateString = "UPDATE COFFEES " + "SET SALES = 75 “ + "WHERE COF_NAME LIKE 'Colombian%' ";
  - stm.executeUpdate(updateString);
    - En este caso, el método devuelve un valor entero que indica el número de filas actualizadas

Ejemplo

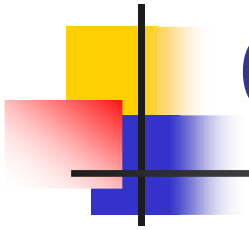
```
import java.sql.*;

public class EjemploJDBC {

    public static void main(String args[]){
        try {
        //Cargar driver especifico de base de datos
        Class.forName("com.mysql.jdbc.Driver");
        //Crear el objeto de conexion a la base de datos
        Connection conexión=
DriverManager.getConnection("jdbc:mysql:Ejemplo");
```

Ejemplo

```
            //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));
            conexion.close();
            }
            catch(ClassNotFoundException e){ System.out.println(e); }
            catch(SQLException e){ System.out.println(e); }
            catch(Exception e){ System.out.println(e); }
        }
    }
```

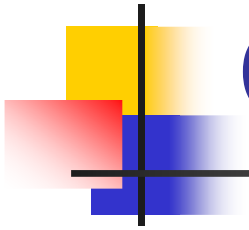
Correspondencia entre tipos Java y SQL estándar

Tipo Java	Tipo SQL
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER
long	BIGINT
float	REAL
double	DOUBLE
java.math.BigDecimal	NUMERIC
String	VARCHAR o LONGVARCHAR
byte[]	VARBINARY o LONGVARBINARY
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP



Conceptos SQL (1)

- **SQL** :lenguaje estándar para comunicarse con cualquier DBMS. Se divide en dos partes:
  - **DDL**: permite definir tablas en la Base de Datos
  - **DML**: permite manipular los datos almacenados



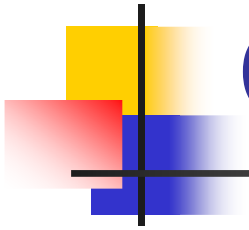
Conceptos SQL (2)

- Consulta SELECT sobre una tabla:

  SELECT Código,Título,Precio FROM Libros WHERE Precio > 35 ORDER BY Código ASC

- Consulta SELECT que regresa todas las columnas y filas:

  SELECT * FROM Libros



Conceptos SQL (3)

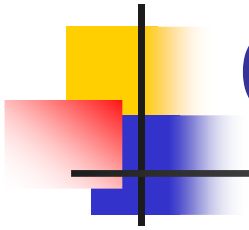
- Agregar registros: (sólo uno)

  INSERT INTO Libros (Código,Descripción,Precio) VALUES ('WARP','War and Peace','14.95')

- Agregar registros múltiples:

  INSERT INTO Archivo

  SELECT * FROM Pedidos WHERE Deuda=0

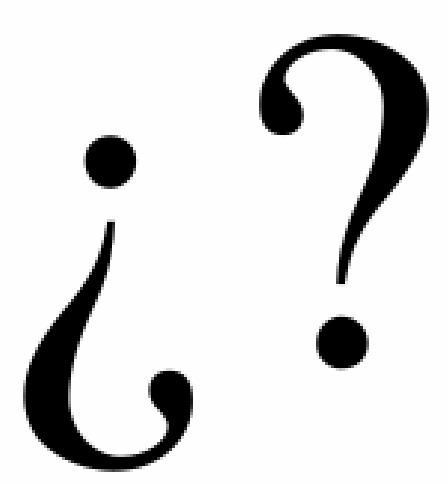


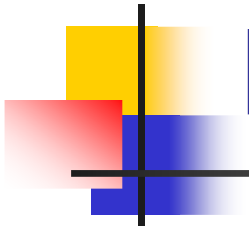
Concepto SQL (4)

- Actualizar registros:
  UPDATE Libros
  SET Título='War and Peace', Precio='14.95'
  WHERE Código='WARP'

- Actualizar registros múltiples:
  UPDATE Libros
  SET Precio='36.95'
  WHERE Precio='36.50'

Consultas...





Próxima clase

- Consultas y actualizaciones mediante sentencias preparada (PreparedStatement)
- Manejo de transacciones
- Ejercitación