Manejo de CONECTORES

UT2

Manejo de Conectores

Los orígenes de los datos a manejar pueden ser diversos:

- Una base de datos relacional, local o remota
- Una hoja de cálculo
- Un fichero de texto
- Un servicio de información online…

Desde los programas necesitaremos software que realice la conexión desde nuestro programa con las fuentes de datos.

En esta unidad, estudiaremos los **conectores** que necesitamos para acceder desde nuestros programas Java a ***Bases de Datos Relacionales***.

Manejo de Conectores

Bases de datos embebidas.

Recuerda algunos de los inconvenientes (y ventajas) de los Sistemas Gestores de Bases de Datos:

Ventajas de las BD

- Independencia lógica y física de los datos
- Consistencia de los resultados
- Mejor disponibilidad de los datos al centralizar la información
- Mayor eficiencia en la recogida y validación de los datos de entrada
- Reducción del espacio de almacenamiento

Inconvenientes de las BD

- Instalación costosa
- Personal especializado
- Implantación larga y difícil
- Falta de rentabilidad a corto plazo
- Ausencia real de normas

Si no necesitamos almacenar un gran volumen de información, podríamos usar una BD embebida, donde el motor esté **incrustado** en la aplicación y sea exclusivo para ella. La BD se inicia cuando se ejecuta la aplicación y termina cuando se cierra.

Manejo de Conectores

Algunas de estas BD son:

SQLite

- SGBD multiplataforma escrito en C
- Motor muy ligero
- Las BD se almacenan en formato ficheros
- Es un proyecto de dominio público

Apache Derby

- BD Relacional de código abierto
- Implementado en Java
- Tamaño reducido
- Soporta estándares SQL
- Aporta controlador JDBC

HSQLDB

- HyperThreaded SQL Database
- Escrito en Java
- Puede mantener la BD en memoria o en disco
- La incluye OpenOffice

Oracle Embedded

- De Oracle
- Ofrece elementos especializados:
  - Oracla Database 11g
  - Oracle TimesTen in Memory Database
  - Oracle Berkeley DB

H2

- SGBD relacional programado en Java
- Software libre bajo licencia Mozilla o Eclipse

DB4O

- DataBase for Objects
- Se puede utilizar embebida o en aplicaciones cliente-servidor
- Para entornos Java y .Net
- Licencia dual GPL/comercial

SQL Server Mobile

- BD de Microsoft compatible con toda su tecnología
- BD compacta

Manejo de Conectores

SQLite.

La Biblioteca implementa la mayor parte el estándar SQL-92.

Los programas que utilizan SQLite lo hace a través de llamadas simples a subrutinas y funciones. Se puede utilizar desde programas C/C++, PHP, Visual Basic, Perl, Delphi, Java, Kotlin, etc, etc, etc … y embebebida en dispositivos móviles.

La instalación es sencilla y permite "*transportar*" el motor de la BD y el fichero de la BD.

http://www.sqlite.org/download.html

Para Windows descarga el fichero ZIP con la última versión
https://www.sqlite.org/2021/sqlite-tools-win32-x86-3360000.zip

Manejo de Conectores

Otras Bases de Datos.

Oracle.

Oracle ofrece un conjunto de soluciones amplio al desarrollo software. Aunque no sigue las mismas características que las bases anteriores, una de las BD es Oracle 11g Edicion Express que podremos instalar en un equipo modesto.

http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html

Fichero para instalación y .jar para acceso desde java

Manejo de Conectores

PostgreSQL.

PostgreSQL permite trabajar también con grandes volúmenes de datos y a diferencia de Oracle es *open source*. Ambas utilizan conceptos similares en Schemas, Tablespaces e índices, pero Oracle es mucho más robusta y ofrece mayor seguridad.

Sin embargo PostgreSQL está haciendo una fuerte competencia tanto a sistemas propietarios como Oracle, como a los libres MySQL o MariaDB. Puedes analizar la comparación en:

https://db-engines.com/en/system/MySQL%3BOracle%3BPostgreSQL

https://www.postgresql.org/download/
https://jdbc.postgresql.org/download.html

Para instalación y .jar para acceso desde java

Manejo de Conectores

Protocolos de acceso a Bases de Datos.

- ***ODBC*** (Open Database Conectivity): define una API que pueden usar las aplicaciones para abrir una conexión con una BD, enviar consultas…

- ***JDBC*** (Java Database Conectivity): define una API que pueden usar los programas Java para conectarse a servidores de base e datos relacionales.

- ***OLE-DB*** (Object Linking and Embedding for Database): API en C++ de Microsoft para orígenes de datos que no son Bases de Datos (con objetivo parecido a los ODBC)

Manejo de Conectores

Acceso a datos mediante ODBC.

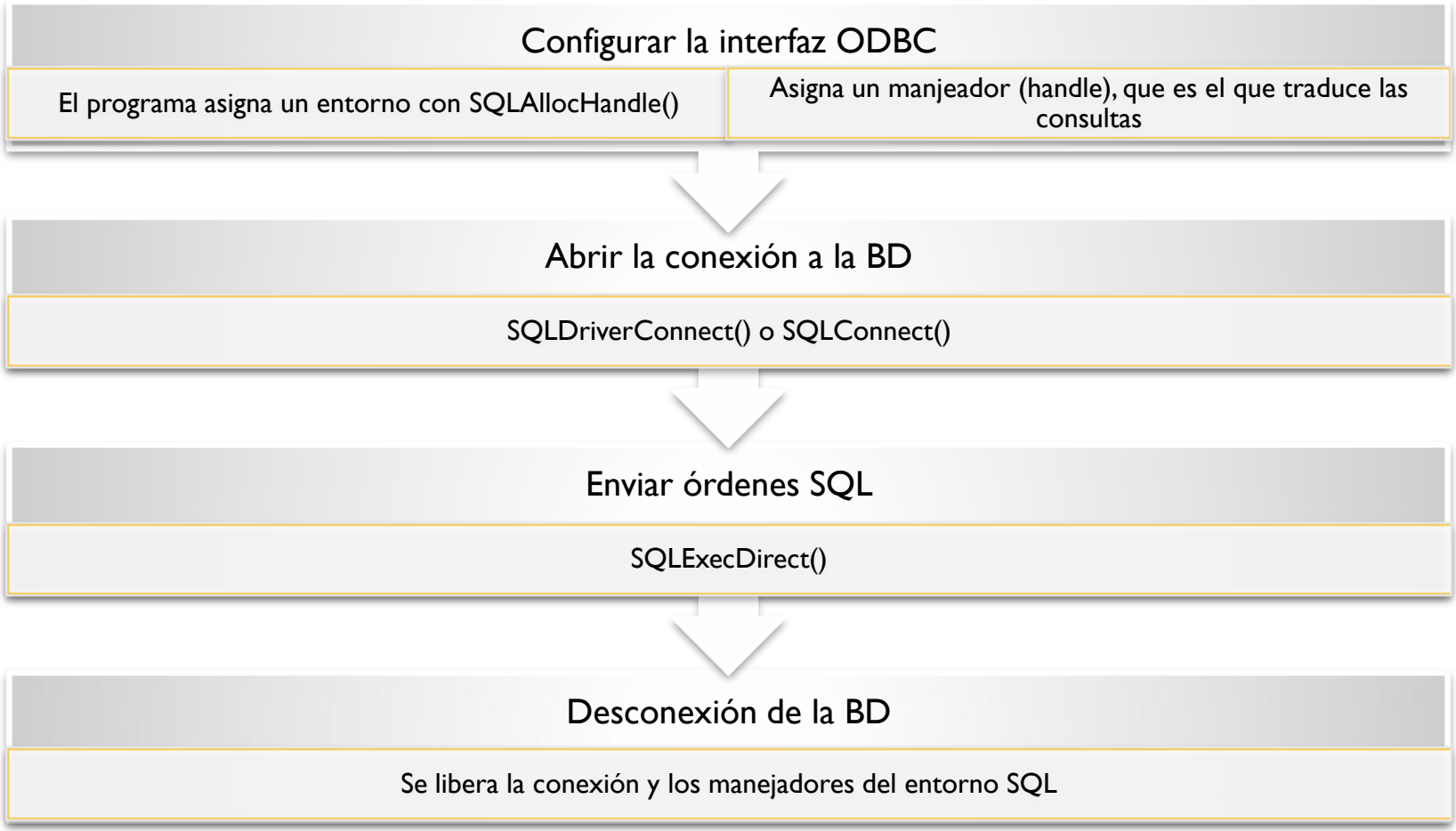
ODBC es un estándar de acceso a bases de datos desarrollado por Microsoft con el objetivo de posibilitar el acceso a cualquier dato desde cualquier aplicación, sin importar el SGBD que almacena los datos.

Cada SGBD proporciona una biblioteca que se debe enlazar con el programa cliente.

La API ODBC usa una interface en lenguaje C y no se recomienda para un uso directo desde Java, por inconvenientes en seguridad, implementación, robustez y portabilidad de las aplicaciones.

Manejo de Conectores

Pasos para usar ODBC:

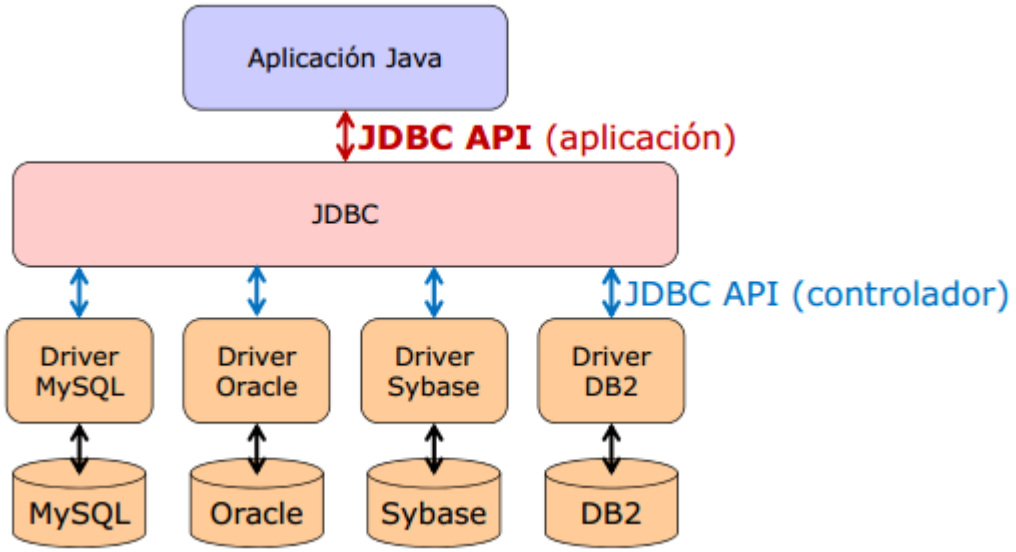


Manejo de Conectores

Acceso a datos mediante JDBC.

JDBC proporciona una librería estándar para acceder a fuentes de datos, normalmente BD relacionales que usan SQL.

JDBC dispone de una interfaz distinta para cada BD, es lo que llamamos ***driver*** o **conector**.



Manejo de Conectores

JDBC define varias interfaces para realizar las operaciones con BD y a partir de ellas se derivan las clases correspondientes, que estarán definidas en el paquete ***java.sql***

Clase e Interface	Descripción
Driver	Permite conectarse a una BD
Connection	Representa una conexión con una BD. Una aplicación puede tener más de una conexión.
DatabaseMetadata	Proporciona información de la BD
Statment	Ejecuta sentencias SQL sin parámetros
ResultSet	Contiene las filas resultantes de una sentencia SELECT
ResultSetMetadata	Permite obtener información sobre un ReultSet, como números de columnas y sus nombre…

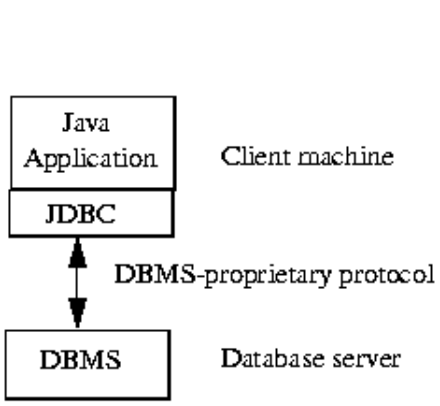
Manejo de Conectores

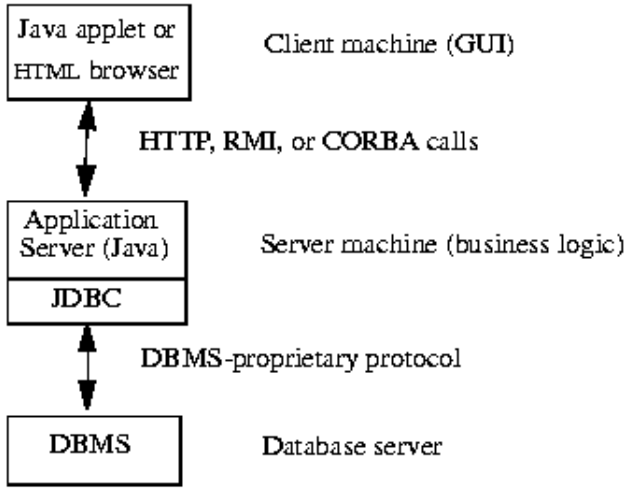
Arquitecturas JDBC.

La API JDBC es compatible con los modelos de dos y de tres capas.

- En el modelo de 2 capas es preciso que el JDBC se encuentre en el mismo lugar que la aplicación. El driver es el encargado de manejar las comunicaciones por la red
- En el modelo de 3 capas, los comandos se envían a una capa intermedia que es la que envía los comandos a la BD y los drivers no tienen que residir en la máquina cliente.

Modelos de dos capas y tres capas





Manejo de Conectores

Tipos de drivers.

El 1 y 2 exigen instalación en puesto cliente. Se recomienda el 3 y 4, que no exigen tampoco instalación en cliente..

Tipo 1: JDBC-ODBC Bridge

- JDBC-ODBC bridge plus ODBC driver

Tipo 2: Native

- Native-API partly-Java driver
- Escrito parcialmente en Java

Tipo 3: Network

- JDBC-Net pure Java driver
- Controlador Java que utiliza un protocolo (por ej.HTTP) para comunicarse con un servidor de BD.
- Traduce llamadas API en llamadas del protocolo

Tipo 4:Thin

- Native-protocol pure Java driver
- Controlador Java puro con protocolo nativo
- Traduce llamadas API de JDBC Java en llamadas propias del protocolo de red usado por el motor de BD.

Manejo de Conectores

Establecimiento de conexiones.

Vemos como se hacen las conexiones a diferentes BD, utilizando el mismo programa java, cambiando la carga del driver y la conexión a la BD.

(1) Conexión a SQLite.

- Hay que utilizar la librería sqlite-jdbc-X.Y.Z.jar (*la versión elegida*)
- La carga del driver, que para SQLite se llama org.sqlite.JDBC
  - *Class.forName("org.sqlite.JDBC");*
- Y la conexión a la BD:
  - *Connection conexion=DriverManager.getConnection ("jdbc:sqlite:C:/Users/SQLite/ejemplo.db");* (c: o ruta de la bd)

Manejo de Conectores

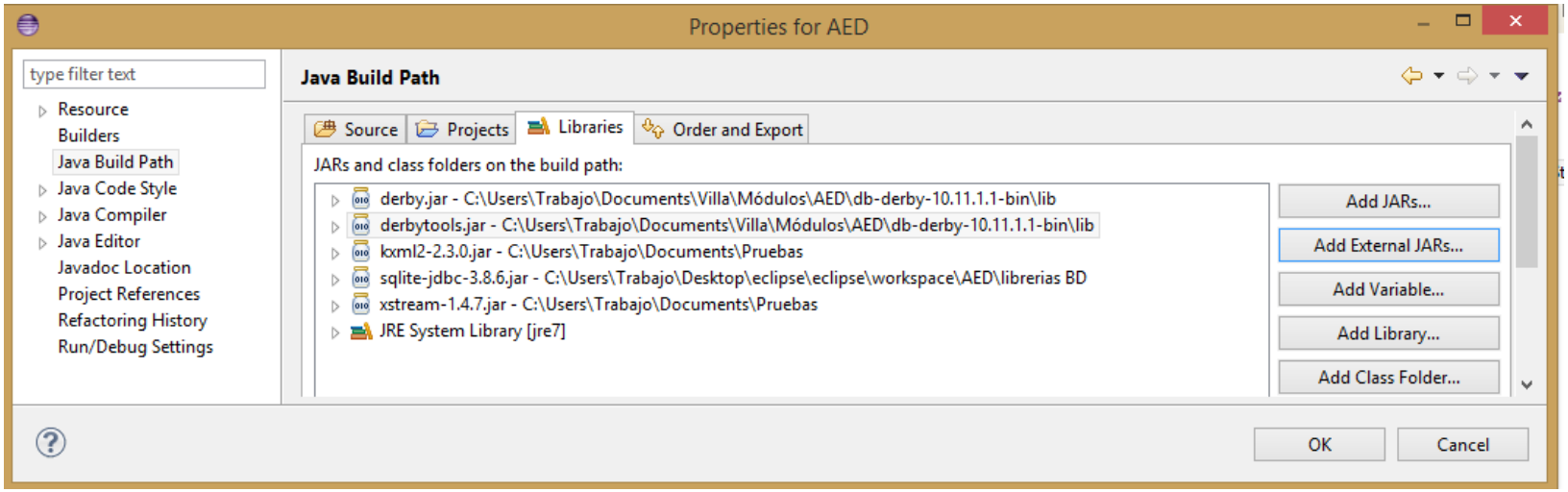
```
import java.sql.*;
public class MiSQLite {
        public static void main(String[] args) {
        try {
        Class.forName("org.sqlite.JDBC");                    //Cargamos el driver
                        //Se establece conexión con BD
        Connection conexion=DriverManager.getConnection ("jdbc:sqlite:C:/Users/Trabajo/Documents/Villa/ejemplo.db");
        Statement sentencia=conexion.createStatement();              //Se prepara una consulta
        ResultSet resul=sentencia.executeQuery("SELECT * FROM emple");

        while (resul.next()){
                        System.out.println(resul.getInt(1)+"  "+resul.getString(2));
        }
        resul.close();
        sentencia.close();
        conexion.close();
        }
        catch (ClassNotFoundException ec) {ec.printStackTrace();}
        catch (SQLException es) {es.printStackTrace();}
        }
}
```

***Observación**: Desde Eclipse importa la librería .jar en el paquete, o bien, si ejecutas comando, actualiza CLASSPATH Debe contener la ruta donde se encuentre el .class y el.jar de sqlite.*

Manejo de Conectores

En eclipse vamos a incluir los jar en las librerías del proyecto, en Build Path:



Manejo de Conectores

Conexión a Oracle Edición Express 11g. ***(ejemplo de versión)***

- Hay que utilizar las librerías de la instalación de Oracle, por ej: ojdb6.jar
- La carga del driver para Oracle
  - *Class.forName("oracle.jdbc.driver.OracleDriver");*
- Y la conexión a la BD:

  Connection conexion=DriverManager.getConnection

  ("jdbc:oracle:thin:@localhost:1521:XE","rita","rita");

  Donde el direccionamiento a la BD se hace mediante la cadena thin:@DireccionIP o nombre máquina:puerto de escucha:Nombre BD

  Seguido por el usuario y la contraseña para el acceso a la tabla.

  El **puerto por defecto** es 1521.

THIN *es un driver de tipo cliente, utilizado por Oracle, escrito en java y sin excesivas dependencias de versiones. Basta con añadir el .jar y normalmente no hay que "tocar" mucho la configuración del tomcat.*

Manejo de Conectores

```
public class MiBD {
    public static void main(String[] args) {
        try {
    //Cargamos el driver
            //SQLite
                //Class.forName("org.sqlite.JDBC");
            //Derby
                //Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
            //HSQLDB
                //Class.forName("org.hsqldb.jdbcDriver");
            //Oracle
                Class.forName("oracle.jdbc.driver.OracleDriver");


    //Se establece conexión con BD
            //SQLite
            //  Connection conexion=DriverManager.getConnection
            //      ("jdbc:sqlite:C:/Users/Trabajo/Documents/Villa/ejemplo.db");
            //Derby
            //  Connection conexion=DriverManager.getConnection
            //      ("jdbc:derby:C:/Users/Trabajo/Documents/pruebasderby");
            //HSQLDB
            //  Connection conexion=DriverManager.getConnection
            //          ("jdbc:hsqldb:file:C:/Users/Trabajo/Documents/mihsqldb");
            //Oracle
                Connection conexion=DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:XE","rita","rita");

                //Se prepara una consulta
            Statement sentencia=conexion.createStatement();
            ResultSet resul=sentencia.executeQuery(
                    "SELECT * FROM emple WHERE deptno>10 and nombre like '%ón'");

            while (resul.next()){
                System.out.println(resul.getInt(1)+"  "+resul.getString(2));
            }
            resul.close();
```

Manejo de Conectores

MySQL

MySQL es una de las bases de datos de código abierto de mayor aceptación. MySQL es un sistema gestor de bases de datos muy usado por su simplicidad y notable rendimiento. Es de libre distribución en Internet bajo licencia GPL.

MySQL está disponible para múltiples plataformas.

Se ha presentado la instalación de MySQL en Windows y se revisa el proceso de instalación en una máquina Linux (Ubuntu, Debian…).

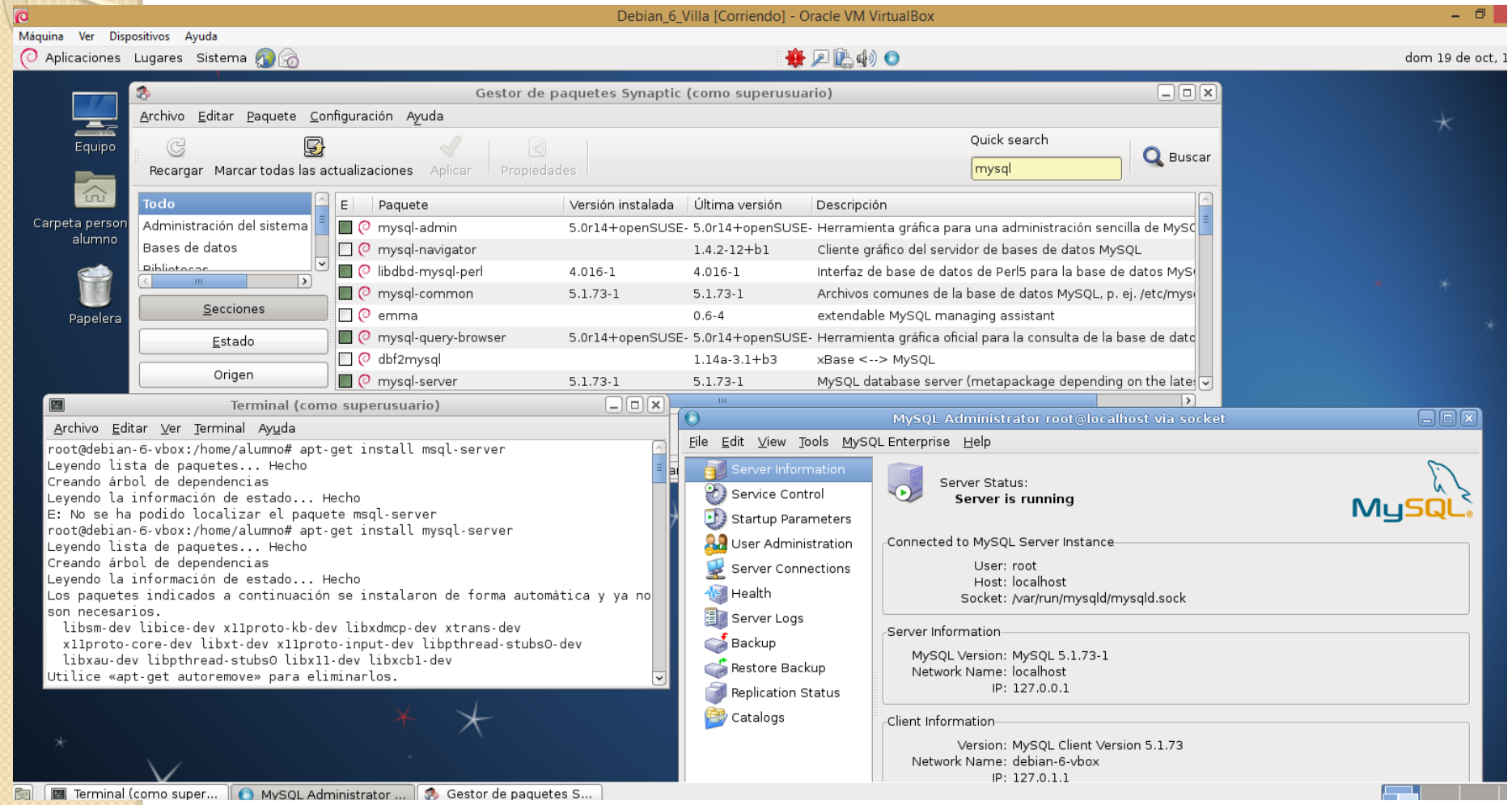
1º) En la máquina Linux, instala mysql-server (*apt-get install mysql-server*)

Asegúrate de poner y recordar la clave de root.

2º) Descarga con la herramienta Synaptic el gestor gráfico de MySQL

3º) En el menú de programación accedemos a la administración de MySQL y preparamos el entorno.

Manejo de Conectores



Manejo de Conectores

Para la conexión desde java, deberás incorporar los .JAR según se trate de Windows o Linux, verificando las versiones.
Para la conexión desde máquina remota, el usuario de acceso debe contener % (cualquier máquina) y verificar cortafuegos

```
        //Class.forName("org.hsqldb.jdbcDriver");
      //Oracle
        //Class.forName("oracle.jdbc.driver.OracleDriver");
      //MySQL
        Class.forName("com.mysql.jdbc.Driver");

'Se establece conexión con BD
      //SQLite
      //  Connection conexion=DriverManager.getConnection
      //      ("jdbc:sqlite:C:/Users/Trabajo/Documents/Villa/ejemplo.db");
      //Derby
      //  Connection conexion=DriverManager.getConnection
      //      ("jdbc:derby:C:/Users/Trabajo/Documents/pruebasderby");
      //HSQLDB
      //  Connection conexion=DriverManager.getConnection
      //          ("jdbc:hsqldb:file:C:/Users/Trabajo/Documents/mihsqldb");
      //Oracle
      //  Connection conexion=DriverManager.getConnection
        //      ("jdbc:oracle:thin:@localhost:1521:XE","rita","rita");

      // MySQL
        Connection conexion=DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/AED","rita","rita");
        //Se prepara una consulta
      Statement sentencia=conexion.createStatement();
      ResultSet resul=sentencia.executeQuery(
              "SELECT * FROM emple WHERE deptno>10 and nombre like '%ón'");
```

Manejo de Conectores

Ejecución de sentencias de DESCRIPCIÓN de datos.

Si no conocemos la estructura y relaciones en la BD podemos obtener esta información a través de ***metaobjetos,*** que son objetos que proporcionan información sobre la base de datos.

Objeto DatabaseMetaData: tiene métodos que proporcionan información:

- *getMetaData():* obtiene información de la BD (nombre, driver, url, usuario…)
- *getTables():* devuelve objeto *ResulSet* con información de tablas y vistas
- *getColumns(): devuelve información de las columnas de una o varias tablas*
- *getPrimaryKeys():* lista de columnas que forman la clave primaria
- *getExportedKeys():* lista de las claves ajenas que utilizan la clave primaria
- *getImportedKeys():* devuelve la lista de claves ajenas
- *getProcedures():* lista de procedimientos almacenados

como puedes ver, no tienes que aprenderte los parámetros

A partir de un ResultSet, mediante la interfaz ResultSetMetaData: proporciona información sobre tipos y propiedades de columnas.

getColumnCount(), getColumnName(), getColumnTypeName(), isNullable(), getColumnDisplaySize()

Manejo de Conectores

Ejecución de sentencias de MANIPULACIÓN de datos.

Al crearse un objeto ***Statement*** se crea un espacio de trabajo para crear consultas SQL, ejecutarlas y para recibir los resultados de las consultas. Se pueden usar los métodos:

- *executeQuery():* para sentencias SELECT
- *executeUpdate():*cuando no se devuele un ResultSet, sino que se ejecutan sentencias DML (Insert, Update y Delete) y DDL (Create, Drop y Alter)
- *execute():* para sentencias que devuelven más de un ResultSet, procedimientos almacenados.

Manejo de Conectores

Procedimientos almacenados en la Base de Datos.

Los procedimientos almacenados consisten en un conjunto de sentencias SQL y del **lenguaje procedural** (PL/SQL) utilizado por el SGBD, que se pueden llamar por su nombre para realizar acciones sobre la BD. Pueden o no tener parámetros, y éstos pueden ser IN, OUT o INOUT. También existen funciones si devuelven un valor.

MySQL no admite parámetros OUT e INOUT.

La interfaz *CallableStatement* permite La llamada a los procedimientos almacenados.