Capítulo 4

Base de Datos MySQL

4.1 MySQL

La base de datos que contiene información relativa al servicio de directorio se ha realizado en lenguaje SQL utilizando concretamente el servidor de bases de datos que ofrece MySQL.

La base de datos al completo es diseñada y gestionada mediante el gestor de bases de datos que ofrece MySQL, pero las distintas consultas que deba realizar el cliente serán llevadas a cabo por el servicio web.

Para llevar a cabo la consulta a la base de datos se ha utilizado el estándar JDBC que ofrece conexión a bases de datos desde servidores realizados en lenguaje Java.

4.1.1 Características de mySQL

El estándar MySQL constituye la base de datos de software libre más popular del mercado. Es desarrollada, distribuida y costeada por el grupo de empresas MySQL AB, el cual fue fundado por los creadores de MySQL con el objetivo de perpetuar esta base de datos.

MySQL es un sistema de gestión de base de datos que ofrece los mecanismos para añadir, acceder y procesar los distintos datos almacenados en una base de datos.

MySQL ofrece una base de datos relacional en lenguaje SQL, la cual almacena los datos en tablas de datos separadas almacenadas todas ellas en un mismo espacio de almacenamiento. Este hecho ofrece unas buenas características de velocidad y flexibilidad.

MySQL es software libre, lo cual significa que cualquiera puede hacer uso del código fuente que constituye a MySQL para usarlo libremente o incluso modificarlo sin restricciones.

Los motivos por los que MySQL se ha hecho tan popular, además de lo expuesto anteriormente, residen en el hecho de que supone un Servidor de bases de datos cuyas características de velocidad, fiabilidad y facilidad de uso son extremadamente atractivas y competitivas respecto del resto de soluciones existentes en el mercado.

El software de MySQL ofrece un modelo de cliente/servidor consistente en un servidor SQL multihilo que es capaz de soportar diferentes clientes, librerías, herramientas administrativas y APIs.

4.1.2 Funcionamiento de mySQL

Las capacidades de MySQL son extremadamente amplias, ya que este servidor de bases de datos cuenta con un gran potencial de funcionamiento. El objetivo de este punto es el de mostrar el uso de MySQL para crear y usar una sencilla base de datos. MySQL ofrece un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar consultas y ver los resultados. Todas estas operaciones se pueden llevar a cabo tanto desde línea de comando en un *shell*, como desde un programa front-end gráfico que presente una interfaz gráfica de control.

Puesto que es imposible que se describan aquí en detalle muchas de las características cubiertas por MySQL, nos centraremos sólo en aquellas de relevancia e invitamos consultar el manual de MySQL para obtener más información al respecto de funcionalidades específicas.

4.1.2.1 Establecimiento de una conexión con el servidor MySQL

Para realizar la conexión con el servidor MySQL solamente es necesario indicarle el nombre o la dirección IP del dicho servidor, el *Login* o nombre de usuario y el *password* o clave. Así se teclearía en el *shell*:

```
shell> mysql -h NombreDelServidor -u NombreDeUsuario -p
```

Tras esto se nos pedirá la contraseña para este nombre de usuario y si la conexión al servidor MySQL se pudo establecer de manera satisfactoria recibiremos el mensaje de bienvenida y estaremos en el *prompt* de mysql:

```
shell>mysql -h COFFEEBREAK -u root -p

Enter password: ******

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5563 to server version: 3.23.41

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Este *prompt* indica que Mysql está listo para recibir comandos.

Después de que nos hemos conectado de manera satisfactoria, podemos desconectarnos en cualquier momento al escribir "*quit*", "*exit*", o presionar CONTROL+D.

4.1.2.2 Creación de una tabla

Supongamos que nuestra base de datos está siendo utilizada por el propietario de un pequeño café llamado "The Coffee Break", donde los granos de café se venden por kilos y el café líquido se vende por tazas.

Supondremos que el sólo se necesitan dos tablas, una para los tipos de café y otra para los suministradores.

Lo primero que vamos a hacer es crear las tablas de la base de datos:

La primera de las tablas es *COFFEES*, que contiene la información esencial sobre los cafés vendidos, esta tabla incluye: los nombres de los cafés, sus precios, el número de euros vendidos la semana actual y el número de euros vendidos hasta la fecha. Esta tabla sería:

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

La columna que almacena el nombre del café es *COF_NAME*, y contiene valores con el tipo *VARCHAR* y una longitud máxima de 32 caracteres, este campo será único para cada tipo de café por lo que puede servir como clave primaria. La segunda columna es *SUP_ID* que contiene un número que identifica al suministrador del café y es de tipo *INTEGER*. La tercera columna, llamada *PRICE*, almacena valores del tipo *FLOAT*. La columna SALES almacena valores de tipo *INTEGER* e indica el número de euros vendidos durante la semana actual, y por último, la columna final de nombre *TOTAL* contiene un *INTEGER* que contiene el número total de euros vendidos hasta la fecha.

Para crear esta tabla, tecleamos en el shell:

```
mysql> CREATE TABLE COFFEES (
    -> COF_NAME VARCHAR(32), SUP_ID INTEGER,
    -> PRICE FLOAT, SALES INTEGER,
    -> TOTAL INTEGER);
Query OK, 0 rows affected (0.02 sec)

mysql>
```

La segunda tabla *SUPPLIERS* tiene información sobre cada uno de los suministradores, se crearía exactamente igual que la anterior, pero con la siguiente estructura:

SUP_ID	SUP_NAME	STREET	CITY	STATE	ZIP
101	Acme, Inc.	99 Market Street	Groundsville	CA	95199
49	Superior Coffee	1 Party Place	Mendocino	CA	95460
150	The High Ground	100 Coffee Lane	Meadows	CA	93966

La columna *SUP_ID* es la clave primaria de la tabla *SUPPLIERS* por lo que es un identificador único para cada uno de los suministradores de café. En la tabla *COFFEES* esta columna es la clave externa.

4.1.2.3 Introducción de datos en una tabla

Para introducir datos en una tabla lo haremos fila a fila, suministrando la información a almacenar en cada columna de la fila. Los valores insertados en las columnas se listarán en el mismo orden en que se declararon las columnas cuando se creó la tabla, que es el orden por defecto.

Para ver un ejemplo vamos a insertar una fila de datos con Colombian en la columna *COF_NAME*, 101 en *SUP_ID*, 7.99 en *PRICE*, 0 en *SALES*, y 0 en *TOTAL*:

```
mysql> INSERT INTO COFFEES
    -> VALUES ('Colombian', 101, 7.99, 0, 0);
```

4.1.2.4 Obtención de datos de una tabla

Para obtener los datos de una tabla, ya sea toda la tabla completa o bien una parte tenemos que realizar una sentencia SELECT que nos de los valores que queremos.

Si queremos obtener toda la tabla *COFFEES* completa tendríamos que teclear en el shell:

```
mysql> SELECT * FROM COFFEES;
```

En cambio si queremos obtener una lista de cafés y sus respectivos precios, ejecutaríamos:

```
mysql> SELECT COF_NAME, PRICE FROM COFFEES;

Su resultado sería:
+--------------------------------+----------+
| COF_NAME                       | PRICE    |
+--------------------------------+----------+
| Colombian                      | 7.99     |
| French_Roast                   | 8.99     |
| Espresso                       | 9.99     |
| Colombian_Decaf                | 8.99     |
| French_Roast_Decaf             | 9.99     |
+--------------------------------+----------+
 5 rows in set (0.00 sec)
```

Si ahora quisiésemos obtener los nombres y precios de todos los cafés de la tabla limitándonos a aquellos cafés que cuesten menos de 9.00 euros, ejecutaríamos:

```
mysql> SELECT COF_NAME, PRICE
    ->  FROM COFFEES
    ->   WHERE PRICE < 9.00;
```

4.1.2.5 Borrado en una tabla

Si quisiésemos borrar alguna columna de una tabla, alguna fila al completo o incluso una tabla entera usaríamos la instrucción *DELETE* de SQL con o sin sentencia *WHERE*, según sea el caso. Así un ejemplo sería:

```
mysql> DELETE SALES FROM COFFEES
    -> WHERE  PRICE < 9.00;
```

Donde se han eliminado de la tabla *COFFEES* todos los valores de la columna *SALES* en aquellas filas en las que *PRICE* es menor de 9 euros.

4.1.2.6 Actualización de tablas

Si por ejemplo queremos realizar una actualización de la columna *SALES* de la tabla *COFFEES* introduciendo el número de euros vendidos de cada tipo de café, ejecutaríamos la siguiente instrucción:

```
mysql> UPDATE COFFEES
    -> SET SALES = 75
    -> WHERE COF_NAME LIKE 'Colombian';
```

4.1.2.7 Ordenación de registros en una tabla

En los ejemplos anteriores las filas devueltas son mostradas sin ningún orden en particular. Para ordenar los resultados tenemos que usar la cláusula *ORDER BY*. Así si queremos ordenar algunos datos según el precio, haríamos:

```
mysql> SELECT COF_NAME, PRICE
    -> FROM COFFEES
    -> ORDER BY PRICE;

La salida será:

+----------------------------+----------+
| COF_NAME                   | PRICE    |
+----------------------------+----------+
| Colombian                  | 7.99     |
| French_Roast               | 8.99     |
| Colombian_Decaf            | 8.99     |
| French_Roast_Decaf         | 9.99     |
| Espresso                   | 9.99     |
+----------------------------+----------+
 5 rows in set (0.00 sec)
```

En las columnas de tipo caracter, el ordenamiento es ejecutado normalmente de sin hacer diferencia entre mayúsculas y minúsculas. Sin embargo, se puede forzar un ordenamiento distinguiendo mayúsculas de minúsculas al usar el operador *BINARY*.

Para ordenar en orden inverso, debemos agregar la palabra clave *DESC* al nombre de la columna que estamos usando en el ordenamiento:

4.1.2.8 Unión de dos tablas

En ocasiones es necesario utilizar más de una tabla para obtener ciertos datos que nos hacen falta. Por ejemplo, supongamos que se quiere una lista de los cafés que se le compran a la empresa *Acme, Inc*, para ello necesitamos información de las tablas *COFFEES* y *SUPPLIERS*. En este caso es necesario realizar un *join* o unión sobre dos tablas.

En este caso nos piden los nombres de los suministradores de la tabla *SUPPLIERS* y los nombres de los cafés de la tabla *COFFEES*, como ambas tablas tienen la columna *SUP_ID* que las relaciona podemos utilizar esta columna para realizar la unión, así:

```
mysql>SELECT COFFEES.COF_NAME
    -> FROM COFFEES, SUPPLIERS
    -> WHERE SUPPLIERS.SUP_NAME LIKE 'Acme, Inc.'
    -> and SUPPLIERS.SUP_ID = COFFEES.SUP_ID

Esto producirá la siguiente salida:

+----------------------------+
| COF_NAME                   |
+----------------------------+
| Colombian                  |
| Colombian_Decaf            |
+----------------------------+
2 rows in set (0.00 sec)
```

4.2 JDBC

El API JBDC no es más que un interfaz de acceso a un sistema de gestión de bases de datos relacionales RDBMS (Relational Database Management System) independiente de la plataforma sobre la que se encuentra la base de datos y del propio gestor de bases de datos utilizado.

Este API consiste en una serie de interfaces Java implementadas por un controlador o *driver* el cual se encarga de la traducción de llamadas a la base de datos expresadas en lenguaje Java al lenguaje que entiende la base de datos. De esta manera el programador puede abstraerse de la programación específica de la base de datos creando código en Java que funcionará para todos los RDBMS que cuenten con un *driver* JDBC con el que realizar la traducción.

Actualmente existen *drivers* JDBC para todos los sistemas de gestión de bases de datos mas populares como pueden ser Informix, Oracle, SQLServer, DB2, InterBase, SyBase, y otros de software libre como pueden ser mSql, mySql y Postgre.

4.2.1 Funcionamiento de JDBC

A lo largo de esta sección asumiremos que la base de datos con la que vamos a trabajar *COFFEEBREAK* ya existe ya que esta operación normalmente lo hace un administrador de bases de datos.

4.2.1.1 Establecimiento de una conexión con la base de datos

El primer paso que debemos realizar para interactuar con la base de datos es establecer una conexión con ella, esto implica dos pasos: cargar el *driver* y hacer la conexión.

- **Cargar el driver**

  Cargar el *driver* es muy sencillo, necesita solamente una línea de código. La documentación del *driver* nos dará el nombre de la clase a utilizar. Por ejemplo, si el nombre de la clase es *jdbc.DriverXYZ*, se cargaría el *driver* de la forma:

```
Class.forName("jdbc.DriverXYZ");
```

  Una vez cargado el *driver*, es posible hacer una conexión con un controlador de base de datos.

- **Hacer la conexión**

  El segundo paso, y último, para establecer una conexión es tener el *driver* apropiado conectado al controlador de base de datos. Para realizar esto utilizamos la siguiente línea de código:

```
Connection con = DriverManager.getConnection(url, "myLogin",
      "myPassword");
```

  En este código hemos indicado la URL a la que habría que conectarse para interactuar con la base de datos, el *login* o nombre de usuario y el *password* o clave. A la hora de escribir la URL la documentación nos dirá el subprotocolo a utilizar, es decir, qué URL debemos poner.

  Si el *driver* cargado reconoce la URL suministrada él mismo establecerá una conexión con el controlador de base de datos especificado en la URL.

  La clase *DriverManager* maneja todos los detalles del establecimiento de la conexión. La conexión devuelta por el método *DriverManager.getConnection*

es una conexión abierta que se puede utilizar para crear sentencias JDBC que pasen nuestras sentencias SQL al controlador de la base de datos.

4.2.1.2 Creación de una tabla

Para realizar la creación de una tabla antes es necesario crear un objeto *Statement*, que es el encargado de enviar las sentencias SQL que escribamos al controlador de la base de datos.

Simplemente creamos un objeto *Statement* y lo ejecutamos, suministrándole el método SQL apropiado con la sentencia SQL que queremos enviar. Para sentencias que crean o modifican tablas, el método a utilizar es *executeUpdate*.

La forma de ejecutarlo sería:

```
Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE COFFEES " +
        "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +
        "SALES INTEGER, TOTAL INTEGER)");
```

4.2.1.3 Introducción de datos en una tabla

Al igual que hicimos anteriormente vamos a introducir datos en una tabla fila a fila. Los valores insertados en las columnas se listarán en el mismo orden en que se declararon las columnas cuando se creó la tabla.

Vamos a insertar una fila de datos con *Colombian* en la columna *COF_NAME*, 101 en *SUP_ID*, 7.99 en *PRICE*, 0 en *SALES*, y 0 en *TOTAL*:

```
Statement stmt = con.createStatement();
stmt.executeUpdate(
    "INSERT INTO COFFEES " +
    "VALUES ('Colombian', 101, 7.99, 0, 0)");
```

4.2.1.4 Obtención de datos de una tabla

Para obtener los datos almacenados en una tabla, ya sea la tabla entera o ciertas filas o columnas en que se cumpla cierta condición, debemos realizar una sentencia *SELECT* de SQL, la cual se hará con el método *executeQuery* de la misma forma que en los ejemplos anteriores.

JDBC devuelve los resultados de esta consulta en un objeto *ResultSet* el cual contendrá todas las filas y columnas resultado de la operación. Para acceder a cada uno de los campos iremos fila a fila y recuperaremos los valores de acuerdo con sus tipos. El método *next* servirá para pasar de una fila a otra comenzando inicialmente justo encima de la primera fila, por lo que habrá que llamar a este método para movernos a la primera fila y convertirla en la fila actual.

Una vez posicionados en la fila debemos extraer los valores de todas las columnas de dicha fila, para ello usaremos el método *get…* del tipo apropiado que más se ajuste al campo a extraer. Por ejemplo, si queremos extraer el campo *COF_NAME* de la tabla *COFFEES*, que almacena un valor del tipo *VARCHAR* el método que hay que usar es *getString*. Para ver esto tenemos el siguiente ejemplo:

```
String query = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
        String s = rs.getString("COF_NAME");
        Float n = rs.getFloat("PRICE");
        System.out.println(s + "   " + n);
}

La salida será:

Colombian   7.99
French_Roast   8.99
Espresso   9.99
Colombian_Decaf   8.99
French_Roast_Decaf   9.99
```

Existe toda una familia de métodos *get…* para cada uno de los tipos de datos SQL que permiten hacer el paso a tipos Java. Esto son: *getByte, getShort, getInt, getLong, getFloat, getDouble, getBigDecimal, getBoolean, getString, getBytes, getDate, getTime, getTimestamp, getAsciiStream, getUnicodeStream, getBinaryStream,* y en general *getObject.*

4.2.1.5 Borrado en una tabla

En el caso de que necesitemos borrar una tabla o una fracción de ésta habría que usar la instrucción *DELETE* de SQL con o sin sentencia *WHERE*, según sea el caso. Así un ejemplo sería:

```
Statement stmt = con.createStatement();
stmt.executeUpdate(
    "DELETE SALES FROM COFFEES " +
    "WHERE  PRICE < 9.00");
```

En el ejemplo anterior se han eliminado de la tabla *COFFEES* todos los valores de la columna *SALES* en aquellas filas en las que *PRICE* es menor de 9 euros.

4.2.1.6 Actualización de tablas

Si por ejemplo queremos realizar una actualización de la columna *SALES* de la tabla *COFFEES* introduciendo el número de euros vendidos de cada tipo de café, haríamos la sentencia SQL necesaria para tal efecto y la ejecutaríamos mediante el método *executeUpdate:*

```
Statement stmt = con.createStatement();

String updateString = "UPDATE COFFEES " +
```

```
    "SET SALES = 75 " +
    "WHERE COF_NAME LIKE 'Colombian'";
stmt.executeUpdate(updateString);
```

4.2.1.7 Ordenación de registros en una tabla

Como se puede observar en los ejemplos anteriores las filas regresadas son mostradas sin ningún orden en particular. Para ordenar los resultados tenemos que usar una cláusula *ORDER BY* de SQL. Así, si queremos ordenar algunos datos según el precio haríamos:

```
String query = " SELECT COF_NAME, PRICE FROM COFFEES" + "ORDER BY PRICE;";
ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
        String s = rs.getString("COF_NAME");
        Float n = rs.getFloat("PRICE");
        System.out.println(s + "   " + n);
}

La salida será:

Colombian   7.99
Colombian_Decaf   8.99
French_Roast   8.99
Espresso   9.99
French_Roast_Decaf   9.99
```

En las columnas de tipo carácter, el ordenamiento es ejecutado normalmente de sin hacer diferencia entre mayúsculas y minúsculas. Sin embargo, si se usa el operador *BINARY* se puede forzar un ordenamiento distinguiendo mayúsculas de minúsculas.

Para especificar el orden inverso debemos agregar la palabra clave *DESC* al nombre de la columna que estamos usando en el ordenamiento.

4.2.1.8 Unión de dos tablas

Algunas veces necesitamos utilizar una o más tablas para obtener los datos que queremos. Por ejemplo, supongamos que se quiere una lista de los cafés que se le compran a la empresa Acme, Inc; para ello necesitamos información de las tablas *COFFEES* y *SUPPLIERS*. En este caso es necesario realizar un *join* o unión sobre dos tablas.

Por ejemplo, supongamos que nos piden por un lado los nombres de los suministradores que están en la tabla *SUPPLIERS*, y los nombres de los cafés de la tabla *COFFEES*. Como ambas tablas están relacionadas por medio de la columna *SUP_ID* podemos utilizarla para realizar la unión.

Para ver esto planteamos el código que habría que ejecutar:

```
String query = "SELECT COFFEES.COF_NAME " + "FROM COFFEES, SUPPLIERS " +
"WHERE SUPPLIERS.SUP_NAME LIKE 'Acme, Inc.'" + "and SUPPLIERS.SUP_ID =
COFFEES.SUP_ID";
```

```
ResultSet rs = stmt.executeQuery(query);
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
        String coffeeName = getString("COF_NAME");
        System.out.println("     " + coffeeName);
}
Esto producirá la siguiente salida:

     Colombian
     Colombian_Decaf
```