BASE DE DATOS

El lenguaje SQL

Qué es SQL

El Lenguaje de Consulta Estructurado (SQL=Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación.

Es el lenguaje utilizado universalmente para interactuar con base de datos, permitiendo realizar consultas, inserciones, actualizaciones y eliminaciones de datos, como así también de base de datos.

Dónde se utiliza

En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los Administradores de Base de Datos comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio. El soporte estándar se denomina ANSI SQL.

Entre los sistemas de gestión de base de datos con soporte SQL más utilizados se encuentran los siguientes:

- DB2
- Oracle
- SQL Server
- MySQL
- PostgreSQL
- Informix

MySQL como Data Base Management System

Introduccion

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Inicialmente, carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad y velocidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre.

Características

Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el

mayor número de operaciones disponibles.

- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

La licencia GNU GPL de MySQL obliga a distribuir cualquier producto derivado (aplicación) bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero no desea distribuirlo bajo licencia GNU GPL, puede adquirir la licencia comercial de MySQL que le permite hacer justamente eso.

DML significa Data Manipulation Language o Lenguaje de Manipulación de Datos, y corresponde las sentencias del SQL que se utilizan para manejar los datos de la base de datos (select, insert, update, delete, etc).
Involucra los comandos necesarios para hacer consultas, inserciones, modificaciones y eliminaciones. Dichos comandos están presentados de la siguiente manera:

- SELECT – obtiene información de una base de datos
- INSERT INTO – inserta información en una base de datos
- UPDATE - actualiza información de una base de datos
- DELETE – elimina información de una base de datos

El comando SELECT

El comando SELECT se utiliza para seleccionar información de una tabla. Para seleccionar todas la columnas se utiliza el * (asterisco), y la cláusula WHERE se utiliza para establecer un criterio de búsqueda.
Para traer todos los datos de una tabla, se realiza la siguiente consulta:

```
SELECT * FROM tabla_nombre
```

Si es necesario traer únicamente los datos de una columna, se realiza de la siguiente manera:

```
SELECT nombre_columna(s) FROM tabla_nombre
```

Para traer los registros segun una condicion, se realiza de la siguiente manera:

```
SELECT nombre_columna(s) FROM tabla_nombre WHERE campo1 = valor1
```

El comando INSERT

El comando INSERT se utiliza para insertar datos en una tabla. La inserción se realiza de la siguiente manera:

[code]
INSERT INTO "nombre_tabla" ("columna1", "columna2", ...)
VALUES ("valor1", "valor2", ...)
[/code]

Se pueden agregar datos en grupo o especificando la columna donde es necesario ingresar la

información. Para agregar información a todos los campos, el uso es el siguiente:

[code]
INSERT INTO nombre_tabla
VALUES (valor1, valor2,....)
[/code]

Para agregar valores en cada columna de manera particular, se realiza de la siguiente forma:

[code]
INSERT INTO nombre_tabla(columna1, columna2, ...)
VALUES (valor1, valor2,....)
[/code]

El comando UPDATE

El comando UPDATE se utiliza para actualizar registros, y su forma de utilización es la siguiente:

[code]
UPDATE nombre_tabla
SET nombre_columna1 = nuevo_valor, nombre_columna2 = otro_valor WHERE nombre_columna = algun_valor
[/code]

El comando DELETE

El comando DELETE se utiliza para eliminar registros, su forma de utilización es la siguiente:

[code]
DELETE FROM nombre_tabla WHERE nombre_columna = algun_valor
[/code]

JDBC: CONEXIÓN CON BASE DE DATOS

Introduccion

Qué es JDBC

JDBC es el acrónimo de Java Database Connectivity, 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 lenguaje SQL del modelo de base de datos que se utilice.

Esta conformada por diversas clases e interfaces ubicadas en el paquete ***java.sql***.

La necesidad de una librería

Al trabajar con JDBC resulta necesario agregar un jar al proyecto que contiene las clases necesarias que se utilizan para “dialogar” con un DBMS. Cada DBMS tiene su propio archivo jar. Estos archivos se pueden obtener de:

http://developers.sun.com/product/jdbc/drivers

Tambien es posible conseguir estos archivos jar en la pagina web correspondiente a cada DBMS, por ejemplo, en caso de usar un DMBS Oracle es posible buscarlo en la pagina de Oracle: www.oracle.com

Conexión con la base de datos

La interfaz Connection

Para poder trabajar con una base de datos, el punto de partida siempre es conseguir una conexión, es decir un objeto del tipo Connection (este objeto pertenece a una clase que implementa la interfaz Connection).

Construcción de un Administrador de Conexiones

Para poder obtener una conexión, una forma simple y cómoda de trabajar es armar una clase llamada, por ejemplo, ***AdministradorDeConexiones***, que contenga dentro de un método ***obtenerConexion()*** el código necesaria para obtenerla. A continuación se presenta un ejemplo de la clase con su correspondiente método:

[code]
```
package ar.com.educacionit.database;
import java.sql.Connection;
import java.sql.DriverManager;
public abstract class AdministradorDeConexiones {
public static Connection obtenerConexion() throws Exception
{

    // Establece el nombre del driver a utilizar
    String dbDriver = "com.mysql.jdbc.Driver";

        // Establece la conexion a utilizar
    String dbConnString = "jdbc:mysql://localhost/baseDatos";

        // Establece el usuario de la base de datos
    String dbUser = "root";

    // Establece la contraseña de la base de datos
    String dbPassword = "";

    // Establece el driver de conexion
    Class.forName(dbDriver).newInstance();

    // Retorna la conexion
    return DriverManager.getConnection(
                                            dbConnString, dbUser, dbPassword);
        }

}
```
[/code]

Cómo consultar datos

El metodo createStatement()

El método ***createStatement()*** se utiliza para crear un objeto que modela a una sentencia SQL. Es un objeto del tipo de una clase que implementa la interfaz Statement, y provee la infraestructura para ejecutar sentencias SQL sobre una conexión con una base de datos.

La forma de construir un objeto de este tipo es:

```
Statement stmtConsulta = laConexion.createStatement();
```

El metodo executeQuery()

El método ***executeQuery()*** se utiliza para ejecutar una sentencia SQL y obtener el resultado correspondiente dentro de un objeto del tipo ResultSet. Este objeto representa un conjunto de resultados que se obtienen como consecuencia de ejecutar la sentencia SQL del tipo SELECT a traves de la conexión.

La forma de generar un objeto de este tipo es:

ResultSet rs = stmtConsulta.executeQuery(laConsulta);

Como realizar una consulta

[code]
// Define la conexion
Connection laConexion = AdministradorDeConexiones.getConnection();

// Arma la consulta y la ejecuta
String laConsulta = "SELECT * FROM alumnos";
Statement stmtConsulta = laConexion.createStatement();
ResultSet rs = stmtConsulta.executeQuery(laConsulta);

// Muestra los datos
while(rs.next()){
    System.out.println("ID: " + rs.getInt("alu_id") + " -- " +
                        "Nombre: " + rs.getString("alu_nombre") + " -- " + “Apellido: " +
                        rs.getString("alu_apellido"));
}

// Cierra el Statement y la Connection
stmtConsulta.close();
laConexion.close();
[/code]

Cómo insertar datos

El método createStatement()

El método createStatement() es el mismo presentado en la sección Consulta.

El método execute()

El método ***execute()*** se utiliza para ejecutar sentencias SQL del tipo INSERT, UPDATE o DELETE, y a diferencia del método ***executeQuery()*** no retorna un conjunto de resultados.

La forma de utilizar el método ***execute()*** es:

[code]
String laInsercion =
    "INSERT INTO alumnos (alu_id, alu_nombre, alu_apellido) VALUES
    (101, 'Manuel', 'Santos')";

stmtInsercion.execute(laInsercion);
[/code]

Cómo realizar una inserción

[code]
// Define la conexion
Connection laConexion = AdministradorDeConexiones.getConnection();

```
// Arma la sentencia de inserción y la ejecuta
String laInsercion = "INSERT INTO alumnos (alu_id, alu_nombre,
                                           alu_apellido) VALUES (101, 'Manuel', 'Santos')";
Statement stmtInsercion = laConexion.createStatement();
stmtInsercion.execute(laInsercion);

// Cierra el Statement y la Connection
stmtInsercion.close();
laConexion.close();

// Informa que la inserción ha sido realizada con éxito
System.out.println("La inserción ha sido realizada con éxito...");
[/code]
```

Cómo actualizar datos

El método createStatement()

El método ***createStatement()*** es el mismo presentado en la sección Consulta.

El método execute()

El método ***execute()*** es el mismo presentado en la sección Inserción.

Cómo realizar una actualización

```
[code]
// Define la conexion
Connection laConexion = AdministradorDeConexiones.getConnection();

// Arma la sentencia de actualización y la ejecuta
String laActualizacion = "UPDATE alumnos SET alu_apellido =
                                                    'Trobbiani' WHERE alu_id = 101";
Statement stmtActualizacion = laConexion.createStatement();
stmtActualizacion.execute(laActualizacion);

// Cierra el Statement y la Connection
stmtActualizacion.close();
laConexion.close();

// Informa que la actualización ha sido realizada con éxito
System.out.println("La actualización ha sido realizada con
                                                éxito...");
                                                [/code]
```

Cómo eliminar datos

El método createStatement()

El método ***createStatement()*** es el mismo presentado en la sección Consulta.

El método execute()

El método ***execute()*** es el mismo presentado en la sección Inserción.

Cómo realizar una eliminación

```
[code]
// Define la conexion
Connection laConexion = AdministradorDeConexiones.getConnection();
```

```
// Arma la sentencia de eliminación y la ejecuta
String laEliminacion = "DELETE FROM alumnos WHERE alu_id = 101";
Statement stmtEliminacion = laConexion.createStatement();
stmtEliminacion.execute(laEliminacion);

// Cierra el Statement y la Connection
stmtEliminacion.close();
laConexion.close();

// Informa que la eliminación ha sido realizada con éxito
System.out.println("La eliminación ha sido realizada con
éxito...");
[/code]
```