Instituto Politcnico Nacional

Escuela Superior de Cómputo

Web App Development.

Tarea 1 : Acceso a Datos con JDBC

Profesor: M. en C. José Asunción Enríquez Zárate

Alumno: Mauro Sampayo Hernández

mauro_luigi@hotmail.com

3CM18

6 de septiembre de 2021

Índice

1. Introducción

En la actualidad, es casi indispensable realizar la clasificación de datos de un sistema por medio de una base de datos. Sin embargo, a partir de esto surge la necesidad de llevar a cabo su correcta administración al momento de recuperar o modificar los datos que se encuentren contenidos dentro de dicha base de datos, especialmente si se trata de una que se muy grande y compleja.

Para satisfacer esta necesidad, hoy en día existen múltiples sistemas que se encargan de facilitar la administración de los datos de las bases de datos, estos sistemas se conocen como Sistemas de Gestión de bases de datos (BDMS por sus siglas en inglés), los cuales se encarga de proveernos de mecanismos con los que se puedan llevar a cabo dichas tareas de manera sencilla.

Existe una gran cantidad de BDMS, sin embargo, para este reporte se hará énfasis en las herramientas que provee SQL para la administración de bases de datos y la API JDBMS que brinda la posibilidad de realizar conexiones a bases de datos desde una aplicación de Java, y a partir de esta poder realizar modificaciones en las mismas.

2. Conceptos

A continuación se enlista una serie de conceptos, que son necesarios para poder entender más a detalle la funcionalidad de una Base de Datos y los Sistemas de Gestión que las administran

2.1. SQL

SQL es un lenguaje estándar internacional que hace uso de bases de datos relacionales para realizar queries (aquellos que realizan la solicitud de información que satisfaga ciertos criterios) y la manipulación de datos.

2.2. Base de Datos Relacional

Una base de datos relacional es una representación lógica de datos que permiten acceso a los datos sin considerar su estructura física.

2.3. Sistemas de Gestión de Bases de Datos (BDMS)

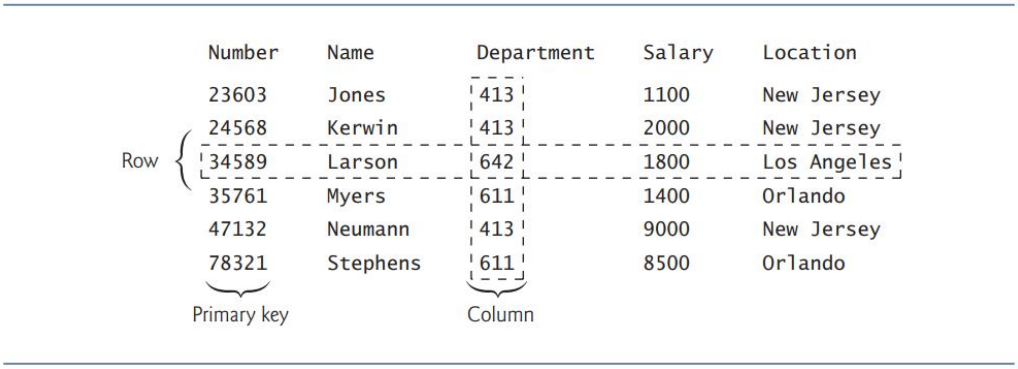
Un Sistema de Gestión de bases de datos (BDMS por sus siglas en inglés) es un sistema que provee mecanismos para el almacenamiento, organización, recuperación y modificación de datos por distintos usuarios sin afectar la representación interna de los datos.

2.4. Statement(sentencia)

Es una herramienta que sirve para procesar una sentencia SQL estática y obtener los resultados producidos por ella.

3. Desarrollo

Una base de datos relacional es una representación lógica de datos que permiten acceso a los datos sin consideración de su estructura física, y almacena dichos datos en tablas. En la figura 3 se muestra un ejemplo de base de datos relacional; en este caso el nombre de la tabla es **Employee**, y su propósito es el de almacenar los atributos de los empleados.



Las tablas se componen de filas y columnas en las cuales se almacenan los valores de la tabla. En el caso de la tabla de la Figura 1 se tienen seis filas. La columna **Number** d cada fila es la clave primaria o primary key, que es aquella columna que contiene un único valor por cada fila, y el cuál no puede ser duplicado, asegurando así que cada fila puede ser identificada por su clave primaria. Algunos ejemplos de llaves primaria son los números de seguridad social, los IDs de empleado, etc. Las filas de una tabla pueden ser ordenadas en un orden en especifico o no tener ningún orden en particular

3.1. Manipulación de datos con SQL

Uno de los Sistema de Gestión de bases de datos más utilizados en la actualidad es SQL, el cual nos permite realizar la consulta y modificación de datos de tablas en una Base de Datos por medio de cláusulas. A continuación, se describen las cláusulas más importantes para la consulta y modificación de datos de una base de datos en SQL.

3.1.1. Clausula SELECTFROM

Realiza la recuperación de datos de una o más tablas, seleccionando aquellas columnas de las tablas de una base de datos que sean requeridas. La forma básica de esta consulta es:

```
1        SELECT * FROM nombreTabla
2
```

Donde el asterisco (*) indica que todas las columnas de la tabla *nombreTabla* deben ser recuperadas. Para realizar la recuperación de solo ciertas columnas de la tabla, se debe remplazar el asterisco (*) con la lista de columnas que se deseen consultar separadas por una coma (,).

```
1        SELECT nombreCulumna1, nombreColumna2,     FROM nombreTabla.
2
```

3.1.2. Clausula WHERE

Indica el criterio de selección que determinará que filas de una tabla serán recuperadas, eliminadas o actualizadas durante una consulta. La forma básica de esta consulta es:

```
1        SELECT nombreCulumna1, nombreColumna2,
2        FROM nombreTabla.
3        WHERE criterioSeleccion.
4
```

La cláusula WHERE puede hacer uso de los operadores $<$, $>$, $\leq, \geq, <>$ $yLIKE$.

El operador *LIKE* es usado para realizar la búsqueda de cadenas de texto que coincidan o cumplan con un determinado patrón. El operador *LIKE* se auxilia de los caracteres de porcentaje (%) y guión bajo (_).

El caracter de porcentaje (%) realizara la búsqueda de cadenas de texto que tengan cero o más caracteres desde la posición del caracter de porcentaje (%), mientras que el carácter guion bajo (_) indica que puede haber cualquier caracter en dicha posición. Por ejemplo:

```
1        SELECT ColumnaA, ColumnaB,
2        FROM Tabla.
3        WHERE ColumnaA LIKE     _o %z
4
```

En este ejemplo a consulta seleccionará únicamente las filas que cumplan con la condición, de que en la Columna B haya una cadena de texto que cumpla con el siguiente patrón: Que inicie con cualquier carácter que sea seguido de una ‘o, posteriormente haya de o a más caracteres que finalicen con ‘z. Cabe destacar que los patrones de búsqueda siempre deben estar escritos entre comillas simples (‘).

3.1.3. Clausula ORDER BY

Indica los criterios a seguir para realizar el ordenamiento de filas de una tabla. Las filas pueden ser ordenadas de manera ascendente o descendente. . La forma básica de esta consulta es:

```
1        SELECT ColumnaA, ColumnaB,      FROM nombreTabla.   ORBER BY ColumnaA ASC
2        SELECT ColumnaA, ColumnaB,      FROM nombreTabla.   ORBER BY ColumnaA DESC
3
```

Donde ASC especifica un ordenamiento de forma ascendente (del menor al mayor valor), y DESC ordenamiento de forma descendente (del mayor al menor valor). En caso de no especificar el tipo de ordenamiento en la cláusula ORDER BY, este se realizará de forma ascendente por defecto.

La cláusula ORDER BY puede ser usada junto a la cláusula WHERE y puede realizar el ordenamiento usando de referencia más de una columna, sin necesidad de que estas deban tener el mismo tipo de ordenamiento.

3.1.4. Clausula INNER JOIN

Realiza la unión de filas de múltiples tablas que conforman una base de datos en una sola. La forma básica de esta consulta es:

```
1        SELECT columna1, columna2,
2        FROM tabla1.
3        INNER JOIN tabla2
4            ON tabla1.columna = table 2.columa
5
```

La cláusula ON especifica las columnas de cada tabla, que tendrán que ser comparadas para determinar que filas serán unidas. Cabe destacar que la utilización del punto (.) en las columnas de la tabla, se realiza para indicar a que tabla pertenece cada columna dado el caso de que haya dos columnas con el mismo nombre en dos tablas distintas.

3.1.5. Clausula INSERT

Realiza la inserción de filas en una tabla especifica. La forma básica de esta consulta es:

```
1        INSERT INTO nombreTabla (columna1, clumna2,    ,columnaN)
2        VALUES (valor1, valor2,     , valorN)
3
```

Donde *nombreTabla* es la tabla donde se insertará la fila y que estará seguida por una lista de elementos separada por comas y entre parntesis, especificando las columnas donde serán insertados los datos. Posteriormente se coloca la clausula VALUES seguida por una lista de elementos separada por comas y entre parntesis, con los valores a insertar en cada una de las columnas especificadas anteriormente, los cuales deben coincidir con el tipo de dato especificado para cada columna.

3.1.6. Clausula UPDATE

Realiza la actualización y modificación de filas en una tabla especifica. La forma básica de esta consulta es:

```
1        UPDATE nombreTabla
2            SET columna1 = valor1, columna2 = valor2,     , columnaN = valorN
3            WHERE    criterioSeleccion
4
```

Donde *nombreTabla* es la tabla por actualizar, seguido de la cláusula SET y una lista separada por comas, con el formato *nombreColumna = valor*. La cláusula WHERE es opcional y provee de un criterio de selección que ayuda a determinar que filas serán actualizadas.

3.1.7. Clausula DELETE

Realiza la eliminación de filas en una tabla especifica. La forma básica de esta consulta es:

```
1        DELETE FROM nombreTabla  WHERE criterioSeleccion
2
```

Donde *nombreTabla* es la tabla en donde se realizará la eliminación. La cláusula WHERE es opcional y provee de un criterio de selección que ayude a determinar que filas serán eliminadas.

3.2. Manipulación de datos con la API JDBC

La API JDBC es una API de Java muy útil, que permite la conexión y el acceso a una base de datos desde una aplicación Java, para poder manipular y consultar la información contenida en estas desde la misma aplicación.

3.2.1. Conexión a una base de datos por medio de la API JDBC

A continuación, se explican los pasos generales para realizar la conexión a una base de datos desde una aplicación Java por medio de la API JDBC.

1. Se importan las interfaces JDBC y las clases del paquete java.sql, las cuáles contienen todas las clases y mtodos necesarios para realizar la conexión a la base de datos desde la aplicación Java, y la manipulación de datos de esta.

2. Se declara una cadena de texto que contiene el URL de la base de datos, la cual servirá como identificación de la base de datos a la cual se realizará la conexión. La URL debe tener el siguiente formato:

   protocoloComunicacion: subprotocoloComunicacion l://localizacionBD/nombreBD

3. En el mtodo main se crea un objeto de tipo *Connection*, el cuál administrará la conexión ente el programa en Java y la base de datos.

4. Se inicializa la conexión llamando al mtodo *getConnection* de la clase *DriveManager* , que intentará realizar la conexión con la base de datos especificada en la URL. Este mtodo recibe 3 atributos tipo string; en el primer atributo se debe ingresar la URL de la base de datos a acceder, en el segundo el usuario y en el tercero la contraseña del usuario.

De esta manera nuestra aplicación de Java se conectará exitosamente a la base de datos.

3.2.2. Manipulación y consulta de datos por medio de la API JDBC

A continuación, se explican los pasos generales para realizar la manipulación de datos utilizando cláusulas de SQL, desde una aplicación Java por medio de la API JDBC.

1. Se invoca el mtodo *createStatement* de la clase *Connection* para la creación de un statement de SQL, el resultado de dicho mtodo se guardará en un objeto tipo *Statement*.

2. Se invoca el mtodo *executeQuery* de la clase *Statement*, que recibirá un atributo de tipo string, donde se ingresará el statement de SQL, que contendrá la instrucción de consulta o modificación de datos que se desee realizar. Est mtodo se encargará de realizar la acción correspondiente a dicho statement y guardará la información resultante en un objeto de tipo ResultSet.

3. Se obtiene la metadata de los resultados del statement, almacenados previamente en el objeto de tipo *ResultSet*, por medio del mtodo *getColumnCount* de la clase *ResultSertMetaData*.

4. Si se desea imprimir los datos de la base de datos para poder visualizar la consulta o modificación realizados sobre esta, se puede hacer lo siguiente:

   a) Por medio del mtodo *getColumnName* de la clase *ResultSertMetaData* y con ayuda de un ciclo for se pueden imprimir los nombres de todas las columnas que conforman la tabla, utilizando la metadata de los resultados de la consulta realizada previamente.

   b) Por medio del mtodo *getObject* de la clase *ResultSet* y con ayuda de un ciclo for se pueden imprimir la información que contienen las filas de la columna de la tabla.

5. Finalmente, y una vez finalizada la manipulación y consulta de datos, dentro de un bloque finally se realiza el cierre de los objetos de tipo *ResultSet*, *Statement* y *Connection*.

3.3. PreparedStatements

Un PreparedStatemente es una herramienta que permite la creación de statements SQL que permiten la ejecución de del mismo query repetidamente con diferentes valores en sus parámetros. A diferencia de los Statements normales, estos están parametrizados de tal forma que resultan tener mayor eficiencia que los Statements.

A continuación, se presenta un ejemplo de un PreparedStatement implementado en Java por medio de la clase *PreparedStatements*:

```
PreparedStatement authorBooks = connection.prepareStatement(
   "SELECT LastName, FirstName, Title " +
   "FROM Authors INNER JOIN AuthorISBN " +
      "ON Authors.AuthorID=AuthorISBN.AuthorID " +
   "INNER JOIN Titles " +
      "ON AuthorISBN.ISBN=Titles.ISBN " +
   "WHERE LastName = ? AND FirstName = ?" );
```

Figura 1: Ejemplo de un *PreparedStatement*

Los dos signos de interrogación (?) representan placeholders, en donde se podrán colocar valores que pasarán a ser parte del query se realice en la base de datos. Para poder colocar dichos valores se tiene que recurrir al mtodo *setString* de la clase *PreparedStatements* como se muestra a continuación:

```
authorBooks.setString( 1, "Deitel" );
authorBooks.setString( 2, "Paul" );
```

Figura 2: Ejemplo mtodo *setString* de la clase *PreparedStatement*

El mtodo *setString* como se puede observar recibe dos argumentos. El primero representa el número de parámetro en donde se colocará el valor en cuestión (la numeración de los parámetros inicia siempre en 1, empezando desde el primer signo de interrogación (?) que haya sido colocado); y el segundo representa el valor que se colocará en el parámetro.

4. Resultados

Tras finalizar la revisión del documento, se logró el entendimiento acerca de que es un Sistema de Gestión de Bases de Datos, la identificación de las instrucciones para manipular datos en bases de datos por medio de SQL y sus respectivas funciones y el funcionamiento de la API JDBC.

De igual manera se logró el entendimiento de todos los pasos a seguir para realizar una aplicación en Java que pueda acceder a Bases de Datos por medio de JDBC.

5. Conclusión

El uso de Sistemas de Gestión para la administración, ordenamiento y modificación de datos de las tablas de una base de datos es de vital importancia para el correcto funcionamiento de un sistema, pues estas son necesarias para llevar a cabo la manipulación de los datos de manera eficaz, segura y sencilla.

De igual forma las herramientas que nos provee JDBC y SQL resultan muy útiles al momento de llevar a cabo el proceso anteriormente mencionado, en especial al momento de desarrollar aplicaciones en Java que se conecten con una base de datos para manipularla.

Mauro Sampayo Hernández

6. Referencias Bibliográficas

Referencias

[1] *Chapter 28. Accesing Databases with JDBC*