Sistemas de Información

Práctica 4 Curso 2016/17 3º curso

JDBC

Introducción

Hasta ahora hemos utilizado SQL desde un intérprete interactivo que nos permite introducir los diferentes comandos uno a uno. También hemos aprendido a utilizar procedimientos almacenados para agrupar sentencias y realizar pequeños programas SQL.

Sin embargo, en muchas ocasiones querremos que nuestras bases de datos actúen como fuentes de información para programas más complejos. Este tipo de programas no se pueden desarrollar en SQL dadas sus limitaciones como lenguaje de propósito general (imaginemos un programa con una interfaz gráfica de usuario compleja, o un servidor de aplicaciones a través de Web)

Para el desarrollo de este tipo de aplicaciones complejas necesitamos utilizar las posibilidades que nos proporcionan los modernos lenguajes de programación de propósito general, junto con sus amplias colecciones de librerías (por ejemplo Java, C, PHP, Perl, …)

En esta práctica nos familiarizaremos con las herramientas que nos permiten utilizar SQL para acceder a nuestras bases de datos desde programas escritos en otro lenguaje de programación (que llamaremos lenguaje "anfitrión")

En concreto, vamos a utilizar Java como lenguaje anfitrión, y el elemento principal para poder utilizar SQL desde Java es el driver JDBC (*Java DataBase Conectivity*).

JDBC es un API Java que independiza los procedimientos de acceso a bases de datos del sistema operativo y del SGBD concreto que estemos utilizando. Gracias a esto, los programas desarrollados dispondrán de una portabilidad elevada entre entornos diferentes.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Aprender a acceder a bases de datos desde programas escritos en otros lenguajes.
- Familiarizarse con el concepto de acceso secuencial a las tuplas de una relación mediante cursores.
- Comprender el intercambio de parámetros entre procedimientos almacenados SQL y Java.

Entorno

Para realizar los ejercicios de práctica es necesario:

1. Tener instalado el entorno como se configuró para prácticas anteriores.
2. Tener instalado el entorno Eclipse para desarrollo en Java.
3. Disponer de driver jdbc para MySQL (disponible en http://dev.mysql.com/downloads/connector/j/)

(En OpenSuse también se puede instalar el driver con: "`sudo zypper install mysql-connector-java`")

Modificando bases de datos vía JDBC

Establecimiento de conexiones

Los accesos a bases de datos desde un lenguaje anfitrión se realizan a través de conexiones. Por tanto, nuestro primer objetivo debe ser establecer una conexión y gestionar adecuadamente los posibles problemas que se puedan producir, y que en Java se manifestarán mediante excepciones.

Las conexiones (*Connection*) se construyen sobre un gestor (*DriverManager*) que se encarga de gestionar las comunicaciones con una determinada fuente de datos. Por tanto, para realizar este ejercicio deberías consultar la documentación (javadoc) de estas dos clases.

Ejercicio 1:

Crea un proyecto Java en Eclipse y añádele la librería que contiene el driver JDBC.

Escribe un programa Java que sea capaz de establecer una conexión a una base de datos, cuyo nombre tomará como parámetro por línea de comandos, y de liberar dicha conexión. El programa gestionará adecuadamente las excepciones que se puedan producir.

Prueba el programa con una base de datos existente y con una que no exista. Prueba también a conectarte utilizando un usuario y/o una clave incorrectas.

Ejecutando sentencias

Una vez que tenemos establecida correctamente una conexión con una base de datos, podemos ejecutar sentencias SQL sobre ella desde nuestro programa Java, tal como hemos hecho hasta ahora en nuestro intérprete *mysql*.

Para ejecutar sentencias SQL que supongan modificaciones sobre una base de datos, bien sea del esquema, bien sea de los datos almacenados, se utiliza una instancia de la clase Java `Statement`. En concreto, debemos utilizar el método `executeUpdate()`. Podemos utilizar la misma instancia de `Statement` para ejecutar varias sentencias.

Para realizar este ejercicio deberás, por tanto, consultar la documentación de esta clase.

Ejercicio 2:

Desde *mysql*, como DBA, crea una nueva base de datos que se denomine `misPeliculas` y dale los permisos adecuados al usuario que estés utilizando habitualmente.

Ejercicio 3:

Vuelve a crear el esquema de base de datos que diseñaste en los 7 primeros ejercicios de la práctica 2 sobre esta nueva base de datos, pero en esta ocasión realizando todas las operaciones desde un programa Java en vez de desde *mysql*. Utiliza como base el programa que realizaste en la práctica 1 de la presente práctica.

El programa se debe poder ejecutar tantas veces como se estime oportuno, sin que su comportamiento dependa del estado de la base de datos en un momento determinado.

Comprueba desde *mysql* que todas las operaciones se realizaron correctamente.

Ejercicio 4:

Modifica el programa Java que has desarrollado en el ejercicio 3 para que también introduzca algunos datos en las diferentes relaciones que configuran tu esquema utilizando la sentencia `INSERT`.

Ejercicio 5:

Modifica el programa Java que has desarrollado en el ejercicio 4 para que, además, permita introducir interactivamente nuevas tuplas a almacenar en las relaciones.

Consultando bases de datos vía JDBC

Consultas mediante cursores

A la hora de obtener datos almacenados en la base de datos nos encontramos con el problema de no saber, en tiempo de diseño, cuántas tuplas va a tener el resultado de una consulta, ya que dicho número varía con el tiempo.

La solución que se adopta para acceder a todas las tuplas de una relación consiste en realizar un acceso tupla a tupla utilizando un bucle y mediante un elemento que denominamos cursor, que ya introdujimos en la práctica anterior, y que nos da acceso a la tupla actual.

Para realizar una consulta que devuelva varias tuplas utilizaremos, por tanto, los siguientes elementos:

- Una instancia de la clase `Statement`, que representará la consulta.
- Una instancia de la clase `ResultSet`, que representará el cursor.

Consulta la documentación de ambas clases.

Los métodos de más interés en este momento para nosotros de estas dos clases son:

Clase `Statement`:

- `executeQuery()`: que ejecutará la consulta y nos devolverá un cursor (instancia de ResultSet) al resultado de la consulta.

Clase `ResultSet`:

- `next()`: que nos permitirá comprobar si quedan más tuplas por acceder y avanzar a la siguiente.
- `getXXX()`: que son un conjunto de métodos(`getInt()`, `getFloat()`, ...) que nos permiten obtener los valores almacenados para los diferentes atributos de la relación que estamos explorando para la tupla a la que está apuntando el cursor.

Ejercicio 6:

Crea un programa Java que implemente alguna de las consultas que diseñaste en el ejercicio 8 de la práctica 2.

Modificación de datos mediante cursores

Las modificaciones de datos almacenados en determinadas tuplas de la base de datos también es una operación que puede afectar a varias tuplas y, por tanto, también se realiza mediante cursores.

Por defecto, cuando creamos un nuevo cursor en JDBC, éste es de sólo lectura, aunque podemos modificar su comportamiento para que permita actualizar datos.

Ejercicio 7:

Crea un programa Java que capitalice (ponga en mayúscula la primera letra de cada palabra) los títulos de todas las películas de tu base de datos.

Uso de transacciones

En la práctica anterior aprendimos a utilizar transacciones para asegurar la integridad de un conjunto de sentencias SQL.

En JDBC las transacciones se controlan de manera similar: Cuando creamos una conexión, ésta, por defecto, toma cada sentencia como una transacción. Para gestionar nosotros las transacciones debemos desactivar ese comportamiento por defecto (*auto-commit*) y cerrar explícitamente las transacciones. La clase `Connection`, que gestiona nuestra sesión con la base de datos nos proporciona métodos para realizar estas operaciones.

Ejercicio 8:

Repite el ejercicio 9 de la práctica 3, pero esta vez desde un programa Java. Dicho ejercicio decía:

Deshabilita la gestión automática de transacciones, comienza manualmente una nueva transacción, realiza un par de operaciones que supongan una modificación de los datos almacenados en tu base de datos y, antes de concluir la transacción, simula un problema en tu SGBD[1].

Una vez recuperado el SGBD observa qué ha pasado con los cambios que habías realizado.

Repite el experimento, pero simulando el problema después de confirmar la transacción.

[1] Para simular un problema, puedes apagar la máquina en la que estés ejecutando el SGBD. Si estás trabajando con tu máquina, de forma alternativa puedes matar el proceso gestor como superusuario.

Procedimientos almacenados

Desde JDBC también podemos ejecutar procedimientos almacenados de nuestra base de datos.

Para que esto sea posible, debemos disponer de las herramientas necesarias para poder pasar los parámetros de entrada desde Java al procedimiento almacenado y para extraer los resultados hacia variables Java.

Para realizar estos intercambios de información se utiliza la clase `callableStatement`. Consulta la documentación de esta clase para comprender su uso.

Ejercicio 9:

Crea un programa Java que llame a alguno de los procedimientos almacenados que creaste en la práctica 3.

Prueba, al menos, algún procedimiento almacenado con parámetros de entrada y alguno con parámetros de salida

Ejercicio final

Ejercicio 10:

Utilizando todo el trabajo realizado en esta práctica, crea un programa Java que permita gestionar tu base de datos de películas: introducir nuevas películas, realizar alguna consulta, borrar datos, …

Resumen

Los principales resultados esperados de esta práctica son:

- Ser capaces de acceder a bases de datos desde programas escritos en Java.
- Aprender cómo las diferentes funcionalidades proporcionadas por SQL se integran en un lenguaje anfitrión.
- Intercambiar parámetros de procedimientos almacenados entre variables Java y variables SQL.

Como trabajo adicional del alumno, se proponen las siguientes líneas:

- Trabaja en completar tu programa Java para gestionar la base de datos de películas, de tal manera que soporte aquellas operaciones que consideres habituales.
- Analiza la conveniencia de realizar determinados conjuntos de operaciones como procedimientos almacenados SQL o bien como métodos Java.
- Estudia la utilidad de la clase `PreparedStatement`.
- Prueba las diferentes opciones de los cursores que permiten realizar movimiento libre a través de las tuplas de una relación (avanzar, retroceder, saltar, …)