Síntesis conceptual

Grado: Desarrollo de Aplicaciones Multiplataforma
Asignatura: Acceso a datos
Unidad: 3. Bases de datos relacionales

Resumen

Los SGBD, sistemas gestores de bases de datos, poseen sus propios lenguajes para el almacenamiento en función de sus tipos, relacionales, XML, de objetos, etc., mientras que las aplicaciones suelen emplear un lenguaje más general como Java, por lo que para que puedan trabajar en conjunción requieren de un conector API.

En las bases de datos relacionales más extendidas, se emplea SQL, ya sea como tal o en una de sus versiones como PL/SQL. Los drivers ayudan a los conectores a traducir las particularidades de cada base de datos. Se forma una arquitectura que entrelaza API y drivers en un sistema de traducción doble. Existen múltiples arquitecturas como la original ODBC, Open DataBase Connectivity, Java emplea una variante de ODBC llamada JDBC, el cual posee drivers para elementos fuera de la base de datos, como son los ficheros CSV y XML.

Los conectores, entre otras operaciones, permiten realizar consultas en las bases de datos relacionales. Ante una consulta, responden fila a fila empleando un objeto a modo de iterador o cursor. Por otro lado, una consulta sin un conector devolverá un conjunto de filas y recordset.

Empleando JDBC podemos llevar a cabo ciertas operaciones basadas en SQL, como son SELECT, UPDATE, DELETE, INSERT o execute().

En JDBC los drivers se almacenan en ficheros jar, a los que podemos conectarnos con la clase DriverManager empleando getConnection (String URL de la conexión). La URL contiene identificadores de acceso.

La ejecución de consultas executeQuery() devolverá como respuesta ResultSet devolverá un conjunto de filas, aunque existen métodos para que la respuesta también muestre distintas columnas señalando estas por su posición o nombre.

En el caso de emplear sentence tendremos que pasar las columnas, una a una mediante next(), con ResultsSet tenemos diferentes métodos.

También nos podemos encontrar resultados scrollable de ResultSet empleando parámetros especiales con Statement o PreparedStatement

Podemos simplificar las sentencias mediante expresiones con variables, pero esto generaría sus propios problemas. Con el fin de evitar este problema podemos emplear sentencias preparadas las cuales, mediante marcadores, o placeholders, generalmente "?", realizan una precompilación usable múltiples veces con tan solo modificar los valores y volver a ejecutar. Las consultas se

realizan con PreparedStatement mediante el método getPreparedStatement de Connection. A este método añadimos el marcador "?".

En ocasiones es necesario llevar a cabo más de una operación de manera simultánea, en estos casos se emplean las transacciones. Las transacciones nos permiten establecer diversas operaciones simultaneas de manera completa, o en caso de error, eliminar cualquier modificación, de modo que se complete el conjunto completo o, en su defecto, no se realice ningún cambio a la base de datos.

La escritura en el código de los bloques no es la única forma de llevar a cabo acciones, con el fin de evitar un exceso de repetición de la escritura se implementaron las llamadas o invocaciones de procedimiento o función. Estas invocaciones nos permiten desarrollar una función tan solo con su nombre, evitándonos la tediosa tarea de escribirlo. Esto nos permite reutilizar código y por tanto ahorrar tiempo y evitar errores. También podemos crear procedimientos especiales llamados Triggers, que cuentan con la particularidad de invocarse a sí mismos cuando ocurre un evento específico a modo de respuesta.

La incorporación de invocaciones y Triggers, permite un código mucho más ligero y ordenado, evitando la repetición excesiva de un mismo código.

Podemos realizar llamadas a funciones y procedimientos.

- Procedimientos: permite que un bloque de código pueda ser llamado con una invocación.
- Funciones: las funciones tienen como final devolver un resultado concreto tras su invocación. Es imprescindible indicar el tipo de dato de la devolución.

Conceptos fundamentales

- **Acceso aleatorio**: acceso que permite leer el dato indicado
- **Acceso secuencial**: acceso que obliga a visitar todos los datos hasta el deseado.
- **Fichero de texto**: fichero que contiene solo texto.
- **Índice**: Fichero que sirve de índice a un segundo fichero para implementar búsquedas.
- **Búfer**: Pequeño espacio de memoria para datos temporales.