ACCESO-A-DATOS-TEMA-6.pdf

user_4383848

Acceso a datos

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

ACCESO A DATOS - TEMA 6

MANEJO DE CONECTORES III - SENTENCIAS

1. CREACIÓN DE BASE DE DATOS (DDL):

CREATE DATABASE: Es necesario disponer del permiso CREATE. Con "create_specification" se establecerán las distintas características de la base de datos. Almacenadas en el fichero con extensión .opt en la raíz de la base de datos. Con la cláusula "character set", se especificará el set de caracteres por defecto de dicha base de datos que se está creando.

CREATE INDEX: Normalmente, con la creación de la tabla, se agregan todos los índices,pero con este comando podemos agregarlos manualmente después de haber creado una tabla. Para el tipo de columnas numeradas (columna1, columna2, etc.), se crea un índice de columnas múltiples. Los índices se forman al unir los valores de las columnas.

ALTER DATABASE: este comando nos facilita cambiar características generales de una base de datos. Para usar este comando se necesita el permiso ALTER.

ALTER TABLE: Modificar la estructura de una tabla que previamente hemos creado. Se pueden realizar numerosas acciones como agregar columnas, borrarlas, crear índices, borrarlos, cambiar la tipología de ciertas columnas…

2. CREACIÓN Y ELIMINACION DE TABLA (DDL):

CREATE TABLE: Creación de una tabla con el nombre definido. También necesitaremos el permiso CREATE para la tabla. Existen algunas restricciones: Ocurrirá un error si la tabla que estamos creando existe previamente en la base de datos que estamos trabajando.

DROP DATABASE: Borrado permanente de todas las tablas de nuestra base de datos y borrar, así, dicha base de datos. Evidentemente, es un **comando muy peligroso**, es por ello por lo que tendremos que tener específicamente habilitado el permiso de DROP. Si la base de datos que estamos borrando está enlazada, se borrarán ambos objetos.

DROP INDEX: Se ejecutará un ALTER TABLE **justo para borrar el índice** que estamos indicando.

DROP TABLE: Podremos **borrar una o más tablas en nuestra base de datos**. El permiso DROP debe estar habilitado para nuestro usuario de la base de datos.
Deberá de ser ejecutado con mucha precaución, ya que **toda la información de la tabla que estamos indicando será eliminada**. Podremos usar la clave "IF EXISTS" para evitar el error de cuando la tabla no exista. Este comando DROP TABLE realizará commit automáticamente al ser ejecutado.

RENAME TABLE: Con este comando podremos renombrar una o más tablas.

3. SENTENCIAS MANIPULACIÓN DE DATOS (DML) INSERCIÓN:

- **DELETE**
- **DO** → Ejecutamos expresiones sin obtener resultado. Es una forma de realizar SELECT, pero con la ventaja de que es más rápido.
- **HANDLER** → Accederemos a las distintas interfaces del motor de la tabla, tendremos comandos complementarios como **OPEN, READ o CLOSE** para leer ciertos datos de dicha tabla.
- **INSERT**
- **LOAD DATA INFILE:** Leer registros.

4. SENTENCIAS MANIPULACIÓN DE DATOS (DML) EDICIÓN:

- **REPLACE**
- **SELECT** → Para realizar consultas de registros de una o más tablas. Podremos realizar consultas simples, complejas o subconsultas.
- **TRUNCATE** → Es equivalente a un DELETE, pero con ligeras diferencias. Dependiendo del motor de base de datos, en algunos se realiza un DELETE y para otros motores de base de datos, se elimina el objeto completo.
- **UPDATE**

5. EJECUCIÓN DE CONSULTAS I: SELECT:

Podemos encontrar cláusulas de tipo **HAVING**, también podemos encontrar **ORDER BY**, **UNION**… La sentencia SELECT se puede combinar de diversas formas.
Además, obtenemos una tabla ficticia, una serie de resultados que se relacionan acorde a nuestros requisitos en la consulta. (ALL, DISTINCT, AS, FROM, WHERE)

6. EJECUCIÓN DE CONSULTAS II: WHERE:

• Operadores: podemos encontrar mayor que (“>”), mayor o igual que (“>=”), menor que (“<”), igual (“=”), distinto que podemos expresarlo: (“<>”) o también: (“!=”).
• Nulos: podremos añadir **IS NULL** o **IS NOT NULL** si lo que queremos es comprobar que el valor de cierta columna sea nulo o no. Un valor es nulo cuando no existe valor, un registro en blanco no sería nulo.
• LIKE: Cuando queramos realizar una **comparación con los registros**. Se usarán **caracteres especiales que son “_” y también “%”**. Con el “%” indicamos que puede ir cualquier cadena de caracteres en esa posición, y con el “_” sería el mismo concepto, pero, en este caso, solo con un carácter.
• BETWEEN: Cuando queramos **establecer un rango de valores**.
• IN (): Cuando nuestro objetivo sea mostrar una serie de resultados cuyos valores coincidan con los especificados en la clave.

7. EJECUCIÓN DE CONSULTAS III: EJEMPLOS:

- **Operadores lógicos:** OR, AND y NOT.
- **ORDER BY:** situaremos dicha cláusula después del WHERE, y la usaremos para establecer un orden a la hora de mostrar resultados según el campo o campos por los que queramos ordenar. → ASC o DESC para ordenar ascendente o descendente.

8. GESTIÓN DE TRANSACCIONES:

Una transacción en SQL son **unidades o conjuntos de acciones que se realizan en serie y de forma ordenada en el sistema gestor de base de datos**.
Los objetivos de las transacciones son dos:
• Proporcionar consistencia en la base de datos realizando secuencias de alta fiabilidad, de tal forma que se pueda volver a estados anteriores fácilmente.
• Ofrecer aislamiento cuando más de un aplicativo está accediendo a los datos simultáneamente.

Los **comandos de control** que se realizan para la ejecución de **transacciones** en SQL:
• **Commit:** con este comando se persistirán los cambios en base de datos.
• **Rollback**: desharemos los cambios que se hubieran ejecutado hasta el momento y se abandonará la transacción.
• **Savepoint:** puntos donde se podrá almacenar y, en caso de rollback, se podrá volver a dicho punto de control.

8.1 - La interfaz Statement:

Es la encargada de ejecutar dichas sentencias en nuestro aplicativo y recoger los resultados para manipularlos más tarde.
Una vez **se crea el objeto Statement**, **disponemos de un lugar adecuado para realizar consultas SQL**. Podremos usar diferentes métodos para ello:

• **executeQuery (String):** utilizamos este método para realizar sentencias SELECT, siendo consultas que como resultado, este método nos devolverá un objeto ResultSet con toda la información resultante.
• **executeUpdate (String):** este método lo usaremos para realizar sentencias de manipulación de datos, ya sean INSERT, DELETE, UPDATE, etc. Una vez ejecutada la sentencia que le indiquemos, como String, nos devolverá un entero que contiene la cantidad de filas que han sido afectadas en la operación.
• **execute (String):** podemos usar este método para ejecutar cualquiera de las acciones propuestas en los dos casos anteriores. Simplemente, destacar que este método devolverá True si devuelve un Resultset, y para acceder a él, tendremos que ejecutar el método **getResultSet ()**, o false, si lo que estamos ejecutando, por ejemplo, es un UPDATE, en ese caso, si queremos saber las filas afectadas consultaríamos el método **getUpdateCount ()**.