Bases de Datos

Tema 5

Tratamiento de datos con SQL

Tema 5: Tratamiento de datos con SQL

1. **Instrucciones SQL para la definición de datos (DDL)**
    1. Crear bases de datos
    2. Acceder a una base de datos
    3. Eliminar bases de datos
    4. Crear tablas
    5. Modificar tablas (añadir, modificar y eliminar columnas)
    6. Eliminar tablas
2. **Instrucciones SQL para la manipulación de datos (DML)**
    1. Insertar registros de una tabla
    2. Actualizar registros de una tabla
    3. Eliminar registros de una tabla

1.- Instrucciones SQL para la definición de datos

El DDL, lenguaje de definición de datos (en inglés *Data Definition Language*), es el lenguaje que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos.

En SQL existen cuatro operaciones básicas en lo que se refiere a la creación y el tratamiento de una base de datos y sus tablas:

1. CREATE – Crear una BBDD o una tabla
2. ALTER – Modificar una tabla
3. DROP – Borrar una BBDD o una tabla
4. TRUNCATE – Borrar una tabla

1.- Instrucciones SQL para la definición de datos

Creación de una bases de datos

La instrucción CREATE sirve tanto para crear bases de datos como tablas si bien crear una base de datos resulta algo extremadamente sencillo:

Sintaxis para crear una BBDD:

```
CREATE DATABASE mi_db;
```

En entornos de tipo Unix, los nombres de las bases de datos son case sensitive (al contrario que las palabras clave), de modo que siempre debe referirse a su base de datos como mi_bbdd, y no Mi_bbdd, MI_BBDD o variantes similares. Aunque esta restricción no existe en Windows sí se recomienda.

1.- Instrucciones SQL para la definición de datos

Acceder a una bases de datos

Al crear una base de datos, ésta no se selecciona para su uso, debe hacerlo explícitamente. Para convertir a menagerie en la base de datos actual, se usa el comando USE:

```
mysql> USE menagerie
Database changed
```

Las bases de datos sólo necesitan ser creadas una vez, pero deben ser seleccionadas cada vez que se inicia una sesión de mysql. Puede hacerse a través del comando **USE** como se muestra en el ejemplo, o puede indicar la base de datos en la línea de comandos al ejecutar mysql. Simplemente debe indicar el nombre de la base de datos a continuación de los parámetros que necesite ingresar.

Por ejemplo:

```
shell> mysql -h host -u user -p password
Enter password: ********
```

1.- Instrucciones SQL para la definición de datos

Eliminar una base de datos

Para la eliminación de una base de datos utilizaremos la instrucción **DROP DATABASE**. Hay que tener cuidado con esta instrucción, pues eliminará nuestra BBDD por completo y de forma definitiva junto con todos los datos que esta contenga.

Sintaxis:

```
DROP DATABASE database_name;
```

1.- Instrucciones SQL para la definición de datos

Creación de tablas

La creación de una tabla se lleva a cabo mediante el comando **CREATE TABLE** y permite, además, construir sus columnas así como los tipos de datos asociados a ellas.

Sintaxis:

```
CREATE TABLE table_name (
  column_name1 data_type(size),
  column_name2 data_type(size),
  column_name3 data_type(size),
  ....
  );
```

Como se puede observar, esta sintaxis es muy similar a como describíamos las tablas a lo largo del primer trimestre.

1.- Instrucciones SQL para la definición de datos

Creación de tablas

Tras el tipo de dato se puede también indicar si alguna de las columnas debe contar con algún tipo de restricción. Las restricciones más comunes en SQL son las siguientes:

- **NOT NULL** – Indica que la columna no puede tener un valor de tipo NULL
- **UNIQUE** – Especifica que cada valor de la columna debe ser único
- **PRIMARY KEY** – Especifica que la columna (o una combinación de varias columnas) serán clave primaria capaz de identificar cada registro de manera unívoca. Una combinación de las restricciones NOT NULL y UNIQUE
- **FOREIGN KEY** – Establece que un campo será considerado como Clave Ajena a otra tabla, propiciando la integridad referencial de los datos (hay que indicar la tabla y la clave de esta a la que se hace referencia)
- **AUTO_INCREMENT** - Especificar que un campo sea de autoincremental
- **DEFAULT** – Permite especificar un valor por defecto para esa columna

1.- Instrucciones SQL para la definición de datos

Creación de tablas – ejemplo (Access / SQL Server / Oracle)

```
CREATE TABLE Personas
  (
  PersonaID int PRIMARY KEY AUTOINCREMENT,
  Nombre varchar(255) NOT NULL,
  Apellidos varchar(255),
  Direccion varchar(255),
  Ciudad int FOREIGN KEY REFERENCES Ciudades(C_Id),
  Telefono int UNIQUE,
  Estado_civil varchar(255) DEFAULT 'soltero'
);
```

1.- Instrucciones SQL para la definición de datos

Creación de tablas – ejemplo (MySQL)

```
CREATE TABLE Personas
  (
  PersonaID int NOT NULL   AUTO_INCREMENT,
  Nombre varchar(255) NOT NULL,
  Apellidos varchar(255),
  Direccion varchar(255),
  Ciudad varchar(255),
  Telefono int,
  Estado_civil varchar(255) DEFAULT 'soltero',
  PRIMARY KEY (PersonaID),
  FOREIGN KEY (Ciudad) REFERENCES Ciudades(C_Id),
  UNIQUE (Telefono)
);
```

1.- Instrucciones SQL para la definición de datos

Modificar tablas – Añadir columnas

ALTER TABLE es la instrucción de SQL orientada a añadir, borrar o modificar columnas de una tabla existente.

Sintaxis para añadir una nueva columna:

```
ALTER TABLE table_name ADD column_name datatype;
```

Ejemplo:

```
ALTER TABLE alumnos ADD edad INT;
```

1.- Instrucciones SQL para la definición de datos

Modificar tablas – Modificar columnas

Sintaxis para modificar una columna:

En SQL Server / MS Access:

```
ALTER TABLE table_name ALTER COLUMN column_name datatype
```

En My SQL / Oracle:

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype
```

Ejemplo:

```
ALTER TABLE alumnos ALTER COLUMN nota INT;
ALTER TABLE alumnos MODIFY COLUMN nota INT;
```

1.- Instrucciones SQL para la definición de datos

Modificar tablas – Borrar columnas

Sintaxis para borrar una columna:

```
ALTER TABLE table_name DROP COLUMN column_name
```

Ejemplo:

```
ALTER TABLE alumnos DROP COLUMN edad;
```

1.- Instrucciones SQL para la definición de datos

Eliminar tablas

Para la eliminación de tablas existen dos instrucciones, DROP y TRUNCATE. Aunque aparentemente ambas sirven para lo mismo, hay una sutil pero importante diferencia entre ambas.

Sintaxis:

```
DROP TABLE alumnos;
TRUNCATE TABLE alumnos;
```

Mientras que DROP elimina la tabla y su contenido, el comando TRUNCATE elimina sólo el contenido de una tabla (la vacía). La ventaja sobre el comando DROP es que resulta mucho más rápido, especialmente si la tabla es muy grande. En realidad TRUNCATE borra la tabla y la vuelve a crear sin ejecutar ninguna transacción. La desventaja es que TRUNCATE sólo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE, cosa que DROP sí puede hacer.

2.- Instrucciones SQL para la manipulación de datos

Un lenguaje de manipulación de datos (*Data Manipulation Language*, o *DML* en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

En el capítulo anterior habíamos trabajado con las instrucciones SQL relacionadas con las consultas, pero en este tema lo haremos con las orientadas a la inserción y el mantenimiento de los datos en una base de datos. Las instrucciones que proporciona SQL para ello son las siguientes:

1. INSERT | INSERTAR datos en una tabla
2. UPDATE | ACTUALIZAR datos en una tabla
3. DELETE | BORRAR datos de una tabla

2.- Instrucciones SQL para la manipulación de datos

Insertar registros

Para la inserción de nuevos registros en una tabla mediante SQL utilizamos las instrucciones INSERT INTO … VALUES:

Sintaxis:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Ejemplo:

```
INSERT INTO clientes (numcli, nombre, ciudad, pais)
VALUES (2234,'Tom B. Erichsen', 'Stavanger','Noruega');
```

2.- Instrucciones SQL para la manipulación de datos

Modificar registros

Para actualizar valores de registros en una tabla utilizamos las instrucciones UPDATE … SET junto con un criterio WHERE, de forma similar a como solemos operar con las consultas SELECT:

Sintaxis:

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Ejemplo:

```
UPDATE clientes
SET nombre='Alfred Schmidt', ciudad='Hamburg'
WHERE nombre='Alfreds Xchmidt';
```

2.- Instrucciones SQL para la manipulación de datos

Modificar registros

Las consultas de actualización también pueden realizarse sin cláusula WHERE, pero hay que tener un especial cuidado en estos casos, dado que podríamos terminar actualizando más datos de los que queremos. Por ejemplo, la siguiente actualización:

```
UPDATE clientes
SET nombre='Alfred Schmidt', ciudad='Hamburg';
```

Actualizaría **TODOS** los registros de mi tabla en las columnas indicadas, implicando la destrucción definitiva de la información original.

2.- Instrucciones SQL para la manipulación de datos

Borrado de registros

Para eliminar registros de una tabla mediante SQL utilizamos la instrucción **DELETE FROM** junto con una cláusula WHERE para indicar qué registros se deben borrar**:**

Sintaxis:

```
DELETE FROM table_name
WHERE some_column=some_value;
```

Ejemplo:

```
DELETE FROM clientes
WHERE nombre='Alfreds Futterkiste' AND ciudad='Castellón';
```

2.- Instrucciones SQL para la manipulación de datos

Borrado de registros

Al igual que en el caso de las consultas de actualización, las consultas de eliminación también pueden realizarse sin cláusula WHERE, pero de nuevo hay que se extremadamente cuidadosos porque borraremos todos los registros de la tabla (sin borrar la tabla):

```
DELETE FROM clientes;
```

O también

```
DELETE * FROM clientes;
```

El borrado de estos registros no se puede deshacer.