Trabajar con una Base de Datos usando *'SQL Server Express'*

Hay que hacerlo con la '***Herramienta de línea de comandos de Microsoft (R) SQL Server'*** **SQLCMD**

SQLCMD.exe es un programa que hay que usarlo desde la consola.

Resumen de sintaxis de SQLCMD:

```
Sqlcmd            [-U id. de inicio de sesión]          [-P contraseña]
  [-S servidor]            [-H nombre de host]          [-E conexión de confianza]
  [-d usar nombre de base de datos] [-l tiempo de espera de inicio de sesión]
  [-t tiempo de espera de consulta]
  [-h encabezados]           [-s separador de columna]      [-w ancho de pantalla]
  [-a tamaño de paquete]        [-e entrada de eco]        [-I habilitar identificadores entre comillas]
  [-c fin de comando]            [-L[c] listar servidores[salida limpia]]
  [-q "consulta de línea de comandos"]   [-Q "consulta de línea de comandos" y salir]
  [-m nivel de error]         [-V nivel de gravedad]     [-W quitar espacios finales]
  [-u salida Unicode]    [-r[0|1] mensajes a stderr]
  [-i archivo de entrada]          [-o archivo de salida]        [-z nueva contraseña]
  [-f <páginaDeCódigos> | i:<páginaDeCódigos>[,o:<páginaDeCódigos>]] [-Z nueva contraseña y salir]
  [-k[1|2] quitar[reemplazar] caracteres de control]
  [-y ancho de pantalla de longitud variable]
  [-Y ancho de pantalla de longitud fija]
  [-p[1] imprimir estadísticas[formato dos puntos]]
  [-R usar configuración regional de cliente]
  [-b anular por lotes si hay errores]
  [-v var = "valor"...]  [-A conexión de administrador dedicada]
  [-X[1] deshabilitar comandos, scripts de inicio, variables de entorno [y salir]]
  [-x deshabilitar sustitución de variable]
   [-? mostrar resumen de sintaxis]
```

Para ejecutar SQLCMD lo mejor es localizar donde está el archivo sqlcmd.exe y hacerse el siguiente **archivo BAT:**

(en mi caso está en: *C:\Archivos de programa\Microsoft SQL Server\100\Tools\Binn*)

```
cd C:\Archivos de programa\Microsoft SQL Server\100\Tools\Binn\
SQLCMD.EXE -S localhost\SqlExpress
```

Ahora aparecerá la consola de SQL Server e iremos escribiendo cada instrucción SQL en una línea diferente.

También podemos ejecutar las siguientes órdenes de SQLCMD:

:!! [<comando>]
- Ejecuta un comando en el shell de comandos de Windows.

:connect server[\instancia] [-l tiempo de espera] [-U usuario [-P contraseña]]
- Se conecta a una instancia de SQL Server.

:ed
- Edita la caché de instrucción actual o ejecutada en último lugar.

:error <destino>
- Redirige el resultado del error a un archivo, stderr o stdout.

:exit
- Sale de sqlcmd inmediatamente.

:exit()
- Ejecuta la caché de instrucción; sale sin valor de retorno.

:exit(<consulta>)
- Ejecuta la consulta especificada; devuelve resultado numérico.

go [<n>]
- Ejecuta la caché de instrucción (n veces).

:help
- Muestra esta lista de comandos.

:list
- Imprime el contenido de la caché de instrucciones.

:listvar
- Lista las variables de scripts sqlcmd establecidas.

:on error [exit|ignore]
- Acción para errores de comandos de lotes o sqlcmd.

:out <nombreDeArchivo>|stderr|stdout
- Redirige el resultado de la consulta a un archivo, stderr o stdout.

:perftrace <nombreDeArchivo>|stderr|stdout
- Redirige el resultado de a un archivo, stderr o stdout.

:quit
- Sale de sqlcmd inmediatamente.

:r <nombreDeArchivo>
- Anexa el contenido del archivo a la caché de instrucción.

:reset
- Descarta la caché de instrucción.

:serverlist
- Lista servidores locales y de SQL Server de la red.

:setvar {variable}
- Quita una variable de script sqlcmd.

:setvar <variable> <valor>
- Establece una variable de scripts sqlcmd.

Tras escribir varias instrucciones SQL para ejecutarlas escribiremos: GO

Ejemplo (Crearemos una Base de Datos llamada ***tfnos***, allí dentro crearemos una Tabla llamada ***teléfonos***, con los campos: *Nombre*, *Direccion*, *Telefono* y *Comentarios*. Luego insertaremos varios registros en esa tabla y visualizaremos la tabla

```
CREATE DATABASE tfnos   Crea una nueva Base de Datos llamada tfnos
USE tfnos   Usa la Base de Datos existente llamada tfnos
CREATE TABLE teléfonos (                                   Crea una tabla con los sig. campos:
                        nombre CHAR(30) NOT NULL,
                        dirección CHAR(30) NOT NULL,
                        teléfono CHAR(12) PRIMARY KEY NOT NULL,
                        observaciones CHAR(240)
                       )
INSERT INTO telefonos VALUES ('Juan Perez', 'Zaragoza',
                              '976111222','Es el profesor')        Inserta una fila de datos
SELECT * FROM teléfonos                                    Muestra la tabla teléfonos en la que se han seleccionado todos sus campos
```

En las cuatro siguientes páginas hay un buen resumen de las instrucciones SQL:

Crear una base de datos

Para crear una base de datos, SQL proporciona la sentencia CREATE DATABASE cuya sintaxis es:

```
CREATE DATABASE <base de datos>
```

Esta sentencia especifica el nombre de la base de datos que se desea crear. Cuando desee eliminarla, ejecute la sentencia:

```
DROP DATABASE <base de datos>
```

Crear una tabla

Para crear una tabla, SQL proporciona la sentencia CREATE TABLE. Esta sentencia especifica el nombre de la tabla, los nombres y tipos de las columnas de la tabla y las claves primaria y externa de esa tabla (también llamada extranjera, en el sentido de que es importada de otra tabla). Su sintaxis es la siguiente:

```
CREATE TABLE <tabla>(<columna 1> [,<columna 2>]...)
```

donde *columna n* se formula según la sintaxis siguiente:

```
<columna n> <tipo de dato> [DEFAULT <expresión>]
    [<constante 1> [<constante 2>]...]
```

Algunos de los tipos de datos más utilizados son los siguientes:

Tipo SQL	*Tipo SQL de .NET Framework*	*Tipo MS Access*
INTEGER	SqlInt32	Número entero largo
REAL	SqlSingle	Número simple
FLOAT	SqlDouble	Número doble
CHAR	SqlString	Texto
VARCHAR	SqlString	Texto
BINARY	SqlBinary	Binario
DATE	SqlDateTime	Fecha/Hora

La cláusula DEFAULT permite especificar un valor por omisión para la columna y, opcionalmente, para indicar la forma o característica de cada columna, se pueden utilizar las constantes: NOT NULL (no se permiten valores nulos: NULL), UNIQUE o PRIMARY KEY.

La cláusula PRIMARY KEY se utiliza para definir la columna como clave principal de la tabla. Esto supone que la columna no puede contener valores nulos ni duplicados; es decir, que dos filas no pueden tener el mismo valor en esa columna. Una tabla puede contener una sola restricción PRIMARY KEY.

La cláusula UNIQUE indica que la columna no permite valores duplicados; es decir, que dos filas no pueden tener el mismo valor en esa columna. Una tabla puede contener varias restricciones UNIQUE. Se suele emplear para que el propio sistema compruebe que no se añaden valores que ya existen.

El ejemplo que se muestra a continuación crea la tabla *telefonos*, en la base de datos con la que estemos trabajando, con las columnas *nombre*, *direccion*, *telefono* y *observaciones* de los tipos especificados. La columna *telefono* es la clave principal; esto implica que en esa columna todos los valores tienen que ser diferentes y no nulos. El resto de las columnas, excepto *observaciones*, tampoco permiten valores nulos:

```
CREATE TABLE telefonos(
                nombre         CHAR(30) NOT NULL,
                direccion      CHAR(30) NOT NULL,
                telefono       CHAR(12) PRIMARY KEY NOT NULL,
                observaciones CHAR(240)
  )
```

La diferencia entre los tipos *CHAR (n)* y *VARCHAR (n)* es que en el primer caso, el campo se rellena con espacios hasta *n* caracteres (longitud fija) y en el segundo no (longitud variable).

Escribir datos en la tabla

Para escribir datos en una tabla, SQL proporciona la sentencia INSERT. Esta sentencia agrega una o más filas nuevas a una tabla. Su sintáxis, de forma simplificada, es la siguiente:

```
INSERT [INTO] <tabla> [(<columna 1>[,<columna 2>]...)]
                      VALUES (<expresión 1>[,<expresión 2>]...),...

INSERT [INTO] ... SELECT ... FROM ...
```

donde *tabla* es el nombre de la tabla en la que se desean insertar las filas, argumento que va seguido por una lista con los nombres de las columnas que van a recibir los datos especificados por la lista de valores que siguen a la cláusula VALUES. Las columnas no especificadas en la lista reciben el valor NULL, si lo permiten, o el valor predeterminado, si se especificó. Si todas las columnas reciben datos, se puede omitir la lista con los nombres de las columnas.

Con respecto al segundo formato, un poco más adelante veremos la sentencia SELECT.

El ejemplo que se muestra a continuación añade a la tabla *telefonos* una nueva fila con los valores de las columnas especificados:

```
INSERT INTO telefonos
       VALUES ('Leticia Aguirre Soriano','Madrid',
               '912345671','Ninguna')
```

Modificar datos de una tabla

Para modificar datos en una tabla, SQL proporciona la sentencia UPDATE. Esta sentencia puede cambiar los valores de filas individuales, grupos de filas o todas las filas de una tabla. Su sintaxis es la siguiente:

```
UPDATE <tabla>
    SET <columna 1 = (<expresión 1> | NULL)
        [, <columna 2 = (<expresión 2> | NULL)]...
    WHERE <condición de búsqueda>
```

La cláusula SET contiene una lista separada por comas de las columnas que deben actualizarse y el nuevo valor de cada columna. El valor suministrado por las expresiones incluye elementos tales como constantes, valores seleccionados de una columna de otra tabla, o valores calculados por una expresión compleja. Y la cláusula WHERE especifica la condición de búsqueda que define la fila de la tabla cuyas columnas se desean modificar.

El ejemplo que se muestra a continuación modifica en la tabla *telefonos* la dirección de la persona cuyo teléfono se especifica:

```
UPDATE telefonos
    SET direccion='Triana, Sevilla'
    WHERE telefono='91234567'
```

Borrar registros de una tabla

Para borrar registros en una tabla, SQL proporciona la sentencia DELETE. Esta sentencia quita una o varias filas de una tabla. Una forma simplificada de la sintaxis de DELETE es:

```
DELETE FROM <tabla> WHERE <condición de búsqueda>
```

El argumento *tabla* nombra la tabla de la que se van a eliminar las filas. Se eliminan todas las filas que reúnan los requisitos de la condición de búsqueda de la cláusula WHERE. Si no se especifica una cláusula WHERE, se eliminan todas las filas de la tabla.

Cualquier tabla de la que se hayan quitado todas las filas sigue permaneciendo en la base de datos. La instrucción DELETE sólo elimina filas de la tabla; si se quiere quitar la tabla de la base de datos, hay que ejecutar la sentencia:

```
DROP TABLE <tabla>
```

El ejemplo que se muestra a continuación quita de la tabla *telefonos* el alumno que tiene la clave especificada:

```
DELETE FROM telefonos WHERE telefono='958324555'
```

Seleccionar datos de una tabla

Para seleccionar datos de una tabla, SQL proporciona la sentencia SELECT. Las cláusulas principales de esta sentencia se pueden resumir del modo siguiente:

```
SELECT [ALL | DISTINCT] <lista de selección>
    FROM <tablas>
    WHERE <condiciones de selección>
    [ORDER BY <columna 1> [ASC|DESC][, <columna 2> [ASC|DESC]]...]
```

Las cláusulas de una instrucción SELECT deben especificarse en el orden indicado.

El argumento *lista de selección* describe las columnas del conjunto de resultados. Es una lista de expresiones separadas por comas. Cada expresión de lista de selección suele ser una referencia a una columna de la tabla de la que provienen los datos, aunque puede ser cualquier otra expresión. Al usar la expresión * en una lista de selección se especifica que se devolverán todas las columnas de la tabla de origen.

La cláusula DISTINCT elimina las repeticiones del conjunto de resultados obtenido por SELECT y ALL especifica que pueden aparecer filas duplicadas en el conjunto de resultados; es el valor predeterminado.

La cláusula FROM especifica una lista de las tablas de donde se recuperan los datos del conjunto de resultados.

La cláusula WHERE describe un filtro que define las condiciones que debe cumplir cada fila de las tablas de origen para satisfacer los requisitos de la instrucción SELECT. Sólo las filas que cumplen las condiciones contribuyen con datos al conjunto de resultados. Los datos de las filas que no cumplen las condiciones no se usan.

La cláusula ORDER BY define el orden de las filas del conjunto de resultados y especifica las columnas que intervienen en la clasificación. Las palabras clave ASC y DESC se utilizan para especificar si las filas se ordenan en una secuencia ascendente o descendente. Por omisión se supone ASC.

El ejemplo siguiente lista todas las filas de la tabla *telefonos*:

```
SELECT * FROM telefonos
```

Este otro ejemplo lista todas las filas de la tabla *alumnos* ordenadas ascendentemente por el campo *nombre*:

```
SELECT * FROM telefonos ORDER BY nombre
```

SQL permite utilizar los operadores <, <=, >, >=, <>, AND, OR, NOT, IS NULL, LIKE, BETWEEN, IN, ALL, ANY, etc. El ejemplo siguiente lista los registros de la tabla *telefonos* cuyo teléfono sea mayor que 958000000:

```
SELECT * FROM telefonos WHERE telefono>'958000000'
```

El siguiente ejemplo lista todos los registros de la tabla *telefonos* que empiecen por 91:

```
SELECT * FROM telefonos WHERE telefono LIKE '91*'
```