Tanto si vamos a empezar a trastear con Microsoft SQLServer como si lo vamos a usar de forma seria, deberemos saber que hay unos pasos aconsejados para interactuar con el motor de base de datos. El protocolo sería el siguiente:

1 – Creamos nuestro propio usuario admin, ejecutando:

```
use master;
create login DBAdmin1 with password = 'Pass1234';
create user DBAdmin1 for login DBAdmin1;
```

...donde Pass1234 puede ser la contraseña que queramos.

2 – Concedemos permisos de control del servidor al usuario que acabamos de crear, ejecutando:

```
use master;
grant control server to DBAdmin1 with grant option;
```

3 – Creamos la base de datos con la que queremos empezar a trabajar, ejecutando:

```
 create database NombreDeLaBaseDeDatos;
```

4 – Creamos las tablas e insertamos todos los datos en la base de datos siguiendo este ejemplo básico:

```
use NombreDeLaBaseDeDatos;
create table NombreDeLaTabla (columna1 varchar, columna2 varchar);
insert into NombreDeLaTabla values ('a','b');
```

5 – Comprobamos que los datos se hayan agregado haciendo un simple select, con:

```
use NombreDeLaBaseDeDatos;
select * from NombreDeLaTabla;
```

6 – Creamos los roles para asignarles a los usuarios que van a interactuar con la base de datos, con:

```
use NombreDeLaBaseDeDatos;
create role diseñadores;
create role operadores;
create role consultores;
```

7 – Asignamos permisos DDL, DML y DQL a esos roles, ejecutando:

```
use NombreDeLaBaseDeDatos;
grant control on database::NombreDeLaBaseDeDatos               to diseñadores;
grant insert, update, delete on database::NombreDeLaBaseDeDatos to operadores;
grant select on database::NombreDeLaBaseDeDatos                to consultores;
```

Como vemos, la **forma correcta** de asignar permisos DDL es con el grant «control». La forma clásica en el que lo hacemos en otro software, por ejemplo:

```
 grant create, alter, drop ...;
```

...no nos serviría, puesto que DROP, en SQLServer no es un permiso «grantable».

8 – Creamos los usuarios a los que luego asignaremos los roles, ejecutando:

```
use NombreDeLaBaseDeDatos;
create login diseñador1 with password = 'Contra123';
create user diseñador1 for login diseñador1;
create login operador1 with password = 'Contra456';
create user operador1 for login operador1;
create login consultor1 with password = 'Contra789';
create user consultor1 for login consultor1;
```

9 – Asignamos los correspondientes roles a cada usuario, con:

```
use NombreDeLaBaseDeDatos;
exec sp_addrolemember diseñadores, diseñador1;
exec sp_addrolemember operadores,  operador1;
exec sp_addrolemember consultores, consultor1;
```

Si en algún futuro queremos agregar un usuario directamente a un rol, podemos ejecutar:

```
alter role NombreDelRole add member NombreDeUsuario;
```

Si queremos asignar permisos específicos sobre algunas tablas, por ejemplo la tabla Proveedores y la tabla Productos, ejecutaríamos:

```
grant insert, update, delete, select on dbo.Proveedores to NombreDeUsuarioORol;
grant insert, update, delete, select on dbo.Productos   to NombreDeUsuarioORol;
```

Si quisiéramos asignar permisos específicos sobre algunas columnas, ejecutaríamos:

```
grant select on dbo.Tabla(Columna1, Columna2) to NombreDeUsuarioORol;
```

Podríamos comprobar si a ese usuario se le ha asignado correctamente ese grant ejecutando un comando en su nombre. Imaginemos que el usuario se llama Pedro. Podríamos hacer entonces:

```
execute as user = 'Pedro';
select * from dbo.Tabla;
```

Y veremos que nos saldrá un mensaje denegando el acceso a todas las otras columnas que no sean las dos a las que tenemos acceso. Para poder hacer la consulta correcta, deberemos ejecutar:

```
execute as user = 'Pedro';
select Columna1, Columna2 from dbo.Tabla;
```