**Logins, Usuarios y Permisos en SQL Server**

**Logins o inicios de sesión:**Es la autenticación al motor de Base de Datos SQL SERVER, los inicios de sesión están asociados a un usuario a nivel de base de datos, aunque se tenga el permiso en el login, este podrá acceder al motor, pero no a la base de datos que queremos.  
  
**Usuarios:**Es la autenticación a la base de Datos como tal, todos los usuarios deben estar enlazados con unlogin, si existen usuarios sin enlazar a un login, estos se llamarán **usuarios huérfanos**, que solo ocupan espacio, pero no tienen utilidad, si se encuentran con éstos usuarios será mejor borrarlos.  
  
  
A los logins, se les puede asignar ciertos roles de sistema, pero el más importante es el rol **sysadmin**, si un login tiene este rol, será administrador y tendrá todos los derechos habilitados, así que es importante que los logins que creemos para nuestros usuarios no tenga el rol **sysadmin**, precisamente este rol, está ligado al login **sa**.

|  |
| --- |
| [Diagrama logins y usuarios Sql SERVER](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhoXIqT1spLX2sBmcMGdZzFPZIT-kT6wNv933vyl0mOkyqMb9liD-rgN6r4BZTGEOlu3Pk_rU-BKGaw9Uz94IgxVCLsRBNtZevkzl_LsxKURYPiE5NZxdHU46DM_O3YQdmYVSX6UeyM0FvD/s1600/logins.png) |
| **Diagrama logins y usuarios SQL SERVER** |

Existen dos tipos de logins: los logins de usuario y los logins de Windows; en este último podemos loguearnos al **SQL SERVER** mediante el dominio de Windows.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como vemos en la imagen anterior el inicio de sesión de sistema **sa**, tiene el rol de sistema **sysadmin** por lo cual, con este usuario seremos “dueños” del servidor y podremos realizar por ejemplo un **DROP DATABASE**.  
  
  
  
**Ejemplo 1: Crear un login llamado sql\_test con acceso únicamente a la base de datos master.**  
Primeramente hay que mencionar que los logins se guardan en la base de datos **master.**Existen dos maneras de crear logins, la primera, será mediante código SQL, la otra, será mediante el entorno gráfico. Vamos a empezar con el código TSQL.

CREATE LOGIN [sql\_test] WITH PASSWORD=N'test',

DEFAULT\_DATABASE=[master], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF

GO

Mediante éste script estamos creando un Login **sql\_test**con password **test,**cuya base de datos predeterminada es **master**, el password nunca expirará y tampoco se forzarán a políticas de contraseña segura. Podemos loguearnos con ese login, pero no tendremos acceso a nada, porque faltará el usuario. Pero vamos a comprobarlo.

Para comprobarlo utilizaremos la vista de sistema llama **syslogins,**mediante la siguiente sintaxis:

use master

go

select name, password from sys.syslogins

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Como vemos con esa sentencia SELECT a la vista **syslogins** podemos ver que se encuentra el login **sql\_test** con su password encriptado

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Abriremos una nueva instancia y colocaremos las credenciales correspondientes, el usuario **sql\_test** y la contraseña **test,**y damos clic en Conectar, observaremos que se conectará sin ningún problema, pero al querer conectarnos a una base de datos obtendremos un mensaje de error

use pruebas

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

No se puede conectar a la base de datos **PRUEBAS,**por dos razones, la primera es que el login **sql\_test** no tiene rol **sysadmin** y la segunda es porque no existe un usuario ligado a el login **sql\_test** en la base de datos **PRUEBAS.**, Así que viene un ejemplo más

**Ejemplo 2: Crear un usuario sql\_pruebas ligado al loginsql\_test para que el loginsql\_test, pueda tener acceso a la base de datos PRUEBAS.**

USE [PRUEBAS]

GO

CREATE USER [sql\_pruebas]

FOR LOGIN [sql\_test] WITH DEFAULT\_SCHEMA=[dbo]

GO

Mediante este script crearemos el usuario sql\_pruebas en la base de datos **PRUEBAS**, ligado al login **sql\_test**, con esto le daremos permiso al login de interactuar únicamente con la base de datos PRUEBAS.

Para comprobarlo utilizaremos la vista de sistema llamada **sysusers,**y esa vista esta a nivel de base de datos

use PRUEBAS

go

select u.name [Usuario], sl.name [Login] from sys.sysusers u

join sys.syslogins sl on sl.sid = u.sid

where u.uid <= 6

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Mediante ese script podemos observar los usuarios que existen en la base de datos y a que login esta ligado.

USE [PRUEBAS]

GO

Utilizamos el comando **USE PRUEBAS** y comprobamos que se tiene acceso a la base de datos, sin embargo al querer ejecutar una sentencia **SELECT**obtendremos un error como el siguiente:

use pruebas

go

select \* from empleados

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este error se da debido a que el usuario sql\_pruebas, tiene acceso a la base, sin embargo, no le hemos dado ningún permiso, lo que nos lleva al siguiente tema y ejemplo

**Permisos en SQL SERVER**

**Grant:**El comando GRANT otorga privilegios al usuario o al ROL sobre objetos específicos o sobre todos los objetos de la base de datos, en el ejemplo anterior, al realizar un SELECT apareció un error por no tener permisos sobre la sentencia, así que le daremos permisos hacia todos los objetos. Si no se especifica el objeto, el permiso se asignará para todos los objetos.

**Deny:**El comando DENY deniega privilegios al usuario o al ROL sobre objetos específicos o sobre todos los objetos de la base de datos. Si no se especifica un objeto, el permiso se denegara hacia todos los objetos.

**Revoke:**El comando REVOKE revoca un provilegio concedido o denegado anteriormente al usuario o al ROL sobre objetos específicos. Si no se específica el objeto, el permiso se revocará para hacia todos los objetos

**Ejemplo 3. Otorgar permisos para realizar SELECT a todas las tablas de la Base de Datos PRUEBAS con el usuario sql\_pruebas.**

Para conceder el privilegio de realizar Select a todas las tablas de la base de datos, utilizaremos la sentencia **GRANT**de la siguiente manera.

use PRUEBAS

go

grant select to [sql\_pruebas]

go

Mediante este script concederemos el uso de la sentencia SELECT al usuario sql\_pruebas, hay que tener en cuenta que la otorgación de permisos se realiza sobre el usuario, no sobre el login, el login únicamente proporciona el acceso al server, pero el usuario es independiente por cada base de datos.

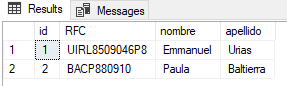
Para realizar la prueba, nuevamente realizaremos el SELECT

use pruebas

go

select \* from empleados

y obtendremos el siguiente resultado:

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjjhrVON_ZI9zETQL8TT9eC2z1IAPF50Zx2f6HrferrC4TiDl95l18Ig_0MQ3gOO_5DJ9sd5gFCe179HT31eyqUxEVJpx2cZh-1OiDuQbK3gar0Ammo2AGZS4_ZM4XMTIf1FSxo8y8wrLxW/s1600/Captura.PNG)

Como vemos, ya tenemos acceso a realizar select en cualquier tabla, pero ¿que pasa si queremos bloquear el select para ciertas tablas?. Pues utilizaremos el comando DENY que veremos a continuación.

**Ejemplo 4. Crear una tabla de auditoria, y no permitir que el usuario sql\_pruebas pueda leer su contenido.**

Para realizar este paso, en una sesión que tenga un login con permisos elevados, crearemos la tabla e insertaremos un registro de prueba.

create table Audit

(id int primary key identity,

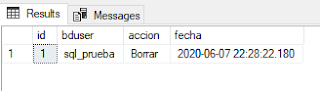
bduser varchar(100),

accion varchar(100),

fecha datetime)

insert into Audit (bduser, accion, fecha) values ('sql\_prueba', 'Borrar', getdate())

select \* from Audit

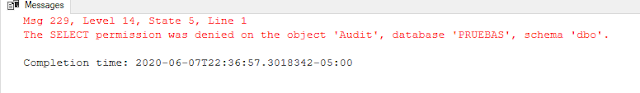
[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgXGxxLK0GSDomE320qVQZgBVsOPCP1nKaWm9m9vYtTIT1gUe586_-zu3hxQ9DYfHaRPgVEUgbsCheaHwgvWLPjqEj_t1tUYhqeL3hLgsEn_GPK1oPbSFMtFBUhOTs3IGq0qPB8vQBhjtll/s1600/Captura.PNG)

Ya tenemos la tabla creada, ahora, esta tabla no queremos que la vea el usuario, por lo cual aplicaremos la sentencia DENY

deny SELECT on object::Audit to [sql\_pruebas]

Para realizar la prueba en una sesión con el loginsql\_test intentaremos realizar un SELECT a la tabla Audit y tendremos un mensaje de error

select \* from audit

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEieTPU7V1qU3GiWHkwkWrcU7rf8Z8ERjvJvPYvgcA-NxLDrL191Jam-HLdFbLWgp-DyCae1gfFVuJjOgurom6rA_k959Vsv3lOaaYKJWdc6BBwvPOGjspePaF_ierrJn7GI05ieLduc0yUk/s1600/Captura.PNG)

Como vemos, esa tabla no puede ser consultada por el usuario sql\_pruebas, sin embargo cualquier otra tabla, si se podrá consultar, estos mismos permisos funcionan para DELETE o INSERT, así como par no permitir crear tablas, triggers, procedimientos almacenados, etc.