

# Seguridad en BD

Cristian ARM

# Tener en cuenta



Limitar el acceso a los datos sensibles tanto por parte de los usuarios como de los procedimientos, es decir, que solo determinados usuarios y procedimientos estén autorizados a realizar consultas en información sensible.



Limitar el uso de los procedimientos importantes solo a usuarios específicos.



Siempre que sea posible, evitar las concurrencias y acceso fuera del horario laboral o habitual.



No todos los datos y no todos los usuarios son creados iguales

## Identificar datos

- Identifica los datos sensibles y los datos críticos
  - **lógica**
  - **arquitectura**



# Cifra la información


- La mejor manera de preservarla es **volverla ilegible** para cualquier persona que llegue a ella sin autorización.




# Enmascaramiento o Anonimización

- alterando los **datos sensibles** para que permanezcan protegidos.
- A partir de esta técnica se cambian los valores **respetando el formato**.
- El método elegido dependerá del administrador, las reglas y formatos que se deban mantener, pero sea cual sea, debe garantizar que el proceso sea **irreversible**; es decir, que no se pueda hacer ingeniería reversa para volver a obtener los datos originales.





# Seguridad en MySQL / MariaDB



# Seguridad Mysql



MySQL trabaja con cuentas (nombre de host, usuario, contraseña y privilegios) organizadas en el diccionario de datos.



Al acceder **primero** se comprueba la identidad del usuario  
**Segundo** los permisos para hacer operaciones sobre la base de datos.



En ambas etapas se usan tablas *user*, *db* y *host*.

# Tablas usadas



*user*: permisos de acceso global.



*host*: permisos de acceso al servidor..



*db*: permisos a nivel de base de datos.



*tables\_priv*: permisos a nivel de tabla.



*columns\_priv*: permisos a nivel de columna.



# Tablas usadas



Los valores en estas tablas son tipo "sí" (Y) y "no" (N).

Un permiso Y, autoriza a realizar la operación.

Un permiso N, hará al sistema pasar a la siguiente tabla (desde *db* hasta todas las *tables\_priv* y *columns\_priv*) hasta dar con permisos positivos.



Si ninguna tabla autoriza al usuario (todas con valor N), la operación será denegada.

# Cada tabla de permisos tiene dos tipos de columnas



**Columnas de alcance:** Determinan el alcance de cada registro en las tablas, es decir, el contexto en que el registro se aplica.



**Columnas de privilegios:** Indican qué privilegios se otorgan a una cuenta sobre ciertas bases de datos, tablas y/o columnas (y otros objetos del SGBD).

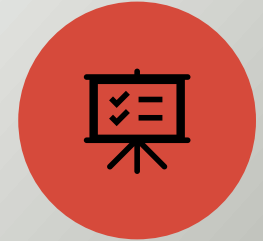
# Al realizar una nueva conexión



EN LA TABLA *USER*, LAS COLUMNAS DE ALCANCE DETERMINAN SI SE RECHAZAN O SE ACEPTAN CONEXIONES ENTRANTES. LAS COLUMNAS DE PRIVILEGIOS OTORGADOS EN ELLA SON DE CARÁCTER GLOBAL (APLICADOS A TODAS LAS BD ALOJADAS EN EL SERVIDOR).



EN LA TABLA *DB*, LAS COLUMNAS DE ALCANCE DETERMINAN QUÉ USUARIOS PUEDEN ACCEDER A QUÉ BASES DE DATOS DESDE QUÉ EQUIPO. LAS COLUMNAS DE PRIVILEGIOS DETERMINAN QUÉ OPERACIONES SE PERMITEN A ESTOS USUARIOS SOBRE ESTAS BD Y SUS TABLAS.



LA TABLA *HOST* SE USA JUNTO A LA TABLA *DB* PARA CONFIGURAR REGISTROS QUE SE APLIQUEN A VARIOS EQUIPOS.

# Creación de un usuario con password seguro

```
mysql> SELECT PASSWORD('mipassword');
```

```
+-----+  
| PASSWORD('mipassword') |  
+-----+  
| *CEE870801502ACAD44FA46CA2CA4F58C2B721A67 |  
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> CREATE USER foo IDENTIFIED BY PASSWORD '*CEE870801502ACAD44FA46CA2CA4F58C2B721A67';
```

```
Query OK, 0 rows affected (0.01 sec)
```

# Saber Permisos de un usuario

```
mysql> SHOW GRANTS for 'foo'@'localhost';
```

```
+-----+
| Grants for foo@localhost                |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'foo'@'localhost' IDENTIFIED BY PASSWORD '...' |
+-----+
```

```
1 row in set (0.00 sec)
```

# Asignar permisos

- `GRANT SELECT (name) ON MyDb.User TO 'MySQLUser'@'MySQLHost';`
- `GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'someuser'@'somehost';`

# Quitar permisos

```
mysql> REVOKE SELECT ON test.* FROM 'foo'@'localhost';
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> REVOKE INSERT ON *.* FROM 'foo'@'localhost';
```

Query OK, 0 rows affected (0.00 sec)

# FLUSH PRIVILEGES

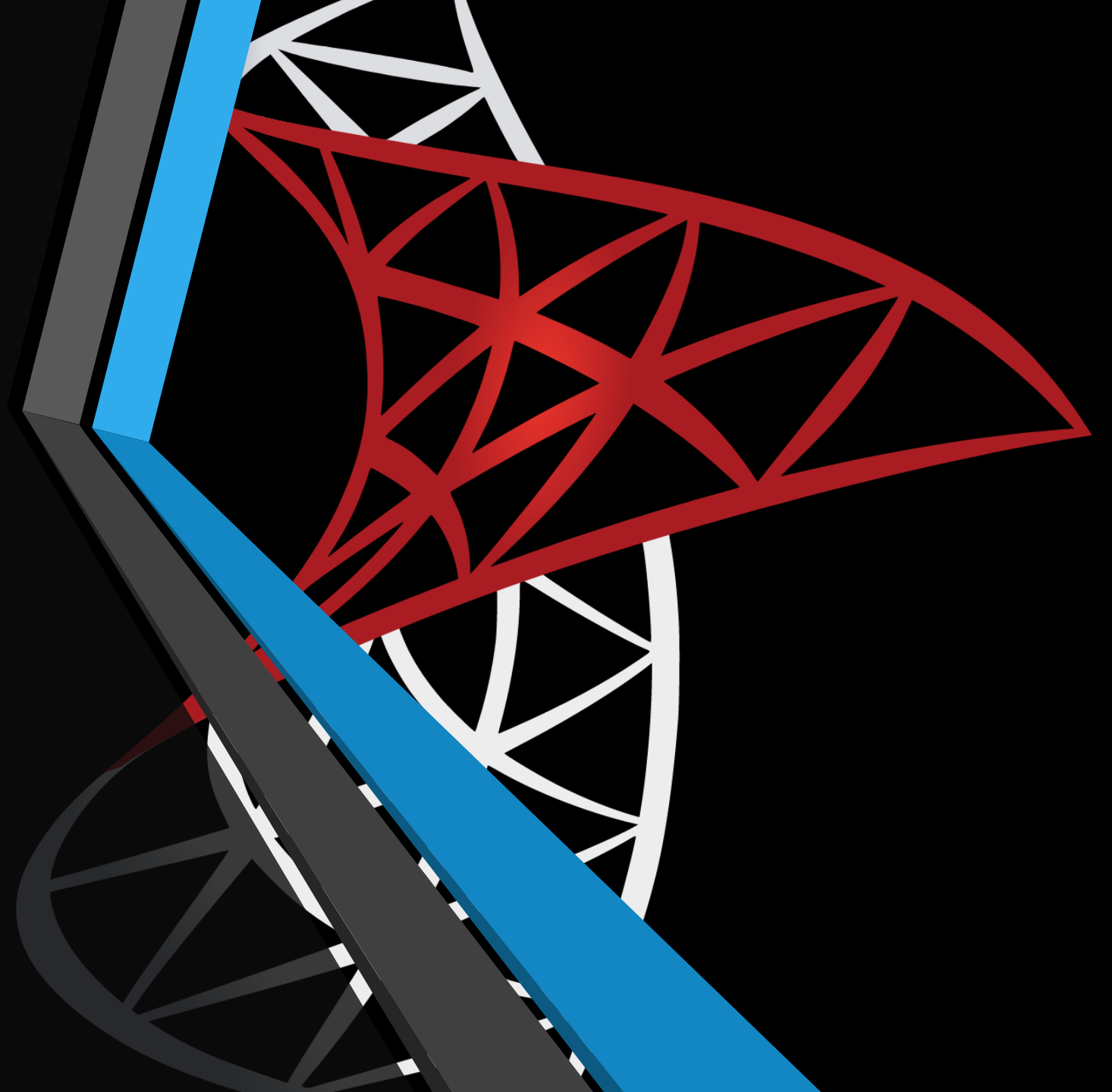
sirve para recargar la tabla de privilegios pero sólo es necesario cuando se **manipulan directamente las tablas de privilegios** ejecutando INSERT, DELETE, etc en lugar de usar los comandos GRANT y REVOKE:

```
mysql> FLUSH PRIVILEGES;
```

```
Query OK, 0 rows affected (0.01 sec)
```



# Seguridad en SQLServer

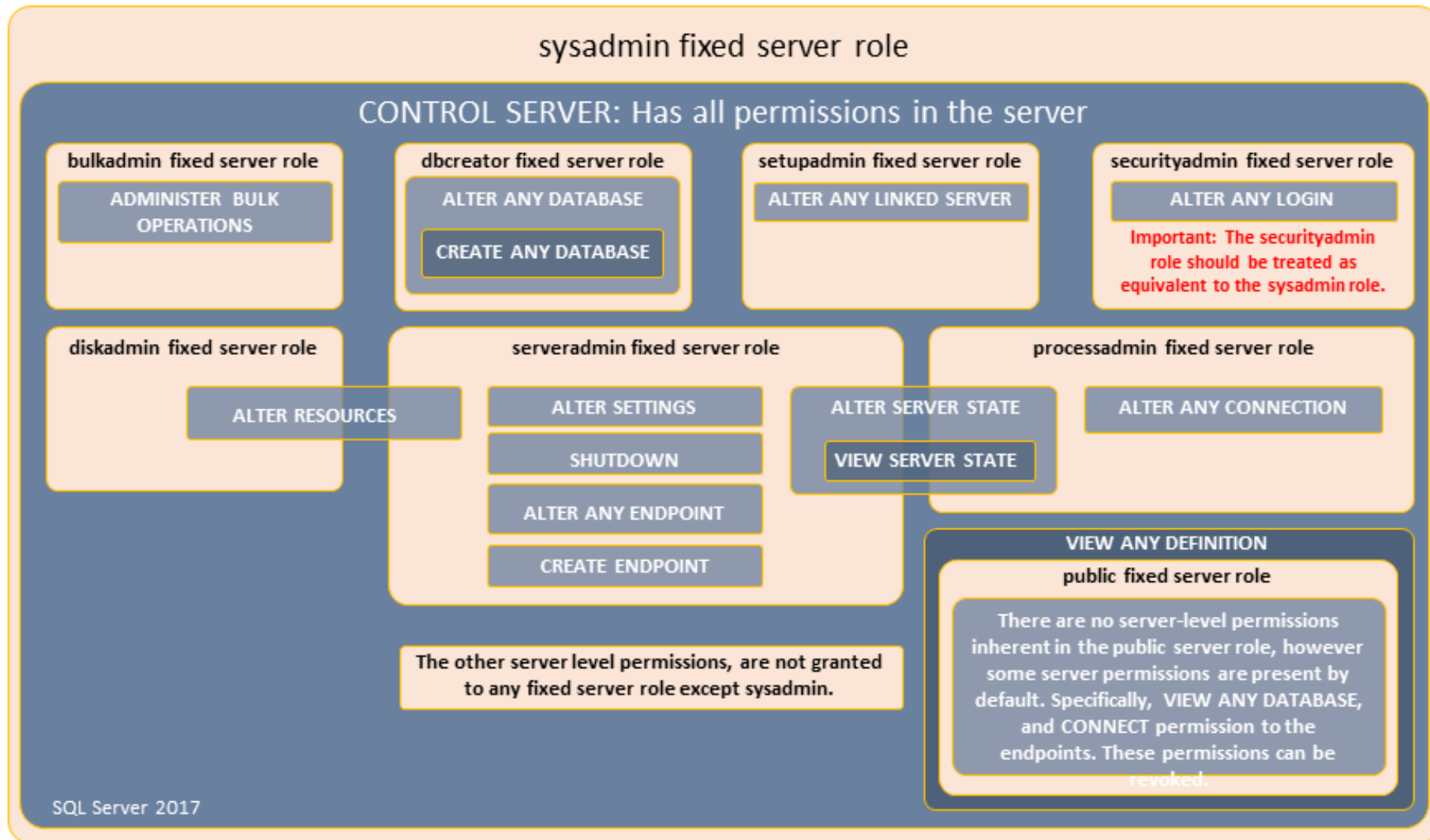


## Roles fijos de nivel de servidor

- **IMPORTANTE:** La capacidad de conceder acceso a Motor de base de datos y configurar los permisos de usuario permite que el administrador de seguridad asigne la mayoría de los permisos de servidor. El rol **securityadmin** se debe tratar como equivalente al rol **sysadmin**.
- **Nota:** El rol **public** se implementa de manera diferente que otros roles y los permisos se pueden conceder, denegar o revocar de los roles fijos de servidor públicos.

Rol fijo de nivel de servidor	Descripción
sysadmin	Los miembros del rol fijo de servidor sysadmin pueden realizar cualquier actividad en el servidor.
serveradmin	Los miembros del rol fijo de servidor serveradmin pueden cambiar opciones de configuración en el servidor y cerrar el servidor.
securityadmin	Los miembros del rol fijo de servidor securityadmin administran los inicios de sesión y sus propiedades. Pueden administrar los permisos de nivel de servidor GRANT, DENY, y REVOKE. También pueden administrar los permisos de nivel de base de datos GRANT, DENY y REVOKE si tienen acceso a una base de datos. Asimismo, pueden restablecer contraseñas para inicios de sesión de SQL Server.
processadmin	Los miembros del rol fijo de servidor processadmin pueden finalizar los procesos que se ejecutan en una instancia de SQL Server.
setupadmin	Los miembros del rol fijo de servidor setupadmin pueden agregar y quitar servidores vinculados mediante instrucciones de Transact-SQL. (Es necesaria la pertenencia a sysadmin cuando se usa Management Studio).
bulkadmin	Los miembros del rol fijo de servidor bulkadmin pueden ejecutar la instrucción BULK INSERT.
diskadmin	El rol fijo de servidor diskadmin se usa para administrar archivos de disco.
dbcreator	Los miembros del rol fijo de servidor dbcreator pueden crear, modificar, quitar y restaurar cualquier base de datos.
public	Cada inicio de sesión de SQL Server pertenece al rol de servidor public. Cuando a una entidad de seguridad de servidor no se le han concedido ni denegado permisos específicos para un objeto protegible, el usuario hereda los permisos concedidos al rol pública para ese elemento. Solo asigne los permisos públicos en cualquier objeto cuando desee que el objeto esté disponible para todos los usuarios. No puede cambiar la pertenencia en public.

## SERVER LEVEL ROLES AND PERMISSIONS: 9 fixed server roles, 34 server permissions



## Roles fijos de base

- Estos roles existen en todas las bases de datos. A excepción del rol de base de datos **public**, no se pueden cambiar los permisos asignados a los roles fijos de base de datos.

Nombre del rol fijo de base de datos	Descripción
db_owner	Los miembros del rol fijo de base de datos db_owner pueden realizar todas las actividades de configuración y mantenimiento en la base de datos y también pueden quitar la base de datos en SQL Server. (En SQL Database y Almacenamiento de datos SQL, algunas actividades de mantenimiento requieren permisos a nivel de servidor y los roles db_ownersno las pueden realizar).
db_securityadmin	Los miembros del rol fijo de base de datos db_securityadmin pueden modificar la pertenencia a roles únicamente para roles personalizados y administrar permisos. Los miembros de este rol pueden elevar potencialmente sus privilegios y se deben supervisar sus acciones.
db_accessadmin	Los miembros del rol fijo de base de datos db_accessadmin pueden agregar o quitar el acceso a la base de datos para inicios de sesión de Windows, grupos de Windows e inicios de sesión de SQL Server .
db_backupoperator	Los miembros del rol fijo de base de datos db_backupoperator pueden crear copias de seguridad de la base de datos.
db_ddladmin	Los miembros del rol fijo de base de datos db_ddladmin pueden ejecutar cualquier comando del lenguaje de definición de datos (DDL) en una base de datos.
db_datawriter	Los miembros del rol fijo de base de datos db_datawriter pueden agregar, eliminar o cambiar datos en todas las tablas de usuario.
db_datareader	Los miembros del rol fijo de base de datos db_datareader pueden leer todos los datos de todas las tablas de usuario.
db_denydatawriter	Los miembros del rol fijo de base de datos db_denydatawriter no pueden agregar, modificar ni eliminar datos de tablas de usuario de una base de datos.
db_denydatareader	Los miembros del rol fijo de base de datos db_denydatareader no pueden leer datos de las tablas de usuario dentro de una base de datos.

## DATABASE LEVEL ROLES AND PERMISSIONS: 11 fixed database roles, 77 database permissions

### db\_owner fixed database role

CONTROL DATABASE: Has all permissions in the database

#### db\_datareader

GRANT SELECT ON DATABASE::

#### db\_denydatareader

DENY SELECT ON DATABASE::

#### db\_datawriter

GRANT INSERT ON DATABASE::

GRANT UPDATE ON DATABASE::

GRANT DELETE ON DATABASE::

#### db\_denydatawriter

DENY INSERT ON DATABASE::

DENY UPDATE ON DATABASE::

DENY DELETE ON DATABASE::

#### db\_accessadmin

CREATE SCHEMA

ALTER ANY USER

CONNECT

#### db\_securityadmin

ALTER ANY ROLE, CREATE ROLE

ALTER ANY APPLICATION ROLE

VIEW DEFINITION

#### public

There are no database-level permissions inherent in the public database role, however some database permissions are present by default. Specifically, VIEW ANY COLUMN MASTER KEY DEFINITION, VIEW ANY COLUMN ENCRYPTION KEY DEFINITION, and SELECT permission on many individual system tables. These permissions can be revoked.

#### db\_backupoperator

BACKUP DATABASE

BACKUP LOG

CHECKPOINT

#### db\_ddladmin

ALTER ANY ASSEMBLY  
ALTER ANY ASYMMETRIC KEY  
ALTER ANY CERTIFICATE  
ALTER ANY CONTRACT  
ALTER ANY DATABASE DDL TRIGGER  
ALTER ANY DATABASE EVENT NOTIFICATION  
ALTER ANY DATASPACE  
ALTER ANY FULLTEXT CATALOG  
ALTER ANY MESSAGE TYPE  
ALTER ANY REMOTE SERVICE BINDING  
ALTER ANY ROUTE  
ALTER ANY SCHEMA  
ALTER ANY SERVICE  
ALTER ANY SYMMETRIC KEY  
CHECKPOINT  
CREATE AGGREGATE  
CREATE DEFAULT  
CREATE FUNCTION  
CREATE PROCEDURE  
CREATE QUEUE  
CREATE RULE  
CREATE SYNONYM  
CREATE TABLE  
CREATE TYPE  
CREATE VIEW  
CREATE XML SCHEMA COLLECTION  
REFERENCES

There are various special purpose roles in the msdb database

The other database level permissions, are not granted to any fixed database role except db\_owner.

# Asignar permisos / remover permisos

```
grant SELECT on etl.dbo.etl (Id,Username) to prueba
```

```
USE AdventureWorks2017
```

```
go
```

```
grant SELECT ON OBJECT::Person.Address (AddressID, City) to prueba
```

```
grant SELECT ON OBJECT::Person.AddressType (Name) to prueba
```

Remover permisos

```
USE AdventureWorks2012;
```

```
REVOKE SELECT ON OBJECT::Person.Address FROM RosaQdM;
```

```
GO
```

# Ejecutar comando como otro usuario

```
EXECUTE AS USER = 'test';
```

```
SELECT * FROM TableTest
```

```
ORDER BY subentity_name, permission_name ;
```

```
REVERT;
```

```
GO
```

# Ejemplos de asignar permisos

Crear en AdventureWorks un usuario AsistenteRH en base a un login del mismo nombre que tenga permisos de lectura y escritura en el esquema Person. Denegar los permisos para insertar y eliminar.

- use master  
go  
create login AsistenteRH with password = '123'  
go  
use AdventureWorks  
go  
create user AsistenteRH from login AsistenteRH  
go  
— Permisos  
Grant Select, Update on Schema::Person to AsistenteRH  
Deny Insert, Delete on Schema::Person to AsistenteRH
- go

Crear un inicio para Northwind llamado Capataz en base al mismo login. Asignar permisos de Lectura, Inserción y Actualización en toda la BD. Denegar el permiso de Eliminación.

- use master  
go  
Create login Capataz with password = '123'  
go  
use Northwind  
go  
Create user Capataz from login Capataz  
go  
grant select, Insert, Update to Capataz  
Deny delete to Capataz  
go



# Ejemplos de asignar permisos

Crear un usuario Auditor, asignar permiso de lectura en la BD y hacerlo miembro de db\_backupoperator. Al login hacerlo miembro de [processadmin]. Denegar el permiso de insertar, eliminar y modificar.

- ```
use master
go
create login Auditor with password = '123'
go
Alter server role processadmin add member Auditor
go
use Northwind
go
create user Auditor from login Auditor
go
Grant Select to Auditor
Deny Insert, Delete, Update to Auditor
go
Alter role db_backupoperator add member Auditor
go
```

Crear un usuario AsistenteVentas usando un login con el mismo nombre en AdventureWorks que tengo permisos de Lectura, modificación, inserción en el esquema Sales (Ventas), Denegar Eliminación.

- ```
use master
go
create login AsistenteVentas with password = '123'
go
use AdventureWorks
go
Create user AsistenteVentas from login AsistenteVentas
go
Grant Select, Update, Insert on Schema::Sales to AsistenteVentas
Deny Delete on Schema::Sales to AsistenteVentas
go
```

# Ejemplos de asignar permisos

**Incluir para AsistenteVentas la lectura de la tabla [HumanResources].[Employee]**

- use AdventureWorks  
go  
Grant Select on  
Object::HumanResources.Employee to  
AsistenteVentas  
Deny Insert, Update, Delete on  
Object::HumanResources.Employee to  
AsistenteVentas  
go

**Crear un usuario llamado Contable que tenga acceso sólo a los esquemas Person y HumanResorces, asegurar que no pueda ver los otros esquemas y que en los esquemas que tiene permisos no pueda modificar ni eliminar los registros.**

- use master  
go  
Create login Contable with password = '123'  
go  
use AdventureWorks  
go  
Create user Contable from login Contable  
go  
Grant select, insert on schema::Person to Contable  
Grant select, insert on schema::HumanResources to Contable  
Deny update, delete on schema::HumanResources to Contable  
Deny update, delete on schema::Person to Contable  
Deny select, insert, update, delete on schema::Production to Contable  
Deny select, insert, update, delete on schema::Purchasing to Contable  
Deny select, insert, update, delete on schema::Sales to Contable  
Deny select, insert, update, delete on schema::dbo to Contable  
go

## Ejemplos de asignar permisos

Crear un SP que liste sólo Id, Descripción, Precio y Stock de Productos, luego crear un usuario Reportes que tenga permiso únicamente al SP creado.  
Denegar los permisos en la tabla productos de listado, inserción, modificación y eliminación.

Permiso en un SP es: EXECUTE

- Create procedure spProductosListado  
As  
SELECT ProductID, ProductName, UnitPrice,  
UnitsInStock from Products  
go
- use master  
go  
Create login Reportes with password = '123'  
go  
use Northwind  
create user Reportes from login Reportes  
go  
Deny Select, insert, Update, Delete to Reportes  
Grant Execute on object::dbo.spProductosListado to Reportes  
go

# Asignar permisos

```
CREATE DATABASE [NewDatabase]
GO
CREATE LOGIN [NewLogin] WITH PASSWORD=N'test', DEFAULT_DATABASE=[NewDatabase]
GO
ALTER SERVER ROLE [sysadmin] ADD MEMBER [NewLogin]
GO
USE [NewDatabase]
GO
CREATE USER [NewUser] FOR LOGIN [NewLogin]
GO
ALTER USER [NewUser] WITH DEFAULT_SCHEMA=[dbo]
GO
ALTER ROLE [db_owner] ADD MEMBER [NewUser]
GO

• -- Clean up
USE master
GO
DROP DATABASE [NewDatabase]
GO
DROP LOGIN [NewLogin]
GO
```

USE master

GO

CREATE LOGIN userprueba WITH PASSWORD=N'test',  
DEFAULT\_DATABASE=Northwind

GO

USE Northwind

GO

CREATE USER userprueba FOR LOGIN userprueba

GO

GRANT SELECT ON [dbo].[Region] to userprueba

GRANT SELECT ON [dbo].[Employees] ([Region]) TO userprueba

GO

- Valida

EXECUTE AS USER = 'userprueba';

SELECT \* FROM Employees

REVERT;

GO

EXECUTE AS USER = 'userprueba';

SELECT Region FROM Employees

REVERT;

GO

-- Clean up

USE Northwind

GO

DROP USER userprueba

go

USE master

GO

DROP LOGIN userprueba

GO

declare @nombreUsuario varchar(200)

declare @sql varchar(200)

set @nombreUsuario='UserPrueba'

USE Northwind

set @sql = 'DROP USER '+ @nombreUsuario

exec(@sql)

use master

set @sql = 'DROP LOGIN '+ @nombreUsuario

exec(@sql)



## Asignar permisos

```
USE master
```

```
go
```

```
create login pruebaETL WITH PASSWORD=N'test'
```

```
USE etl
```

```
go
```

```
create user pruebaETL from login pruebaETL
```

```
grant SELECT ON OBJECT::etl (username,location) to  
pruebaETL
```

```
go
```

```
EXECUTE AS USER = 'pruebaETL';
```

```
SELECT username,location FROM etl;
```

```
REVERT;
```

```
GO
```

- <https://dev.mysql.com/doc/refman/5.7/en/account-management-statements.html>
- <https://rm-rf.es/usuario-mysql-como-crear-borrar-y-asignar-privilegios/>
- <https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/sql/overview-of-sql-server-security>

## Referencias