ADMINISTRACIÓN DE SISTEMES GESTORES DE BASES DE DATOS
UNIDAD 7: USUARIOS Y PERMISOS

Curso 2019 - 2020

I.E.S. Marcos Zaragoza

2º CFGS ASIX

Alberto Alemany

LENGUAJE DE CONTROL DE DATOS (DCL)

- El Data Control Language es un lenguaje proporcionado por el Sistema de Gestión de Base de Datos que incluye una serie de comandos SQL que permiten al administrador controlar el acceso a los datos contenidos en la Base de Datos.
- Los comandos principales del DCL son:
  - GRANT: Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
  - REVOKE: Permite eliminar permisos que previamente se han concedido con GRANT.
- Las tareas sobre las que se pueden conceder o denegar permisos son las siguientes:
  - CONNECT, SELECT, INSERT, UPDATE, DELETE, USAGE

ADMINISTRACIÓN DE BD EN SQL SERVER

LOGINS VS USERS

- Un LOGIN es un uso identificado que permite conectarse a una instancia de SQL Server.
- Un USER permite logearnos a una BD SQL Server y es mapeado con un Login.

  - *Por lo tanto, primero requeriremos de crear un login, para poder crear los usuarios.*

CREATE LOGIN

- Hay cuatro tipos de LOGIN que se pueden crear en SQL Server:
  - Login con la autenticación de Windows.

```
CREATE LOGIN [domain_name\login_name]
FROM WINDOWS
[ WITH DEFAULT_DATABASE = database_name
| DEFAULT_LANGUAGE = language_name ]
```

  - *Ejemplo:*

```
CREATE LOGIN [test_domain\asixmz]
FROM WINDOWS
```

CREATE LOGIN

- Hay cuatro tipos de LOGIN que se pueden crear en SQL Server:
  - Login con la autenticación de SQL Server.

```
CREATE LOGIN login_name
WITH PASSWORD = { 'password' | hashed_password HASHED } [
MUST_CHANGE ]
[ , SID = sid_value
  | DEFAULT_DATABASE = database_name
  | DEFAULT_LANGUAGE = language_name
  | CHECK_EXPIRATION = { ON | OFF }
  | CHECK_POLICY = { ON | OFF }
  | CREDENTIAL = credential_name ]
```

  - *Ejemplo:*

```
CREATE LOGIN asixgbd
WITH PASSWORD = 'pwd123'
```

CREATE LOGIN

- Hay cuatro tipos de LOGIN que se pueden crear en SQL Server:
  - Login desde un certificado.

```
CREATE LOGIN login_name
FROM CERTIFICATE certificate_name;
```

    - *Ejemplo:*

```
CREATE LOGIN asixgbd
FROM CERTIFICATE certificate1;
```

  - Login desde una clave asimétrica

```
CREATE LOGIN login_name
FROM ASYMMETRIC KEY asym_key_name;
```

    - *Ejemplo:*

```
CREATE LOGIN asixgbd
FROM ASYMMETRIC KEY asym_key1;
```

ALTER LOGIN

- Modifica una identidad utilizada para conectarse a una instancia de SQL Server. Esta instrucción se usa para:
  - Cambiar una contraseña, forzar un cambio de contraseña, deshabilitar un inicio de sesión, habilitar un inicio de sesión, desbloquear un inicio de sesión, cambiar el nombre de un inicio de sesión, etc.

```
ALTER LOGIN login_name { ENABLE | DISABLE
| WITH PASSWORD = 'password' | hashed_password HASHED
         [ OLD_PASSWORD = 'old_password' ] | MUST_CHANGE | UNLOCK
       | DEFAULT_DATABASE = database_name
       | DEFAULT_LANGUAGE = language_name | NAME = new_login_name
       | CHECK_EXPIRATION = { ON | OFF } | CHECK_POLICY = { ON | OFF }
       | CREDENTIAL = credential_name | NO CREDENTIAL
| ADD CREDENTIAL new_credential_name | DROP CREDENTIAL credential_name };
```

ALTER LOGIN

- *Ejemplo cambio contraseña:*

```
ALTER LOGIN asixgbd
WITH PASSWORD = ‘asd43210'
```

- *Ejemplo deshabilitar / habilitar Login:*

```
ALTER LOGIN asixgbd DISABLE
```

```
ALTER LOGIN asixgbd ENABLE
```

- *Ejemplo renombrar Login:*

```
ALTER LOGIN asixgbd
WITH NAME = marcoszaragozagbd
```

DROP LOGIN

- Permite borrar un login

```
DROP LOGIN login_name
```

  - *Ejemplo:*

```
DROP LOGIN asixgbd
```

FIND LOGINS

- Permite obtener todos los Logins existentes en la BD:

```
SELECT * FROM master.sys.sql_logins
```

CREATE USER

- Crea un usuario en la BD que será mapeado con un Login ya existente

```
CREATE USER user_name FOR LOGIN login_name
```

  - *Ejemplo:*

```
CREATE USER usuarioasix FOR LOGIN asixgbd
```

DROP USER

- Borra un usuario ya existente

```
DROP USER user_name
```

  - *Ejemplo:*

```
DROP USER usuarioasix
```

FIND USERS

- Permite obtener todos los usuarios existentes en la BD

```
SELECT * FROM master.sys.database_principals
```

PRIVILEGIOS SQL SERVER

GESTIÓN DE PERMISOS/PRIVILEGIOS

- Se emplea para asignar diferentes niveles de permisos de acceso y/o manipulación sobre los objetos de la base de datos a los distintos usuarios de la misma.
- Las operaciones sobre las que se puede realizar son: INSERT, UPDATE, DELETE, SELECT, EXECUTE, CONNECT, USAGE, CREATE, DROP, etc.
- Sus sentencias más importantes son GRANT y REVOKE, para conceder o quitar permisos

GRANT

- Concede privilegios a objetos de la BD para determinados usuarios

```
GRANT privileges ON object TO user
```

- *Ejemplos:*
  - Conceder el permiso EXECUTE para un procedimiento almacenado

```
GRANT EXECUTE ON OBJECT::HumanResources.updateEmployeeHireInfo
    TO Recruiting11
```

GRANT SOBRE TABLA

- Los privilegios que se podrían otorgar sobre una tabla son:
  - SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, ALL
- *Ejemplos:*

```
GRANT SELECT, INSERT, UPDATE, DELETE ON pacientes TO usuarioasix
```

```
GRANT ALL ON pacientes TO usuarioasix
```

- *También podríamos asignar un privilegio determinado a un rol, como en el caso siguiente:*

```
GRANT SELECT ON pacientes TO public
```

REVOKE

- Quita privilegios a objetos de la BD para determinados usuarios

```
REVOKE privileges ON object TO user
```

- *Ejemplos:*
  - Quitar el permiso EXECUTE para un procedimiento almacenado

```
REVOKE EXECUTE ON OBJECT::HumanResources.updateEmployeeHireInfo
    TO Recruiting11
```

REVOKE SOBRE TABLA

- Los privilegios que se podrían eliminar sobre una tabla son:
  - SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, ALL
- *Ejemplos:*

```
REVOKE DELETE ON pacientes TO usuarioasix
```

```
REVOKE ALL ON pacientes TO usuarioasix
```

- *También podríamos eliminar un privilegio determinado de un rol:*

```
REVOKE SELECT ON pacientes TO public
```

VISTAS

VISTAS

- Una VISTA, en esencia, es una tabla virtual que no existe físicamente en SQL Server. Más bien, es creada por una consulta que une una o más tablas y permite que ciertos usuarios sólo puedan acceder a ciertos datos.
- Las vistas proporcionan a un usuario un modelo personalizado de la base de datos. Una vista puede ocultar los datos que un usuario no necesita ver. La capacidad de las vistas para ocultar datos sirve para simplificar el uso del sistema y para mejorar la seguridad. Así, aunque puede ser que al usuario se le niegue el acceso directo a una relación, puede que se le permita el acceso a parte de esa relación mediante una vista.

VISTAS

- Un ejemplo de vista podría ser el caso en el que nuestro trabajador de oficina del departamento de morosos sólo tenga acceso a la información de los clientes morosos. Por lo tanto, se crearía una vista en la cual se indicará que el monto de la deuda fuera superior a 5000€. Así se evitaría que tuvieran los de este departamento acceso a la información de todos los clientes, cuando no lo necesitan
- Los tres tipos de operaciones que se pueden realizar con las vistas son:

Creación de VISTA (CREATE VIEW)

Modificación de VISTA (UPDATE VIEW)

Borrado de VISTA (DELETE VIEW)

CREATE VIEW

- Permite crear una vista

```
CREATE VIEW [schema_name.]view_name AS
  [ WITH { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
  SELECT expressions
  FROM tables
  [WHERE conditions]
```

- Ejemplos:

```
CREATE VIEW prod_inv AS
  SELECT products.product_id, products.product_name, inventory.quantity
  FROM products
  INNER JOIN inventory
  ON products.product_id = inventory.product_id
  WHERE products.product_id >= 1000
```

CREATE VIEW

- Ejemplos:

```
CREATE VIEW morosos AS
   SELECT cli.nombre, cli.direccion, cli.tel, sa.monto
   FROM clientes cli
   INNER JOIN saldos sa
   ON cli.clave = sa.cvecli
   WHERE sa.monto > 5000
   ORDER BY sa.monto
```

Vista para que el departamento de morosos sólo pueda ver los clientes que deben más de 5000 euros

- A partir de aquí simplemente habría que hacer las selects sobre las vistas

```
SELECT * FROM prod_inv
```

```
SELECT * FROM morosos
```

UPDATE VIEW

- Permite modificar una vista

```
ALTER VIEW [schema_name.]view_name AS
  [ WITH { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
  SELECT expressions
  FROM tables
  WHERE conditions
```

- Ejemplos:

```
ALTER VIEW prod_inv AS
  SELECT products.product_name, inventory.quantity
  FROM products
  INNER JOIN inventory
  ON products.product_id = inventory.product_id
  WHERE products.product_id >= 500
  AND products.product_id <= 1000
```

DROP VIEW

- Permite eliminar una vista

```
DROP VIEW view_name
```

- Ejemplos:

```
DROP VIEW prod_inv
```

```
DROP VIEW morosos
```