Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Manejo de permisos a nivel de usuario de base de datos

Integrantes: Miño Micaela
Gonzalez Rodrigo Alejandro
Maciel Lautaro
Lezana Mauricio Sebastian

Profesor a cargo: Walter Vallejos
Materia: Bases de datos I
Año: 2023

Índice

CAPÍTULO I: INTRODUCCIÓN

El tema de este Trabajo Práctico es el “Manejo de permisos a nivel de usuario en bases de datos”. Este tema se centra en la investigación del problema de cómo se manejan los permisos a nivel de usuario en una base de datos, un aspecto crucial en la administración de bases de datos.

Un "rol" es un conjunto de derechos predefinidos que se asignan a un grupo de usuarios en una base de datos.Siendo útil para simplificar la gestión de permisos y accesos en lugar de otorgar permisos individuales a cada usuario en la base de datos.

b. Definición o planteamiento del problema

El problema que se investiga es cómo se pueden gestionar eficazmente los permisos a nivel de usuario en una base de datos. Este es un aspecto importante para garantizar la seguridad y la integridad de los datos. La definición del problema puede formularse como la siguiente pregunta: ¿Cómo se pueden asignar y gestionar los permisos a nivel de usuario en una base de datos para garantizar la seguridad y la integridad de los datos?

c. Objetivo del Trabajo Práctico

El objetivo general del Trabajo Práctico es entender y aplicar los conceptos teóricos relacionados con el manejo de permisos a nivel de usuario en las bases de datos.

i.Objetivos Generales

El objetivo general es producir conocimientos sobre el manejo de permisos a nivel de usuario en las bases de datos y cómo estos conceptos se pueden aplicar en un caso práctico.

ii. Objetivos Específicos

Los objetivos específicos incluyen entender los diferentes roles en una base de datos, cómo se pueden asignar y gestionar los permisos a nivel de usuario, y cómo estos conceptos se aplican en un caso práctico.

CAPÍTULO II: MARCO CONCEPTUAL O REFERENCIAL

Este capítulo proporciona los fundamentos y especificaciones que permitirán comprender mejor el problema planteado.

1. Definición de roles de usuario en bases de datos: Los roles de usuario en las bases de datos son entidades de seguridad que agrupan a otras entidades de seguridad, similares a los grupos del sistema operativo Microsoft Windows[1]. Un rol es un objeto de base de datos que agrupa uno o más privilegios y puede ser asignado a un usuario[2]. Los roles de nivel de base de datos se aplican a toda la base de datos en lo que respecta a su ámbito de permisos. [1]

2. Tipos de roles: Existen dos tipos de roles en el nivel de base de datos: los roles fijos de base de datos que están predefinidos en la base de datos y los roles de base de datos definidos por el usuario que el usuario puede crear.

CAPÍTULO III: METODOLOGÍA SEGUIDA

a. Descripción de cómo se realizó el Trabajo Práctico

Para realizar este trabajo, se siguieron varios pasos. Primero, se realizó una revisión exhaustiva de la literatura disponible sobre el manejo de permisos a nivel de usuario en las bases de datos. Se consultaron varias fuentes, incluyendo libros de texto, artículos de investigación y recursos en línea.

Una vez que se adquirió un entendimiento sólido del tema, se procedió a aplicar estos conceptos teóricos a un caso práctico. Se exploraron diferentes roles en una base de datos y cómo se pueden asignar y gestionar los permisos a nivel de usuario.

Durante todo el proceso, se encontraron varias dificultades. Por ejemplo, la comprensión de los conceptos teóricos requirió un estudio detallado y la aplicación de estos conceptos a un caso práctico presentó sus propios desafíos. Sin embargo, estas dificultades se superaron con perseverancia y dedicación.

b. Herramientas (Instrumentos y procedimientos)

Para la recolección y tratamiento de la información, se utilizaron varias herramientas. La principal fuente de información fue la revisión bibliográfica, que incluyó libros de texto y artículos de investigación. También se utilizaron recursos en línea para obtener información más actualizada.

Además, se utilizó el software de gestión de bases de datos SQL Server para explorar prácticamente cómo se pueden asignar y gestionar los permisos a nivel de usuario. Este software proporcionó una plataforma para aplicar los conceptos teóricos aprendidos durante la revisión bibliográfica.

CAPÍTULO IV: DESARROLLO DEL TEMA / PRESENTACIÓN DE RESULTADOS

En este capítulo, presentamos los hallazgos del estudio.Los roles a nivel de base de datos se aplican a toda la base de datos en lo que respecta a su ámbito de permisos.

Además de los inicios de sesión y usuarios se proporciona un conjunto de roles predefinidos tanto a nivel de servidor como a nivel de base de datos. Estos roles predefinidos contienen conjuntos de permisos comunes que se pueden asignar a los usuarios según sus necesidades.
en sql son los siguientes

Roles fijos de base de datos

en sql server

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 `drop` la base de datos en SQL Server. (En SQL Database y Azure Synapse, algunas actividades de mantenimiento requieren permisos a nivel de servidor y no se pueden realizar por db_owners).
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 eliminar 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. Los miembros de este rol pueden potencialmente aumentar sus privilegios manipulando código que puede ser ejecutado bajo altos privilegios y sus acciones se deben supervisar.
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. En la mayoría de los casos de uso, este rol se combinará con la membresía db_datareader para permitir la lectura de los datos que se van a modificar.
db_datareader	Los miembros del rol fijo de base de datos db_datareader pueden leer todos los datos de todas las tablas y vistas de usuario. Los objetos de usuario pueden existir en cualquier esquema, excepto *sys* e *INFORMATION_SCHEMA*.
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 y vistas de usuario dentro de una base de datos.

[1]

Roles especiales para SQL Database

Los roles de base de datos solo existen en la base de datos "master" que es una base de datos especial en SQL Server. Los permisos de estos roles están limitados a

acciones que se realizan en la base de datos "master". Solo los usuarios de esta base de datos "master" pueden ser miembros de estos roles. Los inicios de sesión (usuarios con acceso al servidor) no pueden ser miembros de estos roles directamente pero se pueden crear usuarios basados en inicios de sesión y luego agregar esos usuarios a estos roles.

Estos roles y permisos se aplican principalmente a la base de datos "master" y suelen estar relacionados con la administración y configuración del servidor.

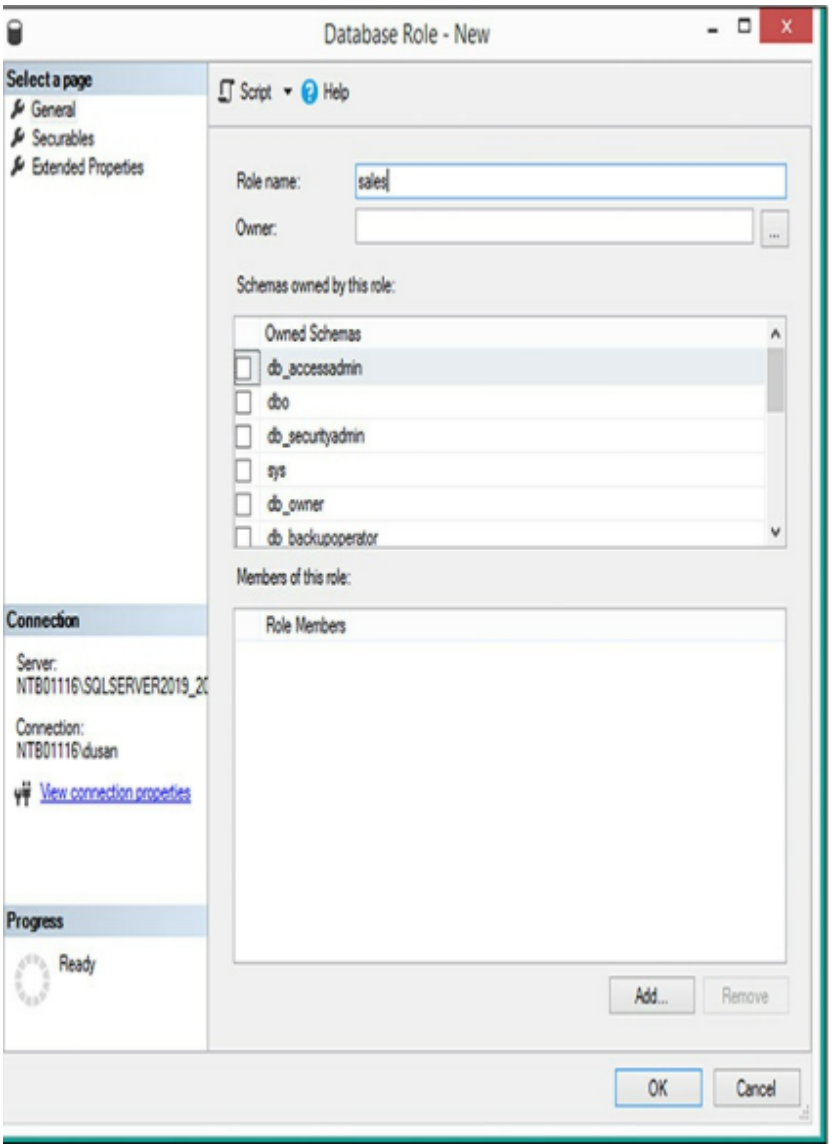
1. **dbmanager**: Puede crear y eliminar bases de datos, convirtiéndose en el propietario. Tiene todos los permisos en las bases de datos que crea, pero no en otras.
2. **db_exporter**: Aplicable a grupos de SQL dedicados de Azure Synapse Analytics. Permite actividades de exportación de datos con permisos como CREATE TABLE, ALTER ANY SCHEMA, ALTER ANY EXTERNAL DATA SOURCE, y ALTER ANY EXTERNAL FILE FORMAT.
3. **loginmanager**: Puede crear y eliminar inicios de sesión en la base de datos "master" virtual.

Roles de Base de Datos Definidos por el Usuario (User-Defined Database Roles)

Por lo general, los roles de base de datos definidos por el usuario se aplican cuando un grupo de usuarios de la base de datos necesita llevar a cabo un conjunto común de actividades dentro de una base de datos y no existe un grupo de Windows aplicable. Estos roles se crean y eliminan utilizando tanto el Management Studio como las declaraciones Transact-SQL **CREATE ROLE**, **ALTER ROLE** y **DROP ROLE**.

Gestionando roles usuarios definidos por la base de datos usando SSMS

Para crear un rol de usuario definido usando Management Studio,seleccione el servidor,seleccione la base de datos y su carpeta de Seguridad. Click derecho a rols,Click nuevo, y luego click en nuevo rol de base de datos. En el cuadro de diálogo Rol de base de datos , ingrese el nombre del nuevo rol. Haga clic en Agregar para agregar miembros al nuevo rol. Elija los miembros (usuarios y/o otros roles) del nuevo rol y haga clic en Aceptar.



[3]

Gestionando Roles de Base de Datos Definidos por el Usuario Usando T-SQL

La declaración **CREATE ROLE** crea un nuevo rol de base de datos definido por el usuario en la base de datos actual. La sintaxis de esta declaración es la siguiente:

```
CREATE ROLE role_name [AUTHORIZATION owner_name]
```

role_name es el nombre del rol definido por el usuario que se va a crear. **owner_name** especifica el usuario o rol de la base de datos que será el propietario del nuevo rol. (Si

no se especifica ningún usuario, el rol será propiedad del usuario que ejecuta la declaración **CREATE ROLE**).

La declaración **ALTER ROLE** cambia el nombre de un rol de base de datos definido por el usuario. La declaración **DROP ROLE** elimina un rol de la base de datos. Los roles que son propietarios de objetos de base de datos (objetos securables) no pueden eliminarse de la base de datos. Para eliminar dicho rol, primero debes transferir la propiedad de esos objetos.

El ejemplo muestra cómo puedes crear y agregar miembros a un rol definido por el usuario.

```
USE sample;
CREATE ROLE marketing AUTHORIZATION peter;
GO
ALTER ROLE marketing ADD MEMBER peter;
ALTER ROLE marketing ADD MEMBER mary;
```

Autorización

Solo los usuarios autorizados pueden efectuar declaraciones o realizar operaciones en una entidad. Si un usuario no autorizado intenta realizar cualquiera de estas tareas, la ejecución de la declaración Transact-SQL o la operación en el objeto de la base de datos será rechazada.

Existen tres declaraciones Transact-SQL relacionadas con la autorización:

GRANT

La declaración **GRANT** otorga permisos a objetos securizables. La sintaxis de la declaración **GRANT** es la siguiente:

```
GRANT {ALL [PRIVILEGES]} | permission_list
     [ON [class::] securable]  TO principal_list [WITH GRANT OPTION]
     [AS principal ]
```

La cláusula **ALL** es una característica obsoleta en las últimas versiones y se mantiene solo por compatibilidad con versiones anteriores. No otorga todos los permisos

posibles, como su nombre sugiere. (Para ver la lista de permisos específicos, consulta Microsoft Docs.)

permission_list especifica declaraciones u objetos (separados por comas) para los cuales se otorgan los permisos. class especifica una clase de objeto securizable o el nombre de un objeto securizable para el cual se otorgan permisos. ON securable especifica el objeto securizable para el cual se otorgan permisos.

principal_list enumera todas las cuentas (separadas por comas) a las cuales se otorgan permisos. principal y los componentes de **principal_list** pueden ser una cuenta de usuario, un inicio de sesión o una cuenta de usuario asignada a un certificado, un inicio de sesión asignado a una clave asimétrica, un usuario de base de datos, un rol de base de datos o un rol de aplicación.

```
USE sample;
GRANT CREATE TABLE, CREATE PROCEDURE
                    TO peter, mary;
```

[3]

DENY

La declaración **DENY** impide que los usuarios realicen acciones. Esto significa que la declaración elimina permisos existentes de las cuentas de usuario o evita que los usuarios obtengan permisos a través de su pertenencia a un grupo/rol que podría concederse en el futuro. Esta declaración tiene la siguiente sintaxis:

```
DENY {ALL [PRIVILEGES] } | permission_list
 [ON [class::] securable]  TO principal_list
[CASCADE]  [ AS principal ]
```

Todas las opciones de la declaración **DENY** tienen el mismo significado lógico que las opciones con el mismo nombre en la declaración **GRANT**. **DENY** tiene una opción adicional, **CASCADE**, que especifica que se denegarán permisos al usuario A y a cualquier otro usuario a los que el usuario A haya transmitido este permiso. (Si no se especifica la opción **CASCADE** en la declaración **DENY** y el permiso correspondiente del objeto se otorgó con la opción **WITH GRANT**, se generará un error).

La declaración **DENY** evita que el usuario, grupo o rol acceda a los permisos otorgados a través de su pertenencia a un grupo o rol. Esto significa que si un usuario pertenece a un grupo (o rol) y el permiso otorgado al grupo se le deniega al usuario, este usuario

será el único del grupo que no podrá usar este permiso. Por otro lado, si se deniega un permiso para todo un grupo, se denegará el permiso a todos los miembros del grupo.

```
USE sample;
DENY CREATE TABLE, CREATE PROCEDURE
   TO peter;
```

REVOKE

La declaración **REVOKE** elimina uno o varios permisos previamente otorgados o denegados. Esta declaración tiene la siguiente sintaxis:

```
REVOKE [GRANT OPTION FOR]
    { [ALL [PRIVILEGES] ]  | permission_list ]}
 [ON  [class:: ] securable ]
 FROM principal_list [CASCADE]  [ AS principal ]
```

La única opción nueva en la declaración **REVOKE** es **GRANT OPTION FOR**. (Todas las demás opciones tienen el mismo significado lógico que las opciones con los mismos nombres en las declaraciones **GRANT** o **DENY**). **GRANT OPTION FOR** se utiliza para eliminar los efectos de la opción **WITH GRANT OPTION** en la declaración GRANT correspondiente. Esto significa que el usuario seguirá teniendo los permisos otorgados previamente, pero ya no podrá otorgar el permiso a otros usuarios.

```
USE sample;
REVOKE SELECT ON project FROM PUBLIC;
```

Ejemplos

Para trabajar con los roles a nivel de base de datos

Para agregar y quitar usuarios en un rol de base de datos, se utilizan las opciones ADD MEMBER y DROP MEMBER.

por ejemplo:

```
CREATE LOGIN Ben WITH PASSWORD = 'Password123';--creamos un login
CREATE LOGIN Artur WITH PASSWORD=’Password3120’
CREATE USER Ben FOR LOGIN Ben;--asignamos un usuario
CREATE USER Artur FOR LOGIN Artur;

ALTER ROLE db_datareader
ADD MEMBER Ben;  --asignamos el rol de usuario

ALTER ROLE db_ddladmin
ADD MEMBER Artur

--creamos el procedimiento almacenado insertarAdministrador
create procedure insertarAdministrador
@apeynom varchar(50),
@viveahi varchar(1),
@tel varchar(20),
@s varchar(1),
@nacimiento datetime
as
begin
        insert into administrador(apeynom,viveahi,tel,sexo,fechnac)
        values (@apeynom,@viveahi,@tel,@s,@nacimiento);
end



GRANT EXECUTE ON InsertarEnAdministrador TO Ben;
--asignamos a Ben la posibilidad de ejecutar el procedimiento almacenado –InsertarEnAdministrador
REVOKE EXECUTE ON InsertarEnAdministrador FROM Ben
--sacamos el permiso a Ben
```

```
ALTER ROLE db_datareader
DROP MEMBER Ben; -- sacamos el permiso al usuario
GO
```

Ejemplo de Vistas

```
-- Crear un nuevo esquema que solo contendrá las vistas
CREATE SCHEMA ViewSchema AUTHORIZATION dbo;
GO
-- Crear una nueva vista en este esquema siendo  Mytable la vista que queremos que posea el usuario
CREATE VIEW ViewSchema.MyView AS SELECT * FROM MyTable;
GO
-- Crear un nuevo rol de base de datos
CREATE ROLE db_viewreader;

-- Otorgar permisos SELECT a todos los objetos en el esquema ViewSchema al nuevo –rol de base de datos
GRANT SELECT ON SCHEMA::ViewSchema TO db_viewreader;


-- Crear un nuevo usuario
CREATE USER userWITHOUT LOGIN;

-- Agregar el nuevo usuario al rol de base de datos
ALTER ROLE db_viewreader ADD MEMBER lucas;
```

las vistas se pueden asignar de forma individual o combinada a continuacion un ejemplo de cada una tomando de base la tabla base_consorcio

Individual:

```
CREATE VIEW ViewSchema.ProvinciaView AS SELECT * FROM provincia;

GO

CREATE VIEW ViewSchema.LocalidadView AS SELECT * FROM localidad;

GO

CREATE VIEW ViewSchema.ZonaView AS SELECT * FROM zona;

GO
```

```
CREATE VIEW ViewSchema.ConsorcioView AS SELECT * FROM consorcio;
GO
CREATE VIEW ViewSchema.GastoView AS SELECT * FROM gasto;
GO
CREATE VIEW ViewSchema.ConserjeView AS SELECT * FROM conserje;
GO
CREATE VIEW ViewSchema.AdministradorView AS SELECT * FROM administrador;
GO
CREATE VIEW ViewSchema.TipoGastoView AS SELECT * FROM tipogasto;
GO
```

agrupando las vistas:

```
CREATE VIEW ViewSchema.MyCombinedView AS
SELECT
   p.*,   l.*,   co.*,   g.* FROM   provincia p
JOIN
   localidad l ON p.idprovincia = l.idprovincia
JOIN
   consorcio co ON l.idprovincia = co.idprovincia AND l.idlocalidad = co.idlocalidad
JOIN
     gasto g ON co.idprovincia = g.idprovincia AND co.idlocalidad = g.idlocalidad AND co.idconsorcio = g.idconsorcio;
```

Para mostrarlo se utiliza lo siguiente

```
EXECUTE AS USER = 'user';


-- Consultar la vista MyCombinedView

SELECT * FROM ViewSchema.MytablaIndividual;
-- Consultar la vista individual

SELECT * FROM ViewSchema.MyCombinedView;


REVERT;
```

CAPÍTULO V: CONCLUSIONES

El manejo efectivo del control de acceso es fundamental para la seguridad y la integridad del sistema. Comprender cómo funcionan los permisos a nivel de usuario en una base de datos es esencial para cualquier administrador o desarrollador que trabaje con bases de datos. Creemos que los objetivos del Trabajo Práctico fueron alcanzados.

Estas conclusiones no son arbitrarias, sino que están basadas en la información recogida durante la realización del trabajo.

CAPÍTULO VI: Referencias

[1]Microsoft,https://learn.microsoft.com/es-es/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver16 .
[2]IBM,bm.com/docs/en/db2woc?topic=SS6NHC/com.ibm.swg.im.dashdb.security.doc/doc/udef_user_roles.html
[3]Dušan Petković,"Microsoft SQL Server 2019 A Begginer's Guide",7a edicion,2019.

Capitulo VII:temas integrados

Procedimientos Almacenados

Un procedimiento almacenado de SQL Server es un conjunto de instrucciones Transact-SQL a las que se les da un nombre, que se almacena en el servidor. Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.
- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos

Transacciones

Una transacción se define como una secuencia de operaciones que se ejecutan de manera indivisible, es decir, todas o ninguna. Estas operaciones pueden ser consultas, inserciones, actualizaciones o eliminaciones de datos. El objetivo principal de las transacciones es garantizar la integridad y la consistencia de los datos, asegurando que se realicen de manera correcta y completa, sin dejar ningún estado intermedio inconsistente.

Triggers

Consiste en una serie de reglas predefinidas que se asocian a una tabla. Estas reglas se aplican a la base de datos cuando se realizan determinadas operaciones en la tabla, por ejemplo, al añadir, actualizar o eliminar registros. Dicho de otra manera, el trigger desencadena determinadas acciones de forma automática en las tablas de la base de datos cuando se insertan, modifican y se añaden nuevos datos. Generalmente es muy utilizado para auditorias en empresas Por otra parte, entre sus principales ventajas es que todas estas funciones se pueden realizar desde la propia base de datos, es decir, no es necesario recurrir a lenguajes externos de programación.

Back up y restore

La importancia de las copias de seguridad (backup) y la capacidad de restauración (restore) son los temas que se van abordar en este documento.

La copia de seguridad y la capacidad de restauración proporcionan una protección fundamental sobre la información de los usuarios, ya sea a nivel personal o empresarial de sus bases de datos. Para minimizar el riesgo de una pérdida de datos catastrófica, se debe realizar de forma periódica copias de seguridad de las bases de datos para conservar las modificaciones realizadas en los mismos. Una estrategia bien diseñada de copia de seguridad y restauración ayuda a proteger las bases de datos frente a la pérdida de datos provocada por diversos errores.