Manejo de permisos a nivel de usuarios de base de datos.

El manejo de permisos y roles en bases de datos es fundamental para asegurar que los usuarios tengan el acceso adecuado según sus necesidades y responsabilidades. Esta práctica permite configurar permisos de lectura, escritura y ejecución, protegiendo así la integridad de los datos y controlando el acceso a la información. En esta actividad, exploraremos cómo asignar permisos tanto a nivel de usuarios como de roles, observando el impacto de estos permisos en el acceso a datos y la ejecución de comandos.

Tareas

1. **Configurar la autenticación** en modo mixto (Windows y usuarios de base de datos).
2. **Permisos a nivel de usuarios:** Crear usuarios con permisos diferenciados (administrador y solo lectura), verificando el acceso de los usuarios en consecuencia.
3. **Permisos a nivel de roles:** Crear y asignar roles específicos, verificando el acceso de los usuarios en consecuencia.

1 Configurar la autenticación en modo mixto (Windows y usuarios de base de datos).

Configurar la autenticación en modo mixto en SQL Server permite el uso tanto de la autenticación de Windows como de la autenticación de SQL Server. Esto es útil en entornos donde se requiere una mezcla de seguridad integrada (como la autenticación de dominio) y usuarios personalizados de base de datos. Aquí te explico cómo configurarlo:

SQL Server ofrece dos modos de autenticación:

- **Autenticación de Windows**: Utiliza las credenciales de Windows, lo cual es más seguro ya que delega la autenticación en el sistema operativo. La **autenticación de Windows en SQL Server generalmente requiere que el cliente y el servidor estén en la misma red o que exista una conexión segura que permita la validación de las credenciales de dominio** (por ejemplo, mediante una VPN o algún túnel seguro que simule la red local).
- **Autenticación de SQL Server**: Usa credenciales de usuario y contraseña almacenadas dentro de SQL Server, sin dependencia de un sistema de dominio. Esto permite a los usuarios conectarse desde cualquier ubicación sin la necesidad de pertenecer al dominio, siempre y cuando tengan las credenciales de SQL Server.

Habilitar o Deshabilitar el Modo Mixto con T-SQL

Mediante T-SQL, sigue estos pasos:

1. Conéctate a SQL Server como usuario con privilegios de **sysadmin**.
2. Ejecuta uno de los siguientes comandos según el modo de autenticación que desees:

   - **Para habilitar el modo mixto (Autenticación de SQL Server y Windows)**:

```
EXEC xp_instance_regwrite
    N'HKEY_LOCAL_MACHINE',
    N'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer',
    N'LoginMode',
    REG_DWORD,
    2;
```

   - **Para habilitar solo la autenticación de Windows**:

```
EXEC xp_instance_regwrite
    N'HKEY_LOCAL_MACHINE',
    N'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer',
    N'LoginMode',
    REG_DWORD,
    1;
```

3. **Reinicia el servicio de SQL Server** para aplicar los cambios.
4. Verificar el Modo de Autenticación Actual

```
DECLARE @LoginMode INT;
EXEC xp_instance_regread
    N'HKEY_LOCAL_MACHINE',
    N'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer',
    N'LoginMode',
    @LoginMode OUTPUT;

SELECT CASE
         WHEN @LoginMode = 1 THEN 'Autenticación de Windows'
         WHEN @LoginMode = 2 THEN 'Autenticación Mixta'
         ELSE 'Desconocido'
       END AS ModoAutenticacion;
```

2 Permisos a nivel de usuarios:

Crear dos usuarios de base de datos.
Se crean dos usuarios, uno con privilegios de administrador y otro con permisos limitados, para observar cómo se comportan en función de los permisos asignados.

1. Crea el login a nivel de servidor

```
CREATE LOGIN UsuarioAdmin WITH PASSWORD = 'Admin12!'
```

2. Crea el usuario en la base de datos seleccionada (EN ESTE CASO LA DEL PROYECTO CON LA SENTENCIA USE [BD])

```
CREATE USER UsuarioAdmin FOR LOGIN UsuarioAdmin;
```

Un LOGIN puede estas asociado a mas de un USER en distintas BD
Login sin Usuario: El login existe y puede autenticarse en el servidor, pero no puede acceder a bases de datos hasta que se le asocie un usuario.

Usuario sin Login: El usuario se crea en la base de datos, pero no podrá autenticarse ni realizar ninguna acción, ya que no hay un login asociado que permita el acceso al servidor

-- Requisitos de Complejidad de Contraseña
-- Longitud Mínima: La contraseña debe tener al menos 8 caracteres de longitud.
--
-- Conjuntos de Caracteres: La contraseña debe incluir caracteres de al menos tres de los siguientes cuatro conjuntos:
--
-- Letras Mayúsculas: (A-Z)
-- Letras Minúsculas: (a-z)
-- Dígitos: (0-9)
-- Símbolos: (por ejemplo, !@#$%^&*()-_+=<>?)
-- Ejemplo de Contraseñas Válidas
-- Password1! (Mayúsculas, minúsculas, dígitos y símbolo)
-- MySecurePassword123! (Mayúsculas, minúsculas y dígitos)

3. Crea el segundo usuario

```
CREATE LOGIN UsuarioLectura WITH PASSWORD = 'lectura123!';
CREATE USER UsuarioLectura FOR LOGIN UsuarioLectura;
```

4. A un usuario darle el permiso de administrador y al otro usuario solo permiso de lectura.
Se asignan roles y permisos adecuados a cada usuario, asegurando que el administrador tenga acceso total y el usuario de lectura solo pueda consultar.

```
EXEC sp_addrolemember 'db_owner', 'UsuarioAdmin'; -- Permiso total
EXEC sp_addrolemember 'db_datareader', 'UsuarioLectura'; -- Solo lectura
```

El comando EXEC sp_addrolemember 'db_owner', 'UsuarioAdmin'; se utiliza en SQL Server para asignar un usuario específico (en este caso, UsuarioAdmin) a un rol de base de datos determinado (db_owner).

Detalles del Comando

sp_addrolemember: Es un procedimiento almacenado del sistema que permite agregar un usuario o grupo a un rol específico en una base de datos.

'db_owner': Este es un rol predefinido en SQL Server que otorga permisos completos sobre la base de datos. Los miembros de este rol pueden realizar cualquier acción dentro de la base de datos, como crear, modificar y eliminar objetos, así como administrar la seguridad de la base de datos.

'UsuarioAdmin': Este es el nombre del usuario que se está agregando al rol db_owner.

Roles de Base de Datos (A nivel solo de BD, distinto de servidor)

-- db_owner
--
-- Descripción: Proporciona control total sobre la base de datos.
-- Permisos: Los miembros pueden realizar cualquier acción en la base de datos, incluidos la creación y eliminación de objetos.

-- db_securityadmin
--
-- Descripción: Permite gestionar la seguridad de la base de datos.
-- Permisos: Pueden modificar permisos y roles en la base de datos.

-- db_accessadmin
--
-- Descripción: Permite gestionar el acceso a la base de datos.
-- Permisos: Pueden añadir o quitar usuarios de la base de datos.

-- db_backupoperator
--
-- Descripción: Permite realizar copias de seguridad de la base de datos.
-- Permisos: Pueden ejecutar copias de seguridad de la base de datos.

-- db_datareader
--
-- Descripción: Permite la lectura de datos de todas las tablas de la base de datos.
-- Permisos: Pueden consultar datos en todas las tablas.

-- db_datawriter
--
-- Descripción: Permite la escritura de datos en todas las tablas de la base de datos.
-- Permisos: Pueden insertar, actualizar y eliminar datos en todas las tablas.

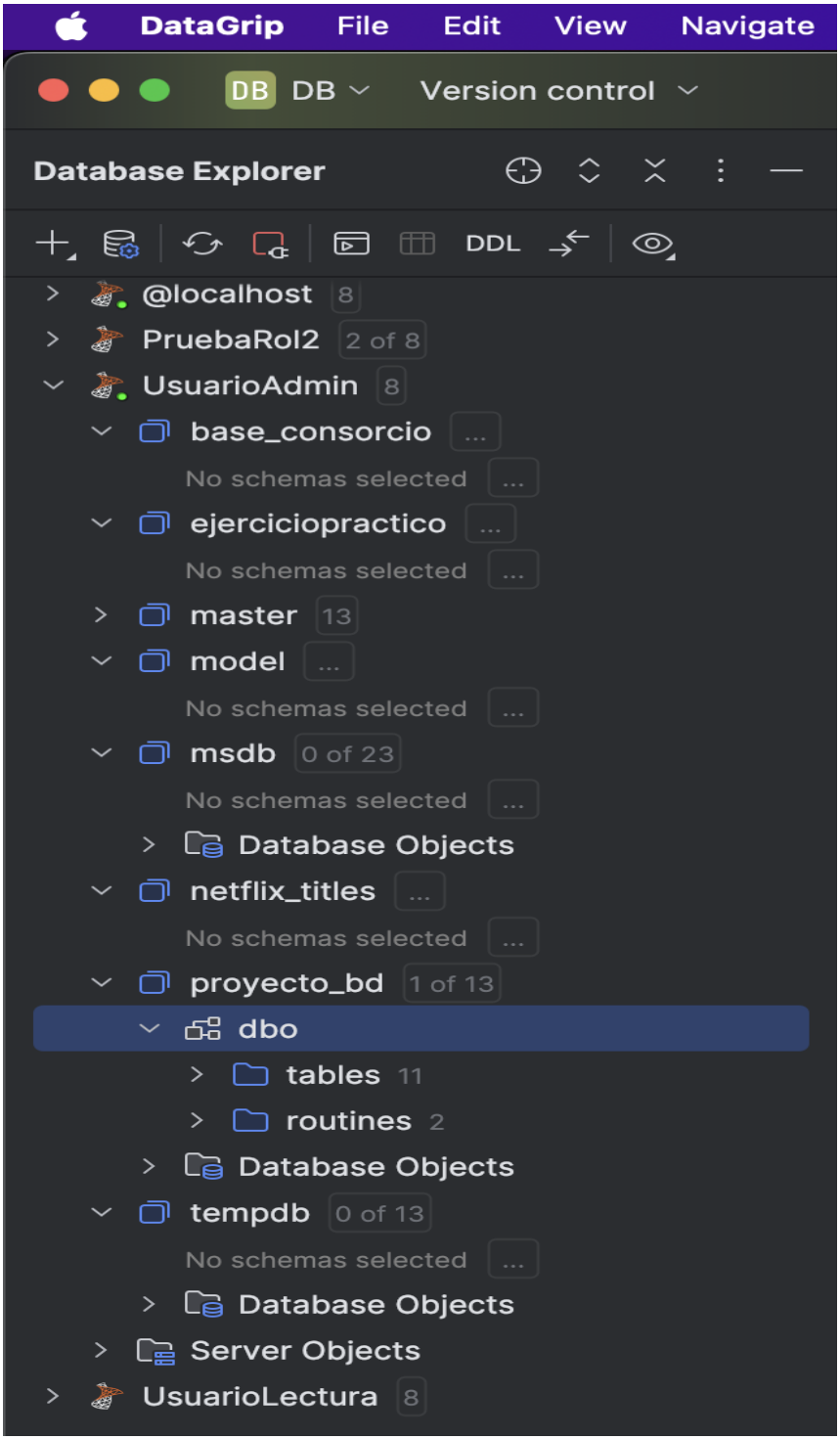
-- db_ddladmin
--
-- Descripción: Permite la modificación de la estructura de la base de datos.
-- Permisos: Pueden ejecutar comandos DDL (Data Definition Language) para crear y modificar objetos de la base de datos.

-- db_monitor
--
-- Descripción: Permite monitorizar el rendimiento de la base de datos.

Diferencias Clave entre db_owner y db_accessadmin

Característica	**db_owner**	**db_accessadmin**
Permisos de Acceso	Completos, incluye todo	Limitados a la administración de acceso
Crear Tablas	Sí	No
Modificar Objetos	Sí	No
Agregar Usuarios	No (aunque puede a través de otras tareas)	Sí
Eliminar Usuarios	No (aunque puede a través de otras tareas)	Sí

Tener en cuenta que al hacer la conexion a la Instancia con los usuarios nuevos, solo tendremos acceso a la base de datos desde la cual se creeo el usario, ‘proyecto_bd’ en este caso.
No podemos visualizar ningún otro esquema de otras DB.



EJECUTAR INSERT CON USUARIO QUE SOLO TIENE PERMISOS DE LECTURA:

```
[2024-10-27 19:13:55] [S0005][229] Line 1: The INSERT permission was denied
on the object 'Provincia', database 'proyecto_bd', schema 'dbo'.
```

EJECUTAR SELECT * CON USUARIO QUE SOLO TIENE PERMISOS DE LECTURA:

```
proyecto_bd> select * from dbo.Provincia
[2024-10-27 19:14
:05] 0 rows retrieved in 65 ms (execution: 39 ms, fetching: 26 ms)
```

Observamos como es posible leer pero no insertar registros con el usuario de lectura.
Otro es el caso cuando ejecutamos en insert con el usuario admin, lo inserta efectivamente:

```
proyecto_bd> insert into dbo.Provincia (nombre_provincia)  values
('Corrientes')
[2024-10-28 23:00:28] 1 row affected in 297 ms
```

Procederemos a ejecutar un procedimientos almacenado que solamente lee en la DB.
EJECUTAR PROCEDIMIENTO (ContarInspeccionesPendientes) QUE LEE CON EL USUARIO ADMIN

```
proyecto_bd> SELECT * FROM dbo.ContarInspeccionesPendientes()
[2024-10-27 21:53:26] 1 rows retrieved in 36 ms (execution: 9 ms, fetching:
27 ms)
```

EJECUTAR PROCEDIMIENTO (ContarInspeccionesPendientes) QUE LEE CON EL USUARIO DE LECTURA

```
proyecto_bd> SELECT * FROM dbo.ContarInspeccionesPendientes()
[2024-10-27 21:53:26] 1 rows retrieved in 36 ms (execution: 9 ms, fetching:
27 ms)
```

Observamos que al tratarse de una sentencia que solo lee y como ambos tienen permisos para hacerlo, la consulta es exitosa.

Procederemos a ejecutar un procedimientos almacenado que escribe en la DB.

EJECUTAR PROCEDIMIENTO (proc_AgregarInspector) QUE ESCRIBE CON EL USUARIO ADMIN

```
proyecto_bd> EXEC proc_AgregarInspector 'Lucas', 'Cabrera', 43108457,
37940283, 'juan.perez@example.com', '1970-01-01', 3000.00
[2024-10-27 21:58:05] 1 row affected in 87 ms
```

Observamos que efectivamente inserta un nuevo inspector

EJECUTAR PROCEDIMIENTO (proc_AgregarInspector) QUE ESCRIBE CON EL USUARIO DE LECTURA

```
proyecto_bd> EXEC proc_AgregarInspector 'Lucas', 'Cabrera', 43108457,
37940283, 'juan.perez@example.com', '1970-01-01', 3000.00
[2024-10-27 22:00:40] [S0005][229] Line 1: The EXECUTE permission was denied
on the object 'proc_AgregarInspector', database 'proyecto_bd', schema 'dbo'.
```

Observamos que el usuario al no tener permisos para escribir arroja un error.
La unica manera de que el usuario de lectura pueda ejecutar el procedimiento, es si le damos permisos de la siguiente manera:

```
GRANT EXECUTE ON proc_AgregarInspector TO UsuarioLectura;
```

Luego de otorgar el permisos:

```
proyecto_bd> EXEC proc_AgregarInspector 'Franco', 'Cabrera', 43103457,
37940283, 'juan2.perez@example.com', '1970-01-01', 3000.00
[2024-10-27 22:04:33] 1 row affected in 38 ms
```

Efectivamente el usuario de lectura logra insertar un segundo inspector

Ahora si hacemos los inserts sobre la tabla inspector con el usuario admin :

```
proyecto_bd> insert into dbo.Inspector(nombre_inspector, apellido_inspector,
dni_inspector, telefono_inspector, correo_inspector, fecha_nacimiento,
salario) values(
 'Feder', 'Cabreros', 43008457, 37940283, 'juan.perez@examp2le.com', '1970-
01-01', 3000.00
)
[2024-10-27 22:09:15] 1 row affected in 25 ms
```

Lo inserta correctamente, sin embargo desde el usuario de lectura:

```
proyecto_bd> insert into dbo.Inspector(nombre_inspector, apellido_inspector,
dni_inspector, telefono_inspector, correo_inspector, fecha_nacimiento,
salario) values ('Feder','Cabreros',43008457,37940283,
'juan.perez@examp2le.com','1970-01-01', 3000.00)
[2024-10-27 22:10:45] [S0005][229] Line 1: The INSERT permission was denied
on the object 'Inspector', database 'proyecto_bd', schema 'dbo'.
```

Obtenemos un error de permisos, por lo tanto la unica manera de hacerlo es atraves del procedimiento.

3 Permisos a nivel de roles

Los permisos a nivel de roles permiten asignar y gestionar derechos de acceso a los usuarios de una base de datos de forma organizada. A través de los roles, se otorgan permisos específicos que pueden aplicarse a varios usuarios de manera consistente

Crear dos usuarios de base de datos.

```
CREATE LOGIN PruebaRol1 WITH PASSWORD = 'PruebaRol1!'
CREATE USER PruebaRolUno FOR LOGIN PruebaRol1;
```

```
CREATE LOGIN PruebaRol2 WITH PASSWORD = 'PruebaRol2!';
CREATE USER PruebaRolDos FOR LOGIN PruebaRol2;
```

Crear un rol que solo permita la lectura de alguna de las tablas creadas.

```
CREATE ROLE LecturaRol;
```

Este comando crea un rol en la base de datos llamado LecturaRol.
Los roles en SQL Server sirven para agrupar permisos de manera que puedan aplicarse de forma consistente a varios usuarios.
Esto facilita la gestión de permisos, ya que, en lugar de asignar permisos individualmente a cada usuario, puedes asignarlos a un rol y luego agregar usuarios a ese rol.

```
GRANT SELECT ON Inspector TO LecturaRol;
```

Este comando otorga el permiso de SELECT (lectura) sobre la tabla Inspector al rol LecturaRol.
SELECT permite a los usuarios miembros de este rol realizar consultas de lectura en la tabla Inspector, pero no modificar los datos.

```
EXEC sp_addrolemember 'LecturaRol', 'PruebaRolUno';
```

sp_addrolemember: Es un procedimiento almacenado en SQL Server que se usa para agregar un usuario a un rol de base de datos.
'LecturaRol': Es el nombre del rol en el cual estás agregando el usuario.
'NombreUsuario': Es el nombre del usuario de la base de datos que estás agregando al rol.

Verificar el comportamiento de ambos usuarios (el que tiene permiso sobre el rol y el que no tiene), cuando intentan leer el contenido de la tabla

Al ejecutar un select * de la tabla Inspector con el usuario que tiene el rol de lectura sucede lo siguiente:

```
proyecto_bd> SELECT * FROM dbo.Inspector
[2024-10-27 22:30:16] 3 rows retrieved starting from 1 in 44 ms (execution:
11 ms, fetching: 33 ms)
```

Se obtienen todos los inspectores, hay que mencionar que las unicas tablas que mi DB managment me permite visualizar son aquellos para los cuales el rol tiene acceso
No se me permite ni ejecutar o visualizar las otras tablas.

Al ejecutar un select * de la tabla Inspecto con el usuario que tiene No tiene Rol sucede lo siguiente:
No es posible ejecutar un comando ya que en db managment no reconoce la tabla y al ejecutar la query..

```
proyecto_bd> select * from dbo.Inspector
[2024-10-27 22:34:16] [S0005][229] Line 1: The SELECT permission was denied
on the object 'Inspector', database 'proyecto_bd', schema 'dbo'.
```

Ahora con respecto a los procedimientos almacenados sucede lo siguiente:

SI TENEMOS PERMISOS DE LECTURA SOBRE LA TABLA LA CUAL EL PROCEDIMIENTO ALMACENADO LEE:

```
SELECT * FROM dbo.ContarInspectoresMayores30()
[2024-10-28 23:11:23] [S0002][208] Line 1: Invalid object name
'dbo.ContarInspectoresMayores30'.
```

Ocurrirá un error no podrá ejecutarlo.
Ahora si le damos permisos sobre el procedimiento y el usuario posee permisos de lectura en la tabla la cual lee el procedimiento almacenado:

```
GRANT EXECUTE ON dbo.ContarInspectoresMayores30 TO PruebaRolUno;
```

Y al ejecutarlo:

```
SELECT dbo.ContarInspectoresMayores30() AS TotalInspectoresMayores30;
```

Obtenemos el resultado: 3 registros

SI NO TENEMOS PERMISOS DE LECTURA SOBRE LA TABLA LA CUAL EL PROCEDIMIENTO ALMACENADO LEE, PERO SI TENEMOS PERMISOS SOBRE EL PROCEDIMIENTO:

```
GRANT EXECUTE ON dbo.ContarProyectosEnCorrientes TO PruebaRolUno;
```

```
SELECT dbo.ContarProyectosEnCorrientes() AS proyectos
```

Obtenemos el siguiente resultado : 0
Efectivamente nos deja ejecutar el procedimiento y logra leer la tabla, a pesar de que el rol no tiene permisos sobre la tabla directamente, si lo tiene sobre el procedimiente

Conclusión

El manejo adecuado de permisos y roles en bases de datos es crucial para una buena administración y seguridad de los datos. La asignación correcta de permisos específicos a los usuarios garantiza que cada persona tenga acceso solo a la información necesaria, minimizando el riesgo de accesos no autorizados y protegiendo así la integridad de los datos. Implementar roles y usuarios bien definidos contribuye a un entorno de trabajo más seguro y controlado, beneficiando la administración y el uso efectivo de los recursos de la base de datos.