Laboratorio 2

Gestión de usuarios y administración de permisos en SQL Server

En la administración de bases de datos, una de las tareas fundamentales es gestionar los usuarios y controlar los permisos que poseen sobre los objetos del sistema. Una configuración adecuada garantiza la seguridad, la integridad de los datos y la correcta operación de las aplicaciones que acceden al sistema.

Por ello es importante conocer los siguientes puntos:

1. Diferencia entre inicio de sesión (Login) y usuario de base de datos

En SQL Server, aunque están relacionados, **login** y **usuario** no son lo mismo:

- **Login (inicio de sesión):** Es la credencial de acceso al motor de base de datos. Se crea a nivel del servidor y permite autenticar a una persona o aplicación.
  Ejemplo: `CREATE LOGIN orange WITH PASSWORD = 'P@ssw0rd!2025'.`

- **Usuario de base de datos:** Es la identidad dentro de una base de datos específica que se asocia a un login. Permite que un inicio de sesión tenga acceso a una base de datos en particular.
  Ejemplo: `CREATE USER orange FOR LOGIN orange`.

Nota:

Un login puede existir sin tener usuario en una base de datos, pero un usuario siempre necesita estar asociado a un login para poder acceder.

2. Principio de mínimo privilegio

La seguridad por mínimo privilegio indica que un usuario o aplicación debe contar únicamente con los permisos necesarios para realizar sus tareas y nada más.
Esto disminuye riesgos de:

- Accesos no autorizados.
- Pérdida o corrupción de datos.
- Escalamiento indebido de privilegios.

3. Roles en SQL Server

En lugar de asignar permisos directamente a cada usuario, SQL Server recomienda el uso de roles. Un rol es un contenedor de permisos que facilita la administración:

- **Roles de servidor:** Definen permisos a nivel global (ejemplo de ello es el: `sysadmin`).

- **Roles de base de datos:** Controlan permisos dentro de una base de datos específica.

Asignar usuarios a roles en lugar de manejar permisos individuales simplifica la gestión y mejora la seguridad.

Ejemplos:

- Un rol de solo lectura que permite ejecutar únicamente consultas `SELECT`.

- Un rol de aplicación que permite realizar operaciones de lectura, inserción y actualización sobre ciertas tablas, pero niega operaciones riesgosas como `DELETE` o `ALTER`.

4. Administración de permisos

Los permisos en SQL Server se pueden otorgar o restringir de diferentes maneras:

- **GRANT:** Concede permisos sobre objetos (ej. `GRANT SELECT`)
- **DENY:** Niega permisos explícitamente, incluso si el rol tiene permisos concedidos.
- **REVOKE:** Revoca permisos previamente otorgados o denegados.

Un diseño seguro incluye defensa en profundidad, combinando permisos mínimos con denegaciones explícitas de operaciones peligrosas.

5. Comprobación de permisos

Para verificar los permisos de un usuario se pueden usar mecanismos como:

- `EXECUTE AS USER`: simula que se está operando como un usuario específico.
- `fn_my_permissions`: función que muestra los permisos efectivos que un usuario posee en un contexto determinado.

6. **Buenas prácticas de seguridad**

1. **Contraseñas seguras:**
   1) Uso de mayúsculas, minúsculas, números y caracteres especiales.
   2) Políticas de expiración y rotación periódica.

2. **Principio de menor privilegio:**
   1) Nunca otorgar permisos amplios a menos que sea estrictamente necesario.

3. **Uso de roles:**
   1) Evitar permisos directos sobre cada usuario.

2) Crear roles específicos para distintos perfiles de uso (ej. lectura, carga de datos, administración).

4. **Defensa en profundidad:**
   1) Permisos mínimos combinados con denegaciones explícitas.

5. **Supervisión de cuentas privilegiadas:**
   1) El rol `sysadmin` debe reservarse para administradores de bases de datos.
   2) Nunca usar `sa` o `sysadmin` en conexiones de aplicaciones.

6. **Auditoría y monitoreo:**
   1) Habilitar auditoría de inicios de sesión fallidos.
   2) Revisar regularmente permisos y roles asignados.