Universidad Tecnológica de Xicotepec de Juárez

Tecnologías de la Información

Ingeniería y Desarrollo en gestión de Software

Unidad 3

ADMINISTRACIÓN DE BASES DE DATOS

Plan de Seguridad

Integrantes del equipo

Jesus Alejandro Cabrera García

Iram Daniel Zúñiga Agustín

Carlos Eduardo Varela Barrios

Objetivo General

Garantizar la confidencialidad, integridad y disponibilidad de la información almacenada en la base de datos, protegiendo los datos personales de los empleados y garantizando el cumplimiento de la normativa vigente en materia de protección de datos.

Objetivos Específicos

- Establecer roles de usuario específicos para cada tipo de usuario (administrador, desarrollador, analista, etc.) con permisos acotados según sus responsabilidades.

- Establecer políticas de acceso y uso de la base de datos que sean claras y que todos los usuarios deben seguir.

- Realizar copias de seguridad periódicas de la base de datos para prevenir la pérdida de información en caso de un incidente de seguridad.

- Mantenerse al tanto de las actualizaciones de seguridad de MySQL y aplicar parches y correcciones de seguridad de manera oportuna.

- Capacitar a los usuarios sobre las mejores prácticas de seguridad para proteger la información confidencial almacenada en la base de datos.

Roles

Roles

Los roles creados en una base de datos son de suma importancia ya que son una herramienta poderosa para gestionar la seguridad, simplificar la administración de usuarios y garantizar el cumplimiento normativo, proporcionando un enfoque flexible y eficiente para controlar el acceso a los datos y funcionalidades de la base de datos.

Para este caso los roles de la base de datos son los siguientes:

- Cliente

Se refiere a la persona o entidad que adquiere productos o servicios de una empresa.

- Gerente

Se trata del individuo responsable de supervisar y dirigir las operaciones de una organización o de un equipo dentro de una empresa.

- Empleado

Es un individuo contratado por una empresa encargado de realizar tareas específicas en función de un contrato laboral o acuerdo del trabajo asignado.

- Vendedor

Sujeto que se dedica a la venta de productos o servicios a clientes potenciales en nombre de la empresa a la que presta servicios o ya sea por cuenta propia.

En MySQL, los roles son una funcionalidad que permite asignar conjuntos específicos de privilegios a un grupo de usuarios. Esto facilita la gestión de permisos al centralizar la asignación de privilegios a un solo objeto.

Algunos roles predefinidos en MySQL son:

1. DBA (Administrador de base de datos): Este rol tiene privilegios para realizar tareas de administración de la base de datos, como crear, modificar o eliminar bases de datos y tablas, así como asignar privilegios a otros usuarios.

2. Developer (Desarrollador): Este rol tiene privilegios para realizar tareas relacionadas con el desarrollo de aplicaciones, como insertar, actualizar y eliminar registros en tablas, así como crear y modificar procedimientos almacenados y funciones.

3. Usuario estándar: Este rol tiene privilegios limitados y solo puede leer datos de las tablas, sin poder realizar cambios en la base de datos.

Es importante tener en cuenta que los roles deben ser creados y asignados por un usuario con privilegios de administración. Los usuarios pueden ser asignados a uno o varios roles, facilitando la gestión de permisos y la seguridad de la base de datos.

En MySQL, los roles no son una funcionalidad nativa que se pueda asignar a un usuario directamente como en otros gestores de bases de datos como PostgreSQL. Sin embargo, es posible simular el comportamiento de roles mediante el uso de privilegios y roles predefinidos.

¿Cómo crear roles en MySQL?

Para asignar roles en MySQL, primero debes tener una cuenta con privilegios de administrador en el servidor de MySQL. Luego, puedes seguir los pasos a continuación:

1. Iniciar sesión en el servidor de MySQL con tu cuenta de administrador.

2. Crear un nuevo rol utilizando el comando CREATE ROLE seguido del nombre del rol que deseas crear. Por ejemplo, para crear un rol llamado 'rol_ejemplo', puedes usar el siguiente comando:

3. Asignar permisos al rol recién creado utilizando el comando GRANT. Por ejemplo, si deseas otorgar permisos de lectura en una tabla específica, puedes utilizar el siguiente comando:

```
GRANT SELECT ON nombre_tabla TO rol_ejemplo;
```

4. Asignar usuarios al rol utilizando el comando GRANT. Por ejemplo, si deseas asignar el rol 'rol_ejemplo' al usuario 'usuario1', puedes utilizar el siguiente comando:

```
GRANT rol_ejemplo TO usuario1;
```

Una vez que hayas asignado el rol a un usuario, ese usuario tendrá los privilegios que se le han asignado a través del rol. También puedes revocar un rol utilizando el comando REVOKE.

Es importante recordar que los roles en MySQL solo están disponibles en la versión 8.0 y posteriores. Si estás utilizando una versión anterior, puedes simular roles utilizando procedimientos almacenados y vistas de MySQL.

¿Cómo crear roles en SQL?

Para crear roles en SQL, se pueden seguir los siguientes pasos:

1. Conectar a la base de datos utilizando un usuario con permisos de administrador.

2. Ejecutar el comando `CREATE ROLE nombre_rol;` para crear un nuevo rol en la base de datos.

3. Para asignar permisos al rol, se pueden utilizar los comandos `GRANT` y `REVOKE`. Por ejemplo, para otorgar permisos de lectura en una tabla específica, se puede ejecutar el comando `GRANT SELECT ON nombre_tabla TO nombre_rol;`.

4. Para asignar el rol a un usuario, se puede ejecutar el comando `GRANT nombre_rol TO nombre_usuario;`.

5. Para revocar un rol de un usuario, se puede ejecutar el comando `REVOKE nombre_rol FROM nombre_usuario;`.

6. Para eliminar un rol, se puede ejecutar el comando `DROP ROLE nombre_rol;`.

Es importante tener en cuenta que el uso de roles en SQL puede variar dependiendo del sistema de gestión de base de datos utilizado (MySQL, PostgreSQL, SQL Server, etc.), por lo que se recomienda consultar la documentación específica de cada sistema.

Para crear roles en SQL, se puede utilizar el comando `CREATE ROLE`. Aquí hay un ejemplo de cómo crear un nuevo rol en SQL:

```
CREATE ROLE mi_rol;
```

También se puede asignar privilegios a un rol utilizando el comando `GRANT`. Por ejemplo, para asignar permisos de lectura en una tabla específica al rol que acabamos de crear, se puede hacer de la siguiente manera:

```
GRANT SELECT ON nombre_tabla TO mi_rol;
```

Para asignar este nuevo rol a un usuario específico, se puede utilizar el comando `GRANT` de la siguiente manera:

```
GRANT mi_rol TO nombre_usuario;
```

De esta forma, se ha creado un nuevo rol en SQL y se ha asignado a un usuario con los permisos necesarios.

¿Cómo crear roles en NoSQL?

En NoSQL, la creación de roles generalmente se gestiona a través de la configuración del sistema de control de acceso o a través de una base de datos específica. Aquí te presento un ejemplo usando MongoDB:

1. Inicia sesión en la base de datos de MongoDB como un usuario con permisos de administrador.

2. Utiliza el comando use admin para cambiar a la base de datos "admin".

3. Crea un nuevo rol utilizando el comando db.createRole() y proporcionando los privilegios necesarios para el rol. Por ejemplo, para crear un rol con acceso de lectura a una base de datos específica, puedes ejecutar el siguiente comando:

```
db.createRole(
   {
      role: "read_only",
      privileges: [
         {
            resource: { db: "mydatabase", collection: "" },
            actions: [ "find", "collStats" ]
         }
      ],
      roles: []
   }
)
```

4. Asigna el nuevo rol a un usuario utilizando el comando db. grantRolesToUser (). Por ejemplo, para asignar el rol "read_only" al usuario "myuser", ejecuta el siguiente comando:

```
db.grantRolesToUser(
   "myuser",
   [ { role: "read_only", db: "mydatabase" } ]
)
```

Estos son solo ejemplos de cómo crear roles en MongoDB, otros sistemas NoSQL pueden tener sus propios comandos y métodos de gestión de roles. Si estás utilizando una base de datos NoSQL diferente, te recomiendo consultar la documentación oficial de dicha base de datos para obtener instrucciones detalladas sobre cómo crear roles.

Usuarios

En MySQL, los usuarios son entidades que tienen permisos para acceder a la base de datos y ejecutar operaciones en ella. Cada usuario tiene un nombre único y una contraseña asociada. Los usuarios pueden tener diferentes niveles de privilegios, lo que les permite realizar distintas acciones en la base de datos, como leer datos, modificar tablas o crear nuevas bases de datos. Los usuarios en MySQL se gestionan a través de comandos específicos, como CREATE USER, DROP USER, GRANT y REVOKE.

¿Cómo crear usuarios en MySQL?

Para crear un usuario en MySQL, debes seguir los siguientes pasos:

1. Iniciar sesión en MySQL con una cuenta que tenga privilegios de administrador, por ejemplo, la cuenta "root".

2. Una vez dentro de MySQL, usa el siguiente comando para crear un nuevo usuario:

```
CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

Reemplaza 'nombre_usuario' por el nombre de usuario que deseas crear y 'contraseña' por la contraseña que deseas asignarle.

3. Después de crear el usuario, debes asignarle los privilegios necesarios. Puedes asignarle todos los privilegios usando el siguiente comando:

```
GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost';
```

Esto le dará al usuario todos los permisos sobre todas las bases de datos y tablas de MySQL. Si deseas limitar los permisos, puedes especificarlos más detalladamente en el comando GRANT.

4. Por último, asegúrate de aplicar los cambios ejecutando el comando:

```
FLUSH PRIVILEGES;
```

¿Cómo crear usuarios en SQL?

Para crear un usuario en SQL, puedes utilizar el comando `CREATE USER` seguido del nombre del usuario y la contraseña. Por ejemplo:

```
CREATE USER 'nombre_usuario' IDENTIFIED BY 'contraseña';
```

También puedes asignarle privilegios al usuario con el comando `GRANT`. Por ejemplo, para darle permisos de lectura y escritura a una tabla:

```
GRANT SELECT, INSERT, UPDATE ON nombre_tabla TO nombre_usuario;
```

Recuerda que es importante asignar los permisos necesarios de acuerdo a las necesidades de cada usuario en tu base de datos.

¿Cómo crear usuarios en NoSQL?

La creación de usuarios en bases de datos NoSQL puede variar dependiendo del tipo de base de datos que estés utilizando. A continuación, se mostrará el paso de cómo crear usuarios en MongoDB, una base de datos NoSQL:

1. Acceder a la consola de Mongo utilizando el comando `mongo`.

2. Conéctate a la base de datos en la que quieres crear el usuario utilizando el comando `use nombre_de_la_base_de_datos`.

3. Ejecuta el comando `db.createUser({user: "nombre_usuario", pwd: "contraseña", roles: ["rol"]})`, donde sustituyes "nombre_usuario" por el nombre que quieres darle al usuario, "contraseña" por la contraseña que deseas asignarle y "rol" por el rol que quieres asignarle al usuario (por ejemplo, "read" para roles de solo lectura y "readWrite" para roles de lectura y escritura).

Con estos pasos habrás creado un usuario en tu base de datos MongoDB. Recuerda que es importante asignar roles adecuados a los usuarios para garantizar la seguridad y el acceso controlado a la base de datos.

Privilegios

En MySQL, los privilegios son permisos que se otorgan a los usuarios para realizar determinadas acciones en la base de datos. Algunos ejemplos de estos privilegios son:

1. SELECT: Permite al usuario consultar datos en las tablas de la base de datos.
2. INSERT: Permite al usuario agregar nuevos registros a las tablas.
3. UPDATE: Permite al usuario modificar los registros existentes en las tablas.
4. DELETE: Permite al usuario eliminar registros de las tablas.
5. CREATE: Permite al usuario crear nuevas tablas en la base de datos.
6. DROP: Permite al usuario eliminar tablas de la base de datos.
7. GRANT: Permite al usuario otorgar privilegios a otros usuarios.
8. REVOKE: Permite al usuario quitar privilegios a otros usuarios.

1. SELECT:
    a. Ejemplo de comando: SELECT * FROM nombre_tabla;

2. INSERT:
    b. Ejemplo de comando: INSERT INTO nombre_tabla VALUES (valor1, valor2, ...);

3. UPDATE:
    c. Ejemplo de comando: UPDATE nombre_tabla SET columna = valor WHERE condicion;

4. DELETE:
    d. Ejemplo de comando: DELETE FROM nombre_tabla WHERE condicion;

5. CREATE:
    e. Ejemplo de comando: CREATE TABLE nombre_tabla (columna1 tipo1, columna2 tipo2, ...);

6. DROP:
    f. Ejemplo de comando: DROP TABLE nombre_tabla;

7. GRANT:
    g. Ejemplo de comando: GRANT privilegio ON nombre_tabla TO usuario;

8. REVOKE:
    h. Ejemplo de comando: REVOKE privilegio ON nombre_tabla FROM usuario;

Estos son solo algunos ejemplos de los privilegios que se pueden otorgar en MySQL. Es importante asignar los privilegios adecuados a cada usuario para garantizar la seguridad y la integridad de la base de datos.

¿Cómo asignar privilegios en MySQL?

Para asignar privilegios en MySQL, primero necesitas tener una cuenta de usuario con privilegios de administrador (por ejemplo, el usuario "root"). Luego puedes seguir estos pasos:

1. Inicia sesión en MySQL usando el comando `mysql -u root -p`.

2. Selecciona la base de datos en la que deseas asignar privilegios utilizando el comando `USE nombre_de_la_base_de_datos;`. Puedes utilizar la siguiente sintaxis para otorgar un privilegio específico a un usuario en una base de datos:

```
GRANT tipo_de_privilegio ON nombre_de_base_de_datos.* TO 'nombre_de_usuario'@'localhost';
```

Por ejemplo, para otorgar todos los privilegios en una base de datos llamada mi_base_de_datos a un usuario llamado mi_usuario, usarías la siguiente sentencia

```
GRANT ALL PRIVILEGES ON mi_base_de_datos.* TO 'mi_usuario'@'localhost';
```

3. Después de ejecutar la sentencia GRANT, debes recargar los privilegios para que los cambios surtan efecto. Puedes hacerlo ejecutando el siguiente comando:

```
FLUSH PRIVILEGES;
```

Es importante recordar que el uso de la cláusula ALL PRIVILEGES puede ser un riesgo de seguridad si no se hace de manera adecuada. Se recomienda otorgar solo los privilegios necesarios para que el usuario realice sus tareas específicas.

¿Cómo asignar privilegios en SQL?

Para asignar privilegios en SQL, se utilizan comandos específicos según el gestor de base de datos que se esté utilizando. A continuación, se muestran algunos ejemplos comunes para asignar privilegios en SQL:

- En MySQL: Para asignar privilegios en MySQL, se utiliza el comando GRANT seguido de los privilegios que se deseen asignar y el nombre del usuario al que se le otorgan los privilegios. Por ejemplo, para dar todos los privilegios a un usuario en una base de datos llamada "testdb", se puede usar el siguiente comando:

```
GRANT ALL PRIVILEGES ON testdb.* TO 'usuario'@'localhost';
```

- En PostgreSQL: Para asignar privilegios en PostgreSQL, se utiliza el comando GRANT seguido de los permisos que se deseen otorgar y el nombre del usuario al que se le otorgan los privilegios. Por ejemplo, para dar SELECT y UPDATE en una tabla llamada "tabla_prueba" a un usuario llamado "usuario1", se puede usar el siguiente comando:

```
GRANT SELECT, UPDATE ON tabla_prueba TO usuario1;
```

Es importante tener en cuenta que se deben tener privilegios suficientes para asignar privilegios a otros usuarios. Además, se recomienda revisar la documentación específica de cada gestor de base de datos para obtener más información sobre cómo asignar privilegios en SQL.

¿Cómo asignar privilegios en NoSQL?

En NoSQL, los privilegios se pueden asignar de varias formas, dependiendo del tipo de base de datos NoSQL que estés utilizando. Aquí hay algunas formas comunes de asignar privilegios en diferentes tipos de bases de datos NoSQL:

1. En MongoDB, puedes asignar privilegios utilizando el sistema de autenticación y autorización integrado en la base de datos. Puedes crear usuarios y roles con diferentes niveles de acceso a las bases de datos y colecciones. Puedes asignar privilegios de lectura, escritura, actualización o eliminación a usuarios específicos o roles.

2. En Couchbase, puedes asignar privilegios utilizando los roles y permisos definidos en la configuración de la base de datos. Puedes crear roles con diferentes niveles de acceso y asignar estos roles a usuarios específicos o grupos de usuarios. En Couchbase, se pueden asignar roles mediante el siguiente comando:

```
cbauth bucket --set-acl nombre_bucket --read-only <nombre_usuario>
```

3. En Cassandra, puedes asignar privilegios utilizando el sistema de autorización basado en roles de la base de datos. Puedes crear roles con diferentes permisos de lectura/escritura y asignar estos roles a usuarios específicos o grupos de usuarios.

En general, la asignación de privilegios en NoSQL se realiza a través de sistemas de autenticación y autorización integrados en la base de datos, que permiten crear usuarios, roles y permisos personalizados para controlar el acceso a los datos. Es importante revisar la documentación específica de la base de datos NoSQL que estés utilizando para obtener instrucciones detalladas sobre cómo asignar privilegios en esa plataforma específica.

En NoSQL, los privilegios se asignan generalmente mediante el uso de roles y usuarios. Algunas bases de datos NoSQL populares como MongoDB y Couchbase pueden utilizar comandos específicos para asignar privilegios.

Por ejemplo, en MongoDB se pueden asignar privilegios a un usuario utilizando el siguiente comando:

```
db.grantRolesToUser("nombre_usuario", [{role: "nombre_rol", db: "nombre_base_de_datos"}])
```

Tablas

Un plan de seguridad en una base de datos debe contemplar una serie de medidas y controles para proteger la información contenida en la base de datos. A continuación, se presenta una lista de tablas que podrían formar parte de un plan de seguridad en una base de datos:

1. Tabla de Usuarios:

- ID de usuario (PK)
- Nombre de usuario
- Contraseña
- Nivel de acceso

2. Tabla de Empleados:

- ID de empleado (PK)
- Nombre
- Apellido
- Cargo
- Departamento
- Fecha de ingreso

3. Tabla de Acceso:

- ID de acceso (PK)
- ID de empleado (FK)
- Fecha y hora de acceso
- Tipo de acceso (entrada/salida)
- Ubicación

4. Tabla de Permisos:

- ID de permiso (PK)
- ID de usuario (FK)
- Nivel de acceso

5. Tabla de Registros de Seguridad:

- ID de registro (PK)

- ID de usuario (FK)

- Fecha y hora del evento

- Acción realizada

- Descripción del evento

Estas tablas ayudarán a mantener un registro y control de acceso, además de definir los roles y permisos de los usuarios en la base de datos de recursos humanos. Se pueden establecer procedimientos y políticas de seguridad para garantizar la confidencialidad e integridad de la información.

Estas tablas formarían parte de un sistema de gestión de seguridad de la base de datos que permitiría controlar y monitorear el acceso a la información de manera efectiva y protegerla de posibles amenazas.

Calendario

Nuestra UDN cuenta con las siguientes tablas, de las cuales somos los encargados:

SQL

1. Empleados
2. Instructores
3. Áreas
4. Puestos
5. Horarios

NOSQL

1. Empleados

Los datos se pueden salvaguardar de la siguiente manera:

- Copia completa de los datos de la plataforma (un día)
- Copias diarias sucesivas de los datos salvaguardando a disco solo las diferencias con el día anterior. De esta manera ahorramos espacio en disco evitando la copia completa de todos los datos a diario.

Las medidas anteriores serán aplicadas a las tablas de mayor importancia que contengan datos relevantes para el sistema como lo son:

1. Empleados
2. Instructores
3. Áreas
4. Puestos
5. Horarios

Para las tablas restantes deberán realizarse copias de respaldo al menos semanalmente, salvo que en dicho periodo no se hubiera producido ninguna actualización de los datos.

Localización de los backups

- Memorias USB.
- la nube (Internet).

Horario de respaldos

Base de Datos SQL

1 vez al día

Respaldo incremental de la Tabla de empleados

Lunes: Respaldo incremental de la Tabla de Empleados

Martes: Respaldo incremental de la Tabla de Usuarios y Tabla de Permisos

Miércoles: Respaldo incremental de la Tabla de Empleados

Jueves: Respaldo incremental de la Tabla de Usuarios y Tabla de Permisos

Viernes: Respaldo incremental de la Tabla de Empleados

Sábado: Respaldo completo de las tablas de Usuarios, Empleados y Permisos

Domingo: Día de verificación de respaldos

Empleados 3:00 AM

Instructores 3:00 AM

Áreas 3:00 AM

Puestos 3:00 AM

Horarios 3:00 AM

Base de Datos SQL

Al ser solo una colección en la base de datos, se vio conveniente que el respaldo sea diario para conservar siempre la información.

Tabla:

Reclutamiento…… 3:00 AM