CBTis#42

DESARROLLA SOFTWARE DE APLICACIÓN WEB CON ALMACENAMIENTO PERSISTENTE DE DATOS.

Actividad 1-Unidad 3

Programación

MÓDULO IV:
DESARROLLA SOFTWARE DE APLICACIÓN WEB CON ALMACENAMIENTO PERSISTENTE DE DATOS.

SUBMÓDULO 1:
CONSTRUYE BASES DE DATOS PARA APLICACIONES WEB.
Unidad 3 Administra la información de la base de datos

Elaborado por: Abraham Castañeda Quintero
Docente: Alejandra Serrano Luna
Semestre y grupo: 5° “B”

Índice.

Contenido

Introducción.

En esta la actividad No. 1 unidad 3 del submódulo 1 de la carrera de programación, se investigara sobre como acceder a una base de datos y como gestionar los distintos tipos de usuario asi comu su creación y sus privilegios.

Desarrollo.

1. Control de accesos en bases de datos

Sistemas Administradores de Bases de Datos (DBMS) como Oracle y Sybase pueden ser accesados a través de servicios Internet, utilizando sus propios protocolos de comunicación a través de TCP/IP. Esto requiere que el usuario utilice un programa cliente especial para el servidor DBMS en el cual realiza transacciones y, probablemente, deba usar un programa aplicación que le permita acceder sólo a una parte de los datos en el servidor, o bien, utilice un programa genérico que le permita accesar varios tipos de bases de datos con una interfaz común, pero limitada sólo a aspectos generales de las bases de datos.

Una nueva forma de accesar bases de datos es a través de los servicios de información tradicionales como Gopher o WWW, en que el usuario utiliza el mismo programa de uso habitual para acceder ese servicio.

El acceso mismo a la base de datos no la realiza el usuario en sí, sino que es el servicio de información el que efectúa las transaciones comunicándose con el DBMS, y luego entrega al usuario los resultados en forma ordenada. El diagrama de la Figura 3.1 muestra el enfoque aquí descrito.

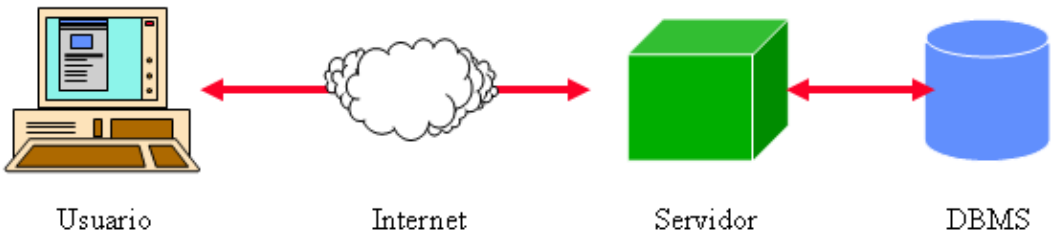

Figura 3.1: Modelo de servicio con acceso a una base de datos

Para que el servicio de información pueda accesar a la base de datos, se requiere que exista un medio de comunicación entre éstos, es decir, que sea posible establecer una conexión al DBMS desde el servicio de información. Esto puede lograrse de diversas formas, dependiendo del tipo de cada uno de los servidores, de la ubicación física de ellos, y de las herramientas disponibles para tales efectos.

Por ejemplo, si el servicio de información es WWW, el DBMS es Oracle, y ambos están instalados en la misma máquina o existe una vía de comunicación entre ellos como SQL*Net, es posible escribir un programa interfaz CGI en el servidor WWW que, en base a datos proporcionados por el usuario en un formulario, realice una consulta usando el producto SQL*Plus, utilizando un script SQL adecuado, y analize la respuesta obtenida para formatear el resultado en un documento hipertexto, el cual es enviado al usuario como respuesta a los datos recién proporcionados. Otra alternativa sería utilizar el producto Oracle

WebServer, quien canaliza las consultas directamente a procedimientos PL/SQL almacenados en el DBMS.

Posibles aplicaciones del acceso a las bases de datos a través de un servicio de información son las siguientes:

Se puede generar documentos al vuelo basado en los datos almacenados en una base de datos que es actualizada en línea, tales como el estado de cuenta de una persona en un banco, o la lista de vuelos en un aeropuerto programados para las próximas 24 horas.
Documentos que entregan información respecto a varios elementos en un mismo tema pueden ser construídos a partir de plantillas que son llenadas tomando los datos desde un servidor de bases de datos. Por ejemplo, se puede tener una base de datos relacional con información geográfica de diversos países, la cual puede ser accesada por el servicio de información para generar documentos con información básica de cada país a partir de una hoja tipo, además de poder resumir estadísticas por región o poder acceder a un mayor nivel de detalle. En un servicio basado en hipertexto, este ejemplo permitiría a un usuario viajar por todo el mundo, pasando de país en país en el caso de existir la relación “paises vecinos”, o bien desde un listado de los países que conforman un continente. Al mantener al día la base de datos, los documentos y sus relaciones serán actualizadas “automáticamente”.

COMO ACCEDER USANDO DE EJEMPLO (SQL SERVER 2019).

En el modelo tradicional de conexión, los usuarios de Windows o los miembros de los grupos de Windows se conectan a la Motor de base de datos al proporcionar las credenciales de usuario o grupo autenticadas por Windows. O bien, puede proporcionar un nombre y la contraseña y conectarse mediante autenticación de SQL Server . En ambos casos, la base de datos maestra debe tener un inicio de sesión que coincida con las credenciales de conexión. Después de que la Motor de base de datos confirme las credenciales de autenticación de Windows o autentica las credenciales de autenticación de SQL Server , la conexión normalmente intenta conectarse a una base de datos de usuario. Para conectarse a una base de datos de usuario, el inicio de sesión se debe poder asignar (es decir, asociar) a un usuario de base de datos en la base de datos de usuario. También es posible que la cadena de conexión especifique la conexión a una base de datos que es opcional en SQL Server pero obligatoria en SQL Database.
El principio importante es que tanto el inicio de sesión (en la base de datos maestra) como el usuario (en la base de datos de usuario) deben existir y estar relacionados entre sí. Esto significa que la conexión a la base de datos de usuario tiene una dependencia en el inicio de sesión en la base de datos maestra y esto limita la capacidad de la base de datos de moverse a un servidor host de SQL Server o Azure SQL Database diferente. Además, si por cualquier motivo no hay una conexión a la base de datos maestra disponible (por ejemplo, una conmutación por error está en curso), aumentará el tiempo total de conexión o

es posible que se agote el tiempo de espera de la conexión. En consecuencia, esto puede reducir la escalabilidad de la conexión.
Modelo de usuario de base de datos independiente
En el modelo de usuario de base de datos independiente, el inicio de sesión en la base de datos maestra no está presente. En su lugar, el proceso de autenticación se produce en la base de datos de usuario y el usuario de base de datos de la base de datos de usuario no tiene asociado ningún inicio de sesión en la base de datos maestra. El modelo de usuario de base de datos independiente admite tanto la autenticación de Windows como la autenticación de SQL Server , y se puede usar tanto en SQL Server como en SQL Database. Para conectarse como un usuario de base de datos independiente, la cadena de conexión siempre debe contener un parámetro para la base de datos de usuario de modo que la Motor de base de datos sepa qué base de datos es responsable de la administración del proceso de autenticación. La actividad del usuario de base de datos independiente se limita a la autenticación de base de datos, por lo que al conectarse como un usuario de base de datos independiente, la cuenta de usuario de base de datos debe crearse independientemente en cada base de datos que el usuario necesitará. Para cambiar las bases de datos, los usuarios de SQL Database deben crear una nueva conexión. Los usuarios de base de datos independiente de SQL Server pueden cambiar bases de datos si hay un usuario idéntico en otra base de datos.
Azure: SQL Database y Azure Synapse Analytics (SQL Data Warehouse) admiten las identidades de Azure Active Directory como usuarios de base de datos independiente. SQL Database admite usuarios de base de datos independientes que usan autenticación de SQL Server , pero Azure Synapse Analytics (SQL Data Warehouse) no. Para más información, consulte Usar la autenticación de Azure Active Directory para autenticación con SQL. Al utilizar la autenticación de Active Directory de Azure, las conexiones de SSMS se pueden realizar mediante autenticación universal de Active Directory. Los administradores pueden configurar la autenticación universal para requerir Multi-Factor Authentication, que comprueba la identidad mediante una llamada de teléfono, un mensaje de texto, una tarjeta inteligente con pin o una notificación de aplicación móvil. Para obtener más información, vea Compatibilidad de SSMS con Azure AD MFA con SQL Database y Azure Synapse Analytics.
Para SQL Database y Azure Synapse Analytics (SQL Data Warehouse), dado que el nombre de la base de datos siempre se requiere en la cadena de conexión, no se necesitan cambios en la cadena de conexión cuando se cambia desde el modelo tradicional al modelo de usuario de base de datos independiente. Para las conexiones de SQL Server , el nombre de la base de datos debe agregarse a la cadena de conexión si no está ya presente.

```
CREATE USER Carlo
WITH PASSWORD='Enterpwdhere*'

SELECT containment_desc FROM sys.databases
WHERE name='Test'
```

1.1. Tipos de usuarios en bases de datos

Usuarios normales. Son usuarios no sofisticados que interactúan con el sistema mediante un programa de aplicación con una interfaz de formularios, donde puede rellenar los campos apropiados del formulario. Estos usarios pueden también simplemente leer informes generados de la base de datos.

Programadores de aplicaciones. Son profesionales informáticos que escriben los programas de aplicación, utilizando herramientas para desarrollar interfaces de usuario, como las herramientas de desarrollo rápido de aplicaciones (DRA), que facilitan crear los formularios e informes sin escribir directamente el programa.

Usuarios sofisticados. Interactúan con el sistema sin programas escritos, usando el lenguaje de consulta de base de datos para hacer sus consultas. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría, usando ellos las herramientas de procesamiento analítico en línea (OLAP, OnLine Analytical Processing), o herramientas de recopilación de datos.

Usuarios especializados. Son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas y adecuadas para el procesamiento de datos tradicional. Entre estas aplicaciones están los sistemas de diseño asistido por computadora, sistemas de base de conocimientos y sistemas expertos, sistemas que almacenan datos de tipos de datos complejos (como gráficos y de audio) y sistemas de modelado de entorno.

Administradores de la base de datos (ABD). Son las personas que tienen el control central del SGBD. Entre las funciones del ABD se encuentran:

Definición del esquema de la base de datos.
Definición de la estructura y el método de acceso.
Modificación del esquema y la organización física.
Concensión de autorización para el acceso a los datos.
Mantenimiento rutinario.

1.1.1. ¿Qué función desempeñan los usuarios normales?

Usuarios normales

Son usuarios no sofisticados que interactúan con el sistema mediante la innovación de algunos programas de aplicación que se ha escrito previamente.

Un ejemplo de lo encontramos en un cajero bancario que necesita transferir determinada cantidad de dinero de una cuenta a otra invocando un programa llamado transferir, este programa pide al cajero el importe de dinero a transferir, la cuenta de donde se tomara el dinero y la cuenta a donde se transferirá el dinero.

De manera que la interfaz de un usuario normal es una interfaz de formularios (ver figura 3), donde el usuario puede rellenar los campos apropiados del formulario. Este tipo de usuarios también pueden leer informes generados de la base de datos.

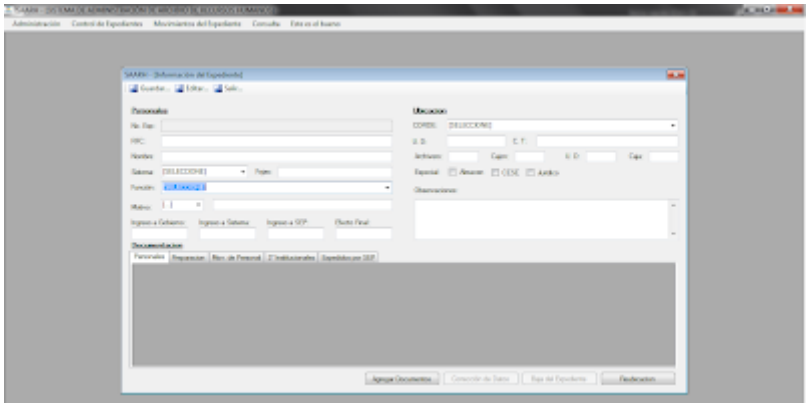

Figura 3.- Interfaz de usuario normal

1.1.2. ¿Qué función desempeñan los programadores de aplicaciones?

Programadores de aplicaciones

Son profesionales informáticos que escriben programas de aplicación.

Los programadores de aplicaciones pueden elegir entre muchas herramientas (ver figura 4) para desarrollar interfaces de usuario.

Las herramientas de desarrollo rápido de aplicaciones (DRA) son herramientas que permiten al programador de aplicaciones construir formularios e informes sin escribir un programa.

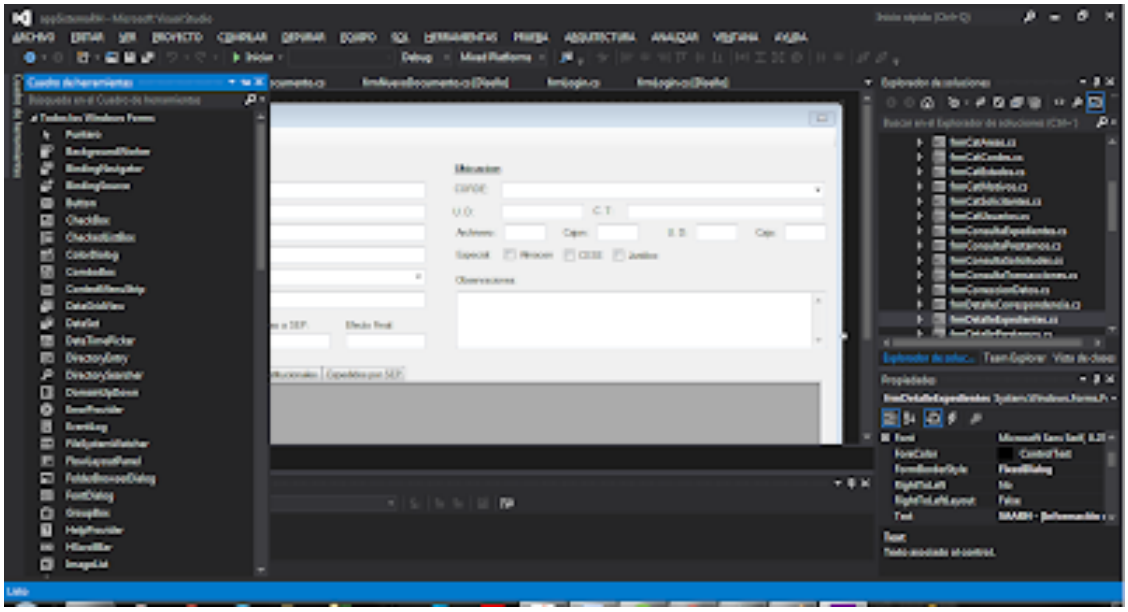

Figura 4.- Interfaz de programador de aplicaciones

Hay también tipos especiales de lenguajes de programación que combinan estructuras de control imperativo (por ejemplo, para bucles for, bucles while e instrucciones if then-else) con instrucciones del lenguaje de manipulación de datos.

Estos lenguajes, llamados a veces lenguajes de cuarta generación, a menudo incluyen características especiales para facilitar la generación de formularios y la presentación de datos en pantalla.

La mayoría de los sistemas de bases de datos comerciales incluyen un lenguaje de cuarta generación.

1.1.2. ¿Qué función desempeñan los usuarios sofisticados?

Interactúan con el sistema sin programas escritos. En su lugar, ellos forman sus consultas en un lenguaje de consulta de bases de datos (ver figura 5).

Cada una de estas consultas se envía al procesador de consultas, cuya función es transformar instrucciones LMD a instrucciones que el gestor de almacenamiento entienda. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría.

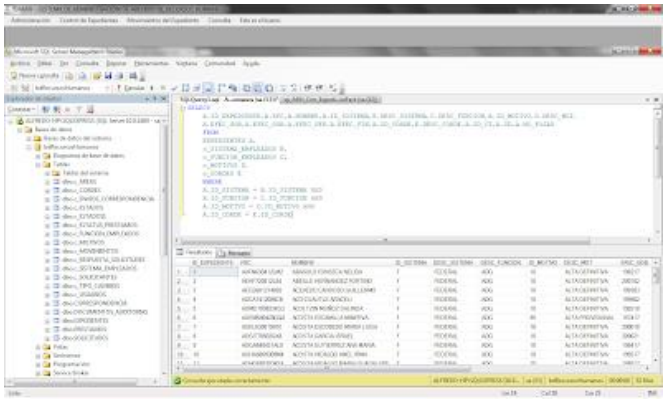

Figura 5.- Interfaz de Usuarios sofisticados

1.1.4. ¿Qué función desempeñan los administradores de la base de datos?

Administrador de la base de datos

Una de las principales razones de usar SGBD es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos. La persona que tiene este control central sobre el sistema se llama administrador de la base de datos (ABD). Las funciones del ABD incluyen las siguientes:

Definición del esquema

El ABD crea el esquema original de la base de datos escribiendo un conjunto de instrucciones de definición de datos en el LDD.

Definición de la estructura y del método de acceso:

Modificación del esquema y de la organización física

Los ABD realizan cambios en el esquema y en la organización física para reflejar las necesidades cambiantes de la organización, o para alterar la organización física para mejorar el rendimiento.

Concesión de autorización para el acceso a los datos.

La concesión de diferentes tipos de autorización permite al administrador de la base de datos determinar a qué partes de la base de datos puede acceder cada usuario.

La información de autorización se mantiene en una estructura del sistema especial que el sistema de base de datos consulta cuando se intenta el acceso a los datos en el sistema.

Mantenimiento rutinario

Algunos ejemplos de actividades rutinarias de mantenimiento del administrador de la base de datos son:

Copia de seguridad periódica de la base de datos, bien sobre cinta o sobre servidores remotos, para prevenir la perdida de datos en caso de desastres como inundaciones.
Supervisión de los trabajos que se ejecuten en la base de datos y asegurarse de que el rendimiento no se degrada por tareas muy costosas iniciadas por algunos usuarios.

1.2. Creación de usuarios

Si se crea un usuario, se le concede acceso a una base de datos, pero no se le concede ningún acceso automáticamente a los objetos de una base de datos. Después de crear un usuario, las acciones habituales son agregar usuarios a los roles de base de datos que tienen permiso para acceder a los objetos de base de datos, o bien conceder permisos de objeto al usuario.

1.2.1. Sintaxis para crear, modificar y borrar a un usuario.

Crear

Usuarios basados en inicios de sesión en la base de datos principal : se trata del tipo de usuario más habitual.
Usuario basado en un inicio de sesión basado en un grupo de Windows Active Directory. CREATE USER [Contoso\Fritz];
Usuario basado en un inicio de sesión basado en un grupo de Windows. CREATE USER [Contoso\Sales];
Usuario basado en un inicio de sesión mediante autenticación de SQL Server. CREATE USER Mary;
Usuarios que se autentican en la base de datos : se recomienda para ayudar a que la base de datos sea más portable.
Siempre se admite en SQL Database. Solo se admite en una base de datos independiente en SQL Server.
Usuario basado en un usuario de Windows sin inicio de sesión. CREATE USER [Contoso\Fritz];

Usuario basado en un grupo de Windows sin inicio de sesión. CREATE USER [Contoso\Sales];

Usuario de SQL Database o Azure Synapse Analytics (SQL Data Warehouse) basado en un usuario de Azure Active Directory. CREATE USER [Fritz@contoso.com] FROM EXTERNAL PROVIDER;

Usuario de base de datos independiente con contraseña. (No está disponible en Azure Synapse Analytics (SQL Data Warehouse)). CREATE USER Mary WITH PASSWORD = '********';
Usuarios basados en entidades de seguridad de Windows que conectan a través de inicios de sesión de grupo de Windows
Usuario basado en un usuario de Windows sin inicio de sesión, pero que se puede conectar a Motor de base de datos mediante la pertenencia a un grupo de Windows. CREATE USER [Contoso\Fritz];

Usuario basado en un grupo de Windows sin inicio de sesión, pero que se puede conectar a Motor de base de datos mediante la pertenencia a un grupo distinto de Windows. CREATE USER [Contoso\Fritz];

Usuarios que no se pueden autenticar : estos usuarios no pueden iniciar sesión en SQL Server ni en SQL Database.

Usuario sin inicio de sesión. No puede iniciar sesión, pero se le pueden conceder permisos. CREATE USER CustomApp WITHOUT LOGIN;
Usuario basado en un certificado. No puede iniciar sesión, pero se le pueden conceder permisos y puede firmar módulos. CREATE USER TestProcess FOR CERTIFICATE CarnationProduction50;
Usuario basado en una clave asimétrica. No puede iniciar sesión, pero se le pueden conceder permisos y puede firmar módulos. CREATE User TestProcess FROM ASYMMETRIC KEY PacificSales09;

```
CREATE USER user_name
    [
        { FOR | FROM } LOGIN login_name
    ]
    [ WITH <limited_options_list> [ ,... ] ]
[ ; ]

-- Users that authenticate at the database
CREATE USER
    {
      windows_principal [ WITH <options_list> [ ,... ] ]

    | user_name WITH PASSWORD = 'password' [ , <options_list> [ ,... ]
    | Azure_Active_Directory_principal FROM EXTERNAL PROVIDER
    }

 [ ; ]

-- Users based on Windows principals that connect through Windows group
logins
CREATE USER
    {
          windows_principal [ { FOR | FROM } LOGIN windows_principal ]
        | user_name { FOR | FROM } LOGIN windows_principal
}
    [ WITH <limited_options_list> [ ,... ] ]
[ ; ]

-- Users that cannot authenticate
CREATE USER user_name
    {
         WITHOUT LOGIN [ WITH <limited_options_list> [ ,... ] ]
       | { FOR | FROM } CERTIFICATE cert_name
       | { FOR | FROM } ASYMMETRIC KEY asym_key_name
    }
 [ ; ]

<options_list> ::=
      DEFAULT_SCHEMA = schema_name
    | DEFAULT_LANGUAGE = { NONE | lcid | language name | language alias }
    | SID = sid
    | ALLOW_ENCRYPTED_VALUE_MODIFICATIONS = [ ON | OFF ] ]

<limited_options_list> ::=
```

```
      DEFAULT_SCHEMA = schema_name ]
    | ALLOW_ENCRYPTED_VALUE_MODIFICATIONS = [ ON | OFF ] ]

-- SQL Database syntax when connected to a federation member
CREATE USER user_name
[;]

-- Syntax for users based on Azure AD logins for Azure SQL Managed
Instance
CREATE USER user_name
    [   { FOR | FROM } LOGIN login_name  ]
    | FROM EXTERNAL PROVIDER
    [ WITH <limited_options_list> [ ,... ] ]
[ ; ]

<limited_options_list> ::=
      DEFAULT_SCHEMA = schema_name
    | DEFAULT_LANGUAGE = { NONE | lcid | language name | language alias }
    | ALLOW_ENCRYPTED_VALUE_MODIFICATIONS = [ ON | OFF ] ]
```

Modificar.

Argumentos
userName especifica el nombre por el que se identifica al usuario en esta base de datos.
LOGIN = loginName reasigna un usuario a otro inicio de sesión cambiando el identificador de seguridad (SID) del usuario para que coincida con el SID de inicio de sesión.
NAME = newUserName especifica el nuevo nombre de este usuario. newUserName no debe existir en la base de datos actual.
DEFAULT_SCHEMA = { schemaName | NULL } especifica el primer esquema donde buscará el servidor cuando resuelva los nombres de objetos de este usuario. El establecimiento del esquema predeterminado en NULL quita un esquema predeterminado de un grupo de Windows. La opción NULL no se puede utilizar con un usuario de Windows.
PASSWORD = ' password ' Se aplica a : SQL Server 2012 (11.x) y posterior, SQL Database.
Especifica la contraseña del usuario que se está cambiando. En las contraseñas se distingue entre mayúsculas y minúsculas.
Nota

Esta opción solo está disponible para los usuarios contenidos. Para más información, vea Bases de datos independientes y sp_migrate_user_to_contained (Transact-SQL).
OLD_PASSWORD = 'oldpassword' Se aplica a : SQL Server 2012 (11.x) y posterior, SQL Database.
La contraseña de usuario actual que se reemplazará por ' password '. En las contraseñas se distingue entre mayúsculas y minúsculas. Para cambiar una contraseña se pide OLD_PASSWORD , a menos que tenga el permiso ALTER ANY USER . Al pedir que se especifique OLD_PASSWORD , se impide que los usuarios con el permiso IMPERSONATION puedan cambiar la contraseña.

Nota

Esta opción solo está disponible para los usuarios contenidos.
DEFAULT_LANGUAGE = { NONE | <lcid> | <language name> | <language alias> } Se aplica a : SQL Server 2012 (11.x) y versiones posteriores.
Especifica el idioma predeterminado que debe asignarse al usuario. Si esta opción se establece en NONE, el idioma predeterminado se establece en el de la base de datos. Si el idioma predeterminado de la base de datos se cambia más tarde, el idioma predeterminado del usuario no se modificará. DEFAULT_LANGUAGE puede ser el identificador local (lcid), el nombre del idioma o el alias del idioma.
Nota

Esta opción solo se puede especificar en una base de datos independiente y solo para los usuarios independientes.
ALLOW_ENCRYPTED_VALUE_MODIFICATIONS = [ON | OFF] Se aplica a : SQL Server 2016 (13.x) y posterior, SQL Database.
Suprime las comprobaciones de metadatos criptográficos en el servidor en operaciones de copia masiva. De esta manera, el usuario puede copiar los datos de forma masiva entre tablas o bases de datos, sin descifrar los datos. El valor predeterminado es OFF.

```
ALTER USER userName
 WITH <set_item> [ ,...n ]
[;]

<set_item> ::=
NAME = newUserName
| DEFAULT_SCHEMA = { schemaName | NULL }
| LOGIN = loginName
| PASSWORD = 'password' [ OLD_PASSWORD = 'oldpassword' ]
| DEFAULT_LANGUAGE = { NONE | <lcid> | <language name> | <language alias>
}
| ALLOW_ENCRYPTED_VALUE_MODIFICATIONS = [ ON | OFF ]
```

Borrar.

IF EXISTS
Se aplica a: SQL Server (desde SQL Server 2016 (13.x) hasta la versión actual, SQL Database).
Quita el usuario condicionalmente solo si ya existe.
user_name
Especifica el nombre por el que se identifica al usuario en esta base de datos.
Observaciones
Los usuarios que poseen elementos protegibles no pueden quitarse de la base de datos. Para poder quitar un usuario de la base de datos que posea elementos protegibles, primero debe quitar o transferir la propiedad de esos elementos protegibles.
El usuario guest no puede quitarse, pero puede deshabilitarse si revoca su permiso CONNECT; para ello, ejecute REVOKE CONNECT FROM GUEST en cualquier base de datos que no sea master o tempdb.

Sintaxis

```
syntaxsql

-- Syntax for SQL Server and Azure SQL Database

DROP USER [ IF EXISTS ] user_name
```

```
syntaxsql

-- Syntax for Azure Synapse Analytics and Parallel Data Warehouse

DROP USER user_name
```

Ejemplos

En este ejemplo se quita el usuario `AbolrousHazem` de la base de datos `AdventureWorks2012`.

```
SQL

DROP USER AbolrousHazem;
GO
```

1.2.2. Sintaxis para visualizar a todos los usuarios de una base de datos.

Al instalar MySQL en Linux, el primer usuario que se creará es el usuario raíz (o usuario root), quien es el administrador de MySQL. El usuario root tiene permiso para hacer todo en la base de datos MySQL, por lo que no es conveniente que otras personas accedan a tus bases de datos utilizando esta cuenta.

Por otro lado, los piratas informáticos siempre intentan iniciar sesión como usuario root para robar la información alojada. O peor, destruir el servicio y los datos asociados.

Con esto en mente, lo ideal es que el administrador del sistema cree usuarios con permisos específicos para las bases de datos y, a su vez, para las tablas. De esta forma, si la seguridad de ese usuario se ve comprometida, el impacto es mínimo y/o manejable.

Escribe tu contraseña de root de MySQL

Luego, tendrás que escribir tu contraseña de root de MySQL. Debe ser diferente de la contraseña de root del sistema.

Una vez estés en la consola de MySQL como usuario root, podrás ejecutar oraciones y comandos.

Mostrar usuarios de MySQL

Ahora podrás enumerar todos los usuarios creados en MySQL a través del siguiente comando MySQL:

```
MYSQL> SELECT USER FROM MYSQL.USER;
```

Como resultado, verás todos los usuarios que se han creado en MySQL.

1.3. Privilegios

Un privilegio es un derecho para ejecutar un tipo particular de sentencia ó para acceder a un objeto de otro usuario.

Una vez creados los usuarios será necesario dotarlos de privilegios para que puedan realizar operaciones específicas en la base de datos.

1.3.1. ¿Qué son los privilegios a usuarios en bases de datos?

Para poder establecer permisos, las siguiente sentencias deben de ejecutarse utilizando el usuario root.

Si queremos que el nuevo usuario tenga permisos de administrador (Todos los permisos), debemos de ejecutar la siguiente sentencia.

GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost';
Los asteriscos indican que los permisos serán asignados a todas las bases de datos y a todas las tablas (primer asteriscos bases de datos, segundo asterisco tablas).

Si queremos asignar permisos para ciertas acciones, la sentencia quedaría de la siguiente manera. Reemplazamos ALL PRIVILEGES y colocamos las acciones que queremos asignar.

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP
    -> ON codigofacilito.*
    -> TO 'nombre_usuario'@'localhost';
```

1.3.2. ¿Qué son los privilegios de sistema?

Privilegios del Sistema (System privileges) permisos a niveles de la base de datos como pueden ser conexión a la base de datos, creación de usuarios, limitar cuentas.

Privilegios del sistema
ALTER ANY INDEX, ALTER ANY ROLE, ALTER ANY TABLE ,ALTER DATABASE, ALTER PROFILE, ALTER USER, BACKUP ANY TABLE, BECOME USER, COMMENT ANY TABLE, CREATE PROFILE, CREATE ROLE, CREATE TABLE, CREATE USER, CREATE VIEW, DELETE ANY TABLE, DROP ANY INDEX, DROP ANY ROLE, DROP ANY VIEW, DROP PROFILE, DROP USER, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, FORCE ANY TRANSACTION, FORCE TRANSACTION, GRANT ANY PRIVILEGE, GRANT ANY ROLE, INSERT ANY TABLE, LOCK ANY TABLE, UPDATE ANY TABLE

1.3.3. ¿Qué son los privilegios sobre objetos?

Privilegios sobre Objetos (Object privileges) vistas, tablas, secuencias, procedimientos, paquetes.

Privilegios sobre objetos
ALTER, EXECUTE, INDEX, INSERT, READ, REFERENCES, SELECT, UPDATE, ALL ó ALL PRIVILEGES

1.3.4. ¿Para qué se utiliza el comando GRANT? (Agrega su sintaxis básica)

Comando GRANT

Se utiliza para crear usuarios y concederle privilegios. La sintaxis general del comando GRANT es la siguiente:

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON \
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [\]
        [CIPHER 'cipher' [AND]]
        [ISSUER 'issuer' [AND]]
        [SUBJECT 'subject']]
    [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR count |
                          MAX_UPDATES_PER_HOUR count |
                          MAX_CONNECTIONS_PER_HOUR count |
                          MAX_USER_CONNECTIONS count]]
```

1.3.5. ¿Para qué se utiliza el comando REVOKE? (Agrega su sintaxis básica)

Comando REVOKE

Es la cláusula predefinida para la remoción de un privilegio. Se utiliza la siguiente sentencia para remover un privilegio a un usuario:

- **REVOKE privilegio FROM usuario;**

1.4. Roles

Función que una persona desempeña en un lugar o en una situación.
Un rol de MySQL es una colección de privilegios con nombre. Al igual que las cuentas de usuario, los roles pueden tener privilegios otorgados y revocados.

1.4.1. ¿Qué son los roles dentro de una base de datos? (Agrega su sintaxis básica).

A una cuenta de usuario se le pueden otorgar roles, lo que otorga a la cuenta los privilegios asociados con cada rol. Esto permite la asignación de conjuntos de privilegios a las cuentas y proporciona una alternativa conveniente a la concesión de privilegios individuales, tanto para conceptualizar las asignaciones de privilegios deseados como para implementarlos.

La siguiente lista resume las capacidades de administración de roles proporcionadas por MySQL:

CREATE ROLEy DROP ROLEcrear y eliminar roles.

GRANTy REVOKEasignar privilegios para revocar privilegios de cuentas y roles de usuario.

SHOW GRANTS muestra las asignaciones de roles y privilegios para las cuentas de usuario y los roles.

SET DEFAULT ROLE especifica qué roles de cuenta están activos de forma predeterminada.

SET ROLE cambia los roles activos dentro de la sesión actual.

La CURRENT_ROLE()función muestra los roles activos dentro de la sesión actual.

Las variables del sistema mandatory_rolesy activate_all_roles_on_loginpermiten definir roles obligatorios y la activación automática de roles otorgados cuando los usuarios inician sesión en el servidor.

Para obtener descripciones de las declaraciones de manipulación de roles individuales (incluidos los privilegios necesarios para utilizarlas), consulte la Sección 13.7.1, “Declaraciones de administración de cuentas” . La siguiente

discusión proporciona ejemplos de uso de roles. A menos que se especifique lo contrario, las declaraciones SQL que se muestran aquí deben ejecutarse utilizando una cuenta MySQL con suficientes privilegios administrativos, como la rootcuenta.

Crear roles y otorgarles privilegios

Considere este escenario:

- Una aplicación usa una base de datos llamada app_db.
- Asociadas con la aplicación, puede haber cuentas para los desarrolladores que crean y mantienen la aplicación y para los usuarios que interactúan con ella.

- Los desarrolladores necesitan acceso completo a la base de datos. Algunos usuarios solo necesitan acceso de lectura, otros necesitan acceso de lectura / escritura.

Para evitar otorgar privilegios individualmente a posiblemente muchas cuentas de usuario, cree roles como nombres para los conjuntos de privilegios requeridos. Esto facilita la concesión de los privilegios necesarios a las cuentas de usuario mediante la concesión de los roles adecuados.

Para crear los roles, use la CREATE ROLEdeclaración:

```
CREATE ROLE 'app_developer', 'app_read', 'app_write';
```

Los nombres de roles son muy parecidos a los nombres de cuentas de usuario y constan de una parte de usuario y una parte de host en formato. La parte del host, si se omite, toma el valor predeterminado . Las partes del usuario y del host pueden estar sin comillas a menos que contengan caracteres especiales como o . A diferencia de los nombres de cuenta, la parte de usuario de los nombres de funciones no puede estar en blanco. Para obtener información adicional, consulte la Sección 6.2.5, “Especificación de nombres de funciones” . '***user_name***'@'***host_name***''%'-%

Para asignar privilegios a los roles, ejecute GRANTdeclaraciones usando la misma sintaxis que para asignar privilegios a las cuentas de usuario:

```
GRANT ALL ON app_db.* TO 'app_developer';
GRANT SELECT ON app_db.* TO 'app_read';
GRANT INSERT, UPDATE, DELETE ON app_db.* TO 'app_write';
```

Ahora suponga que inicialmente necesita una cuenta de desarrollador, dos cuentas de usuario que necesitan acceso de solo lectura y una cuenta de usuario que necesita acceso de lectura / escritura. Utilice CREATE USERpara crear las cuentas:

```
CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1pass';
CREATE USER 'read_user1'@'localhost' IDENTIFIED BY
'read_user1pass';
CREATE USER 'read_user2'@'localhost' IDENTIFIED BY
'read_user2pass';
CREATE USER 'rw_user1'@'localhost' IDENTIFIED BY 'rw_user1pass';
```

Para asignar a cada cuenta de usuario sus privilegios requeridos, puede usar GRANTdeclaraciones de la misma forma que se muestra, pero eso requiere enumerar privilegios individuales para cada usuario. En su lugar, use una GRANTsintaxis alternativa que permita otorgar roles en lugar de privilegios:

```
GRANT 'app_developer' TO 'dev1'@'localhost';
GRANT 'app_read' TO 'read_user1'@'localhost', 'read_user2'@'localhost';
GRANT 'app_read', 'app_write' TO 'rw_user1'@'localhost';
```

El GRANTextracto de la rw_user1cuenta otorga las funciones de lectura y escritura, que se combinan para proporcionar los privilegios de lectura y escritura necesarios.

La GRANTsintaxis para otorgar roles a una cuenta difiere de la sintaxis para otorgar privilegios: hay una ONcláusula para asignar privilegios, mientras que no hay una ONcláusula para asignar roles. Debido a que las sintaxis son distintas, no puede combinar la asignación de privilegios y roles en la misma declaración. (Está permitido asignar tanto privilegios como roles a una cuenta, pero debe usar GRANT declaraciones separadas , cada una con la sintaxis apropiada para lo que se va a otorgar). A partir de MySQL 8.0.16, los roles no se pueden otorgar a usuarios anónimos.

Una función cuando se crea está bloqueada, no tiene contraseña y se le asigna el complemento de autenticación predeterminado. (Estos atributos de rol se pueden cambiar más adelante con la ALTER USER declaración, por usuarios que tienen el CREATE USERprivilegio global).

Mientras está bloqueado, un rol no se puede usar para autenticarse en el servidor. Si está desbloqueado, se puede usar un rol para autenticarse. Esto se debe a que los roles y los usuarios son identificadores de autorización con mucho en común y poco para distinguirlos. Consulte también Intercambiabilidad de roles y usuarios .

Definición de roles obligatorios

Es posible especificar roles como obligatorios nombrándolos en el valor de la mandatory_rolesvariable del sistema. El servidor trata un rol obligatorio como otorgado a todos los usuarios, por lo que no es necesario otorgarlo explícitamente a ninguna cuenta.

Para especificar roles obligatorios al inicio del servidor, defina mandatory_rolesen su my.cnfarchivo de servidor :

```
[mysqld]
mandatory_roles='role1,role2@localhost,r3@%.example.com'
```

Para establecer y persistir mandatory_rolesen tiempo de ejecución, use una declaración como esta:

```
SET PERSIST mandatory_roles =
'role1,role2@localhost,r3@%.example.com';
```

SET PERSISTestablece el valor de la instancia de MySQL en ejecución. También guarda el valor, lo que hace que se transfiera a reinicios posteriores del servidor. Para cambiar el valor de la instancia de MySQL en ejecución sin que se transfiera a reinicios posteriores, use la GLOBALpalabra clave en lugar de PERSIST. Consulte la Sección 13.7.6.1, “Sintaxis SET para la asignación de variables” .

La configuración mandatory_roles requiere el ROLE_ADMIN privilegio, además del SYSTEM_VARIABLES_ADMINprivilegio (o el SUPER privilegio obsoleto) que normalmente se requiere para configurar una variable de sistema global.

Los roles obligatorios, como los roles otorgados explícitamente, no entran en vigencia hasta que se activan (consulte Activación de roles). En el momento del inicio de sesión, la activación de roles se produce para todos los roles otorgados si la activate_all_roles_on_login variable del sistema está habilitada, o para los roles que se establecen como roles predeterminados de lo contrario. En tiempo de ejecución, SET ROLEactiva roles.
Los roles nombrados en el valor de mandatory_rolesno se pueden revocar REVOKEni eliminar con DROP ROLEo DROP USER.
Para evitar que las sesiones se conviertan en sesiones del sistema de forma predeterminada, un rol que tenga el SYSTEM_USER privilegio no se puede incluir en el valor de la mandatory_rolesvariable del sistema:

- Si mandatory_rolesse le asigna un rol en el inicio que tiene el SYSTEM_USERprivilegio, el servidor escribe un mensaje en el registro de errores y sale.
- Si mandatory_rolesse le asigna un rol en tiempo de ejecución que tiene el SYSTEM_USERprivilegio, se produce un error y el mandatory_rolesvalor permanece sin cambios.

Si un rol nombrado en mandatory_rolesno está presente en la mysql.usertabla del sistema, el rol no se otorga a los usuarios. Cuando el servidor intenta la activación de roles para un usuario, no trata el rol inexistente como obligatorio y escribe una advertencia en el registro de errores. Si el rol se crea más tarde y, por lo tanto, se vuelve válido, FLUSH PRIVILEGESpuede ser necesario que el servidor lo trate como obligatorio.
SHOW GRANTSmuestra roles obligatorios de acuerdo con las reglas descritas en la Sección 13.7.7.21, “MOSTRAR Declaración de SUBVENCIONES” .

1.5. Transacciones en SQL

Las transacciones de bases de datos permiten agrupar sentencias (por ejemplo SQL) en bloques, que van a ser ejecutados simultáneamente de tal forma que podamos evaluar si alguna de las sentencias ha fallado y de ser así poder deshacer los cambios en el momento sin alterar de forma alguna la base de datos.

Como funcionan las transacciones MySQL ?

Cada transacción puede verse como una función, y dentro de ella puede haber tantas instrucciones como sea necesario.

```
START TRANSACTION;
   UPDATE users SET name='jane' WHERE id=1;
   UPDATE users SET name='jhon' WHERE id=2;
COMMIT; / ROLLBACK;
```

En este caso tenemos dos sentencias, cada una de ellas para editar un registro, MySQL las evalúa y las guarda en un estado temporal y posteriormente son ejecutadas o desechadas con los comandos COMMIT o ROLLBACK.

Transacciones de base de datos con MySQL en PHP

Cuando trabajamos con PHP podemos hacer uso de sus funciones para implementar las transacciones de bases de datos en nuestros proyectos. Veamos un pequeño ejemplo
Primero debemos crear la conexión a la base de datos

```
$conex = new mysqli("localhost", "root", "pass", "database");
```

Ahora digamos que en nuestra base de datos tenemos una tabla llamada «***users***» con varios usuarios registrados. vamos a agregar un nuevo usuario y editar uno existente en la misma transacción

```
$conex->autocommit(false);

$conex->query("INSERT INTO users (name) VALUES ('marcus')");

$conex->query("UPDATE users SET name = 'jane' WHERE 'id' = 39 ");

$conex->commit();
```

Al cambiar el **autocommit** a «**false**» estamos indicando que queremos enviar los cambios de forma manual que es justo lo que hacemos al finall con **commit().**

1.5.1. ¿Para qué se utilizan las transacciones en SQL?

Por regla general en un sistema de base de datos todas las operaciones relacionadas entre sí que se ejecuten dentro un mismo flujo lógico de trabajo, deben ejecutarse en bloque. De esta manera si todas funcionan la operación conjunta de bloque tiene éxito, pero si falla cualquiera de ellas, deberán retrocederse todas las anteriores que ya se hayan realizado. De esta forma evitamos que el sistema de datos quede en un estado incongruente.
Para permitir que una Web tenga interacción con sus visitantes, permitir el registro de usuarios, dejar comentarios, etc.; son necesarias las bases de datos. Una base de datos consta de cinco parámetros básicos de conexión, y son los siguientes:

Hots: es la ubicación de la base de datos.
Usuario: es el usuario de la base de datos.
Clave: es la clave de la base de datos.
Nombre de la base de datos: es el nombre de la base de datos.
Nombre de la tabla: es el nombre de la tabla con que queremos conectar.

Sobre las bases de datos es importante destacar que guardan la información organizada en tablas, y las tablas guardan la información en un sistema referencial formado por filas y columnas.

Una típica base de datos de una Web sencilla puede tener una tabla donde se guardan los datos de acceso de cada usuario, con el nombre de usuario, el email, el sexo, la fecha de registro, etc. Para que estos usuarios puedan dejar sus comentarios tendríamos que crear una nueva tabla en la base de datos, que recogiese por ejemplo, el nombre de usuario y el comentario. Todo esto lo veremos detalladamente más adelante.

1.5.2. Propiedades de las transacciones.

ACID: Propiedades de las transacciones

Las transacciones tienen las siguientes cuatro propiedades estándar

Atomicidad: asegura que todas las operaciones dentro de la unidad de trabajo se completen con éxito; de lo contrario, la transacción se cancela en el punto de falla y las operaciones anteriores se devuelven a su estado anterior.
Coherencia: garantiza que la base de datos cambie correctamente los estados en una transacción confirmada con éxito.
Aislamiento: permite que las transacciones operen independientemente y transparentes entre sí.
Durabilidad: asegura que el resultado o efecto de una transacción confirmada persista en caso de una falla del sistema.
Las transacciones comienzan con la instrucción START TRANSACTION o BEGIN WORK y terminan con una ROLLBACK COMMIT o ROLLBACK . Los comandos SQL entre las instrucciones de inicio y finalización forman la mayor parte de la transacción.

START TRANSACTION;
SET @transAmt = '500';
SELECT @availableAmt:=ledgerAmt FROM accTable WHERE customerId=1 FOR UPDATE;
UPDATE accTable SET ledgerAmt=ledgerAmt-@transAmt WHERE customerId=1;
UPDATE accTable SET ledgerAmt=ledgerAmt+@transAmt WHERE customerId=2;
COMMIT;
Con START TRANSACTION , la confirmación automática permanece deshabilitada hasta que finalice la transacción con COMMIT o ROLLBACK . El modo de confirmación automática vuelve a su estado anterior.

La FOR UPDATE indica (y bloquea) la (s) fila (s) durante la transacción.

Si bien la transacción permanece sin confirmar, esta transacción no estará disponible para otros usuarios.

Procedimientos generales involucrados en la transacción.

Comience la transacción emitiendo el comando SQL BEGIN WORK o START TRANSACTION .
Ejecute todas sus declaraciones SQL.
Compruebe si todo se ejecuta de acuerdo a sus requerimientos.
Si es así, emita el comando COMMIT ; de lo contrario, emita un comando ROLLBACK para revertir todo al estado anterior.
Compruebe si hay errores incluso después de COMMIT si está utilizando, o podría usar, Galera / PXC.

Conclusión.

En conclusión, en esta actividad los diferentes tipos de acceso a una base de datos asi como los distintos niveles de usuarios y como se pueden dividir en roles las personas encargadas de las bases de datos además a su vez estas tienen distintos privilegios de acceso, en lel desarrollo de la actividad también se explica sobre la creación de nuevos usuarios en la base de datos.

Referencias.

Alumnos del Tecnológico. (s. f.). *3.3 Privilegios a usuarios - 2016_08_TBD_8*. Tecnologico de mexico. Recuperado 20 de noviembre de 2020, de https://sites.google.com/a/tectijuana.edu.mx/2016_08_tbd_8/3-lenguaje-de-manipulacion-de-datos-dml/3-2-1-recuperacion-de-datos

Luis, V. (2019, 6 noviembre). *CREATE USER (Transact-SQL) - SQL Server*. Microsoft Docs. https://docs.microsoft.com/es-es/sql/t-sql/statements/create-user-transact-sql?view=sql-server-ver15

Microsoft. (2019, 28 enero). *Acceso de usuario contenido a las bases de datos independientes - SQL Server*. Microsoft Docs. https://docs.microsoft.com/es-es/sql/relational-databases/security/contained-database-users-making-your-database-portable?view=sql-server-ver15

Parada, V. (1997, 27 mayo). *Acceso a Bases de Datos*. © 1997 Victor Parada. http://www.parada.cl/memoria/doc_3_4.html

Styde, J. (s. f.). *Transacciones de bases de datos MySQL en PHP*. Styde.net. Recuperado 20 de noviembre de 2020, de https://styde.net/transacciones-de-bases-de-datos-mysql-en-php/#:%7E:text=Las%20transacciones%20de%20bases%20de,alguna%20la%20base%20de%20datos.

Tipos de Usuarios de la Base de Datos. (2012, 12 marzo). Bases de Datos. https://uvfdatabases.wordpress.com/2009/02/07/tipos-de-usuarios-de-la-base-de-datos/

Usuarios y administradores de una base de datos - Base de Datos. (s. f.). Base de datos. Recuperado 18 de noviembre de 2020, de https://sites.google.com/site/basdededatosrelacionales/home/contenido/subtema-1/usuarios-y-administradores-de-una-base-de-datos