REPORTE PARA MANEJO DE BASE DE DATOS II

fES ARAGON UNAM | BASE DE DATOS II

pOR: Jared hIDALGO LARA

2023

Contenido

[INTRODUCCION 3](#_Toc133260567)

[1 RESPLADO DE BASES DE DATOS 4](#_Toc133260568)

[1.1 INTRODUCCION AL RESPALDO DE BASES DE DATOS 4](#_Toc133260569)

[1.2 IMPORTANCIA DEL RESPALDO DE BASES DE DATOS 5](#_Toc133260570)

[1.3 TIPOS DE RESLPADO DE BASES DE DATOS 5](#_Toc133260571)

[1.3.1 RESPALDO COMPLETO 5](#_Toc133260572)

[1.3.2 RESPLADO DIFERENCIAL 6](#_Toc133260573)

[1.3.3 RESPALDO INCREMENTAL 6](#_Toc133260574)

[1.3.4 RESPALDO EN LINEA 6](#_Toc133260575)

[1.4 HERRAMIENTAS PARA EL RESPALDO DE LA BASE DE DATOS 7](#_Toc133260576)

[1.4.1 Herramientas nativas del sistema de gestión de base de datos (DBMS) 7](#_Toc133260577)

[1.4.2 Herramientas de terceros 7](#_Toc133260578)

[1.4.3 Servicios de almacenamiento en la nube 7](#_Toc133260579)

[1.4.4 Copias de seguridad en disco 8](#_Toc133260580)

[1.5 COMO HACER UN RESPLADO DE UNA BASE DE DATOS 8](#_Toc133260581)

[1.6 Recomendaciones y buenas prácticas para el respaldo de bases de datos 12](#_Toc133260582)

[2 Niveles de acceso 14](#_Toc133260583)

[2.1 Introducción a los niveles de acceso en bases de datos 14](#_Toc133260584)

[2.2 Tipos de usuarios en una base de datos 15](#_Toc133260585)

[2.3 Niveles de acceso y permisos en bases de datos 15](#_Toc133260586)

[2.3.1 Niveles De Acceso 15](#_Toc133260587)

[2.3.2 permisos 18](#_Toc133260588)

[2.4 Creación de usuarios y asignación de permisos 19](#_Toc133260589)

[2.5 Mejores prácticas para la gestión de niveles de acceso en bases de datos 20](#_Toc133260590)

[3 PROCEDIMIENTOS ALMACENADOS 22](#_Toc133260591)

[3.1 Introducción a los procedimientos almacenados en bases de datos 22](#_Toc133260592)

[ventajas y desventajas de utilizar procedimientos almacenados en una base de datos. 22](#_Toc133260593)

[3.1.1 Ventajas 23](#_Toc133260594)

[3.1.2 Desventajas 23](#_Toc133260595)

[3.2 Creación de procedimientos almacenados 24](#_Toc133260596)

[3.3 Buenas prácticas para la utilización de procedimientos almacenados en bases de datos 25](#_Toc133260597)

[4 TRIGGERS 27](#_Toc133260598)

[4.1 Introducción a los Triggers en bases de datos 27](#_Toc133260599)

[4.2 Ventajas y desventajas de utilizar Triggers 27](#_Toc133260600)

[4.2.1 Ventajas 28](#_Toc133260601)

[4.2.2 Desventajas 28](#_Toc133260602)

[4.3 Creación de Triggers en bases de datos 28](#_Toc133260603)

[4.3.1 Pasos para crear un trigger 28](#_Toc133260604)

[4.3.2 Consideraciones al crear un trigger 29](#_Toc133260605)

[4.4 Buenas prácticas para la utilización de Triggers en bases de datos 30](#_Toc133260606)

[5 Estructuras y usos de if y case 31](#_Toc133260607)

[5.1 Introducción a las estructuras if y case en SQL 31](#_Toc133260608)

[5.1.1 La estructura if en SQL 31](#_Toc133260609)

[5.1.2 La estructura case en SQL 32](#_Toc133260610)

[5.2 Estructuras if y case en SQL: diferencias y similitudes 32](#_Toc133260611)

[5.2.1 Diferencias entre las estructuras if y case 32](#_Toc133260612)

[5.2.2 Similitudes entre las estructuras if y case 34](#_Toc133260613)

# INTRODUCCION

En la actualidad, el manejo de bases de datos se ha convertido en una herramienta esencial en el mundo de la informática y la tecnología. Las bases de datos permiten almacenar grandes cantidades de información, lo que resulta de gran utilidad para empresas, organizaciones e incluso para usuarios individuales.

El lenguaje SQL (Structured Query Language) se ha consolidado como uno de los principales lenguajes de programación para el manejo de bases de datos. A través de SQL, es posible crear, modificar y consultar bases de datos de manera eficiente y sencilla.

En este reporte, se presentará una guía práctica para el manejo de bases de datos en SQL. A lo largo del reporte, se abordarán diferentes temas, desde la creación de usuarios y la otorgación de permisos, hasta la creación de vistas y el uso de tablas temporales. Cada sección del reporte se enfocará en un tema específico, con el objetivo de facilitar el aprendizaje y el entendimiento del manejo de bases de datos en SQL.

También cabe destacar que para propósitos didácticos para este reporte se trabajara con una Base de Datos llamada “tiendaOnline” sobre la cual se realizaran todos los ejercicios.

Es importante mencionar que este reporte requiere conocimientos básicos previos en bases de datos, además cada sección se explicará detalladamente, con ejemplos y ejercicios prácticos que permitirán al lector poner en práctica lo aprendido.

Este reporte está diseñado para ser utilizado tanto por principiantes como por usuarios más experimentados en el manejo de bases de datos. El objetivo principal es proporcionar una guía útil y accesible para el aprendizaje de SQL y el manejo de bases de datos.

¡Espero que este reporte sea de gran utilidad para todas aquellas personas interesadas en el manejo de bases de datos en SQL!

# RESPLADO DE BASES DE DATOS

## INTRODUCCION AL RESPALDO DE BASES DE DATOS

El respaldo de base de datos es una tarea fundamental en la administración de sistemas de información, ya que las bases de datos siempre han sido elementos críticos para el correcto funcionamiento de una organización.

La pérdida de información puede ser causada por diversos factores, como fallos en el hardware, errores de software, ataques cibernéticos, desastres naturales, entre otros. Por lo tanto, es necesario tomar medidas para asegurar la integridad y disponibilidad de los datos en todo momento, ya que en un mundo interconectado es necesario disponer de la información en todo momento a todas horas.

Por lo tanto, el respaldo de base de datos consiste en realizar una copia de seguridad de la información contenida en la base de datos, con el objetivo de restaurarla en caso de pérdida o daño. Dicha copia puede ser realizada en diferentes medios, como discos duros, cintas magnéticas, dispositivos de almacenamiento en la nube, entre otros. Además, es importante destacar que el respaldo de base de datos no es una tarea de una sola vez, sino que debe realizarse de forma periódica para que de esta manera se asegure la actualización de la información y minimizar el riesgo de pérdida de datos.

El proceso de respaldo de base de datos también debe contemplar aspectos como lo son la seguridad de la información, la verificación de la integridad de la copia de seguridad, la documentación de los procedimientos, entre otros. Cabe resaltar que, es importante considerar que el respaldo de base de datos no es una tarea aislada, sino que debe ser parte de una estrategia de continuidad del negocio, pues de esta manera permite asegurar la disponibilidad de los sistemas y la recuperación de la información en caso de desastres.



En los siguientes subtemas se explicará con más detalle las diferentes técnicas y herramientas que existen para realizar el respaldo de base de datos.

## IMPORTANCIA DEL RESPALDO DE BASES DE DATOS

La importancia del respaldo de una base de datos no puede ser subestimada, ya que las bases de datos son un componente fundamental en el almacenamiento y gestión de información crítica de cualquier organización que las use. Cabe recordar que una base de datos se compone de un conjunto de datos organizados en tablas, que contienen información valiosa para la empresa, como datos de clientes, productos, proveedores, transacciones y registros financieros, entre otros.

Por lo tanto, la pérdida de esta información puede ser catastrófica para cualquier empresa, ya que puede afectar su operatividad, su capacidad para tomar decisiones y su reputación. Pongamos, por ejemplo, en el caso de una empresa que pierde los datos de sus clientes de manera repentina, a causa de esto no podrá llevar a cabo transacciones comerciales, enviar correos electrónicos o realizar análisis de marketing, lo que terminaría afectando gravemente sus ventas y su imagen ante los clientes.

Hay que mencionar, existen diferentes factores que pueden causar la pérdida de datos, como errores humanos, fallos de hardware, virus informáticos, desastres naturales, entre otros. Por lo tanto, es crucial contar con un plan de respaldo de bases de datos que permita proteger la información crítica de la empresa y asegurar su recuperación en caso de alguna eventualidad.

Por lo tanto, el respaldo de bases de datos es una tarea vital para cualquier empresa que maneje información crítica en su operación. Permite proteger los datos de pérdidas catastróficas y asegurar la continuidad de la empresa en caso de algún desastre o fallo técnico. Así que es fundamental contar con un plan de respaldo efectivo que permita restaurar los datos en el menor tiempo posible y minimizar el impacto en la operación empresarial.

## TIPOS DE RESLPADO DE BASES DE DATOS

Los tipos de respaldo de bases de datos son una de las consideraciones más importantes a tener en cuenta cuando se trata de la protección de los datos de una organización. Existen diferentes tipos de respaldo, cada uno de ellos cuenta con sus propias ventajas y desventajas.

### RESPALDO COMPLETO

El primer tipo de respaldo es el respaldo completo, también conocido como backup completo. En este tipo de respaldo implica copiar todos los datos de la base de datos en un archivo de respaldo. Esto significa que se creará una copia de todos los datos de la base de datos, incluyendo tablas, vistas, procedimientos almacenados, funciones y otros objetos de base de datos todo a la vez. El respaldo completo es el tipo más completo de respaldo y es útil para restaurar la base de datos completa en caso de un fallo del sistema, sin embargo, su desventaja es que suele ser el más lento debido a que se debe respaldar la base de datos completa de una sola vez.

### RESPLADO DIFERENCIAL

El siguiente tipo de respaldo del que se hablara es el respaldo diferencial. Este tipo de respaldo copia todos los datos que han cambiado desde el último respaldo completo de la base de datos. Por lo tanto, significa que el tamaño de los archivos de respaldo será menor que el de los respaldos completos, lo que ahorra espacio de almacenamiento y tiempo al realizar la copia. La restauración de un respaldo diferencial puede ser más rápida que la restauración de un respaldo completo, ya que se restaura sólo la información que ha cambiado desde el último respaldo completo.

### RESPALDO INCREMENTAL

El tercer tipo de respaldo a tratar en este reporte es el respaldo incremental. Este tipo de respaldo copia todos los datos que han cambiado desde el último respaldo, sea completo o incremental. Esto significa que se creará una copia de los datos que han cambiado desde el último respaldo, incluyendo nuevos registros y cambios en los registros existentes. El tamaño de los archivos de respaldo será menor que el de los respaldos diferenciales, ya que sólo se copian los datos que han cambiado desde el último respaldo, lo que ahorra espacio de almacenamiento y tiempo de copia. Sin embargo, una desventaja al momento de la restauración de un respaldo incremental es que puede ser más lenta que la restauración de un respaldo diferencial, ya que se deben aplicar múltiples respaldos para recuperar la base de datos completa.

### RESPALDO EN LINEA

El último tipo de respaldo a ver en este reporte es el respaldo en línea. Este tipo de respaldo se realiza mientras la base de datos está en uso, lo que significa que no se requiere detener el servicio para realizar un respaldo completo. El respaldo en línea puede ser más lento que otros tipos de respaldo, pero permite que la base de datos siga en uso mientras se realiza el respaldo lo cual lo hace muy útil en ciertas circunstancias.

Es importante mencionar que cada tipo de respaldo tiene sus propias ventajas y desventajas, así que es importante elegir el tipo de respaldo adecuado según las necesidades de cada organización. Cabe mencionar que un enfoque común es utilizar una combinación de respaldos completos y diferenciales o incrementales para proteger los datos y asegurarse de que se puedan restaurar en caso de una pérdida de datos de esta manera se puede combinar las ventajas que pueden tener cada tipo de respaldo y lograr hacer un “backup” más eficiente de la información.

## HERRAMIENTAS PARA EL RESPALDO DE LA BASE DE DATOS

El proceso de respaldo de la base de datos es crucial para la protección y seguridad de los datos almacenados en un sistema. Así que, para realizar esta tarea, existen varias herramientas que permiten la creación y recuperación de los respaldos de manera rápida y eficiente. A continuación, se describen algunas de las herramientas más comunes para el respaldo de bases de datos.

### Herramientas nativas del sistema de gestión de base de datos (DBMS)

La mayoría de los sistemas de gestión de bases de datos (DBMS) ofrecen herramientas nativas para el respaldo y la recuperación de bases de datos. Estas herramientas suelen incluir utilidades de línea de comandos y herramientas gráficas que permiten la configuración de las opciones de respaldo, tales como la ruta de almacenamiento del archivo, el tipo de archivo y la frecuencia de respaldo.

Algunos ejemplos de herramientas nativas de DBMS son:

* mysqldump para MySQL
* pg\_dump para PostgreSQL
* sqlcmd para Microsoft SQL Server
* Oracle Data Pump para Oracle Database

### Herramientas de terceros

Además de las herramientas nativas de DBMS mencionadas anteriormente, también, existen varias herramientas de terceros que ofrecen características adicionales para el respaldo de bases de datos. Estas herramientas pueden incluir opciones de compresión de datos, cifrado, automatización de tareas, programación de tareas y gestión de múltiples bases de datos, por lo tanto, también suelen ser útiles.

Algunos ejemplos de herramientas de terceros más conocidas son:

* Bacula
* Acronis Backup
* Veeam Backup & Replication
* Commvault

### Servicios de almacenamiento en la nube

Los servicios de almacenamiento en la nube también pueden ser utilizados para el respaldo de bases de datos. Estos servicios ofrecen una manera sencilla y rentable de realizar respaldos automatizados de bases de datos y restaurarlos en caso de una pérdida de datos, cabe mencionar que esta es una de las opciones se ha vuelto de las mas populares en los últimos años en algunos sectores empresariales, dada la facilidad y la accesibilidad a los datos, en especial en el mundo interconectado en el que se vive hoy en día.

Algunos ejemplos de servicios de almacenamiento en la nube que están dentro de los más conocidos son:

* Amazon S3
* Google Cloud Storage
* Microsoft Azure Blob Storage

### Copias de seguridad en disco

Las copias de seguridad en disco son una alternativa al almacenamiento en la nube, permitiendo respaldos de bases de datos en discos duros externos, unidades USB y otros dispositivos de almacenamiento físico. Aunque no es tan rentable como los servicios de almacenamiento en la nube, es una opción viable para aquellos que prefieren tener el control total sobre sus respaldos de bases de datos, es una de las opciones más fiables si se quieren tener la información disponible en la empresa en todo momento.

## COMO HACER UN RESPLADO DE UNA BASE DE DATOS

Una vez que se ha comprendido la importancia y los tipos de respaldo de bases de datos, es importante conocer cómo hacer el respaldo, para este caso se usara como ejemplo una base de datos en MySQL Workbench, ya que esta es una de las herramientas más populares y fáciles de usar para gestionar bases de datos MySQL.

Para hacer el respaldo de una base de datos, en primer lugar, se debe abrir MySQL Workbench y seleccionar la opción "Data Export" en la pestaña "Server" en la pantalla de inicio.

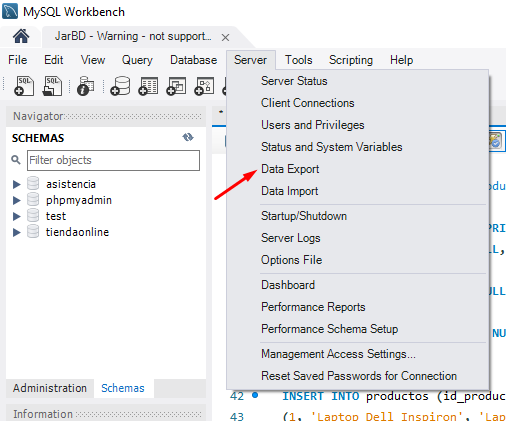


Ilustración 1.5:1

Una vez seleccionada la opción "Data Export", se abrirá una nueva ventana que permitirá elegir la base de datos que se desea respaldar. En esta ventana, se deben seleccionar todas las bases de datos que se desean incluir en el respaldo (en este caso usaremos la base que ya ha sido creada previamente llamada “tiendaOnline” de la cual se hará el respaldo) y, en la parte inferior de la pantalla, elegir el directorio de destino donde se almacenará el archivo de respaldo.

En la sección "Export Options", se deben seleccionar los formatos de archivo de salida. Por defecto, MySQL Workbench genera archivos .sql que incluyen todas las sentencias SQL necesarias para recrear la base de datos y las tablas.

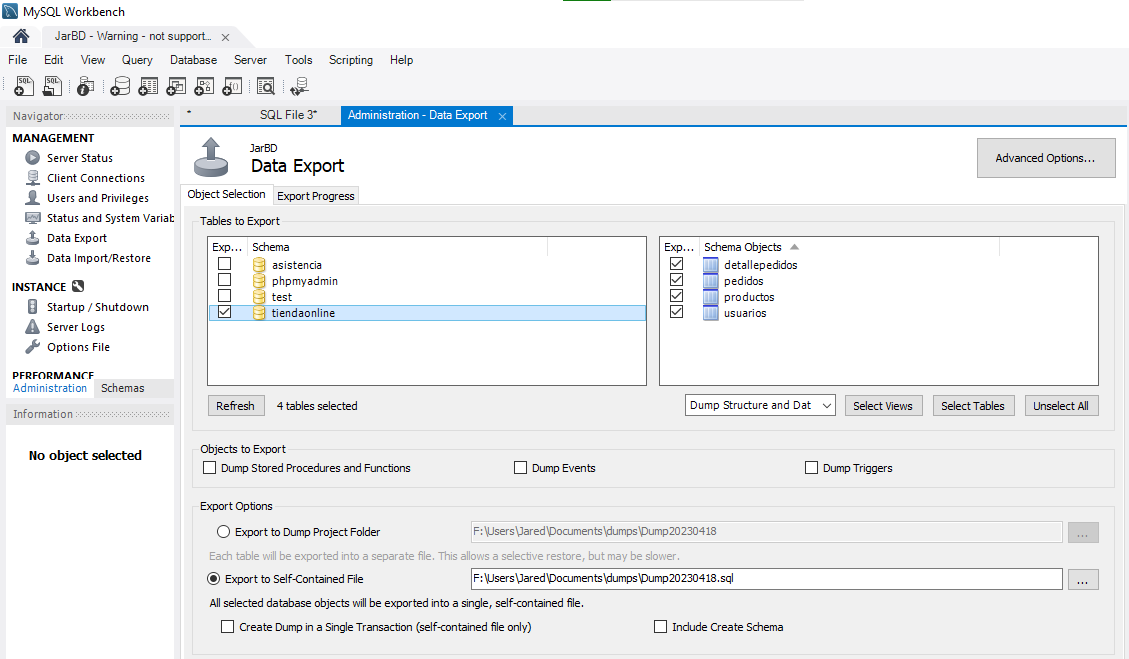
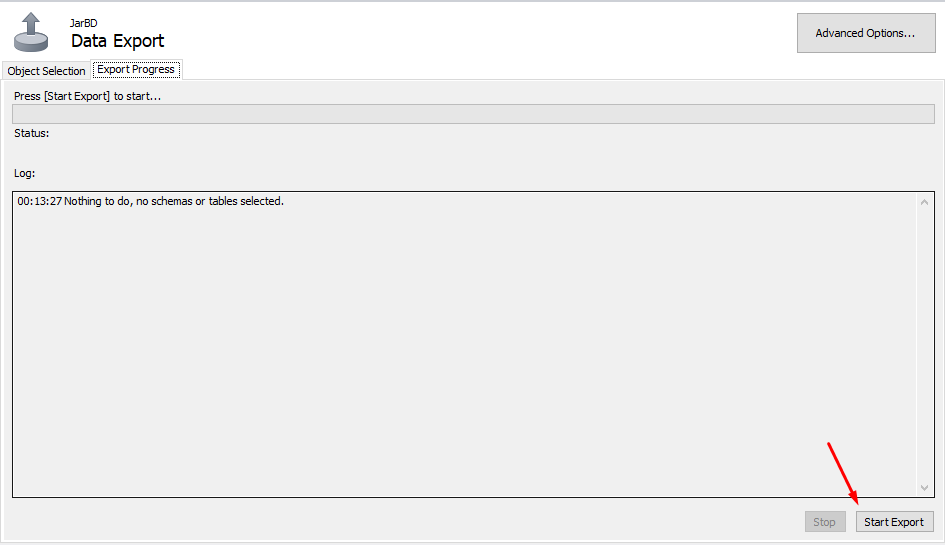


Ilustración 1.5:2

Una vez que se han seleccionado las opciones de exportación, se debe hacer clic en el botón "Start Export" que se encuentra en la pestaña “export Progress” para iniciar el proceso de respaldo. Durante el proceso, se mostrará una barra de progreso que indicará el estado del respaldo.

Es importante tener en cuenta que, dependiendo del tamaño de la base de datos y la cantidad de datos a respaldar, el proceso de exportación puede tardar varios minutos o incluso horas.

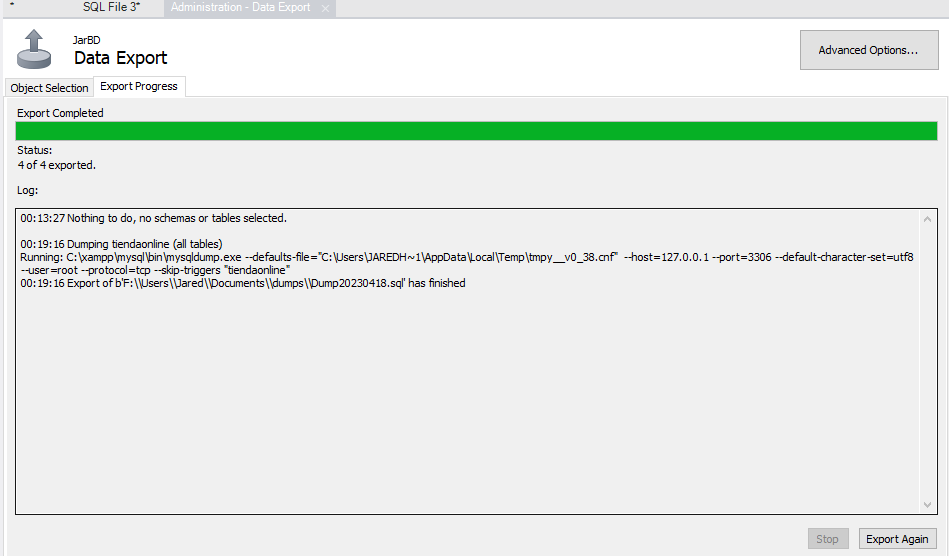


Ilustración 1.5:3

Una vez completado el proceso de respaldo, se debe verificar que el archivo de respaldo se haya generado correctamente y que incluya toda la información necesaria para recuperar la base de datos en caso de una eventualidad.

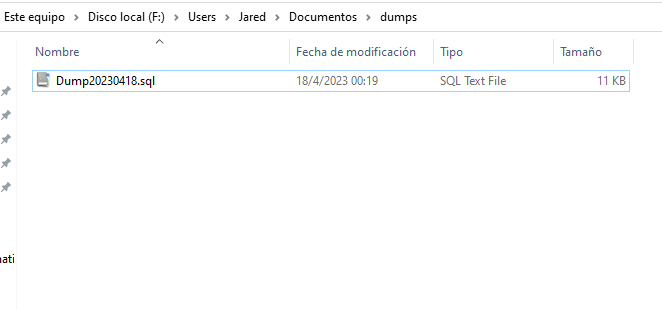


Ilustración 1.5:4

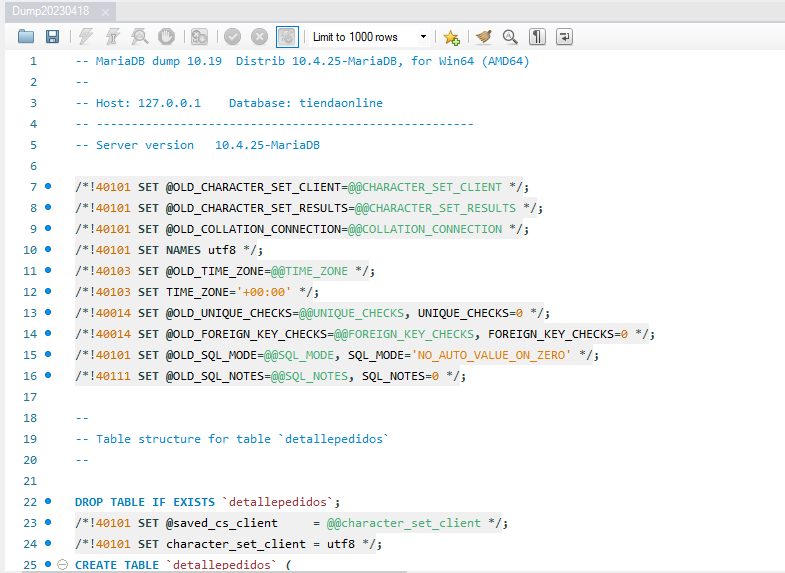


Ilustración 1.5:5

Después de ver este ejemplo, podemos concluir que, hacer el respaldo de una base de datos en MySQL Workbench es un proceso sencillo y rápido que puede ayudar a garantizar la seguridad y disponibilidad de la información en caso de pérdida de datos o fallas en el sistema.

## Recomendaciones y buenas prácticas para el respaldo de bases de datos

Como se acaba de mencionar anteriormente, el respaldo de bases de datos es una tarea crítica para cualquier empresa u organización que maneje información valiosa. Un error en el respaldo puede significar la pérdida de datos importantes, lo que puede resultar en consecuencias costosas para la empresa. Así que para terminar este tema hay que mencionar que, es importante seguir algunas buenas prácticas para asegurar que el respaldo de la base de datos se realice de manera efectiva y segura.

La primera recomendación es hacer respaldos de manera regular. Esto asegura que se tengan versiones actualizadas de la base de datos en caso de cualquier falla o pérdida de datos. La frecuencia del respaldo dependerá del tamaño de la base de datos y de la cantidad de información que se maneje. Algunas empresas pueden necesitar respaldos diarios, mientras que otras pueden hacer respaldos semanales o mensuales.

Cabe recomendar que es importante también verificar la integridad del respaldo. Esto significa comprobar que el archivo de respaldo no esté corrupto o dañado. Esto se puede hacer mediante la realización de pruebas de restauración en un ambiente de pruebas. Es recomendable tener una copia del archivo de respaldo en un lugar seguro y fuera de las instalaciones de la empresa para evitar pérdidas de datos en caso de un desastre natural o un accidente.

Otra buena práctica es asegurar la confidencialidad del respaldo. La información almacenada en el respaldo puede ser tan valiosa como la información almacenada en la base de datos original. Por lo tanto, es importante proteger el archivo de respaldo con medidas de seguridad adecuadas. Esto incluye el cifrado del archivo y la gestión de las credenciales de acceso.

Además, es importante hacer pruebas periódicas de recuperación de datos para verificar que el proceso de restauración sea efectivo. Es recomendable realizar estas pruebas en un ambiente de pruebas para evitar dañar la base de datos original.

Finalmente, es importante mantener el software utilizado para realizar el respaldo actualizado. Esto asegura que el software esté libre de errores y que se estén utilizando las últimas versiones disponibles.

Para terminar, es necesario recordar que, el respaldo de bases de datos es una tarea crítica que debe realizarse de manera efectiva y segura. Siguiendo algunas buenas prácticas, como hacer respaldos regulares, verificar la integridad del respaldo, proteger la confidencialidad del archivo de respaldo, hacer pruebas periódicas de recuperación de datos y mantener el software actualizado, se puede asegurar que el respaldo sea efectivo y que los datos estén seguros, mantén estas practicas y tus datos siempre estarán a salvo.

# Niveles de acceso

## Introducción a los niveles de acceso en bases de datos

Como ya se ha mencionado en muchas ocasiones, las bases de datos son esenciales para el almacenamiento y gestión de la información de una empresa u organización. Por esta razón, es fundamental establecer niveles de acceso adecuados para garantizar la seguridad y privacidad de la información almacenada.

Los niveles de acceso se refieren a los diferentes permisos que se pueden asignar a los usuarios de la base de datos. Cada nivel de acceso se relaciona con diferentes operaciones que los usuarios pueden realizar en la base de datos. Al establecer estos permisos, se puede controlar quién tiene acceso a qué información, así como las acciones que pueden realizar en ella, de esta manera se puede garantizar que solo los usuarios autorizados tengan acceso a ella.

En general, los niveles de acceso se dividen en cuatro categorías: administrador, usuario avanzado, usuario normal y usuario invitado. A continuación, se describen brevemente estas categorías:

* Administrador: este nivel de acceso tiene el control completo de la base de datos. Puede agregar, modificar y eliminar usuarios, así como otorgar permisos a otros usuarios. También tiene acceso a todas las operaciones de la base de datos, incluyendo la capacidad de crear, modificar y eliminar tablas.
* Usuario avanzado: este nivel de acceso tiene permisos para realizar la mayoría de las operaciones en la base de datos, incluyendo la creación y modificación de tablas y la capacidad de ejecutar consultas complejas.
* Usuario normal: este nivel de acceso tiene permisos para realizar operaciones básicas, como insertar, modificar y eliminar registros en las tablas de la base de datos. No puede crear o modificar tablas.
* Usuario invitado: este nivel de acceso se utiliza generalmente para proporcionar acceso de solo lectura a la base de datos. Los usuarios invitados pueden ver los datos, pero no pueden modificarlos.

Es importante mencionar que estos niveles de acceso son solo una guía y pueden variar dependiendo del sistema de gestión de base de datos utilizado. Además, es importante tener en cuenta que los permisos otorgados a un usuario deben ser proporcionales a sus responsabilidades y roles dentro de la organización.

## Tipos de usuarios en una base de datos

Cabe mencionar que, en una base de datos, es común tener varios tipos de usuarios que acceden a ella con diferentes propósitos y niveles de acceso. A continuación, se describirán los tipos más comunes de usuarios en una base de datos:

* Administrador de base de datos (DBA): El administrador de la base de datos es el responsable de la gestión y el mantenimiento de la base de datos. El DBA se encarga de la configuración de la base de datos, la creación y eliminación de usuarios, la asignación de permisos y la monitorización de la seguridad y el rendimiento de la base de datos.
* Desarrollador de aplicaciones: Un desarrollador de aplicaciones es un usuario que crea aplicaciones que interactúan con la base de datos. Los desarrolladores de aplicaciones suelen tener un alto nivel de acceso a la base de datos y pueden crear, actualizar y eliminar tablas, vistas y procedimientos almacenados.
* Usuario final: El usuario final es el que utiliza la aplicación creada por el desarrollador para interactuar con la base de datos. Los usuarios finales tienen un nivel de acceso limitado y solo pueden ver y modificar los datos a los que se les ha otorgado acceso.
* Auditor de seguridad: El auditor de seguridad es un usuario que tiene acceso a la base de datos para realizar auditorías de seguridad. El auditor de seguridad evalúa la seguridad de la base de datos y garantiza que se sigan las políticas de seguridad establecidas.
* Administrador de sistema: El administrador del sistema es el usuario responsable de la gestión de los recursos de hardware y software necesarios para que la base de datos funcione correctamente. El administrador del sistema también se encarga de la copia de seguridad y la recuperación de la base de datos.

Es importante tener en cuenta que cada tipo de usuario debe tener un nivel de acceso adecuado a la base de datos para garantizar la seguridad y la integridad de los datos. Además, es importante limitar el acceso a la base de datos solo a aquellos usuarios que lo necesiten para realizar su trabajo y garantizar que se sigan las políticas de seguridad establecidas, de esta manera se mantiene integra la seguridad de la información.

## Niveles de acceso y permisos en bases de datos

### Niveles De Acceso

Los niveles de acceso son como los controles de seguridad de una base de datos, que controlan lo que los usuarios pueden hacer en la base de datos y en su contenido. Estos niveles van desde "Sin acceso", que impide que un usuario pueda abrir la base de datos, hasta "Gerente", que permite leer, crear y editar la base de datos y su contenido.

Cada usuario puede tener diferentes niveles de acceso asignados, ya sea de forma individual o como parte de un grupo. Si un usuario pertenece a varios grupos con diferentes niveles de acceso, se le otorga el nivel de acceso más alto.

Si un usuario o un grupo tiene diferentes niveles de acceso en la base de datos y en un componente específico de la base de datos, como una lista de lectura o escritura, el nivel de acceso del componente específico tiene prioridad sobre el nivel de acceso del usuario o grupo.

A continuación, se muestra una lista de los diferentes niveles de acceso, desde el más bajo al más alto. Los niveles de acceso más altos tienen todos los privilegios de los niveles de acceso inferiores, lo que significa que un "Autor" puede hacer todo lo que puede hacer un "Depositante" o un "Lector"

#### Gerente

El acceso de “Gerente” permite a los usuarios leer, crear y editar la ACL (Access Controller List) y todos los documentos de las bases de datos, además de poder modificar la configuración de la ACL y eliminar la base de datos. La modificación de la ACL y la supresión de bases de datos son tareas que no permite ningún otro nivel de acceso. Cabe mencionar que este nivel de acceso se suele asignar a administradores de la base de datos de la empresa y dicho nivel de acceso no se recomienda ser otorgado para los usuarios generales.

Es importante mencionar que cada base de datos debe tener, al menos, un “Gerente”. Y por lo general, el nivel de acceso “Gerente” se proporciona en cada base de datos a la persona que ha sido designada como administrador durante el procedimiento de instalación y configuración de la base datos. También es altamente recomendable asignar el acceso de “Gerente” a dos personas, por si una de ellas no estuviese disponible.

#### Diseñador

El acceso de Diseñador permite a los usuarios crear índices de texto completo, modificar todos los elementos de diseño de las bases de datos y leer, crear y editar todos los documentos de la base de datos. Este nivel de acceso es básicamente para programadores y desarrolladores de bases de datos.

Cabe mencionar que la tarea de diseñador de bases de datos se le asigna a la persona responsable de las futuras actualizaciones del diseño.

#### Editor

Este acceso permite a los usuarios leer (salvo que exista un campo de lectores en el formulario), crea y editar todos los documentos de la base de datos, incluidos los que han creado otros usuarios, este nivel de acceso debe otorgársele a Todos los usuarios que estén contemplados con permiso para crear o editar documentos en una base de datos.

#### Autor

El acceso de Autor permite al usuario crear y editar documentos. Sin embargo, para que el “autor” pueda crear documentos, cuando asigne el acceso de autor a un usuario o a un servidor, se deberá especificar también el derecho de crear documentos ya que el acceso de autor no incluye inicialmente la opción de Crear documentos.

Los usuarios con acceso de Autor pueden editar los documentos que ellos mismos han creado, pero no los documentos creados por otros usuarios.

Este tipo de nivel de acceso debe otorgarse a usuarios que necesitan aportar documentos a una base de datos.

Siempre que sea posible, utilice el acceso de autor en lugar del de editor; de esta forma, reducirá los conflictos producidos durante la replicación o durante la grabación que pudieran llegar a presentarse.

#### Lector

El acceso de lector permite al usuario leer documentos en una base de datos, sin embargo, no crear ni editar documentos.

También cabe mencionar que dicho acceso le permitirá leer todos los documentos, salvo que exista un campo de lectores en el formulario. Si fuera así, solo podrá leer los documentos si su nombre está incluido en dicho campo.

Este nivel de permiso de acceso se le otorga a usuarios que solo necesitan leer los documentos de una base de datos pero no deben crear documentos nuevos ni editar los existentes.

#### Depositante

El nivel de acceso de “Depositante” permite a los usuarios crear documentos, pero no ver los documentos de la base de datos, incluidos los documentos creados por el usuario. Este tipo de ACL se utiliza más frecuentemente con los agentes automáticos para grabar documentos en una base de datos de aplicaciones del flujo de trabajo de Domino. por ejemplo, se puede utilizar el nivel de acceso de “Depositante” para una aplicación en la que se efectúen votaciones (ya que evidentemente quien tenga acceso a esta no debe tener ser capaz de ver ninguno de los otros documentos por razones de seguridad).

#### Sin acceso

Este último nivel conocido como “Sin acceso” impide que los usuarios accedan a la base de datos. a menos que el que permita las opciones Leer documentos públicos y Crear documentos públicos, en caso de que se hayan definido.

Este nivel de acceso debe asignarse a usuarios con acceso cancelado, usuarios que no necesitan acceder a la base de datos o usuarios que tienen acceso en circunstancias especiales. De igual manera a, usuarios que no necesitan acceder a la base de datos, pero forman parte de un grupo que sí dispone de ese acceso.

Se recomienda que este nivel de acceso sea el predeterminado para evitar el acceso generalizado a una base de datos con información de naturaleza confidencial o delicada.

### permisos

Los permisos son las operaciones específicas que un usuario puede realizar en una base de datos. Estos permisos pueden ser asignados a los usuarios en diferentes niveles, desde el nivel de la base de datos hasta el nivel de la tabla o incluso el nivel de la columna. Los permisos que pueden ser asignados a los usuarios incluyen los siguientes:

* SELECT: Este permiso permite al usuario leer los datos de una tabla.
* INSERT: Este permiso permite al usuario agregar nuevos datos a una tabla.
* UPDATE: Este permiso permite al usuario modificar los datos existentes en una tabla.
* DELETE: Este permiso permite al usuario eliminar datos de una tabla.
* CREATE: Este permiso permite al usuario crear nuevas tablas en una base de datos.
* DROP: Este permiso permite al usuario eliminar tablas de una base de datos.
* ALTER: Este permiso permite al usuario modificar la estructura de una tabla existente.
* INDEX: Este permiso permite al usuario crear índices en una tabla.
* REFERENCES: Este permiso permite al usuario crear claves foráneas en una tabla.
* ALL: Este permiso permite al usuario realizar todas las operaciones en una base de datos o tabla.

Es importante recordar que asignar permisos a los usuarios de manera indiscriminada puede resultar en vulnerabilidades de seguridad y pérdida de datos. Por lo tanto, se recomienda asignar permisos solo a usuarios específicos que necesiten acceso a datos y solo los permisos necesarios para realizar sus tareas. Además, se debe revisar regularmente los permisos de los usuarios para asegurarse de que sigan siendo apropiados y actualizarlos según sea necesario.

## Creación de usuarios y asignación de permisos

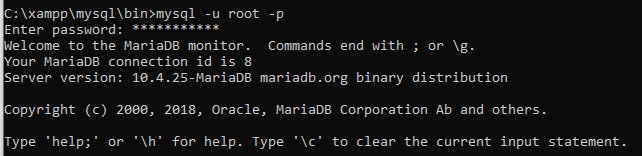
Como ya se ha mencionado antes, en una base de datos, es necesario que los usuarios tengan diferentes niveles de acceso y permisos según sus necesidades y roles en el sistema. Para hacer esto, es necesario crear usuarios y asignarles los permisos adecuados.

Es importante mencionar que la creación de usuarios en una base de datos varía según el sistema de gestión de base de datos utilizado. En el caso de MySQL, primeramente, es necesario utilizar el comando CREATE USER seguido del nombre del usuario y la contraseña. Por ejemplo, para crear un usuario llamado "user1" con la contraseña "password1", el comando sería el siguiente:

CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';

Es importante tener en cuenta que se debe especificar la ubicación del usuario, en este caso se ha establecido en "localhost".

para ello primero se debe abrir una terminal en la carpeta donde se encuentre instalado sql, para posteriormente ingresar como usuario root desde ahí se puede proceder a crear el usuario

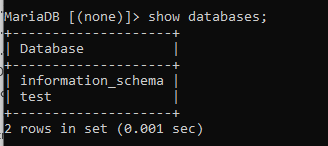


Una vez ingresado se puede crear el usuario.



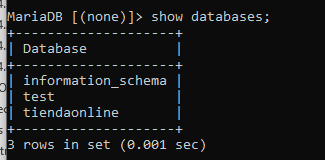
Una vez que se ha creado el usuario, es necesario asignarle los permisos adecuados para acceder a la base de datos y realizar las operaciones necesarias. MySQL proporciona varios comandos para asignar permisos a los usuarios. Por ejemplo, el siguiente comando otorga al usuario "user1" todos los permisos sobre la base de datos "tiendaonline":

GRANT ALL PRIVILEGES ON tiendaonline.\* TO 'user1'@'localhost';

De hecho, cuando aun no se le otorga ningún permiso al usuario podemos ver que al pedirle que muestre todas las bases de datos solo muestra las que trae por defecto MYSQL, en este caso aun no muestra la base que ha sido creada para este ejercicio.

Así que desde el usuario root se le asignaran todos los privilegios sobre esta base de datos, se puede ver que ya la muestra y se le pueden hacer todas las modificaciones necesarias.



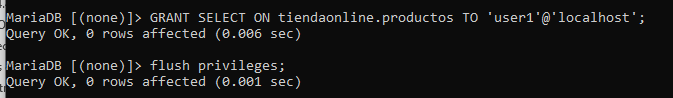


Es importante tener en cuenta que se debe especificar la ubicación del usuario, en este caso se ha establecido en "localhost".

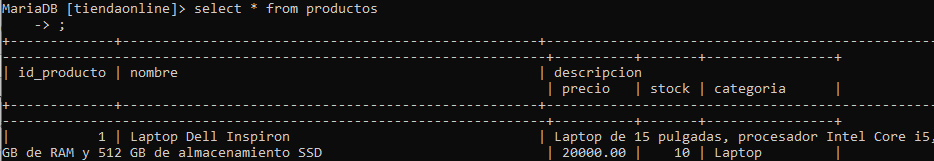
También es posible otorgar permisos específicos a un usuario en lugar de todos los permisos, esto con el fin de limitar los permisos que puede tener cada usuario, por motivos de seguridad; Por ejemplo, el siguiente comando otorga al usuario "user1" el permiso SELECT para la tabla "productos" en la base de datos "tiendaonline": (la sentencia “flush privileges” hace que estos cambios se den al instante y que no deba esperarse a que se reinicie el servidor para aplicarlos)

GRANT SELECT ON tiendaonline.productos TO 'user1'@'localhost';

Flush privileges;



Ahora podremos ver como no se pueden hacer otras acciones desde el usuario “user1” que no sean select





Es importante recordar que los permisos deben ser otorgados de manera cuidadosa y solo a aquellos usuarios que necesiten acceder a los datos y realizar operaciones en la base de datos. De lo contrario, se pueden generar problemas de seguridad en la base de datos.

Por lo tanto, la creación de usuarios y asignación de permisos es una tarea importante en la gestión de bases de datos. Es necesario crear usuarios con las credenciales adecuadas y asignarles los permisos necesarios para acceder a la base de datos y realizar las operaciones requeridas. Con una correcta asignación de permisos, se puede garantizar la seguridad y protección de los datos en la base de datos.

## Mejores prácticas para la gestión de niveles de acceso en bases de datos

Por muy bien diseñada que esté la estructura de una base de datos y por muy potentes que sean los sistemas de seguridad, la gestión de niveles de acceso es una tarea crítica en la administración de una base de datos. Una buena gestión de permisos de usuarios garantiza la protección de la información sensible, la prevención de errores humanos y la reducción de riesgos.

Por ende, a continuación, se presentan algunas de las mejores prácticas a seguir en la gestión de niveles de acceso en bases de datos:

* Identificar los roles y perfiles de usuario: antes de otorgar permisos a un usuario, es importante identificar su perfil y los roles que desempeña en la organización. De esta manera, se pueden definir los permisos y restricciones correspondientes.
* Asignar permisos mínimos necesarios: los usuarios deben tener acceso solo a la información y funciones necesarias para realizar su trabajo. No se deben otorgar permisos innecesarios, ya que esto aumenta el riesgo de un posible uso malintencionado o la exposición de información sensible.
* Aplicar el principio de "menos privilegios": se debe asignar a cada usuario el menor nivel de permisos necesarios para desempeñar su trabajo. De esta manera, se minimiza el riesgo de que un usuario con permisos elevados pueda causar daños en la base de datos.
* Revocar permisos no utilizados: es importante realizar una revisión periódica de los permisos otorgados a los usuarios y revocar aquellos que ya no sean necesarios. Esto reduce la exposición de la información sensible y evita que se produzcan errores humanos.
* Registrar los accesos y acciones realizadas: es importante llevar un registro de los accesos a la base de datos y las acciones realizadas por los usuarios. Esto permite una mayor trazabilidad y la identificación de posibles anomalías o situaciones de riesgo.
* Realizar auditorías de seguridad periódicas: es importante realizar auditorías de seguridad periódicas para identificar posibles vulnerabilidades en la gestión de permisos y corregirlas a tiempo.

En resumidas cuentas, es importante tener en cuenta que, la gestión de niveles de acceso en bases de datos es una tarea crítica en la administración de la información. Por lo tanto, es importante seguir las mejores prácticas y mantener una revisión constante para minimizar el riesgo de exposición de información sensible y proteger la integridad de la base de datos.

# PROCEDIMIENTOS ALMACENADOS

## Introducción a los procedimientos almacenados en bases de datos

Los procedimientos almacenados son una característica importante de los sistemas de gestión de bases de datos (DBMS). Estos procedimientos son programas almacenados en la base de datos que pueden ser llamados por otras aplicaciones o por otros procedimientos almacenados. Los procedimientos almacenados se utilizan para simplificar la programación y mejorar la seguridad en una base de datos.

Un procedimiento almacenado se compone de una serie de instrucciones SQL que realizan una tarea específica. Estas instrucciones SQL se almacenan en la base de datos como un objeto. La ventaja de almacenar estas instrucciones en la base de datos es que se pueden llamar desde cualquier parte de la aplicación, lo que reduce la cantidad de código repetitivo en la aplicación y facilita el mantenimiento.

Además, los procedimientos almacenados también ofrecen una mayor seguridad en la base de datos. Al almacenar la lógica de negocio en la base de datos, se reduce el riesgo de errores en la lógica y se mejora la integridad de los datos. así que también se pueden aplicar permisos de acceso a los procedimientos almacenados, lo que limita el acceso a la lógica de negocio y mejora la seguridad en la base de datos.

Así que, para concluir, se puede destacar que los procedimientos almacenados son una herramienta poderosa para simplificar la programación, también contribuyen a mejorar la seguridad y la integridad de los datos en una base de datos. además, en los siguientes subtemas, se describirá cómo crear y utilizar los procedimientos almacenados en una base de datos.

ventajas y desventajas de utilizar procedimientos almacenados en una base de datos.

Los procedimientos almacenados son una herramienta muy útil para la gestión de bases de datos. cabe destacar como se mencionó anteriormente que su uso puede mejorar significativamente el rendimiento y la seguridad de una base de datos, así como reducir el tiempo y el esfuerzo necesarios para mantener y actualizar una aplicación. Sin embargo, es importante mencionar que también existen algunas desventajas que deben ser consideradas antes de su implementación, por lo tanto, a continuación, se presentaran algunas de las ventajas y desventajas de su uso.

### Ventajas

* Mejora el rendimiento de la base de datos: Los procedimientos almacenados se ejecutan de forma más rápida que las consultas SQL tradicionales. Esto se debe a que, una vez compilado, el procedimiento almacenado se almacena en el servidor de la base de datos y se ejecuta directamente desde allí, evitando el tiempo de compilación y reduciendo el tiempo de acceso a la base de datos.
* Mayor seguridad: Los procedimientos almacenados pueden ser configurados para que sólo sean accesibles a usuarios autorizados, lo que aumenta la seguridad de la base de datos y evita posibles vulnerabilidades de seguridad.
* Reutilización de código: Los procedimientos almacenados son muy útiles para la reutilización de código. Si una operación se realiza con frecuencia en diferentes partes de una aplicación, se puede crear un procedimiento almacenado para llevar a cabo esa operación y llamarlo cada vez que sea necesario, en lugar de tener que escribir el código de nuevo en cada parte de la aplicación.
* Facilidad de mantenimiento: Cuando se realiza una actualización o modificación en un procedimiento almacenado, esa actualización se aplica automáticamente en todas las partes de la aplicación que utilizan ese procedimiento almacenado. Esto significa que se pueden realizar cambios de manera más rápida y sencilla, y se reduce el riesgo de errores al actualizar código en múltiples partes de la aplicación.

### Desventajas

* Dificultad de depuración: La depuración de procedimientos almacenados puede ser más complicada que la depuración de consultas SQL tradicionales, ya que los procedimientos almacenados no se ejecutan en un entorno de desarrollo interactivo.
* Dificultad para entender el código: A menudo, los procedimientos almacenados pueden contener una gran cantidad de código, lo que puede dificultar su comprensión y mantenimiento. Además, los cambios realizados en el código de un procedimiento almacenado pueden tener consecuencias imprevistas en otras partes de la aplicación.
* Dependencia de la base de datos: Los procedimientos almacenados están estrechamente vinculados a la base de datos en la que se ejecutan. Esto significa que, si se cambia el sistema de base de datos, puede haber problemas con los procedimientos almacenados y, por lo tanto, con la aplicación en general.

## Creación de procedimientos almacenados

Los procedimientos almacenados son un conjunto de instrucciones que se ejecutan de manera secuencial en una base de datos y que se pueden invocar mediante un nombre específico. Estos procedimientos pueden ser creados para realizar tareas específicas en una base de datos y se pueden llamar desde aplicaciones externas.

Para crear un procedimiento almacenado, se necesita tener privilegios para realizar la operación en la base de datos. A continuación, se presenta un ejemplo de cómo crear un procedimiento almacenado en MySQL:

CREATE PROCEDURE NombreProcedimiento (IN parametro1 INT, OUT parametro2 VARCHAR(50))

BEGIN

-- Cuerpo del procedimiento

END;

En este ejemplo, se crea un procedimiento llamado NombreProcedimiento con dos parámetros, parametro1 de tipo INT y parametro2 de tipo VARCHAR con una longitud máxima de 50 caracteres. El IN indica que el parámetro es de entrada y el OUT indica que el parámetro es de salida.

Dentro del cuerpo del procedimiento, se pueden realizar diferentes operaciones, como insertar, actualizar o eliminar registros en la base de datos, entre otras. Al final del procedimiento, se debe definir qué valor se asignará a cada uno de los parámetros de salida.

A continuación, se presenta un ejemplo con base en la base de datos que hemos creado para los ejercicios para dar un ejemplo más detallado de un procedimiento almacenado, que en este caso lo que hará sera \*\*inserta un nuevo registro en la tabla Clientes:\*\*

CREATE PROCEDURE InsertarCliente (IN nombre VARCHAR(50), IN apellido VARCHAR(50), IN edad INT)

BEGIN

INSERT INTO Clientes (Nombre, Apellido, Edad) VALUES (nombre, apellido, edad);

END;

En este ejemplo, se crea un procedimiento llamado InsertarCliente con tres parámetros de entrada, nombre, apellido y edad. Dentro del cuerpo del procedimiento, se inserta un nuevo registro en la tabla Clientes con los valores de los parámetros de entrada.

Una vez creado el procedimiento almacenado, se puede invocar mediante su nombre y pasarle los parámetros correspondientes. A continuación, se presenta un ejemplo de cómo invocar el procedimiento InsertarCliente:

arduino

Copy code

CALL InsertarCliente('Juan', 'Pérez', 25);

En este ejemplo, se llama al procedimiento InsertarCliente y se le pasan los valores Juan, Pérez y 25 para los parámetros nombre, apellido y edad, respectivamente.

Es importante tener en cuenta que la creación de procedimientos almacenados puede mejorar el rendimiento de una base de datos, ya que se pueden reducir el tráfico de red y el tiempo de respuesta. Sin embargo, también se debe tener cuidado al crear procedimientos almacenados complejos, ya que pueden dificultar la mantenibilidad de la base de datos.

Con lo visto anteriormente, se puede decir que, la creación de procedimientos almacenados puede ser muy útil para realizar operaciones específicas en una base de datos. Para crear un procedimiento almacenado, se necesita tener privilegios en la base de datos y se debe seguir la sintaxis específica del motor de base de datos utilizado.

## Buenas prácticas para la utilización de procedimientos almacenados en bases de datos

Los procedimientos almacenados son una herramienta muy útil en el mundo de las bases de datos. como vimos anteriormente permiten encapsular una serie de comandos SQL en un solo objeto, además también se vio como facilita su reutilización y mantenimiento. Sin embargo, su uso también puede presentar algunos riesgos si no se siguen las mejores prácticas.

Así que por esta razón y para finalizar este capítulo, se presentan algunas recomendaciones para la utilización de procedimientos almacenados en bases de datos:

* Documenta tus procedimientos almacenados: Es importante que los procedimientos almacenados estén debidamente documentados, tanto en términos de su propósito como de su funcionamiento interno. De esta manera, podrás entender fácilmente lo que hace cada procedimiento almacenado y cómo funciona.
* Valida tus entradas: Es importante validar todas las entradas que recibe un procedimiento almacenado. Si un usuario malintencionado o un error de programación envía una entrada malformada, el procedimiento almacenado podría fallar o incluso comprometer la seguridad de la base de datos.
* Utiliza transacciones: Los procedimientos almacenados pueden incluir varias sentencias SQL, algunas de las cuales pueden fallar en caso de un error. Por lo tanto, es recomendable utilizar transacciones para asegurarse de que todas las sentencias SQL se ejecuten correctamente o ninguna se ejecute en absoluto.
* Limita los permisos: Es importante limitar los permisos que se otorgan a los procedimientos almacenados. Los permisos deberían ser los mínimos necesarios para que el procedimiento almacenado pueda realizar su función.
* Realiza pruebas: Es recomendable realizar pruebas exhaustivas de los procedimientos almacenados antes de ponerlos en producción. Esto incluye pruebas tanto de su funcionamiento interno como de su integración con otras partes del sistema.
* Optimiza el rendimiento: Los procedimientos almacenados pueden afectar significativamente el rendimiento de la base de datos. Es importante optimizar el rendimiento de los procedimientos almacenados mediante el uso de índices, consultas eficientes y una arquitectura de base de datos adecuada.

Después de esto, siguiendo estas mejores prácticas, se podrá aprovechar al máximo los beneficios de los procedimientos almacenados sin comprometer la seguridad ni el rendimiento de la base de datos.

# TRIGGERS

## Introducción a los Triggers en bases de datos

En una base de datos, los triggers son objetos que permiten ejecutar una serie de instrucciones automáticamente en respuesta a ciertos eventos, como la inserción, actualización o eliminación de datos de una tabla específica.

Los triggers pueden ser muy útiles para automatizar procesos en una base de datos y reducir la cantidad de código que se necesita para realizar determinadas operaciones. Además, pueden ser una forma efectiva de garantizar la integridad de los datos y aplicar reglas de negocio en la base de datos.

Existen dos tipos de triggers en las bases de datos: los triggers de fila y los triggers de tabla. Los triggers de fila se activan para cada fila que se inserta, actualiza o elimina de una tabla específica, mientras que los triggers de tabla se activan una vez por instrucción SQL, independientemente del número de filas afectadas.

Para crear un trigger, se debe especificar el evento que activará el trigger (por ejemplo, INSERT, UPDATE o DELETE), la tabla en la que se activará el trigger y el conjunto de instrucciones que se ejecutarán cuando se active el trigger.

Es importante tener en cuenta que los triggers pueden afectar el rendimiento de la base de datos si no se usan correctamente. Por lo tanto, se deben crear con cuidado y solo se deben utilizar cuando sea necesario.

En el siguiente subtema, se profundizara en las ventajas y desventajas de utilizar triggers en una base de datos.

## Ventajas y desventajas de utilizar Triggers

Los Triggers como ya se mencionó, se trata de objetos de base de datos que se utilizan para automatizar ciertas acciones en una base de datos. cómo se explicó en el subtema anterior estos objetos se ejecutan automáticamente en respuesta a eventos específicos en la base de datos, como la inserción, actualización o eliminación de datos en una tabla. Los Triggers pueden ser muy útiles para mantener la integridad de los datos y automatizar ciertas tareas en la base de datos. Sin embargo, como cualquier otra herramienta, tienen sus ventajas y desventajas, por lo tanto, es importante evaluar cuidadosamente si el uso de Triggers es apropiado para su base de datos y, si se decide utilizarlos, asegurarse de hacerlo de manera correcta y cuidadosa para minimizar los posibles riesgos.

A continuación, se profundizará más en algunas de las ventajas y desventajas de utilizar Triggers en una base de datos.

### Ventajas

* Automatización de tareas: Los Triggers permiten automatizar ciertas tareas, lo que puede ahorrar tiempo y esfuerzo. Por ejemplo, se puede utilizar un Trigger para actualizar una tabla automáticamente cuando se insertan o eliminan datos en otra tabla relacionada.
* Integridad de datos: Los Triggers también pueden utilizarse para garantizar la integridad de los datos en la base de datos. Por ejemplo, se puede utilizar un Trigger para evitar que se inserten datos duplicados en una tabla o para garantizar que ciertas restricciones de integridad se cumplan.
* Flexibilidad: Los Triggers son muy flexibles y se pueden utilizar para realizar una amplia variedad de tareas en la base de datos. Por ejemplo, se pueden utilizar para auditar cambios en la base de datos o para enviar alertas cuando se producen ciertos eventos en la base de datos.

### Desventajas

* Rendimiento: Los Triggers pueden afectar al rendimiento de la base de datos. Si se utilizan de manera incorrecta, pueden ralentizar las operaciones de la base de datos, lo que puede afectar negativamente al rendimiento general del sistema.
* Complejidad: Los Triggers pueden ser bastante complejos, lo que puede dificultar su mantenimiento y gestión. Si se utilizan varios Triggers en una base de datos, pueden ser difíciles de rastrear y depurar.
* Errores: Si se utiliza un Trigger de manera incorrecta, puede dar lugar a errores en la base de datos. Por ejemplo, un Trigger mal escrito podría causar una actualización incorrecta de los datos en la base de datos.

## Creación de Triggers en bases de datos

Como ya se menciono anteriormente, un trigger es un tipo de objeto de base de datos que se utiliza para realizar acciones automáticamente en respuesta a un evento específico en una tabla, como una inserción, actualización o eliminación de registros, además también ya se presentaron algunas ventajas y de ventajas de usarlos. En este subtema, se explicará cómo crear un trigger en una base de datos.

### Pasos para crear un trigger

* Identificar el evento que desencadenará el trigger: El primer paso para crear un trigger es identificar el evento que desencadenará la acción que se desea automatizar. Esto podría ser una inserción, actualización o eliminación de registros en una tabla específica.
* Crear la lógica del trigger: Una vez identificado el evento, el siguiente paso es escribir la lógica del trigger. Esto determinará qué acción se llevará a cabo cuando se desencadene el evento. Por ejemplo, se puede crear un trigger que actualice automáticamente la fecha de última modificación de un registro cada vez que se actualice ese registro.
* Crear el trigger: Después de haber identificado el evento y escrito la lógica del trigger, el siguiente paso es crear el trigger en la base de datos. Para ello, se utiliza la sintaxis específica del lenguaje de programación de la base de datos. Aquí hay un ejemplo de sintaxis para crear un trigger en MySQL:

CREATE TRIGGER nombre\_trigger

AFTER INSERT ON nombre\_tabla

FOR EACH ROW

BEGIN

-- Lógica del trigger aquí

END;

Este ejemplo crea un trigger que se activa después de una inserción en una tabla específica y ejecuta la lógica definida dentro del bloque "BEGIN" y "END". Es importante tener en cuenta que la sintaxis específica del trigger variará dependiendo del lenguaje de programación de la base de datos que se esté utilizando.

\*insertar ejemplo real\*

### Consideraciones al crear un trigger

También es importante mencionar que al crear un trigger en una base de datos, se debe considerar lo siguiente:

* Efectos secundarios: Los triggers pueden tener efectos secundarios no deseados, como un aumento en el tiempo de procesamiento o un uso excesivo de recursos de la base de datos. Por lo tanto, es importante probar cuidadosamente el trigger antes de implementarlo en una base de datos en producción.
* Reglas de negocio: Los triggers pueden usarse para implementar reglas de negocio, pero también pueden introducir complejidad adicional en una base de datos. Es importante tener en cuenta la complejidad y la escalabilidad de los triggers al diseñar una base de datos.
* Seguridad: Los triggers pueden ser utilizados para introducir vulnerabilidades de seguridad en una base de datos si no se configuran correctamente. Es importante limitar el acceso a los triggers solo a aquellos usuarios que necesiten utilizarlos y asegurarse de que se implementen de manera segura.

## Buenas prácticas para la utilización de Triggers en bases de datos

Como se ha mencionado durante los temas anteriores, se trata de una herramienta muy útil para automatizar ciertas acciones en una base de datos. Y como se ha mencionado, su uso debe ser cuidadoso, ya que, si se utilizan de manera incorrecta, pueden causar problemas y errores en la base de datos. por lo tanto a continuación, se presentan algunas buenas prácticas que al seguirlas, pueden asegurar que los Triggers se utilicen de manera efectiva y sin afectar el rendimiento de la base de datos.

* Limitar el número de Triggers por tabla: Es importante limitar el número de Triggers por tabla, ya que un gran número de Triggers puede disminuir el rendimiento de la base de datos. Si se requieren múltiples Triggers en una tabla, se debe considerar combinarlos en un solo Trigger para evitar afectar el rendimiento.
* Establecer un orden de ejecución para los Triggers: Cuando se utilizan múltiples Triggers en una tabla, es importante establecer un orden de ejecución para evitar conflictos entre ellos. El orden de ejecución puede establecerse utilizando la cláusula AFTER o BEFORE en la definición del Trigger.
* Asegurarse de que los Triggers no entren en conflicto con las restricciones de integridad: Los Triggers deben diseñarse cuidadosamente para asegurarse de que no entren en conflicto con las restricciones de integridad de la base de datos. Por ejemplo, si se utiliza un Trigger para actualizar una columna en una tabla, es importante asegurarse de que el nuevo valor no viole ninguna restricción de integridad.
* Utilizar los Triggers para fines específicos: Es importante utilizar los Triggers para fines específicos y evitar hacer demasiadas acciones en un solo Trigger. Por ejemplo, es recomendable utilizar un Trigger para realizar una sola acción, como actualizar una columna en una tabla, en lugar de realizar múltiples acciones en un solo Trigger.
* Documentar los Triggers: Es importante documentar los Triggers para que otros usuarios puedan entender su funcionalidad y para facilitar su mantenimiento. La documentación debe incluir información sobre qué hace el Trigger, cuándo se ejecuta y cualquier restricción o dependencia que tenga.

# Estructuras y usos de if y case

## Introducción a las estructuras if y case en SQL

Una de las características más importantes de SQL es su capacidad para ejecutar instrucciones condicionales, lo que permite controlar el flujo de la ejecución de los programas y tomar decisiones basadas en ciertos criterios.

Por lo tanto, las estructuras de control if y case son dos de las herramientas más comunes que se utilizan en SQL para implementar instrucciones condicionales. Ambas estructuras permiten tomar decisiones basadas en una o varias condiciones.

La estructura if se utiliza para ejecutar un bloque de código si una condición se cumple y, de lo contrario, ejecutar otro bloque de código. Esta estructura se utiliza para tomar decisiones basadas en una única condición.

Por otro lado, la estructura case se utiliza para ejecutar diferentes bloques de código en función del valor de una expresión. Esta estructura se utiliza para tomar decisiones basadas en múltiples condiciones.

### La estructura if en SQL

La estructura if en SQL funciona de manera similar a la estructura if en otros lenguajes de programación. La sintaxis básica de la estructura if en SQL es la siguiente:

IF (condición)

BEGIN

-- código a ejecutar si la condición es verdadera

END

ELSE

BEGIN

-- código a ejecutar si la condición es falsa

END

En este primer ejemplo general de la sintaxis, si la condición es verdadera, se ejecuta el bloque de código que se encuentra dentro del BEGIN y END después de la condición. Si la condición es falsa, se ejecuta el bloque de código que se encuentra dentro del BEGIN y END después de la cláusula ELSE.

### La estructura case en SQL

La estructura case en SQL es una herramienta muy útil cuando se necesitan tomar decisiones basadas en múltiples condiciones. La sintaxis básica de la estructura case es la siguiente:

CASE expresión

WHEN valor1 THEN

-- código a ejecutar cuando la expresión es igual a valor1

WHEN valor2 THEN

-- código a ejecutar cuando la expresión es igual a valor2

ELSE

-- código a ejecutar cuando la expresión no es igual a ningún valor

END

En este ejemplo muy general, la estructura case evalúa la expresión y luego busca una coincidencia con los valores especificados en la cláusula WHEN. Si la expresión coincide con valor1, se ejecuta el bloque de código que se encuentra después de la cláusula THEN. Si no hay coincidencias, se ejecuta el bloque de código que se encuentra después de la cláusula ELSE.

Cabe recordar que La estructura if se utiliza para tomar decisiones basadas en una única condición, mientras que la estructura case se utiliza para tomar decisiones basadas en múltiples condiciones. Con una buena comprensión de estas estructuras, los programadores de SQL pueden escribir código más eficiente y efectivo, sin embargo, en el siguiente tema se ahondará más en las diferencias en estas estructuras de control.

## Estructuras if y case en SQL: diferencias y similitudes

Las estructuras if y case se utilizan para implementar instrucciones condicionales y tomar decisiones basadas en ciertos criterios y aunque ambas estructuras tienen un propósito similar, también tienen diferencias significativas en su sintaxis y uso. En este artículo, exploraremos las diferencias y similitudes entre las estructuras if y case en SQL.

### Diferencias entre las estructuras if y case

#### Sintaxis:

La sintaxis de la estructura if en SQL es la siguiente (que ya se ha mostrado en el ejemplo anterior de la estructura básica de la sintaxis de un if:

IF (condición)

BEGIN

-- código a ejecutar si la condición es verdadera

END

ELSE

BEGIN

-- código a ejecutar si la condición es falsa

END

Mientras que la sintaxis de la estructura case en SQL es la siguiente (que igual ya se ha mostrado en el capítulo anterior):

CASE expresión

WHEN valor1 THEN

-- código a ejecutar cuando la expresión es igual a valor1

WHEN valor2 THEN

-- código a ejecutar cuando la expresión es igual a valor2

ELSE

-- código a ejecutar cuando la expresión no es igual a ningún valor

END

Como se puede ver, la sintaxis de la estructura if es más similar a la de otros lenguajes de programación, utilizando una condición booleana para determinar qué bloque de código ejecutar. Por otro lado, la estructura case utiliza una expresión que se evalúa y busca coincidencias con los valores especificados en las cláusulas WHEN.

#### Uso:

La estructura if se utiliza para tomar decisiones basadas en una única condición. Se puede utilizar para ejecutar un bloque de código si la condición es verdadera y otro bloque de código si la condición es falsa. Es especialmente útil cuando se necesita realizar una acción específica en función de una condición booleana.

Por otro lado, la estructura case se utiliza para tomar decisiones basadas en múltiples condiciones. Se puede utilizar para ejecutar diferentes bloques de código en función del valor de una expresión. Es especialmente útil cuando se necesita realizar diferentes acciones en función de diferentes valores posibles de una expresión.

### Similitudes entre las estructuras if y case

Aunque hay diferencias importantes entre las estructuras if y case en SQL como se vio antes, también comparten algunas similitudes:

* Ambas son estructuras condicionales: Tanto la estructura if como la estructura case son utilizadas para implementar instrucciones condicionales en SQL. Permiten controlar el flujo de la ejecución del código en función de ciertas condiciones, lo que facilita la toma de decisiones en función de los datos almacenados en una base de datos.
* Ambas pueden contener bloques de código: Tanto la estructura if como la estructura case pueden contener bloques de código que se ejecutan en función de las condiciones especificadas. Esto permite realizar acciones específicas en función de si se cumple o no una condición determinada.
* Ambas son ampliamente utilizadas en SQL: Tanto la estructura if como la estructura case son ampliamente utilizadas en SQL para implementar lógica condicional en procedimientos almacenados, disparadores (triggers) y otras partes del código SQL. Son herramientas esenciales para la programación SQL y son utilizadas por los desarrolladores para controlar el flujo de ejecución de los programas y por supuesto, la toma de decisiones.