UNIDAD DIDÁCTICA 3

TRATAMIENTO DE DATOS

MÓDULO PROFESIONAL:
BASES DE DATOS

CESUR
Tu Centro Oficial de FP

Índice

RESUMEN INTRODUCTORIO

En la presente unidad se estudiarán conceptos relacionados con el lenguaje SQL. En este sentido, se da una introducción a dicho lenguaje, qué objetivos persigue y cuáles son las características y ventajas que ofrece, entre ellas: independencia de proveedores, portabilidad entre sistemas, fundamento relacional, arquitectura cliente/servidor, etc. Se analizan los diversos usuarios que pueden interactuar con la base de datos, tales como administradores, programadores y usuarios finales. La base de esta unidad y de las siguientes es conocer las órdenes DDL y DML más habituales. En este sentido, se analizan operaciones para crear, eliminar y modificar una base de datos, e introducir, eliminar y modificar datos de la base de datos.

INTRODUCCIÓN

Se puede considerar el lenguaje SQL como la piedra angular de las bases de datos relacionales. Aparte de ser un lenguaje universal e independiente de la plataforma en la que se ejecuta, no posee filtros a la hora de ser implementado tanto en software libre como en software privativo. En este sentido, se puede afirmar que SQL ofrece muchas más ventajas que inconvenientes, y así se ha corroborado desde su aparición hasta nuestros días.

En las factorías de software no se entiende el desarrollo de aplicaciones sin la ejecución de consultas y llamadas a SQL desde cualquier tipo de base de datos, ya sea MySQL, Microsoft Access, PostgreSQL o Microsoft SQL Server, entre otras. Además de brindar esta universalidad en el ámbito de ejecución sobre los diferentes SGBD, ofrece una amplia variedad de funciones, como son, entre otras, la creación, la inserción, el borrado y la actualización de la información almacenada.

Por último, cabe destacar la gran capacidad de adaptación a servidores e infraestructuras en red, tanto para arquitecturas de bases de datos centralizadas como distribuidas.

CASO INTRODUCTORIO

Uno de los equipos de desarrollo de la empresa que trabajas decide tomar las riendas de un proyecto software. En él se han establecido una serie de requisitos que deben ser implementados y, a su vez, se debe realizar un diseño en una base de datos en MySQL.

Para ello debes llevar a cabo una serie de acciones que involucren la creación de tablas y la definición de estructuras para permitir almacenar la información según los requerimientos establecidos.

Al finalizar el estudio de la unidad, serás capaz de crear, modificar y eliminar bases de datos, crearás tablas y cómo relacionarlas y conocerás la forma de insertar, modificar y suprimir datos de determinadas tablas.

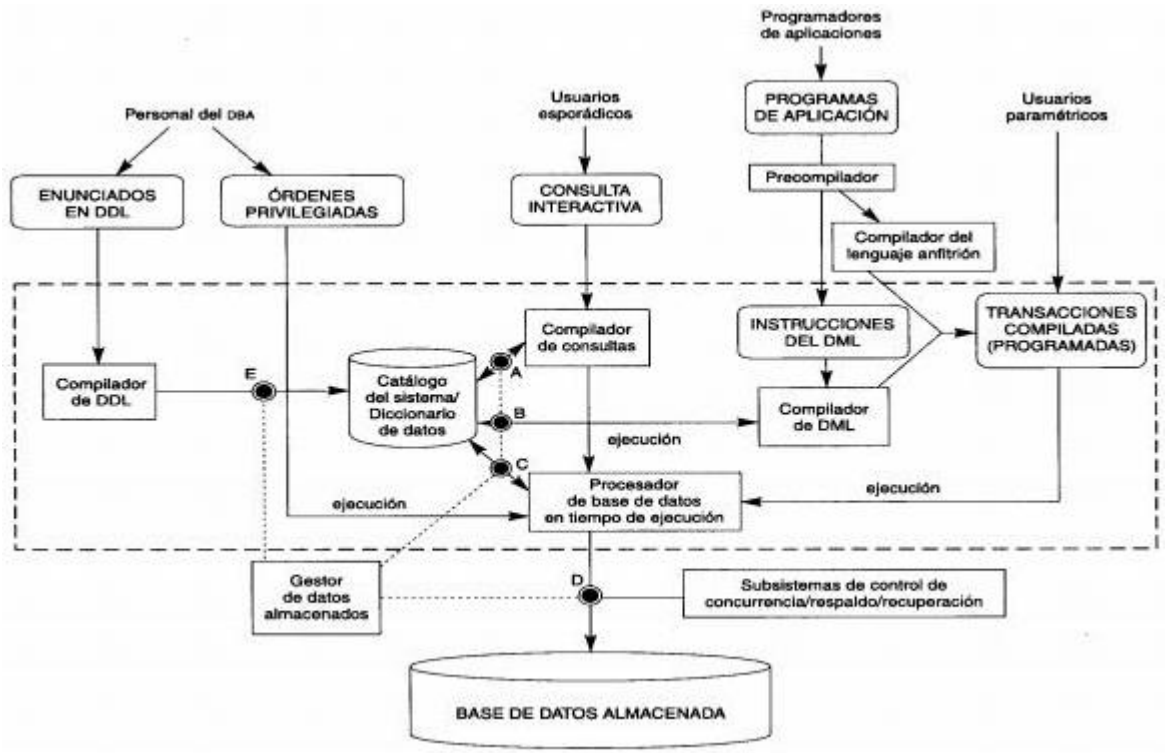
1. SQL. OBJETIVOS Y CARACTERÍSTICAS

Vas a formar parte del equipo de desarrollo encargado de la creación de la base de datos de la empresa y, debido a la importancia que tiene este proyecto para la misma, debes ponerte al día con el lenguaje SQL y los diferentes tipos de instrucciones que ofrece, ya que es el lenguaje en el que se basan muchos de los SGBD del mercado. Además, para asegurarte de que SQL es la elección correcta, debes investigar sobre sus principales ventajas y características.

SQL es una herramienta que permite organizar, gestionar y recuperar datos almacenados en una base de datos. SQL, por tanto, es un lenguaje informático que se utiliza para interactuar con una base de datos. De hecho, SQL funciona con un tipo específico de base de datos, llamado base de datos relacional. En la actualidad es el lenguaje de uso y programación de bases de datos relacionales más extendido.

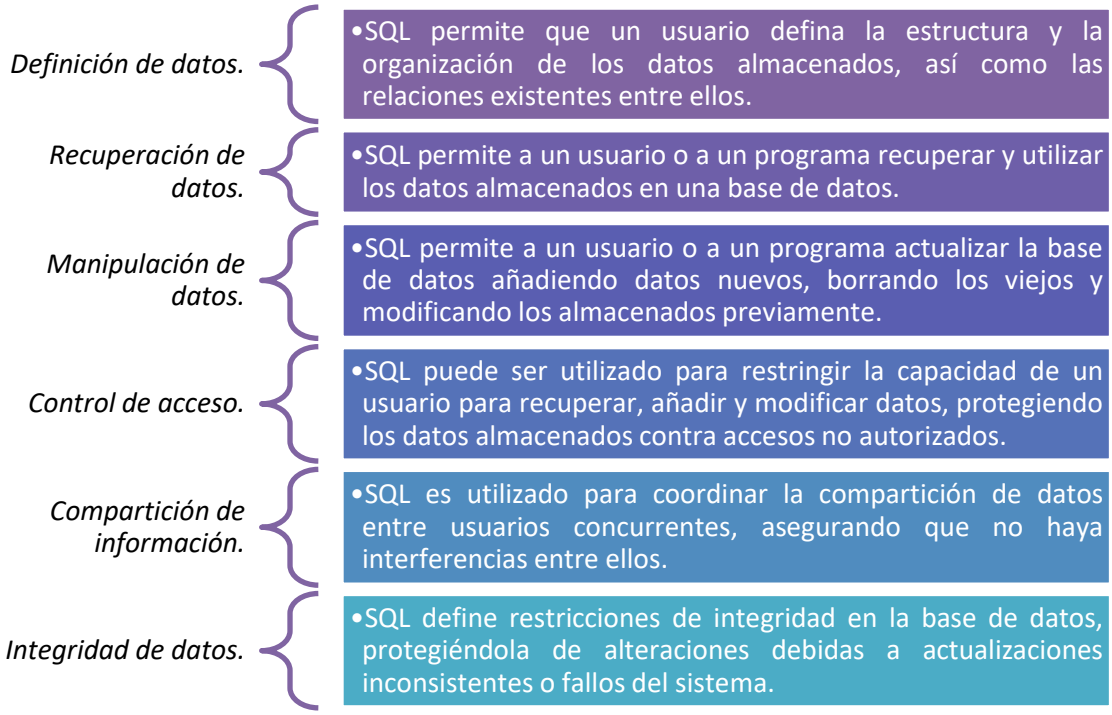
Representación del lenguaje SQL.

Como ya se ha mencionado en unidades anteriores, el programa que controla la base de datos recibe el nombre de Sistema de Gestión de Base de Datos (SGBD o SGBD).



Representación del funcionamiento de un SGBD.

Cuando es necesario recuperar datos de una base de datos, la petición se realiza utilizando el lenguaje SQL. El SGBD procesa la petición SQL, recoge los datos solicitados y los devuelve a quien los solicitó. Este proceso de petición de datos de la base de datos y posterior recepción de resultados recibe el nombre de consulta o query. De aquí el nombre lenguaje estructurado de consultas. Además, accede a la base de datos a lo largo de todo su ciclo de vida, trabajando en modo declarativo. Cuando un usuario realiza una operación, no debe describirla paso a paso, basta con especificar el resultado que se desea mediante cláusulas y predicados. El gestor de la base de datos se ocupará de realizar las tareas necesarias para hacer efectiva la petición. SQL se utiliza para controlar todas las funciones que suministra un SGBD a sus usuarios, incluyendo:



Funciones que proporciona SQL.

En segundo lugar, SQL no es realmente un lenguaje completo ni contiene sentencias IF para probar condiciones, ni sentencias GOTO de salto, ni siquiera sentencias DO o FOR para realizar bucles. En vez de eso, SQL es un sublenguaje de base de datos que consiste en un conjunto de sentencias especializadas en tareas de gestión de base de datos. Estas sentencias están embebidas en otro lenguaje, que las utiliza para acceder a las bases de datos.

Finalmente, SQL no es un lenguaje particularmente estructurado, especialmente cuando se compara con lenguajes altamente estructurados como C o Pascal. Hay bastantes inconsistencias en el lenguaje SQL y además existen reglas especiales que evitan la construcción de sentencias SQL que carecen de mucho sentido.

ENLACE DE INTERÉS

Visita la web CHM y conoce más sobre Donald Chamberlin:

1.1 Objetivos de SQL

El motor de la base de datos es el corazón del SGBD, responsable de su estructura, almacenamiento y recuperación de los datos del disco. Acepta peticiones SQL de otros componentes del SGBD, tales como facilidades de formularios, generadores de informes o facilidades de consultas interactivas de programas implementados por usuarios e incluso de otros sistemas informáticos.

Lenguaje interactivo de consultas	• Los usuarios escriben órdenes SQL en un programa SQL interactivo para recuperar datos y presentarlos en la pantalla.
Lenguaje de programación de bases de datos	• Los programadores introducen órdenes SQL en sus programas de acceder a los datos de las bases de datos.
Lenguaje de administración de base de datos	• El administrador responsable de la gestión de la base de una minicomputadora o sistema basado en una computadora central utiliza SQL para definir la estructura de la base de datos y el control de acceso a los datos almacenados.
Lenguaje cliente/servidor	• Los programas de las computadoras personales utilizan SQL para comunicarse, a través de una red de área local, con los servidores de bases de datos que almacenan los datos compartidos.
Lenguaje de bases de datos distribuidos	• Los sistemas de gestión de bases de datos distribuidas utilizan SQL para ayudar a distribuir los datos a través de muchos sistemas informáticos conectados.

Objetivos de SQL.

VÍDEO DE INTERÉS

Fue en los laboratorios de IBM donde se fraguó el primer lenguaje de lo que hoy se conoce como SQL. Visualiza este vídeo y conoce los principales logros de esta compañía:

1.2 Características y ventajas de SQL

A continuación, se muestran las principales características de SQL y las tendencias del mercado que le han hecho conseguir el éxito, entre ellas, ser independiente de los proveedores que lo utilizan, ser portable sin atender el sistema informático que lo ejecuta (un PC, dispositivo móvil, tablet, etc.), el estándar ODBC de Microsoft para conectar con bases de datos, el fundamento relacional, las múltiples vistas de la información almacenada, lenguaje completo y consistente, definición dinámica de los datos y arquitectura cliente/servidor.

- **Independencia de los proveedores:** Ningún nuevo producto de base de datos puede tener éxito si no ofrecen SQL. Una base de datos basada en SQL y los programas que la usan, se pueden transferir del SGBD de un proveedor a otro con un esfuerzo mínimo de conversión y un pequeño reciclaje personal. El SQL es tan popular en el entorno de las bases de datos, que muchos lenguajes de programación incluyen sentencias SQL como parte de su repertorio de instrucciones. Tal es el caso del lenguaje Visual Basic que, cuando se utiliza para acceder a las bases de datos creadas con cualquier producto relacional, permite describir un subconjunto de los registros mediante una sentencia SQL.

- **Portabilidad entre sistemas informáticos**: Los proveedores de SGBD basados en SQL ofrecen sus productos para un rango de computadoras que incluye las computadoras personales y estaciones de trabajo en redes de área local, las minicomputadoras y los sistemas basados en una computadora central. Las aplicaciones basadas en SQL que comienzan en sistemas monousuarios pueden ser transferidas a minicomputadoras mayores o a sistemas basados en una computadora central, según crecen. SQL ha sido utilizado con diferentes sistemas comerciales, como lenguaje nativo o como una incorporación posterior a un sistema relacional preexistente, como INGRES. Tras convertirse en un

lenguaje estándar por ISO, se han desarrollado multitud de productos comerciales SQL. Existen Intérpretes de SQL en micros, minis y grandes ordenadores, sobre diferentes sistemas operativos.

- **Acuerdos con Microsoft (ODBC)**: Microsoft considera el acceso a las bases de datos como un elemento clave de la arquitectura software de Windows para computadoras personales. El estándar de Microsoft que proporciona acceso a bases de datos es ODBC (conexión entre base de datos abiertas, Open Database Connectivity), una facilidad basada en SQL. Las principales aplicaciones Windows, tanto de Microsoft como de otros proveedores líderes de aplicaciones Windows son compatibles con ODBC, y el acceso ODBC está disponible o anunciado en las principales bases de datos SQL.

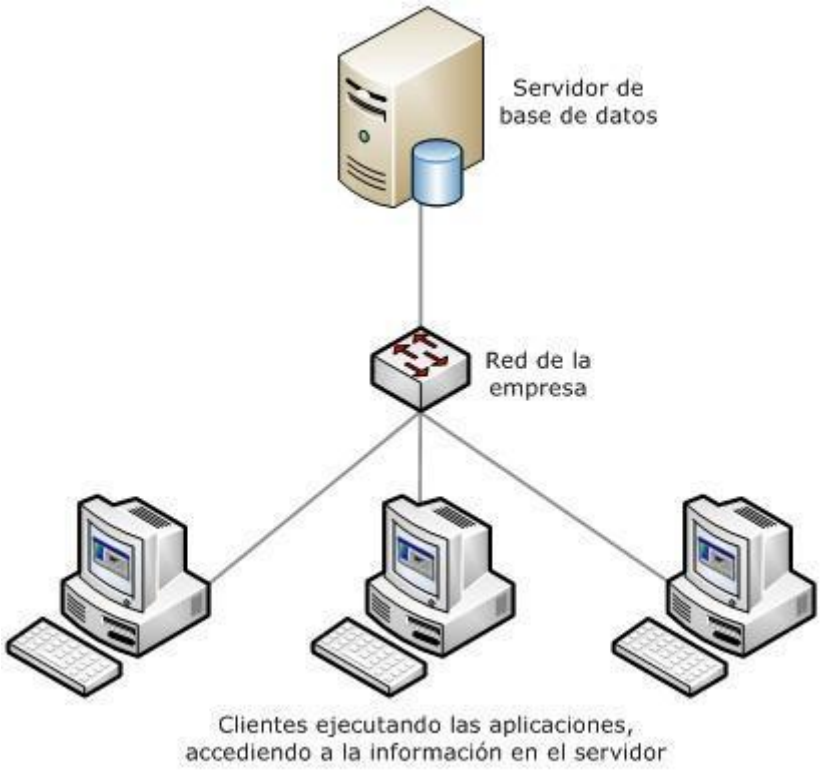
- **Fundamento relacional:** La estructura tabular fila/columna de las bases de datos relacionales es intuitiva para los usuarios, manteniendo el lenguaje sencillo y fácil de entender. El modelo relacional también tiene un fuerte fundamento teórico que ha guiado la evolución y la implementación de las bases de datos relacionales.

- **Múltiples vistas de los datos**: Utilizando SQL, el creador de una base de datos puede dar vistas diferentes de su estructura y contenidos a diferentes usuarios de la base de datos. Por ejemplo, la base de datos puede ser construida de modo que cada usuario sólo vea datos de su propio departamento o de su propia región de ventas. Además, los datos procedentes de diferentes partes de la base de datos pueden combinarse y presentarse al usuario como una simple fila/columna de una tabla. Las vistas de SQL pueden ser utilizadas de este modo para mejorar la seguridad de una base de datos y para acomodarla a las necesidades particulares de los usuarios individuales.

- **Lenguaje completo de base de datos:** SQL fue inicialmente desarrollado como un lenguaje de consulta ad hoc, pero su potencia va más allá de la recuperación de datos. Es un lenguaje completo y consistente para crear una base de datos, gestionar su seguridad, actualizar contenidos, recuperar datos y compartirlos entre usuarios concurrentes.

- **Definición dinámica de datos:** Utilizando SQL, la estructura de una base de datos puede ser modificada y ampliada dinámicamente, incluso mientras los usuarios están accediendo a los contenidos de la base de datos. Este es un avance importante sobre los lenguajes de definición de datos estáticos.

- **Arquitectura Cliente/Servidor:** SQL es un vehículo natural para implementar aplicaciones utilizando una arquitectura de tipo cliente/servidor distribuida.



Arquitectura cliente/servidor.

2. TIPOS DE DATOS

Debido a que eres uno de los encargados de implementar la creación de la base de datos, es imprescindible que conozcas los tipos de datos existentes para poder identificar el tipo de dato correcto para almacenar según qué información, por lo que buscas más información para poder sacar el máximo partido.

SQL define una serie de tipos de datos que va a gestionar. El propósito de la definición de tipos es múltiple.

- Es necesario conocer el tamaño a reservar para cada uno de los datos nuevos que se van a introducir.
- Dividiendo los datos en diferentes tipos se pueden implementar restricciones de integridad (valores posibles contemplados en los campos).
- Una buena definición de los tipos de datos ayuda a los administradores y programadores en su tarea.

SQL es un lenguaje que varía sensiblemente de unos SGBD a otros, por lo que por ejemplo los nombres de tipos de datos utilizados en un SGBD determinado pueden no coincidir con los de otro.

NÚMEROS EXACTOS:

- NÚMEROS ENTEROS: **INT o INTEGER** → Hace referencia a un tipo de dato entero de 32 bits (0 y 2^{32}).

- NÚMEROS ENTEROS CORTOS: **SMALLINT** → Hace referencia a un tipo de dato entero de 16 bits (0 y 2^{16}).

NÚMEROS DECIMALES:

- NUMERIC (precisión, escala) y DECIMAL (precisión, escala).

En ambos hay que especificar dos datos, la precisión, que es el número de dígitos en el número; y la escala, que es la cantidad de dígitos que están a la derecha del punto decimal. La diferencia entre ellos es que NUMERIC tiene una longitud fija (parte entera más decimal), mientras que esto no ocurre con DECIMAL. En ambos casos se tiene que cumplir que escala <= precisión.

NÚMEROS APROXIMADOS:

- NÚMEROS REALES: **REAL** → Define un número decimal de coma flotante con hasta 7 dígitos de mantisa. Ocupa 4 bytes de almacenamiento.

- NÚMEROS FLOTANTES: **FLOAT (precisión)** → Es necesario especificar la precisión del número. Ocupa 8 bytes de almacenamiento. Tiene 15 dígitos de precisión.

CADENAS DE CARACTERES:

- CARACTERES FIJOS: **CHAR (numero) o CHARACTER (numero)** → Siempre se almacena el número de caracteres indicado en número, aun cuando sea necesario rellenar los espacios sobrantes con caracteres en blanco porque la longitud de la cadena de caracteres sea inferior a número.

- CARACTERES VARIABLES: **VARCHAR (numero)** → Sólo se almacenan los caracteres que se hayan introducido, hasta un máximo que viene dado por número.

CADENAS DE BITS:

- **BIT (numero)** → Teniendo número el mismo significado que en el tipo anterior. Habitualmente las cadenas de bits se usan para almacenar flags (banderas) para el control de algún proceso.

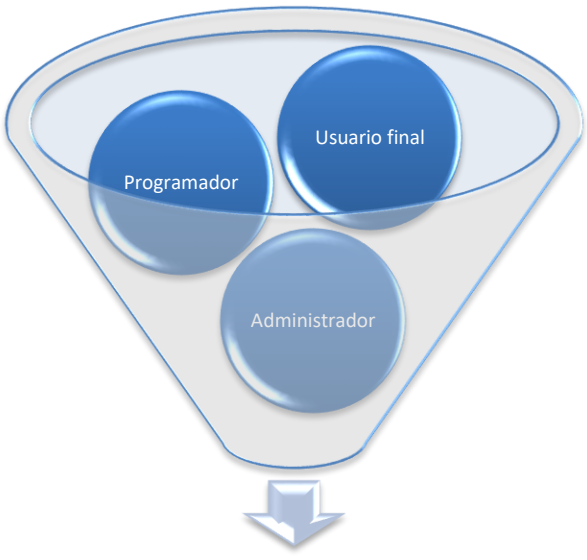
DATOS DE FECHA Y HORA:

- **DATE** → Permite definir datos en formato fecha.
- **TIME** → Permite definir datos en formato hora.
- **DATETIME o TIMESTAMP** → Resulta de la unión de un tipo DATE y de un tipo TIME.

3. TIPOS DE USUARIOS

Los responsables de la empresa se están planteando si todos los empleados de la misma van a hacer el mismo uso de la base de datos. Por ello, te piden que identifiques los diferentes perfiles que hay en la empresa y definas qué función o papel tendrá cada uno de ellos con respecto a la base de datos.

La existencia de una base de datos de tamaño medio requiere un mantenimiento bastante costoso. Por ello, se ponen en marcha bases de datos cuando hay varias personas interesadas en su aprovechamiento.



Usuarios que interactúan con la Base de Datos.

Un grupo de personas que intenta utilizar unos datos genera problemas ya conocidos, como las violaciones de integridad y los posibles errores por acceso simultáneo. Para resolver estos y otros conflictos aparece la figura del administrador de la base de datos (ABD o DBA, Data Base Administrator).

Esta persona posee todos los privilegios o permisos de acceso sobre los datos, y se ocupa de otorgar algunos de ellos a los usuarios para que éstos realicen su trabajo.

Además, las personas (o programas) que acceden a la base de datos reciben el nombre genérico de usuarios. Pero no todos realizan las mismas tareas. Se pueden clasificar en programadores y usuarios finales.

3.1 Programadores

Se trata de personas que no están interesadas propiamente en los datos, sino en desarrollar un programa que trabaje sobre ellos. Este programa será utilizado por un usuario final cuando esté acabado. Para acceder a los datos, su programa dialoga con el sistema gestor de la base de datos (SGBD). Los programadores acceden a los datos al menos para probar la aplicación que están desarrollando.

3.2 Usuarios finales

Son aquellos a los que interesan los datos almacenados en la base de datos, para modificarlos (como el empleado de un banco) o simplemente consultarlos (acceso a fondos de una biblioteca). Estos usuarios tienen dos opciones para trabajar: utilizar un lenguaje conversacional propio del SGBD para el diálogo, o bien aprovechar una aplicación o programa hecho a medida por un programador.

3.3 Administrador

Posee todos los privilegios o permisos de acceso sobre los datos. Sus tareas son:

- **La creación y destrucción de los objetos de la base de datos** (tablas, vistas, usuarios, grupos). Diseña la organización de la base de datos y la pone en marcha para que los usuarios obtengan buen servicio de ella. Creará los esquemas de la base de datos y los dotará de contenido.

- **La autorización de acceso a los objetos**. Tanto un usuario que pregunta por un valor como al ejecutar un programa necesitan permiso para ver los datos. Estos permisos son otorgados por el administrador, de acuerdo con el tipo de usuario que se trate. Por ejemplo, en una biblioteca no disponen de iguales permisos el

encargado que registra los préstamos (y puede hacer anotaciones) que el lector que examina los títulos.

- **La gestión del almacenamiento físico y del espacio en disco**.

- **La política de copias de seguridad y la restauración de la base de datos en caso de caída**.

EJEMPLO PRÁCTICO

Una base de datos de una empresa está dando errores de forma constante a la hora de añadir nueva información. Se debe revisar el almacenamiento para comprobar si se ha agotado la memoria disponible. ¿Quién es el encargado de realizar esta función?

Únicamente el administrador es el encargado de revisar el almacenamiento interno de la base de datos.

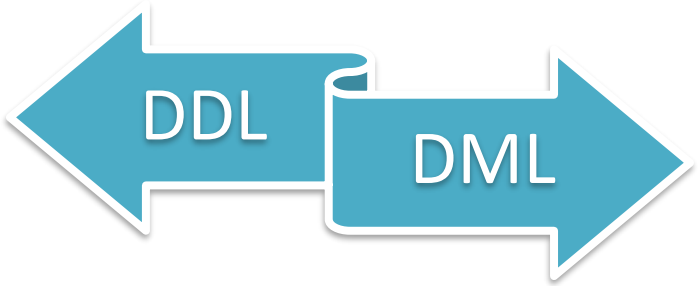
4. TIPOS DE ÓRDENES DEL LENGUAJE SQL

Para acelerar el proceso de creación de la base de datos, se ha incorporado un nuevo miembro al equipo de desarrollo. Como esta persona tiene poca experiencia, Paula te pide que la pongas al día y le expliques cuáles son los diferentes tipos de órdenes o lenguajes que componen SQL.

Existen dos fases en la vida de una base de datos: la etapa de preparación y puesta en marcha, y la etapa de explotación. Esta última es el objetivo de todo el sistema, y las tareas preparatorias se realizan antes de entrar en esa fase de utilidad para la organización.

Un lenguaje para manejar bases de datos se compone de un conjunto o repertorio de instrucciones, al igual que en un lenguaje de programación habitual. Debido a la clasificación de las tareas que se realizan sobre la base de datos en dos grupos, se suelen distinguir dos sublenguajes, cada uno de los cuales sirve para un tipo de actividad diferente.

En particular, en los lenguajes sobre bases de datos relacionales se distinguen dos partes:



Usuarios que interactúan con la Base de Datos.

4.1 DDL (Data Description Language o Data Definition Language)

Es el **lenguaje de descripción de datos o lenguaje de definición de datos**, utilizado para la creación (y mantenimiento) de la estructura de la base de datos. Con este lenguaje se definen y modifican los esquemas de las relaciones, se crean o destruyen índices y se eliminan relaciones. Además, pueden definirse vistas y permisos de acceso para los usuarios. Permite definir los subesquemas externos, el esquema conceptual y el esquema interno. De este modo, se detalla:

- La descripción de subesquemas externos define la vista particular para cada aplicación o usuario; indica las tablas y vistas disponibles, ocultando campos, mostrando campos calculados, etc.
- La descripción del esquema conceptual detalla entidades, atributos y relaciones, sin hacer referencia a la forma de almacenamiento.
- La descripción del esquema interno detalla la estructura de almacenamiento interno, los modos de organización y acceso, etc.

El resultado de la ejecución de las instrucciones de este lenguaje, para cualquiera de los tres niveles, se almacena en el diccionario de datos, que es una metabase de datos que contiene información de la base de datos. Además, DDL cuenta con un sublenguaje encargado del control y seguridad de los datos, que se denomina Lenguaje de Control de Datos (DCL), y define los privilegios, tipos de acceso, etc.

4.2 DML (Data Manipulation Language)

Es el **lenguaje de manipulación de datos**, que incluye las instrucciones para consultar la base de datos, así como para insertar o eliminar tuplas y modificar valores de datos. Este lenguaje es el utilizado para la fase de explotación o de trabajo útil de la base de datos, y es empleado por los programadores y usuarios finales. Estas operaciones pueden ser realizadas por usuarios expertos que conocen tanto este lenguaje como el modelo de datos (tablas, campos, relaciones, etc.), pero también es posible realizar estas

operaciones a través de peticiones ejecutadas desde programas escritos con lenguajes convencionales que incluyen sentencias de manipulación de datos como parte de su lógica.

PARA SABER MÁS

Para saber más información sobre las sentencias DDL y DML, accede a este enlace:

4.3 DCL (Data Control Language)

El Lenguaje de Control de Datos, también conocido como DCL por sus siglas en inglés, es un conjunto de comandos y sentencias utilizados para administrar los permisos de acceso y controlar la seguridad en los sistemas gestores de bases de datos (SGBD), lo que permite gestionar los derechos de acceso y los privilegios de los usuarios en una base de datos. Para ello, proporciona una serie de comandos que permiten a los administradores de la base de datos conceder o revocar permisos a los usuarios para realizar distintas operaciones, como consultar, modificar o eliminar datos en la base de datos.

Los principales comandos son:

- **GRANT**: se utiliza para otorgar permisos específicos a los usuarios, de tal modo que se pueden conceder los privilegios de SELECT (consulta), UPDATE (actualización) o DELETE (borrado) sobre ciertas tablas a un usuario en concreto. Por ejemplo, para conceder el permiso de consulta sobre una tabla llamada Empleado al usuario Usuario1 procederíamos de la siguiente manera: **GRANT SELECT ON Empleado TO Usuario1**. Para otorgar más de un permiso a la vez, estos deben ser separados por comas. Si se quieren asignar todos los privilegios se puede usar la opción ALL PRIVILEGES, por ejemplo: **GRANT** ALL PRIVILEGES **ON Empleado TO Usuario1.**

- **REVOKE**: usado para revocar los permisos previamente concedidos a los usuarios. Por ejemplo, para revocar el permiso concedido en el apartado anterior: **REVOKE SELECT ON Empleado FROM Usuario1.**

Para crear un usuario en MySQL podemos usar la siguiente sentencia:

CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña_usuario';
En este ejemplo, estamos creando un usuario llamado "nombre_usuario" con la contraseña "contraseña_usuario". El "@'localhost'" indica que este usuario solo podrá acceder desde el mismo equipo donde se encuentra la base de datos. Puedes cambiar "localhost" por la dirección IP o el nombre del host si deseas que el usuario pueda acceder desde otros equipos en la red.

EJEMPLO PRÁCTICO

Hoy en la oficina acaba de incorporarse un nuevo compañero y te han pedido que crees un usuario llamado Antonio con acceso a la base de datos local y dale permisos de lectura y borrado sobre la tabla Clientes. Por último, quítale el permiso de borrado.

```
CREATE USER 'Antonio'@'localhost' IDENTIFIED BY 'pwdant';
                              GRANT DELETE, SELECT ON clientes TO
Antonio@localhost;
                              REVOKE DELETE ON clientes FROM
Antonio@localhost;
```

El DCL es una parte esencial de la administración de bases de datos, ya que permite controlar quién tiene acceso a los datos y qué tipo de operaciones pueden realizar. Esto permite a los administradores de bases de datos (DBA) mantener la seguridad e integridad de dichas bases de datos, asegurando que solo los usuarios autorizados puedan acceder o manipular cierta información.

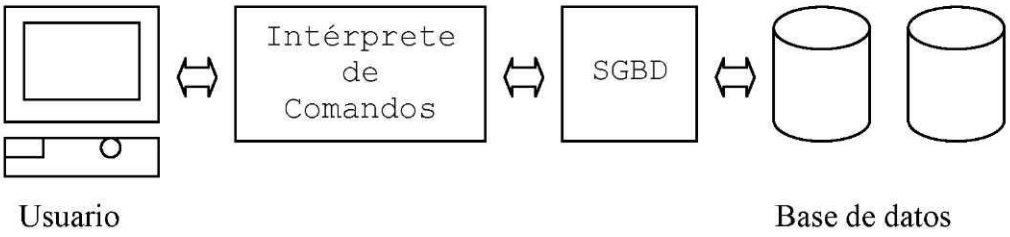
5. FORMAS DE TRABAJAR CON SQL

Como bien sabes, existen diferentes maneras de trabajar con SQL. Debes debatir con tu equipo sobre las mismas para decidir qué forma usaréis en la empresa para trabajar con SQL.

El lenguaje SQL se utiliza para actuar sobre una base de datos de alguno de estos modos: modo interactivo y desde un programa. En el primero de ellos se dispondrá de algún tipo de terminal o consola que permita interactuar a través de comandos propios del lenguaje SQL. En el segundo de ellos, es el propio software el encargado de generar las llamadas y consultas SQL al SGBD que lo implementa.

5.1 Modo interactivo

Desde un terminal se establece una conversación entre el usuario y el programa gestor de la base de datos (SGBD). Es similar al modo en que se mantiene un diálogo con un sistema operativo. Cualquier orden SQL, sea DDL, DML u otra, puede introducirse por este método, y no hay restricciones en el orden entre ellas (pueden mezclarse consultas con creaciones de tablas, por ejemplo).



Esquema de trabajo en modo interactivo.

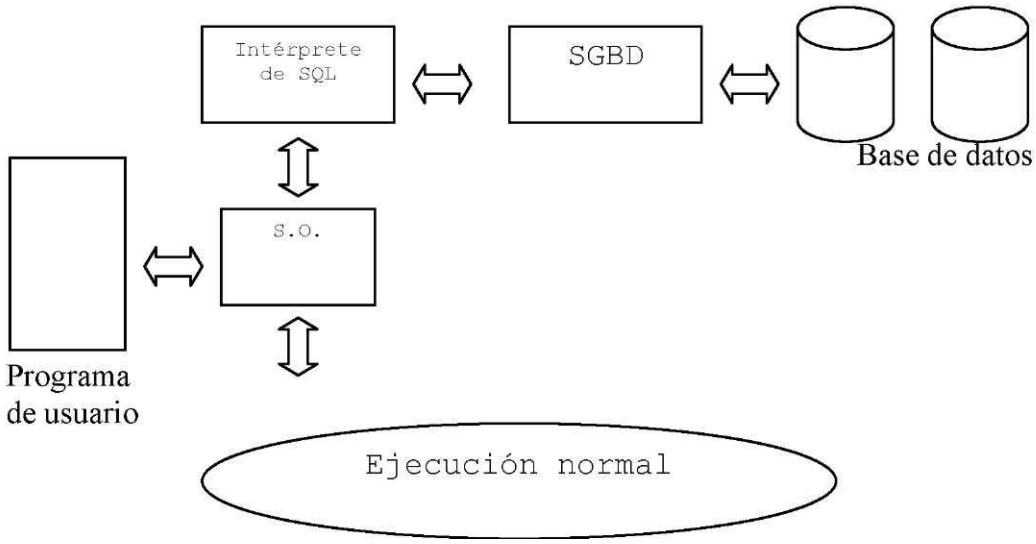
5.2 Desde un programa

Se ejecuta sobre el sistema operativo una aplicación. Esta puede ser de 2 tipos:

- Un programa escrito íntegramente en SQL. Existen extensiones a SQL que lo dotan de las estructuras de programación imperativa usuales (bucles, selección). Se trata de un lenguaje procedural que permite desarrollar programas imperativos que acceden a la base de datos vía SQL.

- Un programa escrito en un lenguaje convencional de programación parte de cuyo texto está escrito en SQL.

- A esta variante de utilización del lenguaje se le llama SQL inmerso o embebido y al lenguaje que contiene el texto se le conoce como lenguaje huésped (host). Las instrucciones del lenguaje de programación se ejecutan por el procedimiento usual, y las instrucciones SQL se le pasan a un módulo especial de ejecución del SGBD. Puede verse un esquema en el siguiente gráfico. Una implementación de SQL embebido establece las relaciones que deben mantener los objetos de la base de datos con los objetos del programa huésped, así como ciertas restricciones de funcionamiento.



Esquema de trabajo desde un programa.

El modo de trabajo con SQL programado aún admite dos variantes:

- **SQL estático**. El programa no admite cambios durante la ejecución. Éste es el método usado en la mayoría de las aplicaciones.

- **SQL dinámico**. Si durante la ejecución varía algún parámetro o parte del programa, se necesita utilizar SQL dinámico. Es menos eficiente que un programa en SQL estático y utiliza técnicas dinámicas de manejo de variables, lo que hace su uso más difícil para el programador.

ENLACE DE INTERÉS

Esta web recoge información sobre SQL dinámico y estático, accede a ella para conocer más información:

6. ENTORNO DE TRABAJO

Llegados a este punto, antes de empezar a crear las tablas que compondrán la base de datos, debes decidir las herramientas y arquitectura que usaréis para llevar a cabo el proyecto.

Para el uso del lenguaje SQL es necesario disponer de un entorno de trabajo. Éste se compone de varios elementos que se describen a continuación. Un usuario de una base de datos bajo SQL debe disponer de:

Sistema operativo. La máquina en que el usuario realiza su trabajo se denomina local. Lo usual es que una base de datos SQL se active sobre una plataforma multiusuario con varios puestos de trabajo. El ordenador local no suele tener instalado todo el SGBD, sino que estará distribuido entre varias máquinas. Por tanto, el usuario deberá tener acceso al sistema operativo o red que se utilice.

Conexión con la base de datos. Una vez que el usuario se ha conectado al sistema operativo, debe conectarse a la base de datos.

Acceso a la base de datos. Debe existir como usuario autorizado en la base de datos.

Utilidades. Se trata de programas diseñados para realizar operaciones sobre la base de datos. La utilidad más básica es un terminal de texto en el que introducir órdenes SQL.

Útiles de administración del sistema. El ABD utilizará para su trabajo algunas herramientas que están vedadas al resto de usuarios, como programas que evalúen el rendimiento del sistema o aplicaciones para la gestión de usuarios.

Compiladores. El usuario puede trabajar con SQL sólo en modo interpretado, para lo cual le basta con los útiles enumerados anteriormente. Pero si desarrolla o simplemente utiliza un programa ya desarrollado por un programador puede que necesite compilarlo.

Elementos de un entorno de trabajo.

6.1 SQL y la interconexión por la red

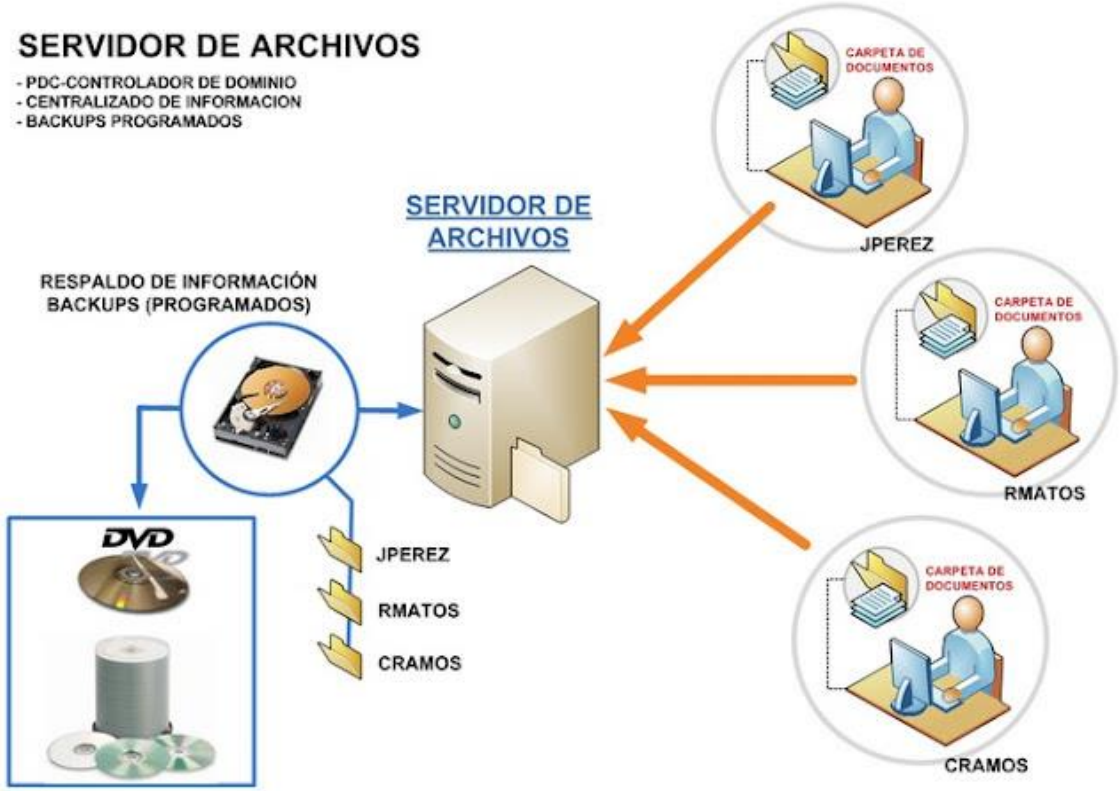
La creciente popularidad de la interconexión de computadoras por red durante los últimos años ha tenido un fuerte impacto en la gestión de base de datos y ha dado a SQL una nueva importancia. En esta arquitectura, tanto el SGBD como los datos físicos residen en un mainframe o minicomputadora central, junto con el programa de aplicación que acepta entradas desde el terminal de usuario y muestra los datos en la pantalla del usuario.

Un caso puede ser al teclear una consulta que requiere una búsqueda secuencial en una base de datos, como por ejemplo la petición para hallar la cantidad media de mercancías de todos los pedidos. El SGBD recibe la consulta, explora la base de datos para acceder a cada uno de los registros de datos del disco, calcula el promedio y muestra el resultado en la pantalla terminal. Tanto el procesamiento de la aplicación como el procesamiento de la base de datos se producen en la computadora central, y como el sistema es compartido por muchos usuarios, cada usuario experimenta una degradación del rendimiento cuando el sistema está más cargado.

6.2 Arquitectura servidora de archivos

En esta arquitectura, una aplicación que se ejecuta en una computadora personal puede acceder de forma transparente a los datos localizados en un servidor de archivos, donde se almacenan los archivos compartidos. Cuando la aplicación de PC solicita los datos de un archivo compartido, el software de red recupera automáticamente el bloque solicitado del archivo en el servidor.

Esta arquitectura proporciona un rendimiento excelente para consultas típicas, ya que cada usuario dispone de la potencia completa de una computadora personal ejecutando su propia copia del SGBD.



Arquitectura servidora de archivos.
Fuente: http://aquitecturasoftware.blogspot.com/p/servidor-de-archivos.html

Esta arquitectura proporciona un rendimiento excelente para consultas típicas, ya que cada usuario dispone de la potencia completa de una computadora personal ejecutando su propia copia del SGBD.

6.3 Arquitectura Cliente/Servidor

En esta arquitectura las computadoras personales están combinadas en una red de área local junto con un servidor de base de datos que almacena las bases de datos compartidas. Las funciones del SGBD están divididas en dos partes. Los frontales (fronts-ends) de base de datos, tales como herramientas de consulta interactiva, generadores de informe y programas de aplicación, se ejecutan en la computadora personal. El motor de soporte (back-end) de la base de datos que almacena y gestiona los datos se ejecuta en el servidor. SQL se ha convertido en el lenguaje de base de datos estándar para comunicación entre las herramientas frontales y el motor soporte de esta arquitectura.

En la arquitectura cliente/servidor, la consulta viaja a través de la red hasta el servidor de base de datos como una petición SQL. El motor de base de datos el servidor procesa la petición y explora la base de datos, que también reside en el servidor. Cuando se calcula el resultado, el motor de la base de datos envía de vuelta, a través de la red, una única contestación a la petición inicial, y la aplicación frontal la muestra en la pantalla del PC.

La arquitectura cliente/servidor reduce el tráfico de red y divide la carga de trabajo de la base de datos. Las funciones íntimamente relacionadas con el usuario, tales como el manejo de entradas y la visualización de datos, se concentran en el PC. Las funciones de intenso procesamiento de datos, tales como la entrada/salida de archivos y el procesamiento de consultas, se concentran en el servidor de la base de datos. Lo que es más importante, el lenguaje SQL proporciona un interfaz bien definido entre los sistemas frontales y de soporte, comunicando las peticiones de acceso a la base de datos de una manera eficiente. Las implementaciones de SQL Server, Oracle, Informix e Ingres para LAN de PC y SQLBase de Gupta Technologies utilizan este enfoque.

PARA SABER MÁS

Para ampliar información acerca de la arquitectura cliente servidor, lee este post:

7. CREACIÓN Y ELIMINACIÓN DE BASES DE DATOS

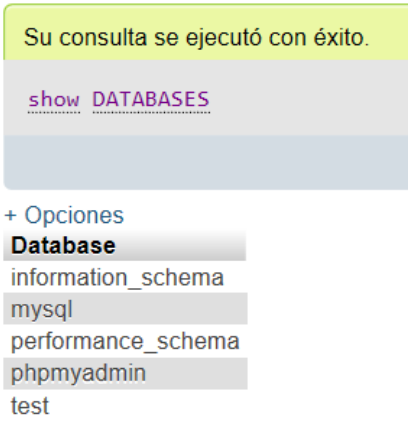
Ya tienes todo lo necesario para comenzar con la creación de las tablas que formarán la base de datos. Para ello, sabes que previamente debes haber creado la base de datos en sí misma.

Para crear una base de datos, se debe estar conectado como administrador del sistema o root (o tener el permiso de utilizar CREATE DATABASE). Mediante el uso de esta instrucción seguido del nombre de la base de datos podemos crear dicha base de datos.

```
CREATE DATABASE Concesionario
```

Creación de la base de datos Concesionario en phpMyAdmin.

Desde línea de comandos se pueden ver las bases de datos existentes, mediante el comando **show databases.**



Ejecución del comando show databases en phpMyAdmin.

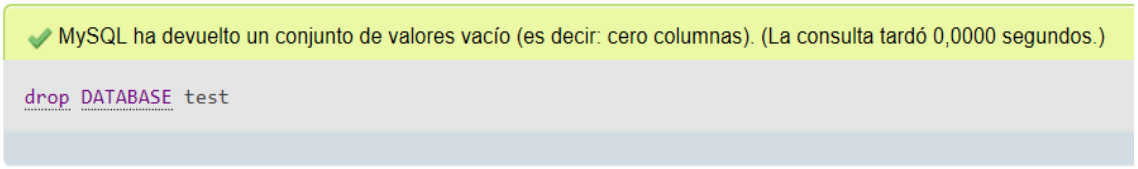
También se pueden crear bases de datos utilizando cualquiera de los programas MySQL Administrator o MySQL Query Browser.

PARA SABER MÁS

Amplía información sobre estas herramientas de MySQL:

También se pueden crear bases de datos utilizando MySQL Workbench, que permite la definición del modelo y la manipulación de los datos.

Para eliminar una base de datos desde la línea de comando, debe ejecutarse la instrucción **DROP DATABASE**, indicando el nombre de la base de datos a borrar.



Ejecución del comando DROP DATABASE en phpMyAdmin.

8. MODIFICACIÓN DE BASES DE DATOS

Una vez creadas todas las tablas en la base de datos, te das cuenta de que habéis cometido algunos errores que debéis subsanar. Para ello, en algunos casos decides modificar algunas de las tablas mientras que, para otros casos, decides que lo mejor es eliminar la tabla y crearla nuevamente.

La modificación de bases de datos implica entender una serie de funcionalidades como son la definición de tablas a través de la cláusula CREATE TABLE, la eliminación de tablas a través de DROP TABLE y la modificación de tablas según la cláusula ALTER TABLE.

8.1 Definición de tablas (CREATE TABLE)

La estructura más importante de una base de datos relacional es la tabla. En una base de datos multiusuario, las tablas principales son típicamente creadas una vez por el administrador de la base de datos y utilizadas luego día tras día.

Creación de una tabla: La sentencia CREATE TABLE crea una nueva tabla en la base de datos y la prepara para aceptar datos. Las diferentes cláusulas de la sentencia especifican los elementos de la definición de la tabla.

La creación de una nueva tabla es relativamente sencilla. Cuando se ejecuta una sentencia CREATE TABLE, uno se convierte en el propietario de la tabla recién creada, a la cual se le da el nombre especificado en la sentencia. El nombre de la tabla debe ser un nombre SQL legal y no debe entrar en conflicto con el nombre de alguna otra tabla ya existente. La tabla recién creada está vacía, pero el DBMS la prepara para aceptar datos, añadidos con la sentencia INSERT.

VÍDEO DE INTERÉS

En el siguiente vídeo se puede ver el funcionamiento de la sentencia CREATE DATABASE de una forma práctica:

Definiciones de columnas: Las columnas de la tabla recién creada se definen en el cuerpo de la sentencia CREATE TABLE. Las definiciones de columnas aparecen en una lista separada por comas e incluida entre paréntesis. El orden de las definiciones de las columnas determina el orden de izquierda a derecha de las columnas en la tabla. Cada definición especifica:

El ***nombre de la columna***, que se utiliza para referirse a la columna en sentencias SQL. Cada columna de la tabla debe tener un nombre único, pero los nombres pueden ser iguales a los de las columnas de otras tablas.

El ***tipo de dato*** de los valores que tomarán los registros en dicha columna (int, varchar, byte,…).

Las ***restricciones*** asociadas a dicha columna (clave primaria, foránea, valor no nulo, etc).

Definiciones de columnas.

La sintaxis de la sentencia es como sigue:

```
CREATE TABLE nombre_tabla (
    nombrecolumna        tipo_columna    [restricciones]
    [, nombrecolumna .......]
    [, restricciones]);
```

Las restricciones establecen reglas de integridad (UNIQUE, REFERENCES, NULL, DEFAULT...) y pueden indicarse junto a la columna correspondiente, o bien al final de la tabla.

EJEMPLO PRÁCTICO

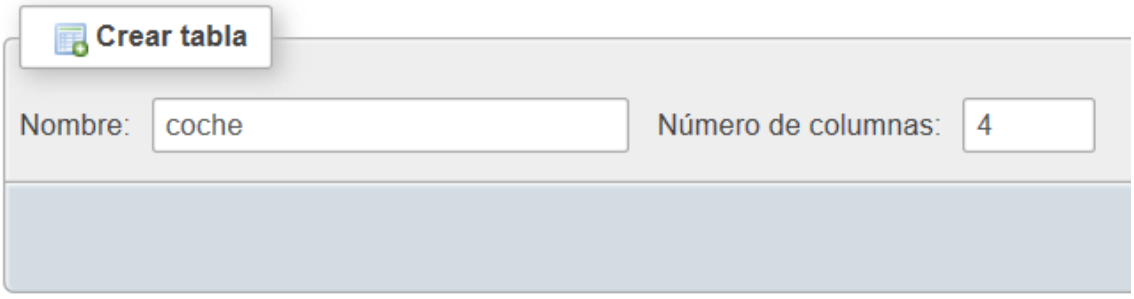
¿Cómo sería la definición de la tabla DEPARTAMENTOS?

```
CREATE TABLE DEPARTAMENTOS (
    COD INTEGER           PRIMARY KEY,
    NOMBRE  VARCHAR(45)     NOT NULL,
    DIR     VARCHAR(45),
    CIUDAD  VARCHAR(45),
    TLF     VARCHAR(9));
```

PARA SABER MÁS

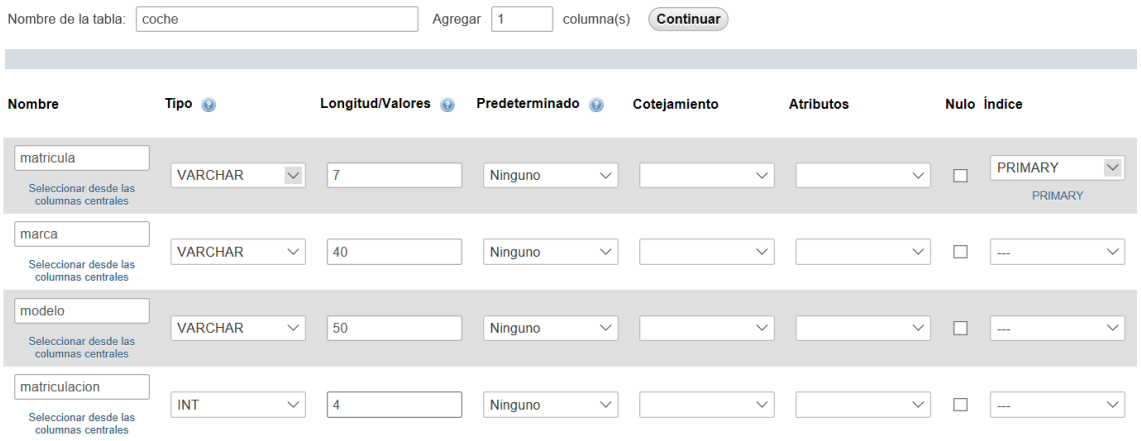
Para ampliar más información sobre la sentencia CREATE TABLE se puede consultar el manual de referencia en la web de MySQL:

En phpMyAdmin se puede crear una nueva tabla sobre la base de datos que se tenga seleccionada, pulsando sobre la propia base de datos y escribiendo el nombre de la tabla y el número de columnas que tendrá. Por ejemplo, la tabla coche que tendrá 4 columnas:



Crear tabla en phpMyAdmin.

Se abrirá una ventana donde se indicará el nombre de la tabla y aparecerán el número de columnas indicadas anteriormente para añadir la información de cada columna (nombre, tipo de dato, PRIMARY KEY, NOT NULL, etc.).



Añadir información a las columnas.

Finalmente, para validar los cambios, se debe pulsar sobre Guardar, tras lo cual se mostrará una ventana con la consulta SQL que se realizará en el SGBD.

VALORES POR OMISIÓN (DEFAULT): En consultas con valores NOT NULL como restricciones para algunos atributos obliga a que cuando se inserte un nuevo registro, dichos campos no se dejen sin rellenar. Como ya se conoce, las claves primarias siempre tienen esta restricción.

DEFINICIONES DE CLAVE PRIMARIA Y AJENA: Además de la definición de las columnas de una tabla, la sentencia CREATE TABLE identifica la clave primaria de la tabla y las relaciones de la tabla con otras tablas de la base de datos.

RECUERDA

La clave primaria se utiliza para identificar en forma única cada línea en la tabla. Puede ser parte de un registro real o puede ser un campo artificial (no tiene nada que ver con el registro real). Una clave primaria puede consistir en uno o más campos en una tabla. Cuando se utilizan múltiples campos como clave primaria, se los denomina claves compuestas.

La cláusula PRIMARY KEY especifica la columna o columnas que forman la clave primaria de la tabla. Esta columna (o combinación de columnas) sirve como identificador único para cada fila de la tabla. El DBMS requiere automáticamente que el valor de clave primaria sea único en cada fila de la tabla. Además, la definición de columna para todas las columnas que forman la clave primaria debe especificar que la columna es NOT NULL.

Se debe recordar que, si la clave principal está compuesta únicamente por un atributo, entonces la restricción de NOT NULL puede ir directamente aplicada sobre la columna

que hace de clave principal. Sin embargo, si la clave principal está compuesta por varias columnas, entonces la restricción de clave primaria irá tras la definición de la última columna, como restricción PRIMARY KEY (columna1, columna2, …). En el ejemplo anterior, la tabla fue creada de esta forma.

La cláusula FOREIGN KEY especifica una clave ajena en la tabla y la relación que crea con otra tabla (padre) de la base de datos. La cláusula especifica:

- La columna o columnas que forman la clave ajena, todas las cuales son columnas de la tabla que está siendo creada.
- La tabla que es referenciada por la clave ajena. Esta es la tabla padre en la relación; la tabla que está siendo definida es la hija.
- Un nombre opcional para la relación. El nombre no se utiliza en ninguna sentencia SQL, pero puede aparecer en mensajes de error y es necesaria si se desea poder suprimir la clave ajena posteriormente.
- Cómo debe tratar el DBMS un valor NULL en una o más columnas de la clave ajena, cuando la compare con las filas de la tabla padre.
- Una restricción de comprobación opcional que restrinja los datos de la tabla para que sus filas encuentren una condición de búsqueda especificada.

En phpMyAdmin se puede realizar una clave ajena de la siguiente forma:

Ejecutar la(s) consulta(s) SQL en la base de datos test:

```
1 CREATE TABLE conductor (
2     DNI VARCHAR(10) PRIMARY KEY,
3     NOMBRE VARCHAR(50) NOT NULL,
4     MATRICULA VARCHAR(7) NOT NULL,
5     FOREIGN KEY (MATRICULA) REFERENCES coche(matricula));
```

Creación de clave ajena en tabla conductor hacia tabla coche.

Como se puede observar, es necesario tener creada la tabla a la cual se va a hacer la referencia. En este caso, ya estaba creada la tabla coche (matrícula, marca, modelo, año de matriculación).

Se ha creado una nueva tabla conductor con las columnas (DNI, nombre, matrícula) y una clave ajena entre las columnas matrícula de ambas tablas a través de la cláusula FOREIGN KEY.

CONSTRAINT: Palabra clave que indica el principio de la definición de una restricción PRIMARY KEY, UNIQUE, FOREIGN KEY o CHECK. Las restricciones son propiedades especiales que exigen la integridad de los datos y crean tipos especiales de índices para la tabla y sus columnas. Al crear una restricción ésta debe tomar un nombre. Los nombres de restricción deben ser únicos en una base de datos. Se usan prefijos para dar más claridad a las restricciones; así las claves primarias comenzarán con el prefijo PK y para las claves foráneas FK.

```
CREATE TABLE PERSONA (
DNI VARCHAR(9) NOT NULL,
nombre VARCHAR(50),
CONSTRAINT PK_DNI PRIMARY KEY (DNI));

CREATE TABLE EMPLEADO (
DNI VARCHAR(9) NOT NULL,
nombre VARCHAR(50),
CONSTRAINT FK_DNI FOREIGN KEY (DNI) REFERENCES persona(DNI));
```

PRIMARY KEY: Es una restricción que exige la integridad de entidad para una o varias columnas dadas a través de un índice único. Sólo se puede crear una restricción PRIMARY KEY por cada tabla.

UNIQUE: Es una restricción que proporciona la integridad de entidad para una o varias columnas dadas a través de un índice único. Una tabla puede tener varias restricciones UNIQUE.

FOREIGN KEY...REFERENCES: Es una restricción que proporciona integridad referencial para los datos de la columna o columnas. Las restricciones FOREIGN KEY requieren que cada valor de la columna exista en la columna de referencia correspondiente de la tabla a la que se hace referencia. Las restricciones FOREIGN KEY pueden hacer referencia sólo a columnas que sean restricciones PRIMARY KEY o UNIQUE en la tabla de referencia. Es decir, forzosamente una clave foránea de una tabla hace referencia a la clave primaria de otra tabla.

CHECK: Es una restricción que exige la integridad del dominio al limitar los valores posibles que se pueden escribir en una o varias columnas. Las restricciones CHECK se conocen también como 'restricciones de valores', ya que los valores introducidos en las columnas se deben verificar. Se debe comentar en este punto que MySQL aún no soporta las restricciones de tipo CHECK, aunque sí soporte la introducción de éstas. Es decir, se pueden crear restricciones de tipo CHECK desde línea de comandos, pero el motor no las verificará.

PARA SABER MÁS

Una de las restricciones más difíciles de entender son las restricciones FOREIGN KEY. Consultas estos ejemplos y sintaxis que pueden resultar de interés:

EJEMPLO PRÁCTICO

Realiza una restricción tipo CHECK para un salario positivo.

```
CONSTRAINT CHK_salario_positivo CHECK (salario > 0)
```

ARTÍCULO DE INTERÉS

Para indicar restricciones lee este interesante artículo:

8.2 Eliminación de tablas (DROP TABLE)

Con el tiempo la estructura de una base de datos crecerá y se modificará. Nuevas tablas serán creadas para representar nuevas entidades, y algunas tablas antiguas ya no serán necesarias. Se puede eliminar de la base de datos una tabla que ya no es necesaria con la sentencia **DROP TABLE** seguido del nombre de la tabla.

Normalmente se eliminará una de las tablas propias del usuario y se utilizará un nombre de tabla no cualificado. Con el permiso adecuado, también se puede eliminar una tabla propiedad de otro usuario especificando un nombre de tabla cualificado.

Cuando la sentencia DROP TABLE suprime una tabla de la base de datos, su definición y todos sus contenidos se pierden. No hay manera de recuperar los datos, y habría que utilizar una nueva sentencia CREATE TABLE para volver a crear la definición de la tabla. Debido a sus serias consecuencias, debe utilizarse la sentencia DROP TABLE con mucho cuidado. La supresión de una tabla suprimirá los datos y los índices asociados. La supresión no será posible si la tabla es referenciada por una clave externa.

Se debe tener cuidado al eliminar una tabla pues una vez eliminada ésta no se puede recuperar. Como se ha comentado antes, si la tabla que se desea eliminar está referenciada por otra (como la tabla PERSONA, referenciada por la tabla EMPLEADOS a través de la clave ajena FK_DNI) se producirá un error al intentar realizar la acción.

RECUERDA

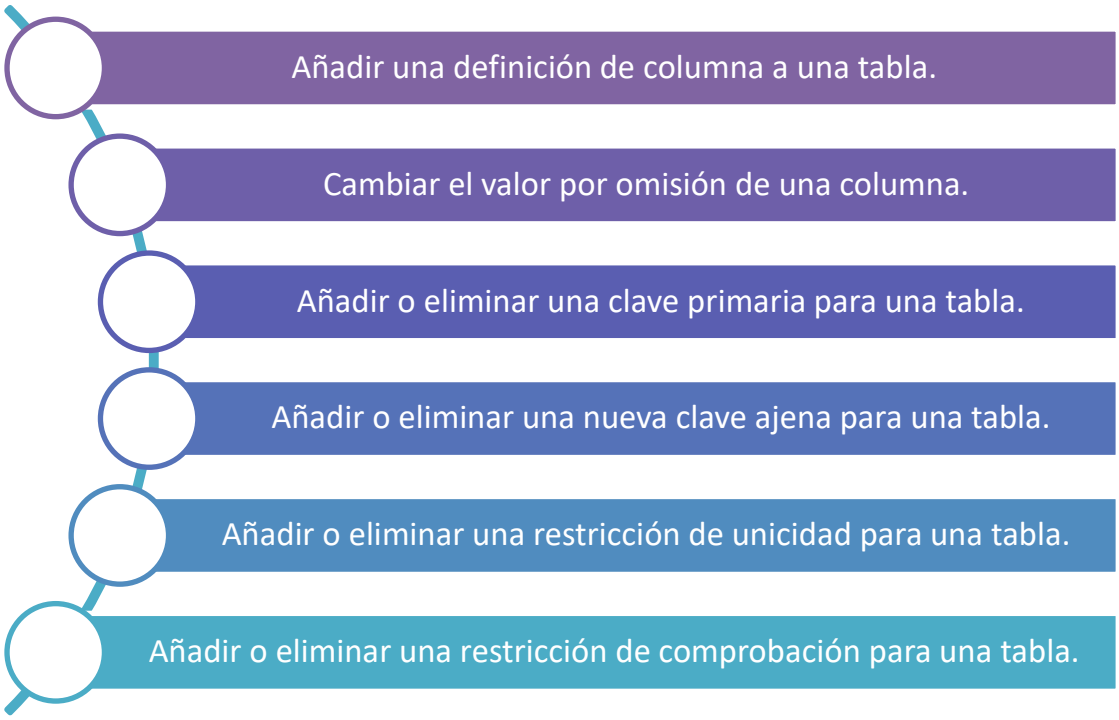
En la web w3schools se dispone de un amplio manual con sintaxis y ejemplos sobre el lenguaje SQL.

8.3 Modificación de tablas (ALTER TABLE)

Después de que una tabla ha sido utilizada durante algún tiempo, los usuarios suelen descubrir que desean almacenar información adicional con respecto a las entidades representadas en la tabla. En una base de datos de ejemplo formada por empleados, se podría desear:

- Añadir la comisión que cobran los empleados por sus ventas.
- Hacer que un campo sea numérico en lugar de poder introducir texto.
- Eliminar un campo de la tabla.

Cada uno de estos cambios, y algunos otros, pueden ser realizados con la sentencia ALTER TABLE. Al igual que con la sentencia DROP TABLE, ALTER TABLE se utilizará normalmente sobre tablas propias. Con el permiso adecuado, sin embargo, se puede especificar un nombre de tabla cualificado y alterar la definición de la tabla de otro usuario. La sentencia ALTER TABLE puede:



Funcionalidad de ALTER TABLE.

La sentencia ALTER TABLE nombre_tabla va seguida de dos posibles palabras reservadas: ADD o DROP. La primera de ellas permite añadir mientras que la segunda eliminar. Como se ha visto anteriormente, las tablas están formadas por columnas y restricciones. Entonces éstos serán los posibles elementos a modificar en una tabla. Por tanto, se tienen las siguientes posibilidades:

- Añadir una columna:
  - ALTER TABLE NombreTabla ADD COLUMN...
- Eliminar una columna:
  - ALTER TABLE NombreTabla DROP COLUMN…
- Añadir una restricción:
  - ALTER TABLE NombreTabla ADD CONSTRAINT…
- Eliminar una restricción:
  - ALTER TABLE NombreTabla DROP CONSTRAINT…

Cada una de las cláusulas de la sentencia ALTER TABLE puede aparecer sólo una vez en la sentencia. Se puede añadir una columna y definir una clave ajena en una única sentencia ALTER TABLE. Asimismo, también podemos añadir más de una columna en la misma sentencia ALTER TABLE.

EJEMPLO PRÁCTICO

Creación de una tabla para, posteriormente, añadir dos nuevos campos con la sentencia ALTER TABLE

```
CREATE TABLE Coche(
   Matricula VARCHAR(7) PRIMARY KEY,
   Marca VARCHAR(15) NOT NULL
   Marca VARCHAR(15) NOT NULL);
ALTER TABLE Coche
ADD COLUMN Modelo VARCHAR(15),
ADD COLUMN Precio DECIMAL(8,2);
```

ENLACE DE INTERÉS

En este enlace dispones de un simulador que consiste en un área de texto en la que podrás probar tus sentencias y ver si funcionan:

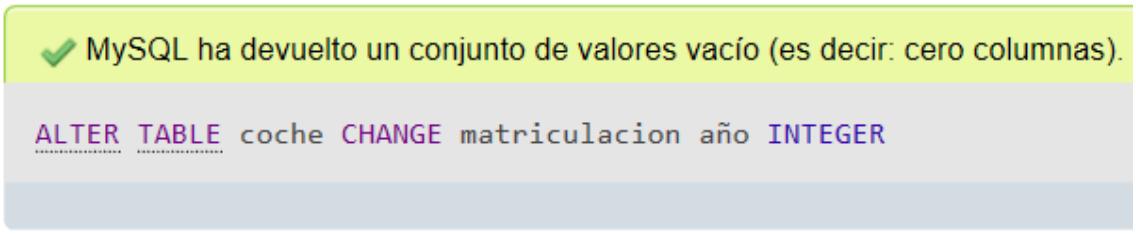
Añadir una columna:

El uso más común de la sentencia ALTER TABLE es añadir una columna a una tabla existente. La cláusula de definición de columna en la sentencia ALTER TABLE es prácticamente idéntica a la de la sentencia CREATE TABLE y funciona del mismo modo. La nueva columna se añade al final de las definiciones de la columna de la tabla y aparece como la columna más a la derecha en consultas posteriores. El DBMS asume normalmente un valor NULL para la columna recién añadida en todas las filas existentes de la tabla. Si la columna se declara NOT NULL con un valor por omisión, el DBMS supone que el valor por omisión es el del tipo de datos de la columna.

Observa que no puede declararse simplemente la nueva columna NOT NULL, ya que el DBMS asumiría valores NULL para la columna en las filas existentes, violando inmediatamente la restricción.

Por ejemplo: ALTER TABLE EMPLEADOS ADD COMISION DOUBLE;

En phpMyAdmin es posible hacer uso de ALTER TABLE a través de cualquiera de sus posibilidades. Por ejemplo, cambiar el nombre de una columna a la tabla coche: ALTER TABLE coche CHANGE matriculacion año INTEGER.



ALTER TABLE en phpMyAdmin.

Modificación de claves primaria y ajena:

El otro uso habitual para la sentencia ALTER TABLE es cambiar o añadir definiciones de clave primaria y clave ajena a una tabla.

Las cláusulas que añaden definiciones de claves primaria y ajena son exactamente las mismas que las de la sentencia CREATE TABLE y funcionan del mismo modo.

Sólo se puede eliminar una clave ajena si la relación que crea tuvo asignado un nombre originalmente. Si la relación no tiene nombre, no hay manera de especificarla en la sentencia ALTER TABLE. En este caso no se puede suprimir la clave ajena a menos que se elimine y vuelva a crear la tabla utilizando el procedimiento descrito para suprimir una columna.

Si quisiéramos añadir por ejemplo una clave ajena, la sentencia quedaría de la siguiente manera:

ALTER TABLE Pedidos ADD FOREIGN KEY (ID_Empleado) REFERENCES Empleado(ID);
De esta manera estamos creando una clave ajena pero no estamos indicando ningún nombre por lo que, en un futuro, no podríamos eliminarla salvo borrando la tabla completa. Para indicar el nombre para dicha FK podríamos hacer los siguiente:

ALTER TABLE Pedidos

ADD CONSTRAINT **FK_Empleado**

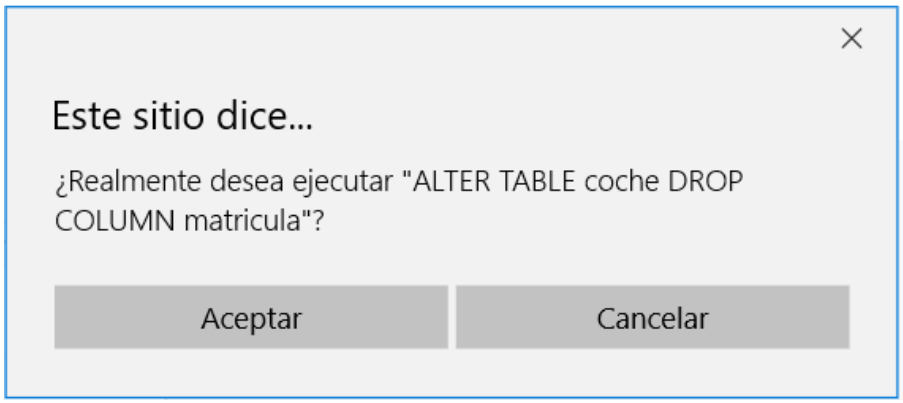
FOREIGN KEY (ID_Empleado) REFERENCES Empleado(ID);

Por otro lado, si quisiéramos eliminar esta clave ajena, la sentencia quedaría de la siguiente manera: ALTER TABLE Pedidos DROP CONSTRAINT **FK_Empleado**;

Eliminar una columna:

Eliminar una columna de una tabla es tan sencillo como ejecutar la sentencia: ALTER TABLE NombreTabla DROP COLUMN NombreColumna;

Al eliminar una columna, es muy posible que el programa quiera corroborar si es cierto. Para ello realizará una consulta al usuario para proceder o no al borrado. La siguiente ilustración muestra un ejemplo de cómo phpMyAdmin realiza esta acción:



Pregunta de seguridad previa al borrado de una columna.

ENLACE DE INTERÉS

Lee este post donde se puede encontrar una amplia guía de esta sentencia:

Modificar el tipo de una columna:

Se supone que se quiere evitar la introducción de caracteres distintos a los numéricos que conforman un número de teléfono.

Añadir una restricción:

Como ya se ha citado, existen restricciones de clave primaria, de clave foránea y de valores. En cualquiera de los casos la restricción se añade mediante la sentencia: ALTER TABLE NombreTabla ADD CONSTRAINT NombreRestriccion;

A continuación, se añadirán las condiciones propias de cada restricción:

CLAVE PRIMARIA
ALTER TABLE NombreTabla
ADD CONSTRAINT PK_NOMBRETABLA PRIMARY KEY (CAM1, CAM2, ETC);
Donde (CAM1, CAM2, ETC) son el conjunto de campos, separados por comas, que conforman la clave primaria.

CLAVE FORÁNEA
ALTER TABLE NombreTabla
ADD CONSTRAINT FK_NomTabla_NomRelac FOREIGN KEY (CF1_FK, CF2_FK, …)
REFERENCES NombreTablaRef (CP1, CP2, …);
Donde 'FK_NomTabla_NomRelac' es un nombre que identificará la relación entre ambas tablas. Por otro lado, (cf1_fk, cf2_fk,…) son el conjunto de claves foráneas de la tabla que harán referencia una por una, y en el orden que aparecen, a las claves primarias (cp1, cp2,…) presentes en la tabla referenciada. 'NombreTablaRef'.

RESTRICCIONES CHECK
ALTER TABLE NombreTabla
ADD CONSTRAINT CK_NomTabla_NomRestric CHECK Condición;
Donde *CK_NomTabla_NomRestric* es el nombre dado a la restricción, y *Condición* es una condición establecida sobre alguna de las columnas de la tabla.

Como se ha indicado, aunque esta restricción es válida en SQL, no está implementada con MySQL. Aunque no produce error, no chequea la condición.

Eliminar una restricción:

Para eliminar una restricción bastará con ejecutar la sentencia: ALTER TABLE NombreTabla DROP CONSTRAINT NombreRestricción;

ARTÍCULO DE INTERÉS

Conoce el porqué del uso de las bases de datos en determinados sectores de la vida:

8.4 Creación de índices

Los índices son estructuras que se crean sobre una o varias columnas de una tabla con el objetivo de mejorar el rendimiento de las consultas. Permiten que los SGBD encuentren más rápidamente los registros que coinciden con las condiciones de búsqueda, evitando así tener que recorrer todas las filas de una tabla. Son como un apuntador o puntero a una fila de una tabla determinada.

Un ejemplo sencillo de cómo funcionan los índices en bases de datos podría ser la forma en la que se usa un diccionario o una enciclopedia. Si estamos buscando una palabra que sabemos que empieza por C, iremos directamente a la página donde empiezan las palabras con C y nos saltaremos las letras A y B. Según el caso, podría resultar útil crear un índice por el campo Apellido de la tabla Empleado, pero ¿cómo saber qué campo a campos conviene usar como índices? Existen ciertas consideraciones a tener en cuenta:

- **Frecuencia de consultas**: se refiere a la frecuencia en que las consultas incluyen ese campo en las condiciones de búsqueda. Si la frecuencia es alta, puede ser un buen candidato a índice.
- **Cardinalidad del campo:** se refiere a la cantidad de valores únicos que tiene un campo. Si tenemos una tabla con muchos apellidos distintos, entonces un índice puede resultar útil.
- **Tamaño de la tabla y consultas concurrentes:** si la tabla Empleado es grande y hay consultas concurrentes, es decir, se realizan varias consultas sobre la tabla al mismo tiempo, un índice puede ser beneficioso para mejorar el rendimiento.

A continuación, se procede a enumerar los principales tipos de índices:

- **INDEX (NON-UNIQUE):** permite la existencia de duplicados en los valores indexados.
- **UNIQUE:** garantiza la unicidad de los valores indexados en una columna.
- **PRIMARY:** son un tipo especial de índice único que se crea en la clave primaria de la tabla. En la mayoría de los SGBD, al definir una clave primaria en una tabla, se crea automáticamente un índice primario.
- **FULL-TEXT:** es un tipo especial de índice utilizado para realizar búsquedas en texto no estructurado, como páginas web, documentos, etc.
- **SPATIAL:** son un tipo especial de índice utilizado en bases de datos para optimizar y acelerar consultas que involucran datos espaciales, como coordenadas geográficas, puntos en un plano cartesiano, polígonos, líneas, etc.

Para crear un índice se utiliza la siguiente estructura:

- CREATE [UNIQUE | FULL-TEXT | SPATIAL] INDEX nombre_indice ON nombre_tabla (nombre_columna).

```
CREATE UNIQUE INDEX idx_UniqueEmail ON Empleado (Email);
CREATE INDEX idx_NonUnique ON Empleado (Nombre);
CREATE FULLTEXT INDEX idx_FullText ON Empleado (Contenido);
                              CREATE SPATIAL INDEX idx_Spatial ON
Empleado (Ubicacion);
```

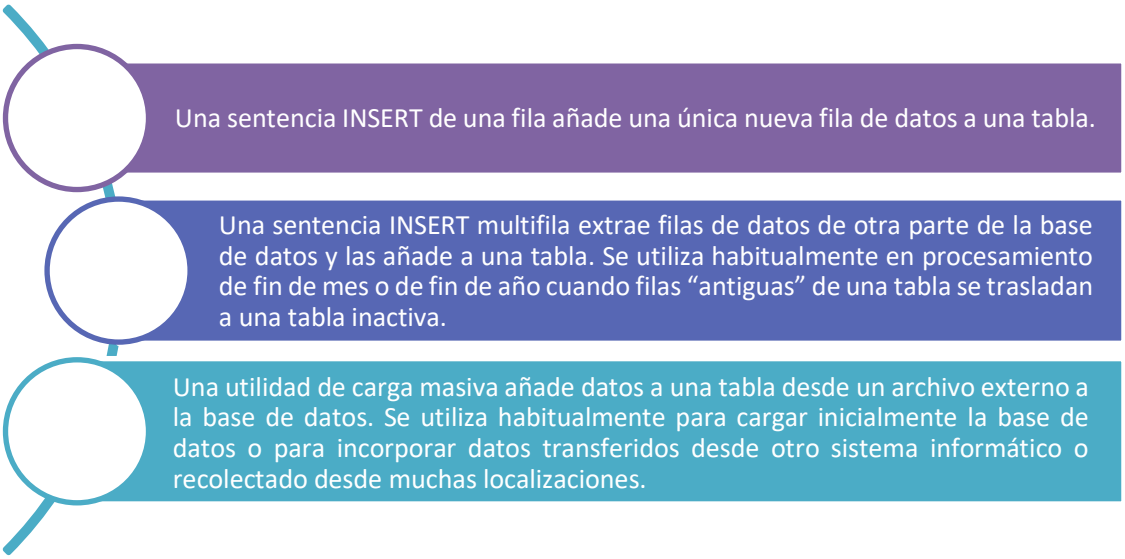
9. INTRODUCCIÓN DE DATOS EN LA BASE DE DATOS

Finalmente has terminado con la creación de la base de datos, por lo que el siguiente paso será volcar toda la información de la empresa en la misma. Para ello, debes decidir la forma en que se realizará esa carga.

Típicamente, una nueva fila de datos se añade a una base de datos relacional cuando una nueva entidad representada por la fila aparece en el mundo exterior. La cláusula adecuada para la introducción de datos en una base de datos es INSERT, seguida de INTO y de las columnas y valores a introducir.

En cada caso, la nueva fila se añade para mantener la base de datos como un modelo preciso del mundo real. La unidad de datos más pequeña que puede añadirse a una base de datos relacional es una fila.

En general, un DBMS basado en SQL proporciona tres maneras de añadir nuevas filas de datos a una base de datos:



Inserción de datos en una base de datos.

9.1 La sentencia INSERT

La sentencia **INSERT** de una fila, añade una nueva fila a una tabla. La cláusula INTO especifica la tabla que recibe la nueva fila (la tabla destino) y la cláusula VALUE especifica los valores de los datos que contendrá la nueva fila. La lista de columnas indica qué valor va a qué columna de la nueva fila.

EJEMPLO PRÁCTICO

Se supone que se acaba de recibir un nuevo coche, Juan Cobos, con los siguientes datos: 1234BCD, Audi, A3, 2016. ¿Cuál sería la sentencia?

```
INSERT INTO `coche` (`matricula`, `marca`, `modelo`, `año`) VALUES
('1234BCD', 'Audi', 'A3', '2016');
```

Como muestra este ejemplo, la sentencia INSERT puede ser larga si hay muchas columnas de datos, pero su formato sigue siendo muy sencillo. El propósito de la lista de columnas en la sentencia INSERT es hacer corresponder los valores de datos en la cláusula VALUES con las columnas que van a recibirlos.

La lista de valores y la lista de columnas deben contener el mismo número de elementos y el tipo de dato de cada valor debe ser compatible con el tipo de dato de la columna correspondiente, o en caso contrario se producirá un error. Además, es importante fijarse en que los campos de texto se deben introducir entrecomillados, mientras que los numéricos no. Esto es muy importante ya que, si no se hace así, la consultará no se ejecutará.

Cuando SQL inserta una nueva fila de datos en una tabla, automáticamente asigna un valor NULL a cualquier columna cuyo nombre falte en la lista de columnas de la sentencia INSERT. Sin embargo, se debe tener en cuenta que en el ejemplo actual se ha incluido en la definición de estos campos la cláusula DEFAULT, con los valores ‘’ para los campos de texto, y 0 para los numéricos, por lo que al insertar registros no se les asignará nulos a esas columnas, sino el valor que tengan indicado por defecto.

VÍDEO DE INTERÉS

Visualiza la sección que trata sobre la sentencia INSERT INTO:

9.2 Inserción de todas las columnas

Por conveniencia, SQL permite omitir la lista de columnas de la sentencia INSERT. Cuando se omite la lista de columnas, SQL genera automáticamente una lista formada por todas las columnas de la tabla, una secuencia de izquierda a derecha. Esta es la misma secuencia de columnas generadas por SQL cuando se utiliza una consulta SELECT. Utilizando esta forma abreviada, la sentencia INSERT anterior podría escribirse de la siguiente forma:

EJEMPLO PRÁCTICO

Insertar un coche en la tabla correspondiente.

```
INSERT INTO `coche` VALUES ('4567FTG', 'Mercedes', 'Benz', '2017')
```

Para comprobar todos los valores introducidos, se vuelven a mostrar:

Esta opción no es recomendable, ya que si se almacena la instrucción para su posterior ejecución (o se utiliza dentro de un programa), es muy posible que los valores se introduzcan de forma incorrecta si se producen cambios en la estructura de la tabla. El estándar SQL especifica varias restricciones lógicas sobre la consulta que aparece dentro de la sentencia INSERT multifila:

- La consulta no puede contener una cláusula ORDER BY.
- El resultado de la consulta debe contener el mismo número de columnas que hay en la lista de columnas de la sentencia INSERT y los tipos de los datos deben ser compatibles columna a columna.
- La consulta no puede ser la UNION de varias sentencias SELECT diferentes. Sólo puede especificarse una única sentencia SELECT.
- La tabla destino de la sentencia INSERT no puede aparecer en la cláusula FROM de la consulta o de ninguna subconsulta que ésta contenga.

9.3 Carga masiva

Los datos a insertar en una base de datos son con frecuencia extraídos de otro sistema automático, o recolectados de otros lugares y almacenados en un archivo secuencial. Para cargar los datos en una tabla, se podría escribir un programa con un bucle que leyera cada registro del archivo y utilizara la sentencia INSERT de una fila para añadir la fila a la tabla. Sin embargo, el recargo de hacer que el DBMS ejecute repetidamente sentencias INSERT de una fila puede ser bastante alto.

Si insertar una sola fila tarda medio segundo para una carga de sistema típico, éste es un rendimiento probablemente aceptable para un programa interactivo. Pero este rendimiento rápidamente pasa a ser inaceptable cuando se aplica a la tarea de cargar 50.000 filas de datos de una vez. En este caso, la carga de los datos requeriría muchas horas. Por esta razón, todos los productos DBMS comerciales incluyen una capacidad de carga masiva que carga los datos desde un archivo a una tabla a alta velocidad.

10. SUPRESIÓN DE DATOS DE LA BASE DE DATOS

Una vez terminada la carga de datos, Paula detecta que hay cierta información que ya no se usa y que no es necesario seguir almacenando, por lo que te pide que procedas al borrado de dicha información.

Típicamente, una fila de datos se suprime de una base de datos cuando la entidad representada por la fila desaparece del mundo exterior.

10.1 La sentencia DELETE

La sentencia DELETE elimina filas seleccionadas de datos de una única tabla. La cláusula FROM especifica la tabla destino que contiene las filas. La cláusula WHERE especifica qué filas de la tabla van a ser suprimidas.

EJEMPLO PRÁCTICO

Si se quisiera eliminar un coche de la base de datos.

```
DELETE FROM `coche` WHERE `marca`='Audi'
```

Pero esto no es muy recomendable, ya que podrían existir varios coches con marca 'Audi'. Sería más correcto indicar cuál es la matrícula del coche a borrar, ya que dicho campo lo identificará como único.

```
DELETE FROM `coche` WHERE `matricula`='4567FTG'
```

Se puede comprobar cómo efectivamente el registro se ha eliminado de la tabla coches.

10.2 Supresión de todas las filas

La cláusula WHERE en una sentencia DELETE es opcional, pero casi siempre está presente. Si se omite la cláusula WHERE de una sentencia DELETE, se suprimen todas las filas de la tabla destino.

EJEMPLO PRÁCTICO

Si se quisieran eliminar todos los coches.

```
DELETE   FROM COCHES
```

Aunque esta sentencia DELETE produce una tabla vacía, no borra la tabla COCHES de la base de datos. La definición de la tabla y sus columnas siguen estando almacenadas en la base de datos. La tabla aún existe, y nuevas filas pueden ser insertadas con la sentencia INSERT. Para eliminar la definición de la tabla de la base de datos, debe utilizarse la sentencia DROP TABLE.

Debido al daño potencial que puede producir una sentencia DELETE, es importante especificar siempre una condición de búsqueda, y tener cuidado de que se seleccionan realmente filas que se desean. Cuando se utiliza SQL interactivo, es buena idea utilizar primero la cláusula WHERE en una sentencia SELECT para visualizar las filas seleccionadas, asegurarse de que son las que realmente se desea suprimir y sólo entonces utilizar la sentencia DELETE con la cláusula WHERE correspondiente.

Otra forma de eliminar los datos de una tabla es mediante el uso de la sentencia TRUNCATE. Dicha sentencia se utiliza para eliminar todos los registros de una tabla (sin opción de usar WHERE para filtrar el borrado), pero la estructura de la misma, sus columnas, índices y restricciones se mantienen intactos. TRUNCATE es una operación más rápida que DELETE, ya que no registra cada eliminación de fila en el registro de transacciones y no activa los disparadores (triggers) asociados con la tabla. Sin embargo, es una operación que no se puede deshacer. La manera de usar esta sentencia es:

```
TRUNCATE TABLE nombre_tabla;
```

PARA SABER MÁS

Otra web donde encontrar información sobre SQL y las sentencias estudiadas es:

11. MODIFICACIÓN DE DATOS DE LA BASE DE DATOS

Durante el proceso de eliminación anterior, te has dado cuenta de que en una de las tablas hay datos que no son correctos, por lo que te dispones a modificarlos para subsanar el error.

Típicamente, los valores de los datos almacenados en una base de datos se modifican cuando se producen cambios correspondientes en el mundo exterior. En cualquier momento puede ocurrir que un dato almacenado cambie de valor.

11.1 La sentencia UPDATE

La sentencia **UPDATE** modifica los valores de una o más columnas en las filas seleccionadas de una tabla única. La tabla destino a actualizar se indica en la sentencia y es necesario disponer de permiso para actualizar la tabla, así como cada una de las columnas individuales que serán modificadas. La cláusula WHERE selecciona las filas de la tabla a modificar. La cláusula SET especifica qué columnas se van a actualizar y calcula los nuevos valores.

EJEMPLO PRÁCTICO

Si se quiere asignar año de matriculación del 2018 a todos los coches de marca Audi, se podría ejecutar la siguiente instrucción.

```
UPDATE COCHE SET año = 2018 WHERE marca = 'Audi'
```

La cláusula SET en la sentencia UPDATE es una lista de asignaciones separadas por comas. Cada asignación identifica una columna a actualizar, y especifica cómo calcular su nuevo valor. Cada columna destino debería aparecer solamente una vez en la lista; no debería haber dos asignaciones para la misma columna.

La expresión en cada asignación puede ser cualquier expresión SQL válida que genere un valor tipo de dato apropiado para la columna destino. La expresión se debe poder calcular con los valores de la fila que actualmente está en actualización en la tabla destino.

11.2 Actualización de todas las filas

La cláusula WHERE en la sentencia UPDATE es opcional. Si se omite la cláusula WHERE, entonces se actualizan todas las filas de la tabla destino.

EJEMPLO PRÁCTICO

Si se quiere cambiar el año de matriculación a todos los coches.

```
UPDATE COCHE SET año = 2018
```

ENLACE DE INTERÉS

Conoce más sobre el control de concurrencia en bases de datos distribuidas:

A modo de resumen, se deben tener en cuenta los siguientes puntos:

- La sentencia INSERT de una fila añade una fila de datos a una tabla. Los valores para la nueva fila se especifican en la sentencia como constantes.

- La sentencia INSERT multifila añade cero o más filas a una tabla. Los valores para las nuevas filas provienen de una consulta, especificada como parte de la sentencia INSERT.

- La sentencia DELETE suprime cero o más filas de datos de una tabla. Las filas a suprimir son especificadas mediante una condición de búsqueda.

- La sentencia TRUNCATE elimina todas las filas de la tabla sin opción de filtrado.

- La sentencia UPDATE modifica los valores de una o más columnas en cero o más filas de una tabla. Las filas a actualizar son especificadas mediante una condición de búsqueda.

- A diferencia de la sentencia SELECT, que puede operar sobre múltiples tablas, las sentencias INSERT, DELETE y UPDATE funcionan solamente sobre una única tabla cada vez.

RESUMEN FINAL

En la presente unidad se han expuesto una serie de conceptos muy importantes en el ámbito de las bases de datos, como son la creación, modificación y borrado de tablas, así como los diferentes tipos de datos con los que se puede trabajar en una base de datos MySQL, como son: números exactos, números decimales, números aproximados, cadenas de caracteres, cadenas de bits y datos y fechas. En cuanto a las formas de trabajar hemos estudiado el modo interactivo y el cómo se ejecuta desde un programa.

También hemos conocido cómo se pueden insertar y eliminar valores en las tablas previamente creadas (INSERT y DELETE), así como las instrucciones necesarias para la creación de nuevos usuarios y la concesión de diferentes permisos.

Por último, se ha introducido el concepto de índice, los cuales son estructuras que se crean sobre una o varias columnas de una tabla para mejorar el rendimiento de las consultas, explicando los diferentes tipos existentes y la utilidad de los mismos.