¿Qué es un sistema gestor de bases de datos?

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben garantizar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o de los intentos de acceso no autorizados. Si los datos van a ser compartidos entre diferentes usuarios, el sistema debe evitar posibles resultados anómalos. Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado un gran cuerpo de conceptos y técnicas para la gestión de los datos. Estos conceptos y técnicas constituyen el objetivo central de este libro. En este capítulo se presenta una breve introducción a los principios de los sistemas de bases de datos.

Aplicaciones de los sistemas de bases de datos.

Las bases de datos se usan ampliamente. Algunas ejemplos son:

- Banca: para información de los clientes, cuentas, préstamos y transacciones bancarias.
- Líneas aéreas: para reservas e información de horarios. Las líneas aéreas fueron de las primeras en
- Universidades: para información de los estudiantes, matrículas en las asignaturas y cursos.
- Transacciones de tarjetas de crédito: para compras con tarjeta de crédito y la generación de los extractos mensuales.
- Telecomunicaciones: para guardar un registro de las llamadas realizadas, generar las facturas mensuales, mantener el saldo de las tarjetas telefónicas de prepago y para almacenar información
- Finanzas: para almacenar información sobre compañías tenedoras, ventas y compras de productos financieros, como acciones y bonos; también para almacenar datos del mercado en tiempo real

Como muestra esta lista, las bases de datos forman una parte esencial de casi todas las empresas actuales. Durante las últimas cuatro décadas del siglo veinte, el uso de las bases de datos creció en todas las empresas. En los primeros días, muy pocas personas interactúan directamente con los sistemas de bases de datos, aunque sin darse cuenta interactúan directamente con bases de datos—con informes impresos como los extractos de las tarjetas de crédito, o mediante agentes como los cajeros de los bancos y los agentes de reservas de las líneas aéreas. Después vinieron los cajeros automáticos y permitieron a los usuarios interactuar directamente con las bases de datos. Las interfaces telefónicas con las computadoras (sistemas de respuesta vocal interactiva) también permitieron a los usuarios tratar directamente con las bases de datos—la persona que llamaba podía marcar un número y pulsar las teclas del teléfono para introducir información o para seleccionar opciones, para conocer las horas de llegada o salida de los vuelos, por ejemplo, o para matricularse de asignaturas en una universidad.

Propósito de los sistemas de bases de datos

Los sistemas de bases de datos surgieron en respuesta a los primeros métodos de gestión informatizada de los datos comerciales. A modo de ejemplo de dichos métodos, típicos de los años sesenta, considérese parte de una entidad bancaria que, entre otros datos, guarda información sobre todos los clientes y todas las cuentas de ahorro. Una manera de guardar la información en la computadora es almacenarla en archivos del sistema operativo. Para permitir que los usuarios manipulen la información, el sistema tiene varios programas de aplicación que gestionan los archivos, incluyendo programas para:

- Efectuar cargos o abonos en las cuentas.
- Añadir cuentas nuevas.
- Calcular el saldo de las cuentas.
- Generar los extractos mensuales.

Estos programas de aplicación los han escrito programadores de sistemas en respuesta a las necesidades del banco. Se añaden nuevos programas de aplicación al sistema según surgen las necesidades. Por ejemplo, supóngase que una caja de ahorros decide ofrecer cuentas corrientes. En consecuencia, se crean nuevos archivos permanentes que contienen información acerca de todas las cuentas corrientes abiertas en el banco y puede que haya que escribir nuevos programas de aplicación para afrontar situaciones que no se dan en las cuentas de ahorro, como los descubiertos. Así, con el paso del tiempo, se añaden más archivos y programas de aplicación al sistema. Los sistemas operativos convencionales soportan este sistema de procesamiento de archivos típico. El sistema almacena los registros permanentes en varios archivos y necesita diferentes programas de aplicación para extraer y añadir a los archivos correspondientes. Antes de la aparición de los sistemas gestores de bases de datos (SGBDs), las organizaciones normalmente almacenaban la información en sistemas de este tipo. Guardar la información de la organización en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes:

- Redundancia e inconsistencia de los datos. Debido a que los archivos y programas de aplicación los crean diferentes programadores en el transcurso de un largo período de tiempo, es probable que los diversos archivos tengan estructuras diferentes y que los programas estén escritos en varios lenguajes de programación diferentes. Además, puede que la información esté duplicada en varios lugares (archivos). Por ejemplo, la dirección y el número de teléfono de un cliente dado pueden aparecer en un archivo que contenga registros de cuentas de ahorros y en un archivo que contenga registros de cuentas corrientes. Esta redundancia conduce a costes de almacenamiento y de acceso más elevados. Además, puede dar lugar a la inconsistencia de los datos; es decir, puede que las diferentes copias de los mismos datos no coincidan. Por ejemplo, puede que el cambio en la dirección de un cliente esté reflejado en los registros de las cuentas de ahorro, pero no en el resto del sistema.
- Dificultad en el acceso a los datos. Supóngase que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en un código postal dado. El empleado pide al departamento de procesamiento de datos que genere esa lista. Debido a que esta petición no fue prevista por los diseñadores del sistema original, no hay un programa de aplicación a mano para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de todos los clientes. El empleado del banco tiene ahora dos opciones: bien obtener la lista de todos los clientes y extraer manualmente la información que necesita, o bien pedir a un programador de sistemas que escriba el programa de aplicación necesario. Ambas alternativas son obviamente insatisfactorias. Supóngase que se escribe el programa y que, varios días más tarde, el mismo empleado necesita reducir esa lista para que incluya únicamente a aquellos clientes que tengan una cuenta con saldo igual o superior a 10.000 e. Como se puede esperar, no existe ningún programa que genere tal lista. De nuevo, el empleado tiene que elegir entre dos opciones, ninguna de las cuales es satisfactoria. La cuestión aquí es que los entornos de procesamiento de archivos convencionales no permiten recuperar los datos necesarios de una forma práctica y eficiente. Hacen falta sistemas de recuperación de datos más adecuados para el uso general.

• Aislamiento de datos. Como los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos correspondientes.
• Problemas de integridad. Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo, el saldo de ciertos tipos de cuentas bancarias no puede nunca ser inferior a una cantidad predeterminada (por ejemplo, 25 e). Los desarrolladores hacen cumplir esas restricciones en el sistema añadiendo el código correspondiente en los diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema se complica cuando las restricciones implican diferentes elementos de datos de diferentes archivos.
• Problemas de atomicidad. Los sistemas informáticos, como cualquier otro dispositivo mecánico eléctrico, está sujeto a fallos. En muchas aplicaciones es crucial asegurar que, si se produce algún fallo, los datos se restauren al estado consistente que existía antes del fallo. Considérese un programa para transferir 50 e desde la cuenta A a la B. Si se produce un fallo del sistema durante la ejecución del programa, es posible que los 50 e fueran retirados de la cuenta A pero no abonados en la cuenta B, dando lugar a un estado inconsistente de la base de datos. Evidentemente, resulta esencial para la consistencia de la base de datos que tengan lugar tanto el abono como el cargo, o que no tenga lugar ninguno. Es decir, la transferencia de fondos debe ser atómica debe ocurrir en su totalidad o no ocurrir en absoluto. Resulta difícil asegurar la atomicidad en los sistemas convencionales de procesamiento de archivos.
• Anomalías en el acceso concurrente. Para aumentar el rendimiento global del sistema y obtener una respuesta más rápida, muchos sistemas permiten que varios usuarios actualicen los datos simultáneamente. En realidad, hoy en día, los principales sitios de comercio electrónico de Internet pueden tener millones de accesos diarios de compradores a sus datos. En tales entornos es posible la interacción de actualizaciones concurrentes y puede dar lugar a datos inconsistentes. Considérese una cuenta bancaria A, que contenga 500 e. Si dos clientes retiran fondos (por ejemplo, 50 e y 100 e, respectivamente) de la cuenta A aproximadamente al mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente). Supóngase que los programas que se ejecutan para cada retirada leen el saldo anterior, reducen su valor en el importe que se retira y luego escriben el resultado. Si los dos programas se ejecutan concurrentemente, pueden leer el valor 500 e, y escribir después 450 e y 400 e, respectivamente. Dependiendo de cuál escriba el valor en último lugar, la cuenta puede contener 450 e o 400 e, en lugar del valor correcto, 350 e. Para protegerse contra esta posibilidad, el sistema debe mantener alguna forma de supervisión. Pero es difícil ofrecer supervisión, ya que muchos programas de aplicación diferentes que no se han coordinado con anterioridad pueden tener acceso a los datos.

• Problemas de seguridad. No todos los usuarios de un sistema de bases de datos deben poder acceder a todos los datos. Por ejemplo, en un sistema bancario, el personal de nóminas sólo necesita a ver la parte de la base de datos que contiene información acerca de los diferentes empleados del banco. No necesitan tener acceso a la información acerca de las cuentas de clientes. Pero, como los programas de aplicación se añaden al sistema de procesamiento de datos de una forma ad hoc, es difícil hacer cumplir tales restricciones de seguridad. Estas dificultades, entre otras, motivaron el desarrollo de los sistemas de bases de datos. En el resto del libro se examinarán los conceptos y los algoritmos que permiten que los sistemas de bases de datos resuelvan los problemas de los sistemas de procesamiento de archivos. En general, en este libro se usa una entidad bancaria como ejemplo de aplicación típica de procesamiento de datos que puede encontrarse en una empresa.

Bases de datos Relacional

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.

Resumen e implicaciones

Cinco funciones principales del administrador de bases de datos

- Asegurar el buen funcionamiento de las bases de datos
- Retención de información de la base de datos
- Evitar pérdida de datos
- Solucionar incidencias y pérdidas de datos
- Asegurar la integridad de los datos

Cinco responsabilidades del sistema gestor de bases de datos

- Instalar, configurar y gestionar bases de datos.
- Dar soporte al equipo de desarrollo, seguridad informática y redes.
- Definir el esquema del diccionario de datos.
- Especificar restricciones de integridad para asegurar los datos.
- Garantizar la alta disponibilidad de la base de datos.

Agregar archivos nuevos en la base de datos, insertar, eliminar, actualizar y obtener datos de archivos existentes de la base de datos son operaciones sobre la misma que se le permiten a usuario del sistema

Un sistema de Información

Es un conjunto de elementos orientados al tratamiento y administración de datos en información, organizados listos para su posterior uso, generados para cubrir una necesidad.

La estructura de bases de datos relacionales

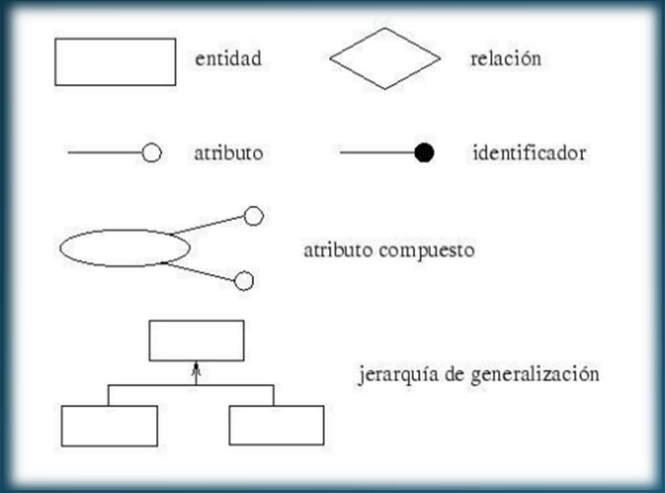
Una base de datos relacional consiste en un conjunto de tablas, a las cuales se les asigna un nombre exclusivo. Cada fila de la tabla representa una relación entre un conjunto de valores. De manera informal, cada tabla es un conjunto de entidades, y cada fila es una entidad. Dado que cada tabla es un conjunto de tales relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación, del que toma su nombre el modelo de datos relacional. A continuación, se introduce el concepto de relación. En este capítulo se usarán varias relaciones diferentes para ilustrar los diversos conceptos subyacentes al modelo de datos relacional. Estas relaciones representan parte de una entidad bancaria. Puede que no se correspondan con el modo en que se pueda estructurar realmente una base de datos bancaria, pero así se simplificará la presentación.

Ejemplo de una tabla.

Modelo Relacional

El modelo de entidad-relación.

El modelo entidad-relación es el modelo conceptual más utilizado para el conceptual de bases de datos. Fue introducido por Peter Chan en 1976. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas. Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido.



Modelo Relacional

Entidad: Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. Hay dos tipos de entidades: fuertes y débiles. Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil.

Diagramas entidad relación

Los diagramas entidad relación son sencillos y claros cualidades que pueden ser responsables en gran parte de la popularidad del modelo entidad relación estos diagramas constan de los siguientes componentes principales:

- Rectángulos, que representan conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan conjuntos de relaciones.
- Líneas, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con los conjuntos de relaciones.
- Elipses dobles, que representan atributos multivaluados
- Elipses discontinuas, que denotan atributos derivados.
- Líneas dobles, que indican participación total de una entidad en un conjunto de relaciones.
- Rectángulos dobles que representan conjuntos de entidades débiles.

Ejemplo de diagrama entidad relación

Aspectos de diseño entidad relación.

Los conceptos de conjunto de entidades y de conjunto de relaciones no son precisos, y es posible definir el conjunto de entidades y las relaciones entre ellas de diferentes formas. En este apartado se examinan aspectos básicos del diseño de esquemas de bases de datos entidad relación.

Lenguaje estructurado de consulta (SQL)

Características del lenguaje SQL. Utilidad del lenguaje.

El lenguaje SQL, o lenguaje de consulta estructurado, es hoy el más utilizado en cualquier ámbito en el que se trabaje con bases de datos. Se trata de un lenguaje que permite acceder, modificar o eliminar la información que se almacena en las bases de datos. Esta información está relacionada entre sí, por lo que debe ser estructurada y almacenada siguiendo un sistema. El lenguaje SQL permite interactuar con esa información.

En la década de los 70, en pleno desarrollo de las bases de datos, IBM creaba un lenguaje con el que gestionar los datos almacenados en el nuevo software System R. Era el SEQUEL, que más tarde pasaría a llamarse SQL (Structured Query Language). En 1986, fue declarado estándar del Instituto Nacional Estadounidense de Estándares (ANSI) y, un año después, de la Organización Internacional de Normalización (ISO). Fue tal su impacto, que a partir de ese momento, varias compañías lanzaron su propia versión. Hoy continúa siendo el principal referente, tanto en el uso individual como en servidores.

Las distintas aplicaciones del lenguaje SQL. El internet ha llevado a las bases de datos a otro nivel. Son la clave del funcionamiento de las páginas web, por lo que SQL tiene una especial importancia en el ámbito digital. Cualquier sitio web recurre al sistema de base de datos para que sus contenidos puedan ser utilizados. SQL simplifica en gran medida su gestión. Hoy, todas las empresas manejan una cantidad importante de información, pero la clave está en saber gestionarla. En ese sentido, un experto en SQL en el equipo de trabajo es una garantía, ya sea en el sector de las telecomunicaciones como en la industria de la automoción, la hostelería, la educación, la banca, el marketing... Todos ellos son sectores muy distintos, pero con una misma necesidad: el manejo de las bases de datos. Una base de datos correctamente estructurada es una herramienta enormemente útil con un rendimiento muy alto. Entonces, ¿quién debe aprender SQL? Pues no solamente programadores o gestores de bases de datos. Cualquier persona que trabaje con análisis de datos o tratamiento de información debería tener conocimientos de SQL, independientemente de que esté en el departamento de RR.HH., Estadística o en el de Marketing. La ventaja está en que no es difícil de aprender. SQL permite manejar la información contenida en una base de datos, sobre todo su utilidad radica en que facilita la búsqueda y la edición de esos datos. Es una alternativa mucho más eficaz que la manual a la hora de organizar información, algo que hacemos constantemente. Con el Máster online en Marketing Intelligence de UNIR, aprenderás las últimas tendencias de la analítica de datos aplicadas al Marketing.

Comandos SQL .Grupos de comando

Los comandos del SQL pueden dividirse en tres grupos:

- Comandos de definición de datos (DDL = Data Definition Language),que permiten crear y definir nuevas bases de datos,campos etc.
- Comandos de manipulación de datos (DML = Data ManipulationLanguage),que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
- Comandos de control y seguridad de datos, que gobiernan los privilegios de los usuarios, los controles de acceso.

Los principales comandos del lenguaje SQL son:

Comandos de DDL	
Comando	**Descripción**
CREATE	Encargado de crear nuevas tablas, campos, …
DROP	Encargado de eliminar tablas
ALTER	Encargado de modificar las tablas, agregando campos o cambiando la definición de los campos

Comandos de DML	
Comando	**Descripción**
SELECT	Encargado de consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Encargado de cargar lotes de datos en la base de datos en una única operación
UPDATE	Encargado de modificar los valores de los campos y registros especificados
DELETE	Encargado de eliminar registros de una tabla

Operadores Lógicos

Operador	Descripción
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Lenguaje estructurado de consulta (SQL)

Condiciones o criterios: por medio de ciertos modificadores, llamados cláusulas, se consigue generar criterios con el fin de definir los datos que se desea seleccionar o manipular.

Cláusula	Descripción
FROM	Sirve para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Sirve para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Sirve para especificar un criterio adicional por el que agrupar los registros seleccionados
HAVING	Sirve para expresar la condición que debe satisfacer cada grupo anterior
ORDER BY	Sirve para ordenar los registros seleccionados de acuerdo con el orden especificado

Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

Operadores de Comparación

Operador	Descripción
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó igual que
>=	Mayor ó igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
IN	Utilizado para especificar registros de una base de datos

Definición de datos.

El conjunto de relaciones de cada base de datos debe especificarse en el sistema en términos de un lenguaje de definición de datos (LDD). El LDD de SQL no sólo permite la especificación de un conjunto de relaciones, sino también de la información relativa a esas relaciones, incluyendo:

- El esquema de cada relación.
- El dominio de valores asociado a cada atributo.
- Las restricciones de integridad.
- El conjunto de índices que se deben mantener para cada relación.
- La información de seguridad y de autorización de cada relación.
- La estructura de almacenamiento físico de cada relación en el disco.

Tipos básicos de dominios:

- char(n).Una cadena de caracteres de longitud fija, con una longitud especificada por el usuario.También se puede utilizar la palabra completa character.
- varchar(n).Una cadena de caracteres de longitud variable con una longitud máxima n especificada por el usuario. La forma completa, character varying, es equivalente.
- int. Un entero (un subconjunto finito de los enteros dependiente de la máquina). La palabra completa, integer, es equivalente.
- smallint. Un entero pequeño (un subconjunto dependiente de la máquina del tipo de dominio entero).
- numeric(p, d). Un número de coma fija, cuya precisión la especifica el usuario. El número está formado por p dígitos (más el signo), y de esos p dígitos, d pertenecen a la parte decimal. Así, numeric(3,1) permite que el número 44.5 se almacene exactamente, pero ni 444.5ni 0.32 se pueden almacenar exactamente en un campo de este tipo.
- real, double precision. Números de coma flotante y números de coma flotante de doble precisión, con precisión dependiente de la máquina.
- float(n). Un número de coma flotante cuya precisión es, al menos, de n dígito.

Estructura básica de las consultas SQL

Las bases de datos relacionales están formadas por un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. Cada relación posee una estructura similar. SQL permite el uso de valores nulos para indicar que el valor es desconocido o no existe. También permite al usuario especificar los atributos que no pueden contener valores nulos.
La estructura básica de una expresión SQL consta de tres cláusulas: select, from y where.

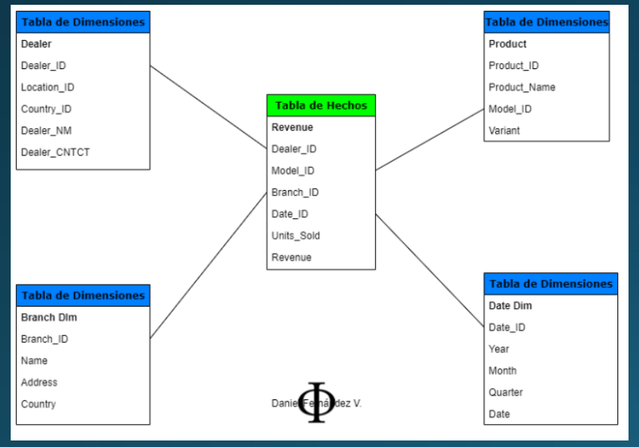
- La cláusula select se corresponde con la operación proyección del álgebra relacional. Se usa para obtener una relación de los atributos deseados en el resultado de una consulta.
- La cláusula from se corresponde con la operación producto cartesiano del álgebra relacional. Genera una lista de las relaciones que deben ser analizadas en la evaluación de la expresión.
- La cláusula where se corresponde con el predicado selección del álgebra relacional.Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula from.

Que el término select tenga un significado diferente en SQL que en el álgebra relacional es un hecho histórico desafortunado. En este capítulo se destacan las diferentes interpretaciones para reducir al mínimo las posibles confusiones. Las consultas habituales de SQL tienen la forma:

$$\begin{aligned}&\textbf{select } A_1,\ A_2, \ldots, A_n\\ &\textbf{from } r_1,\ r_2, \ldots, r_m\\ &\textbf{where } P\end{aligned}$$

Esquemas de almacenes de datos

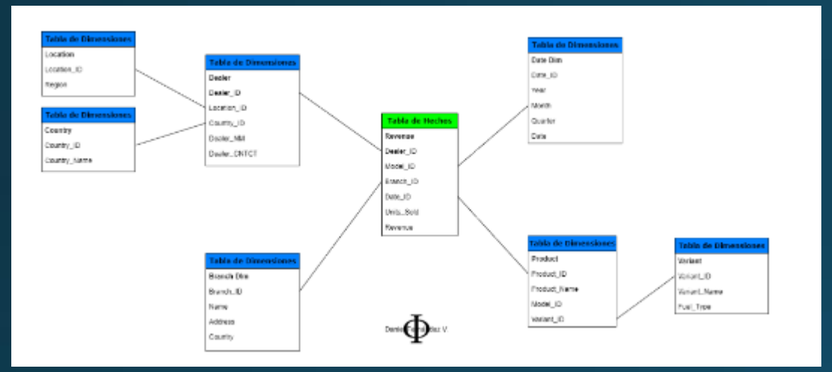
Data warehouse es un sistema que agrega y combina información de diferentes fuentes en un almacén de datos único y centralizado; consistente para respaldar el análisis empresarial, la minería de datos, inteligencia artificial y Machine Learning. Data warehouse permite a una organización o empresa ejecutar análisis potentes en grandes volúmenes petabytes y petabytes de datos históricos de formas que una base de datos estándar simplemente no puede. Los data warehouses han sido parte de las soluciones de inteligencia empresarial durante más de tres décadas, pero han evolucionado significativamente en los últimos años. Tradicionalmente, un data warehouse tenía una implementación , a menudo en un mainframe central, y su funcionalidad se centraba en extraer datos de otras fuentes, limpiar y preparar la información, y cargar y mantener los documentos en una base de datos relacional. Esquema estrella: En el esquema de estrella, el centro de la estrella puede tener una tabla de hechos y varias tablas de dimensiones asociadas. Se conoce como esquema estelar ya que su estructura se asemeja a una estrella. El esquema en estrella es el tipo más simple de esquema de Data Warehouse. También se conoce como Star Join Schema y está optimizado para consultar grandes conjuntos de datos.



Características del esquema estelar:

- Cada dimensión en un esquema de estrella se representa con la única tabla de una dimensión.
- La tabla de dimensiones debe contener el conjunto de atributos.
- La tabla de dimensiones se une a la tabla de hechos utilizando una clave foránea.
- Las tablas de dimensiones no están unidas entre sí.
- La tabla de hechos contendría clave y medida.

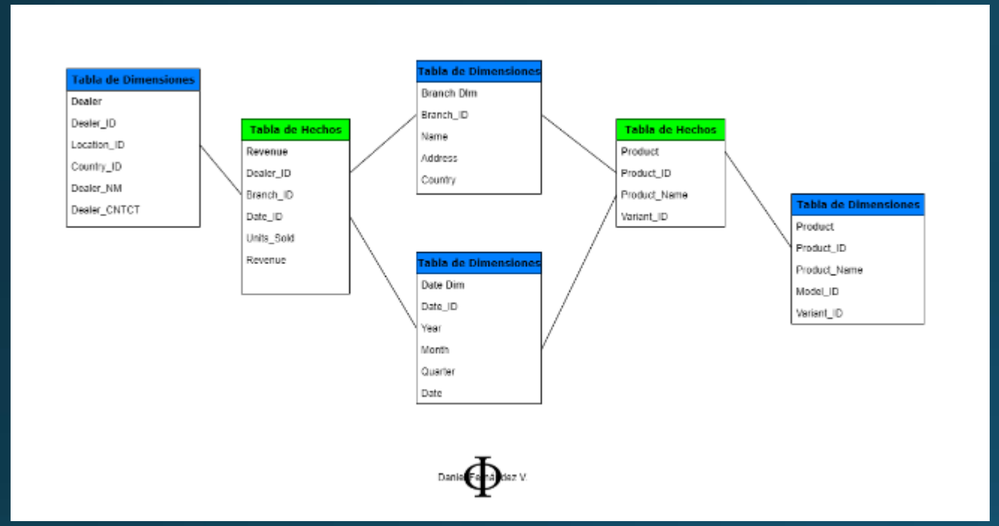
El esquema estrella es fácil de entender y proporciona un uso óptimo del disco. Las tablas de dimensiones no están normalizadas. Esquema copo de nieve: Un esquema de copo de nieve es una extensión de un esquema de estrella y agrega dimensiones adicionales. Se llama como de nieve porque su diagrama se asemeja a un copo de nieve. Las tablas de dimensiones están normalizadas, lo que divide los datos en tablas adicionales. En el siguiente ejemplo, País se normaliza aún más en una tabla individual.



Características del esquema de copo de nieve:

- El principal beneficio del esquema de copo de nieve es que utiliza un espacio en disco más pequeño.
- Debido a múltiples tablas, el rendimiento de la consulta se reduce.
- El principal desafío que enfrentará al usar el esquema de copo de nieve es que necesita realizar más esfuerzos de mantenimiento debido a que hay más tablas de búsquedas.

Esquema Galaxy: Un esquema Galaxy contiene dos tablas de hechos que comparten tablas de dimensiones. También se llama Fact Constellation Schema. El esquema se ve como una colección de estrellas, de ahí el nombre Galaxy Schema.



Dentro de la metodología de diseño de un almacén de datos se encuentran tres etapas diferenciadas que deben ejecutarse en orden secuencial para la obtención del modelo multidimensional deseado.

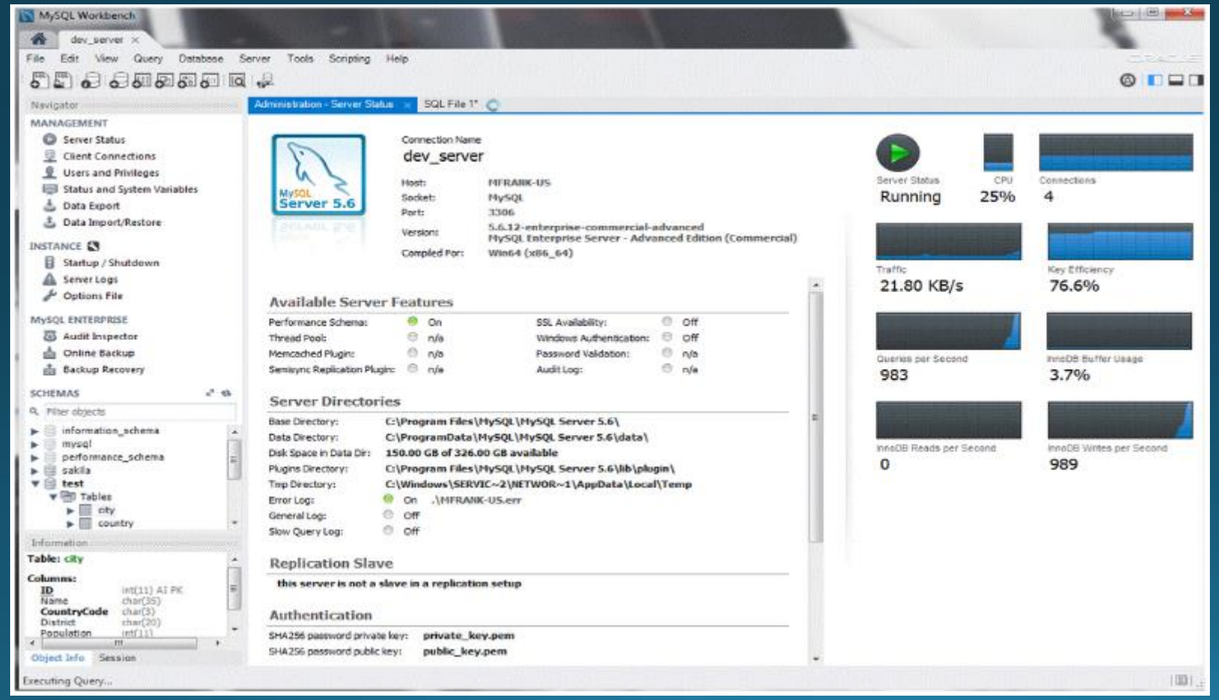
Diseño **conceptual**: Tras el análisis y la recogida de requisitos, se debe hacer un diseño conceptual multidimensional sobre los datos que recibimos como input. En un esquema multidimensional se representa una actividad que es objeto de análisis (hecho) y las dimensiones que caracterizan la actividad (dimensiones). La información relevante sobre el hecho se representa por un conjunto de indicadores (medidas o atributos de hecho).La información descriptiva de cada dimensión se representa por un conjunto de atributos (atributos de dimensión). Tal y como se ha comentado anteriormente, el esquema utilizado en las construcciones de almacenes de datos es el ROLAP y se puede encontrar varios tipos de representaciones del esquema
multidimensional. Comenzamos presentando el esquema en estrella, en este esquema, como se puede ver en la figura, la actividad objeto de estudio se representa en el centro y las dimensiones en las puntas de la estrella.

Diseño **lógico**: Una vez definido el modelo multidimensional con el esquema que se haya seleccionado, se transformará el diagrama de clases UML a un diagrama relacional sobre el que trabajaremos y hemos comentado anteriormente, ROLAP. Las dimensiones pasarán a ser tablas de una base de datos relacional donde se definirá un identificador único por cada casuística que se produzca en la dimensión. Este identificador será utilizado más adelante por la tabla de hechos. Diseño, implementación y explotación de un almacén de datos. El hecho, al igual que las dimensiones, será una tabla de una base de datos relacional la cual contendrá los identificadores de cada casuística producida en las dimensiones y que estén relacionadas entre ellas. Se
definirán claves ajenas a las dimensiones por dichos identificadores. También, se definirá una clave primaria compuesta por los identificadores.

Diseño **físico**: La fase de diseño físico trata de buscar una optimización de los tiempos de consulta sobre las tablas de hechos y dimensiones. Tal y como se ha comentado anteriormente, es muy importante que el usuario obtenga una respuesta rápida a pesar de tener que consultar una cantidad cuantiosa de datos, del orden de cientos de millones de registros. En la tabla de hechos se van a realizar acciones de creación de índices sobre los identificadores de las dimensiones. El orden de las columnas al crear el índice es muy importante. Además, se valorará la posibilidad de particionar la tabla a partir de las propias claves, creando un índice local para cada partición.

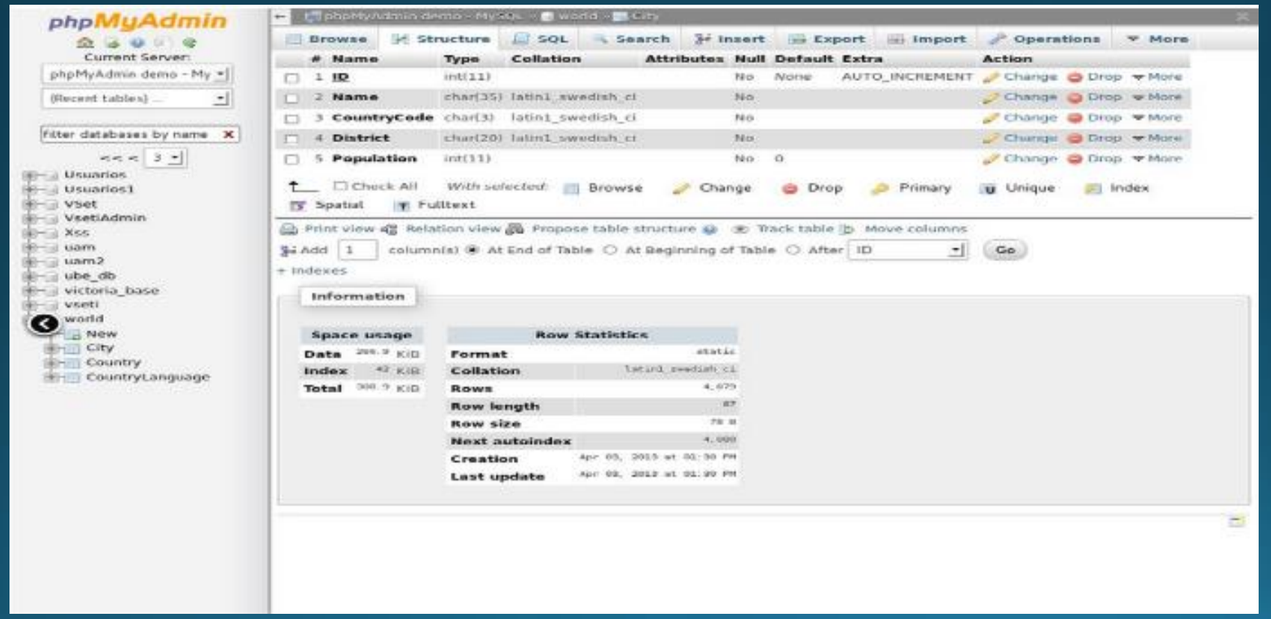
MySQL Workbench

Un editor visual de base de datos MySQL que cuenta con el respaldo oficial de MySQL. Sin duda la herramienta se caracteriza por su editor de diagramas; desde su lienzo podrás arrastrar elementos desde el catálogo o bien añadirlos desde la opción herramientas, disponible en el menú lateral. Y si deseas analizar visualmente el esquema podrás exportarlo en formato imagen o PDF o bien generar un script SQL CREATE o ALTER. MySQL Workbench es un software libre disponible para Windows Mac OS X y Linux. La herramienta dispone de una versión comercial la cual dispone de una validación del modelado o la opción de ingeniaría inversa de base de datos, características no disponibles en la versión gratuita.



phpMyAdmin

Si quieres crear y manejar base datos de forma local (desde tu disco duro) pero bajo un entorno web, sin duda phpMyAdmin es la herramienta que estas buscando. Con la utilidad podrás crear, gestionar y eliminar bases de datos, tablas y campos. También podrás ejecutar secuencias de comandos SQL. phpMyAdmin sin duda esta orientadas a usuarios profesionales, te recomendamos que si eres un usuario novato escojas otra herramienta. Con la aplicación podrás importar en CSV y SQL y exportar en CSV, SQL,
XML, PDF, ISO / IEC 26300, Word y Excel entre otros. phpMyAdmin es una herramienta de software libre disponible en español y desarrollada bajo entorno PHP (y existente en la gran mayoría de los servidores web).



Funciones en SQL.

El lenguaje SQL tiene funciones incorporadas para hacer cálculos sobre los datos. Las funciones se pueden dividir en dos grupos (existen muchas más, que dependen del sistema de bases de datos que se utilice): Funciones agregadas SQL, devuelve un sólo valor, calculado con los valores de una columna.

- AVG() - La media de los valores
- COUNT() - El número de filas
- MAX() - El valor más grande
- MIN() - El valor más pequeño
- SUM() - La suma de los valores
- GROUP BY - Es una sentencia que va muy ligada a las funciones agregadas
- Funciones escalares SQL, devuelven un sólo valor basándose en elvalor de entrada.
- UCASE() - Convierte un campo a mayúsculas
- LCASE() - Convierte un campo a minúsculas
- MID() - Extrae caracteres de un campo de texto
- LEN() - Devuelve la longitud de un campo de texto
- NOW() - Devuelve la hora y fecha actuales del sistema
- FORMAT() - Da formato a un formato para mostrarlo Para la explicación de las funciones, una de las tablas con las que vamos a trabajar es Productos:

ProductoID	NombreProducto	Descripcion	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

1. AVG()

La función AVG() devuelve la media de valores de una columna numérica.

```
SELECT AVG (Stock) FROM Productos;
```

2. COUNT()

La función COUNT() devuelve el número de filas que cumplen con un determinado criterio:

```
SELECT COUNT (nombreColumna) FROM nombreTabla;
```

3. MAX()

La función MAX() devuelve el mayor valor de la columna seleccionada:

```
SELECT MAX (nombreColumna) FROM nombreTabla;
```

4. MIN()

La función MIN() devuelve el menor valor de la columna seleccionada:

```
SELECT MIN (nombreColumna) FROM nombreTabla;
```

5. SUM()

La función SUM() devuelve la suma de una columna numérica:

```
SELECT SUM (nombreColumna) FROM nombreTabla;
```

6. GROUP BY

La sentencia GROUP BY se utiliza junto con las funciones agregadas para agrupar en un result-set una o más columnas.

```
SELECT nombreColumna, funcion_agregada(nombreColumna)
FROM nombreTabla
WHERE nombreColumna operador valor
GROUP BY nombreColumna;
```

Triggers en SQL

Un trigger o disparador es un script que se usa en lenguaje de programación SQL, en especial en bases de datos como MySQL o PostgreSQL. Consiste en una serie de reglas predefinidas que se asocian a una tabla. Estas reglas se aplican a la base de datos cuando se realizan determinadas operaciones en la tabla, por ejemplo, al añadir, actualizar o eliminar registros. Dicho de otra manera, el trigger desencadena determinadas acciones de forma automática en las tablas de la base de datos cuando se insertan, modifican y se añaden nuevos datos. Estos
disparadores se llevan usando en MySQL desde la versión 5.0.2., mientras que PostgreSQL ya los incluyó en el año 1997.

¿Para qué sirve?

La principal función de los trigger es contribuir a mejorar la gestión de la base de datos. Gracias a ellos muchas operaciones se pueden realizar de forma automática, sin necesidad de intervención humana, lo que permite
ahorrar mucho tiempo. Otra de sus funciones es aumentar la seguridad e integridad de la información. Esto lo consiguen gracias a la programación de restricciones o requerimientos de verificación que permiten minimizar los
errores y sincronizar la información.

¿Cuándo se puede usar un Trigger?

Los trigger se puede ejecutar cuando el usuario realizar alguna acción relacionada con añadir, actualizar o eliminar información de una tabla. Es decir, al usar los comandos INSERT, UPDATE o DELETE. Por tanto, para poder usar un trigger es necesario que el usuario posea permisos INSERT y DELETE e dicha base de datos.

Triggers en SQL

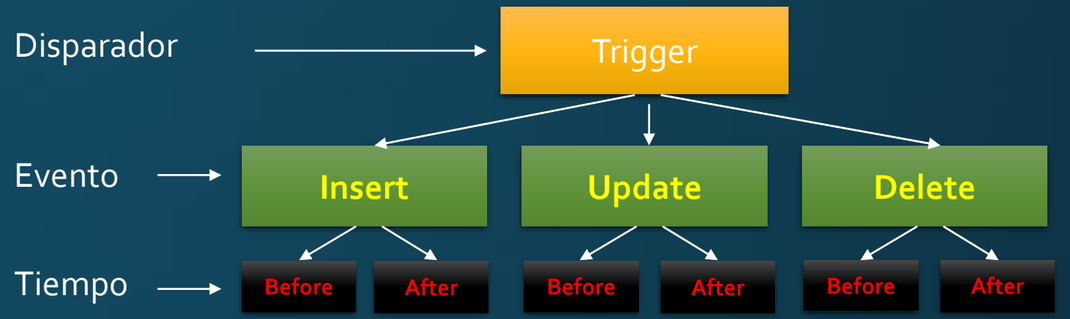
Tablas

```
create table Usuarios(
Folio varchar(50),
Nombre varchar(50),
Apellido varchar(50),
Telefono varchar(50),
Direccion varchar(50),
Edad int);
```

```
create table Originales_Usuarios (Folio
varchar(50),Nombre varchar(50),Apellido
varchar(50),Telefono varchar(50),Direccion
varchar(50),Edad int,Folio_Ant
varchar(50),Nombre_Ant
varchar(50),Apellido_Ant
varchar(50),Telefono_Ant
varchar(50),Direccion_Ant
varchar(50),Edad_Ant int);
```

```
create table DatosContacto(
Folio varchar(50),
Nombre varchar(50),
Telefono varchar(50));

create table Movimientos(
Usuario varchar(50),
Movimiento varchar(10),
Fecha date);
```



Ejemplos

Insert

```
delimiter //
create trigger  validar_edad_negativa before insert on Usuarios for each row begin
   if new.edad < 0 then set new.edad = -new.edad; end if;
   if new.edad > 50 then set new.edad = 50; end if;  end; //
delimiter ;
```

```
create trigger Tr_DatosContacto after insert on Usuarios for each row insert into DatosContacto (Folio,Nombre,Telefono) values
(new.Folio,new.Nombre,new.Telefono);
```

Update

```
create trigger Respaldo_Usuarios before update on Usuarios for each row insert into Originales_Usuarios
(Folio,Nombre,Apellido,Telefono,Direccion,Edad,Folio_Ant,Nombre_Ant,Apellido_Ant,Telefono_Ant,Direccion_Ant,Edad_Ant) values
(new.Folio,new.Nombre,new.Apellido,new.Telefono,new.Direccion,new.Edad,old.Folio,old.Nombre,old.Apellido,old.Telefono,old.Direccion,old.Edad);
```

```
create trigger Movimiento_Up after update on Usuarios for each row insert into Movimientos(Usuario,Movimiento,Fecha) values (USER(),'Actualizar',NOW());
```

Delete

```
create trigger Respaldo_Usuarios before delete on Usuarios for each row insert into Originales_Usuarios
(Folio_Ant,Nombre_Ant,Apellido_Ant,Telefono_Ant,Direccion_Ant,Edad_Ant) values
(old.Folio,old.Nombre,old.Apellido,old.Telefono,old.Direccion,old.Edad);
```

```
create trigger Movimiento_Del after delete on Usuarios for each row insert into Movimientos(Usuario,Movimiento,Fecha) values (USER(),'Borrar',NOW());
```

El DELIMITER es un comando en SQL que se utiliza para cambiar el delimitador predeterminado (;) que separa las sentencias SQL en un script. La necesidad de cambiar el delimitador surge cuando estás creando objetos más complejos en SQL, como funciones, procedimientos almacenados o triggers, que contienen bloques de código que a su vez pueden contener múltiples sentencias SQL.

El delimitador predeterminado (;) se usa para separar cada sentencia SQL en un script y decirle al motor de la base de datos que debe ejecutar cada sentencia de manera independiente. Sin embargo, cuando estás definiendo un objeto que contiene bloques de código (como un procedimiento almacenado) y este bloque interno también contiene punto y coma (;), puede causar problemas de interpretación, ya que el motor podría pensar que el bloque interno está terminando.

Por lo tanto, se usa el comando DELIMITER para temporalmente cambiar el delimitador a un valor distinto, como //, $$, ||, etc. Esto permite que el motor de la base de datos interprete correctamente los bloques de código internos y externos.

Consultas de Acción

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir, borrar y modificar registros. Tanto las sentencias de actualización como las de borrado desencadenan (según el motor de datos) las actualizaciones en cascada, borrados en cascada, restricciones y valores por defecto definidos para los diferentes
campos o tablas afectadas por la consulta.

Las bases de datos relacionales están formadas por un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. Cada relación posee una estructura similar el SQL permite el uso de valores nulos para indicar que el valor es desconocido o no existe. También permite al usuario especificar los atributos que no pueden contener valores nulos. La estructura básica de una expresión SQL consta de tres cláusulas: select, from y where.

- La cláusula select se corresponde con la operación proyección del álgebra relacional. Se usa para obtener una relación de los atributos deseados en el resultado de una consulta.
- La cláusula from se corresponde con la operación producto cartesiano del álgebra relacional. Genera una lista de las relaciones que deben ser analizadas en la evaluación de la expresión.
- La cláusula where se corresponde con el predicado selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula from.

Creación de tablas

Las tablas se utilizan para almacenar datos en la base de datos. Las tablas tienen nombres únicos dentro de una base de datos y un esquema. Cada tabla contiene una o más columnas y cada columna tiene un tipo de datos asociado que define el tipo de datos que puede almacenar, por ejemplo, números, cadenas o datos temporales.

```
CREATE TABLE nombre_tabla (
      nombre_columna1 tipo_datos(longitud) restricciones,
      nombre_columna2 tipo_datos(longitud),
         .....
      nombre_columnaN tipo_datos(longitud)
      )
```

Esta es la forma básica de crear una tabla: ponemos las palabras claves CREATE TABLE seguidos del nombre de la tabla, y a continuación entre paréntesis escribimos las columnas de la tabla separadas por comas. Para cada columna pondremos en primer lugar el nombre seguido de la descripción del tipo de datos. La mayoría de los tipos de datos requieren que pongamos después la longitud máxima que puede tener (en número de caracteres) el dato a insertar. Opcionalmente podemos insertar (y a veces debemos hacerlo), unas "restricciones".

Actualización: Update es la instrucción del lenguaje SQL que nos sirve para modificar los registros de una tabla. Como para el caso de Delete, necesitamos especificar por medio de Where cuáles son los registros en los que queremos hacer efectivas nuestras modificaciones. Además, obviamente, tendremos que especificar cuáles son los nuevos valores de los campos que deseamos actualizar.
La sintaxis es de este tipo:

```
UPDATE nombre_tabla SET columna1 = 'nuevo_valor' WHERE columna1 = 'valor1';
```

Eliminación: La instrucción DELETE permite eliminar uno o múltiples registros. Incluso todos los registros de una tabla, dejándola vacía.
Su sintaxis es general es:

```
1  DELETE [FROM] NombreTabla
2  WHERE Condición
```

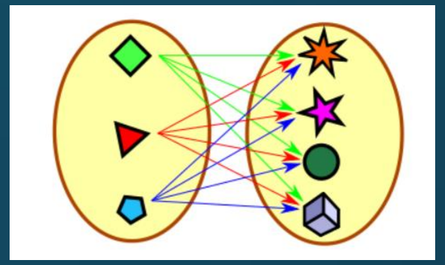
La condición, como siempre, define las condiciones que deben cumplir los registros que se desean eliminar. Se puede aplicar todo lo visto para esta cláusula anteriormente, incluidas las sub-consultas.

Inserción: INSERT es una sentencia SQL que añade datos a una tabla.
La sentencia INSERT tiene el formato siguiente:

```
INSERT INTO nombtabla
VALUES (valor1, valor2, ...)
```

Inserción: INSERT es una sentencia SQL que añade datos a una tabla.
La sentencia INSERT tiene el formato siguiente:

```
INSERT INTO nombtabla
VALUES (valor1, valor2, ...)
```



En esta sintaxis, nombtabla es el nombre de la tabla o vista en la que se desea insertar datos y valor1, valor2 (etc.), son los valores que va a insertar. La lista de valores de datos después de VALUES debe corresponderse con la lista de columnas de la tabla en la que van a ser insertados. Debe haber el mismo número de valores que de columnas, y cada valor debe tener un tipo de datos que coincida con el de su columna. Como se indica en el ejemplo siguiente, los valores nulos se pueden insertar especificando NULL.

Consultas de selección:
Cuando quiera seleccionar datos específicos de uno o varios orígenes, use una consulta de selección. Una consulta de selección le ayuda a recuperar únicamente los datos que le interesen y a combinar datos de varios orígenes. Puede usar tablas y otras consultas de selección como orígenes de datos para una consulta de selección. En este tema se facilita una descripción general de las consultas de selección y se indican los pasos para crear una en vista Diseño o mediante el Asistente para consultas.

Consulta Multi tabla
Las consultas multi tabla nos permiten consultar información en más de una tabla. La única diferencia respecto a las consultas sencillas es que vamos a tener que especificar en la cláusula FROM cuáles son las tablas que vamos a usar y cómo las vamos a relacionar entre sí. Composiciones cruzadas (Producto cartesiano) El producto cartesiano de dos conjuntos, es una operación que consiste en obtener otro conjunto cuyos elementos son todas las parejas que pueden formarse entre los dos conjuntos. Por ejemplo, tendríamos que coger el primer elemento del primer conjunto y formar una pareja con cada uno de los elementos del segundo conjunto. Una vez hecho esto, repetimos elmismo proceso para cada uno de los elementos del primer conjunto.

```
SELECT *
FROM empleado;
```

codigo	nif	nombre	apellido1	apellido2	codigo_departamento
1	32481596F	Aarón	Rivero	Gómez	1
2	Y5575632D	Adela	Salas	Díaz	2
3	R6970642B	Adolfo	Rubio	Flores	3

```
SELECT *
FROM departamento;
```

codigo	nombre	presupuesto
1	Desarrollo	120000
2	Sistemas	150000
3	Recursos Humanos	280000

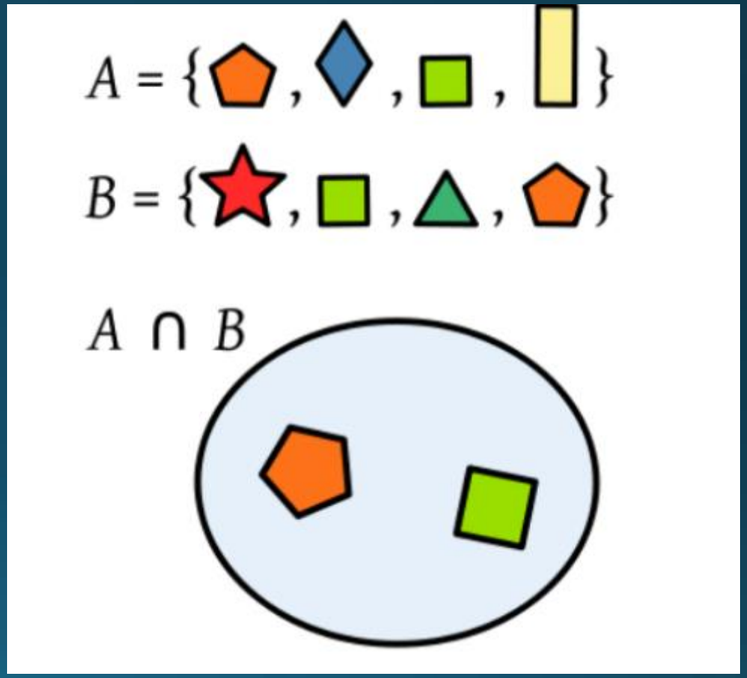
El producto cartesiano de las dos tablas se realiza con la siguiente consulta:

```
SELECT *
FROM empleado, departamento;
```

EL resultado es el siguiente:

codigo	nif	nombre	apellido1	apellido2	codigo_departamento	codigo	nombre	presupuesto
1	32481596F	Aarón	Rivero	Gómez	1	1	Desarrollo	120000
2	Y5575632D	Adela	Salas	Díaz	2	1	Desarrollo	120000
3	R6970642B	Adolfo	Rubio	Flores	3	1	Desarrollo	120000
1	32481596F	Aarón	Rivero	Gómez	1	2	Sistemas	150000
2	Y5575632D	Adela	Salas	Díaz	2	2	Sistemas	150000
3	R6970642B	Adolfo	Rubio	Flores	3	2	Sistemas	150000
1	32481596F	Aarón	Rivero	Gómez	1	3	Recursos Humanos	280000
2	Y5575632D	Adela	Salas	Díaz	2	3	Recursos Humanos	280000
3	R6970642B	Adolfo	Rubio	Flores	3	3	Recursos Humanos	280000

Composiciones internas (Intersección)
La intersección de dos conjuntos es una operación que resulta en otro conjunto que contiene sólo los elementos comunes que existen en ambos conjuntos.



Ejemplo:
Para poder realizar una operación de intersección entre las dos tablas debemos utilizar la cláusula WHERE para indicar la columna con la que queremos relacionar las dos tablas. Por ejemplo, para obtener un listado de los empleados y el departamento donde trabaja cada uno podemos realizar la siguiente consulta:

```
SELECT *
FROM empleado, departamento
WHERE empleado.codigo_departamento = departamento.codigo
```

El resultado sería el siguiente:

```
+--------+-----------+--------+-----------+-----------+--------------------+--------+-----------------+------------
| codigo | nif       | nombre | apellido1 | apellido2 | codigo_departamento | codigo | nombre          | presupuesto
+--------+-----------+--------+-----------+-----------+--------------------+--------+-----------------+------------
| 1      | 32481596F | Aarón  | Rivero    | Gómez     | 1                  | 1      | Desarrollo      | 120000
| 2      | Y5575632D | Adela  | Salas     | Díaz      | 2                  | 2      | Sistemas        | 150000
| 3      | R6970642B | Adolfo | Rubio     | Flores    | 3                  | 3      | Recursos Humanos | 280000
+--------+-----------+--------+-----------+-----------+--------------------+--------+-----------------+------------
```

Consultas de Agrupación:
La cláusula GROUP BY te permite organizar las filas de una consulta en grupos. Los grupos están determinados por las columnas que se especifican en la cláusula GROUP BY
A continuación, se ilustra la sintaxis de la cláusula GROUP BY:

```
SELECT select_list FROM table_name
GROUP BY    column_name1,   column_name2 ,...;
```

En esta consulta, la cláusula GROUP BY regresa un grupo para cada combinación de los valores en las columnas enumeradas en la cláusula GROUP BY.