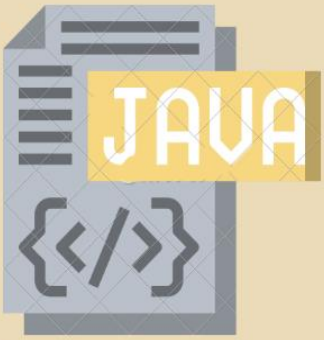
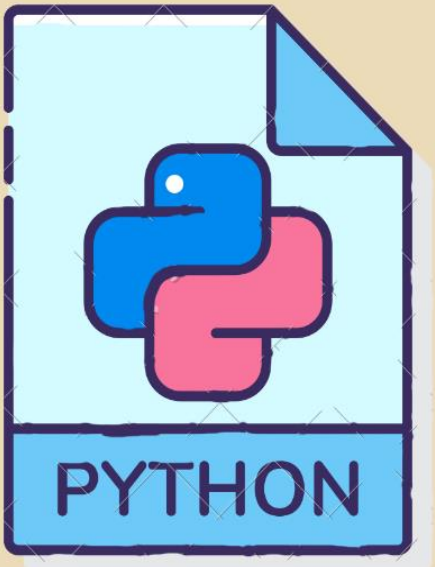


Resumen de clases ISPC



Conceptos fundamentales

Una base de datos (database) almacena datos y los conecta en una unidad lógica junto a los metadatos necesarios para su procesamiento.

Siendo los datos y el almacenamiento de los mismos, un pilar fundamental de la vida moderna.

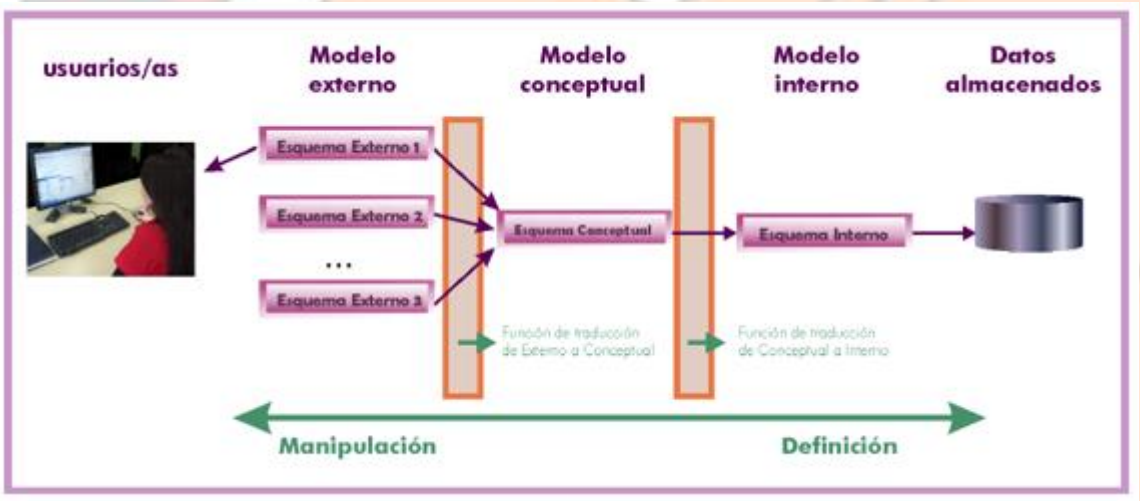
Una Base de datos es un “Sistema” (conjunto de programas o aplicaciones) que permite guardar un conjunto de información almacenada, que nos permite consultar, insertar, modificar y borrar esa información en forma **“SISTEMATIZADA”.**

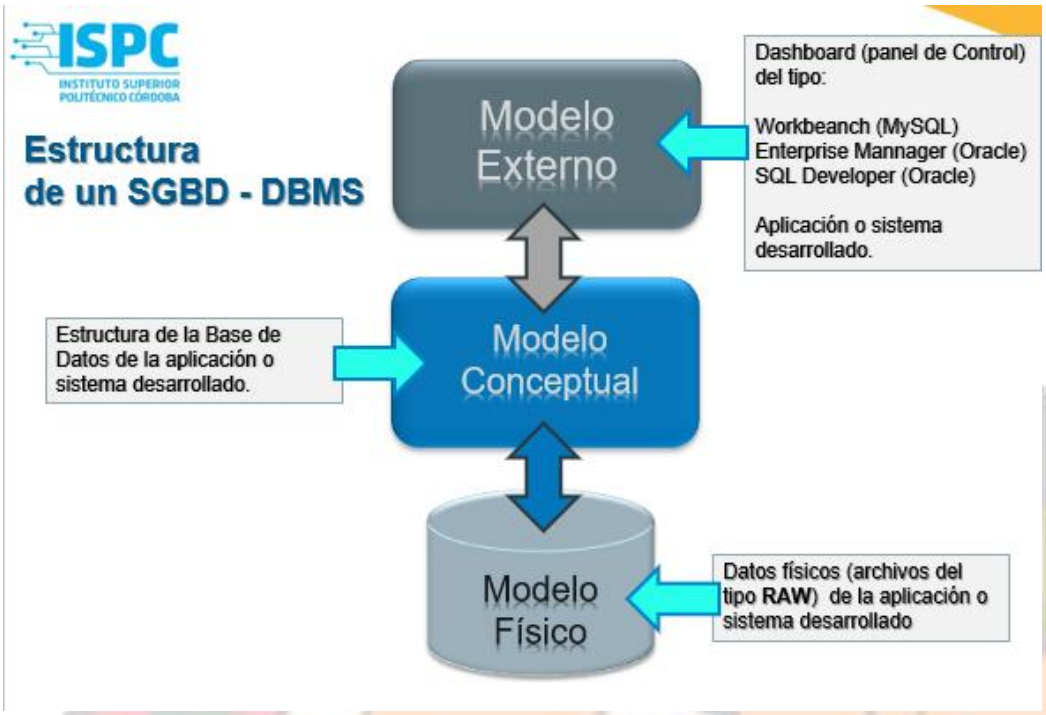
Un sistema de Gestión de Bases de datos (**SGBD Sistema Gestor de Base de Datos**) es un tipo de software muy específico o lo que es lo mismo, es una agrupación de programas que sirven para definir, construir y manipular una base de datos, permitiendo así almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

¿Qué es un Sistema de Gestión de base de datos (DBMS)?

Es un software que permite la creación y administración de Base de Datos y además actúa como un intermediario entre los usuarios, las aplicaciones y la propia base de datos.

Estructura





Edgar Frank "Ted" Codd

En las décadas de los sesenta y los setenta trabajó en sus teorías sobre modelado de datos, publicando su trabajo **"Un modelo relacional de datos para grandes bancos de datos compartidos"**

Codd definió las tres primeras formas normales que se aplican para la normalización de bases de datos. Además, la forma normal de Boyce-Codd lleva el nombre en su honor.

También acuñó el término OLAP y redactó las 12 reglas de Codd del modelo relacional para las bases de datos.

Peter P. Chen

y el modelo Entidad / Relación

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue creado por Peter Chen 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.

este modelo no tiene nada que ver con las bases de datos relacionales, los esquemas entidad/relación se pueden utilizar con cualquier SGBD ya que son conceptuales. Confunde el uso de la palabra relación, pero el concepto de relación en este esquema no tiene nada que ver con la idea de relación expuesta por Codd en su modelo relacional.

Una **Entidad** es 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.

Es importante recordar que una entidad no es un propiedad concreta sino un objeto que puede poseer múltiples propiedades. Entonces una entidad es un objeto concreto, no un simple dato: el Smartphone que tienen en sus bolsillos es una entidad, “Samsung” sin embargo es la marca de ese Smartphone, es decir es un **atributo** de esa entidad.

Relación

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. *Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.*

Existen distintos tipos de relaciones en las cuales se encuentra:

Relaciones Binarias: Son las relaciones típicas. Se trata de relaciones que asocian dos entidades.

Relaciones Ternarias: Relacionan tres entidades. A veces se pueden simplificar en relaciones binarias, pero no siempre es posible.

Relaciones n-arias: Relacionan n entidades

Relaciones dobles: Se llaman así a dos relaciones distintas que sirven para relacionar a las mismas relaciones. Son las más difíciles de manejar ya que al manipular las entidades hay que elegir muy bien la relacionan a utilizar para relacionar los datos.

Relación reflexiva: Es una relación que sirve para relacionar ejemplares de la misma entidad (personas con personas, piezas con piezas, etc.)

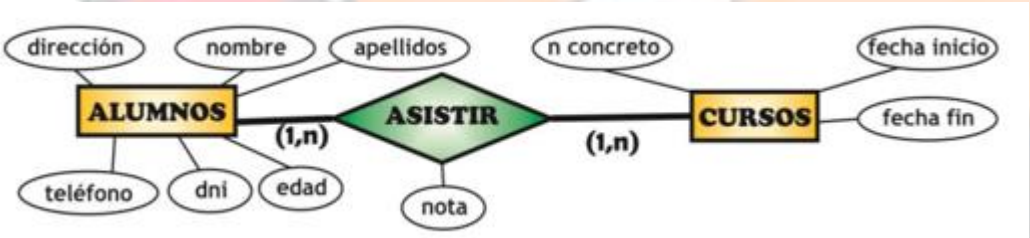
Cardinalidad

Indica el número de relaciones en las que una entidad puede aparecer. Se anota en términos de:

- **cardinalidad mínima**. Indica el número mínimo de asociaciones en las que aparecerá cada ejemplar de la entidad (el valor que se anota es de cero o uno, aunque tenga una cardinalidad mínima de más de uno, se indica sólo un uno)
- **cardinalidad máxima.** Indica el número máximo de relaciones en las que puede aparecer cada ejemplar de la entidad. Puede ser uno, otro valor concreto mayor que uno (tres por ejemplo) o muchos (se representa con n). Normalmente la cardinalidad máxima es 1 ó n

Atributo

Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. *se representan mediante elipses que cuelgan de las entidades o relaciones a las que pertenecen*. Cada atributo tiene un conjunto de valores asociados denominado dominio. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio.



Estos atributos se dividen en

Atributo Compuesto: Este atributo esta compuesto por otros atributos, si tomamos el ejemplo anterior, el atributo fecha inicio estaría compuesto por otros tres atributos: día, mes y año.

Atributo múltiple: Pueden tomar varios valores, por ejemplo un Alumno puede tener varios teléfonos, por lo tanto este seria un atributo múltiple. Cabe destacar que debe tener una cardinalidad mínima de 1 y máxima de n.

Atributo opcional: Lo son si pueden tener valor nulo. Siguiendo el mismo ejemplo anterior si quisiéramos que el teléfono fuera un atributo no obligatorio, podríamos marcar una cardinalidad mínima de 0 (nulo) y una máxima de n.

El Modelo Relacional

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

El modelo relacional desarrolla un esquema de base de datos (data base schema) a partir del cual se podrá realizar el modelo físico o de implementación en el DBMS.

Este modelo esta basado en que todos los datos están almacenados en tablas (entidades/relaciones) y cada una de estas es un conjunto de datos, por tanto una base de datos es un conjunto de relaciones. La agrupación se origina en la tabla: tabla -> fila (tupla) -> campo (atributo).

El Modelo Relacional se ocupa de:

- La estructura de datos
- La manipulación de datos
- La integridad de los datos

Donde las relaciones están formadas por :

- Atributos (columnas)

Tuplas (Conjunto de filas)

Existen dos formas para la construcción de modelos relacionales:

- Creando un conjunto de tablas iniciales y aplicando operaciones de normalización hasta conseguir el esquema más óptimo,
- O, convertir el modelo entidad relación (**ER**) en tablas, con una depuración lógica y la aplicación de restricciones de integridad.

Los objetivos que este modelo persigue son:

Independencia Física: La forma de almacenar los datos no debe influir en su manipulación. Si el almacenamiento físico cambia, los usuarios que acceden a esos datos no tienen que modificar sus aplicaciones.

Independencia Lógica: Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se inserten, actualicen y eliminen datos.

Flexibilidad: En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera.

Uniformidad: Las estructuras lógicas de los datos siempre tienen una única forma conceptual (las tablas), lo que facilita la creación y manipulación de la base de datos por parte de los usuarios.

Sencillez: Las características anteriores hacen que este Modelo sea fácil de comprender y de utilizar por parte del usuario final.

Relación o tabla

Según el modelo relacional (desde que Codd lo enunció) el elemento fundamental es lo que se conoce como **relación**, Las relaciones constan de:

- ✓ **Atributos**. Referido a cada propiedad de los datos que se almacenan en la relación (nombre, dni,...).
- ✓ **Tuplas.** Referido a cada elemento de la relación. Por ejemplo si una relación almacena personas, una tupla representaría a una persona en concreto.

Puesto que una relación se representa como una tabla; podemos entender que las columnas de la tabla son los atributos; y las filas, las tuplas.

Tupla o registro.

Cada una de las filas de la relación. Se corresponde con la idea clásica de registro. Representa por tanto cada elemento individual de esa relación.

Tiene que cumplir que:

- ✓ Cada tupla se debe corresponder con un elemento del mundo real.
- ✓ No puede haber dos tuplas iguales (con todos los valores iguales).
- ✓ **Atributo – Columnas.**
- ✓ Un Atributo en el Modelo Relacional representa una propiedad que posee esa Relación y equivale al atributo del Modelo E-R.
- ✓ Se corresponde con la idea de campo o columna.
- ✓ En el caso de que sean varios los atributos de una misma tabla, definidos sobre el mismo dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre.
- ✓ **Clave candidata**
- ✓
- ✓ Conjunto de atributos que identifican unívocamente cada tupla de la relación. Es decir columnas cuyos valores no se repiten en ninguna otra tupla de esa tabla. Toda tabla en el modelo relacional debe tener al menos una clave candidata (puede incluso haber más)
- ✓ **Clave primaria**
- ✓ Clave candidata que se escoge como **identificador** de las tuplas. Se elige como primaria la candidata que identifique mejor a cada tupla en el contexto de la base de datos.
- ✓ **Clave alternativa**
- ✓ Cualquier clave candidata que no sea primaria.

Clave externa, ajena o secundaria

Son los datos de atributos de una tabla cuyos valores están relacionados con atributos de otra tabla

Historia y evolución de los Sistemas Gestores de Bases de Datos

Los orígenes de las bases de datos se remontan a la Antigüedad donde ya existían bibliotecas y toda clase de registros. Además también se utilizaban para recoger información sobre las cosechas y censos. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual.

En 1884 Herman Hollerith creó la máquina automática de tarjetas perforadas, siendo nombrado así el primer ingeniero estadístico de la historia. En esta época, los censos se realizaban de forma manual.

Ante esta situación, Hollerith comenzó a trabajar en el diseño de una maquina tabuladora o censadora, basada en tarjetas perforadas.

Decada de los 50

- ✓ Procesamiento centralizado
- ✓Almacenamiento en cintas magnéticas
- ✓Acceso secuencial
- ✓Procesamiento de la información por lotes BATCH
- ✓Monousuario y Monotarea
- ✓Se llamaron en un primer momento Data Banks

Decada de los 60

✓ Invención del transistor

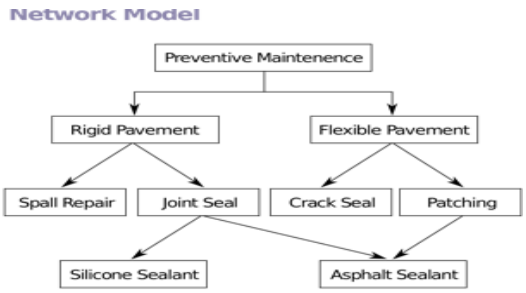
✓ Nacimiento de los circuitos integrados (microcontroladores)

✓ Popularización de los discos rígidos como dispositivos de almacenamiento

✓ Procesamiento centralizado

✓Aparición de las primeras bases de datos jerarquicas y de red

✓Primeras conexiones telefónicas



CODASYL

Decada de los 90

- ✓ Creación y desarrollo de la primera Red de redes tipo *"***WAN"**, llamada **Internet**
- ✓ Desarrollo de los primeros **SGBD "Distribuidos"**
- ✓ Aparece la tecnología de la arquitectura **Cliente – Servidor**, impactando fuertemente en los SGBD del momento y en los del futuro.
- ✓ Nace un nuevo proyecto de SGBD libre (código abierto) del tipo GPL (Licencia Pública General), llamado **MySQL.** Como proyecto derivado nace **MariaDB**
- ✓ El éxito de las BD, incluso en sistemas personales, ha llevado a la aparición de los Fourth Generation Languages (4GL), lenguajes muy fáciles y fundamentadas en BD.
- ✓ Nace el SGBD **PostgreSQL** derivado del proyecto Ingres y los SGBD Orientados a Objetos.

Del 2000 a nuestra época

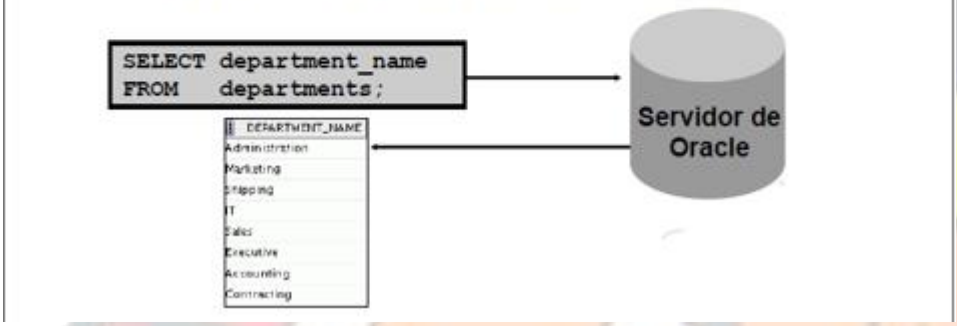
- ✓ Con la creación y desarrollo de la primera Red de redes tipo *"***WAN"**, llamada **Internet**, los SGBD toman muchas nuevas configuraciones de acuerdo al tipo de prestación
- ✓ Surgen nuevos paradigmas de cómo resguardar la información
- ✓ Se hacen presentes conceptos como **Big Data** y SGBD del tipo **NoSQL.** No estructurados en base al concepto del modelo relacional.
- ✓ Aparecen SGBD que sus servicios pueden ser accedidos desde la nube, como el proyecto de **Firebird**
- ✓ Empresas denominadas BigTech, se hacen del control del manejo de grandes cantidades de volúmenes de información. Al tener recursos suficientes, crean **Servicios en la nube** para el resguardo completo de la información de un SI (Sistema de Información). Con una infinidad de herramientas y tecnologías a tal efecto, se convierten en los amos y señores del manejo de la información. Algunos de los ejemplos son: **AWS** de Amazon, **Azure** de Microsoft, **GCP** de Google. Haciendo que el concepto de Base de datos cambie en forma radical.

Introducción a SQL ANSI

Uso de SQL para Consultar Base de Datos

El lenguaje de consulta estructurado (SQL) es:

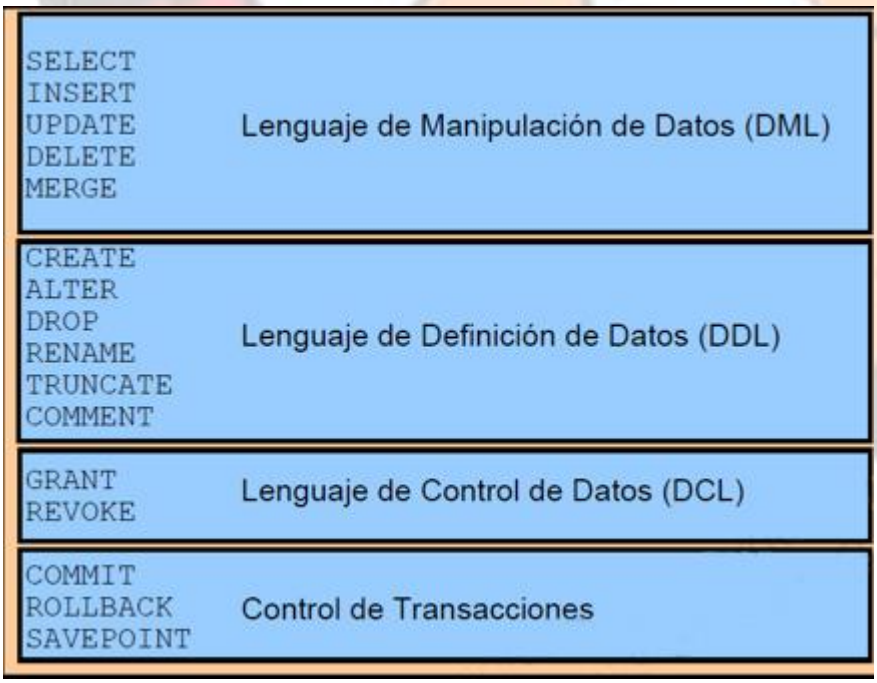
- Lenguaje estándar de ANSI para el funcionamiento de bases de datos relacionales
- Uso y aprendizaje sencillos y eficaces
- Funcionalidad completa (con SQL, puede definir, recuperar y manipular datos en las tablas)

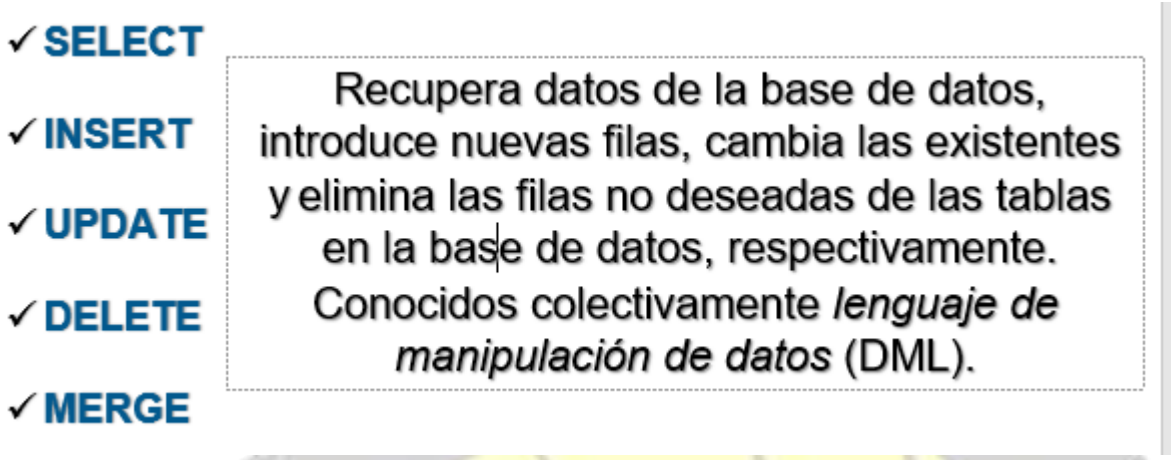


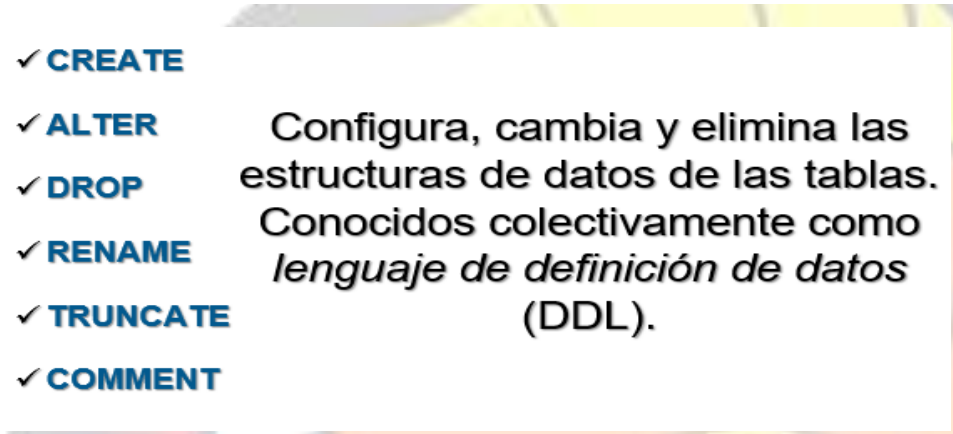
SQL proporciona sentencias para distintas tareas, que incluyen las siguientes:

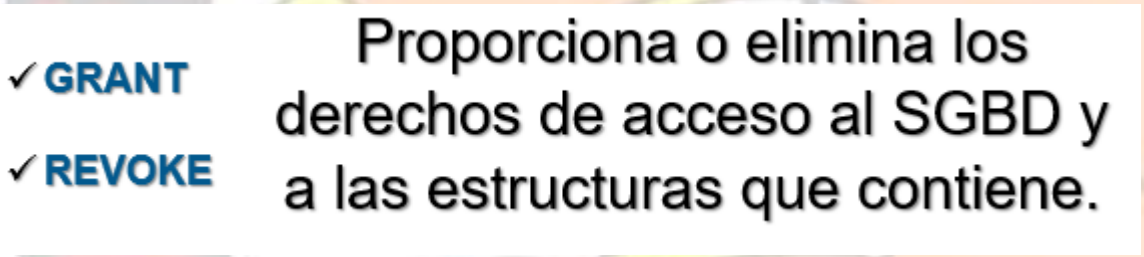
- **Consulta de datos**
- **Inserción, actualización y supresión de filas en una tabla**
- **Creación, sustitución, modificación y borrado de objetos**
- **Control de acceso a la base de datos y los objetos**
- **Garantía de integridad y consistencia de la base de datos**

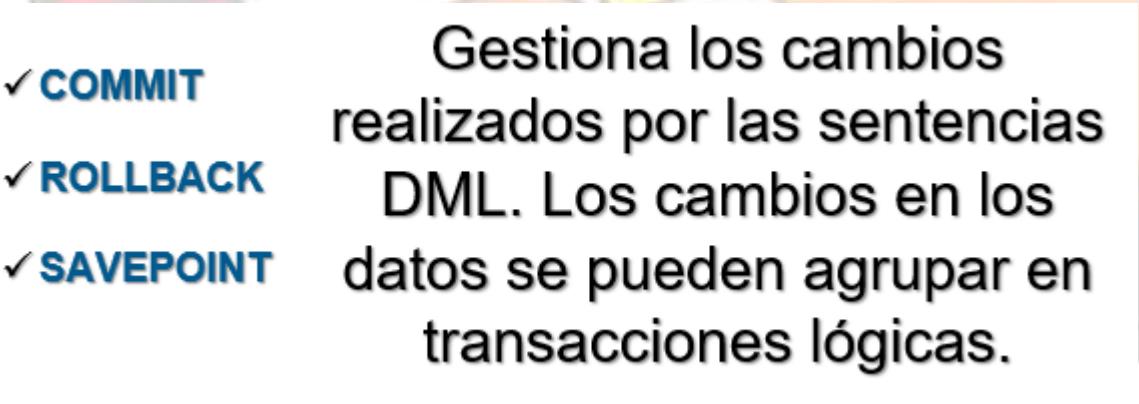
SQL unifica todas las tareas anteriores en un lenguaje consistente y permite trabajar con datos en el nivel lógico.











Problemas en el diseño de los modelos

Redundancia. Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos. Cuando es excesiva es evidente que el diseño hay que revisarlo, es el primer síntoma de problemas y se detecta fácilmente.

Ambigüedades. Datos que no clarifican suficientemente el elemento al que representan. Los datos de cada fila podrían referirse a más de un ejemplar de esa tabla o incluso puede ser imposible saber a qué ejemplar exactamente se están refiriendo. Es un problema muy grave y difícil de detectar.

Pérdida de restricciones de integridad. Normalmente debido a dependencias funcionales. Más adelante se explica este problema. Se arreglan fácilmente siguiendo una serie de pasos concretos.

Anomalías en operaciones de modificación de datos. El hecho de que al insertar un solo elemento haya que repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas necesariamente (por ejemplo que eliminar un cliente suponga borrar seis o siete filas de la tabla de clientes, sería un error muy grave y por lo tanto un diseño terrible).

¿Cuál es el objetivo de la normalizacion?

Normalización

Proceso de simplificación de los datos

- *Tener almacenado con el menor espacio posible.*
- *Eliminar datos repetidos.*
- *Eliminar errores lógicos.*
- *Datos ordenados*

Primer Forma Normal
Segunda Forma Normal
Tercer Forma Normal ←
Forma Normal Boyce Codd
Cuarta Forma Normal
Quinta Forma Normal

La simplificación debe darse sin que haya pérdida de información.

Dependencias

Segunda Forma Normal

- *La tabla debe estar en Primer Forma Normal*
- *Identificar las dependencias funcionales y transitivas.*

Dependencia Funcional

Dependencia Transitiva

¿Qué es git/github?

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

Este sistema te ofrece la posibilidad de **colaborar en otros proyectos y publicar los tuyos propios**.

La plataforma es de **código abierto** por defecto, por lo que cualquier persona puede utilizar tu código y tú también puedes ver el código de otros proyectos

¿Qué es una rama?

En cada confirmación de cambios (commit), Git almacena una instantánea de tu trabajo preparado. Dicha instantánea contiene además unos metadatos con el autor y el mensaje explicativo, y uno o varios apuntadores a las confirmaciones (commit) que sean padres directos de esta (un padre en los casos de confirmación normal, y múltiples padres en los casos de estar confirmando una fusión (merge) de dos o más ramas).

Cuando creas una confirmación con el comando git commit, Git realiza sumas de control de cada subdirectorio (en el ejemplo, solamente tenemos el directorio principal del proyecto), y las guarda como objetos árbol en el repositorio Git. Después, Git crea un objeto de confirmación con los metadatos pertinentes y un apuntador al objeto árbol raíz del proyecto.

Ramas Remotas

Las ramas remotas son referencias al estado de las ramas en tus repositorios remotos. Son ramas locales que no puedes mover; se mueven automáticamente cuando estableces comunicaciones en la red. Las ramas remotas funcionan como marcadores, para recordarte en qué estado se encontraban tus repositorios remotos la última vez que conectaste con ellos.

COMANDOS DE GIT

El **comando git add** añade un cambio del directorio de trabajo en el entorno de ensayo. De este modo

El **comando git init** crea un nuevo repositorio de **Git**. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de **Git**, o para inicializar un nuevo repositorio vacío.

El **comando Git clone** es un **comando** para descargarte el código fuente existente desde un repositorio remoto

El **comando git pull** se emplea para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.

El **comando git push** se usa para cargar contenido del repositorio local a un repositorio remoto.

El **comando git merge** permite tomar las líneas independientes de desarrollo creadas por **git** branch e integrarlas en una sola rama