Bases de Datos

Tema 1
Sistemas de Almacenamiento.

Javier Ivorra

1. Introducción.
    1.1 Historia y Evolución Bases de Datos

2. Sistema basado en archivos
    2.1.Tipos de archivos
        2.1.1.Según su contenido
        2.1.2.Según su acceso:
            2.1.2.1 Archivos secuenciales
            2.1.2.2 Archivos aleatorios
            2.1.2.3 Archivos indexados

    2.2. Cómo se almacenaba la información antes de los SGBD
    2.3.Inconvenientes de un sistema de gestión de archivos

3. Sistemas Gestores de Bases de Datos (SGBD)
    3.1.Características de un SGBD
    3.2.Ventajas SGBD sobre sistema de archivos
    3.3. Concepto de transacción
    3.4.Arquitectura SGBD
    3.5.Funciones SGBD
    3.6.Componentes de un SGBD
        3.6.1.Usuarios
        3.6.2. Consultas y almacenamiento
        3.6.3 Implementación Física del sistema
        3.6.4 Ejecución de procesos en un SGBD

4.Tipos de SGBD

5.Fases del diseño de una BD.

Introducción

Las Bases de Datos se desarrollan cuando el volumen de datos alto y tiempo de respuesta que necesitamos es vital. Hay dos cosas que requieren una optimización: el almacenamiento de datos y la consulta de datos

El uso de estos sistemas incrementa la eficiencia de las operaciones en la empresa y reduce los costes totales.

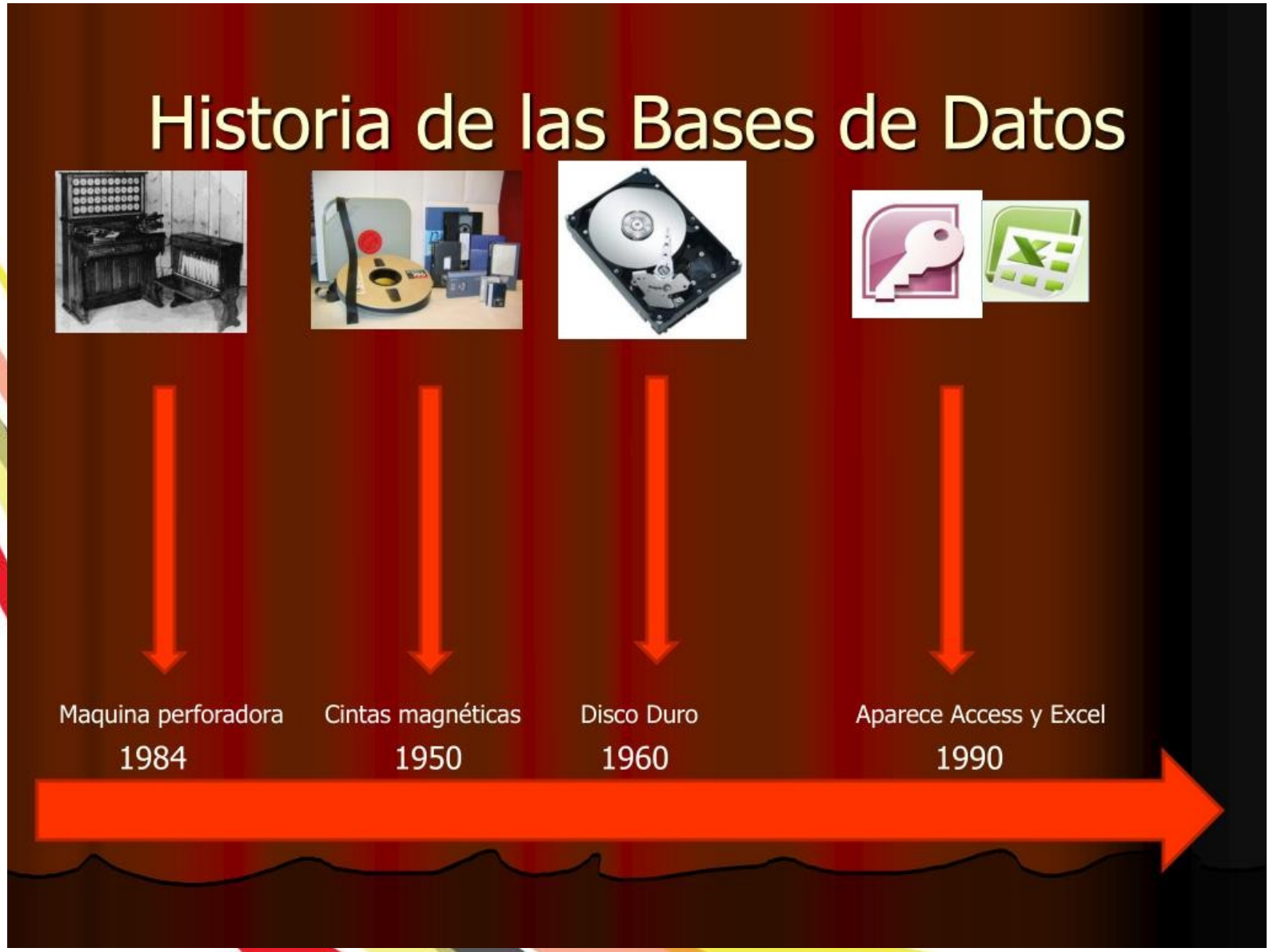
Hoy en día prácticamente todas las empresas usan bases de datos. Cuando compras en un supermercado varias bases de datos están presentes:

La base de datos del almacén cambiará la cantidad de productos.

El código de barras será consultado.

Si pagas con tarjeta de crédito, ésta actualizará la cantidad en tu cuenta corriente.

1.1 Historia de las Bases de Datos



1.1 Historia de las Bases de Datos

En 1884 Herman Hollerit creó la máquina perforadora, con la que consiguió bajar de 7 años a 2 años el tiempo en censar a la población de USA. Compuesta por un perforadora y una lectora que leía orificios.

1.1 Historia de las Bases de Datos

•Década de 1950. Se inventó la cinta magnética. Se empezó a automatizar la información de las nóminas, comienzan a usarse bases de datos basadas en sistemas de archivos secuenciales. Estas cintas solo se podían leer secuencial y ordenadamente.

1.1 Historia de las Bases de Datos

Década de 1960. El uso de los discos en ese momento fue un adelanto muy efectivo, ya que, por medio de este soporte se podía consultar la información directamente usando ficheros aleatorios e indexados, donde el acceso ya no tiene por qué ser secuencial. El sistema de ficheros era el sistema más común de almacenamiento de datos, lo que ayudó a ahorrar tiempo. No era necesario saber exactamente donde estaban los datos en los discos.

Primer disco duro de IBM 1961. Pesaba una tonelada y almacenaba 5 MB.

Disco duro actual

1.1 Historia de las Bases de Datos

Antigüedad: Los seres humanos comenzaron a almacenar información hace mucho tiempo. En la antigüedad, los elaborados sistemas de bases de datos fueron desarrollados por oficinas gubernamentales, bibliotecas, hospitales y organizaciones empresariales, y algunos de los principios básicos de estos sistemas todavía se utilizan hoy en día.

Los años 60: La base de datos por ordenador comenzó en los años 60, cuando el uso de ordenadores se convirtió en una opción más rentable para las organizaciones privadas. Hubo dos modelos de datos populares en esta década: un modelo de red llamado **CODASYL** y un modelo **jerárquico** llamado **IMS**. Un sistema de base de datos que resultó ser un éxito comercial fue el sistema **SABRE** que fue utilizado por **IBM** para ayudar a American Airlines a gestionar sus datos de reservas.

1970 a 1972: E.F. Codd publicó un importante artículo para proponer el uso de un modelo de base de datos relacional, y sus ideas cambiaron la forma en que la gente pensaba acerca de las bases de datos. En su modelo, el esquema de la base de datos, u organización lógica, se desvincula del almacenamiento de información física, y esto se convirtió en el principio estándar para los sistemas de bases de datos.

1.1 Historia de las Bases de Datos

Década de 1970: Se crearon dos prototipos principales de sistemas de bases de datos relacionales entre los años 1974 y 1977, y fueron Ingres y System R. Ingres utilizó un lenguaje de consulta conocido como QUEL, que llevó a la creación de sistemas como Ingres Corp., **MS SQL Server**, Sybase, PACE. Por otro lado, System R utilizó el lenguaje de consulta SEQUEL, y contribuyó al desarrollo de **SQL / DS**, DB2, Allbase, **Oracle** y Non-Stop SQL. Fue también en esta década que **Relational Database Management System**, o **RDBMS**, se convirtió en un término reconocido.

1976: Un nuevo modelo de base de datos denominado **Entidad-Relación**, o **ER**, fue propuesto por **P. Chen** este año. Este modelo hizo posible que los diseñadores se centraran en la aplicación de datos, en lugar de la estructura lógica de la tabla.

1980s: Structured Query Language, o **SQL**, se convirtió en el lenguaje de consulta estándar. Los sistemas de bases de datos relacionales se convirtieron en un éxito comercial, ya que el rápido aumento de las ventas de computadoras impulsó el mercado de las bases de datos, lo que provocó una importante disminución en la popularidad de la red y los modelos de bases de datos jerárquicos. DB2 se convirtió en el principal producto de base de datos para IBM, y la introducción de **IBM PC** hizo que surgieran muchas nuevas compañías de bases de datos y el desarrollo de productos como PARADOX, RBASE 5000, RIM, Dbase III y IV, OS / y Watcom SQL.

1.1 Historia de las Bases de Datos

Principios de 1990: Después de una industria de bases de datos sacudida, la mayoría de las empresas supervivientes vendieron productos de base de datos complejos a precios altos. Durante este tiempo, se lanzaron nuevas herramientas de cliente para el desarrollo de aplicaciones, que incluyeron **Oracle Developer**, **PowerBuilder**, **VB** y otros. También se desarrollaron varias herramientas para la productividad personal, como **ODBC** y **Excel** / **Access**. Los prototipos para los sistemas de gestión de bases de datos de objetos, o **ODBMS**, se crearon a principios de los años noventa.

Mediados de los 90: La llegada de Internet llevó al crecimiento exponencial de la industria de la base de datos. Los usuarios de escritorio promedio comenzaron a usar sistemas de base de datos cliente-servidor para acceder a sistemas informáticos que contenían datos heredados.

1.1 Historia de las Bases de Datos

A finales de la década de 1990: El aumento de la inversión en negocios en línea provocó un aumento en la demanda de conectores de bases de datos de Internet, tales como **Front Page, Páginas Active Server, Servidores Java, DreamWeaver, ColdFusion, Enterprise Java Beans y Oracle Developer 2000**. El uso de **cgi, gcc, MySQL, Apache** y otros sistemas trajeron la solución de código abierto a Internet. Con el uso creciente de la tecnología de punto de venta, el procesamiento de transacciones en línea y el procesamiento analítico en línea comenzaron a mejorar.

2000: Aunque la industria de Internet experimentó pronto una disminución en esta década, las aplicaciones de bases de datos continúan creciendo. Se desarrollaron nuevas aplicaciones interactivas para PDA, transacciones de punto de venta y consolidación de proveedores. Actualmente, las tres principales compañías de bases de datos en el mundo occidental son **Microsoft, IBM y Oracle**.

2- Sistema basado en archivos

Como hemos dicho toda base de datos es un conjunto de datos estructurados y almacenados en un medio.

En el ordenador estos datos al final están almacenados en ficheros.

Antes de aparecer los Sistemas Gestores, las primeros Sistemas de Bases de Datos usaron sistemas de ficheros dependientes del S.O.

2.1.Tipos de archivos

¿Qué es un archivo?

•Son estructuras de información que crean los S.O. para poder almacenar datos.

•Suelen tener: <nombre>.<extensión>

Un método popular utilizado por muchos sistemas operativos, Mac OS X, CP / M y DOS, es determinar el formato de un archivo basado en la extensión. Por ejemplo, los documentos HTML se identifican por nombres que terminan con .html (o .htm) y las imágenes GIF por .gif

En el sistema de archivos FAT original (de las versiones MS DOS y Windows previas) los nombres de los archivos estaban limitados a un identificador de 8 caracteres y a una extensión de 3 caracteres.

2.1.1. Según el contenido

- Planos: ficheros de texto o fichero ASCII.
  - Configuración: .ini .inf .conf
  - Código fuente: .sql .c .java
  - Página Web: .html .php . asp .xml
  - Enriquecidos: .rtf .ps. tex

Binarios: no son de texto, y requieren un formato para interpretarlo

- Imagen: jpg .gif .bmp
- Vídeo: .mpg .mov .avi
- Comprimidos: .zip .gz .rar .tar
- Ejectubles: .exe .com .cgi
- Procesadores de texto: .doc .odt

2.1.2. Según su acceso

Según el acceso/organización tenemos tres tipos de archivos principales:

- Archivos secuenciales
- Archivos de acceso aleatorio
- Archivos indexados

2.1.2.1 Archivos secuenciales

- Primeros en aparecer, el medio físico de almacenamiento eran las cintas magnéticas.
- Cada registro tenía unos campos fijos, que se grababan en la cinta sucesivamente, de forma ordenada.
- Muy útil para imprimir etiquetas (en aquella época, de correo, por ejemplo).

2.1.2.1 Archivos secuenciales

- **Características:**

–Lectura ordenada obligatoria

–No permite el retroceso

–Monousuarios

–Estructura rígida de campos

–Lecturas parciales pero escrituras totales

–EOF

–Borrado: hay que reescribir todo menos el registro a borrar (o marcarlo)

–No deja huecos

2.1.2.2 Archivos de acceso aleatorio

- Aparecen con la llegada de los disquetes y discos duros.
- Podemos acceder directamente a la posición que indiquemos del archivo. Ya no hace falta recorrerse todos los anteriores. Pero hay que calcularlo y traducirlo.
- Posición = NumRegistro * LongRegistro

Ejemplos

•Pongamos por caso que tenemos un registro como éste codificado en ANSI (1 byte por carácter):

–Nombre: 80 caracteres ANSI.
–Dirección: 100 caracteres ANSI.
–Población: 50 caracteres ANSI.

•Su longitud será de 230 bytes. Esto implica que el primer registro se encontrará en la posición 0;el segundo registro se encontrará en la posición 230: el tercero estará en la posición 460 y así sucesivamente.

•Si en cambio codificamos los caracteres en Unicode (UTF-16) de forma que su longitud es de 16 bits por carácter, esto es 2 bytes, el registro quedará así:

–Nombre: 80 caracteres UTF-16.
–Dirección: 100 caracteres UTF-16.
–Población: 50 caracteres UTF-16.

•Su longitud será de 460 bytes para un solo registro, ya que cada carácter ocupa 2 bytes.

2.1.2.2 Archivos de acceso aleatorio

- **Características:**
  - Posicionamiento inmediato.
  - Registros de longitud fija
  - Apertura lectura y/o escritura
  - Uso concurrente
  - Borrado de registro mediante ceros o Marcado
  - Problema que deja muchos huecos

2.1.2.3 Archivos indexados

- Usa un índice del archivo para soportar los accesos.
- El índice provee una capacidad de búsqueda para llegar rápidamente a las proximidades de un registro deseado

•Características

–Son básicamente archivos de acceso aleatorio con utilidad (estructura o tabla de índices, para no calcular) para acceder directamente al registro buscado.

–La estructura guarda índices a las posiciones de los registros.

–Árbol de búsqueda de claves

2.2. Cómo se almacenaba la información antes de los SGBD

Antes de aparecer los SGBD, la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos.

- **Cada aplicación tenía:**

  Un conjunto de archivos de datos

  Un conjunto de programas (otros archivos) que gestionaban estos archivos de datos.

- **Cada programa gestionaba sus propios ficheros de datos.**
- **Se usaron mucho en nóminas, control de pedidos y mantenimiento de productos**

2.2. Cómo se almacenaba la información antes de los SGBD

Los problemas aparecen cuando:

- Aumenta el número de usuarios
- Aumentan las necesidades de los programas
- Necesidad de interconectar programas.

2.3 Inconvenientes de un sistema de gestión de archivos

- **Redundancia e inconsistencia de los datos,** se produce porque los archivos son creados por distintos programas y van cambiando a lo largo del tiempo, es decir, pueden tener distintos formatos y los datos pueden estar duplicados en varios sitios. Por ejemplo, el teléfono de un alumno puede aparecer en más de un archivo. La redundancia aumenta los costes de almacenamiento y acceso, y trae consigo la inconsistencia de los datos: las copias de los mismos datos no coinciden por aparecer en varios archivos.
- **Dependencia de los datos física-lógica**, o lo que es lo mismo, la estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación. Cualquier cambio en esa estructura implica al programador identificar, modificar y probar todos los programas que manipulan esos archivos.

2.3 Inconvenientes de un sistema de gestión de archivos

- **Dificultad para tener acceso a los datos y ampliaciones:,** proliferación de programas, es decir, cada vez que se necesite una consulta que no fue prevista en el inicio implica la necesidad de codificar el programa de aplicación necesario o buscarlo manualmente. Es decir, no permiten recuperar los datos necesarios de una forma conveniente y eficiente.
- **Separación y aislamiento de los datos**, es decir, al estar repartidos en varios archivos, y tener diferentes formatos, es difícil escribir nuevos programas que aseguren la manipulación de los datos correctos. Antes se deberían sincronizar todos los archivos para que los datos coincidiesen.

2.3 Inconvenientes de un sistema de gestión de archivos

- **Problemas de atomicidad.** Es crucial asegurar que una vez un fallo ha ocurrido y se ha detectado, los datos se restauran al estado de consistencia anterior al fallo, o ocurre todo o no ocurre nada. Es difícil asegurar esta propiedad en un sistema de archivos.
- **Dificultad para el acceso concurrente**, pues en un sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente. Las actualizaciones concurrentes pueden dar por resultado datos inconsistentes, ya que se puede acceder a los datos por medio de diversos programas de aplicación.
- **Dependencia de la estructura del archivo con el lenguaje de programación**, pues la estructura se define dentro de los programas. Esto implica que los formatos de los archivos sean incompatibles. La incompatibilidad entre archivos generados por distintos lenguajes hace que los datos sean difíciles de procesar.

2.3 Inconvenientes de un sistema de gestión de archivos

- **Problemas en la seguridad de los datos.** Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando.
- **Problemas de integridad de datos,** es decir, los valores almacenados en los archivos deben cumplir con restricciones de consistencia. Por ejemplo, no se puede insertar una nota de un alumno en una asignatura si previamente esa asignatura no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores a una cantidad. Esto implica añadir gran número de líneas de código en los programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos.

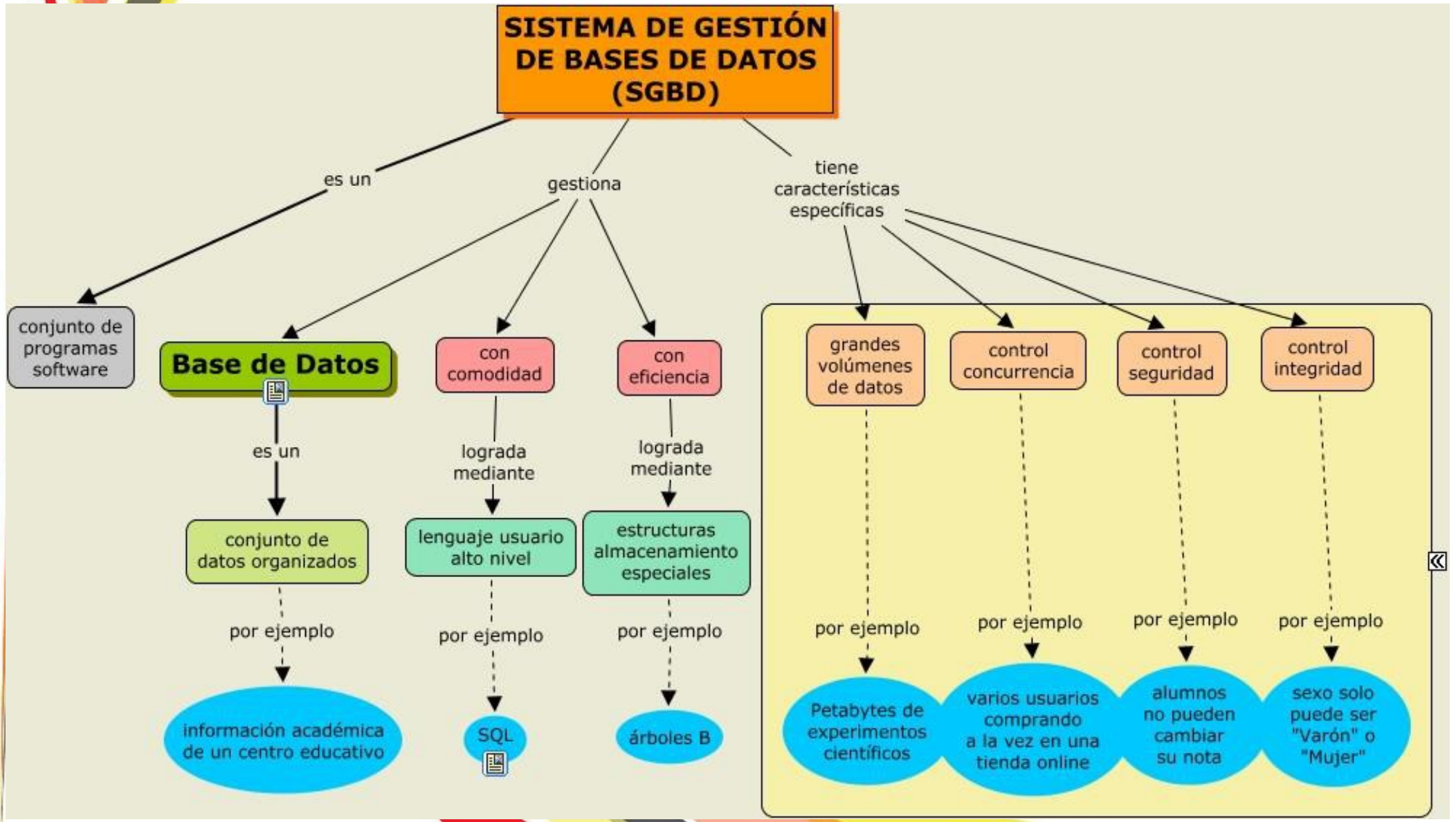
2.3 Inconvenientes de un sistema de gestión de archivos

- Solución ->Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan
- Se trata de estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan.
- Y lo más importante que **TODOS** los datos estén en un único repositorio.
- Todos estos inconvenientes hacen posible el fomento, desarrollo y aparición de los **SGBD**

3.SGBD

- SGBD significa Sistema de Gestión de Bases de Datos. Podemos descomponerlo como Base de Datos + Sistema Gestor.
- La base de datos es una colección de datos y el sistema de gestión es un conjunto de programas para almacenar y recuperar esos datos.
- Basándonos en esto podemos definir SGBD como: SGBD es una colección de datos interrelacionados y un conjunto de programas para almacenar y acceder a esos datos de una manera fácil y efectiva.

3.SGBD. SISTEMA GESTOR DE BASES DE DATOS



3.1 Características de un SGBD

Independencia física y lógica. Estructuras físicas independientes de la lógica y viceversa, acceso a datos sin necesidad de conocer la estructura interna.

- **Eficaz acceso a los datos:** Sin necesidad de conocer como están guardados, facilidad de manipulación (si está autorizado) sin necesidad de ser Informático. (Herramientas gráficas)
- **Mantener la integridad y consistencia** Garantizar que la transacción se haga o se rechace (Atomicidad)
- **Permiten la concurrencia.** Acceso compartido a la BD, controlando la interacción entre usuarios concurrentes. Que no se enteren que ambos están a la vez.
- **Mecanismos de respaldo y recuperación** para restablecer la información en caso de fallos en el sistema
- **Coherencia de datos: Cumplir restricciones.** Reglas de la realidad
- **Redundancia controlada**
- **Administración centralizada. Herramientas de administración.**
- **Seguridad de los datos:** Acceso controlado a los datos de la BD mediante mecanismos de seguridad de acceso a los usuarios.
- **Uso de transacciones**

3.2Ventajas de SGBD sobre el sistema de archivos

Hay varias ventajas de sistema de gestión de base de datos sobre el sistema de archivos. Algunos de ellos son :

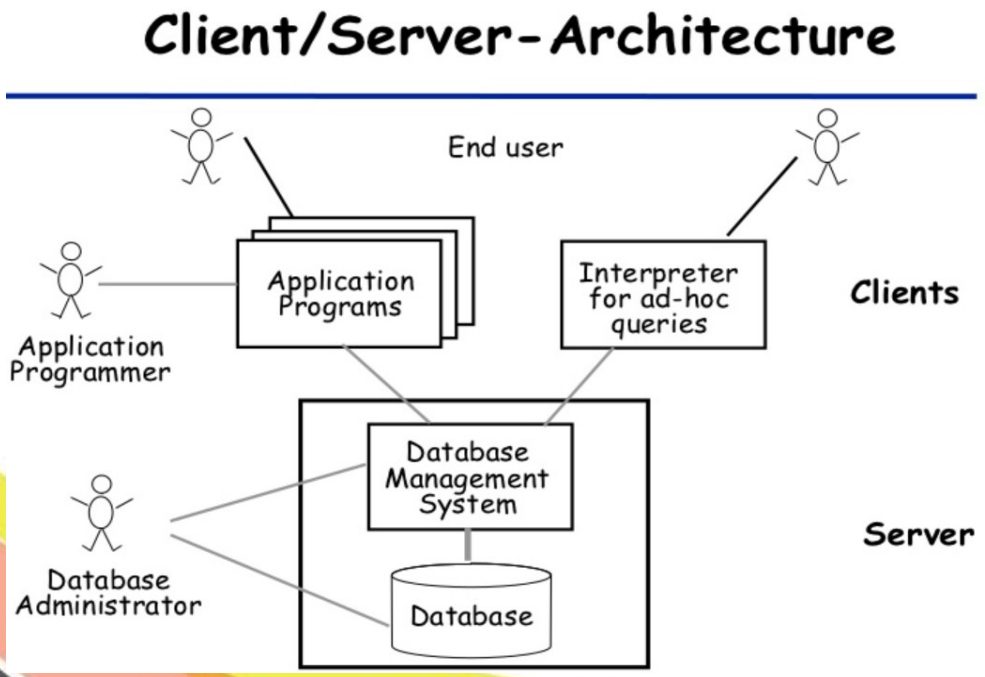
- Sin datos redundantes -Redundancia eliminada por normalización de datos
- Coherencia e integridad de los datos: la normalización de los datos también se ocupa de ella
- Seguridad -Cada usuario tiene un conjunto diferente de acceso
- Privacidad-Acceso limitado
- Fácil acceso a los datos
- Fácil recuperación
- Flexible

Desventajas de DBMS:

- El **coste** de implementación de SGBD es alto en comparación con el sistema de archivos
- **Complejidad**: Los sistemas de base de datos son complejos de entender
- **Rendimiento**: Los sistemas de bases de datos son genéricos, haciéndolos adecuados para diversas aplicaciones. Sin embargo esta característica afecta su rendimiento para algunas aplicaciones.

BBDD y SGBD

- Usuarios finales
- Aplicaciones
- Programador de aplicaciones
- SGBD
- Administrador de base de datos de base de datos (DBA)



3.3 Concepto de Transacción

Una transacción es un conjunto de instrucciones que realizan una acción y tiene las características ACID (**A**tomicity, **C**onsistency, **I**solation and **D**urability) :

–**Atomicidad:** es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

–**Consistencia:** *Integridad*. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.

–**Aislamiento:** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.

–**Durabilidad:** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. .

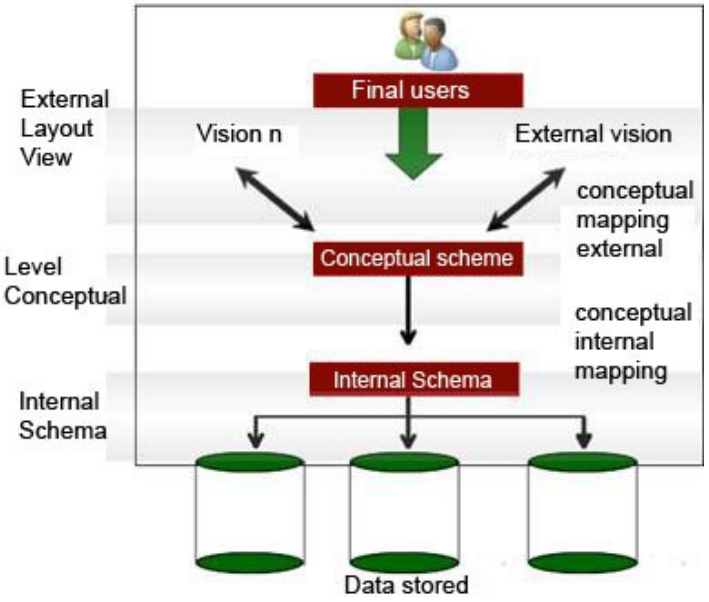
3.4 Arquitectura SGBD

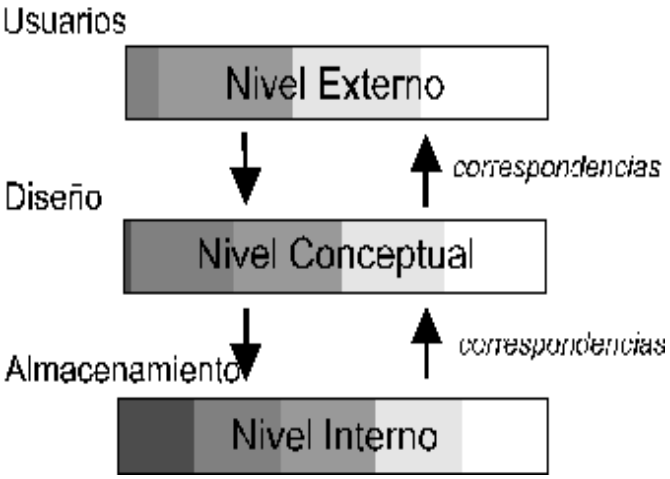
Laa arquitectura SGBD tiene como objetivo principal que las aplicaciones de usuario estén separadas de los datos físicos y estos se dividen a partir de los siguientes diagramas:

Nivel interno: utiliza un modelo de datos que muestra la estructura de almacenamiento físico de la base de datos, los detalles de los datos guardados y las rutas de acceso. Es el nivel más cercano al **almacenamiento físico**, es decir, tal y cómo están almacenados los datos en el ordenador (archivos, tipos de registro, longitud..)

Nivel conceptual o lógico: realiza una descripción completa de la estructura de la base de datos pero no ofrece detalles de los datos almacenados en la base de datos. Define qué datos hay almacenados y cómo se relacionan (**entidades, relaciones, atributos...**)

Nivel externo: describe las vistas de la base de datos a un grupo de usuarios y muestra qué usuarios tienen acceso a esta base de datos. Nivel más cercano a los **usuarios.** Vistas de partes de la BD, bien para restringir o para simplificar.





Ejemplo con Modelo Relacional

Nivel externo: Visión parcial de las tablas de la BD según el usuario. Por ejemplo, la vista que se muestra en la Tabla, obtiene el listado de notas de alumnos con los siguientes datos: Curso, Nombre, Nombre de asignatura y Nota.

Nivel lógico y conceptual: Definición de todas las tablas, columnas, restricciones, claves y relaciones. En este ejemplo, disponemos de las tablas que están relacionadas, estudiantes y actividades, cada una con su clave que es ID

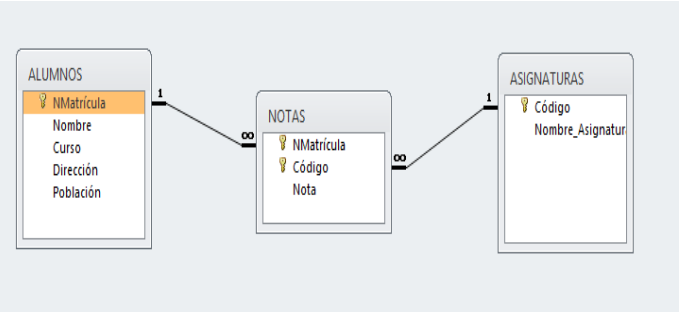
Nivel Interno: En una BD las tablas se almacenan en archivos de datos de la BD. Si hay claves, se crean índices para acceder a los datos, todo esto contenido en el disco duro, en una pista y en un sector, que sólo el SGBD conoce. Ante una petición, sabe a qué pista, a qué sector, a qué archivo de datos y a qué índices acceder

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity1	Cost1	Activity2	Cost2
084	Tennis	$36	Swimming	$17
100	Squash	$40	Swimming	$17
182	Tennis	$36		
219	Swimming	$15	Golf	$47



Ventajas de la arquitectura

Independencia lógica: la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se podrá modificar el esquema conceptual para ampliar la BD o para reducirla, por ejemplo, si se elimina una entidad, los esquemas externos que no se refieran a ella no se verán afectados.

•**Independencia física**: la capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual, ni los externos. Por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización, o se pueden añadir nuevos archivos de datos porque los que había se han llenado. La independencia física es más fácil de conseguir que la lógica, pues se refiere a la separación entre las aplicaciones y las estructuras físicas de almacenamiento.

•Las correspondencias entre niveles requiere ampliar el diccionario de datos

3.5 Funciones SGBD

La función principal: Permitir a los usuarios las 4 operaciones fundamentales:

–Creación e inserción

–Consulta

–Actualización

–Borrado

Con 2 objetivos:

1.Visión abstracta de los datos (esconder como se almacenan y mantienen los datos)

2.Velocidad (buen tiempo de respuesta) y seguridad (que sea verdad lo que resulta).

Función de Definición

- Lenguaje SQL de definición
- Nivel Interno:
  - Espacio físico
  - Longitud de campos
  - Modo de representación
  - Caminos de acceso, punteros, índices
- Nivel Conceptual
  - Definición de entidades, atributos y relaciones
  - Autorización de accesos
- Nivel Externo
  - Construcciones gráficas de tablas y relaciones (Acces)
- Todo se almacena en el Diccionario de datos

Función Manipulación

Permite buscar, añadir, suprimir o modificar los datos de una BD siempre de acuerdo a las especificaciones y restricciones de seguridad.

- Utiliza el lenguaje DML
- Nivel Interno: Algoritmos eficientes de acceso a los datos.
- Nivel Conceptual: Abstracción de la parte interna, construcción de consultas o actualizaciones.. Vistas a programadores
- Nivel externo: Vistas a usuarios. Facilidad de uso con herramientas gráficas (Access)

3.6 Componentes de un SGBD

- Componentes humanos. USUARIOS.
- Componentes técnicos

  Componentes Funcionales

  - Componentes de procesamiento de consultas
  - Componentes de Gestión de almacenamiento

  Implementación Física del sistema

  - Archivos de datos
  - Diccionario de datos

3.6.1 Usuarios

- **Administradores: diseño físico**

Un administrador de bases de datos, un analista de bases de datos o un desarrollador de bases de datos es la persona responsable de gestionar la información dentro de una organización.

La función principal del Administrador de la base de datos es administrar, desarrollar, mantener e implementar las políticas y procedimientos necesarios para garantizar la seguridad e integridad de la base de datos corporativa. Las funciones secundarias dentro de la clasificación de Administrador de bases de datos pueden incluir seguridad, arquitectura, almacenamiento y / o análisis de negocio. Otras funciones principales incluyen:

- Implementación de modelos de datos
- Diseño de base de datos
- Accesibilidad a la base
- Problemas de desempeño
- Problemas de capacidad
- Replicación de datos
- Mantenimiento de la tabla

- **Diseñadores**: diseño lógico.
- **Programadores**. Implementan los programas
- **Usuarios finales**. Utilizan el resultado final.

3.6.2 Consultas y almacenamiento

- **Procesador de consultas:** el procesador de consultas se transforma en una serie de instrucciones de bajo nivel. Se utiliza para interpretar la consulta del usuario en línea y convertirla en una serie eficaz de operaciones en una forma capaz de ser enviada al gestor de datos de tiempo de ejecución para su ejecución. El procesador de consultas utiliza el diccionario de datos para encontrar la estructura de la parte relevante de la base de datos y utiliza esta información para modificar la consulta y preparar el plan óptimo para acceder a la base de datos.
- **Optimizador de consultas:** Los optimizadores de consultas determinan una estrategia óptima para la ejecución de la consulta

Componentes del gestor de almacenamiento

- **Gestor de datos:** El gestor de datos es responsable de los datos de la base de datos. Proporciona un sistema capaz de recuperación al sistema después de algún fallo. Incluye administrador de recuperación y gestor de búfer. El gestor de búfer es responsable de la transferencia de datos entre la memoria principal y el almacenamiento secundario (como el disco) También se refiere como el gestor de caché.
- **Comprobador de integridad:** comprueba las restricciones de integridad para que sólo se puedan introducir datos válidos en la base de datos.
- **Gestor de transacciones:** El gestor de transacciones asegura que las propiedades de la transacción deben ser coherentes en el sistema
- **Planificador:** Proporciona un entorno en el que múltiples usuarios pueden trabajar con la misma información al mismo tiempo, es compatible con la simultaneidad

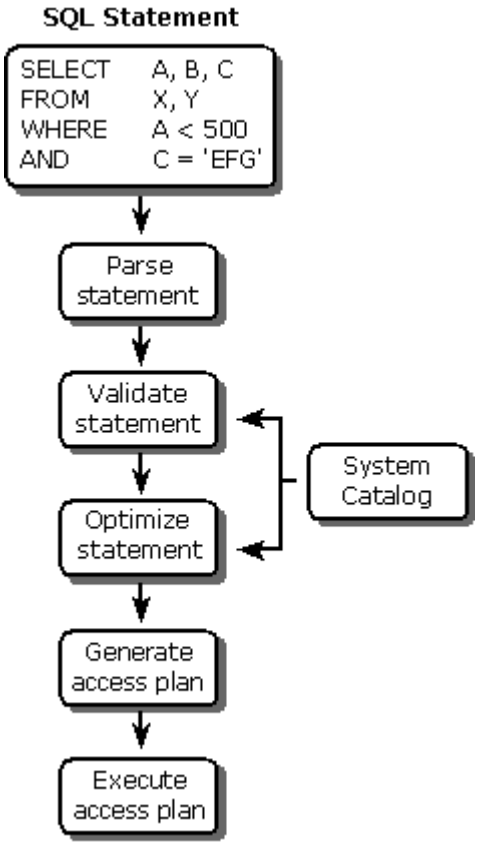
3.6.3 Implementación física del sistema

- Archivos de datos: Almacena la base de datos en sí.
- Diccionario de datos: Almacena metadatos acerca de la estructura de los mismos.
- Índices: Proporciona acceso rápido a los datos.
- Datos estadísticos: Almacena datos estadísticos sobre los datos. El procesador de consultas usa esta información para planificarlas.

3.6.4 Ejecución de procesos en un SGBD

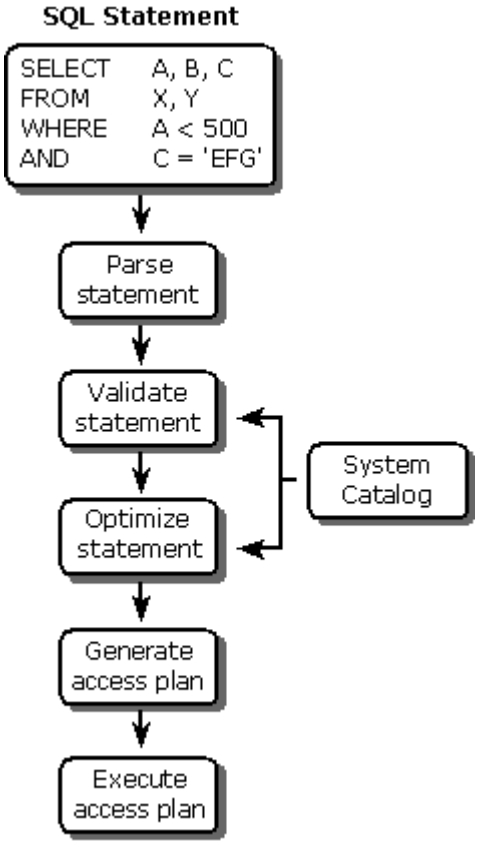
Para procesar una sentencia SQL, un SGBD realiza los siguientes cinco pasos: +

- **El SGBD primero analiza la instrucción SQL.** Se rompe la declaración en palabras individuales, llamadas tokens, se asegura de que la declaración tiene un verbo válido y cláusulas válidas, y así sucesivamente. En este paso se pueden detectar errores de sintaxis y errores ortográficos.
- **El SGBD valida la sentencia.** Comprueba la instrucción en el catálogo del sistema. ¿Existen todas las tablas de la sentencia en la base de datos? ¿Existen todas las columnas y los nombres de columna no son ambiguos? ¿Tiene el usuario los privilegios necesarios para ejecutar la sentencia? En este paso se pueden detectar ciertos errores semánticos.
- **El SGBD genera un plan de acceso para la sentencia.** El plan de acceso es una representación binaria de los pasos que se requieren para llevar a cabo la declaración, es el equivalente DBMS del código ejecutable.
- **El SGBD optimiza el plan de acceso**. Explora varias maneras de llevar a cabo el plan de acceso. ¿Se puede usar un índice para acelerar la búsqueda? ¿Debería el SGBD aplicar primero una condición de búsqueda a la Tabla A y luego unirla a la Tabla B, o debería comenzar con la unión y usar la condición de búsqueda después? ¿Se puede evitar una búsqueda secuencial a través de una tabla o reducirla a un subconjunto de la tabla? Después de explorar las alternativas, el SGBD elige uno de ellos.



Ejecución de procesos en un SGBD

- **El SGBD ejecuta la instrucción ejecutando el plan de acceso.**
- Los pasos utilizados para procesar una sentencia SQL varían en la cantidad de acceso a la base de datos que requieren y la cantidad de tiempo que toman. El análisis de una sentencia SQL no requiere acceso a la base de datos y puede realizarse muy rápidamente.
- La optimización, por otro lado, es un proceso que requiere mucha CPU y requiere acceso al catálogo del sistema. Para una consulta múltiple compleja, el optimizador puede explorar miles de maneras diferentes de llevar a cabo la misma consulta. Sin embargo, el costo de ejecutar la consulta de manera ineficiente suele ser tan alto que el tiempo invertido en la optimización es más que recuperado en la mayor velocidad de ejecución de la consulta. Esto es aún más significativo si el mismo plan de acceso optimizado se puede utilizar una y otra vez para realizar consultas repetitivas.



4. Tipos de SGBD

- Según el modelo:
  - Jerárquico
  - Red
  - Relacional
  - Orientado a Objetos. UML y XML
- Según arquitectura
  - Centralizadas.
  - Cliente/Servidor
  - Distribuidas. Homogéneos y heterogéneos.

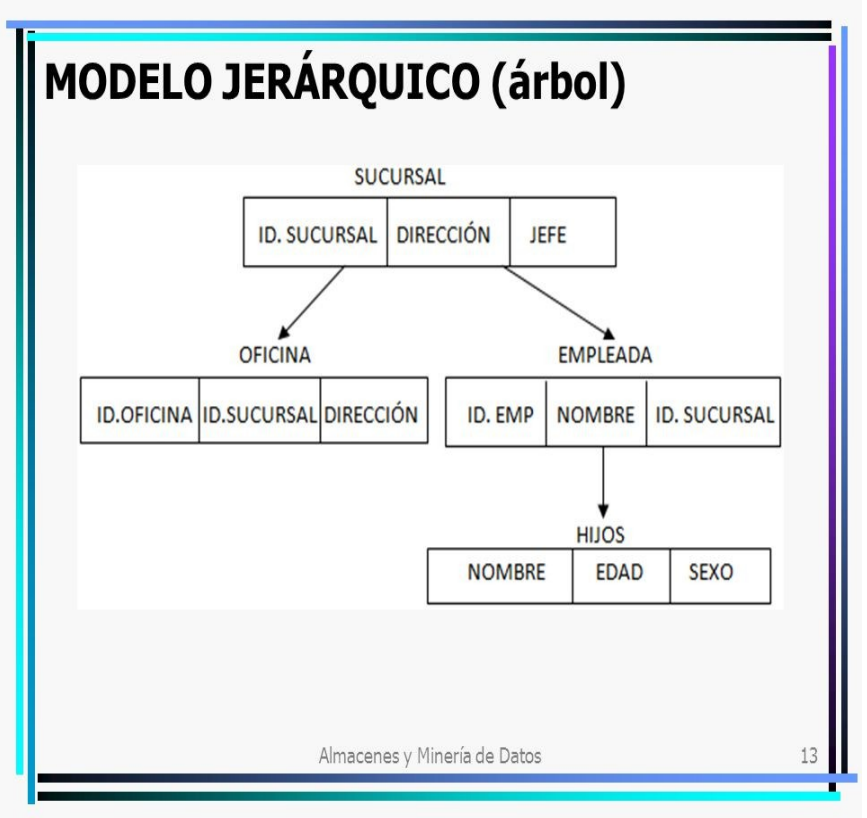
Modelo Jerárquico

Un modelo de base de datos jerárquico es un modelo de datos en el que los datos se organizan en una estructura similar a un árbol.

Los datos se almacenan como registros que están conectados entre sí a través de enlaces. Un registro es una colección de campos, con cada campo que contiene sólo un valor. El tipo de entidad de un registro define los campos que contiene el registro.

Un registro en el modelo de base de datos jerárquico corresponde a una fila (o tupla) en el modelo de base de datos relacional y un tipo de entidad corresponde a una tabla (o relación).

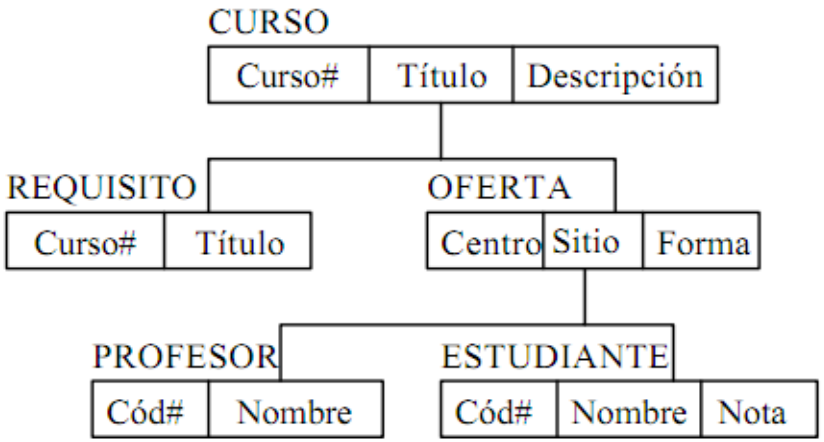
No soporta relaciones muchos-a-muchos.



Modelo en red

Parecido al jerárquico, conjunto de registros enlazados mediante puntero, pero esta vez en forma de grafo.

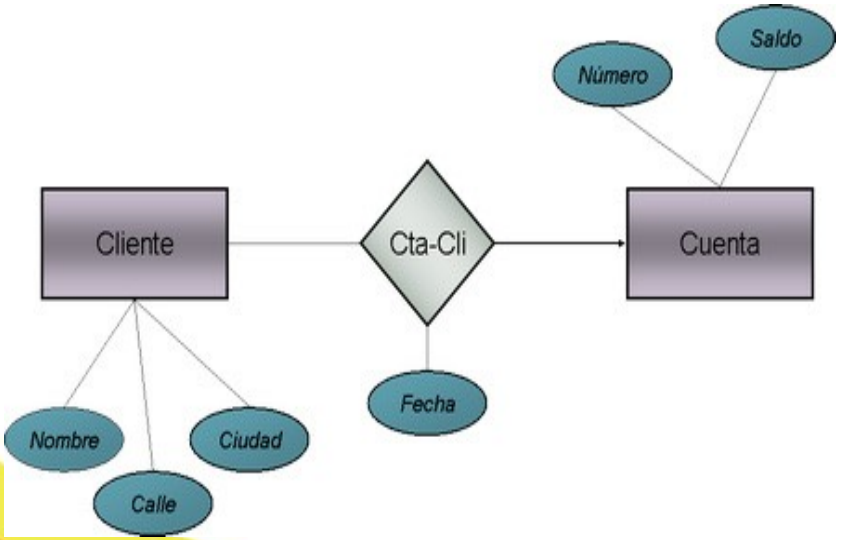
•Al menos teóricamente sí disponía de comunicación bidereccional, pero en la práctica era demasiado costoso y lo hacía con varios punteros. Simulaba la M:M con varías 1:M



Modelo Relacional

El más extendido y comercialmente el único con éxito, los datos se describen como relaciones que se suelen representar como tablas bidimensionales consistentes en filas y columnas.

- Cada fila (tupla, en terminología relacional) representa una ocurrencia.
- Las columnas (atributos) representan propiedades de las filas.
- Cada tupla se identifica por una clave primaria o identificador.
- Los enlaces no son punteros son los mismos datos.



Modelo Orientado a Objetos

- En una base de datos de objetos (OODBMS) la información se representa en forma de objetos tal como se utiliza en la programación orientada a objetos. Las bases de datos de objetos son diferentes de las bases de datos relacionales que están orientadas a tablas. Las bases de datos objeto-relacionales son un híbrido de ambos enfoques.
- La mayoría de las bases de datos de objetos también ofrecen algún tipo de lenguaje de consulta, permitiendo que los objetos se encuentren usando un enfoque de programación declarativa. Es en el área de los lenguajes de consulta de objetos, y la integración de la consulta y las interfaces de navegación, donde se encuentran las mayores diferencias entre los productos. Un intento de normalización fue hecha por el ODMG con el lenguaje de consulta de objetos, OQL.
- El acceso a los datos puede ser más rápido porque un objeto puede ser recuperado directamente sin una búsqueda, siguiendo los punteros.
- Otra gran variación entre los modelos es la forma en que se define el esquema de una base de datos. Además, una característica general es que el lenguaje de programación y el esquema de la base de datos utilizan las mismas definiciones de tipo.
- Las aplicaciones multimedia se simplifican porque los métodos de clase asociados con los datos son responsables de su interpretación
- La eficiencia de la base de datos también ha mejorado mucho en áreas que demandan cantidades masivas de datos. Por ejemplo, un banco podría obtener la información de la cuenta del usuario y proporcionarles eficientemente mucha información, como transacciones, información de la cuenta, etc.

BBDD Centralizada

- La BBDD está localizada, guardada y mantenida en una sola localización
- Ventajas:
  - Tener una visión completa de los datos
  - Manejar la BBDD es sencillo
- Desventajas:
  - Necesitas estar en el ordenador donde se encuentre la BBDD

Cliente / Servidor

- Esta arquitectura es la más usada hoy en día
- Esta BBDD usa el modelo cliente servidor. Los clientes hacen peticiones a la BBDD usando una red
- **Ventajas**:
  - Tienen una visión completa de los datos
  - Manejar la BBDD es más fácil
  - Conectar desde distintos sitios
- **Desventajas**:
  - Cuellos de botella por los accesos concurrentes de distintos usuarios al mismo archivo.

Base de Datos Distribuida

Toda la información se almacena en múltiples ubicaciones físicas.

Ventajas:

- Los usuarios no interferirán entre sí al acceder / manipular datos
- Aumenta la velocidad desde que los archivos se recuperan desde la ubicación más cercana
- Si falla un sitio, el sistema puede recuperarse

Desventajas

- Tiempo de sincronización de las múltiples bases de datos
- Replicación de datos para cada archivo de base de datos diferente

Open Source y BBDD Comerciales

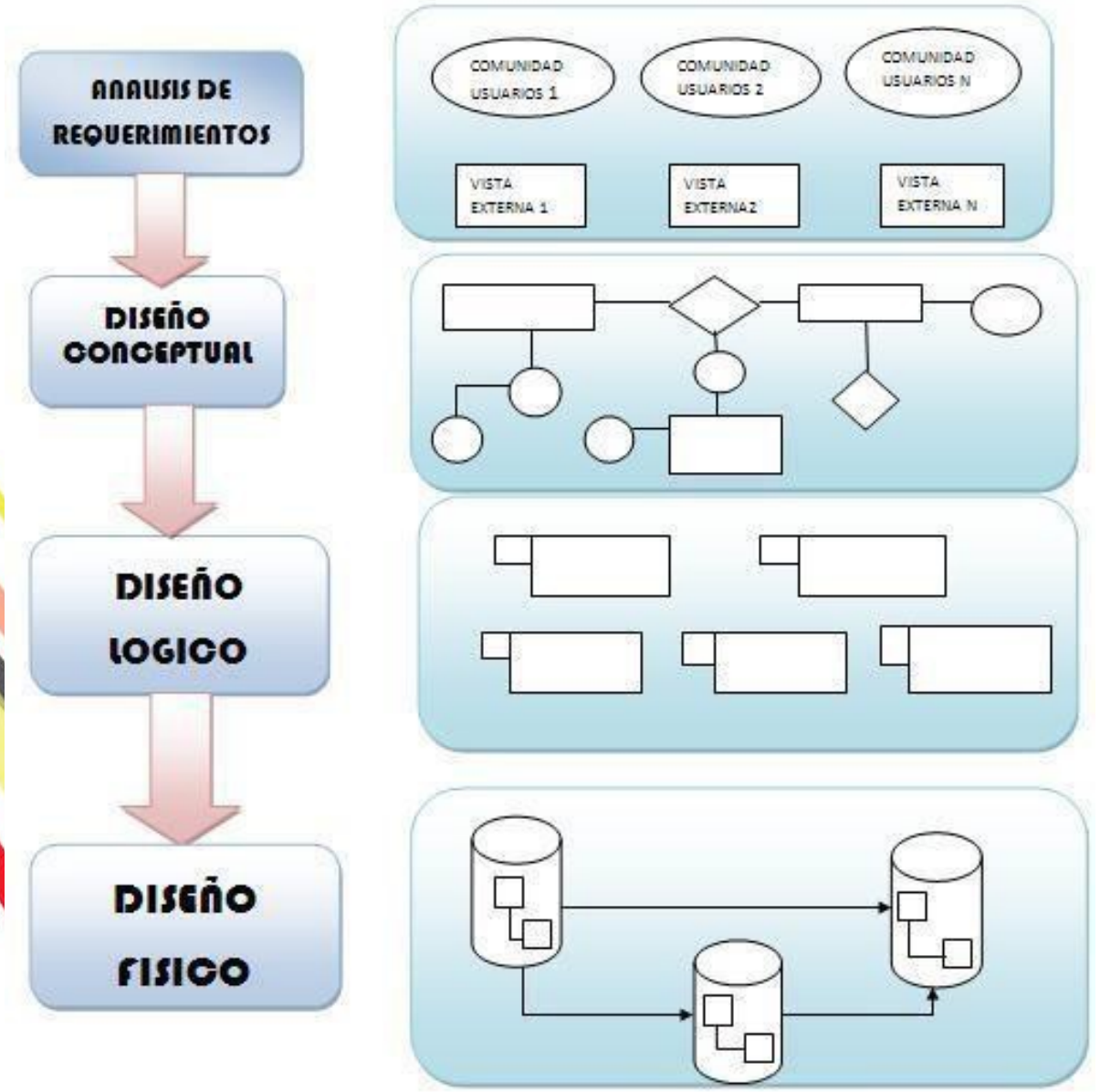
NoSQL

- La Base de datos proporciona un mecanismo para el almacenamiento y recuperación de datos diferentes a las relaciones utilizadas en las bases de datos relacionales.
- Han existido desde finales de los años 1960, pero la popularidad creció debido a las necesidades de la Web 2.0 y empresas como Facebook, Google y Amazon
- Las bases de datos NoSQL se utilizan en grandes aplicaciones de datos y en tiempo real.
- Muchos DB NoSQL comprometen la consistencia en favor de la disponibilidad, tolerancia de partición y velocidad.
- Basado en las clasificaciones de popularidad de 2015, las bases de datos NoSQL más populares son **MongoDB, Apache Cassandra y Redis.**

5. Fases del Diseño en una Base de Datos

- Los requisitos y la fase de análisis de la recolección producen tanto requisitos de datos como requisitos funcionales. Los requisitos de datos se utilizan como una fuente de diseño de base de datos. Los requisitos de datos deben especificarse en la forma más detallada y completa posible.
- Paralelamente a la especificación de los requisitos de datos, es útil especificar los requisitos funcionales conocidos de la aplicación. Los requisitos funcionales se utilizan como una fuente de diseño de software de aplicación.
- Tened en cuenta que algunas fases son independientes del sistema de gestión de bases de datos y algunas son dependientes. Ejemplo: SQL será diferente si elegimos MySQL u Oracle

Fases del diseño en una Base de Datos



Diseño Conceptual

- Una vez reunidos y analizados todos los requisitos, el siguiente paso es crear un esquema conceptual para la base de datos, utilizando un modelo de datos conceptuales de alto nivel. Esta fase se llama diseño conceptual.
- El resultado de esta fase es un d**iagrama de Entidad-Relación (ER)** o diagrama de clases UML.
- Es un modelo de datos de alto nivel del área de aplicación específica. Describe cómo diferentes entidades (objetos, elementos) están relacionadas entre sí. También describe qué atributos (características) tiene cada entidad. Incluye las definiciones de todos los conceptos (entidades, atributos) del área de aplicación.
- Hay varias anotaciones para dibujar el diagrama ER.

Diseño Logico

- El resultado de la fase de diseño lógico es un conjunto de esquemas de relación. **El diagrama ER** o diagrama de clases es la base de estos esquemas de relación.
- Crear los diagramas de relación es una operación mecánica. Hay reglas de cómo el modelo ER o diagrama de clases se transfiere a diagramas de relación.
- Los diagramas de relación son la base para las definiciones de tablas.
- En esta fase (si no se realiza en la fase anterior) se definen las claves primarias y las claves externas.

Normalización

La normalización es la última parte del diseño lógico. El objetivo de la normalización es eliminar la redundancia y las posibles anomalías de actualización.

Redundancia significa que los mismos datos se guardan más de una vez en una base de datos. Actualizar la anomalía es una consecuencia de la redundancia. Si una parte de datos se guarda en más de un lugar, los mismos datos deben actualizarse en más de un lugar.

La **normalización** es una técnica mediante la cual se puede modificar el diagrama E-R para reducir la redundancia. Cada fase de normalización añade más relaciones (tablas) a la base de datos.

Diseño Físico

El objetivo de la última fase del diseño de la base de datos, el diseño físico, es implementar la base de datos.

En esta fase se debe saber qué sistema de gestión de base de datos (SGBD) se utiliza. Por ejemplo, diferentes SGBD tienen diferentes nombres para mimos tipos de datos y tienen diferentes tipos de datos.

Se escriben las cláusulas SQL para crear la base de datos . Se definen los índices, las restricciones de integridad (reglas) y los derechos de acceso de los usuarios.

Conceptos del SGBD Relacional

SGBDR = Sistema de gestión de base de datos relacional

Tabla / Relación: Una tabla es una colección de datos representados en filas y columnas.

Registros / Tupla / Fila: Cada fila de una tabla se conoce como registro o también se conoce como tupla

Nombre de campo / columna: La tabla de abajo tiene tres campos: Rollno, Nombre y Teléfono.

Columna / Atributo: Cada atributo y sus valores se conocen como atributos en una base de datos. Por ejemplo, El conjunto de valores del campo Nombre es una de las tres columnas de la tabla Estudiantes.

Conceptos del SGBD Relacional

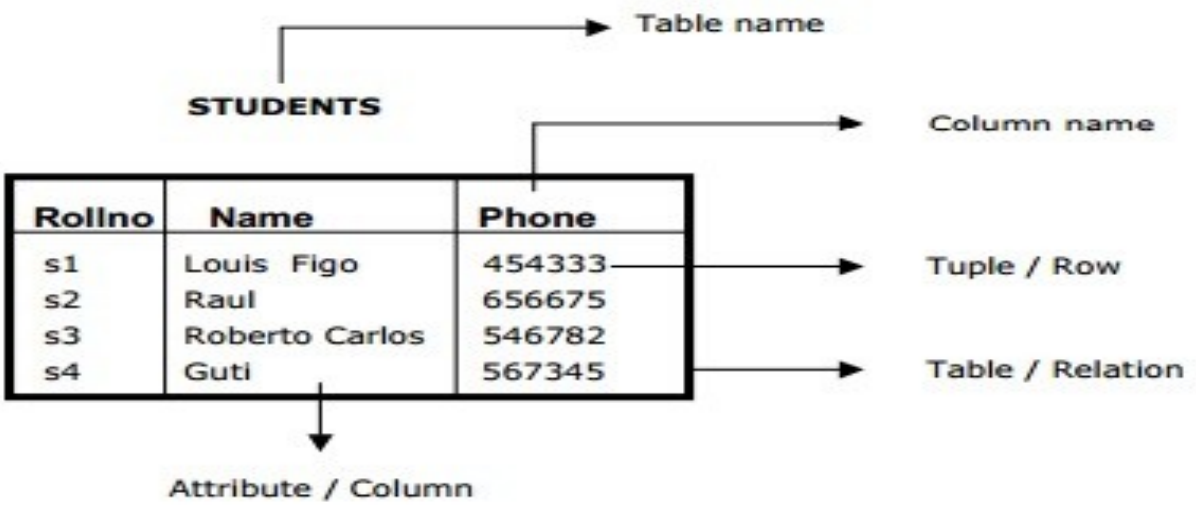


Figure 1: A table in relational model.

Lenguajes de SGBD

Los lenguajes de base de datos se utilizan para leer, actualizar y almacenar datos en una base de datos. Hay varios lenguajes que pueden ser usados para este propósito; uno de ellos es SQL (Structured Query Language).

SQL es un lenguaje estándar para acceder y manipular bases de datos.

Lenguaje de Definition Datos (DDL)

Lenguaje de definición de datos (DDL): DDL se utiliza para especificar el esquema de la base de datos. Tomemos SQL por ejemplo para categorizar las declaraciones que vienen bajo DDL.

- Para crear la instancia de base de datos: CREATE
- Para alterar la estructura de la base de datos - ALTER
- Para eliminar instancias de base de datos - DROP
- Para eliminar tablas en una instancia de base de datos - TRUNCATE
- Para cambiar el nombre de las instancias de base de datos - RENAME

Lenguaje de Manipulación de Datos (DML)

- **Lenguaje de manipulación de datos (DML):** DML se utiliza para acceder y manipular datos en una base de datos.
- Para leer registros de la (s) tabla (s) - SELECT
- Para insertar registros en la (s) tabla (s) - INSERT
- Para actualizar los datos en la (s) tabla (s) - UPDATE
- Para borrar todos los registros de la tabla - DELETE

Lenguaje de Control de Datos (DCL)

Lenguaje de control de datos (DCL): DCL se utiliza para conceder y revocar el acceso de usuario en una base de datos.

- Para conceder acceso al usuario - GRANT
- Para revocar el acceso del usuario - REVOKE