

# **BASES DE DADES**

## **Tema 1**

### **Sistemes d'Emmagatzematge..**

Javier Ivorra

# 1.Introducció.

## 1.1 Història i Evolució Bases de dades

## 2 Sistema basat en arxius

### 2.1.Tipus d'arxius.

2.1.1.Segons el seu  
contingut 2.1.2.Segons  
el seu accés:

2.1.2.1 Arxius seqüencials

2.1.2.2 Arxius aleatoris

2.1.2.3 Arxius indexats

### 2.2. Com s'emmagatzemava la informació abans dels SGBD

### 2.3.Inconvenients d'un sistema de gestió d'arxius.

## 3. Sistemes Gestors de Bases de dades (SGBD)

### 3.1.Característiques d'un SGBD

### 3.2 Avantatges SGBD sobre sistema d'arxius.

### 3.3 Concepte de transacció

### 3.4.Arquitectura SGBD

### 3.5.Funciones SGBD

### 3.6.Componentes d'un SGBD

#### 3.6.1 Usuaris

#### 3.6.2 Consultes i emmagatzematge

#### 3.6.3 Implementació Física del sistema

#### 3.6.4 Execució de processos en un SGBD

## 4. Tipus de SGBD.

## 5. Fases del disseny d'una BD.

# INTRODUCCIÓ

Les Bases de dades es desenvolupen quan el volum de dades alt i temps de resposta que necessitem és vital. Hi ha dues coses que requereixen una optimització: l'emmagatzematge de dades i la consulta de dades.

L'ús d'aquests sistemes incrementa l'eficiència de les operacions en l'empresa i redueix els costos totals.

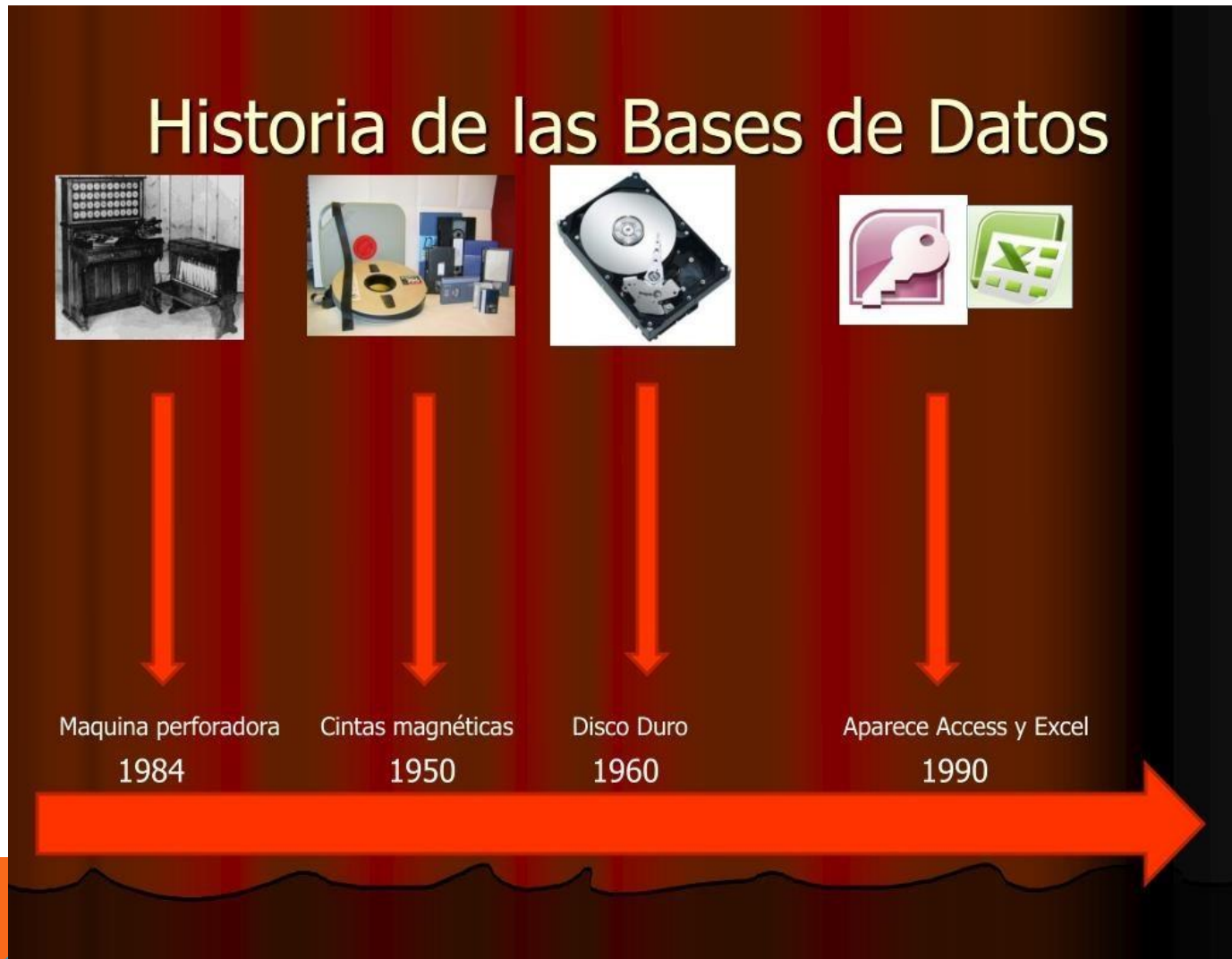
Hui dia pràcticament totes les empreses usen bases de dades. Quan compres en un supermercat diverses bases de dades són presents:

La base de dades del magatzem canviarà la quantitat de productes. El codi de barres serà consultat.

Si pagues amb targeta de crèdit, aquesta actualitzarà la quantitat en el teu compte corrent.



# 1.1 HISTÒRIA DE LES BASES DE DADES



# 1.1 HISTÒRIA DE LES BASES DE DADES



En 1884 Herman Hollerit va crear la màquina perforadora, amb la qual va aconseguir baixar de 7 anys a 2 anys el temps a censar a la població d'USA. Composta per un perforadora i una lectora que llegia orificis.



# 1.1 HISTÒRIA DE LES BASES DE DADES

- Dècada de 1950. Es va inventar la cinta magnètica. Es va començar a automatitzar la informació de les nòmines, comencen a usar-se bases de dades basades en sistemes d'arxius seqüencials. Aquestes cintes només es podien llegir seqüencial i ordenadament.



# 1.1 HISTÒRIA DE LES BASES DE DADES

Dècada de 1960. L'ús dels discos en aqueix moment va ser un avançament molt efectiu, ja que, per mitjà de aquest suport es podia consultar la informació directament usant fitxers aleatoris i indexats, on l'accés ja no té per què ser seqüencial. El sistema de fitxers era el sistema més comú d'emmagatzematge de dades, la qual cosa va ajudar a estalviar temps. No era necessari saber exactament on estaven les dades en els discos.



Disc dur actual

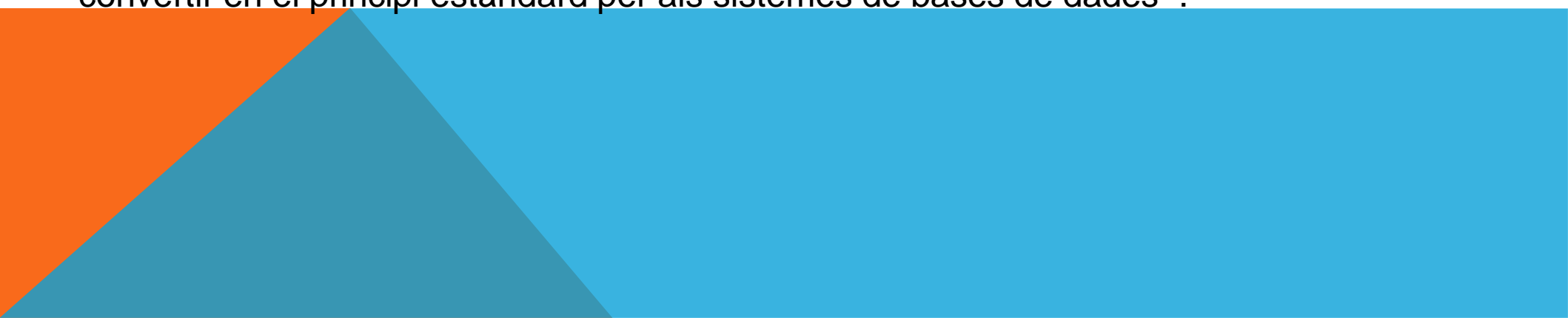
Primer disc dur d'IBM 1961. Pesava una tona i emmagatzemava 5 MB.

# 1.1 HISTÒRIA DE LES BASES DE DADES

**Antiguitat:** Els éssers humans van començar a emmagatzemar informació fa molt temps. En l'antiguitat, els elaborats sistemes de bases de dades van ser desenvolupats per oficines governamentals, biblioteques, hospitals i organitzacions empresarials, i alguns dels principis bàsics d'aquests sistemes encara s'utilitzen hui dia .

**Els anys 60:** La base de dades per ordinador va començar en els anys 60, quan l'ús d'ordinadors es va convertir en una opció més rendible per a les organitzacions privades. Va haver-hi dos models de dades populars en aquesta dècada: un model de xarxa anomenat **CODASYL** i un model **jeràrquic** anomenat **IMS**. Un sistema de base de dades que va resultar ser un èxit comercial va ser el sistema **SABRE** que va ser utilitzat per IBM per a ajudar a American Airlines a gestionar les seues dades de reserves..

**1970 a 1972:** **E.F. Codd** va publicar un important article per a proposar l'ús d'un model de base de dades relacional, i les seues idees van canviar la forma en què la gent pensava sobre les bases de dades . En el seu model, l'esquema de la base de dades, o organització lògica, es desvincula de l'emmagatzematge d'informació física, i això es va convertir en el principi estàndard per als sistemes de bases de dades .





# 1.1 HISTÒRIA DE LES BASES DE DADES

**Dècada de 1970:** Es van crear dos prototips principals de sistemes de bases de dades relacionals entre els anys 1974 i 1977, i van ser Ingres i System R. Ingres va utilitzar un llenguatge de consulta conegut com QUEL, que va portar a la creació de sistemes com Ingres Corp., **MS SQL Server**, Sybase, PASTURA. D'altra banda, System R va utilitzar el llenguatge de consulta SEQUEL, i va contribuir al desenvolupament de SQL / **DS**, DB2, Allbase, **Oracle** i Senar-Stop SQL. Va ser també en aquesta dècada que **Relational Database Management System**, o **RDBMS**, es va convertir en un terme reconegut.

**1976:** Un nou model de base de dades denominat **Entitat-Relació**, o **ER**, va ser proposat per P. **Chen** enguany. Aquest model va fer possible que els dissenyadors se centraren en l'aplicació de dades, en lloc de l'estructura lògica de la taula.

**1980s: Structured Query Language**, o **SQL**, es va convertir en el llenguatge de consulta estàndard. Els sistemes de bases de dades relacionals es van convertir en un èxit comercial, ja que el ràpid augment de les vendes de computadores va impulsar el mercat de les bases de dades, la qual cosa va provocar una important disminució en la popularitat de la xarxa i els models de bases de dades jeràrquics. DB2 es va convertir en el principal producte de base de dades per a IBM, i la introducció d'IBM **PC** va fer que sorgiren moltes noves companyies de bases de dades i el desenvolupament de productes com PARADOX, RBASE 5000, RIM, Dbase III i IV, US / i Watcom SQL.

# 1.1 HISTÒRIA DE LES BASES DE DADES

**Principis de 1990:** Després d'una indústria de bases de dades sacsejada, la majoria de les empreses supervivents van vendre productes de base de dades complexos a preus alts. Durant aquest temps, es van llançar noves eines de client per al desenvolupament de aplicacions, que van incloure **Oracle Developer**, **PowerBuilder**, **VB** i altres. També es van desenvolupar diverses eines per a la productivitat personal, com **ODBC** i **Excel / Access**. Els prototips per als sistemes de gestió de bases de dades d'objectes, o **ODBMS**, es van crear a principis de els anys noranta.

**Mediats dels 90:** L'arribada d'Internet va portar al creixement exponencial de la indústria de la base de dades. Els usuaris d'escriptori mitjà van començar a usar sistemes de base de dades client-servidor per a accedir a sistemes informàtics que contenien dades heretades.



# 1.1 HISTÒRIA DE LES BASES DE DADES

**A la fi de la dècada de 1990:** L'augment de la inversió en negocis en línia va provocar un augment en la demanda de connectors de bases de dades d'Internet, com ara **Front Page, Pàgines Active Server, Servidors Java, DreamWeaver, ColdFusion, Enterprise Java Beans i Oracle Developer 2000**. L'ús de cgi, **gcc, MySQL, Apatxe** i altres sistemes van portar la solució de codi obert a Internet. Amb l'ús creixent de la tecnologia de punt de venda, el processament de transaccions en línia i el processament analític en línia van començar a millorar..

**2000:** Encara que la indústria d'Internet va experimentar prompte una disminució en aquesta dècada, les aplicacions de bases de dades continuen creixent. Es van desenvolupar noves aplicacions interactives per a PDA, transaccions de punt de venda i consolidació de proveïdors. Actualment, les tres principals companyies de bases de dades en el món occidental són **Microsoft, IBM i Oracle**.



## 2- SISTEMA BASAT EN ARXIUS

Com hem dit tota base de dades és un conjunt de dades estructurades i emmagatzemades en un mitjà.

En l'ordinador aquestes dades al final estan emmagatzemats en fitxers.

Abans d'aparèixer els Sistemes Gestors, los primers Sistemes de Bases de dades van usar sistemes de fitxers dependents del S.O.



# 2.1.TIPUS D'ARXIUS.

Què és un arxiu?

- Són estructures d'informació que creguen els S.O. per a poder emmagatzemar dades.
- Solen tindre: <nom>.<extensió>

Un mètode popular utilitzat per molts sistemes operatius, Mac OS X, CP / M i DOS, és determinar el format d'un arxiu basat en l'extensió. Per exemple , els documents HTML s'identifiquen per noms que acaben amb .html (o .htm) i les imatges GIF per .gif.

En el sistema d'arxius FAT original (de les versions MS DOS i Windows prèvies) els noms dels arxius estaven limitats a un identificador de 8 caràcters i a una extensió de 3 caràcters.






# 2.1.1. SEGONS EL CONTINGUT

**Plans:** fitxers de text o fitxer ASCII.

- Configuració: .ini .inf .conf Codi font: .sql
- .c .java
- Pàgina Web: .html .php .asp .xml
- Enriquits: .rtf .pg. tex

**Binaris:** no són de text, i requereixen un format per a interpretar-lo.

- Imatge: .jpg .gif .bmp
  - Vídeo: .mpg .mov .avi
  - Comprimits: .zip .gz .rar .tar
  - Ejecotables: .exe .com .cgi
  - Processadors de text: .doc .odt
- 

## **2.1.2. SEGONS EL SEU ACCÉS**

**Segons l'accés/organització tenim tres tipus d'arxius principals:**

- Arxius seqüencials**
- Arxius d'accés aleatori**
- Arxius indexats**




## 2.1.2.1 ARXIUSeqÜENCIALS

- Primers a aparéixer, el medi físic d'emmagatzematge eren les cintes magnètiques.
- Cada registre tenia uns camps fixos, que es gravaven en la cinta successivament, de forma ordenada.
- Molt útil per a imprimir etiquetes (en aquella època, de correu, per exemple ).



## 2.1.2.1 ARXIUS SEQÜENCIALS

- **Característiques:**

- Lectura ordenada obligatòria
  - No permet la reculada
  - Monousuaris
  - Estructura rígida de camps.
  - Lectures parcials però escriptures totals
  - EOF
  - Esborrat: cal reescriure tot menys el registre a esborrar (o marcar-lo)
  - No deixa buits
- 

## 2.1.2.2 ARXIUS D'ACCÉS ALEATORI

- Apareixen amb l'arribada dels disquets i discos durs.
- Podem accedir directament a la posició que indiquem de l'arxiu. Ja no fa falta recórrer-se tots els anteriors. Però cal calcular-ho i traduir-ho.
- $\text{Posició} = \text{NumRegistro} * \text{LongRegistro}$



# EXEMPLES

- Posem per cas que tenim un registre com aquest codificat en ANSI (1 byte per caràcter):.

- Nom: 80 caràcters ANSI.

- Adreça: 100 caràcters ANSI.

- Població: 50 caràcters ANSI.

- La seua longitud serà de 230 bytes. Això implica que el primer registre es trobarà en la posició 0; el segon registre es trobarà en la posició 230; el tercer estarà en la posició 460 i així successivament.

- Si en canvi codifiquem els caràcters en Unicode (UTF-16) de manera que la seua longitud és de 16 bits per caràcter, això és 2 bytes, el registre quedarà així:

- Nom: 80 caràcters UTF-16.

- Adreça: 100 caràcters UTF-16.


- Població: 50 caràcters UTF-16.

- La seua longitud serà de 460 bytes per a un sol registre, ja que cada caràcter ocupa 2 bytes.



## 2.1.2.2 ARXIUS D'ACCÉS ALEATORI

- **Característiques:**

- Posicionament immediat.
  - Registres de longitud fixa
  - Obertura lectura i/o escriptura
  - Ús concurrent
  - Esborrat de registre mitjançant zeros o marquejat
  - Problema que deixa molts buits
- 

## 2.1.2.3 ARXIUS INDEXATS

- Usa un índex de l'arxiu per a suportar els accessos.
- L'índex proveeix una capacitat de cerca per a arribar ràpidament a les proximitats d'un registre desitjat
- **Característiques**
  - Són bàsicament arxius d'accés aleatori amb utilitat (estructura o taula d'índexs, per a no calcular) per a accedir directament al registre buscat.
  - L'estructura guarda índexs a les posicions dels registres.
  - Arbore de cerca de claus.

## 2.2. COM S'EMMAGATZEMAVA LA INFORMACIÓ ABANS DELS SGBD

Abans d'aparèixer els SGBD, la informació es tractava i es gestionava utilitzant els típics sistemes de gestió d'arxius..

- **Cada aplicació tenia:**

Un conjunt d'arxius de dades.

Un conjunt de programes (altres arxius) que gestionaven aquests arxius de dades..

- **Cada programa gestionava els seus propis fitxers de dades..**
- **Es van usar molt en nòmines, control de comandes i manteniment de productes.**



## 2.2. COM S'EMMAGATZEMAVA LA INFORMACIÓ ABANS DELS SGBD

Els problemes apareixen quan:

- Augmenta el nombre d'usuaris.
- Augmenten les necessitats dels programes
- Necessitat d'interconnectar programes.





## 2.3 INCONVENIENTS D'UN SISTEMA DE GESTIÓ D'ARXIUS.

- **Redundància i inconsistència de les dades**, es produeix perquè els arxius són creats per diferents programes i van canviant al llarg del temps, és a dir, poden tindre diferents formats i les dades poden estar duplicats en diversos llocs. Per exemple, el telèfon d'un alumne pot aparéixer en més d'un arxiu. La redundància augmenta els costos d'emmagatzematge i accés, i porta amb si la inconsistència de les dades: les còpies de les mateixes dades no coincideixen per aparéixer en diversos arxius.
- **Dependència de les dades física-lògica**, o cosa que és el mateix, l'estructura física de les dades (definició d'arxius i registres) es troba codificada en els programes d'aplicació. Qualsevol canvi en aqueixa estructura implica el programador identificar, modificar i provar tots els programes que manipulen aqueixos arxius.

## 2.3 INCONVENIENTS D'UN SISTEMA DE GESTIÓ D'ARXIVS.

- **Dificultat per a tindre accés a les dades i ampliacions:**, proliferació de programes, és a dir , cada vegada que es necessita una consulta que no va ser prevista en l'inici implica la necessitat de codificar el programa d'aplicació necessari o buscar-lo manualment. És a dir , no permeten recuperar les dades necessàries d'una forma convenient i eficient.
- **Separació i aïllament de les dades**, és a dir , en estar repartits en diversos arxius, i tindre diferents formats, és difícil escriure nous programes que assegurin la manipulació de les dades correctes. Abans s'haurien de sincronitzar tots els arxius perquè les dades coincidiren.



## 2.3 INCONVENIENTS D'UN SISTEMA DE GESTIÓ D'ARXIUS.

- **Problemes de atomicidad.** És crucial assegurar que una vegada una fallada ha ocorregut i s'ha detectat, les dades es restauren a l'estat de consistència anterior a la fallada, o ocorre tot o no ocorre res. És difícil assegurar aquesta propietat en un sistema d'arxius..
- **Dificultat per a l'accés concurrent,** perquè en un sistema de gestió d'arxius és complicat que els usuaris actualitzen les dades simultàniament. Les actualitzacions concurrents poden donar per resultat dades inconsistents, ja que es pot accedir a les dades per mitjà de diversos programes d'aplicació..
- **Dependència de la estructura de l'arxiu amb el llenguatge de programació ,** perquè l'estructura es defineix dins dels programes. Això implica que els formats dels arxius siguin incompatibles. La incompatibilitat entre arxius generats per diferents llenguatges fa que les dades siguin difícils de processar..



## 2.3 INCONVENIENTS D'UN SISTEMA DE GESTIÓ D'ARXIVS.

- **Problemes en la seguretat de les dades.** Resulta difícil implantar restriccions de seguretat perquè les aplicacions es van afegint al sistema segons es van necessitant.
- **Problemes d'integritat de dades,** és a dir , els valors emmagatzemats en els arxius han de complir amb restriccions de consistència. Per exemple , no es pot inserir una nota d'un alumne en una assignatura si prèviament aqueixa assignatura no està creada. Un altre exemple, les unitats en magatzem d'un producte determinat no han de ser inferiors a una quantitat. Això implica afegir gran nombre de línies de codi en els programes. El problema es complica quan existeixen restriccions que impliquen diverses dades en diferents arxius.



## 2.3 INCONVENIENTS D'UN SISTEMA DE GESTIÓ D'ARXIVS.

- Solució ->Sorgeix així la idea de separar les dades contingudes en els arxius dels programes que els manipulen, és a dir , que es puga modificar l'estructura de les dades dels arxius sense que per això s'hagen de modificar els programes amb els quals treballen
- Es tracta d'estructurar i organitzar les dades de manera que es puga accedir a ells amb independència dels programes que els gestionen.
- I el més important que **TOTES** les dades estiguen en un únic repositori.
- Tots aquests inconvenients fan possible el foment, desenvolupament i aparició dels **SGBD**



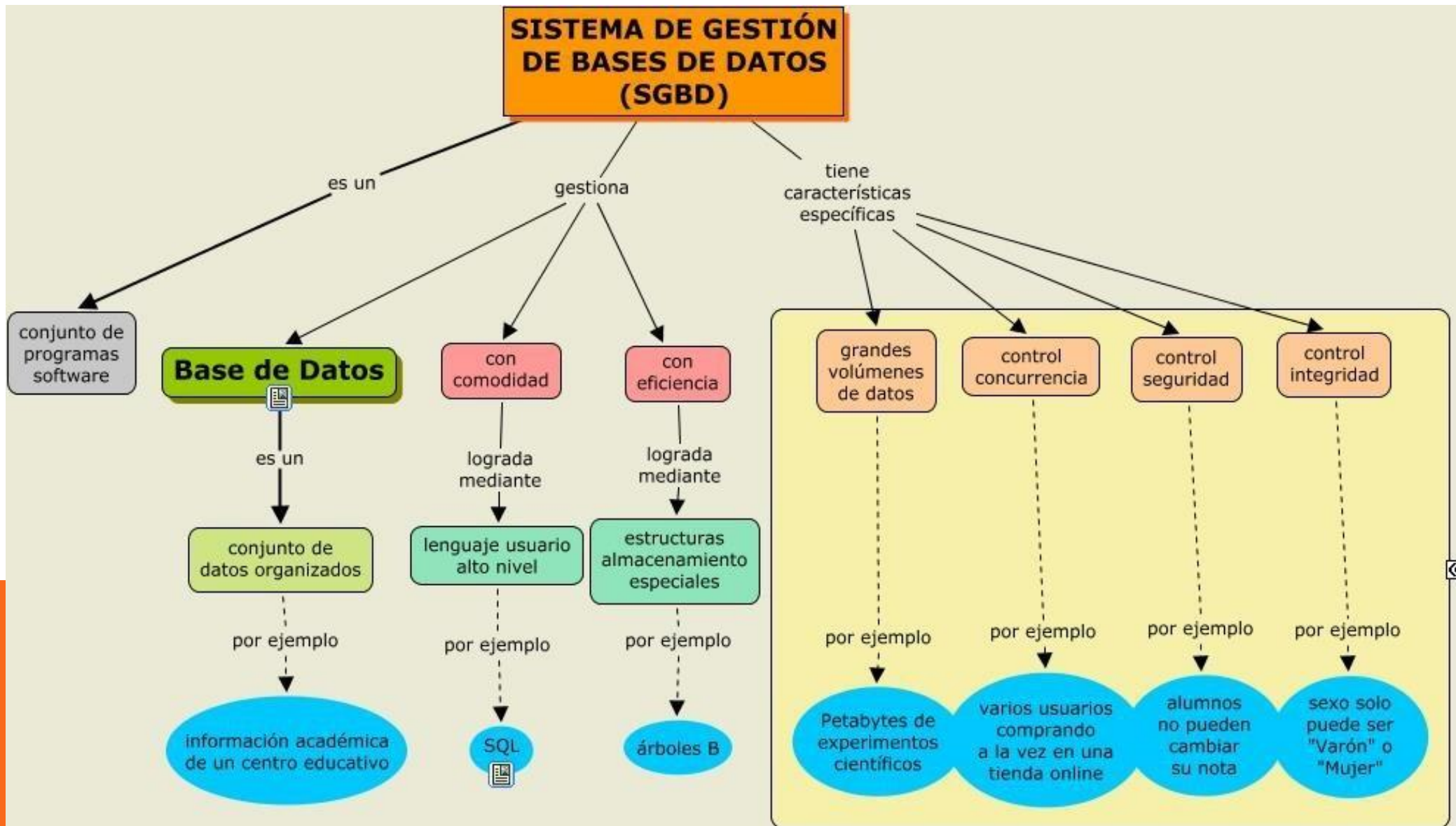


# 3.SGBD

- SGBD significa Sistema de Gestió de Bases de dades . Podem descompondre-ho com a Base de dades + Sistema Gestor.
- La base de dades és una col·lecció de dades i el sistema de gestió és un conjunt de programes per a emmagatzemar i recuperar aqueixes dades.
- Basant-nos en això podem definir SGBD com: SGBD és una col·lecció de dades interrelacionades i un conjunt de programes per a emmagatzemar i accedir a aqueixes dades d'una manera fàcil i efectiva.



# 3.SGBD. SISTEMA GESTOR DE BASES DE DADES



# 3.1 CARACTERÍSTIQUES D'UN SGBD

**Independència física i lògica.** Estructures físiques independents de la lògica i viceversa, accés a dades sense necessitat de conèixer l'estructura interna.

- **Eficaç accés a les dades:** Sense necessitat de conèixer com estan guardats, facilitat de manipulació (si està autoritzat) sense necessitat de ser Informàtic. (Eines gràfiques)
- **Mantindre la integritat i consistència** Garantir que la transacció es faça o es rebutge (Atomicidad)
- **Permeten la concurrència.** Accés compartit a la BD, controlant la interacció entre usuaris concurrents. Que no s'assabenten que tots dos estan alhora .
- **Mecanismes de suport i recuperació** per a restablir la informació en cas de fallades en el sistema
- **Coherència de dades: Complir restriccions.** Regles de la realitat
- **Redundància controlada**
- **Administració centralitzada. Eines d'administració..**
- **Seguretat de les dades:** Accés controlat a les dades de la BD mitjançant mecanismes de seguretat d'accés als usuaris.
- **Ús de transaccions.**

## 3.2 AVANTATGES SGBD SOBRE SISTEMA D'ARXIUS.

Hi ha diversos avantatges de sistema de gestió de base de dades sobre el sistema d'arxius. Alguns d'ells són :

- Sense dades redundants -Redundància eliminada per normalització de dades.
- Coherència i integritat de les dades: la normalització de les dades també s'ocupa d'ella
- Seguretat -Cada usuari té un conjunt diferent d'accés.
- Privacitat-Accés limitat
- Fàcil accés a les dades Fàcil
- recuperació Flexible

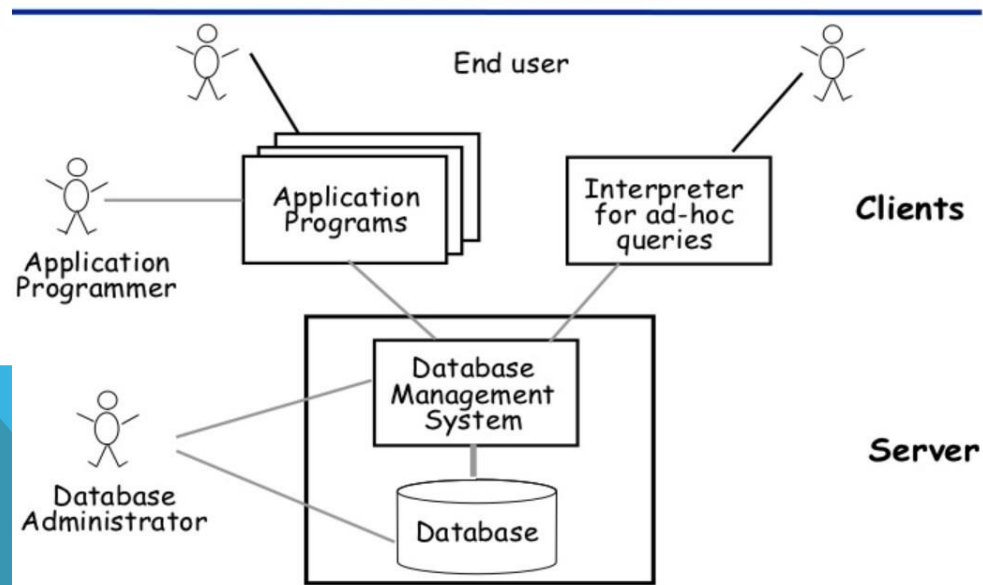
### Desavantatges de DBMS:.

- El **cost** d'implementació de SGBD és alt en comparació amb el sistema d'arxius.
- **Complexitat**: Els sistemes de base de dades són complexos d'entendre.
- **Rendiment**: Els sistemes de bases de dades són genèrics, fent-los adequats per a diverses aplicacions. No obstant això aquesta característica afecta el seu rendiment per a algunes aplicacions.

# BBDD I SGBD

- Usuaris finals
- Aplicacions
- Programador d'aplicacions
- SGBD
- Administrador de base de dades de base de dades (DBA)

## Client/Server-Architecture



## 3.3 CONCEPTE DE TRANSACCIÓ.

Una transacció és un conjunt d'instruccions que realitzen una acció i té les característiques **ACID** (**A**tomicity, **C**onsistency, **I**solation and **D**urability) :

- Atomicidad:** és la propietat que assegura que l'operació s'ha realitzat o no, i per tant davant una fallada del sistema no pot quedar a mig fer .
- Consistència:** *Integritat.* És la propietat que assegura que només es comença allò que es pot acabar. Per tant s'executen aquelles operacions que no trencaran les regles i directrius d'integritat de la base de dades .
- Aïllament:** és la propietat que assegura que una operació no pot afectar a unes altres. Això assegura que la realització de dues transaccions sobre la mateixa informació siguen independents i no generen cap mena d'error..
- Durabilitat:** és la propietat que assegura que una vegada realitzada l'operació, aquesta persistirà i no es podrà desfer encara que falle el sistema. .



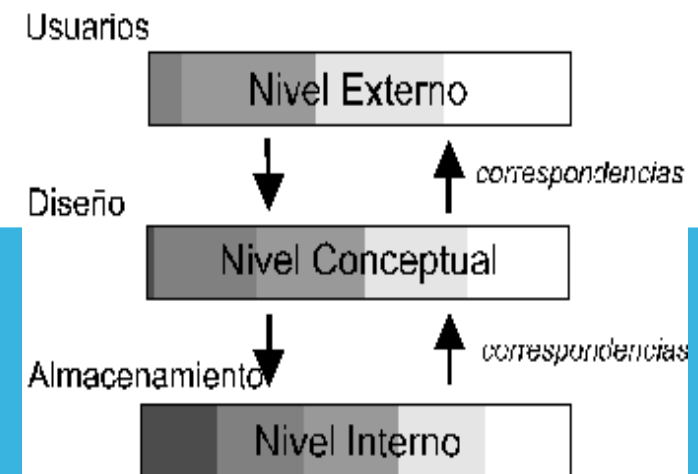
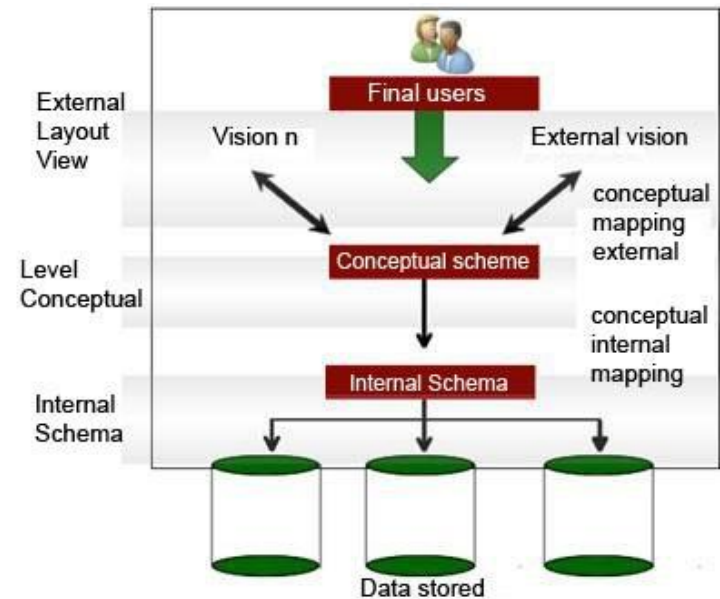
## 3.4 ARQUITECTURA SGBD

La arquitectura SGBD té com a objectiu principal que les aplicacions d'usuari estiguen separades de les dades físiques i aquests es divideixen a partir dels següents diagrames:

**Nivell intern:** utilitza un model de dades que mostra l'estructura d'emmagatzematge físic de la base de dades, els detalls de les dades guardades i les rutes d'accés. És el nivell més pròxim a l'emmagatzematge **físic**, és a dir, tal i com estan emmagatzemats les dades en l'ordinador (arxius, tipus de registre, longitud..)

**Nivell conceptual o lògic:** realitza una descripció completa de l'estructura de la base de dades però no ofereix detalls de les dades emmagatzemades en la base de dades. Defineix quines dades hi ha emmagatzemats i com es relacionen (**entitats, relacions, atributs...**)

**Nivell extern:** descriu les vistes de la base de dades a un grup d'usuaris i mostra quins usuaris tenen accés a aquesta base de dades. Nivell més pròxim als **usuaris**. Vistes de parts de la BD, bé per a restringir o per a simplificar..





# EXEMPLE AMB MODEL RELACIONAL

**Nivell extern:** Visió parcial de les taules de la BD segons l'usuari. Per exemple, la vista que es mostra en la Taula, obté el llistat de notes d'alumnes amb les següents dades: Curs, Nom, Nom d'assignatura i Nota.

**Nivell lògic i conceptual:** Definició de totes les taules, columnes, restriccions, claus i relacions. En aquest exemple, disposem de les taules que estan relacionades, estudiants i activitats, cadascuna amb la seua clau que és ID

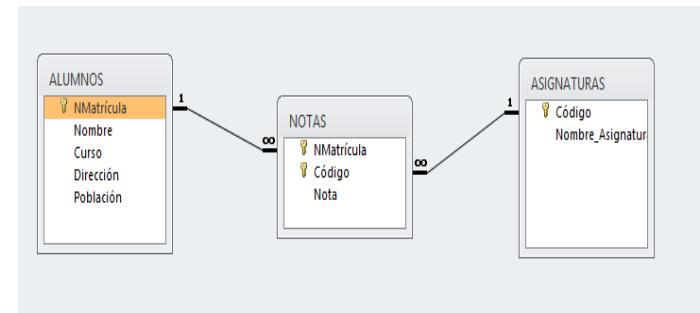
**Nivell Intern:** En una BD les taules s'emmagatzemen en arxius de dades de la BD. Si hi ha claus, es creen índexs per a accedir a les dades, tot això contingut en el disc dur, en una pista i en un sector, que només el SGBD coneix. Davant una petició, sap a quina pista, a quin sector, a quin arxiu de dades i a quins índexs accedir

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





# AVANTATGES DE L'ARQUITECTURA.

**Independència lògica:** la capacitat de modificar l'esquema conceptual sense haver de alterar els esquemes externs ni els programes d'aplicació. Es podrà modificar l'esquema conceptual per a ampliar la BD o per a reduir-la, per exemple , si s'elimina una entitat, els esquemes externs que no es referisquen a ella no es veuran afectats.

•**Independència física:** la capacitat de modificar l'esquema intern sense haver de alterar ni l'esquema conceptual, ni els externs. Per exemple , es poden reorganitzar els arxius físics amb la finalitat de millorar el rendiment de les operacions de consulta o d'actualització, o es poden afegir nous arxius de dades perquè els que hi havia s'han omplit. La independència física és més fàcil d'aconseguir que la lògica, perquè es refereix a la separació entre les aplicacions i les estructures físiques d'emmagatzematge..

•Les correspondències entre nivells requereix ampliar el diccionari de dades.



# 3.5 FUNCIONS SGBD

La funció principal: perm als usuaris les 4 operacions fonamentals:

- Creació i inserció
- Consulta
- Actualització
- Esborrat

Amb 2 objectius:

1. Visió abstracta de les dades (amagar com s'emmagatzemen i mantenen les dades)
2. Velocitat (bon temps de resposta) i seguretat (que siga veritat el que resulta).




# FUNCIÓ DE DEFINICIÓ.

- Llenguatge SQL de
- definició Nivell Intern:
  - Espai físic Longitud
  - de camps.
  - Manera de representació.
  - Camins d'accés, punters, índexs
- Nivell Conceptual
  - Definició d'entitats, atributs i relacions
  - Autorització d'accessos.
- Nivell Extern
  - Construccions gràfiques de taules i relacions (Acces)
- Tot s'emmagatzema en el Diccionari de dades.

# FUNCIÓ MANIPULACIÓ

Permet buscar, afegir, suprimir o modificar les dades d'una BD sempre d'acord amb les especificacions i restriccions de seguretat..

- Utilitza el llenguatge DML
  - Nivell Intern: Algorismes eficients d'accés a les dades.
  - Nivell Conceptual: Abstracció de la part interna, construcció de consultes o actualitzacions.. Vistes a programadors.
  - Nivell extern: Vistes a usuaris. Facilitat d'ús amb eines gràfiques (Access)
- 

# 3.6 COMPONENTS D'UN SGBD

- Components humans. USUARIS.
- Components tècnics

## Components Funcionals

- Components de processament de consultes.
- Components de Gestió d'emmagatzematge.

## Implementació Física del sistema

- Arxius de dades.
- Diccioniari de dades.



# 3.6.1 USUARIS

- **Administradors: disseny físic**

Un administrador de bases de dades , un analista de bases de dades o un desenvolupador de bases de dades és la persona responsable de gestionar la informació dins d'una organització.

La funció principal de l'Administrador de la base de dades és administrar, desenvolupar, mantindre i implementar les polítiques i procediments necessaris per a garantir la seguretat i integritat de la base de dades corporativa. Les funcions secundàries dins de la classificació d'Administrador de bases de dades poden incloure seguretat, arquitectura, emmagatzematge i / o anàlisi de negoci. Altres funcions principals inclouen:

- Implementació de models de dades
- Disseny de base de dades
- Accessibilitat a la base
- Problemes d'acompliment
- Problemes de capacitat.
- Replicació de dades
- Manteniment de la taula

- **Dissenyadors: disseny lògic. Programadors.**
- Implementen els programes **Usuaris finals.**
- Utilitzen el resultat final.

## 3.6.2 CONSULTES I EMMAGATZEMATGE

- **Processador de consultes** : el processador de consultes es transforma en una sèrie d'instruccions de baix nivell . S'utilitza per a interpretar la consulta de l'usuari en línia i convertir-la en una sèrie eficaç d'operacions en una forma capaç de ser enviada al gestor de dades de temps d'execució per a la seua execució. El processador de consultes utilitza el diccionari de dades per a trobar l'estructura de la part rellevant de la base de dades i utilitza aquesta informació per a modificar la consulta i preparar el pla òptim per a accedir a la base de dades .
- **Optimizador de consultes:** Els optimizadores de consultes determinen una estratègia òptima per a la execució de la consulta



# COMPONENTS DEL GESTOR D'EMMAGATZEMATGE.

- **Gestor de dades:** El gestor de dades és responsable de les dades de la base de dades . Proporciona un sistema capaç de recuperació al sistema després d'alguna fallada. Inclou administrador de recuperació i gestor de búfer. El gestor de búfer és responsable de la transferència de dades entre la memòria principal i l'emmagatzematge secundari (com el disc) També es refereix com el gestor de caixet..
- **Comprobador d'integritat:** comprova les restriccions d'integritat perquè només es puguin introduir dades vàlides en la base de dades .
- **Gestor de transaccions:** El gestor de transaccions assegura que les propietats de la transacció han de ser coherents en el sistema
- **Planificador:** Proporciona un entorn en el qual múltiples usuaris poden treballar amb la mateixa informació al mateix temps , és compatible amb la simultaneïtat



## 3.6.3 IMPLEMENTACIÓ FÍSICA DEL SISTEMA

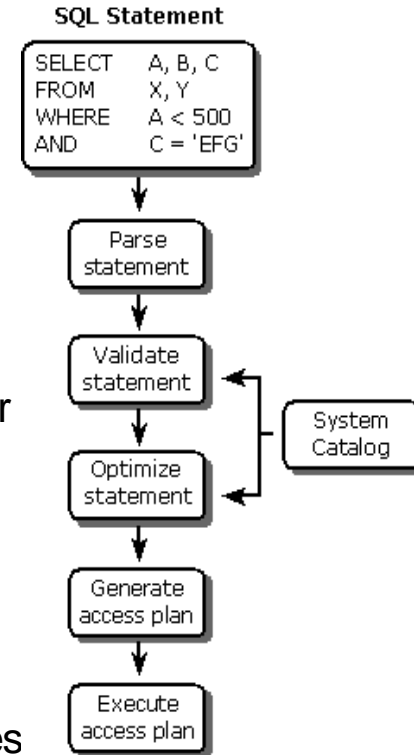
- Arxius de dades: Emmagatzema la base de dades en si.
- Diccionari de dades: Emmagatzema metadades sobre l'estructura d'aquests .
- Índexs: Proporciona accés ràpid a les dades.
- Dades estadístiques: Emmagatzema dades estadístiques sobre les dades. El processador de consultes usa aquesta informació per a planificar-les..



## 3.6.4 EXECUCIÓ DE PROCESSOS EN UN SGBD

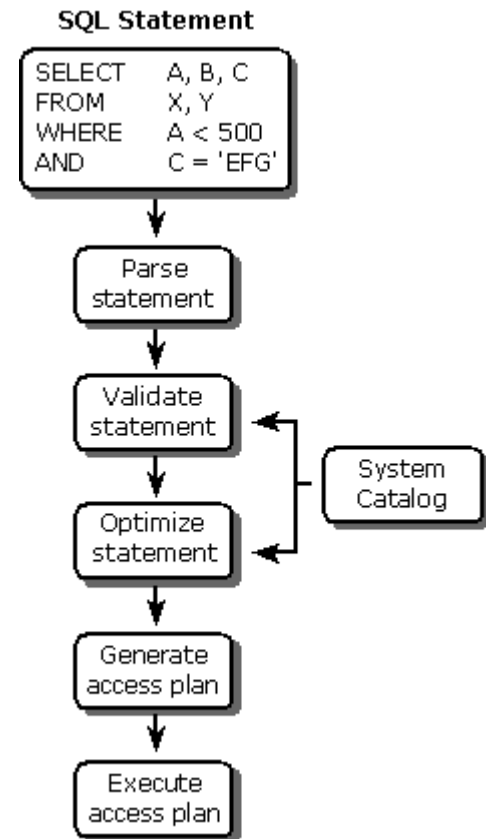
Per a processar una sentència SQL, un SGBD realitza els següents cinc passos:

- **El SGBD primer analitza la instrucció SQL.** Estrenca la declaració en paraules individuals, anomenades tokens, s'assegura que la declaració té un verb vàlid i clàusules vàlides, i així successivament. En aquest pas es poden detectar errors de sintaxis i errors ortogràfics.
- **El SGBD valguda la sentència.** Comprova la instrucció en el catàleg del sistema. Existeixen totes les taules de la sentència en la base de dades ? Existeixen totes les columnes i els noms de columna no són ambigus? Té l'usuari els privilegis necessaris per a executar la sentència? En aquest pas es poden detectar uns certs errors semàntics.
- **El SGBD genera un pla d'accés per a la sentència.** El pla d'accés és una representació binària dels passos que es requereixen per a dur a terme la declaració; és l'equivalent DBMS del codi executable.
- **El SGBD optimitza el pla d'accés.** Explora diverses maneres de dur a terme el pla d'accés. Es pot usar un índex per a accelerar la cerca? Deuria el SGBD aplicar primer una condició de cerca a la Taula A i, després unir-la a la Taula B, o hauria de començar amb la unió i usar la condició de cerca després? Es pot evitar una cerca seqüencial a través d'una taula o reduir-la a un subconjunt de la taula? Després d'explorar les alternatives, el SGBD tria un d'ells..



# EXECUCIÓ DE PROCESSOS EN UN SGBD

- **El SGBD executa la instrucció executant el pla d'accés..**
- Els passos utilitzats per a processar una sentència SQL varien en la quantitat d'accés a la base de dades que requereixen i la quantitat de temps que prenen. L'anàlisi d'una sentència SQL no requereix accés a la base de dades i pot realitzar-se molt ràpidament.
- L'optimització, d'altra banda, és un procés que requereix molta CPU i requereix accés al catàleg del sistema. Per a una consulta múltiple complexa, el optimitzador pot explorar milers de maneres diferents de dur a terme la mateixa consulta. No obstant això, el cost d'executar la consulta de manera ineficient sol ser tan alt que el temps invertit en l'optimització és més que recuperat en la major velocitat d'execució de la consulta. Això és encara més significatiu si el mateix pla d'accés optimitzat es pot utilitzar una vegada i una altra per a realitzar consultes repetitives.



# 4. TIPUS DE SGBD.

- Segons el model:
  - Jeràrquic
  - Xarxa
  - Relacional.
  - Orientat a Objectes. UML i XML
- Segons arquitectura
  - Centralitzades.
  - Client/Servidor
  - Distribuïdes. Homogenis i heterogenis.



# MODEL JERÀRQUIC

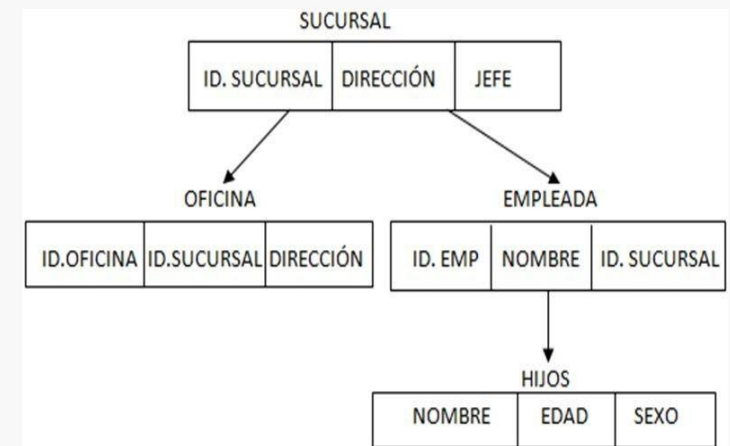
Un model de base de dades jeràrquic és un model de dades en el qual les dades s'organitzen en una estructura similar a un arbre.

Les dades s'emmagatzemen com a registres que estan connectats entre si a través d'enllaços. Un registre és una col·lecció de camps, amb cada camp que conté només un valor. El tipus d'entitat d'un registre defineix els camps que conté el registre.

Un registre en el model de base de dades jeràrquic correspon a una fila (o tupla) en el model de base de dades relacional i un tipus d'entitat correspon a una taula (o relació).

No suporta relacions molts-a-molts.

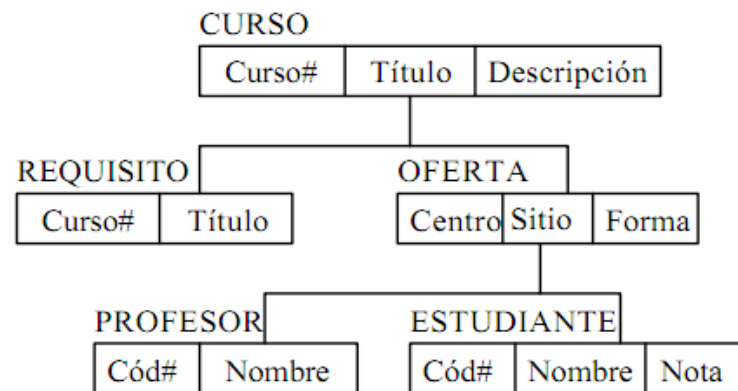
## MODELO JERÁRQUICO (árbol)



# MODEL EN XARXA

Semblança al jeràrquic, conjunt de registres enllaçats mitjançant punter, però aquesta vegada en forma de graf..

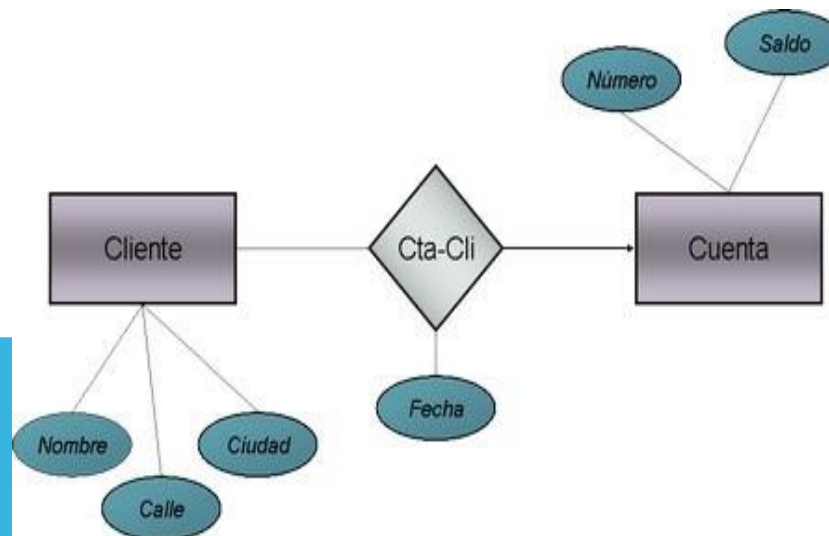
- Almenys teòricament sí que disposava de comunicació bidireccional, però en la pràctica era massa costós i ho feia amb diversos /aries 1:M



# MODEL RELACIONAL

El més estés i comercialment l'únic amb èxit, les dades es descriuen com a relacions que se solen representar com a taules bidimensionals consistents en files i columnes.

- Cada fila (tupla, en terminologia relacional) representa una ocurrència.
- Les columnes (atributs) representen propietats de les files.
- Cada tupla s'identifica per una clau primària o identificador.
- Els enllaços no són capdavanters són les mateixes dades.



# MODEL ORIENTAT A OBJECTES.

- En una base de dades d'objectes (OODBMS) la informació es representa en forma d'objectes tal com s'utilitza en la programació orientada a objectes. Les bases de dades d'objectes són diferents de les bases de dades relacionals que estan orientades a taules. Les bases de dades objecte-relacionals són un híbrid de tots dos enfocaments.
- La majoria de les bases de dades d'objectes també ofereixen algun tipus de llenguatge de consulta, permetent que els objectes es troben usant un enfocament de programació declarativa. És en l'àrea dels llenguatges de consulta d'objectes, i la integració de la consulta i les interfícies de navegació, on es troben les majors diferències entre els productes. Un intent de normalització va ser feta pel ODMG amb el llenguatge de consulta d'objectes, OQL.
- L'accés a les dades pot ser més ràpid perquè un objecte pot ser recuperat directament sense una cerca, seguint els punters.
- Una altra gran variació entre els models és la forma en què es defineix l'esquema d'una base de dades. A més, una característica general és que el llenguatge de programació i l'esquema de la base de dades utilitzen les mateixes definicions de tipus..
- Les aplicacions multimèdia se simplifiquen perquè els mètodes de classe associats amb les dades són responsables de la seua interpretació
- L'eficiència de la base de dades també ha millorat molt en àrees que demanden quantitats massives de dades. Per exemple, un banc podria obtenir la informació del compte de l'usuari i proporcionar-los eficientment molta informació, com a transaccions, informació del compte, etc.





# BBDD CENTRALITZADA

- La BBDD està localitzada, guardada i mantinguda en una sola localització
- Avantatges:
  - Tindre una visió completa de les dades
  - Manejar la BBDD éssenzill
- Desavantatges:
  - Necessites estar en l'ordinador on es trobe la BBDD



# CLIENT /SERVIDOR

- Aquesta arquitectura és la més usada hui dia
- Aquesta BBDD usa el model client servidor. Els clients fan peticions a la BBDD usant una xarxa
- **Avantatges:**
  - Tenen una visió completa de les dades
  - Manejar la BBDD és més fàcil
  - Connectar des de diferents llocs
- **Desavantatges:**
  - Colls de botella pels accessos concurrents de diferents usuaris al mateix arxiu.

# BASE DE DADES DISTRIBUÏDA

Tota la informació s'emmagatzema en múltiples ubicacions físiques.

## **Avantatges:**

- Els usuaris no interferiran entre si en accedir / manipular dades
- Augmenta la velocitat des que els arxius es recuperen des de la ubicació més pròxima
- Si falla un lloc, el sistema pot recuperar-se

## **Desavantatges**

- Temps de sincronització de les múltiples bases de dades
- Replicació de dades per a cada arxiu de base de dades diferent



# OPEN SOURCE | BBDD COMERCIALS

## Commercial

ORACLE®



## Open source



PostgreSQL



# NOSQL

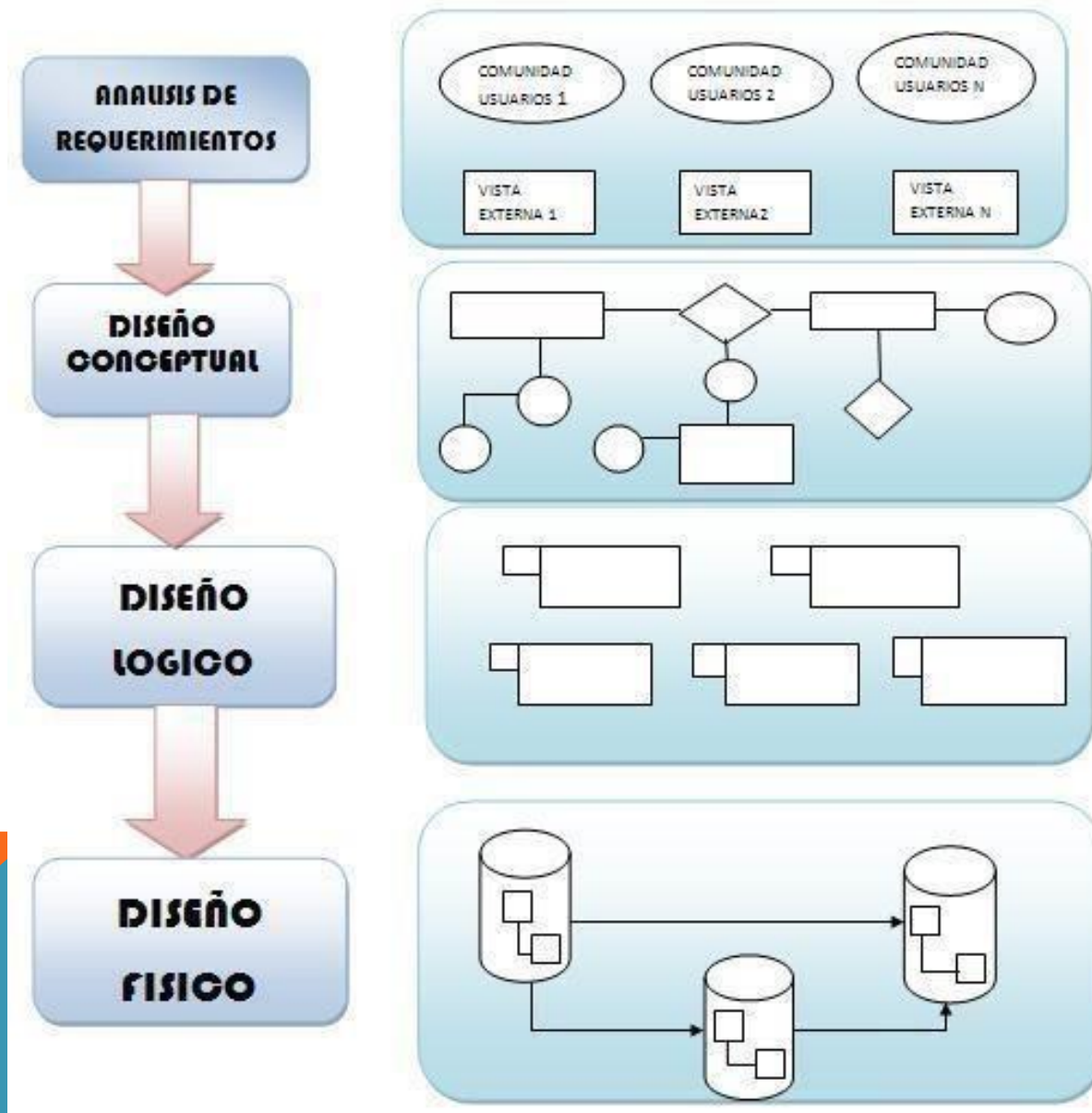
- La Base de dades proporciona un mecanisme per al emmagatzematge i recuperació de dades diferents a les relacions utilitzades en les bases de dades relacionals.
- Han existit des de finals dels anys 1960, però la popularitat va créixer a causa de les necessitats de la Web 2.0 i empreses com Facebook, Google i Amazon
- Les bases de dades NoSQL s'utilitzen en grans aplicacions de dades i en temps real.
- Molts DB NoSQL comprometen la consistència en favor de la disponibilitat, tolerància de partició i velocitat.
- Basat en les classificacions de popularitat de 2015, les bases de dades NoSQL més populars són **MongoDB, Apatxe Cassandra i Redis.**



# 5. FASES DEL DISSENY EN UNA BASE DE DADES

- Els requisits i la fase d'anàlisi de la recol·lecció produeixen tant requisits de dades com requisits funcionals. Els requisits de dades s'utilitzen com una font de disseny de base de dades. Els requisits de dades han d'especificar-se en la forma més detallada i completa possible.
- Paral·lelament a la especificació dels requisits de dades, és útil especificar els requisits funcionals coneguts de la aplicació. Els requisits funcionals s'utilitzen com una font de disseny de programari d'aplicació..
- Tingueu en compte que algunes fases són independents del sistema de gestió de bases de dades i algunes són dependents. Exemple: SQL serà diferent si triem MySQL o Oracle

# FASES DEL DISEÑO EN UNA BD



# DISSENY CONCEPTUAL

- Una vegada reunits i analitzats tots els requisits, el següent pas és crear un esquema conceptual per a la base de dades , utilitzant un model de dades conceptuais d'alt nivell. Aquesta fase es diu disseny conceptual.
- El resultat d'aquesta fase és un **diagrama d'Entitat-Relació (ER)** o diagrama de classes UML.
- És un model de dades d'alt nivell de l'àrea d'aplicació específica. Descriu com diferents entitats (objectes, elements) estan relacionades entre si. També descriu quins atributs (característiques) té cada entitat. Inclou les definicions de tots els conceptes (entitats, atributs) de l'àrea d'aplicació..
- Hi ha diverses anotacions per a dibuixar el diagrama ER.



# DISSENYE LÒGIC

- El resultat de la fase de disseny lògic és un conjunt d'esquemes de relació. **El diagrama ER** o diagrama de classes és la base d'aquests esquemes de relació..
- Crear els diagrames de relació és una operació mecànica. Hi ha regles de com el model ER o diagrama de classes es transfereix a diagrames de relació..
- Els diagrames de relació són la base per a les definicions de taules..
- En aquesta fase (si no es realitza en la fase anterior) es defineixen les claus primàries i les claus externes.

# NORMALITZACIÓ

La normalització és l'última part del disseny lògic. L'objectiu de la normalització és eliminar la redundància i les possibles anomalies d'actualització..

**Redundància** significa que les mateixes dades es guarden més d'una vegada en una base de dades . Actualitzar l'anomalia és una conseqüència de la redundància. Si una part de dades es guarda en més d'un lloc, les mateixes dades han d'actualitzar-se en més d'un lloc.

La **normalització** és una tècnica mitjançant la qual es pot modificar el diagrama E-R per a reduir la redundància. Cada fase de normalització afig més relacions (taules) a la base de dades .



# DISSENY FÍSIC

L'objectiu de la última fase del disseny de la base de dades , el disseny físic, és implementar la base de dades .

En aquesta fase es deu saber quin sistema de gestió de base de dades (SGBD) s'utilitza. Per exemple , diferents SGBD tenen diferents noms per a moltes tipus de dades i tenen diferents tipus de dades..

S'escriuen les clàusules SQL per a crear la base de dades . Es defineixen els índexs, les restriccions d'integritat (regles) i els drets d'accés dels usuaris.



# CONCEPTES DEL SGBD RELACIONAL

**SGBDR** = Sistema de gestió de base de dades relacional

**Taula / Relació:** Una taula és una col·lecció de dades representades en files i columnes.

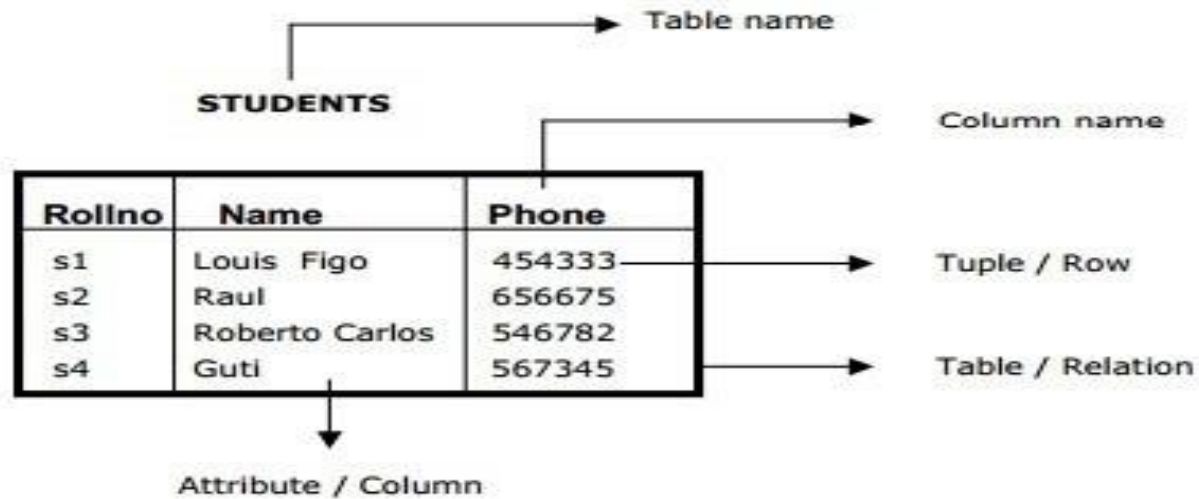
**Registres / Tupla / Fila:** Cada fila d'una taula es coneix com a registre o també es coneix com tupla

**Nom de camp / columna:** La taula de baix té tres camps: Rollno, Nom i Telèfon.

**Columna / Atribut:** Cada atribut i els seus valors es coneixen com a atributs en una base de dades . Per exemple , El conjunt de valors del camp Nom és una de les tres columnes de la taula Estudiants.



# CONCEPTES DEL SGBD RELACIONAL



**Figure 1:** A table in relational model.

# LLENGUATGES DE SGBD.

Els llenguatges de base de dades s'utilitzen per a llegir, actualitzar i emmagatzemar dades en una base de dades . Hi ha diversos llenguatges que poden ser usats per a aquest propòsit; un d'ells és SQL (Structured Query Language).

SQL és un llenguatge estàndard per a accedir i manipular bases de dades .



# LLENGUATGE DE DEFINICIÓ DE DADES (DDL)

Llenguatge de definició de dades (DDL): DDL s'utilitza per a especificar l'esquema de la base de dades . Prenguem SQL per exemple per a categoritzar les declaracions que venen baix DDL.

- Per a crear la instància de base de dades : CREATE

- Per a alterar l'estructura de la base de dades -
- ALTER Per a eliminar instàncies de base de dades -
- DROP

Per a eliminar taules en una instància de base de dades

- - TRUNCATE

Per a canviar el nom de les instàncies de base de dades -  
RENAME

# LLENGUATGE DE MANIPULACIÓ DE DADES (DML)

**Llenguatge de manipulació de dades (DML):** DML s'utilitza per a accedir i manipular dades en una base de dades .

- Per a llegir registres de la (les) taula (es) - SELECT
- Per a inserir registres en la (les) taula (es) - INSERT
- Per a actualitzar les dades en la (les) taula (es) - UPDATE
- Per a esborrar els registres de la taula - DELETE



# LLENGUATGE DE CONTROL DE DADES (DCL)

**Llenguatge de control de dades (DCL):** DCL s'utilitza per a concedir i revocar l'accés d'usuari en una base de dades .

- Per a concedir accés a l'usuari - GRANT
- Per a revocar l'accés de l'usuari - REVOKE

