ASIX M02. Gestió de Bases de Dades
DAW M02. Bases de Dades

UF1. Introducció a les Bases de Dades

Autor: *Robert Ventura Vall-llovera*

Data creació: Setembre2015
Actualització: Setembre 2021

NF1 Introducció a les base de dades

Introducció a les base de dades i a Sistemes Gestora de Base de Dades. Història i tipologia de models de dades. Introducció a sistemes distribuïts

Contingut

Introducció i conceptes

1.1. Dades i base de dades

Dades:

> *Fets coneguts que poden enregistrar-se i que tenen un significat implícit. Per exemple: noms, números de telèfon, adreces, ...*

Un dels elements més preuats per una empresa ja no són els treballadors i l'experiència d'aquests, que també, sinó les **dades** que aquesta empresa gestiona, acumula i interroga per millorar-ne el rendiment i extreure'n estratègies de mercat.

Aquestes dades i/o fets no ens serviran de res sinó les endrecem de forma adequada.

Per això moltes vegades parlem del concepte de **base de dades**:

Definicions de base de dades:

> *Una base de dades és un conjunt endreçat d'informació que s'emmagatzema mitjançant algun tipus de suport i que es pot consultar i mantenir. Ex: recull de temperatures, padró d'habitants.*

> *Conjunt estructurat de dades relacionades entre sí. Entenent com a dades fets coneguts que poden registrar-se i que tenen un significat implícit. Per exemple: recull de temperatures, padró d'habitants d'un municipi, noms i telèfons de les persones que coneixem, etc...*

Donada la definició anterior podríem considerar que el conjunt de paraules i frases que formen aquesta pàgina de text com a dades relacionades entre sí, de manera que són una base de dades. Per això l'acceptació comú és un pèl més restrictiva i ha de complir les següents propietats:

- Representen algun aspecte del món real (minimón o univers de discurs).
    - **Exemple:** Un taller mecànic:
- És una col·lecció coherent de dades amb un significat. Un conjunt aleatori de dades no es pot considerar una BD.
    - **Exemple:** gestió de matriculació d'una universitat: alumnes, assignatures, taxes, convocatòries, etc...

- Es dissenya, es construeix i s'omple amb dades amb un propòsit específic, destinat a un grup d'usuaris concret i amb alguna utilitat o finalitat per aquests usuaris.

    - **Exemple:** les dades de temperatura, humitat, precipitació de les principals ciutats del món no es podrien considerar una BD, però per el meteoròlegs o instituts de meteorologia si que tenen un sentit i propòsit específic.

A més a més de poder emmagatzemar la informació de forma estructurada i gestionar millor el nostre mini-món a través de les base de dades. Aquestes a dia d'avui són clau per prendre decisions estratègiques.

Les principals característiques que han de tenir les BD són:

- **Persistència:**

    - Les dades han de durar en el temps.

    - Només hem d'emmagatzemar dades potencialment rellevants

- **Relacions:**

    - Entitats: conjunt de dades respecte un tema (client, producte, estudiant, assignatura, vehicle, peces d'un vehicle, etc...)

    - Relacions: connexions que existeixen entre les entitats (client compra productes, el vechicle està format per peces, un assignatura té diferents estudiants matriculats,etc...)

- **Compartir:**

    - Múltiples usos: diferents maneres d'entrar, mantenir i visualitzar les dades (a través de web, mòbil, programes específics, programes automatitzats, visualitzar les dades en informes, gràfiques, etc..)

    - Múltiples usuaris: Interessa que moltes persones puguin utilitzar al mateix temps la base de dades. Tant per mantenir-la, com per visualitzar-ne informació. (dades bancàries: un comercial d'una oficina realitza gestions sobre un compte, mentre el propietari del compte està extraient diners mitjançant un caixer automàtic).

La generació i manteniment de les base de dades poden ser **manuals** o **automatitzats**.

El catàleg de targetes que algunes biblioteques encara utilitzen guarda la informació de tots els llibres que tenen seria un bon exemple de base de dades manual.

Les base de dades computaritzades/automatitzades es poden crear i mantenir amb un grup de programaris específics per aquesta tasca o bé mitjançant un Sistema Gestor de Base de Dades (SGBD)

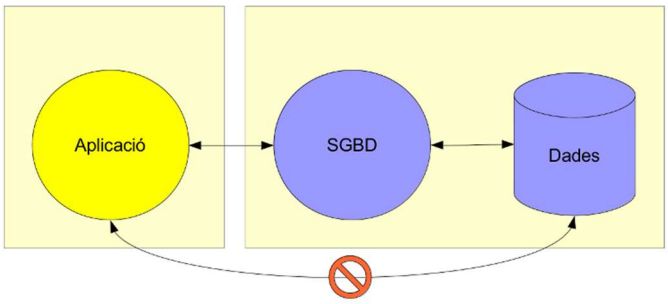
1.2. Sistemes Gestors de Base de Dades (SGBD)

Definició

> *Un sistema de gestor de base de dades (SBGD), en inglès Database Management System (**DBMS**) és un programari(1) que permet gestionar un conjunt de fitxers interrelacionats(2) permeten als usuaris crear i mantenir una BD oferint un mínim de prestacions(3).*

(1) Pot ser una aplicació o unes llibreries. En cas de ser aplicació caldrà disposar d'un sistema de comunicació de processos per tal de poder-nos connectar (per exemple model client-servidor). Existeixen diferents sistemes de comunicació: per pipes, per xarxa, per memòria compartida. En cas de ser llibreries caldrà enllaçar-les al nostre programa, ja sigui de manera estàtica en temps de compilació del programa o bé de manera dinàmica durant l'execució del mateix.

(2) Les aplicacions no accedeixen directament a les dades, li demanen al SGBD (sistema gestor de base de dades) les operacions que volen realitzar. Exemple d'operacions poden ser modificació de dades o consulta de dades existents. El SGBD és qui realitza aquestes operacions mantenint la integritat de les dades i aplicant la seguretat que estigui definida al sistema.



(3) Per tal que un programari pugui considerar-se SGBD ha de ser capaç de facilitar una sèrie d'operacions. Una de les capacitats més importants que ha de tenir és la independència de les dades respecte les aplicacions (podem seguir accedint a les dades mitjançant les utilitats del SGBD sense necessitat d'altres aplicacions, podem canviar l'estructura de les dades des del propi SGBD). Altres capacitats serien controlar l'accés a les dades i mantenir la seguretat i la integritat de les mateixes.

Alguns SGBD:

Quin SGBD hem d'escollir?

Aquesta pregunta no té una única resposta ja que molts del SGBD que trobem al mercat tenen diferents característiques entre uns i altres i ens ajuden i s'adapten millor a certs problemes que nosaltres volem resoldre.

Abans d'incorporar un SGBD en el nostre sistema d'informació cal abans preguntar-nos moltes preguntes: **què necessitem** i **què estem disposats a pagar, quin suport hi ha de la solució adoptada,...**.

Un bon indicador de com està el mercat de les noves tecnologies és veure el famós quadrant màgic de la consultora americana Gartner, destinada a la consultoria i investicació en el mercat de les noves tecnologies.

El quadrant màgic de Gartner és una representació gràfica de la situació del mercat d'un producte tecnològic concret. En aquest cas SGBD.

http://www.gb-advisors.com/es/cuadrante-de-gartner/

http://www.bigdata-social.com/informe-cuadrante-magico-gartner/

El gràfic es divideix en quatre parts a on es distribueixen les diferents companyies o solucions SGBD en funció de la seva tipologia depenent de dos eixos **l'habilitat d'execució** (*ability to execute*): Habilitat que l'empresa/producte ven i ofereix suport als seus productes i serveis a nivell global. També podríem definir com l'habilitat que té l'empresa per executar amb èxit la seva visió de mercat. I **l'abast de visió** (*completeness of vision*): Que es refereix al potencial de l'empresa/producte)

- **Líders** (*leaders*): Són qui tenen la millor puntuació resultant de combinar l'habilitat d'execució i l'abast de visió.

- **Aspirants** (challengers): Caracteritzats per oferir bones funcionalitats i un nombre considerable d'instal·lacions del producte, però sense la visió dels líders.

- **Visionarios** (visionaries): Tenen habilitat per anticipar-se a les necessitats del mercat, però en canvi no responen amb una plataforma sòlida per respondre a les necessitats a nivell global.

- **Nínxols específics** (niche players): Enfocats a determinades àreas de les tecnologies d'emmagatzematge de BD, però sense disposar d'una suite completa o bé poca presència en el mercat.

A continuació es veu l'evolució dels diferents Quadrants de Gartner en "*Operational Database Management Systems*".

Si ens fixem per exemple amb l'empresa AWS, no apareixia en el 2014 i a l'any següent era una empresa leader!

L'empresa MongoDB es va moure de ser una empresa d'arribar a un alt nivell d'execució a predir i anticipar-se en el mercat.

Source: Gartner (October 2016)

Source: Gartner (October 2016)

Source: Gartner (November 2017)

La informació és un recurs imprescindible a la presa de decisions de les empreses i, per tant, genera la necessitat de mantenir algun tipus d'infraestructura per manipular la informació ràpidament i amb el mínim esforç.

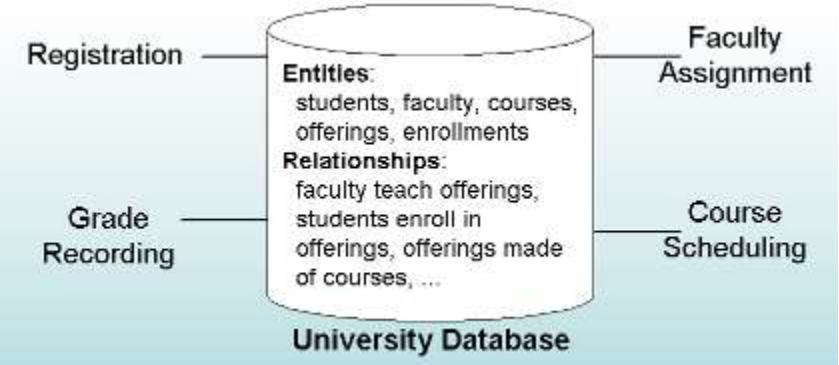
A aquesta infraestructura se l'anomena "Sistema d'Informació".

Sistema d'informació

> *Conjunt d'elements relacionats entre sí d'acord a certes regles que aporten a l'organització la informació necessària per a l'acompliment dels seus propòsits.*

Com hem dit anteriorment, els sistemes d'informació no només són els programari que utilitzem en la nostra empresa per gestionar el nostre negoci. Un Sistema d'Informació el component tots els elements que per tal de dur a terme la gestió del nostre negoci: Els procediments, les persones i la pròpia informació, etc...

1.3. Exemple: Universitat



Dins d'una Universitat volem dur el manteniment d'informació diversa (procés de matrícula, informes de les diferents qualificacions dels alumnes, programació/horari de les matèries i/o cursos, assignació de professors i alumnes als diferents cursos, etc...). Cada submón conté o comparteix diferents entitats (estudiants, facultats, cursos, professors, semestres) i relacions (els estudiants realitzen assignatures, els estudiants estudien en una facultat, la universitat té diferents facultats, cada facultat ofereix una sèrie d'assignatures,...).

Per definir aquesta base de dades, hem d'especificar l'estructura de cada element que en volem guardar informació, indicant els diferents tipus d'elements d'informació que volem emmagatzemar en cada element.

Per exemple: en el cas d'un estudiant volem guardar: El nom, el DNI, el Nº d'estudiant, les carreres i les assignatures que està matriculat.

La manipulació de la base de dades consisteix en l'actualització de les dades i en les consultes que en podem fer.

Consultes:

- Obtenir un llistat de tots els alumnes que estan matriculats en una assignatura concreta
- Quins alumnes van obtenir més d'un 7 en l'assignatura de Base de dades l'any passat.
- Quants professors diferents de Base de dades hi han hagut al llarg de tots els anys.

Actualització:

- Modificar l'adreça d'un alumne
- Modificar el professor que imparteix una assignatura concreta.
- Esborrar una assignatura perquè no hi ha cap alumne que s'hi hagi matriculat.
- Assignar un alumne en una assignatura.

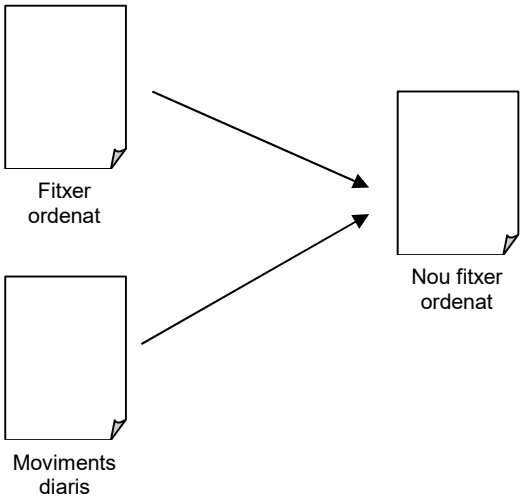
Breu història de les base de dades

1.1. Dècada dels 50s

La dècada dels 50s es van inventar les cintes magnètiques, que només es podien llegir de forma seqüencial i ordenadament.

Aquestes cintes, emmagatzemaven fitxers amb registres que es processaven seqüencialment juntament amb fitxers de moviments per generar nous fitxers actualitzats.

Aquestes es coneixien coma aplicacions basades en sistemes de fitxers i constitueixen la generació zero dels sistemes de les base de dades, ja que llavors no existia ni el concepte de base de dades.



Unitat de Cinta IBM 2420

La idea era mantenir un fitxer mestre, per exemple les comptes bancàries d'un banc i al llarg del dia acumular els canvis i moviments que s'ocasionaven a les comptes a un segon fitxer (fill). A la nit es tractava el fitxer de moviments diaris juntament amb el fitxer mestre per obtenir un tercer i nou fitxer mestre amb les dades actualitzades.

A banda de la problemàtica de la lectura seqüencial, trobem altres inconvenients:

- Afegir nous atributs
- Duplicitat de la informació ja que cada programa mantenia el seu propi conjunt de dades, malbaratament de l'espai ja que hi ha moltes dades duplicades.
- Dependència de les dades (L'estructura de les dades venia definida dins el programa).
- Formats incompatibles: Programes escrits en diferents llenguatges era molt difícil accedir a fitxer d'altres programes.

- Consultes predefinides: Els programes satisfeien certes consultes. Qualsevol nou requisit o consulta necessitava d'un nou programa.

1.2. Dècada dels 60s

Es generalitza l'ús dels discs magnètics, a on la seva característica principal és que es podia accedir de forma directa a qualsevol part dels fitxers, sense haver d'accedir a totes les dades anteriors. Es podia accedir a un registre determinat d'un fitxer sense necessitat de llegir prèviament tots els registres anteriors.

Perquè un programa sabés en quina adreça física del disc es trobava una informació concreta (per exemple la compte número 1537), en el moment de l'escriptura ens hauríem guardat la direcció física a on ha estat gravada aquell número de compte per llavors accedir-hi directament.

S'utilitzen sistemes centralitzats amb un gran ordinador i terminals "*tontos*" per interaccionar-hi. Sistemes basats en el model de dades jeràrquic: IMS (Information Management System).

El 1961 **Charles Bachman** va dissenyar el primer SGBD generalitzat. El magatzem de dades integrats (Integrated Data Store, **IDS**) de General Electric.


Charles Bachman

La idea fundacional de les Base de Dades (en realitat els SGBD) era solucionar tots els problemes derivats de l'ús aïllat dels diferents fitxers.

Un SGBD havia de, en primer lloc, assegurar la coherència de les dades en tot moment. Un SGBD havia de complir les següents característiques (més endavant ja especificarem quines funcions, components i objectius han de complir els SGBD):

- Davant d'una fallada de programa, ha de ser capaç de recuperar la informació.

- Ha de gestionar de forma eficaç la concurrència entre processos. Això exigeix d'un control automàtic del bloquejos: Quan un programa accedeix a un registre de la Base de Dades amb la intenció de modificar-lo, el SGBD ha de deixar-lo bloquejat perquè qualsevol altre procés no interfereixi en aquest canvi.

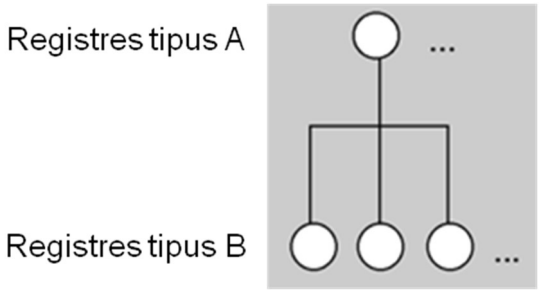
- Abstracció del model físic. Es dissenya el model lògic sense tenir en compte com aquest guarda la informació físicament.

Gràcies els esforços de Bachman, a l'octubre de 1969 es concep el primer model de dades en xarxa, conegut com **CODASYL** (Conference On Data Systems Language), que posteriorment **IBM** va refinar i millorar desenvolupant el seu Sistema de Gestió d'Informació (Information Management System, **IMS**) pel programa Apollo de la NASA.

Amb aquesta tecnologia apareixen les base de dades **jeràrquiques** i en **xarxa**, que aprofitaven les capacitat d'accés directa a la informació dels discs magnètics per estructurar la informació en forma de llistes enllaçades i arbres d'informació.

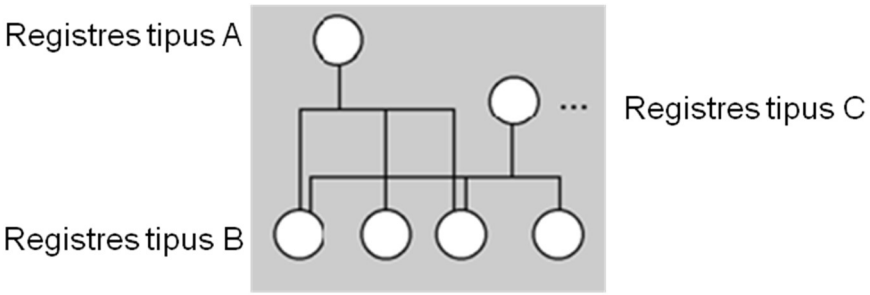
Model Jeràrquic

Interrelacions en forma d'arbre



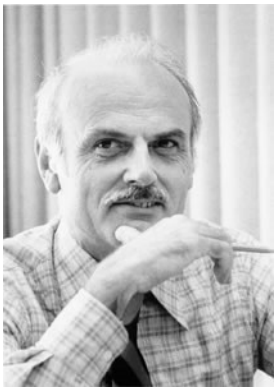
Model en Xarxa

Interrelacions en forma de graf



1.3. Dècada dels 70s

Els anys 70 és a on la tecnologia de base de dades experimenta un ràpid creixement.



Edgar Frank Codd (Wiki en, Wiki cat), científic informàtic anglès de IBM, conegut per les seves aportacions a la teoria de bases de dades relacionals, defineix el model relacional a publicant un article anomenat "Un model relacional de dades

per grans grans bancs de dades compartits (*A Relationa Model of Data for Large Shared Data Banks* . https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf). Aquest fet va donar pas al naixement de la segona generació dels SGBD.

Gràcies al treball de Codd durant la dècada de 1970, Lawrence J. Ellison, conegut per **Larry Ellison**, va desenvolupar el *Relational Software System*. El que actualment es coneix com Oracle Corporation (~1982), desenvolupant així un SGBD relacional amb el mateix nom de la companyia.

Avui en dia, el model relacional de Codd, tot i les existents variants i alternatives, segueix essent el més utilitzat a tots els nivells i la seva potent base matemàtica del model va ser la clau del seu èxit.

Codd va definir un conjunt de 13 regles/lleis (0-12) per establir les característiques que ha de tenir una base de dades relacional. Actualment, tots els SGBD implementen aquestes regles (Wiki cast).

Tal i com hem dit durant aquesta dècada i gràcies a Codd es van crear els dos SGBD: *Relational Software System* i *System R*, creat per IBM i utilitzant el llenguatge de consultes SEQUEL essent el precursor del llenguatge SQL.

Al voltant del 1976 el doctor Peter.Chen (Wiki en) va proposar un model conceptual anomenat Entitat-Relació (*Entity-Relationship* / ER) per tal de dissenyar Base de Dades.

1.4. Dècada dels 80s

Durant la dècada dels 80 IBM llança el seu motor de base de dades DB2 basant-se amb el seu System R.

Encara que el sistema relacional s'inventés en la dècada anterior, és en aquesta a on agafa més volada i es postula com a principal model de dades per moltes empreses utilitzant els SGBD relacionals.

Una anys més tard, IBM crea el llenguatge SQL (Structured Query Language), un potent llenguatge de consultes per manipular la informació de base de dades relacionals.

El 1986, l'Institut Nacional Nordamericà de Normalització (American National Standards Institute, o ANSI, en anglès) va publicar les primeres normes que enunciaven la sintaxi i la semàntica de l'SQL i aquest es converteix en el llenguatge estàndard de les BD relacionals.

1.5. Dècada dels 90s

Apareixen els primers prototips de SGBD orientats a objectes (Object Database Management Systems, ODBMS

A mitjans del 90s, IBM treu una versió de DB2 capaç de dividir una base de dades enorme en varis servidors comunicats per línies de gran velocitat, creant d'aquesta manera les base de dades parel·leles.

A finals de dècada IBM i Oracle incorporen a les seves BD la capacitat de manipular objecte, creant així, les base de dades orientades a objectes. Aquestes base de dades orientades a objectes es basen en la existència d'objectes persistents que s'emmagatzemen per el seu processament mitjançant programes orientats a objectes. En comptes d'emmagatzemar relacions i taules.

Com ja sabem, els primers sistemes de BD eren centralitzats: totes les dades del sistema estaven emmagatzemades en un únic gran ordinador al qual es podia accedir des de diferents terminals. Però l'èxit gradual dels ordinadors personals (PCs), cada vegada més potents i amb preus més competitius, juntament amb el desenvolupament de les xarxes, va possibilitar la distribució d'una mateixa BD en diferents ordinadors (o nodes).

Això fa que neixin les bases de dades distribuïdes, que consisteixen en multiplicar el número d'ordinadors (anomenats nodes) que controlen una base de dades, intercanviant informació i actualitzacions a través de la xarxa.

La tecnologia utilitzada habitualment en la distribució de BD és l'arquitectura client/servidor (coneguda també com a arquitectura C/S). Actualment, tots els SGBD comercials estan adaptats a aquesta realitat.

Finalment, durant els anys noranta, la implantació arreu de les BD, fins i tot en petits sistemes personals, va motivar l'aparició dels anomenats llenguatges de quarta generació (fourth generation languages, o 4GL, en anglès), els quals es continuen utilitzant en l'actualitat.

Exemples de BD orientades a objectes: db4o (Versant) o Gemstone (VMWare), Twig (Google), Pers (McObject)

1.6. En l'actualitat

La gran quantitat d'informació emmagatzemada durant aquest anys ha fet necessària l'aparició d'un nou software anomenat software per l'ajuda de presa de decisions juntament amb base de dades multidimensionals, formant el que s'anomenen cubs d'informació.

Avui en dia, la majoria de SGBD professionals incorporen els recursos necessaris per donar suport als servidors de pàgines web dinàmiques (és a dir, amb accés a les dades contingudes en un SGBD allotjat en el servidor web corresponent).

Una altra línia d'innovació que segueixen alguns SGBD és el treball amb els anomenats magatzems de dades (data warehouse, en anglès). Aquests magatzems consisteixen en rèpliques elaborades de les dades generades pel funcionament quotidià de l'organització o l'empresa de què es tracti, durant un cert període de temps per tal de realitzar anàlisis estratègiques d'índole financera, de mercats, etc.

El llenguatge de marques extensible (extensible markup language, o XML, en anglès) també influeix en el món dels SGBD, tot i que inicialment no es va concebre

com una tecnologia per donar servei a les BD, sinó per estructurar documents molt grans. Però aquesta capacitat per emmagatzemar les dades de què es compon un document el fan susceptible de ser utilitzat, també, en l'àmbit de les BD.

Actualment s'han popularitzat molt uns nous SGBD etiquetats com a NoSQL pel simple fet de no ser relacionals. El fet que portem quasi 30 anys amb la mateixa tecnologia ens ha fet estandarditzar un model i veure els altres com a diferents.

Alguns exemples són:

Base de dades espaials o geogràfiques: base de dades que emmagatzemen mapes i símbols que representen superfícies geogràfiques. Tot i que igual que amb XML molts del productes relacionals incorporen aquestes característiques.

Base de dades documentals: Permeten la indexació de text per poder realitzar cerques complexes en textos de gran longitud.

Base de dades deductives: Són sistemes de base de dades que emmagatzemen fets i que permeten, a través de procediments d'inferència, extreure nous fets. Es basen en la lògica deductiva.

Models de dades

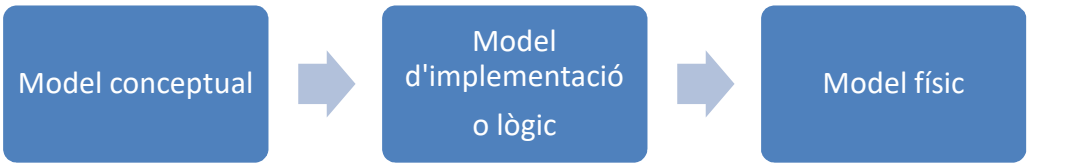
Una BD és una representació de la realitat. És a dir, un model de la realitat. Nosaltres volem transformar la realitat i plasmar-la en un entorn el qual el puguem tractar manipular i gestionar.

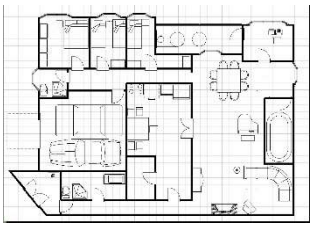
Col·lecció de conceptes que podeu utilitzar per descriure l'estructura d'una base de dades: tipus de dades, relacions, restriccions, etc...

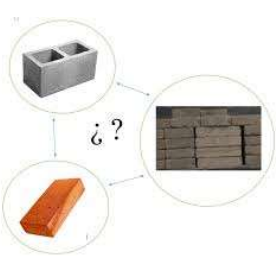
Per exemple en una base de dades d'una universitat tenim els conceptes de professor, assignatura, alumne, i aquests s'interrelacionen entre ells. A més hi ha restriccions del tipus un alumne de primer no es pot matricular d'assignatures de últim curs, etc...

Tipus de models de dades: Els tipus de dades són les diferents maners de representar la realitat que estem tractant.

- **Model de dades conceptual:** model de representació d'alt nivell capaç de representar part de la realitat i sense dependre de cap tipus de llenguatge. Per exemple ER/ERE, UML, etc...
- **Model de dades d'implementació / lògic**: Conceptes que els usuaris poden entendre, però no estan massa allunyats de l'organització de dades físiques. S'amaguen alguns detalls d'emmagatzematge. Per exemple: model jeràrquic, model de xarxa, model relacional, model relacional amb objectes,..
    - Per exemple, el component fonamental per a modelar en un SGBD relacional són les taules, però en altres tipus de SGBD s'utilitzen altres components.
- **Model de dades físic**: Conceptes a baix nivell que descriuen en detall de quina manera s'emmagatzemen les dades.







Els quatre models més utilitzats en els SI és el model relacional o el model orientat a objectes.

Tot tipus de model de BD ens proporciona tres tipus d'eines:

1. Estructures de dades amb les quals es pot construir la BD: taules, arbres, etc..

2. Diferents tipus de restriccions (o regles) d'integritat que el SGBD haurà de fer complir a les dades: dominis, claus, etc..

3. Tot un seguit d'operacions per a treballar amb les dades. Per exemple, en el model relacional l'operació SELECT per seleccionar o llegir files que compleixen una condició. Un exemple d'operació dels models jeràrquic i en xarxa podria ser la que ens diu si un determinat registre té fills o no.

1.6.1. Model jeràrquic

Tal i com hem explicat en l'apartat d'història el primer model de BD utilitzat va aparèixer a principis dels anys seixanta i va ser el model jeràrquic.

Les seves estructures són registres interrelacionats en forma d'arbres. El SGBD clàssic d'aquest model és l'IMS/DL1 d'IBM.

Des de fa molt de temps s'ha utilitzat l'organització de la informació en jerarquies per entendre millor el món, per exemple trobem esquemes de classificació per les espècies animals i vegetals, classificacions dels llenguatges humans.

No existeix cap document original que descrigui el model jeràrquic, com passa amb el model en xarxa i el relacional. Sinó que els primers SGBD utilitzaven estructures d'emmagatzematge jeràrquiques.

No és l'objectiu d'aquest document abordar tot el model jeràrquic, sinó comprendre'n el seu model i veure'n unes pinzellades. Podeu trobar més informació sobre aquest model i el sistema IMS en la bibliografia proporcionada.

El model jeràrquic va ser desenvolupat per permetre la representació de situacions de la vida real a on predominen les relacions de tipus 1:N (Un curs(1) hi ha diferent alumnes(N))

Aquest model s'organitza per múltiples nivells d'acord amb la relació (PARE/FILL) de manera que un pare pot tenir més d'un fill i aquests estan dins el mateix nivell i un fill només pot tenir un pare situat al nivell superior.

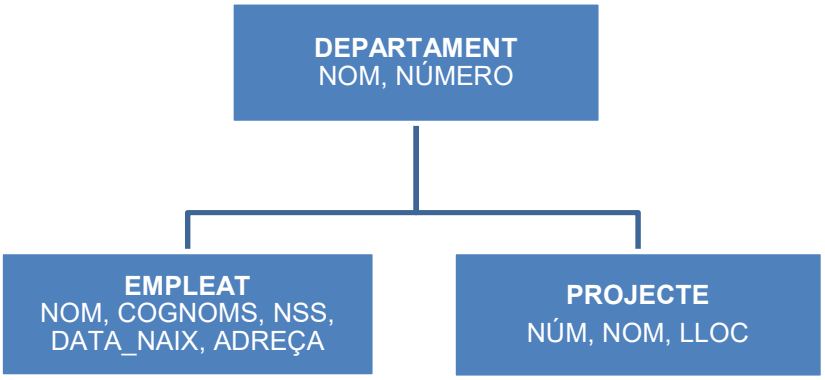
El model jeràrquic utilitza dos conceptes principals per l'estructuració de les dades:

- **Registres:** Un registre és una col·lecció de valors de camps que proporcionen informació respecte una entitat. Els registres del mateix tipus s'agrupen en segments o tipus de registres. Cada segment o tipus de registre rep un nom.

- **Relacions pare-fill:** El tipus de relació pare-fill és una relació 1:N entre dos tipus de registre diferents. El tipus de reigstre del costat 1 s'anomena pare i el del costat N fill. Una instància(exemplar) del tipus de relació consisteix en un registre del pare i diferents (zero o més) del fill.

Un esquema de base de dades jeràrquic consisteix en diferents esquemes jeràrquics.

La representació gràfica d'un model jeràrquic es realitza mitjançant una estructura d'arbre invertit. A on en el nivell superior hi ha una única entitat de la que en depenen totes les altres.



Una **instància** d'un segment d'una base de dades jeràrquica és el conjunts de valors particulars que agafen tots els camps que el composen en un moment determinat.

La relació **pare-fill** amb el que es base el model de base de dades jeràrquic determina el camí d'accés a les dades i aquest és únic, anomenat camí de **seqüència jeràrquica**.

Les relacions pare-fill no tenen nom, tot i que el dissenyador associa un cert significat a cada relació.

Propietats d'un esquema jeràrquic:

- Existència d'un tipus de registre **arrel.** Aquest no participa com a tipus de registre fill en cap tipus de relació pare-fill.
- Tot tipus de registre, amb excepció de l'arrel, participa com tipus de registre fill en un o diferents tipus de relació pare-fill.
- Un tipus de registre pot participar com tipus de registre pare en moltes tipus de relació pare-fill
- Un tipus de registre que no participa com tipus de registre pare en cap tipus de relació pare-fill s'anomena **fulla**
- Si un tipus de registre participa com a pare en més d'un tipus de relació pare-fill, aleshores els seus tipus de registres fill estan ordenat. L'orde es visualitza, per convenció d'esquerra a dreta en els diagrames.

Exemple:

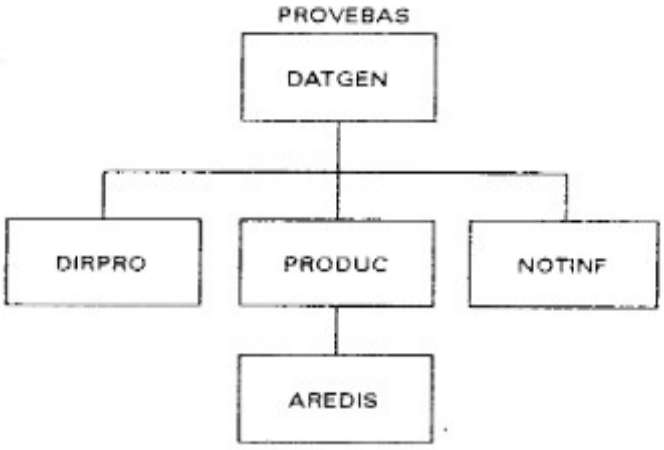
Imaginem una empresa d'àmbit nacional amb delegacions per tot el país. Aquesta empresa té centralitzades totes les compres de material de les seves delegacions a la oficina central i disposa d'una base de dades jeràrquica per permet emmagatzemar les dades de tots els seus proveïdors.

BD anomenada PROVEBAS, presenta 5 segments.

El segment arrel a on es guarden les dades comunes a tots els proveïdors, el nom, el director, el NIF, etc... Aquest segment l'anomenarem **DATGEN**.

El segon nivell hi ha 3 segments dependents del segment arrel:

- **DIRPRO**: Adreces de les sucursals de l'empresa proveïdora indicant carrer, número, ciutat, etc..
- **PRODUC**: Dades dels productes subministrats per cada una de les empreses proveïdores
- **NOTINF**: Permet guardar les diferents notes informatives que va generant un proveïdor



En el tercer nivell de l'arbre hi trobem un sol segment, anomenat **AREDIS**, que permet emmagatzemar les zones de distribució de cada un dels productes subministrats pels diferents proveïdors. Aquest segment depèn del segment PRODUC.

Preguntes:

- Què passa si diferents proveïdors ens proveeixen del mateix producte?
- Com podem fer un llistat per les diferents zones de distribució?
- Com representem les relacions N:M. Per exemple entre professors i alumnes? Un alumne té diferents professors i un professor té diferents alumnes?

Les actualitzacions en les base de dades jeràrquiques també poden originar problemes.

Una alta d'una instància d'un segment ha de tenir un pare. Per exemple en el cas anterior, no podíem afegir un nova zona de distribució si aquesta no té assignada un producte.

Una baixa d'una instància implica l'esborrat de totes les instàncies (subarbre) que en depenen.

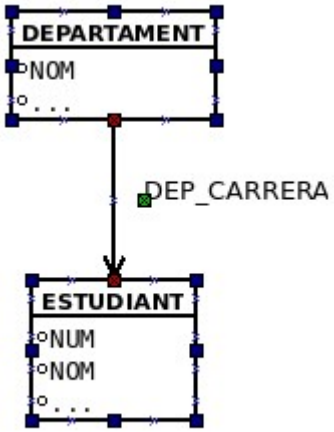
Presenten problemes alhora d'afegir restriccions d'usuari. Per exemple no podem limitar l'ús d'una instància d'un segment per un usuari sense afectar als nodes fills.

1.6.2. Model en xarxa

Des del punt de vista històric les estructures i construccions del llenguatge per el model en xarxa van ser definides per el comite CODASYL, per això moltes vegades també s'anomena model en xarxa CODASYL.

El model de dades en xarxa representa les entitats en forma de nodes d'un graf i les relacions entre aquestes mitjançant arcs que uneixen els diferents nodes.

Amb aquest model es millorava l'anterior de tal manera que es donava una solució més eficient al problema de redundància de dades, però tot i així la dificultat d'administrar la informació d'una base de dades en xarxa ha significa que els seus usuaris siguin purament tècnics.



En principi aquesta representació no imposa cap limitació en el tipus i el nombre d'arcs.

De la mateixa manera que en el model jeràrquic, les dades s'emmagatzemen en registres a on cada registre es classifica en **tipus de registre** i aquests tenen una sèrie d'**atributs** o elements d'informació del tipus de registre i aquests es relacionen entre si amb el que s'anomenen **tipus de conjunts** (relacions 1:N).

Els atributs poden ser element d'informació reals o elements d'informació virtuals o derivats. Els primer és informació que s'emmagatzema realment en els registres, però els derivats so s'emmagatzemen en el registre i s'obtenen a partir d'algun procediment que el calculi a través dels element reals. Per exemple, un element d'informació virtual edad per el tipus de registre estudiant s'haurà d'escriure un procediment que calculi el valor a partir de l'element d'informació data_naixement.

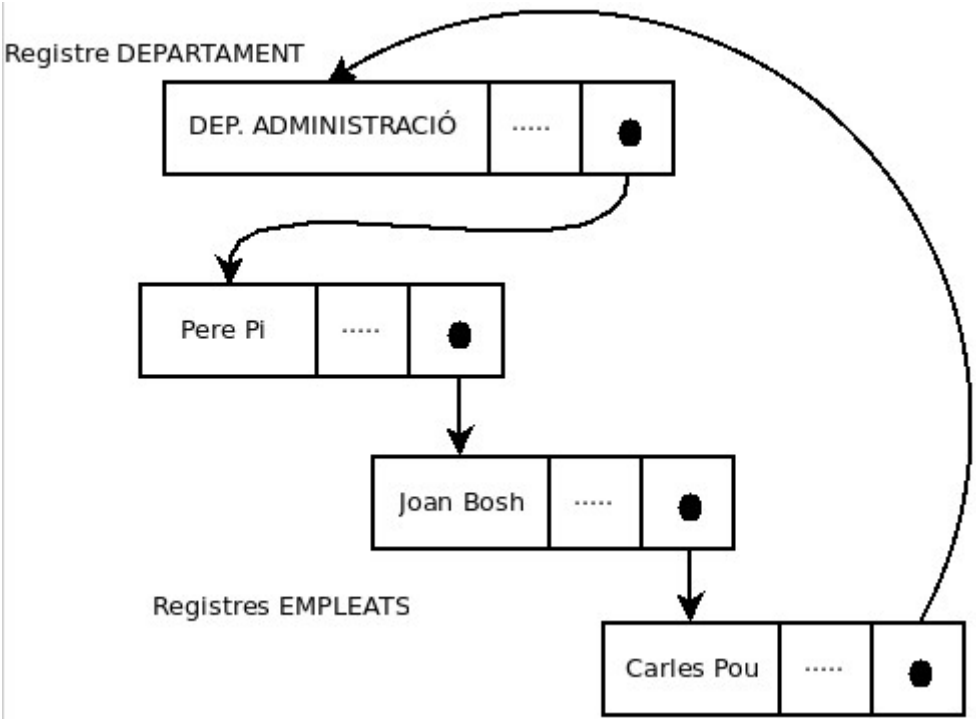
Tipus de conjunts i propietats bàsiques.

Un tipus de conjunts és una descripció d'una relació 1:N entre dos tipus de registres. Cada definició de tipus de conjunts consta de 3 elements bàsics:

- Un nom per el tipus de conjunt: en l'exemple DEP_CARRERA

- Un tipus de registre propietari: en l'exemple DEPARTAMENT
- Un tipus de registre membre: en l'exemple ESTUDIANT

Cada instància de conjunt es composa d'un registre propietari del tipus de registre propietari i varis registres membre relacionats del tipus de registre membre. D'aquesta manera la restricció de que un tipus des conjunt representa una relació 1:N.



1.6.3. Model relacional

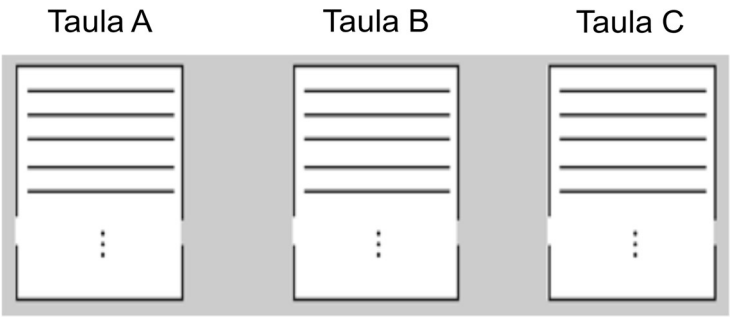
Una altra gran diferència important entre els models prerelacionals i el model relacional és que aquest es limita al nivell lògic, no fa absolutament cap consideració sobre les representacions físiques i d'aquesta forma aconsegueix una independència física de dades total.

Tal i com hem dit anteriorment aquest model va ser introduït per Codd (1970).

Formalment el model relacional representa la base de dades com una col·lecció de relacions i aquestes relacions contenen tuples organitzades per atributs. El tipus de dades que descriu els tipus de valors que pot aparèixer en les diferents tuples s'anomena domini.

Així com els models prerelacionals (jeràrquic i en xarxa) les estructures de dades consten de dos elements bàsics, els registres i les interrelacions, en el model relacional la forma informal d'un sol element, la **taula**, formada per **files** i **columnes**. Les **interrelacions** s'han de modelitzar utilitzant les taules.

La forma formal només l'utilitzarem quan estiguem parlant del model relacional de forma molt purista o puntual i passarem a utilitzar la nomenclatura de taula, fila i columna, quan estiguem en la vida quotidiana.



La major part dels SGBD actuals utilitzen encara aquest model dissenyat per Cood. Moltes d'elles incorporen nous elements i funcionalitat, però la idea bàsica és el model relacional.

Encara que trobem altres SGBD que tenen diferents funcionalitats i característiques són els SGBD que tenen més presència en el mercat i funcionen en la majoria de sistemes d'informació actuals de moltes de les empreses arreu del món.

1.6.4. Altres models

Els últims anys s'han estès el model de BD relacional amb objectes. Es tracta d'ampliar el model relacional, afegint-hi la possibilitat que els tipus de dades siguin tipus abstractes de dades, TAD.

Des de fa poc, han entrat en joc els SGBD anomenats noSQL considerades la pròxima generació de SGBD amb els següents punts característics: no relacionals, distribuïdes, no tenir un esquema fix, open-source (algunes), fàcilment escalables de forma horitzontal.

Per què NoSQL?

La majoria dels SGBD clàssics es basen en transaccions per tal de garantir la integritat de les dades. Això assegura la consistència de les dades en totes les situacions de gestió de les dades de forma concurrent.

Aquestes característiques transaccionals també conegudes com ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability) fan que no sigui fàcil ampliar-los i escalar-los tal com es demostra en el teorema CAP.

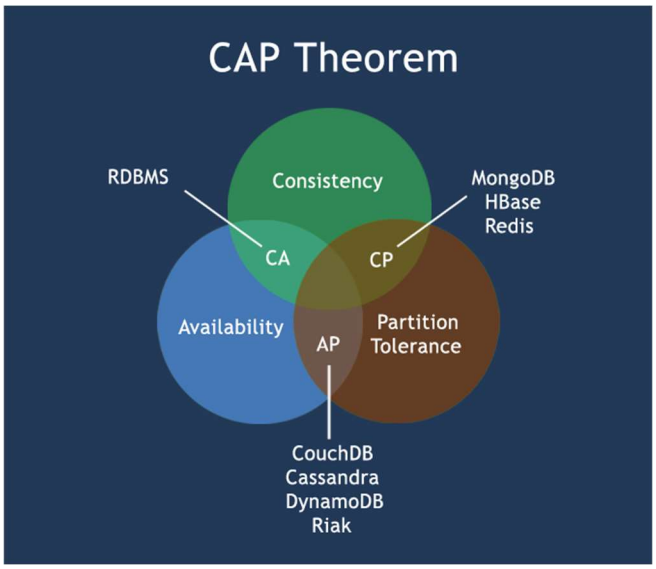
Teorema de CAP

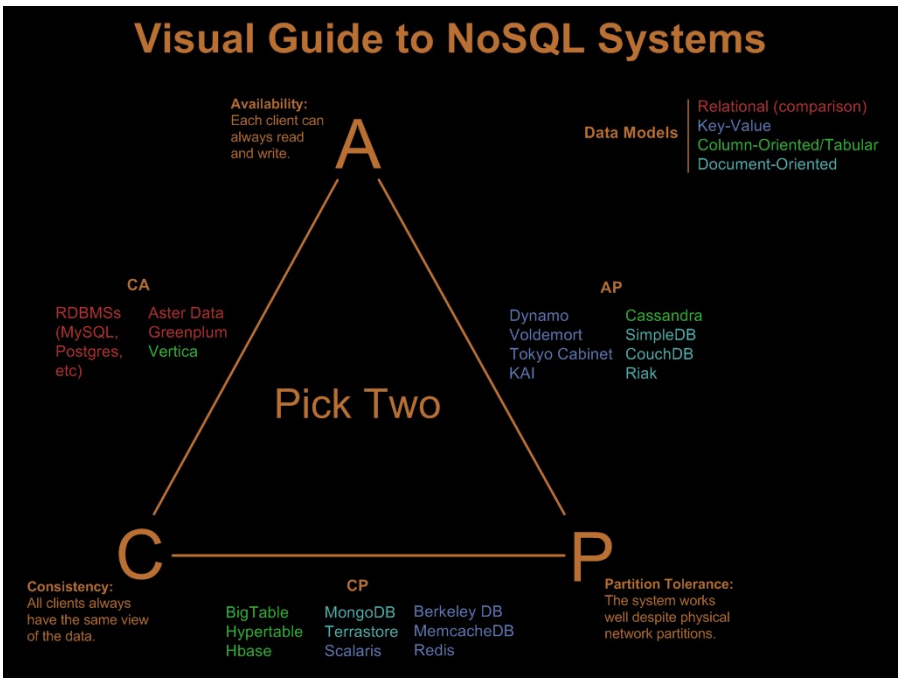
https://es.wikipedia.org/wiki/Teorema_CAP

https://www.youtube.com/watch?v=Jw1iFr4v58M

En informàtica teòrica, el teorema CAP (**C**onsistency, **A**vailability, **P**artition Tolerance), també conegut com a teorema de Brewer (Eric Brewer), formula que és impossible garantir **simultàniament** les tres característiques següents en una aplicació distribuïda:

- **Consistència**: Tots els nodes veuen la mateixa dada al mateix temps.
- **Disponibilitat**: La garantia que cada petició a un node rep una resposta de si ha tingut èxit o a fallat.
- **Tolerància a la partició**: El sistema continua operant malgrat la partició arbitrària a causa d'errors en la xarxa i això és totalment transparent pels clients





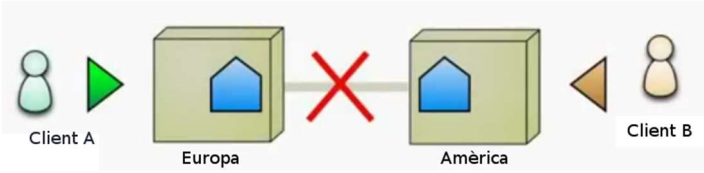
Molts del SGBD NoSQL, s'han preocupat més per tenir una alta disponibilitat i partionat disminuint les seves característiques de Consistència/Coherència.

Un dels principals objectius dels sistemes NoSQL és l'escalabilitat horitzontal (augmentar el número de màquines) i per fer això es necessita d'una forta

tolerància de partició de xarxa que requereix renuncia a qualsevol consistència o disponibilitat.

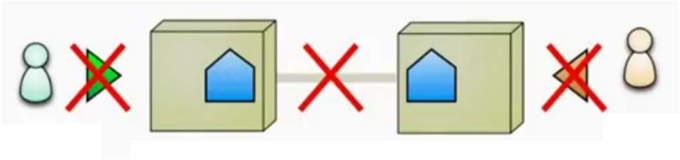
Per exemple imaginem que tenim una aplicació distribuïda de reserves d'apartaments .

Per millorar el rendiment tenim la base de dades distribuïda en dos continents: Europa i Amèrica.

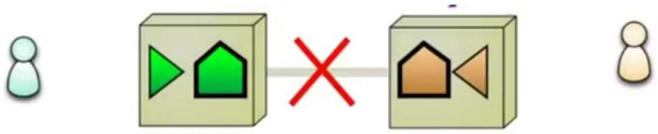


Imaginem que la comunicació entre Europa i Amèrica falla. Què fem amb els clients que volen fer una reserva en el mateix apartament? Els deixem fer reserves o no? Què premiem?

- **Consistència**: No deixem fer la reserva, perquè no sabem si Europa o Amèrica tenen una reserva per aquell apartament concret.

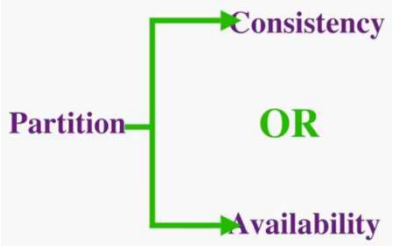


- **Disponibilitat**: Deixem fer la reserva i ja arreglarem els possibles problemes de consistència més endavant.



Important: ÉS EL NEGOCI/CLIENT QUE DECIDIRÀ QUÈ VOL FER!!!

Això és el que diu el teorema de CAP. Si estem davant d'un problema de xarxa (Partition tolerance). Hauem d'escollir consistència (Consistency) o disponibilitat (Availability)



Per exemple si tenim una base de dades funcionant en un sol servidor no tindrem problemes de *partition network.*

Dit això podem fer unes certes consideracions de quan cal utilitzar un sistema o un altre:

Quan utilitzar NoSQL

- Volem implementar a gran escala una base de dades d'alta concurrència (cents de GB, milers d'usuaris).
- No requerim de les regles ACID
- No requerim de relacions i restriccions a l'esquema de dades
- Quan utilitzem poques taules (5-10 taules en el model relacional).
- Quan volem implementar alta disponibilitat amb costos baixos.

Aquests sistemes són bons en molts dels llocs web (web sites). Per exemple, Google, Twitter Creieu que és realment important si es retrassen uns segons alguns tweets o fins i tot que es perdin? I si Google no mostra la informació de resultats de l'última pàgina web que s'ha creat o ell ha escanejat? O per exemple si els resultats de Google perd algun enllaç pel camí?

Quan utilitzar Relacional

- Estem al davant d'una mitjana o gran escala de base de dades (10-100 GB) amb poca concurrència (cents d'usuaris com a màxim);
- Quan necessitem de ACID, especialment l'**A**tomicitat i la **C**onsistència;
- Les dades tenen moltes relacions entre elles (jerarquies, mestre-esclau, etc..);
- Quan volem emmagatzemar una àmplia varietat de dades en cents o milers de taules normalitzades.
- Ejecutarse en hardware de gama alta;
- Las porciones de capital disponible..

Els sistemes relacions estan dissenyats per ser escalats verticalment i de bon inici necessiten màquines amb molta quantitat de memòria, ràpid I/O through mitjançant SANs i SSDs. Se escalades ocasionalment a través e clustering i partició de els dades

Si considerem el Teorema CAP per classificar les base de dades NoSQL trobem 4 tipus de SGBD:

- **Orientats a documents**
    - Són SGBD que gestionen dades semi estructurades. És a dir documents. Aquest documents normalment són formats estàndard (XML, JSON, ...)

- Exemples: MongoDB, CouchDB. (JSON document)

- **Orientats a columnes**
  - Aquest tipus de base de dades estan pensades per realitzar consultes i agregacions sobre grans quantitats de dades. El seu model de dades és similar al de les BD relacionals, però emmagatzemen columnes de dades en comptes de registres.
  - Exemples:HBase, Cassandra(aquesta és un híbrid entre orientada a columnes i clau-valor).

- **Clau-Valor**
  - Aquestes són les més senzilles d'entendre. Simplement guarden una dada (string, enter, JSON) mitjançant una clau única. Cada vegada que volem fer referència aquella dada ho hem de fer mitjançant aquesta clau.
  - Exemples: DynamoDB, Redis, Memcaché (només a memòria principal i no es consideraria SGBD).

- **Graf**
  - Basades en la teoria de grafs utilitzant nodes i arestes per representar les dades emmagatzemades. Són molt útils per guarda informació en models amb moltes relacions, com xarxes, i connexions socials. Fan pensar amb els models de xarxa.
  - Exemples: Infinite Graph, Neo4j, MongoDB

Important: Cada SGBD té un paper/rol dins del nostre sistema d'informació o aplicatiu. Cal escollir el tipus de SGBD que millor s'adapti allà que volem guardar/fer:

Sistemes Gestors de Base de Dades

1.1. Objectius i funcions d'un SGBD

D'ençà que Charles Bachman va dissenyar el primer SGBD generalitzat intentant solucionar els problemes derivats per l'ús de fitxers hem anat incorporant nous requisits i funcions que volem que facin els nostres SGBD. Actualment quasi tots els SGBD del mercat compleixen amb els següents objectius:

- **Independència física i lògica de les dades:**
  - Els usuaris han de poder emmagatzemar dades, accedir i actualitzar-les de forma senzilla i amb un gran rendiment ocultant la complexitat i les característiques físiques dels dispositius d'emmagatzematge.
  - Donar flexibilitat als canvis en l'estructura de les dades. Qualsevol canvi en l'esquema de la BD no ha d'afectar als usuaris.
  - Poder realitzar consultes no predefinides i complexes sense necessitat de saber de quina manera s'emmagatzema la informació físicament.
    - Per exemple: Obtenir el llistat dels alumnes d'una escola que enguany cursin 1r i 2n curs.

- **Garantir la Integritat de les dades:** Les dades no poden ser alterades per errades humanes o errades del maquinari. El sistema no ha de permetre operacions que deixin cert conjunt de dades incompletes o incorrectes.

- **Evitar problemes de redundància:** Algú pot pensar que el problema de la redundància recau en l'espai perdut pel fet de tenir dades repetides. Això podia ser un problema anteriorment quan el cost d'emmagatzematge d'un byte era elevat, però ara mai no ho és. El veritable problema de la redundància és amb el fet de tenir dues dades en dos llocs diferents, això comporta molts problemes alhora de realitzar les actualitzacions fins que dur problemes de coherència entre les dues dades. No se sap quina és la bona. En resum pèrdua d'integritat de la informació.

- **Control de concurrència i simultaneïtat:** Un dels objectius fonamentals per un SGBD és permetre que diversos usuaris puguin concurrentment a la mateixa BD. Proporcionar mecanismes que permetin arbitrar operacions conflictives en l'accés o modificació d'una dada al mateix temps per part de diferents usuaris.

- **Ús de transaccions:** Relacionat amb el punt anterior s'ha de poder permetre l'ús d'operacions transaccionals. És a dir que es realitzin correctament o en cas que hi hagi alguna operació fallida es desfacin els canvis realitzats des de l'inici de la transacció. Per exemple: la transferència bancària entre dos comptes.

- **Reserva i seguretat:**. Incorporació de mecanismes que garanteixin l'accés a la informació sigui exclusivament a aquells usuaris que disposin de l'autorització pertinent. Permetre definir autoritzacions d'accés i nivells de seguretat per cada tipus d'usuari

- **Còpies de seguretat i recuperació.** Incorporació d'eines que facilitin la còpia i restauració de la informació en cas de desastres.

- **Oferir connectivitat amb l'exterior:** D'aquesta manera, es poden replicar i distribuir base de dades.

- **Oferir eines estadístiques:** El gestor ha de guardar informació estadística de l'ús del mateix: registrant operacions efectuades, consultes sol·licitades, operacions fallades i qualsevol tipus d'incidència. D'aquesta manera poder monitoritzar l'ús del sistema i permetre detectar hipotètics malfuncionaments.

Hi ha 3 característiques importants d'una base de dades dins d'un SGBD:

1) L'ús d'un catàleg per emmagatzema la descripció (esquema) de la base de dades.

Una base de dades no només ha de contenir les dades que volem guardar sinó també la descripció complerta de la base de dades, l'estructura de la base de dades (*database schema*).

Aquesta definició s'emmagatzema en un catàleg del sistema que conté informació com l'estructura de cada fitxer, el tipus i format d'emmagatzematge de cada element de la informació i diverses restriccions que s'apliquen a les dades.

Important: Cal distingir entre la descripció de les dades i les dades en sí.

2) Separació entre els programes i les dades

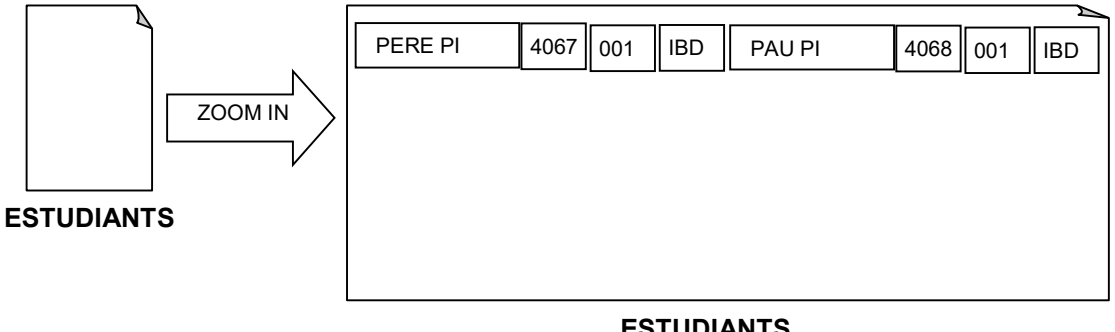
En els processament dels fitxers tradicionals, l'estructura dels fitxers de dades venia integrada amb els programes d'accés. D'aquesta manera es creava una forta interdependència entre el programa i la base de dades. Qualsevol modificació de l'estructura d'un fitxer de dades normalment produïa una forta modificació en tots els programes que tenien accés a aquell fitxer.

Amb els SGBD volem aconseguir una independència respecte el programa i les dades.

Per exemple:

Un programa tenia definits la mida en bytes dels de les dades i si es feia un canvi en un camp això comportava la modificació de moltes parts del programa. Un programa que tenia accés als registres d'un ESTUDIANT de 42 caràcters de longitud.

Nom	Posició inicial del registre	Longitud en caràcters (bytes)
NomEstudiant	1	30
NúmEstudiant	31	4
Curs	35	4
Assignatura	39	4



ESTUDIANTS

ESTUDIANTS

Si volíem afegir una altra dada a cada registre de l'ESTUDIANT, per exemple la data de naixement, el programa no podria funcionar i s'hauria de modificar. En canvi en un entorn de SGBD es modificarà la descripció dels registres ESTUDIANTS en el catàleg. La pròxima vegada que un programa del SGBD consulti el catàleg, tindrà accés a la nova estructura de registres.

Exemple de cerca de les dades d'una persona d'un nom concret.

File-based (pseudo-code)

```
find person (name:input, record:output)
begin
open people
repeat while people has next
 people → go to next record
 record := people → current record
 if record → person is name
       done
 else
       continue
 end
end
record := invalid
end
```

Database (SQL)

```
SELECT *
FROM people
WHERE
name = $NAME
```

3) Manipulació des de diferents vistes d'usuari

Una base de dades sol tenir molts usuaris, cada un dels quals pot requerir d'una perspectiva o vista diferent de la mencionada base de dades. Una vista pot ser un subconjunt de la base de dades o contenir dades virtuals (derivades d'altres dades). És possible que alguns usuaris no necessitin saber si les dades que estan manipulant són dades derivades o no.

Per exemple un usuari de secretaria no té perquè saber la nota de l'estudiant però si si l'alumne ha pagat el curs o no. En canvi l'usuari professor no té perquè saber la data en la qual ha pagat la matrícula l'alumne, però en canvi ha de poder posar una nota a l'alumne per l'assignatura el qual li pertany.

Els SGBD necessiten que els donem una descripció o definició de la BD, descripció que rep el nom d'esquema de la BD, i que els SGBD tindran contínuament a l'abast.

L'esquema de la BD és un element fonamental de l'arquitectura d'un SGBD que permet independitzar el SGBD de la BD. Canviar el disseny de la BD, el seu esquema, sense d'haver de fer cap canvi en el SGBD.

1.2. Arquitectura dels SGBD

Tal i com hem dit anteriorment i de forma molt resumida podem definir un Sistema Gestor de Base de Dades (SGBD), com un conjunt d'eines (programari) que faciliten la consulta, ús i actualització (manteniment) d'una o vàries base de dades.

1.2.1. L'arquitectura de 3 esquemes

El comitè conegut com a ANSI/SPARC va marcar la referència per la construcció d'un SGBB i va recomanar que l'arquitectura d'un SGBD preveiés 3 nivells d'esquemes, 3 nivells d'abstracció.

ANSI (***A**merican **N**ational **S**cience **I**nstitute*) és un organisme científic d'Estats Units que ha definit diferents estàndards. Link

X3 és la part de l'ANSI encarregada dels estàndards en el món de la electrònica

SPARC (***S**ystem **P**lanning and **R**epairments **C**ommittee*) és el comité de planificació de sistemes i reparacions. És una subsecció de X3 encarregada dels estàndards en Sistemes Informàtics, en especial en el camp de les base de dades.

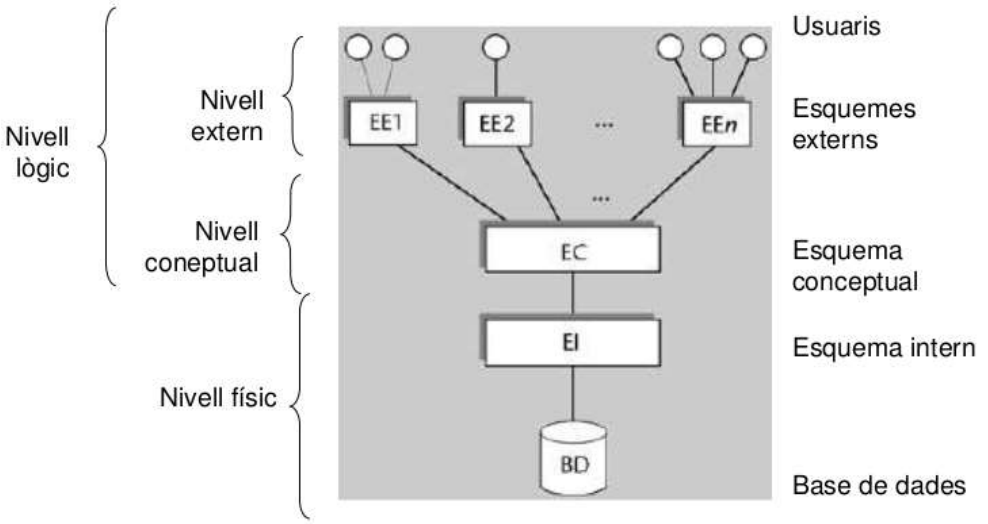
En l'apartat anterior que descrivíem els objectius i serveis que havia d'oferir un SGBD ja parlàvem de dos nivells de representació informàtica: el nivell lògic i el físic.

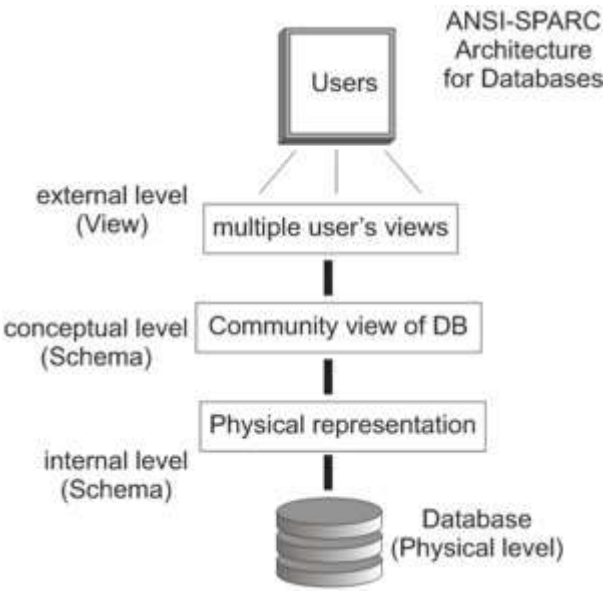
El **nivell lògic** ens amaga els detalls de com s'emmagatzemen, com es mantenen i com s'accedeix físicament a les dades. En aquest només es parla d'entitats, atributs i regles d'integritat.

El nivell físic és a on trobem de quina manera es guarda la informació en els dispositius d'emmagatzematge, com i a on s'agrupen els registres, de quina mida han de ser. Si existeixen estructures alternatives per millorar-ne el rendiment, etc...

La idea bàsica d'ANSI/SPARC consistia a descompondre el nivell lògic en dos: **nivell extern** i **nivell conceptual**. Anomenaven **nivell intern** el que hem anomenat *nivell físic*

Arquitectura ANSI/X3/SPARC





Nivell extern

Se situaran les diferents visions lògiques que els processos usuaris (programes d'aplicació i usuaris directes) tindran de les parts de la BD que utilitzaran. A aquestes visions s'anomenen **esquemes externs.**

En l'esquema extern només hi haurà aquells atributs i entitats que interessin, els podrem renumerar, podrem definir dades derivades, podrem redefinir una entitat perquè el programa que l'utilitzen creguin que en són dos, etc..

Nivell conceptual

Hi ha una sola descripció lògica bàsica, única i global que anomenem esquema conceptual, i que serveix de referència per a la resta d'esquemes.

L'esquema conceptual oculta els detalls de les estructures físiques d'emmagatzematge i es concentra en descriure les entitats, els seus atributs i tipus de dades, les interrelacions i també les restriccions o regles d'integritat.

L'esquema conceptual correspon a les necessitats d'un conjunt de l'empresa o món real que anomenarem **disseny lògic** de la BD

Nivell físic

Al nivell físic hi ha una única descripció física, que anomenem esquema intern que descriu l'estructura física d'emmagatzematge de la base de dades. L'esquema intern utilitza un model físic de les dades i descriu tots els detalls per el seu emmagatzematge.

Contindrà la descripció de l'organització física de la BD: camins d'accés(índexs, hashing, apuntadors,....), codificació de les dades, gesió de l'espai, mida de la pàgina, etc...

D'acord amb l'arquitectura ANSI/X3/SPARC, per a crear una BD cal definir-ne prèviament l'esquema conceptual, definir-ne com a mínim un esquema extern i, eventualment, definir-ne l'esquema intern. Si aquest últim no es defineix, el mateix SGBD haurà de decidir els detalls de l'organització física.

El propi SGBD s'ha d'encarregar de fer les correspondències (*mappings*) entre els tres nivells d'esquemes ja que cada grup d'usuaris fa referència exclusivament al seu propi esquema extern. Per tant, el SGBD ha de transformar qualsevol petició expressada en termes d'un esquema extern a una petició expressada en termes del esquema conceptual i llavors, en una petició en l'esquema intern que processarà sobre la base de dades. Aquestes correspondències poden necessitar temps, per això molts SGBD no contemplen l'ús de vistes externes.

En els SGBD relacionals (en el món de SQL), s'utilitza una terminologia lleugerament diferent. No se separen clarament 3 nivells de descripció.

Es parla d'un sol esquema (*schema*), però dins d'aquest s'inclouen descripcions dels tres nivells. En l'*schema* es descriuen els elements del que en l'arquitectura ANSI/SPARC s'anomena *esquema conceptual* (entitats, atributs, tipus de dades, restriccions) més les vistes (*view*), que corresponen aproximadament als *esquemes externs* d'ANSI SPARC.

El model relacional es limita al món lògic. Per això l'estàndard ANSI-ISO de l'SQL no parla en absolut del món físic o intern; ho deixa en mans dels SGBD relacionals del mercat.

Exemple d'esquema extern:

Imaginem una BD que a l'esquema conceptual té definida, entra moltes altres, una entitat *alumne* amb els atributs següents: *num_alumne, nom, cognom1, cognom2, numDNI, adreça, codi_postal, municipi, data_naixement, telèfon*. Però ens interessa que uns determinats programes o usuaris vegin la BD formada d'acord amb un esquema extern que tingui definides dues entitats, anomenades *estudiant* i *persona*.

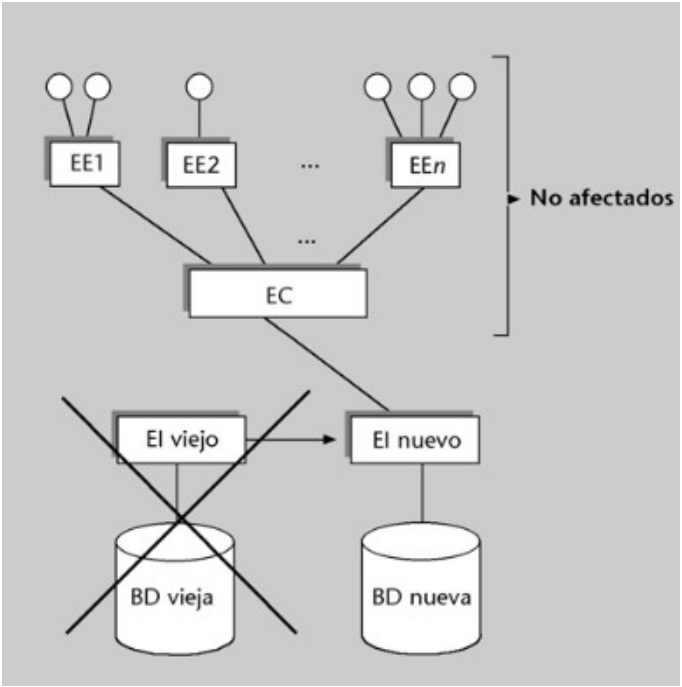
a) L’entitat estudiant podria tenir definit l’atribut num_matricula (definit com a derivat del num_alumne), l’atribut nom_pila (format per l’atribut nom i els 40 primers caràcters de l’atribut cognom).

b) L’entitat persona podria tenir l’atribut DNI (obtingut de numDNI), nom_complet (format pels atributs nom, cognom1 i cognom2), l’atribut adreça_postal (obtinguda per la concatenació dels atributs adreça, codi_postal i municipi)

1.2.2. Independència de les dades

A continuació veurem com l'arquitectura dels 3 nivells que acabem de descriure proporciona els dos tipus d'independència de les dades: **lògica** i la **física**.

D'acord amb l'arquitectura ANSI/SPARC, hi haurà **independència física** quan els canvis en l'esquema intern no afectin a l'esquema conceptual ni als esquemes externs.

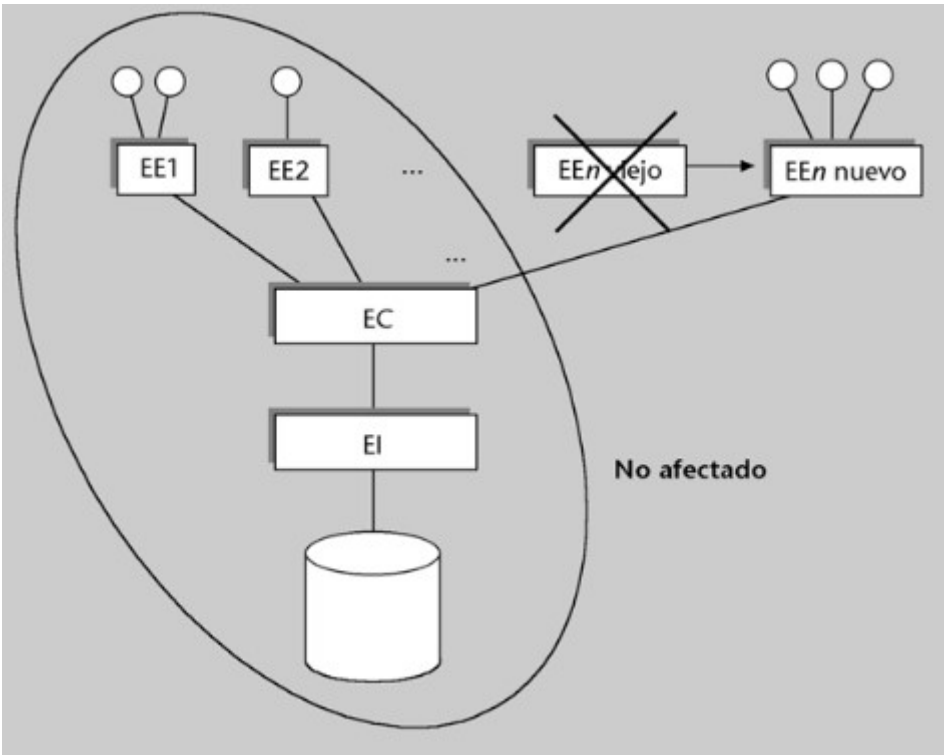


És obvi que quan canviem unes dades d'un suport d'emmagatzematge a un altres, , no es veuran afectats ni els programes d'aplicació, ni els usuaris directes, ja que no es modificarà l'esquema conceptual ni l'extern.

Tampoc s'ha de veure afectat l'esquema conceptual si canviem el mètode d'accés a uns registres determinats (per exemple, eliminant un índex en arbre-B o substituint-lo per un índex hashing), el format o la codificació de caràcters.

Ara bé que no afecti a l'esquema no voldrà dir que no afecti als usuaris, ja que les dades s'han de traslladar d'un lloc a un altre. Que no afecti a l'esquema conceptual significarà que tots els programes i consultes que tinguem fetes no s'hauran de tocar per res.

Tindrem **independència lògica** quan els usuaris (programes d'aplicació o usuaris directes) no es vegin afectats pels canvis en el nivell lògic.



Tindrem en consideració dos casos:

1) **Canvis en l'esquema conceptual.** Aquest tipus de canvi no afectarà als esquemes externs que no facin referència a les entitats o els atributs modificats. Per exemple, si afegim un nou atribut (data de naixement) per l'entitat alumne, els esquemes externs (ni els usuaris) que no facin referència a aquest atribut. O si es decideix augmentar de longitud d'un atribut en l'esquema conceptual, no serà necessari modificar l'esquema extern a on s'hagi definit.

2) **Canvis en els esquemes externs**. Tal i com es mostra en el dibuix anterior un canvi en l'esquema extern només afectarà als usuaris que utilitzin aquell esquema, però no ha d'afectar a els altres usuaris ni a l'esquema conceptual. I òbviament tampoc en l'esquema intern i a la BD física.

Casos a discutir

Si eliminem l'atribut cognom1 de l'esquema conceptual. Què passarà amb l'atribut nom_complert de l'esquema extern si aquest en depèn de l'atribut cogonom1?

Afectarà a tots els esquemes externs que depenguin de l'atribut cognom1. Si en algun esquema extern teníem un atribut anomenat nom_complert format per la concatenació de nom i cognom1 s'haurà de modificar aquest esquema extern, però no caldria modificar el programa simplement el nom_complert es mostraria sense cognom1

Què passa si augmentem la longitud de l'atribut cognom1 de tal manera que ara s'hi podran emmagatzemar 30 caràcters en comptes de 25 com fins ara.

Com a molt també s'hauran de modificar els esquemes externs que en depenen, però en la majoria dels casos no comportarà cap modificació

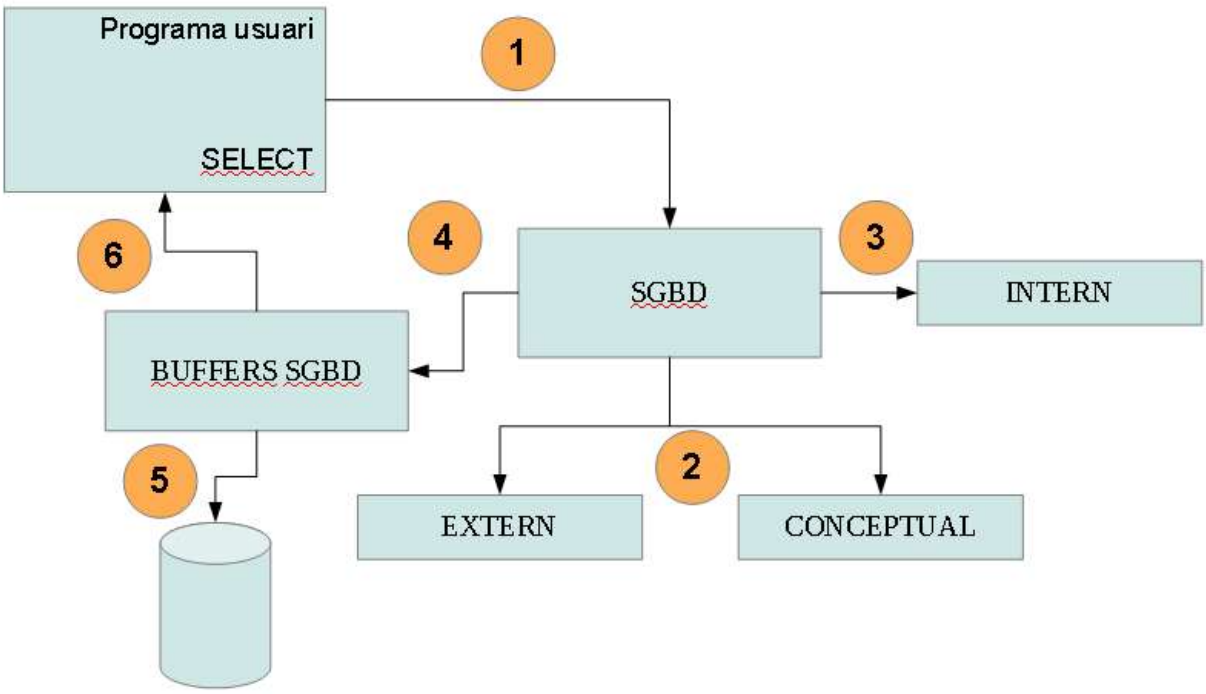
Si afegim un nou atribut data_naixement a l'entitat alumne de l'esquema conceptual?

No afectarà a cap esquema extern incloent aquells que es referien als atributs de l'entitat alumne.

1.2.3. Flux de dades i de control

Per entendre el funcionament d'un SGBD, en aquest esquema veurem els principals passos que segueix l'execució d'una consulta enviada al SGBD per un programa extern (usuari).

Encara que entre diferents SGBD comercials poden haver-hi enormes diferències, solen seguir el següent esquema



Imagineu que voleu obtenir les dades d'un alumne que té un determinat DNI.

1) El programa envia la consulta al SGBD

2) Aquest mitjançant els esquemes externs i conceptuals comprova que la sintaxi sigui correcta, que l'usuari tingui autorització per realitzar-la, etc..

3) Si la consulta es vàlida, el SGBD consulta l'esquema intern per veure quin és el mecanisme més eficient de respondre la pregunta i accedir a les dades.

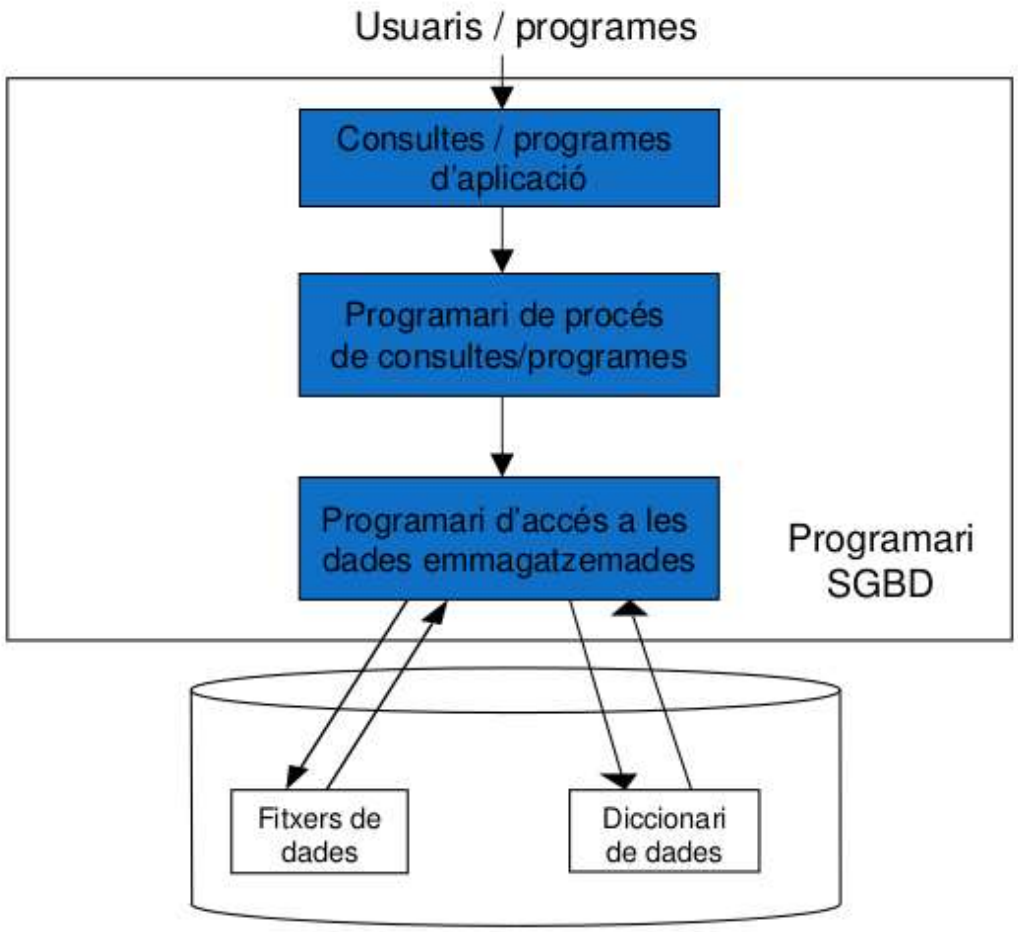
4) El resultat és l'adreça de la pàgina a on es troba el registre (entre molts altres) de l'alumne amb el DNI passat per paràmetre.

    a. Si la pàgina es troba en els *buffers (memòria principal)* en el moment de saber la pàgina. Per exemple perquè algú ja ha realitzat aquesta consulta, passem directament al pas 6.

5) Si el resultat no està als buffers el SGBD, amb l'ajuda del sistema operatiu (SO), busca en el dispositiu d'emmagatzematge i carrega els buffers amb les pàgines demanades

6) Un cop la pàgina està a memòria principal, busquem el registre dins la pàgina i la retornem al programa que ens ha realitzat la petició.

En el següent esquema podem veure d'una forma molt simplificada l'estructura d'un SGBD amb els seus elements de programari que el compon.



Diccionari de dades (data base schema)

- ESTUDIANT (número, nom, cognom, adreça, email)
- ASSIGNATURA (codi, nom)
- CLASSE (codi, nom)
- PROFESSOR (nom, cognom, email)
- ...

Fitxers de dades

ESUTDIANT

Número	Nom	Cognom	Adreça	Email
0165854	John	Doe	C/ Ponent, 17	jdoe@universitat.com
0168975	Fred	Astere	C/ Sol, 24	fastere@universitat.com
0157895	Kyara	Welton	Av. Esport, 13	kwelton@universitat.com

....

1.3. Llenguatges i interfícies de base de dades

Per comunicar-se amb el SGBD, els usuaris utilitzen un llenguatge. Hi ha molts llenguatges diferents segons els tipus d'usuari.

Cada SGBD marca quin o quins són els llenguatges mitjançant els quals podem interaccionar amb ell.

Els llenguatges especialitzats en l'escriptura d'esquemes, és a dir, en definir i descriure els objectes de la base de dades, la seva estructura, les relacions i restriccions es coneixen genèricament com a **DDL** (**D**ata **D**efinition **L**anguage)

El SGBD ofereix un llenguatge perquè els usuaris puguin manipular les dades que hi ha en els esquemes de la base de dades. D'aquesta manera els usuaris poden inserir, esborrar, modificar i recuperar les dades emmagatzemades. Aquests llenguatges s'anomenen **DML** (**D**ata **M**anipulation **L**anguage).

Dins del llenguatge DML trobaríem dos tipus:

- DML d'alt nivell o no per procediments (Non-*Procedural DMLs*): Aquest llenguatge es pot utilitzar de manera independent per especificar operacions a la BD. En molts SGBD es possible introduir interactivament instruccions de DML d'alt nivell des d'un terminal o bé incorporades en un llenguatge de programació de propòsit general. En aquest tipus de llenguatges l'usuari especifica què vol obtenir sense saber com s'obtindrà la informació (llenguatges del tipus declaratiu). Això allibera l'usuari de com s'estructuren les dades internament. SQL n'és un clar exemple i són els que estudiarem al llarg del curs.

- DML de baix nivell o per procediments (Procedural DMLs): Aquest DML han d'estar incorporats dins d'un llenguatge de propòsit general. En general, aquest tipus de DML l'usuari ha de definir quines dades vol obtenir i com fer-ho. Això significa que l'usuari ha d'expressar totes les operacions d'accés a dades per obtenir la informació. La informació s'obté registre a registres de forma individual, per tant, es necessita l'ús d'elements (bucles) del llenguatge de programació per obtenir i processar cada registre. Aquest tipus de llenguatges requereixen que l'usuari conegui el funcionament del SGBD. Llenguatges molt lligats a l'àlgebra relacional. Aquests tipus de llenguatge el trobem en models de dades jeràrquics o de xarxa

En els SGBD actuals que incorporen controls d'accés i permisos s'ha incorporat un tercer llenguatge anomenat **DCL** (**D**ata **C**ontrol **L**anguage). Que és el que ens permet definir quins permisos pot tenir un usuari respecte un element de la base de dades o definir diferents autoritzacions d'accés al propi SGBD.

Si el SGBD manté una clara separació entre els diferents nivells (extern, conceptual i intern). El DDL només ens servirà per especificar l'esquema conceptual. Per especificar l'esquema intern s'utilitzarà el llenguatge de definició d'emmagatzematge **SDL** (**S**torage **D**efinition **L**anguage). Per especificar l'esquema extern necessitarem del llenguatge anomenat llenguatge de definició de vistes **VDL** (**V**iew **D**efinition **L**anguage).

En el SGBD actuals no s'acostuma a distingir entre els tipus de llenguatges citats (DDL, VDL, DML, DCL i SDL(retirat)), normalment s'utilitza un llenguatge ampli

integrat que conté els elements necessaris per definir esquemes, definir vistes, manipulació de les dades. Un exemple representatiu és el llenguatge de base de dades relacional SQL, que representa una combinació de DDL, DML i DCL. Aquest llenguatge permet realitzar consultes (SELECT), manteniment de dades (INSERT, UPDATE i DELETE), creació de taules i restriccions (CREATE TABLE), creació de permisos (GRANT, REVOKE), control de transaccions (COMMIT, ROLLBACK), entre altres. Aquest llenguatge és implementat en la majoria dels SGBD relacionals.

Els SGBD actual tenen l'SQL com a llenguatge nadiu, però actualment ofereixen altres possibilitats com per exemple l'incorporació de llenguatges 4GL i eines visuals:

- Llenguatges de 4a generació solen combinar elements procedimentals amb elements declaratius. Aquests tipus de llenguatges moltes vegades s'incorporen dins del mateix SGBD de tal manera que el programador por realitzar programes dins del SGBD per la manipulació de les dades.

- Eines o interfícies visuals que permeten utilitzar el SGBD mitjançant diàlegs amb finestres, icones, ratolí, etc.. per facilitar-ne la interacció.

1.4. Usuaris

En una BD personal i petita, com una llista d'adreces d'amics, habitualment una sola persona defineix, construeix i manipula la BD.

En canvi, en una BD gran, amb un volum d'informació important i amb centenars d'usuaris, moltes persones participen en el seu disseny, ús diari i manteniment.

Podem diferenciar els diferents tipus d'usuaris que interactuen amb els SGBD depenent de l'ús o el coneixement que tenen.

Tipus d'usuaris:

- **Usuaris finals**

  Són les persones que necessiten l'accés a la base de dades per consultar-la actualitzar-la i generar informes. L'existència de la BD està justificada perquè aquests tipus d'usuari l'utilitzin. Trobem diferents categories d'aquests usuaris:

  - **Esporàdics**: Tenen accés de forma puntual a la BD i normalment necessiten informació diferent cada vegada.. Solen ser gerents de nivell intermig/alt.

  - **No informàtics / simples**: Normalment són els que constitueixen una porció més gran de la totalitat dels usuaris finals. La seva funció principal gira en l'entorn de consultar i actualitzar constantment la BD utilitzant operacions que s'han programat i provat prèviament.
    Utilitzen programari desenvolupat per realitzar les seves tasques i interactuar amb la BD. Utilitzen BD sense saber-ne res (caixers d'un supermercat, gestors d'un banc, recepcionistes d'hotel,...)

- **Avançats**: Persones que coneixen bé SGBD (enginyers, analistes, caps financers) Saben treure suc BD a partir de llenguatges de 4a generació o altres.

- **Personals autònoms**: Treballen amb una BD amb una interfície gràfica i tenen nocions (bd personals). Utilitzen bases de dades personalitzades gràcies als paquets de programes comercials basats en menús o en gràfics. Per exemple, l'usuari ha creat una petita BD pel mateniment de clients amb MS Access.

- **Dissenyadors de BD**

  Els dissenyadors de la base de dades són els encarregats de dissenyar l'esquema d'una BD, identificar les dades que s'emmagatzemaran a la BD i d'escollir les estructures adequades per a representar i emmagatzemar aquestes dades.

  En general, aquestes tasques es realitzen abans d'implementar-les a la base de dades.

  Els dissenyadors tenen la responsabilitat de comunicar-se amb els futurs usuaris de la BD, per a conèixer les seves necessitats, i presentar un disseny que satisfaci aquests requeriments.

  La funció dels dissenyadors és clau perquè un projecte surti bé, ja que un error en el disseny de la BD podrà significar tornar a iniciar el procés de presa de requisits.

  En moltes ocasions els dissenyadors formen part del personal del DBA i un cop finalitzat el disseny, assumeixin responsabilitats dins aquest grup.

- **Administradors de BD (DBA)**

  També anomenat tècnic de la BD és qui dur les feines d'administració i control de la BD. A aquestes funcions es coneix amb el nom d'Administració de BD (ABD) i els usuaris que fan aquest tipus especial de feina són els Administradors de BD (**DBA D**ata**B**ase **A**dministrators)

  Una empresa que tingui Sistemes d'Informació construïts entorn a una BD necessitarà una persona per tal de que l'explotació de la BD sigui correcta.

  Els administradors de BD són els responsables del correcte funcionament de la BD i vigila que sempre sigui operativa i es mantingui útil. Intervenen en situacions problemàtiques o d'emergència i una de les seves grans responsabilitats es vetllar perquè no es produeixin incidents.

  També és qui es coneix més bé la tecnologia emprada en la BD i en pot treure el màxim rendiment i aplicar polítiques de seguretat per implementar plans de contingència

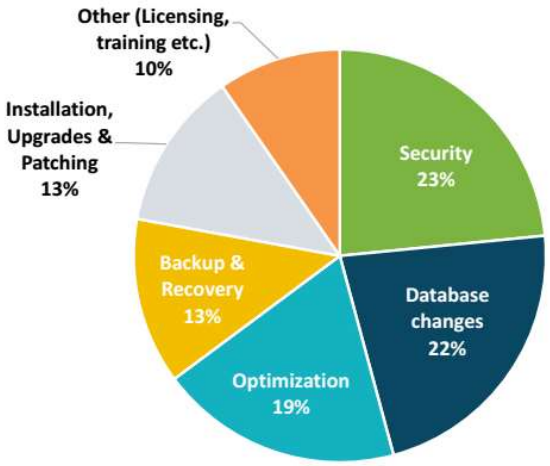
  A continuació trobem una llista de les tasques típiques d'un DBA:

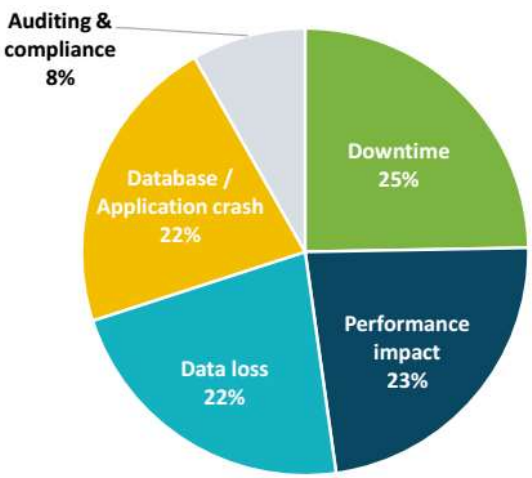
- Manteniment, administració i control dels esquemes. Comunicació dels canvis als usuaris.
- Assegurar la màxima seguretat i disponibilitat de les dades, preparant plants de contingència en cas de desastres.
- Resolució d'emergències
- Vigilància de la integritat i de la qualitat de les dades.
- Disseny físic de la BD i control del rendiment prenent decisions relatives a les modificacions en els esquemes i/o en els paràmetres del SGBD i del SO, per millorar-lo. (quins índex, a on es guarda la informació, com,..) D'aquesta manera per treure'n el màxim rendiment.
- Establir una normativa, assessorament i criteris d'ús als programadors i als usuaris finals sobre la utilització de la BD
- Dur un control i administració de les autoritzacions, restriccions, etc.(qui té accés sobre què).

Depenent de l'entorn a on ens trobem podem trobar més o menys DBA. En sistemes molt grans i crítics s'arriba a tenir grups de més de 10 DBAs amb equips de treball única i exclusivament a realitzar aquestes tasques.

Bona part del software que acompanya els SGBD comercials està orientat a facilitar la gran diversitat de tasques que hem anomenat anteriorment. Proporcionant monitors de rendiments, monitors de seguretat, verificadors de la consistència entre índexs i dades, reorganitzadors, gestors de còpies de seguretat, etc....

Com gasten el seu temps els DBA segons l'enquesta: *2018 Database DevOps Survey* de DBmaestro





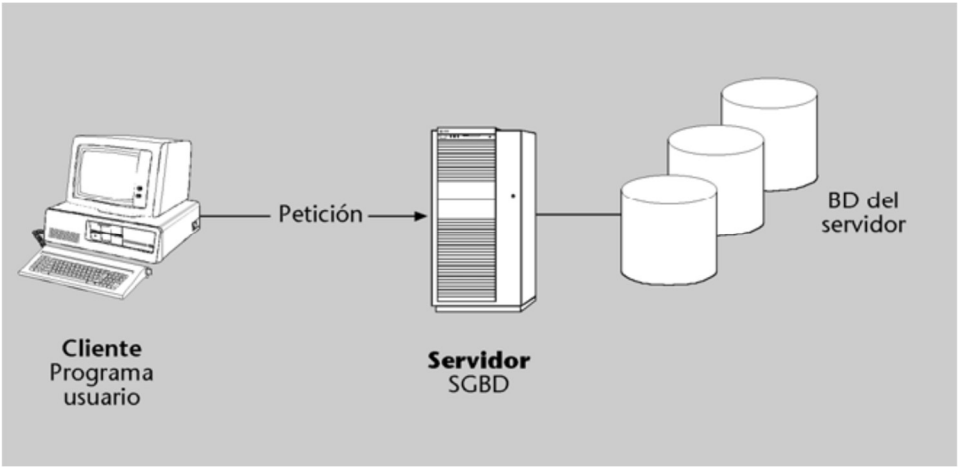
Quins són els riscos quan despleguem canvis a la nostra base de dades segons l'enquesta: *2018 Database DevOps Survey* de DBmaestro

SGBD distribuïts i arquitectura client-servidor

1.5. Arquitectura Client-Servidor

La idea del C/S és senzilla. Dos processos diferents, que s'executen un una mateix sistema o en sistemes separats, actuant de manera que un fa de client o peticionari d'un servei i l'altre fa de servidor o proveïdor del servei.

Per exemple un programa d'aplicació d'un PC d'un usuari demana certes dades d'una BD que resideixen en un equip UNIX on s'executa el SGBD relacional que la gestiona.



La facilitat per a disposar de distribució de dades no és l'única raó, ni tan sols la bàsica, del gran èxit dels entorns C/S als anys noranta. Potser el motiu fonamental ha estat la flexibilitat per a construir, fer modificacions i fer créixer la configuració informàtica global de l'empresa, a base de maquinari i programari molt estàndard i barat.

Per exemple, tot el que fa referència al món web es basa en un arquitectura client/servidor.

Existeix un navegador que realitza una petició en un servidor, en aquest cas web, i aquest recepciona la petició l'analitza i retorna la resposta, que normalment serà contingut HTML al client, en aquest cas el navegador web.

Resumint, existeixen dos processos: client i servidor.

Client: Realitza la petició al servidor

Servidor: Recepciona la petició la processa i la retorna al client.

1.6. Introducció als conceptes de SGBD distribuïts

Una base de dades distribuïda és una col·lecció de dades que pertany al mateix sistema, però està dispersa físicament entre diferents llocs d'una xarxa d'ordinadors.

Podem definir una base de dades distribuïda com un conjunt de base de dades interconnectades entre si. La informació emmagatzemada en aquestes base de dades han d'estar relacionades entre si.

Diferents factors ha donat peu a la creació SGBDD (Sistemes Gestors de Base de Dades Distribuïts):

- La pròpia naturalesa de distribució d'algunes aplicacions: Moltes de les aplicacions estan distribuïdes de forma natural en diferents llocs. Per exemple, una companyia que tingui diferents oficines en diferents ciutats. Molts usuaris locals tenen accés exclusivament a les dades que estan en el mateix lloc. I altres usuaris (globals) necessiten tenir accés ocasional a les dades emmagatzemades en diferents llocs.

- Millor fiabilitat i disponibilitat: La fiabilitat (la probabilitat de que un sistema estigui en funcionament en un moment determinat) i la disponibilitat (probabilitat que el sistema estigui disponible contínuament durant un cert període de temps). Quan les dades i el software del SGBD està distribuït en diferents llocs, un lloc pot fallar mentre que els demés segueixen operant. **En un sistema centralitzat la fallada en un lloc fa caure el sistema complert**.

- Possibilitat de compartir dades.

- Millor rendiment: Quan una base de dades gran està distribuïda en múltiples llocs. Això fa que les consultes i manipulació de cadascuna de les base de dades distribuïdes tinguin un millor rendiment a nivell local i el volum de les dades a manipular sigui menor.

La **distribució** implica un **augment de la complexitat del disseny i la implantació** del sistema.

Els SGBDD han de comptar amb les característiques dels sistemes centralitzats i a més:

- La capacitat de tenir accés a llocs remots i transferir consultes i dades entre els diferents llocs.

- La capacitat de seguir la pista a la distribució i la replicació de les dades en el diccionari de dades del SGBD

- La capacitat d'elaborar estratègies d'execució per consultes i transaccions que tenen accés a dades de més d'un lloc.

- La capacitat de decidir a quina còpia d'un element replicat es tindrà accés.

- La capacitat de mantenir la consistència de les còpies d'un element d'informació replicat
- La capacitat de recuperació de caigudes de llocs individuals i de nous tipus de fallades com la comunicació entre nodes.

Inconvenients que ens podem trobar:

- Complexitat d'administració: calen més habilitats per tal de manegar una base de dades distribuïda: seguretat, configuracions, etc.
- Complexitat de software: en cas que les transaccions estiguin suportades pel SGBDD aquestes són molt més complexes de manegar.
- Seguretat: els nodes han d'estar connectats entre ells, cal fer-ho de forma segura per evitar intrusos.

1.7. Distribució de la base de dades

En aquest apartat analitzarem les diferents tècniques per dividir la base de dades en unitats lògiques, anomenades fragments, per tal d'emmagatzemar-los en diferents llocs i/o replicar-les.

El primer problema que tenim que resoldre és com distribuirem la nostra base de dades. Imaginem que tenim una taula 'R' existeixen dues formes de distribuir-la:

- **Rèplica:** El sistema guarda vàries còpies de les dades en diferents ubicacions. (redundància intencionada).
    - Disminuir els costos de comunicació entre els diferents nodes.
    - Si cau un node de la xarxa, pot seguir funcionant la base de dades.
    - La duplicació té com a principal inconvenient que les actualitzacions de dades són molt més complexes.
- **Fragmentació:** Consisteix en dividir les dades en diferents trossos de manera que a partir dels fragments, podem recomposar-los.
    - **Horitzontal**: Dividim les taules en conjunts de registres complets.
    - **Vertical**: Dividim les taules en els seus camps corresponents.

Aquestes formes de distribució es poden combinar per donar lloc en formes mixtes.

1.7.1. Rèplica

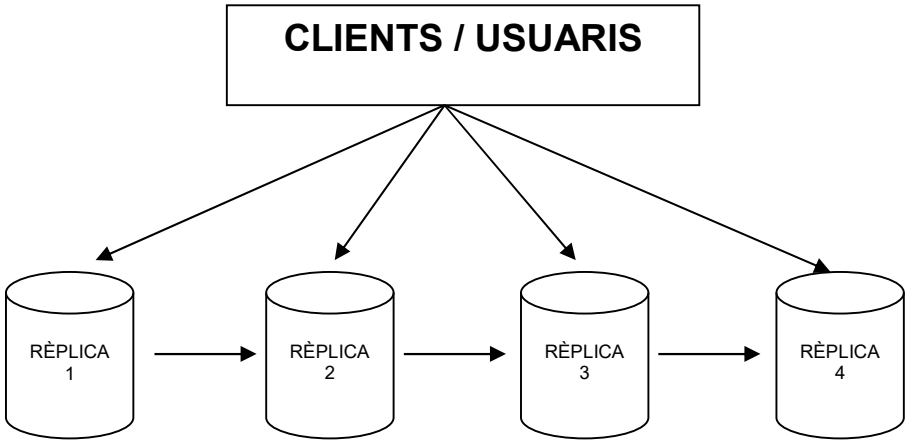
Tal i com hem dit anteriorment, consisteix en guardar còpies de les taules en diferents ubicacions.

L'opció de rèplica ens facilitarà molt les operacions de lectura, ja que disposarem de les mateixes dades en diferents localitzacions. Això comportarà que augmenti molt la disponibilitat de les base de dades.

La rèplica també afavoreix en l'execució de consultes complexes. Un node pot estar realitzar una consulta complexa quan un altre rep peticions petites d'aquesta manera la càrrega de consulta es reparteix en els diferents nodes.

Per contrapartida aquest sistema ens complicarà les operacions d'actualització. Cada vegada que haguem de modificar una dades, hem de realitzar-la a totes les ubicacions al mateix temps, ja que sinó perdríem consistència de els dades. Normalment es en què un node té més pes que uns altres en les actualitzacions.

Per exemple, la còpia de les dades financeres localitzada en la sucursal a on té oberta el seu compte, tindrà més pes que en les còpies d'altres sucursals i d'aquesta manera si es troba una inconsistència s'agafarà aquesta còpia com a referència de rèplica principal per aquella dada.



1.7.2. Fragmentació

Fragmentar una relació (taula) R, consisteix en dividir-la en diferents relacions: R1, R2, R3,...Rn. De manera que a partir d'aquests fragments, podem recomposar la relació original R.

Bàsicament tenim 2 tipus de fragmentació

- **Fragmentació horitzontal:** Les taules les dividim en conjunts de tuples completes. Per poder crear una fragmentació horitzontal utilitzarem l'operador de selecció o delta (δ) sobre la relació original en el model relacional.

  Si volem recomposar la relació utilitzarem l'operador d'unió ($\cup$).

Exemple: En una cada sucursal de banc es guarden dades de clients locals.

R

BANC	NOM_BANC	SUCURSAL	DC	COMPTE	TITULAR
534	MNUH	1234	23	234234234	Joan Ciurana
534	MNUH	1234	22	236666666	Marga Mir
534	MNUH	5678	55	777777777	Pere Roure
534	MNUH	5678	66	888888888	Martí Pol
534	MNUH	3456	77	666666666	Laia Ripoll

Realitzant una fragmentació horitzonal per per sucursal obtindrem 3 noves relacions (taules) R1, R2 i R3.

R1:

BANC	NOM_BANC	SUCURSAL	DC	COMPTE	TITULAR
534	MNUH	1234	23	234234234	Joan Ciurana
534	MNUH	1234	22	236666666	Marga Mir

R2

BANC	NOM_BANC	SUCURSAL	DC	COMPTE	TITULAR
534	MNUH	5678	55	777777777	Pere Roure
534	MNUH	5678	66	888888888	Martí Pol

R3

BANC	NOM_BANC	SUCURSAL	DC	COMPTE	TITULAR
534	MNUH	3456	77	666666666	Laia Ripoll

- **Fragmentació vertical:** Si dividim les taules per un conjunt de camps. Per crear aquesta fragmentació vertical utilitzarem l'operador de projecció pi (π). Si volem recomposar la relació utilitzarem l'operador de join natural (⋈).

  Exemple: Realitzarem una fragmentació vertical agrupant les tuples banc, nom_banc, sucursal i DC en una relació i titular en una altre relació.

  R

BANC	NOM_BANC	SUCURSAL	DC	COMPTE	TITULAR
534	MNUH	1234	23	234234234	Joan Ciurana
534	MNUH	1234	22	236666666	Marga Mir
534	MNUH	5678	55	777777777	Pere Roure

  Realitzant una fragmentació vertical escollint 2 grups de tuples obtindrem 2 noves relacions (taules) R1, R2.

  En aquest cas cal distingir que tindrem una tupla comuna que haurà de ser comú en totes les relacions.

  R1:

COMPTE	BANC	NOM_BANC	SUCURSAL	DC
234234234	534	MNUH	1234	23
236666666	534	MNUH	1234	22
777777777	534	MNUH	5678	55

  R2

COMPTE	TITULAR
234234234	Joan Ciurana
236666666	Marga Mir
777777777	Pere Roure

També podem trobar fragmentacions mixtes de tal manera que utilitzen fragmentació horitzontal i vertical alhora.

En aquests casos podem trobar fragmentacions VH i HV. Les primeres realitzen una fragmentació vertical sobre la relació original i llavors per cada relació nova se li aplica una fragmentació horitzontal. En canvi a les HV és al revés, a la relació original se li aplica una fragmentació horitzontal i a cada nova relació se li aplica una de vertical.

1.7.3. Exemples de distribució

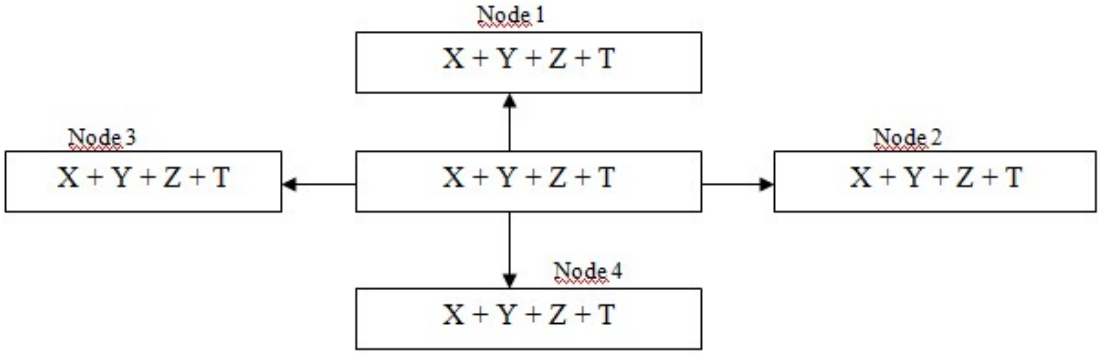
Moltes vegades la fragmentació i la rèplica no es donen de forma independent, sinó que se'n fan combinacions.

Suposarem que la nostra base de dades està dividida en quatre parts **X, Y, Z i T**. Som un petit banc que només compte amb quatre oficines/sucursals, a on cada sucursal l'anomenarem **node**.

Els diferents dissenys de distribució els podem englobar en les següents categories:

- **Base de dades multiplicada**

Aquest esquema de distribució es regeix perquè la nostra base de dades es dupliqui en cada node.



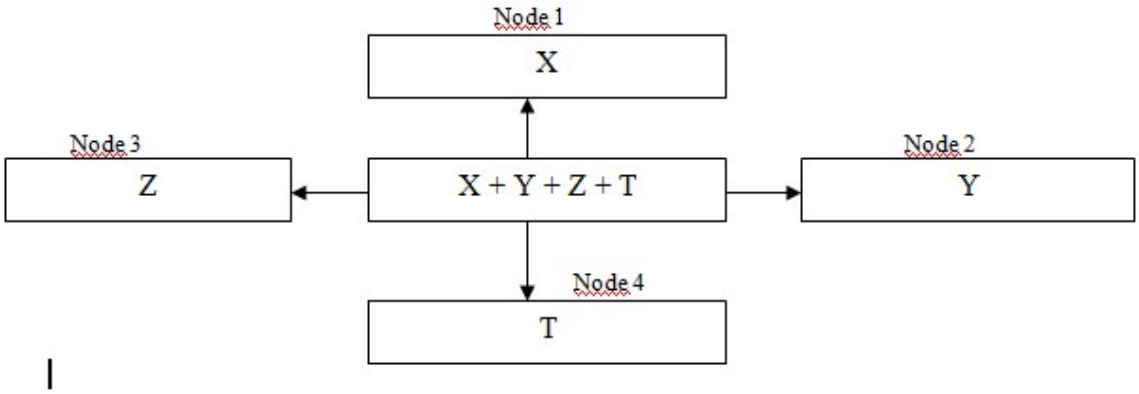
L'avantatge d'aquest disseny consisteix en què la base de dades està a tots i cadascun dels nodes. Si apareix un problema en qualsevol dels nodes els altres seguirien funcionant sense problemes.

Coma desavantatge, lògicament, trobem que necessitem 4 vegades l'espai normal d'emmagatzematge. Una altre problemàtica serien les operacions d'esborrat, inserció i modificació de les dades. Si hem d'esborrar una dada d'un node, l'haurem d'esborrar de tots els nodes, generant molt de tràfic per aquestes operacions i possibles problemes de concurrència i inconsistència de la nostra base de dades.

Aquest esquema és difícil de portar-lo a la pràctica ja que implicaria duplicar la infraestructura a cada node.

- **Base de dades partionada**

Aquest esquema és l'oposat en l'esquema anterior. Consisteix en què cada node emmagatzemi una part de la nostra base de dades sense utilitzar rèplica de dades.



L'avantatge d'aquest esquema és que ocupa el mateix espai que la base de dades de forma centralitzada. Les operacions d'actualització són molt senzilles, només hem d'actualitzar les dades en un sol node.

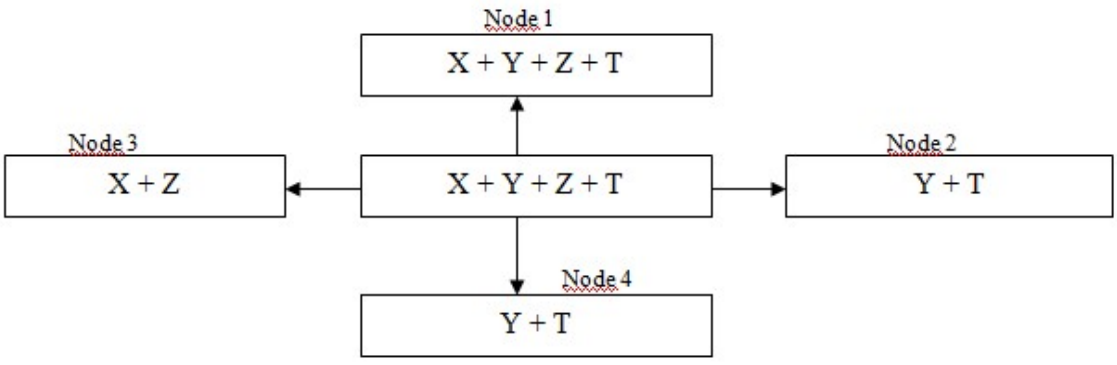
Aquí es compliquen les operacions de consulta, ja que si volem consultar part de les dades Z i T hauríem d'accedir a dos nodes, concretament al node 3 i 4.

Si aparegués algun problema en algun dels nodes, no podríem accedir a part de les dades.

Depenent de la fragmentació utilitzada podríem causar un alt trànsit a la xarxa. Una bona solució seria emmagatzemar les dades dels clients de cada localitat en cada node.

- **Base de dades amb node principal**

En aquest cas, cada node conté una part de la base de dades i un node conté la base de dades completa i en els altres nodes guardaríem parcialment la base de dades utilitzant la rèplica de les dades. Imaginem, per exemple, que el node 1 seria el node central del banc i els altres nodes les sucursals.

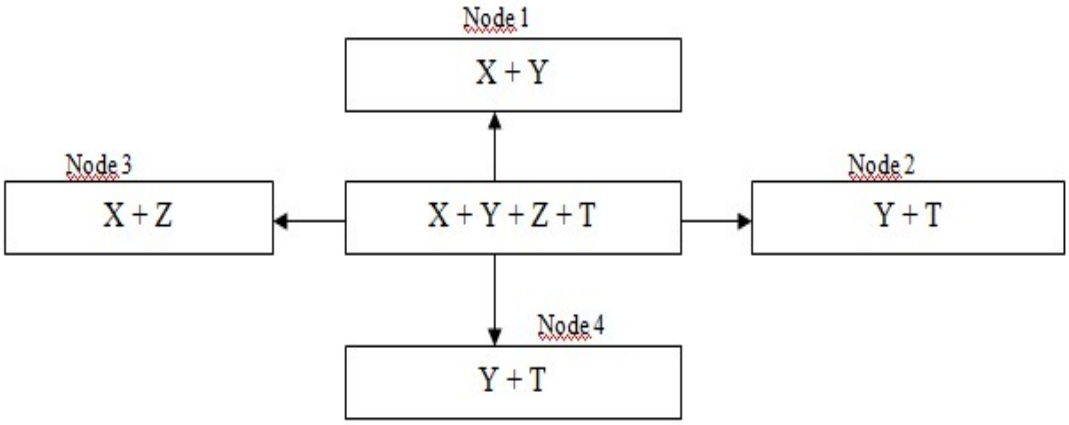


Com avantatge és que les actualitzacions només es realitzarien en un sol node al principal i les consultes en els nodes replicats. D'aquesta manera es podria dividir la càrrega de cadascun dels nodes.

Com a desavantatge trobem que al tenir la base de dades centralitzada si cau el node principal haurem de tenir un mecanisme perquè les dades o bé es repliquin o bé construir un nou node principal en base a les seves rèpliques.

- **Base de dades distribuïda amb duplicació en nodes concrets**

En aquest últim cas cap node conté la base de dades completa. En cada node es duplica una part de la mateixa. En aquest cas al no disposar d'un node principal, si cau qualsevol node, la xarxa pot seguir funcionant. Per exemple cada sucursal podria tenir la informació dels seus clients i la d'una altre sucursal.



1.8. Sistemes gestors de base de dades

En el punt anterior hem vist com distribuir la nostra base de dades. Ara ens cal veure el sistema gestor de base de dades que triarem. Cada node pot tenir un sistema gestor diferent i compartir informació amb els altres nodes.

Atenent aquest principi podem classificar-los en:

Sistemes homogenis

Tots els nodes utilitzen el mateix SGBD.

L'avantatge d'aquestes sistemes é que les interaccions entre les diferents base de dades són més senzilles.

Com a desavantatge podem dir que haurem de tenir els mateixos, o molt similars, requisits de hardware i software a cada node. Per exemple, si utilitzem una versió d'un SGBD segurament ens obligarà a tenir la mateixa versió a cada node amb els requisits de hardware i de sistema operatiu corresponents. A més si estem davant d'un producte comercial els costos de les llicències pot acabar sent un problema econòmic que no pas tècnic.

Sistemes heterogenis

En aquest cas cada node pot utilitzar un sistema gestor de base de dades diferent.

En aquest cas, hem de realitzar conversions entre els diferents sistemes gestors de base de dades. Això ens obligarà a tenir una cap de software addicional que es situï entre les base de dades i els usuaris i s'encarregui de les conversions necessàries.

Una de les principals desavantatges és la comunicació entre les diferents base de dades.

1.9. Transparència de les dades i disponibilitat

Quan estem davant d'un sistema distribuït l'usuari no ha de saber de quina manera estan distribuïdes les dades o els SGBD escollits.

Per exemple si algú realitza una consulta a una base de dades distribuïda, se li ha de retornar el resultat de la mateixa forma que amb un sistema centralitzat. Suposem que és el SGBD l'encarregat de decidir a on haurà d'anar a buscar la informació a cada node si cal i retornar-la

D'això em podríem denominar **transparència de les dades** d'un SGBD distribuït. A mesura en què podem desentendre'ns de la localització física de les dades.

Una de les motivacions destacables quan es crea una base de dades distribuïda és generar una **alta disponibilitat**. Com a disponibilitat entenem que la nostra base de dades estigui disponible pel seu ús el major temps possible. Seguint amb l'exemple d'una entitat bancària, ens interessa que les seves base de dades siguin operatives les 24 hores del dia durant els 365 dies l'any.

La disponibilitat està relacionada amb la tolerància a les fallades del sistema, la seva capacitat de recuperació i reconfiguració. Això és el que s'anomena la **robustesa del sistema**.

Terminologia

En aquest mòdul i en altres s'utilitzaran termes i noms que cal que tot tècnic informàtic cal que sàpiga.

- **Dades:**
  - Fets coneguts dels guals podem guardar informació: Nom, data de naixement d'una persona
- **Informació:**
  - Resultat de la transofmració i interpretació de les dades.
- **Bit o dígit binari**
  - Unitat mínima d'informació i pot prendre dos valors: 0 o 1.
- **Byte**
  - Agrupació de 8 bits, que s'utilitzen en la majoria dels computadors per poder representar informació o una posició de memòria (també 16, 32 o 64.
- **Camp:**
  - Unitat més petita a la qual es pot referir un programa
  - Format per qualsevol número de bits
  - Entre tots els camps que formen un registre sol haver-hi un camp clau o principal que té la propietat que el seu valor identifica a un únic registre lògic dins del fitxer.
- **Agregat de dades:**
  - Col·lecció de dades a les que se'ls anomena com un tot.
  - Ex: DATA (dia, mes, any).
- **Registre lògic o registre:**
  - Conjunt de camps o agregats de dades relacionats.

Camp 1	*Camp 2*	*Camp 3*	*Agregat de dades*			*Camp 4*	
Número	**Cognoms**	**Nom**	**Data alta**			**Departament**	
			Dia	**Mes**	**Any**		
2345	Pi Soler	Pere	12	05	1989	C	*Registre lògic*
1234	Alech Pujol	Maria	30	04	1990	A	*Registre lògic*
6789	Coderch Vila	Joan	30	01	1990	A	*Registre lògic*
5667	Soler Pibernat	Anna	28	10	1999	B	*Registre lògic*
…	…	…	…	…	…	…	

- **Registre físic o bloc:**
    - Unitat de transferència de dades entre el dispositiu d'emmagatzematge de dades i la memòria principal.
    - Pot estar constituït per varis registres lògics
    - Al nombre de registres lògics que formen un registre físic se l'anomena factor de bloqueig.
- **Fitxer:**
    - Conjunt de registres del mateix tipus, guardat en dispositius d'emmagatzematge secundari.
- **OLTP**
    - ¿¿??
- **OLAP**
    - ¿¿??

Big Data

1.10. Introducció

1.10.1. Definicions

> *"Big data is like teenage sex; everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."*
>
> *Dan Ariely, Duke University*

> *Big data o dades a gran escala es un concepte que fa referencia a un conjunts de dades tan grans que aplicacions informàtiques tradicionals de processat de dades no són suficients per tractar-les*
>
> *Wikipedia*

> *Big data fa referència a dades d'un mida molt gran, de forma que la seva manipulació i gestió presenten desafiaments significatius.*
>
> *Oxford English Dictionary*

> *Big data fa referència al conjunt de dades que sobrepassa la capacitat del software tradicional per poder ser capturats, emmagatzemats, gestionats i analitzats.*
>
> *McKinsey&Company*

> *L'habilitat d'aprofitar la informació per aconseguir crear noves idees i amb això valor*
>
> *Viktor Mayer*

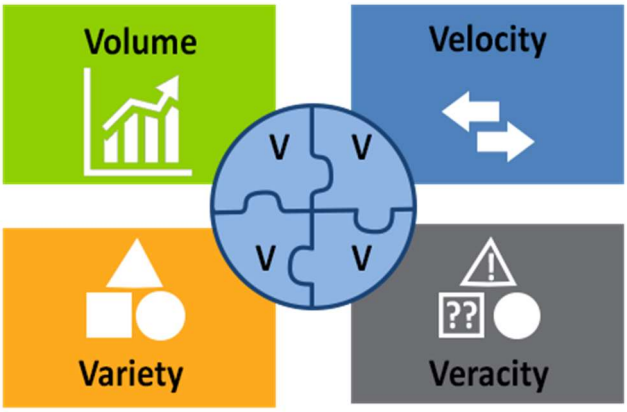
1.10.2. V's del Big Data

Aquests 4 conceptes van molt lligats en el Big Data i com s'ha dit en alguna definició anterior, podríem afegir-ne algunes més com **valor, variabilitat, visualització**.

Volum: Grans quantitats de dades a tractar. Ordes de magnitud de Petabyte – Zettabyte,...

Varietat: Dades estructurades, no estructurades, textos, imatges, etc...

Velocitat: Quantitat de temps que invertim en el processat de les dades. Terme important i a tenir en compte



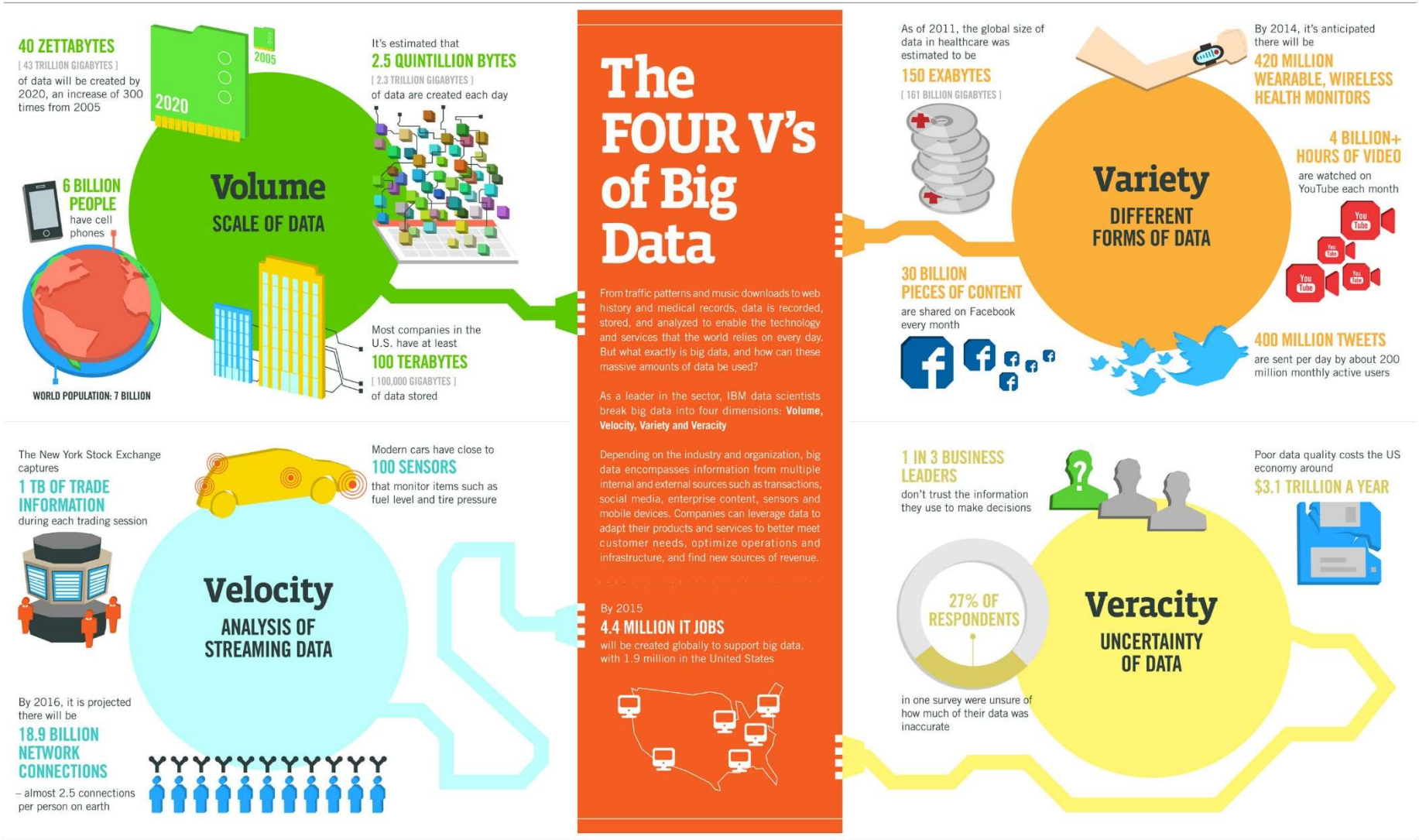
Veracitat: Fiabilitat de la informació que obtenim.

Variabilitat: Capacitat de les dades de canviar constantment. El significat d'una paraula pot variar depenent del context en el qual ens trobem.

Valor: Convertir les dades en brut en informació útil. De tot el que fem hem d'aconseguir-ne un valor.

Visualització: Cal un mecanisme de facilitat de lectura i comprensió d'aquestes dades i els resultats obtinguts. Gràfiques i no mitjançant fulles de càlcul.

Un altra infografia a on explica les 4 V's del Big Data de IBM: Link


40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005
2005
2020
It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day
6 BILLION PEOPLE
have cell phones
WORLD POPULATION: 7 BILLION
Volume
SCALE OF DATA
Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored
The Four V's of Big Data
From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?
As a leader in the sector, IBM data scientists break big data into four dimensions: Volume, Velocity, Variety and Veracity
Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.
By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States
As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]
By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS
Variety
DIFFERENT FORMS OF DATA
4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month
30 BILLION PIECES OF CONTENT
are shared on Facebook every month
400 MILLION TWEETS
are sent per day by about 200 million monthly active users
The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session
Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure
Velocity
ANALYSIS OF STREAMING DATA
By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth
1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions
Poor data quality costs the US economy around
$3.1 TRILLION A YEAR
27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate
Veracity
UNCERTAINTY OF DATA
Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS
IBM

1.10.3. Conceptes bàsics

Processament distribuït

Les dades es processen en diferents màquines (nodes) aconseguint millorar la velocitat.

No treballar amb un sol fitxer, sinó dividir les dades en diferents màquines i que aquestes processin la seva parcel·la de dades.

Escalabilitat horitzontal

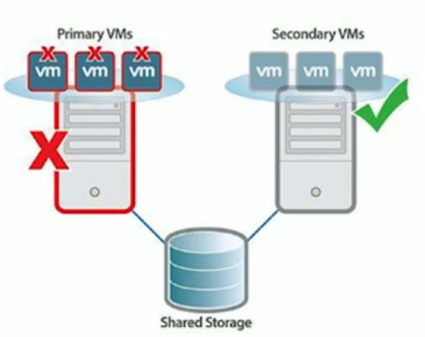
Els sistemes creixen molt ràpid. Per tant necessitem sistemes que ens permetin afegir nous nodes i no afegir nou HW a les màquines existents.

Amb això aconseguirem que mitjançant hardware econòmic puguem realitzar la feina sense necessitat de fer grans inversions.

Tolerància a fallades

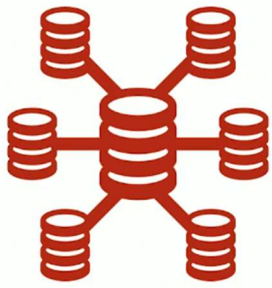
Cal que la infraestructura sigui tolerant a fallades. Si un node cau, el sistema ha de seguir funcionant i redistribuir la feina de les màquines que han caigut.

Si una màquina cau no cau tot el sistema i poder seguir processant les dades encara que en més temps.



Localitat de les dades

Cada node/màquina treballarà amb les dades que tingui més a prop. Així aconseguim evitar colls d'ampolla i minimitzar latències.

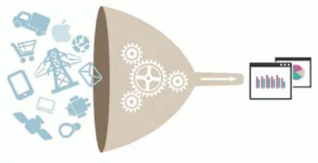


1.10.4. Cicle de vida

El cicle de vida normalment el podem definir en 4 fases:

1. **Ingesta:** Les dades procedents d'una font externa (web, BBDD,..) entren en el nostre sistema per començar a treballar amb elles.

   Per exemple un procés nocturn que reculli les dades de la BBDD i les tracti o un procés en *streming* que vagi agafant tweets o dades IoT.



2. **Persistència:** Cal guardar les dades i emmagatzemar-les en el nostre sistema. Normalment en una BD relacional o no relacional. D'alguna manera les estructurem com volem.

3. **Processament:** Fase en la que processem i analitzem les dades realitzant operacions per tal d'aconseguir valor.

4. **Visualització:** Mostrar gràficament o d'una forma amigable els resultats obtinguts per aquelles persones que n'extrauran conclusions.

1.10.5. Casos d'ús

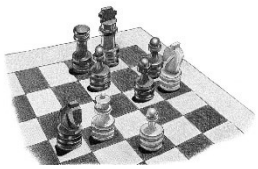
Processament de logs: Anàlisi de grans quantitats de dades desestructurades que es converteixen en valor. Per exemple, dades dels consums energètics d'una vivenda, processament de logs de servidors per trobar colls d'ampolla, a on hi ha més errors, etc...

Sistemes de recomanació: Recomanació de productes gracies a l'anàlisi de les preferències i perfils d'altres usuaris.

Cercadors web: Cerca i ordenació de millons de pàgines d'acord amb unes paraules clau.

Salut: Processament d'anàlisi clínics, ADN, hàbits alimentaris i ajuda a comprendre malalties. Si controlem que certs pacients amb els següents hàbits i han tingut certes malalties podem treuen conclusions.

Esport: Estudiar i analitzar estadísticament i vídeos per treure patrons típics de joc, analitzar rivals, etc...



Món Financer: Quan comprar, vendre, anàlisi de mercat, etc....

1.10.6. Evolució de les dades

En el 2013 hi havia 4,4 zettabytes en el mon i es preveu que en el 2020 hi hagi 44 zettabytes.

El creixement de les dades són exponencials.

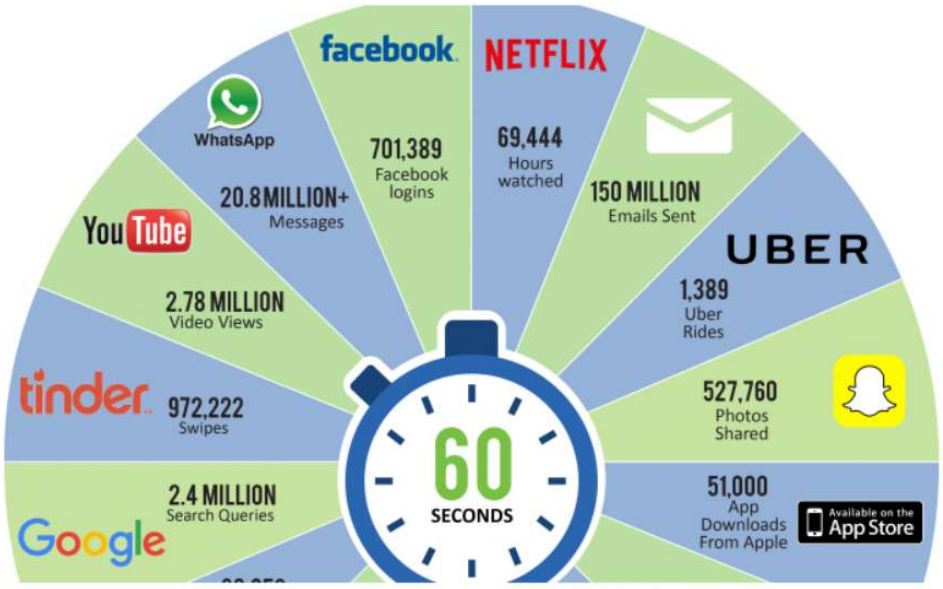
En el 2020 cada ser humà crearà 1,7MB d'informació cada segon.

Facebook:

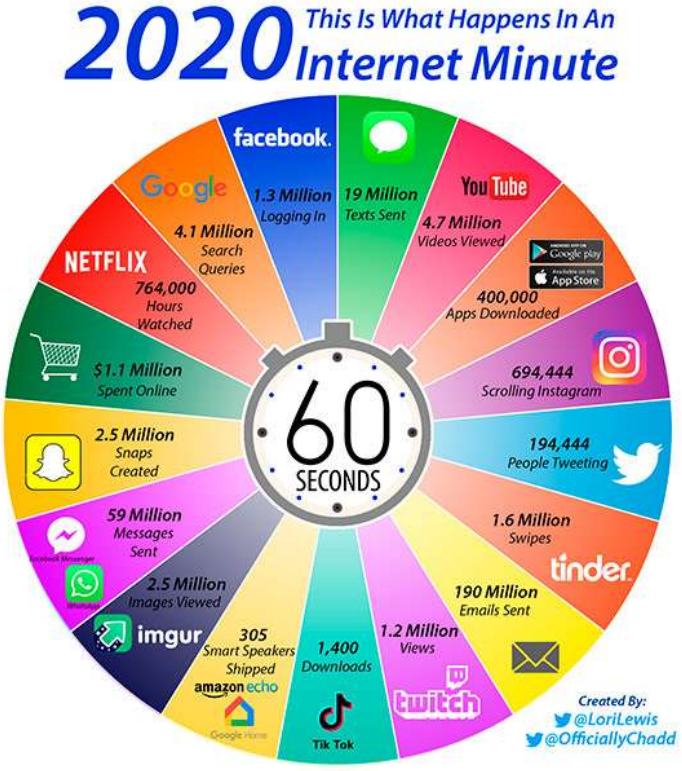
- Agost 2015 1 bilió d'usuaris actius
- Juny 2017 1'32 bilions d'usuaris actius cada dia.
- Cada minut s'envien 31,25 milions de missatges i es visualitzen 2.77 milions de vídeos.

A Youtube es pugen 300hores de vídeos diaris.

Actualment menys del 0,5% de les dades s'estan analitzant.




2018 This Is What Happens In An Internet Minute
facebook
973,000 Logins
18 Million Text Messages
YouTube
4.3 Million Videos Viewed
375,000 Apps Downloaded
174,000 Scrolling Instagram
481,000 Tweets Sent
1.1 Million Swipes
tinder
187 Million Emails Sent
936,073 Views
twitch
67 Voice-First Devices Shipped
amazon echo
38 Million Messages
25,000 GIFs Sent via Messenger
2.4 Million Snaps Created
$862,823 Spent Online
NETFLIX
266,000 Hours Watched
Google
3.7 Million Search Queries
60 SECONDS
Created By:
@LoriLewis
@OfficiallyChadd


2020 This Is What Happens In An Internet Minute
facebook
1.3 Million Logging In
19 Million Texts Sent
YouTube
4.7 Million Videos Viewed
400,000 Apps Downloaded
694,444 Scrolling Instagram
194,444 People Tweeting
1.6 Million Swipes
tinder
190 Million Emails Sent
1.2 Million Views
twitch
1,400 Downloads
Tik Tok
305 Smart Speakers Shipped
amazon echo
Google Home
2.5 Million Images Viewed
imgur
59 Million Messages Sent
2.5 Million Snaps Created
$1.1 Million Spent Online
NETFLIX
764,000 Hours Watched
Google
4.1 Million Search Queries
60 SECONDS
Created By:
@LoriLewis
@OfficiallyChadd

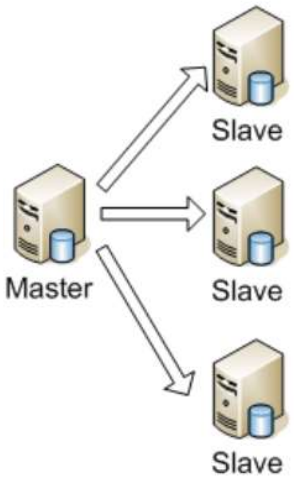
1.11. Infraestructures per el Big Data

1.11.1. Concepte de clúster i rèplica.

Conjunt de màquines interconnectades i capaç de treballar en paral·lel.

Normalment segueixen una arquitectura de *master-slave* composada per un node mestre (*master*) i varis nodes esclau (*workers*).

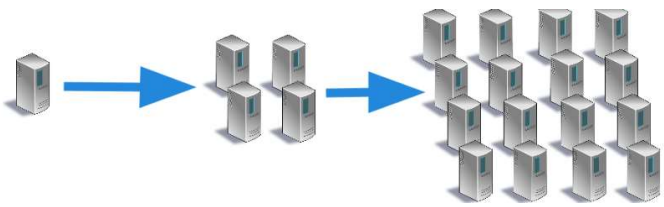
El *master* reparteix les tasques i les envia en els *workers*.



Cal tenir alta disponibilitat. El fet de que un node caigui no ha de provocar una caiguda. El procés que havia de desenvolupar aquell node s'enviarà a un altre.

Amb la **replicació** tindrem les dades repartides en vàries màquines i zones per i cada dada estarà replicada en més d'un node.

L'**escalabilitat horitzontal.** Podem augmentar la capacitat del nostre clúster simplement afegint nous nodes.



1.11.2. Tipus d'infraestructures

Servidors virtuals

Una màquina física que els seus recursos es comparteixen en diferents màquines virtuals.

Servidors en el núvol: Es tracta d'un servidor virtual altament escalable que abstreu el servidor físic. Aquests servidors estan recolzats per una infraestructura redundada. Normalment es paguen per ús i servei.

- Escalabilitat
- Tolerancia a fallades
- Eines de gestió

Exemples: AWS (Amazon Web Services), Microsoft Azure, GPC (Google Cloud Computing)

1.11.3. Capes que ens trobem en un sistema BigData

Capa de monitorització i seguretat que són horitzontal.

DataSource (Els orígens de dades) → Cap d'ingesta → Capa d'emmagatzematge

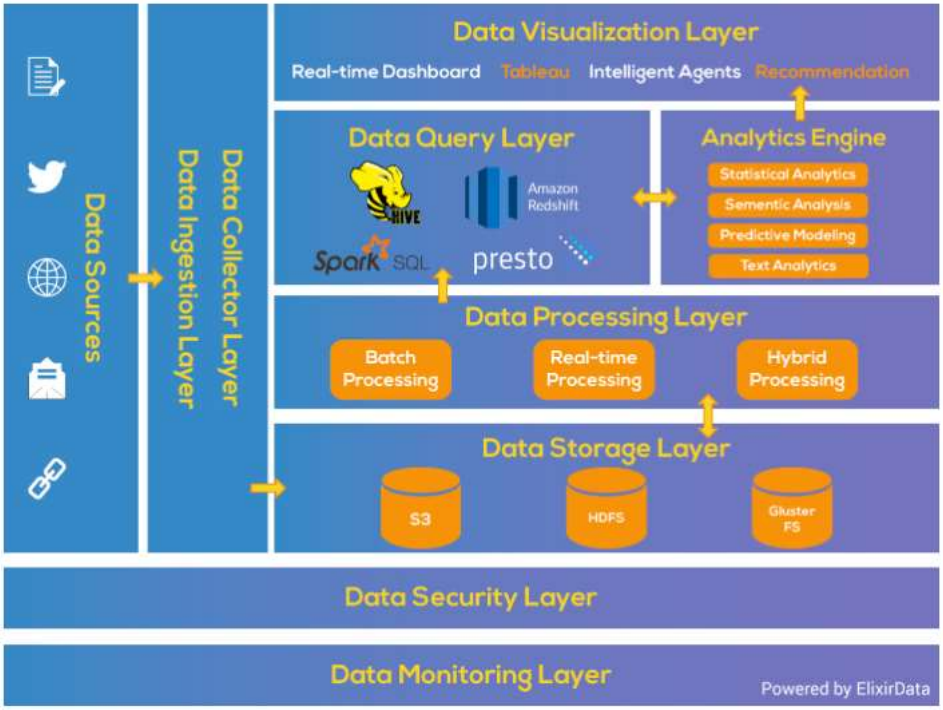
Capa de processat: Processament en diferit (nocturn), processament en temps real , processament híbrid

Capa analística (models predictius, estadística)

Capa de consultes

Capa de visualització.

Capes estan relacionades amb les Fases.

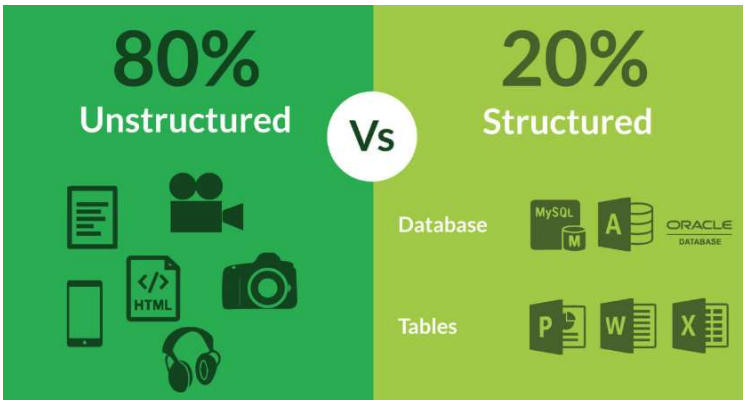


Les capes les podem relacionar amb les 4 fases:

- **Fase 1 Ingesta:** Data Sources, Data Collector, Data Ingestion
- **Fase 2 Persistència:** Data Storage Layer
- **Fase 3 Processament:** Data ProcessingLayer, Data Query Layer, Analytics Engine.
- **Fase 4 Visualització:** Data Visualization Layer.

1.11.4. Emagatzematge de les dades

Com guardem la informació? SQL (relacional), noSQL, etc...



Les dades del BigData la majoria de dades són desestructurades, però això no vol dir que les guardem en BD relacionals.

Si tenim poques relacions amb les dades, necessitat d'escalabilitat si que podem pensar en **NoSQL**.

Apache HBASE: BD intrigada amb Hadoop

Cassandra, Redis: BD de clau valor

MongoDB i CouchDB: documental

Neo4j: BD orientada a grafs

En tots els projectes necessitarem parts de dades **SQL**. Això ens permet d'una forma fàcil realitzar consultes complexes necessàries per l'anàlisi,. tot i que actualment molts sistemes NoSQL.

Per exemple una BD SQL per guardar dades del tipus tarifes, preus del mercat, , relacionats amb les tarifes, etc.. i una BD Cassandra per anem guardant les dades que anem calculant.