INTRODUCCIÓ A LES BASES DE DADES

1 Les dades: conceptes introductoris

Les dades que s'utilitzen en els sistemes d'informació (SI) s'acostumen a emmagatzemar en **bases de dades (BD)**. Per a poder parlar i raonar amb certa propietat sobre les BD, ens convindrà tenir clar què són les **dades** i la **informació**, abstraccions que els informàtics representem físicament sobre discs magnètics, CD-ROM, cartutxos de cinta, etc. Amb aquest objectiu, haurem d'adquirir algunes nocions teòriques fonamentals i disposar d'eines formals en què basar-nos.

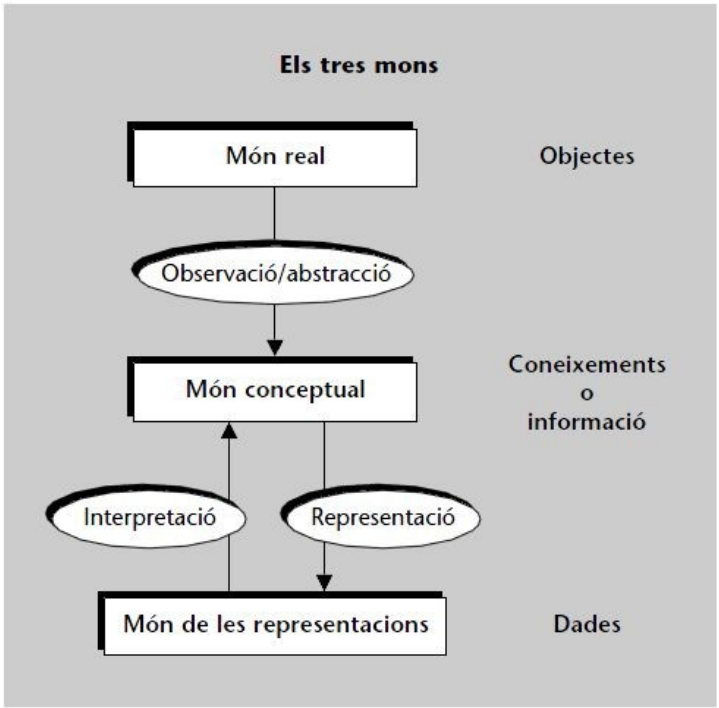
1.1 Els tres mons: el real, el conceptual i el de les representacions

Per a tenir un marc on situar els termes i els conceptes que explicarem en l'assignatura, distingirem tres àmbits diferents:

• El món real amb els objectes del nostre interès.

• El món de les conceptualitzacions lògiques.

• El món de les representacions informàtiques.



1.1.1 La realitat: els objectes

Com a informàtics que hem d'analitzar o construir un sistema d'informació (SI) determinat, ens cal conèixer el món real al qual aquest SI ha de fer referència o modelitzar; així, el nostre món real podrà ser un hospital, una empresa distribuïdora de productes alimentaris, la matriculació dels alumnes d'una universitat, etc. Així doncs podem definir el món real com:

El **món real**, la part de la realitat que ens interessa, és el que percebem amb els nostres sentits i és compost per objectes concrets, físics o no.

Com que treballem en el camp dels SI, els mons reals que ens interessen són les organitzacions: empreses, institucions, etc.

Exemples d'objectes concrets poden ser el malalt Joan Garcia, el llit 34 de la segona planta, el magatzem de Sòria, el camió B-3452-AG, l'alumna Maria Pi, l'assignatura Química I, la malaltia meningitis, la devolució d'una comanda concreta, un determinat accident de trànsit, aquestos són alguns exemples d'objectes que pertanyen al món real.

1.1.2. Les concepcions: la informació

Observant el món real, els humans som capaços de deduir-ne coneixements, informació. L'observació dels objectes del món real ens porta a fer-ne l'anàlisi i la síntesi; després, n'obtenim abstraccions, en fem classificacions (podem saber que dos objectes són de la mateixa classe malgrat que siguin diferents), en deduïm propietats i interrelacions, etc.

Per exemple l'observació del camp de la matriculació en una universitat ens permet reconèixer diferents classes o tipus d'objectes, com ara l'alumne o l'assignatura. Deduïm que tot alumne tindrà les propietats (són abstraccions) data de naixement, DNI, nom, etc. i així obtenim informacions com les següents: l'alumne de nom Joan Garcia té el DNI 34.567.854 i el seu any de naixement és el 1979.

El conjunt dels coneixements obtinguts observant un món real, l'anomenem **món conceptual o món de les concepcions**. En l'esfera de les concepcions construïm un model abstracte, conceptual, del món real, i això ens ajuda a raonar i a expressar-nos.

El procés d'observació/abstracció és bàsicament un procés per a modelitzar l'estructura, les propietats i el funcionament de la realitat.

De fet, hi ha diferències entre coneixement i informació. La informació és un coneixement transmissible, és a dir, que es pot representar. **Els únics coneixements que ens interessaran aquí són, doncs, les informacions**.

Un mateix món real pot ser vist, concebut, modelitzat, de maneres diferents per diferents observadors (fins i tot per un mateix observador) **segons el seu entorn o marc de referència**.

Per exemple, no veu de la mateixa manera l'àmbit de la gestió d'un centre universitari un professor que un administratiu de secretaria. Tenen marcs de referència diferents. No estan interessats en els mateixos conceptes. El professor, a diferència de l'administratiu, no necessitarà conèixer l'import de la matrícula, no voldrà distingir les abstraccions alumne amb beca i alumne sense beca. Els professors estaran interessats en la qualificació numèrica, mentre que el servei administratiu potser només tindrà en compte la forma textual de la qualificació.

Veiem, doncs, que en el pas del món real al de les concepcions hi ha pluralisme. L'observació i l'anàlisi d'una mateixa part d'una organització o empresa poden portar a concepcions diferents, totes igualment vàlides i que poden haver de coexistir.

1.1.3. Les representacions: les dades

El món de les concepcions o dels coneixements és un món mental. Però per a poder treballar amb aquests coneixements i poder comunicar-los, necessitem projectar els pensaments a l'exterior representant-los físicament d'alguna manera. Aquest és el món de les representacions.

Nosaltres ens ocuparem de les representacions informàtiques, i parlarem de dades, fitxers, bases de dades, registres, camps, bytes, discs, etc.

Donem el nom de **dades** a les representacions físiques dels coneixements que tenim dels objectes del món real. El pas dels coneixements a les dades, el pas d'una concepció a una representació informàtica, no és automàtic.
És un procés humà, un procés de disseny.

Òbviament, en aquest cas, com en el cas del pas del món real al de les concepcions, també hi ha pluralisme. Un mateix conjunt de coneixements es pot representar de moltes maneres, per exemple: en forma de base de dades relacional o com a fitxers tradicionals, amb vectors o sense, amb longitud fixa o variable, amb codificació ASCII o EBCDIC, etc. Una visió o concepció del món real d'un hospital, d'una universitat o d'una distribuïdora de productes podrà ser representada de moltes maneres sobre suports físics informàtics. Sense cap mena de dubte, les feines més importants de l'analista/dissenyador de SI o d'aplicacions informàtiques són les següents:

1) Analitzar els objectes del món real, fer-ne abstraccions i obtenir-ne una concepció lògica, un model conceptual.

2) Dissenyar una representació informàtica concreta que es pugui tractar eficientment. El fet de saber observar la realitat, fer-ne les abstraccions lògiques més escaients, l'habilitat per a l'anàlisi i la síntesi, esdevenen les qualitats fonamentals que ha de tenir el desenvolupador de SI. I aquestes qualitats s'han d'educar i conrear.

Es poden fer molts dissenys diferents de representació informàtica corresponents a un únic model conceptual d'una realitat. Tots poden representar la mateixa realitat, però tindran una eficiència diferent segons la utilització que se'n faci

Les dades emmagatzemades en les que hem de treballar han de presentar les següents propietats:

- No han de ser efímeres
- Estructurades
- Operacionals i transaccionals
- Amb sentit semàntic
- Íntegres

1.1.4. La interpretació

Acabem de veure el camí que ens porta de la realitat als coneixements, i d'aquests a les dades o representacions. Però ens farà falta poder interpretar la representació. El procés invers al de representació, l'anomenem **interpretació**.

Si veiem una dada, una representació extreta d'una base de dades relativa a la matriculació d'alumnes, que consta de la sèrie de símbols: 1999, no en podrem obtenir cap informació si no sabem si representa l'any de matriculació, l'any de naixement, l'import de la matrícula, el número de la matrícula, etc., i en qualsevol cas no sabrem de quin alumne concret (de quin objecte del món real) es tracta.

Veiem, doncs, que per a poder interpretar les dades s'ha de saber, a més, a qui i a què (a quins conceptes) fan referència. **Hem dit que una informació és un coneixement que es pot representar, però ara, mirant el camí invers, podrem dir que la informació és el significat que donem a les dades.**

2 Els sistemes d'informació

La societat actual necessita conèixer tota la informació existent d'una manera ràpida i eficient. De la mateixa manera, necessita emmagatzemar i gestionar aquesta informació de manera que aquesta pugui ser mostrada, emmagatzemada, recuperada i actualitzada en qualsevol moment. Una bona gestió de la informació facilita el desenvolupament econòmic i social. Així doncs podem definir un sistema d'informació com:

Conjunt de persones, procediments i equips dissenyats, construïts i gestionats per tractar la informació d'acord a les necessitats de l'empresa.

Un sistema d'informació està format per diversos tipus d'elements: recursos físics, RRHH, protocols o normes d'ús i transmisió de fluxes d'informació. Quan la gestió d'un sistema d'informació és realitzada parcial o totalment per ordinadors s'anomena **Sistema Informàtic**, així es pot dir que:

Un sistema informàtic és un conjunt d'elements tecnològics relacionats que cooperen per a la consecució d'un determinat objectiu professional, personal o empresarial.

En resum, els tres tipus d'elements que formen un sistema informàtic són:

- Recursos físics
- Recursos humans
- Normes o protocols

2.1 Importància de les dades en els sistemes informàtics

Per a emmagatzemar la informació estructurada de l'empresa en un sistema informàtic, és necessari, per una banda el seu disseny lògic o l'aparença que presenten les dades quan són mostrades als usuaris del sistema. Per altra banda, s'ha de conèixer el disseny físic o mode d'emmagatzemament de les dades en un dispositiu d'emmagatzemament (disc dur, ...)

Segons el model empleat en el disseny físic d'emmagatzemament es poden utilitzar dos sistemes:

Sistemes basats en fitxers de dades: emmagatzemen les dades com elements independents separats entre sí. P.e., Cobol, C,...

Sistemes basats en la gestió de bases de dades: tots els elements es guarden en un o varis fitxers interrelacionats entre ells. P.e.: Access, Informix, Oracle, SQL Server, MySQL...

3 Concepte i origen de les BD i dels SGBD

Abans que res definirem el concepte d'entitat:

Una **entitat** és una classe d'objectes (persona, concepte, succés,...) del món real que és distingible de tots els demés objectes i sobre el que es desitja emmagatzemar informació. Cada objecte que pertany a la classe s'anomena ocurrència o instància de l'entitat.

Les aplicacions informàtiques dels anys seixanta acostumaven a ser totalment per lots (batch) i estaven pensades per a una feina molt específica relacionada amb molt poques entitats tipus.

Cada aplicació (una o vàries cadenes de programes) utilitzava fitxers de moviments per a actualitzar (creant una còpia nova) i/o per a consultar un o dos fitxers mestres o, molt rarament, més de dos. Cada programa tractava com a màxim un fitxer mestre, que solia estar sobre cinta magnètica i, en conseqüència, es treballava amb accés seqüencial. Cada vegada que s'hi volia afegir una aplicació que requeria l'ús d'algunes de les dades que ja existien i d'altres de noves, es dissenyava un fitxer nou amb totes les dades necessàries (cosa que provocava redundància) per a evitar que els programes haguessin de llegir molts fitxers.

A mesura que es van anar introduint les línies de comunicació, els terminals i els discs, es van anar escrivint programes que permetien a diversos usuaris consultar on-line i simultàniament els mateixos fitxers. Més endavant va anar sorgint la necessitat de fer les actualitzacions també on-line.

A mesura que s'integraven les aplicacions, es van haver d'interrelacionar els seus fitxers i va caldre eliminar la redundància. El nou conjunt de fitxers s'havia de dissenyar de manera que estiguessin interrelacionats i, al mateix temps, les informacions redundants, com el nom i l'adreça dels clients o el nom i el preu dels productes, que figuraven en els fitxers de més d'una de les aplicacions, ara havien d'estar en un sol lloc.

L'accés on-line i la utilització eficient de les interrelacions exigien estructures físiques que donessin un accés ràpid com, per exemple, els índexs, les multillistes, les tècniques de hashing, etc.

Aquests conjunts de fitxers interrelacionats, amb estructures complexes i compartits per diversos processos simultàniament (els uns on-line i els altres per lots), al principi van rebre el nom de Data Banks i després, al començament dels anys setanta, el de Data Bases. Aquí els anomenem bases de dades (BD).

El programari de gestió de fitxers era massa elemental per a donar satisfacció a totes aquestes necessitats. Per exemple, el tractament de les interrelacions no estava previst; no era possible que diversos usuaris actualitzessin dades simultàniament, etc. La utilització d'aquests conjunts de fitxers per part dels programes d'aplicació era excessivament complexa, de manera que, especialment durant la segona meitat dels anys setanta, van anar sortint al

mercat programaris més sofisticats, els anomenats Data Base Management Systems, que aquí denominem sistemes de gestió de BD (SGBD).

Per tot el que hem dit fins ara, podríem definir el terme BD; **una base de dades d'un SI és la representació integrada dels conjunts d'entitats instància** corresponents a les diferents entitats tipus del SI i de les seves interrelacions. Aquesta representació informàtica, o conjunt estructurat de dades, ha de poder ser utilitzada de manera compartida per molts usuaris de tipus diversos.

En altres paraules, una base de dades és un conjunt estructurat de dades que representa entitats i les seves interrelacions. La representació serà única, integrada, malgrat que ha de permetre utilitzacions diverses i simultànies.

Així també podem definir un sistema gestor de bases de dades (SGBD) com:

Un conjunt de programes que permet als usuaris crear i mantenir una base de dades. Per tant, el SGBD és un sistema de software de propòsit general que facilita el procés de definir, construir i manipular les bases de dades per a diverses aplicacions.

3.1 Els fitxers tradicionals i les BD

Encara que de manera molt simplificada, podríem enumerar les principals diferències entre els fitxers tradicionals i les BD tal com s'indica tot seguit:

1) Entitats tipus:
• Fitxers: tenen registres d'una sola entitat tipus.
• BD: tenen dades de diverses entitats tipus.

2) Interrelacions:
• Fitxers: el sistema no interrelaciona fitxers.
• BD: el sistema té previstes eines per a interrelacionar entitats.

3) Redundància:
• Fitxers: es creen fitxers a la mida de cada aplicació, amb totes les dades necessàries encara que algunes siguin redundants respecte d'altres fitxers.
• BD: totes les aplicacions treballen amb la mateixa BD i la integració de les dades és bàsica,de manera que s'evita la redundància.

4) Usuaris:
• Fitxers: serveixen per a un sol usuari o una sola aplicació. Donen una sola visió del món real.
• BD: és compartida per molts usuaris de tipus diversos. Ofereix diverses visions del món real.

3.2 Avantatges i inconvenients de les bases de dades sobre els sistemes de fitxers

Els avantatges de les bases de dades sobre els sistemes de gestió de dades basats en fitxers són les següents:

- Control centralitzat de les dades, dels usuaris i dels seus accessos, el que facilita la disponibilitat a aquelles persones que posseeixen permisos d'accés sobre els diferents objectes, sense dependre de diferents propietaris de porcions físiques de l'aplicació i augmentant els nivells de seguretat del sistema degut als diferents nivells de seguretat que s'implementen en les bases de dades comercials.

- A l'estar integrats tots els elements de la base de dades s'obtenen els següents avantatges: major coherència de les dades a l'eliminar la inconsistència de dades, disminuir la redundància i més fiabilitat en la gestió d'errors.

- Emmagatzemament físic eficient (més compacte) i independència de l'equip lògic en que s'instal·li la base de dades.

- Manteniment de la integritat de la base de dades encara que es comparteixin dades entre els diferents usuaris o es facin modificacions.

- Emmagatzemament dels requisits d'anàlisi i especificacions de la base de dades al guardar les definicions dels objectes en un catàleg.

Els inconvenients de les bases de dades sobre els sistemes basats en fitxers són:

- Cost inicial elevat degut a la compra de programes de gestió de bases de dades. A més, hi haurà que actualitzar els equips existents sobre els quals instal.lar el software corresponent, així com formar personal especialitzat en utilitzar aquest nou sistema.

- Encara que la velocitat de processos d'informació i obtenció de resultats és immediat, la rendibilitat que el sistema ofereix és a curt o llarg termini.

- No existeix un únic disseny de bases de dades que permeti donar solució a un problema empresarial, de manera que diferents analistes o administradors de bases de dades poden obtenir objectes diferents per aconseguir el mateix resultat, motiu pel qual poden haver problemes de redundància.

- Cap dels sistemes de gestió de bases de dades existents al mercat acompleix totes les especificacions i a més, posseeixen característiques afegides que no són compatibles entre elles.

4 Evolució dels SGBD

Per a entendre millor què són els SGBD, farem un repàs de la seva evolució des dels anys seixanta fins als nostres dies.

4.1 Els anys seixanta i setanta: sistemes centralitzats

Els SGBD dels anys seixanta i setanta (l'IMS d'IBM, l'IDS de Bull, el DMS d'Univac, etc.) eren **sistemes totalment centralitzats**, com correspon als sistemes operatius d'aquells anys i al maquinari per al qual estaven fets: un gran ordinador per a tota l'empresa i una xarxa de terminals sense intel·ligència ni memòria.

Els **primers SGBD** –als anys seixanta encara no se'ls anomenava així– estaven orientats a facilitar la utilització de grans conjunts de dades en què les interrelacions eren complexes. Aquests sistemes treballaven exclusivament per lots (*batch*).

En sortir els terminals de teclat, connectats a l'ordinador central mitjançant una línia telefònica, es comencen a construir grans **aplicacions *on-line* transaccionals (OLTP)**. Els SGBD estaven íntimament lligats al programari de comunicacions i de gestió de transaccions.

Encara que per a escriure els **programes d'aplicació** s'utilitzaven llenguatges d'alt nivell com el Cobol o el PL/I, es disposava d'instruccions i subrutines especialitzades per a tractar les BD que requerien que el programador conegués molts detalls del disseny físic, i que feien que la programació fos molt complexa.

Els programes, com que estaven lligats al nivell físic, s'havien de modificar contínuament quan es feien canvis en el disseny i l'organització de la BD. La preocupació bàsica era maximitzar el rendiment: el temps de resposta i les transaccions per segon.

4.2 Els anys vuitanta: SGBD relacionals

Els **ordinadors *minis***, en primer lloc, i després els **ordinadors *micros***, van estendre la informàtica a pràcticament totes les empreses i institucions. Això exigia que el desenvolupament d'aplicacions fos més senzill. Els SGBD dels anys setanta eren massa complexos i inflexibles, i només els podia utilitzar un personal molt qualificat.

L'aparició dels **SGBD relacionals** suposa un avenç important per a facilitar la programació d'aplicacions amb BD i per a aconseguir que els programes siguin independents dels aspectes físics de la BD.

Tot això fa que s'estengui l'ús dels SGBD. L'estandardització, l'any 1986, del **llenguatge SQL** va produir una autèntica explosió dels SGBD relacionals.

4.3 Els anys noranta: distribució, C/S i 4GL

En acabar la dècada dels vuitanta, els SGBD relacionals ja eren utilitzats pràcticament a totes les empreses. Malgrat tot, fins a la meitat dels noranta, quan s'ha necessitat un rendiment elevat s'han seguit utilitzant els SGBD prerelacionals.

Al final dels vuitanta i començament dels noranta, les empreses s'han trobat que els seus departaments han anat comprant ordinadors departamentals i personals, i han anat fent aplicacions amb BD. El resultat ha estat que en el si de l'empresa hi ha nombroses BD i diversos SGBD de tipus o proveïdors diferents. Aquest fenomen de **multiplicació de les BD i dels SGBD** s'ha vist incrementat per la febre de les fusions d'empreses.

La necessitat de tenir una visió global de l'empresa i d'interrelacionar diferents aplicacions que utilitzen BD diferents, juntament amb la facilitat que donen les xarxes per a la intercomunicació entre ordinadors, ha portat als SGBD actuals, els quals permeten que un programa pugui treballar amb diferents BD com si es tractés d'una de sola. És el que es coneix com a **base de dades distribuïda**.

Aquesta distribució ideal s'aconsegueix quan les diferents BD són suportades per una mateixa marca de SGBD, és a dir, quan hi ha homogeneïtat. Però si els SGBD són heterogenis, això no és tan senzill. Actualment, gràcies principalment a l'estandardització del llenguatge SQL, els SGBD de marques diferents poden donar-se servei els uns als altres i col·laborar per a donar servei a un programa d'aplicació. Però, en general, en els casos d'heterogeneïtat no s'arriba a poder donar en el programa que els utilitza l'aparença que es tracta d'una única BD.

La tecnologia que es fa servir habitualment per a distribuir dades és la que es coneix com a **entorn (**o arquitectura) **client/servidor (C/S)**. Tots els SGBD relacionals del mercat hi han estat adaptats. La idea del C/S és senzilla. Dos processos diferents, que s'executen en un mateix sistema o en sistemes separats, actuen de manera que un fa de **client** o peticionari d'un servei i l'altre fa de **servidor** o proveïdor del servei.

L'èxit de les BD, fins i tot en petits sistemes personals, ha portat a l'aparició dels ***Fourth Generation Languages*** **(4GL)**, llenguatges molt fàcils i potents, especialitzats en el desenvolupament d'aplicacions basades en BD. Donen moltes facilitats per a definir, generalment de manera visual, diàlegs per a introduir, modificar i consultar dades en entorns C/S.

4.4 Tendències actuals

Avui dia, els SGBD relacionals estan en plena transformació per a adaptar-se a tres tecnologies d'èxit recent fortament interrelacionades: la multimèdia, la d'orientació a l'objecte (OO), Internet i el web i el Data Warehousing.

5 Objectius i serveis dels SGBD

Els SGBD que actualment són al mercat pretenen satisfer tot un conjunt d'objectius directament deduïbles del que hem explicat fins ara. Aquestos objectius son:

5.1 Consultes no predefinides i complexes

L'objectiu fonamental dels SGBD és permetre que es facin **consultes no predefinides** (*ad hoc*) **i complexes**.

Els usuaris podran fer consultes de qualsevol tipus i complexitat directament a l'SGBD. Aquest haurà de respondre immediatament sense que estiguin preestablertes, és a dir, sense que s'hagi d'escriure, compilar i executar un programa específic per a cada consulta.

La solució estàndard per a assolir aquest doble objectiu (consultes no predefinides i complexes) és el **llenguatge SQL**

5.2 Flexibilitat i independència

La complexitat de les BD i la necessitat d'anar-les adaptant a l'evolució del SI fan que un objectiu bàsic dels SGBD sigui donar **flexibilitat als canvis.**

Per altra banda interessa també obtenir la **màxima independència** possible entre les dades i els processos usuaris perquè es puguin fer tot tipus de canvis tecnològics i variacions en la descripció de la BD, sense que s'hagin de modificar els programes d'aplicació ja escrits ni canviar la manera d'escriure les consultes (o actualitzacions) directes.

Per a aconseguir aquesta independència, tant els usuaris que fan consultes (o actualitzacions) directes, com els professionals informàtics que escriuen programes que les porten incorporades, **han de poder desconèixer les característiques físiques de la BD amb la qual treballen**. No necessiten saber res sobre el suport físic, ni estar al corrent de quin SO s'utilitza, quins índexs hi ha, si es té compressió de dades o no, etc.

D'aquesta manera, es poden fer canvis de tecnologia i canvis físics per a millorar el rendiment sense afectar ningú. Aquest tipus d'independència rep el nom d'**independència física de les dades**.

Però amb la independència física no en tenim prou. Volem també que els usuaris (els programadors d'aplicacions o els usuaris directes) no hagin de fer canvis quan es modifica la descripció lògica, l'esquema, de la BD, com ara quan s'hi afegeix/suprimeix entitats o interrelacions, o en afegir/suprimir atributs, etc.

I encara més: **volem que els diferents processos usuaris puguin tenir diferents visions lògiques d'una mateixa BD, i que aquestes visions es puguin mantenir com més independents millor de la BD i entre elles**. Aquest tipus d'independència s'anomena **independència lògica de les dades**, i dóna flexibilitat i elasticitat als canvis lògics.

Exemples d'independència lògica

1) El personal administratiu de secretaria podria tenir una visió de l'entitat *alumne* com si no existís l'atribut *nota*. Però els usuaris professors (o els programes dirigits a ells) podrien tenir una visió en què existís l'atribut *nota* però no l'atribut *data de pagament*.
2) Decidim ampliar la longitud de l'atribut *nom* i l'augmentem de trenta a cinquanta caràcters, però els programes que ja tenim escrits no caldria modificar-los si no ens importa que els valors obtinguts tinguin només els primers trenta caràcters del nom.

5.3 Problemes de la redundància

En el món dels fitxers tradicionals, cada aplicació utilitzava el seu fitxer. Però com que hi havia molta coincidència de dades entre aplicacions, hi havia molta redundància entre els fitxers. Un dels objectius dels SGBD és **facilitar l'eliminació de la redundància**.

Segurament penseu que el problema de la redundància és l'espai perdut. Antigament, quan el preu del byte de disc era elevadíssim, això era un problema greu, però actualment gairebé mai no ho és. Quin problema hi ha, doncs?

Simplement, allò que tots hem patit més d'una vegada; si tenim apuntada alguna cosa en dos llocs diferents no passarà gaire temps fins que les dues anotacions deixin de ser coherents, perquè haurem modificat l'anotació en un dels llocs i ens haurem oblidat de fer-ho en l'altre.

El veritable problema és, doncs, el greu risc d'inconsistència o incoherència de les dades, és a dir, la **pèrdua d'integritat** que les actualitzacions poden provocar quan hi ha redundància.

Convindria, doncs, evitar la redundància. En principi, ens convé fer que una dada només figuri una sola vegada a la BD. Però això no sempre serà cert. Per exemple, per a representar una interrelació entre dues entitats, se sol repetir un mateix atribut a les dues entitats, perquè l'una faci referència a l'altra.

Un altre exemple podria ser quan per raons de fiabilitat, disponibilitat o costos de comunicacions tenim rèpliques de les dades.

Els SGBD han de permetre que el dissenyador defineixi dades redundants, però llavors hauria de ser el mateix SGBD el que fes automàticament l'**actualització de les dades** a tots els llocs on estiguessin repetides.

La duplicació de dades és el tipus de redundància més habitual, però també tenim redundància quan guardem a la BD dades derivades (o calculades) a partir d'altres dades de la mateixa BD. D'aquesta manera podem respondre ràpidament a consultes globals, ja que ens estalviem la lectura de gran quantitat de registres.

En els casos de dades derivades, perquè el resultat del càlcul es mantingui consistent amb les dades elementals, és necessari refer el càlcul sempre que aquestes són modificades. Si el nou càlcul l'ha de fer l'usuari (ja sigui programador o no), se'n pot descuidar; així, doncs, convindrà que el mateix SGBD el faci automàticament

5.4 Integritat de les dades

Ens interessarà que els SGBD assegurin el **manteniment de la qualitat de les dades** en qualsevol circumstància. Acabem de veure que la redundància pot provocar pèrdua d'integritat de les dades, però no n'és l'única causa possible. Es podria perdre la correcció o la consistència de les dades per moltes altres raons: errades de programes, errades d'operació humana, avaria de disc, transaccions incompletes per tall de l'alimentació elèctrica, etc.

En el subapartat anterior hem vist que podrem dir a l'SGBD que ens porti el control de les actualitzacions en el cas de les redundàncies per a garantir la integritat. De la mateixa manera podrem donar-li altres **regles d'integritat** –o restriccions– perquè asseguri que els programes les compleixen quan efectuen les actualitzacions.

Quan l'SGBD detecti que un programa vol fer una operació que va contra les regles establertes en definir la BD, no li ho haurà de permetre, i li haurà de tornar un estat d'error.

En dissenyar una BD per a un SI concret i escriure'n l'esquema, no només definirem les dades, sinó també les regles d'integritat que volem que l'SGBD faci complir.

Per exemple, podrem declarar que l'atribut *DNI* ha de ser clau o que la *data de naixement* ha de ser una data correcta i, a més, s'ha de complir que l'alumne no pugui tenir menys de divuit anys ni més de noranta-nou, o que el nombre d'alumnes matriculats d'una assignatura no sigui superior a vint-i-set,etc.

A part de les regles d'integritat que el dissenyador de la BD pot definir i que l'SGBD entendrà i farà complir, el mateix SGBD té regles d'integritat inherents al model de dades que utilitza i que sempre s'acompliran. Són les anomenades **regles d'integritat del model**. Les regles definibles per l'usuari són les **regles d'integritat de l'usuari**.

El concepte d*'integritat de les dades* va més enllà de la prevenció que els programes usuaris emmagatzemin dades incorrectes. En casos d'errades o

desastres, també podríem perdre la integritat de les dades. L'SGBD ens ha de donar les eines per a poder reconstruir o restaurar les dades malmeses.

Els **processos de restauració** (*restore* o *recovery*) de què tot SGBD disposa poden reconstruir la BD i donar-li l'estat consistent, correcte, anterior a l'incident. Això s'acostuma a fer gràcies a l'obtenció de còpies periòdiques de les dades (s'anomenen *còpies de seguretat* o *backup*) i mitjançant el manteniment continu d'un diari (*log*) on l'SGBD va anotant totes les escriptures que es fan a la BD.

5.5 Concurrència d'usuaris

Un objectiu fonamental dels SGBD és permetre que diversos usuaris puguin accedir concurrentment a la mateixa BD.

Quan els accessos concurrents són tots de lectura, és a dir, quan la BD només es consulta, el problema és simplement un problema de rendiment causat per les limitacions dels suports de què es disposa: pocs mecanismes d'accés independents, moviment del braç i de gir del disc massa lents, *buffers* locals massa petites, etc.

Quan un usuari o més d'un estan actualitzant les dades, es poden produir **problemes d'interferència** que tinguin com a conseqüència l'obtenció de dades errònies i la pèrdua d'integritat de la BD.
Per a tractar els accessos concurrents, els SGBD fan servir el concepte de transacció de BD, concepte d'especial utilitat per a tot allò que fa referència a la integritat de les dades.

Anomenem **transacció de BD** o, simplement, **transacció** un conjunt d'operacions simples que s'executen com una unitat. Els SGBD han d'aconseguir que el conjunt d'operacions d'una transacció mai no s'executi parcialment. O s'executen totes o no se n'executa cap.

Exemples de transaccions

1) Imaginem un programa pensat per a fer l'operació de transferència de diners d'un compte X a un altre Y. Suposem que la transferència fa dues operacions: en primer lloc el càrrec a X i després l'abonament a Y. Aquest programa s'ha d'executar de manera que es facin totes dues operacions o cap, ja que si per qualsevol raó (per exemple, per interrupció del flux elèctric) el programa executés només el càrrec de diners a X sense abonar-los a Y, la BD quedaria en un estat incorrecte. Volem que l'execució d'aquest programa sigui tractada per l'SGBD com una transacció de BD.

2) Un altre exemple de programa que voldríem que tingués un comportament de transacció podria ser un que augmentés el 30% la nota de tots els alumnes. Si només s'augmentés la nota a uns quants alumnes, la BD quedaria incorrecta.

Per a indicar a l'SGBD que donem per acabada l'execució de la transacció, el programa farà servir l'operació COMMIT. Si el programa no pot acabar normalment, és a dir, si el conjunt d'operacions s'ha fet només parcialment, l'SGBD haurà de desfer tot el que la transacció ja hagi fet. Aquesta operació s'anomena ROLLBACK.

Acabem de veure la utilitat del concepte de *transacció* per al manteniment de la integritat de les dades en cas d'interrupció d'un conjunt d'operacions lògicament unitari. Però entre transaccions que s'executen concurrentment es poden produir problemes d'interferència que facin obtenir resultats erronis o que facin perdre la integritat de les dades.

Conseqüències de la interferència entre transaccions

1) Imaginem que una transacció que transfereix diners de X a Y s'executa concurrentment amb una transacció que mira el saldo dels comptes Y i X, en aquest ordre, i en mostra la suma. Si l'execució concurrent de les dues transaccions casualment és tal que la transferència s'executa entre l'execució de les dues lectures de la transacció de suma, pot produir resultats incorrectes. A més, si els decideix escriure a la BD, aquesta quedarà inconsistent

2) Si simultàniament amb el generós programa que augmenta la nota dels alumnes en un 30%, s'executa un programa que determina la nota mitjana de tots els alumnes d'una determinada assignatura, es podrà trobar alumnes ja gratificats i alumnes no gratificats, cosa que produirà resultats erronis. Aquest és només un exemple de les diferents conseqüències negatives que pot tenir la interferència entre transaccions en la integritat de la BD i en la correcció del resultat de les consultes.

Ens interessarà que l'SGBD executi les transaccions de manera que no s'interfereixin, és a dir, que quedin aïllades les unes de les altres. Per a aconseguir que les transaccions s'executin com si estiguessin aïllades, els SGBD utilitzen diverses tècniques, de les quals la més coneguda és el **bloqueig** (*lock*).
El bloqueig d'unes dades en benefici d'una transacció consisteix a posar limitacions als accessos que les altres transaccions podran fer a aquestes dades.
Quan es provoquen bloqueigs es produeixen esperes, retencions i, en conseqüència, el sistema és més lent. Els SGBD s'esforcen a minimitzar aquests efectes negatius.

5.6 Seguretat

En el camp dels SGBD, el terme *seguretat* se sol utilitzar per a fer referència als temes relatius a la confidencialitat, les autoritzacions, els drets d'accés, etc.

Aquestes qüestions sempre han estat importants en els SI militars, les agències d'informació i en àmbits similars, però durant els anys noranta han anat

adquirint importància en qualsevol SI on s'emmagatzemin dades sobre persones.
Recordeu que a l'Estat espanyol tenim lleis que exigeixen la protecció de la confidencialitat d'aquestes dades.

Els SGBD permeten definir **autoritzacions** o drets d'accés a diferents nivells: al nivell global de tota la BD, al nivell entitat i al nivell atribut. Aquests mecanismes de seguretat requereixen que l'usuari es pugui identificar. S'acostuma a utilitzar codis d'usuari (i grups d'usuaris) acompanyats de contrasenyes (*passwords*), però també s'utilitzen targetes magnètiques, identificació per reconeixement de la veu, etc.

Ens pot interessar emmagatzemar la informació amb una codificació secreta, és a dir, amb **tècniques d'encriptació** (com a mínim s'haurien d'encriptar les contrasenyes).Molts dels SGBD actuals tenen prevista l'encriptació.

Gairebé tots els SGBD del mercat donen una gran varietat d'eines per a la vigilància i l'administració de la seguretat. N'hi ha que, fins i tot, tenen opcions (amb preu separat) per als SI amb unes exigències altíssimes, com ara els militars.

5.7 Altres objectius

Acabem de veure els objectius fonamentals dels SGBD actuals. Però a mesura que els SGBD evolucionen, s'imposen nous objectius adaptats a les noves necessitats i les noves tecnologies. Com ja hem vist, en aquests moments podríem citar com a objectius nous o recents els següents:

1) Servir eficientment els *Data Warehouse*.

2) Adaptar-se al desenvolupament orientat a l'objecte.

3) Incorporar el temps com un element de caracterització de la informació.

4) Adaptar-se al món d'Internet.

6 Arquitectura dels SGBD

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.

Un dels objectius d'un sistema de bases de dades es proporcionar una visió el més abstracta possible de la informació, és a dir, ocultar detalls referents a la forma en que les dades estan organitzades i emmagatzemades als usuaris.

El comité X3 d'estudi ANSI/X3/SPARC del Standard Planning and Requeriments Committee (SPARC) de l'ANSI (American National Standard Institute) s'ocupa de l'estandardització d'elements informàtics, tant software

com hardware. Aquest comitè és el responsable de l'estandardització de l'arquitectura a tres nivells que s'utilitza en el disseny, creació i gestió de bases de dades.

L'objectiu de l'arquitectura de tres esquemes consisteix a formar una separació entre les aplicacions de l'usuari i la base de dades física. En aquesta arquitectura, els esquemes es poden definir en tres nivell; el nivell intern, el nivell conceptual i el nivell extern.

Així, l'arquitectura més utilitzada és la que fa una divisió en funció del sistema en nivells de la base de dades. Així, hem dit que el nivell intern és el nivell més baix on es descriu la informació en funció del sistema en que s'implantarà la base de dades. Per sobre, es troba el nivell conceptual, que representa a alt nivell tota la informació de la base de dades independentment de la màquina en que vagi a utilitzar-se. El nivell extern és el que gestiona la informació des del punt de vista individual de cada usuari.

Vegem amb més detall cadascun d'aquests nivells:

6.1 Nivell intern

El nivell intern (o també anomenat físic) té un esquema intern, que descriu com emmagatzemar físicament les estructures de dades. L'esquema intern utilitza un model físic de les dades i descriu tots els detalls per al seu emmagatzematge, així com els camins d'accés per a la base de dades. Aquest esquema intern és un conjunt de definicions i regles que permeten definir taules i relacionar-les.

Així, en aquest nivell, s'especifiquen els arxius que contenen la informació, la seva organització, la forma d'accedir als registres, el tipus i longitud del registre, els camps que el composen, els camps claus, etc.

6.2 Nivell conceptual

El nivell conceptual descriu l'organització de les dades en la base de dades i les relacions existents entre elles. La descripció d'aquest nivell es realitza mitjançant un esquema conceptual que permet definir: les regles de validació, les relacions entre taules i els camps i les seves propietats. L'esquema conceptual amaga els detalls de les estructures físiques d'emmagatzematge i es concentra en descriure entitats, tipus de dades, vincles, operacions dels usuaris i restriccions.

6.3 Nivell extern

El nivell extern inclou varis esquemes externs o vistes d'usuari. Cada esquema descriu la part de la base de dades que interessa a un grup d'usuaris determinat, i amaga a aquest grup la resta de la base de dades. Es podria definir com el conjunt de percepcions individuals de la base de dades. Cada visió individual es denomina subesquema o vista. Aquests subesquemes defineixen la part de l'esquema conceptual. Un subesquema podrà ser

compartit per varis usuaris, i cada usuari tindrà la possibilitat d'accedir a diferents subesquemes. Al crear un subesquema és possible barrejar camps de diferents registres, ometre camps, canviar l'ordre dels camps, afegir-ne de nous, etc.

El nivell extern, des del punt de vista funcional dels usuaris, consisteix en un conjunt d'eines de desenvolupament visual de consultes, formularis i informes i un llenguatge de consulta que servirà per la creació de programes que permeten accedir a les dades des de la base de dades.

Cal destacar que, per a una base de dades específica, hi ha un únic esquema intern i conceptual, però hi pot haver varis esquemes externs, cadascun definit per a un o varis usuaris.
A més, cal notar que els tres esquemes no són descriptors de les dades; les úniques dades que existeixen realment estan a nivell físic.

6.4 Independència física i lògica de les dades

L'arquitectura dels tres esquemes pot servir per explicar el concepte d'independència respecte a les dades, que podem definir com la capacitat per a modificar l'esquema en un nivell del sistema de bases de dades sense tenir que modificar l'esquema del nivell immediat superior. Podem definir dos tipus d'independència respecte a les dades.

La independència lògica respecte a les dades és la capacitat de modificar l'esquema conceptual sense tenir que alterar els esquemes externs ni els programes d'aplicació. Podem modificar l'esquema conceptual per ampliar la base de dades (afegint un nou registre o un element d'informació), o per a reduir la base de dades (eliminant un tipus de registre o un element d'informació). En el segon cas, la modificació no haurà d'afectar els esquemes externs que sols es referiran a les dades restants.

La independència física respecte a les dades és la capacitat de modificar l'esquema intern sense tenir que alterar l'esquema conceptual, és a dir, sense que els programes d'aplicació hagin de tornar a escriure's. Potser que calgui modificar l'esquema intern per la necessitat de reorganitzar certs arxius físics per a millorar el rendiment de les operacions d'obtenció i actualització. Si la base de dades conté les mateixes dades, no caldrà modificar l'esquema conceptual.

En la següent figura es pot observar un esquema general de l'arquitectura d'una base de dades, en la qual es detallen els principals components a més de les relacions entre ells i la base de dades lògica.

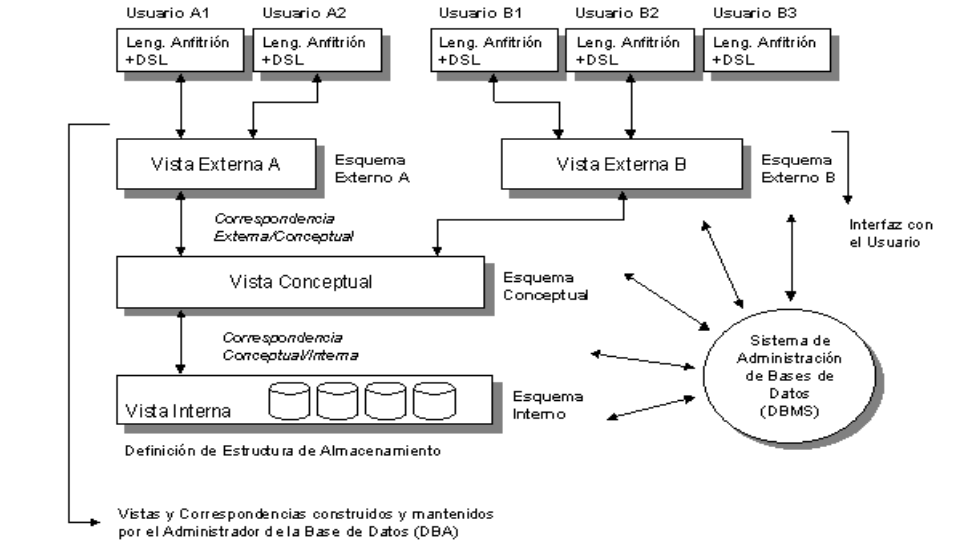


Figura 1.3.- Arquitectura de una Base de Datos

Recordem el significat d'aquests elements:

Una Vista Externa és una visió particular d'un usuari o un grup d'usuaris de la Base de dades.

L'Esquema Extern representa una forma de definició o formalització d'aquesta vista externa.

La Vista Conceptual pretén ser la representació total i abstracta de les dades que componen la Base; la formalització d'aquesta s'assoleix mitjançant l'Esquema Conceptual.

La Vista Interna és d'un nivell molt baix i correspon a l'emmagatzematge físic de les dades de la Base, sobre un Esquema Intern que és la formalització d'això, és a dir, tipus de registres emmagatzemats, índexs, etc. Les correspondències es poden definir com una associació de diferents representacions per a una mateixa dada.

Un DSL és un subllenguatje de dades, és una combinació de dos llenguatges: un llenguatge de definició de dades (DDL) i un llenguatge de manipulació de Dades (DML). Aquest llenguatge representa un nexe entre el Sistema de Base de dades i algun llenguatge amfitrió (p.i. PHP, ASP, Java, C, etc.); i.d., el DSL proveïx eines als llenguatges tradicionals perquè s'integrin al Sistema de Base de dades. Pot haver diferents tipus de DSL per a un mateix sistema.

DBMS és la sigla en anglès de Sistema Gestor de Bases de dades, que correspon al Programari que maneja tots els accessos a la Base de dades, és a dir, cada sol·licitud d'accés d'un usuari al SGBD és interpretada i inspeccionades les correspondències, generant, a continuació, una resposta coherent a les necessitats de la pregunta.

La interfície amb l'Usuari és el límit d'accés que té un Usuari comú a la Base, tot el que està sota aquest límit és transparent (desconegut) per a ell.

L'Administrador de Bases de dades (DBA) correspon a la persona o grup de persones encarregada del control general del sistema.

7. Estructura General d'un SGBD.

Un SGBD es divideix en mòduls que s'encarreguen de tasques específiques. Normalment el S.O. proporciona una sèrie de funcions bàsiques que són utilitzades pel SGBD, per a realitzar altres tasques més complexes. Els components més importants del SGBD són:

-Gestor de fitxers (File Manager o File System).

Encarregat d'assignar espai en disc i mantenir l'estructura dels fitxers on s'emmagatzemaran les dades. A més és el responsable de localitzar el bloc d'informació que es requereix sol·licitant-se'l al Gestor de discos del S.O. i proporcionant-se'l al Gestor de Dades. Pot implementar-se simplement com una interfície al Gestor de Fitxers del Sistema Operatiu o el SGBD pot tenir un propi.

-Gestor de dades (Data Manager).

És el principal component del SGBD. Entre les seves funcions podem citar:

- Accedir a les dades interactuant amb el gestor de fitxers propi o del S.O.
- Assegurar que es compleixen els requisits per a mantenir la integritat i consistència de les dades, a més de controlar la seguretat.
- Sincronitzar les operacions dels usuaris que estan accedint de manera concorrent a la BD.
- Intervindre en operacions de copies de seguretat i recuperacions.

-Processador de consultes (Query Processor).

Interpreta les consultes (queries) dels usuaris online i les converteix en una sèrie d'operacions que seran enviades al Gestor de Dades per a la seva execució. Entre les seves funcions estan l'optimitzar aquestes operacions fent ús de la informació del diccionari de dades i trobar una estratègia eficient per a la seva execució (pla d'execució)

-Compilador de LDD (DDL, Data Definition Languaje).

Converteix les sentències de definició d'esquemes en taules i la informació sobre les mateixes s'emmagatzema en el Diccionari de Dades.

-Precompilador de LMD (DML, Data Manipulation Languaje).

Converteix les sentències en LMD dintre d'un programa d'aplicació a crides a procediments en el llenguatge hoste. El precompilador ha d'interactuar amb el processador de consultes per a generar el codi apropiat.

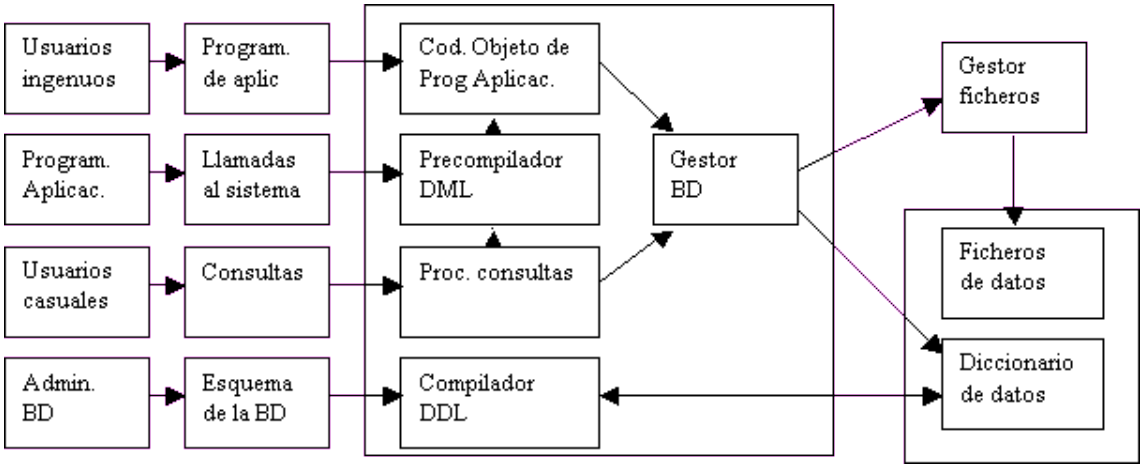
A més dels elements anteriors, es requereixen una sèrie d'estructures de dades perquè els mòduls del SGBD puguin realitzar correctament les seves funcions, aquestes estructures son les següents:

-Fitxers de dades, on es guarden les dades de la BD.

-Diccionari de dades o catàleg del sistema, on es guarda la informació relativa a l'estructura de la BD. És una base de dades que serveix per a documentar les dades. El diccionari de dades emmagatzema dades sobre les dades contingudes a la BD, p ex tipus de dada, grandaria, relacions amb altres dades... S'usa constantment, pel que el seu disseny i implementació han de ser adequats per a obtenir el millor rendiment possible. A més de la informació esmentada sol contenir informació de auditoria i estadístiques que ajuden a optimitzar l'accés a les dades.

-Índexs, que permeten l'accés ràpid a elements d'informació amb determinats valors.

A continuació podem observar en la següent figura un esquema general de l'estructura d'un SGBD:



8 Avantatges i desavantatges d'un SGBD

Sobre els avantatges i/o objectius dels SGBD ja hem parlat suficientment abans. Resumint podem dir que un SGBD proporciona:

- Gestió i control centralitzat de les dades.
- Reducció de redundàncies.
- Compartició de dades.
- Integritat.
- Seguretat.
- Control de concurrència.
- Independència de dades, permetent canvis i creixement.

Pel que fa als desavantatges (relacionades sobretot amb SGBD grans), hem de citar les següents:

- **Cost de programari i maquinari:** A més del cost de comprar el programari, probablement calgui ampliar o pujar de versió el maquinari (upgrade) perquè el programa pugui emmagatzemar-se i funcioni correctament; a més, si no, el sistema podria degradar-se i els temps de resposta pujar dràsticament per la sobrecàrrega que afegix el SGBD a l'haver d'implementar la seguretat, integritat i dades compartides. Recordem que estem parlant de SGBD grans, no per descomptat dels d'ordinadors personals.

- **Cost de migració d'aplicacions**: Un cost addicional, també, és la migració de les aplicacions per a passar a un entorn integrat.

- **Les operacions de copies de seguretat i recuperació de les dades** en cas de fallada són complexes, a causa del accés concorrent de múltiples usuaris.

- **Criticisme del SGBD:** La centralització també significa que les dades de l'organització, estan emmagatzemats en un únic lloc, en la base de dades, amb el que qualsevol fallada de seguretat o operació del SGBD pot afectar a tota l'organització, la solució potser son els sistemes distribuïts i compartits.

9 Models de bases de dades: Jeràrquic, Xarxa i Relacional

Una de les parts més importants de la base de dades és el model de dades, és a dir, una col·lecció d'eines conceptuals per descriure les dades, les relacions de les dades, la semàntica de les dades i els lligams de consistència. Aquests tres models de dades es classifiquen en tres grups:

9.1 Models lògics basats en objectes

S'utilitzen per descriure les dades a nivell conceptual i extern. Es caracteritzen pel fet de que proporcionen capacitats estructurals molt flexibles. Un dels més coneguts és el model entitat-relació.

9.2 Models lògics basats en registres

S'utilitzen també per descriure dades a nivell conceptual i extern, però s'utilitzen tant per especificar l'estructura conceptual de la base de dades com per fer una descripció de la implementació. S'anomenen així perquè la base de dades s'estructura en registres de format fix. A cada tipus de registre es defineix un nombre fix de camps i cada camp té normalment una longitud fixa. Els tres models basats en registres més àmpliament acceptats són el model relacional, el model de xarxa i el model jeràrquic.

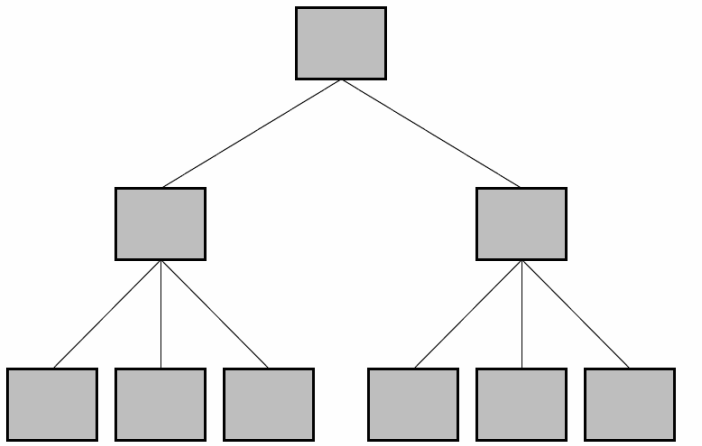
9.3 Models físics

El model físic s'utilitza per descriure dades en un nivell més baix.

Així, passem a detallar en què consistirien cadascun dels models lògics comentats anteriorment. La idea consisteix en, una vegada dissenyat el model conceptual de la base de dades mitjançant un model basat en objectes, l'esquema resultant s'ha de traduir a un model lògic basat en registres. L'objectiu dels tres models mencionats (relacional, xarxa i jeràrquic) és l'emmagatzemament i recuperació de la informació de manera segura i eficient.

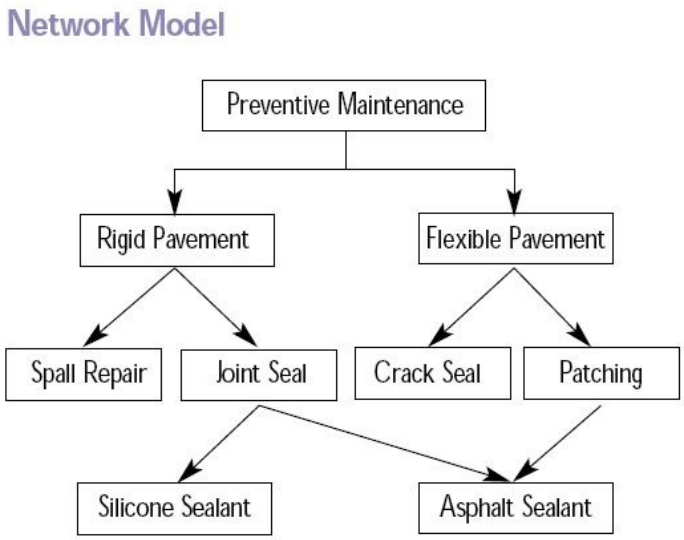
9.4 Model jeràrquic

Un model de dades jeràrquic és un model de dades en el qual les dades són organitzades en registres amb una estructura semblant a un arbre. L'estructura permet a la informació que repeteix i usa relacions pare/Fill: cada pare pot tenir molts fills però cada fill només té un pare. Tots els atributs d'un registre específic són catalogats sota un tipus d'entitat.



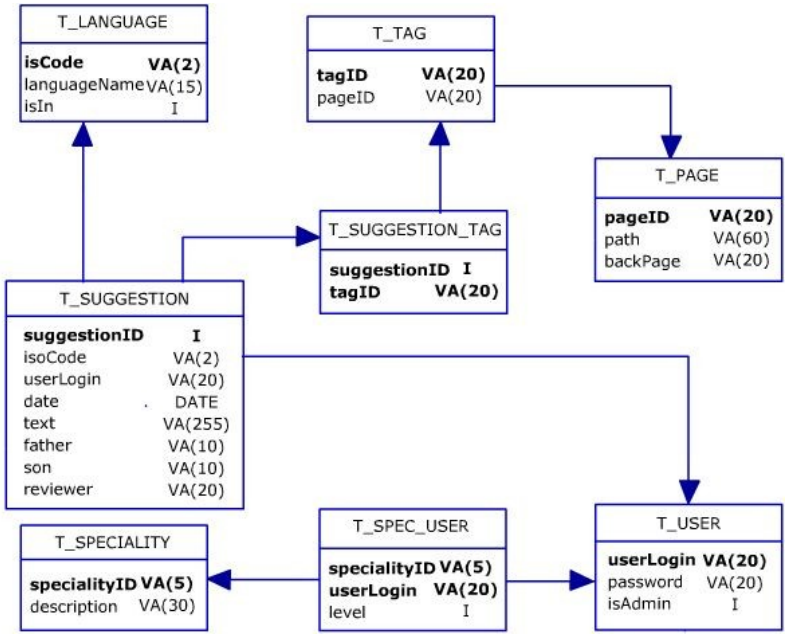
9.5 Model de Xarxa

L'estructura de dades del model jeràrquic és com un arbre de registres, amb cada registre que té un registre paternal i molts fills, el model de xarxa permet a cada registre tenir múltiples registres paternals i de fills, formant una estructura d'enreixat. L'argument principal a favor del model de xarxa, en comparació del model jeràrquic, era que va permetre un modelatge més natural de relacions entre entitats. Diguem que els registres a la base de dades s'organitzen com col.leccions de "grafos" dirigits.



9.6 Model Relacional

En aquest model s'utilitza una col.lecció de taules per representar tant les dades com les relacions entre aquestes dades. Cada taula té diverses columnes i cada columna té un nom únic.



El model relacional per a la gestió d'una base de dades és un model de dades basat en la lògica de predicats i en la teoria de conjunts. És el model més utilitzat en l'actualitat per a modelar problemes reals i administrar dades dinàmicament.

9.7 Diferències entre els models

El model relacional es diferència dels models de xarxa i jeràrquic en que no utilitza punters o enllaços. Aquest model relaciona registres mitjançant els

valors que contenen aquests registres. Aquesta alliberació de l'ús de punters permet definir-ho mitjançant un fonament matemàtic formal.

10 Llenguatge de base de dades

Un sistema de bases de dades proporciona dos tipus de llenguatges diferents: un per especificar l'esquema de la base de dades i l'altre per expressar les consultes i actualitzacions de la base de dades.

10.1 Llenguatge de definició de dades (LDD o DDL)

Un esquema de base de dades s'especifica mitjançant un conjunt de definicions expressades amb un llenguatge especial anomenat llenguatge de definició de dades. El resultat de la interpretació de les instruccions del LDD és un conjunt de taules que s'emmagatzemen en un arxiu especial anomenat diccionari de dades.

Un diccionari de dades és un arxiu que conté metadades, és a dir, dades sobre les dades. Aquest arxiu es consulta abans de llegir o modificar les dades reals del sistema de base de dades.

Podem distingir dos tipus de DDL: l'extern i l'intern, el primer ens permet crear, eliminar o modificar vistes i l'intern permet a l'administrador de la base de dades especificar com es guarden físicament les dades del nivell conceptual.

10.2 Llenguatge de manipulació de dades (LMD o DML)

Per manipulació de dades entenem:

- Recuperació d'informació emmagatzemada a la base de dades. (consulta)
- Inserció de nova informació a la base de dades.
- L'eliminació d'informació de la base de dades.
- La modificació d'informació emmagatzemada a la base de dades.

Al nivell intern hem de definir algorismes que permeten un accés eficient a les dades amb l'objectiu de proporcionar una bona interacció entre nosaltres i el sistema.
El LMD és un llenguatge que permet als usuaris accedir o manipular les dades organitzades amb el model apropiat. Hi ha bàsicament dos tipus:

LMD procedimentals: requereixen que l'usuari especifiqui quines dades necessita i com obtenir aquestes dades.

LMD no procedimental: requereixen que l'usuari especifiqui quines dades necessita, sense necessitat d'especificar com obtenir aquestes dades.

Els LMD no procedimentals són més fàcils d'aprendre i d'utilitzar que els LMD procedimentals ja que en els primers no cal especificar com obtenir les dades.

Però precisament per això també poden generar codis que no siguin tan eficients com els que generen els LMD procedimentals.

11 Administració de BD

En l'ambit de les bases de dades podem diferenciar els següents usuaris :

- **Administradors de bases de dades** (DBA). S'encarrega d'autoritzar l'accés a la base de dades, de coordinar i vigilar el seu ús, i d'adquirir els recursos necessaris de software i hardware. El DBA és la persona responsable quan sorgeixen problemes com violacions a la seguretat o una resposta lenta del sistema.

- **Dissenyadors de bases de dades**. S'encarreguen d'identificar les dades que s'emmagatzemarien en la base de dades i d'escollir les estructures apropiades per representar i emmagatzemar les dades. Els dissenyadors han de comunicar-se amb els futurs usuaris de la base de dades, a fi d'entendre les seves necessitats, i de representar un disseny que acompleixi aquests requeriments. Dintre d'aquest grup de dissenyadors tindríem els analistes de sistemes, que determinen els requeriments dels usuaris finals i desenvolupen especificacions per operacions programades, i els programadors d'aplicacions que són els que implementen aquestes especificacions en forma de programes, i després proven, depuren, documenten i mantenen aquestes transaccions programades.

- **Usuaris finals.** Són les persones que necessiten tenir accés a la base de dades per consultar-la, actualitzar-la i generar informes.

Com hem comentat hi ha un tipus d'usuari especial: el que fa feines d'administració i control de la BD. Una empresa o institució que tingui SI construïts entorn de BD necessita que algú faci tot un seguit de funcions centralitzades de gestió i administració per a assegurar que l'explotació de la BD és correcta. Aquest conjunt de funcions es coneix amb el nom d**'administració de BD** (**ABD**), i els usuaris que fan aquest tipus especial de feina s'anomenen *administradors de BD*.

Els **administradors de BD** són els responsables del correcte funcionament de la BD i vigilen que sempre es mantingui útil. Intervenen en situacions problemàtiques o d'emergència, però la seva responsabilitat fonamental és vetllar perquè no es produeixin incidents.

Tot seguit donem una llista de tasques típiques de l'ABD:

1) Manteniment, administració i control dels esquemes. Comunicació dels canvis als usuaris.

2) Assegurar la màxima disponibilitat de les dades, per exemple, fent còpies (*back-ups*), administrant diaris (*journals* o *logs*), reconstruint la BD, etc.

3) Resolució d'emergències.

4) Vigilància de la integritat i de la qualitat de les dades.

5) Disseny físic, estratègia de camins d'accés i reestructuracions.

6) Control del rendiment i decisions relatives a les modificacions en els esquemes i/o en els paràmetres de l'SGBD i del SO, per a millorar-lo.

7) Normativa i assessorament als programadors i als usuaris finals sobre la utilització de la BD.

8) Control i administració de la seguretat: autoritzacions, restriccions, etc.

La feina de l'ABD no és senzilla

Els SGBD del mercat procuren reduir al mínim el volum d'aquestes feines, però en sistemes molt grans i crítics s'arriba a tenir grups d'ABD de més de 10 persones. Bona part del programari que acompanya l'SGBD està orientat a facilitar la gran diversitat de tasques controlades per l'ABD: monitors del rendiment, monitors de la seguretat, verificadors de la consistència entre índexs i dades, reorganitzadors, gestors de les còpies de seguretat, etc. La majoria d'aquestes eines tenen interfícies visuals per a facilitar la feina de l'ABD.