

# Introducció a les bases de dades

Carlos Manuel Martí Hernández



# Índex

<b>Introducció</b>	<b>5</b>
<b>Resultats d'aprenentatge</b>	<b>7</b>
<b>1 Introducció a les bases de dades i als sistemes gestors de bases de dades</b>	<b>9</b>
1.1 Les dades i les bases de dades	9
1.1.1 Les dades i la seva representació	10
1.1.2 Entitats, atributs i valors	11
1.1.3 Entitats tipus i entitats instància	12
1.1.4 Tipus de dada i domini dels atributs	12
1.1.5 Valor nul dels atributs	13
1.1.6 Atributs identificadors i claus	14
1.1.7 El món de les representacions	15
1.1.8 Les representacions tabulars i la seva implementació: els fitxers	15
1.1.9 Les BD	17
1.1.10 El nivell lògic i el nivell físic	18
1.2 Conceptes de fitxers i bases de dades	19
1.2.1 Concepte i origen de les BD	19
1.2.2 Fitxers i BD	21
1.2.3 L'accés a les dades: tipologies	23
1.2.4 Les diferents visions de les dades	24
1.3 Els SGBD	26
1.3.1 Evolució dels SGBD	26
1.3.2 Objectius i funcionalitats dels SGBD	31
1.3.3 Llenguatges de SGBD	37
1.3.4 Usuaris i administradors	38
1.3.5 Components funcionals dels SGBD	40
1.3.6 Diccionari de dades	42
<b>2 Models de bases de dades</b>	<b>45</b>
2.1 Arquitectura dels SGBD	45
2.1.1 Esquemes i nivells	46
2.2 Els models de bases de dades més comuns	47
2.2.1 Model jeràrquic	47
2.2.2 Model en xarxa	48
2.2.3 Model relacional	49
2.2.4 El paradigma de l'orientació a objectes	50
2.2.5 Modelització de dades amb l'UML	52
2.3 Bases de dades distribuïdes	55
2.3.1 Arquitectures de sistemes de bases de dades: centralitzades, descentralitzades, client-servidor	56
2.3.2 Disseny de bases de dades distribuïdes. Estratègies. Metodologies	61
2.3.3 Conseqüències de la distribució de les dades: duplicació, fragmentació	66

2.3.4 Transaccions i protocols de compromís . . . . . 71

## Introducció

Avui en dia la interacció amb les bases de dades és una cosa tan freqüent que moltes vegades ni tan sols ens adonem d'aquest fet. Senzillament, quan traiem diners d'un caixer automàtic estem utilitzant bases de dades. Per tant, tot tècnic superior en informàtica ha de tenir molt clars una sèrie de conceptes al voltant de les dades i de la seva representació informàtica.

Aquesta unitat didàctica té com a objectiu principal el d'aproximar-nos conceptualment al món de les dades i al de la seva representació informàtica per excel·lència, les bases de dades.

Al llarg de l'apartat "Introducció a les bases de dades i als sistemes gestors de bases de dades", examinarem els tres àmbits que hem de ser capaços de diferenciar per tal de treballar correctament amb les dades: el món real, el món conceptual i el món de les representacions.

Per món real, s'entén la part de la realitat (tingui un caire tangible o no) que en un moment determinat ens interessa informatitzar perquè hem rebut un encàrrec en aquest sentit. Mitjançant una sèrie de processos que impliquen, en primer lloc, l'observació de la realitat i, a continuació, un conjunt d'abstraccions de les informacions considerades rellevants, es construeix un model que conceptualitza els aspectes de la realitat amb els quals volem treballar. Finalment, cal implementar alguna representació informàtica concreta dels conceptes abstractes durant la fase anterior, per tal de poder-hi treballar fent servir les tecnologies que ens ofereixen les bases de dades i, també, els seus sistemes gestors.

També coneixerem les diferències que comporta treballar amb bases de dades en contraposició a treballar utilitzant altres mitjans, com ara els fitxers. I abordarem el programari que gestiona i controla les bases de dades, és a dir, els sistemes gestors. Farem un repàs d'algunes de les orientacions més importants que han tingut, que tenen i que (potser) tindran, i dels objectius generals que persegueixen. I, finalment, ens introduïrem en algunes nocions bàsiques relatives a l'arquitectura dels SGBD.

Al llarg de l'apartat "Models de bases de dades", es plantegen diferents models de bases de dades (jeràrquic, en xarxa, relacional, orientat a objectes, etc.) que han tingut, tenen i tindran, més o menys implantació al llarg del temps, segons l'evolució tant de les necessitats tècniques com dels mercats. I també es fa una mirada en les bases de dades distribuïdes, en la seva arquitectura i el disseny d'aquests sistemes d'emmagatzematge d'informació.



## Resultats d'aprenentatge

En finalitzar aquesta unitat l'alumne/a:

1. Reconeix els elements de les bases de dades analitzant les seves funcions i valorant la utilitat dels sistemes gestors.

- Identifica els diferents elements, objectes i estructures d'emmagatzematge físic disponibles en un SGBD corporatiu i relacionar-lo amb els elements de l'esquema físic de la base de dades.
- Identifica els diferents sistemes lògics d'emmagatzematge i les seves característiques.
- Identifica els diferents tipus de bases de dades en funció de la ubicació de la informació.
- Identifica un sistema gestor de bases de dades: funcions, components, objectius, tipus de llenguatge de bases de dades i diferents usuaris de la base de dades.
- Identifica l'estructura d'un diccionari de dades.
- Diferencia entre el nivell intern, el nivell conceptual i el nivell físic d'una base de dades.
- Diferencia entre els diferents models de bases de dades.
- Identifica les bases de dades distribuïdes: utilitat, diferències, avantatges i inconvenients, distribució de les dades, arquitectura, seguretat i recuperació.
- Identifica el disseny d'una base de dades distribuïda.
- Identifica les bases de dades centralitzades i les bases de dades distribuïdes: utilitat, diferències, avantatges i inconvenients.
- Diferencia entre les diferents tècniques de fragmentació en un model distribuït.
- Identifica les tècniques de distribució de dades.





## 1. Introducció a les bases de dades i als sistemes gestors de bases de dades

Les dades que s'utilitzen de manera informatitzada s'emmagatzemen, habitualment, en bases de dades (BD).

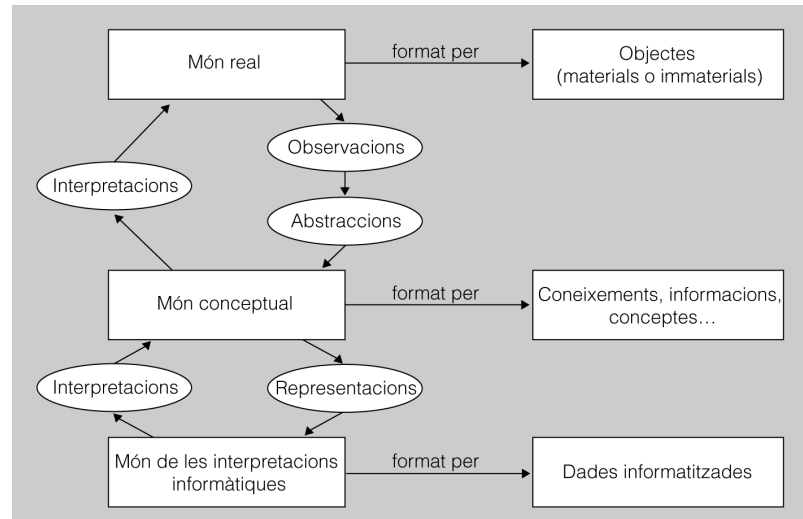
Per tal d'estar en condicions d'abordar l'estudi de les BD i del programari que serveix per gestionar-les, és a dir, els sistemes gestors de bases de dades (SGBD), és imprescindible entendre prèviament uns quants conceptes teòrics fonamentals, referents a les dades i a la seva representació. A banda, doncs, dels conceptes relatius a les dades i a les bases de dades, caldrà conèixer quines representacions de la informació s'utilitzen habitualment així com l'evolució del programari d'aquest àmbit.

### 1.1 Les dades i les bases de dades

Per a tot informàtic que hagi de treballar amb bases de dades (BD), és imprescindible saber distingir tres àmbits ben diferenciats, però que al mateix temps estan fortament interrelacionats, els quals fan referència, respectivament, a la realitat, a la seva conceptualització, i a la seva representació informàtica ulterior. Així, doncs, considerem els tres “mons” següents:

- **El món real.** Està constituït pels objectes (materials o no) de la realitat que ens interessen i amb els quals haurem de treballar.
- **El món conceptual.** És el conjunt de coneixements o informacions obtinguts gràcies a l'observació de la part del món real que ens interessa. Un mateix món real pot donar lloc a diferents mons conceptuals, en funció de la manera de percebre la realitat, o els interessos de l'observador d'aquesta.
- **El món de les representacions.** Està format per les representacions informàtiques, o dades, del món conceptual, necessàries per poder treballar.

En la figura 1.1 es representen, de manera esquematitzada, els tres mons que els informàtics han de considerar, i les interrelacions que mantenen.

**FIGURA 1.1.** Els tres móns

### 1.1.1 Les dades i la seva representació

Un cop estructurats, els conceptes entorn de la realitat passen a ser veritables informacions, amb les quals els humans ens podem comunicar i començar a treballar.

Però encara cal donar un altre pas que ens permeti representar aquestes informacions, de tal manera que les puguem tractar informàticament mitjançant BD i aplicacions, i aprofitem així tot el potencial de les noves tecnologies.

Les **dades** són representacions informàtiques de la informació disponible, relativa als objectes del món real del nostre interès.

El **món de les representacions** està format per les dades informatitzades amb les quals treballem.

Ara bé, la conversió de les concepcions en dades no és automàtica, ni de molt. Requereix passar per dues fases successives de disseny, en què es prenen decisions que poden derivar en resultats dispars. Aquestes dues fases de disseny són les següents:

1. **Fase de disseny lògic.** Es treballa amb el model abstracte de dades obtingut al final de l'etapa de disseny conceptual, per tal de traduir-ho al model de dades utilitzat pel SGBD amb el qual es vol implementar i mantenir la futura BD.
2. **Fase de disseny físic.** Es poden fer certes modificacions sobre l'esquema lògic obtingut en la fase de disseny anterior, per tal d'incrementar l'eficiència en algunes de les operacions que s'hagin de fer amb les dades.

Per tant, cal ser conscients que, en un mateix conjunt de coneixements entorn d'una mateixa realitat, aquests es poden representar de maneres diferents a causa, per exemple, dels factors següents:

- Les decisions de disseny preses (tant a nivell conceptual, com a nivell lògic i físic).
- La tecnologia emprada (fitxers, BD relacionals, BD distribuïdes, etc.).

La possibilitat que hi hagi aquestes diferències no implica que tots els resultats es puguin considerar equivalents, sense més ni més, ja que, normalment, les representacions diferents donen lloc a nivells d'eficiència també diferents. Aquest fet pot tenir conseqüències importants, ja que la responsabilitat de tot informàtic inclou garantir la correcció de les representacions, però també l'eficiència de les implementacions.

### 1.1.2 Entitats, atributs i valors

Tres elements caracteritzen fonamentalment les informacions:

1. Les **entitats** són els objectes del món real que conceptualitzem. Són identificables, és a dir, distingibles els uns dels altres. I ens interessen algunes (com a mínim una) de les seves propietats.
2. Els **atributs** són les propietats de les entitats que ens interessen.
3. Els **valors** són els continguts concrets dels atributs, les determinacions concretes que assolixen.

En principi, els atributs haurien d'emmagatzemar un sol valor en cada instant. D'aquesta manera, els nostres models seran, de bon principi, compatibles amb el model lògic de dades més àmpliament utilitzat: el **model relacional**.

#### Exemple d'entitat, atributs i valors

Considerarem que una pel·lícula concreta és una entitat, perquè és un objecte del món real, que hem conceptualitzat dins d'una categoria (la dels films cinematogràfics), i que al mateix temps és distingible d'altres entitats de la mateixa categoria (és a dir, d'altres films).

D'aquesta pel·lícula ens interessaran alguns aspectes, que anomenarem *atributs*, com per exemple, el títol, el director i l'any de producció.

Finalment, aquests atributs adoptaran uns valors concrets com ara, i respectivament, *Paths of glory*, Stanley Kubrick i 1957.

Si només coneixem dos d'aquests tres elements, no disposarem d'una veritable informació, ja que es produirà alguna de les mancances següents:

- Desconeixerem l'entitat (l'objecte) a què va associat l'atribut i el valor respectiu, i per tant no servirà de gaire conèixer els altres aspectes.

#### Atributs multivaluats

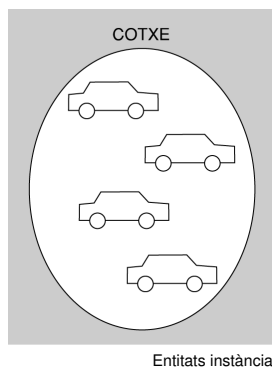
Permeten emmagatzemar més d'un valor simultàniament. El seu ús no és gaire recomanable perquè són incompatibles amb el model relacional i amb la immensa majoria dels SGBD que hi ha en el mercat.

- Desconeixerem quin atribut ( propietat) de l'entitat adopta el valor obtingut, la qual cosa pot donar lloc a equívocs.
- Sabrem que l'entitat té una certa propietat, però en desconeixerem el valor, i per tant aquest coneixement difícilment ens resultarà útil.

### 1.1.3 Entitats tipus i entitats instància

El terme *entitat* fa referència a algun objecte concret del món real, conceptualitzat, i del qual ens interessen algunes característiques. Però aquest terme pot tenir dos significats diferents, segons a què faci referència. Per tal de distingir-los, es pot afegir un adjectiu al substantiu esmentat i obtenir, així, els dos sintagmes següents:

**Entitat tipus.** Es tracta d'un tipus genèric d'entitat o, si es prefereix, d'una abstracció, que fa referència a una classe de coses com, per exemple, els cotxes, en general.



**Entitat instància.** Es refereix a la conceptualització d'un objecte concret del món real, com ara un cotxe concret, distingible dels altres objectes del mateix tipus, gràcies a alguna propietat (com podria ser el valor de l'atribut Matricula).

En terminologia de teoria de conjunts, podríem dir que una entitat tipus és un conjunt, i que cada entitat instància és un element del conjunt, tal com es reflecteix en la figura adjacent.

### 1.1.4 Tipus de dada i domini dels atributs

Anomenem **domini** tot el conjunt de valors que un atribut determinat pot prendre vàlidament.

El concepte domini no es correspon amb el de *tipus de dada*, utilitzat tant en l'àmbit de les BD com també en el de programació.

Un **tipus de dada** defineix un conjunt de valors amb unes característiques comunes que els fan compatibles, per la qual cosa també defineix una sèrie d'operacions admissibles sobre aquests valors.

### Exemple de tipus de dada

Podem considerar els nombres enters com un tipus de dada (diferent d'altres tipus, com per exemple els nombres reals, els caràcters, etc.), sobre el qual es poden definir certes operacions, com la suma, la resta, la multiplicació o la divisió entera (però no la divisió exacta, que només és possible entre els nombres reals).

Així, doncs, ambdós conceptes (domini i tipus de dada) s'assemblen, ja que tots dos limiten els valors acceptables. Allò que els diferencia és que un domini no estableix per si mateix cap conjunt d'operacions, mentre que un tipus de dada, per definició, sí que ho fa.

Una altra diferència és que, en la pràctica, un domini és un subconjunt de valors possibles d'un tipus de dada. En alguns casos, pot interessar delimitar el rang de valors admesos per un tipus de dada determinat. Això es fa establint un domini.

### Exemple de domini

Imaginem que, en l'àmbit d'uns estudis determinats, s'exigeix un mínim d'assistència a classes presencials per tal d'aconseguir el títol corresponent, independentment de les qualificacions obtingudes.

Imaginem que s'admet, durant tot el curs acadèmic, un màxim de vint faltes injustificades. Doncs bé, hi podria haver un atribut de l'entitat ALUMNE, anomenat, per exemple, NombreFaltes, que recollís aquesta circumstància.

Aquest atribut podria emmagatzemar dades de tipus enter. I també se'n podria limitar el domini de 0 (per indicar que no hi ha hagut cap inassistència) a vint faltes injustificades, ja que en arribar a aquest límit es produiria l'expulsió de l'alumne.

## 1.1.5 Valor nul dels atributs

De vegades, el valor d'un atribut és desconegut o, fins i tot, no existeix. Per representar aquesta circumstància, l'atribut en qüestió haurà d'admetre el valor nul.

L'expressió **valor nul** indica que no hi ha cap valor associat a un atribut determinat d'una entitat instància concreta.

Perquè un atribut admeti el valor nul, s'ha d'especificar aquesta possibilitat a l'hora de definir-ne el domini.

### Exemple de valor nul

Considerem ara que l'entitat ALUMNE disposa d'un atribut anomenat Telefon, per emmagatzemar un número telefònic de contacte de cadascuna de les persones matriculades.

Si l'atribut Telefon no admetés valors nuls, el sistema no permetria que es matricuessin persones que no disposessin de telèfon.

En canvi, si s'ha admès aquesta possibilitat en definir el domini de l'atribut que ara ens ocupa, el sistema acceptarà la matrícula de les persones que no disposin de telèfon, o bé de les que senzillament no se'l saben de memòria i que, per tant, no el poden indicar en el moment de formalitzar la matrícula, de manera que la seva incorporació en la BD queda pendent.

No s'ha de confondre valor nul amb el zero, si estem tractant amb valors numèrics, o amb l'espai en blanc, si estem treballant amb caràcters. Tant l'un com l'altre són valors amb un significat propi. En canvi, el valor nul implica l'absència total de valor.

### 1.1.6 Atributs identificadors i claus

Un **atribut identificador** és el que permet distingir inequívocament cada entitat instància de la resta, pel fet que el seu valor és únic, i no es repeteix en diferents entitats instància.

Els atributs d'una entitat seran identificadors, o no, en funció de l'objecte del món real que l'entitat vulgui modelitzar.

#### Exemple d'atribut identificador

L'atribut DNI pot servir molt bé per identificar les instàncies d'una entitat que modelitzi els alumnes d'un centre docent, ja que cada alumne tindrà un DNI diferent.

Però si l'entitat amb què treballem vol modelitzar les qualificacions finals dels alumnes, el DNI per si sol no permetrà identificar les diferents entitats instància, ja que a cada alumne correspondrà una nota final per a cada assignatura en la qual s'hagi matriculat.

De vegades, un sol atribut no és suficient per identificar inequívocament les diferents instàncies d'una entitat. Aleshores, cal recórrer a la combinació dels valors de dos o més atributs de la mateixa entitat.

Tot atribut o conjunt d'atributs que permeten identificar inequívocament les instàncies d'una entitat s'anomenen **claus**.

Tot atribut identificador és, al mateix temps, una clau. Però els atributs que formen part d'un conjunt de més d'un atribut que actua com a clau de l'entitat no són identificadors, ja que, per si mateixos, no són capaços d'identificar les entitats instància.

#### Exemple de clau formada per més d'un atribut

Per tal de diferenciar les instàncies d'una entitat que vol reflectir les notes finals dels alumnes en cada assignatura en què s'hagin matriculat, cal combinar els valors de dos atributs: un que designi l'alumne de què es tracti (típicament, el DNI), i un altre que indiqui l'assignatura a la qual correspon la nota (que podria ser una cosa com ara CodiAssignatura).

Això és així perquè un alumne podrà estar matriculat en diferents assignatures, per la qual cosa el seu DNI es repetirà en diferents instàncies. I, al mateix temps, diversos d'alumnes podran estar matriculats d'una mateixa assignatura i, per tant, els valors de l'atribut CodiAssignatura també es podran repetir.

En canvi, cada alumne tindrà una instància per a cada assignatura en què s'hagi matriculat, fins que l'aprovi, per tal de reflectir la nota final. Modelitzada l'entitat d'aquesta manera, la combinació de valors de DNI i CodiAssignatura no es repetirà mai, i el conjunt format per aquests dos atributs podrà servir com a clau.

Per definició, ni els atributs identificadors ni els que formen part d'una clau poden admetre mai el valor nul, perquè aleshores no servirien per distingir les entitats instància sense valor en un dels tipus d'atribut esmentat de la resta.

### 1.1.7 El món de les representacions

Ja sabem que les dades són informacions representades informàticament. Per tant, també podríem anomenar *món de les dades* el món de les representacions.

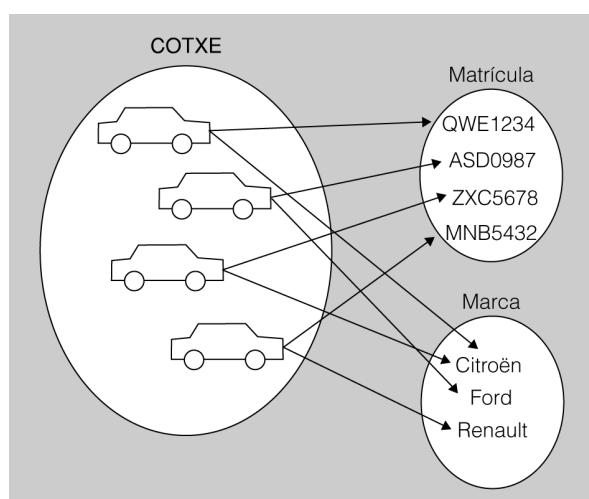
La representació informàtica més freqüent en l'àmbit de les BD és la representació tabular, la qual s'implementa habitualment en fitxers que s'estructuren en registres i camps.

En el fons, les BD només són conjunts de fitxers interrelacionats (o, si es prefereix, que emmagatzemen dades que estan interrelacionades). Però no serveix de res emmagatzemar dades si, posteriorment, no hi accedim. Hi ha diferents tipus d'accés a les dades que convé examinar: seqüencial, directe, per valor i per posició.

### 1.1.8 Les representacions tabulars i la seva implementació: els fitxers

Les informacions són conceptualitzacions obtingudes a partir de l'observació del món real. Ara bé, les informacions no són gaire còmodes per treballar.

**FIGURA 1.2.** Exemple de representació no informatitzada



En la figura 1.2 podem veure una representació gràfica, no informatitzada, de l'entitat COTXES, que consta de dos atributs: Matrícula i Marca. Evidentment, si augmentés el nombre d'entitats instància, o bé el nombre d'atributs a considerar, aquest tipus de representació no serviria per a res.

És necessari representar les informacions per facilitar les tasques a fer amb elles, com ara les consultes, els processaments, les transmissions, etc.

La representació més freqüent en l'àmbit informàtic de les BD és l'anomenada **representació tabular** (o, el que és el mateix, *en forma de taula*).

Cada taula representa una entitat genèrica, i està estructurada en files (agrupacions horitzontals de cel·les) i columnes (agrupacions verticals de cel·les):

- Cada fila representa una entitat instància.
- Cada columna representa un atribut.
- Cada cel·la (és a dir, cada intersecció d'una fila i d'una columna) emmagatzema el valor que tingui l'atribut de l'entitat instància de què es tracti.

#### Fitxer

El terme fitxer té altres acepcions, com ara en l'àmbit dels sistemes operatius, que no tenen gaire a veure amb el concepte que hem exposat aquí.

La implementació informàtica de les representacions tabulars es materialitza mitjançant els anomenats **fitxers de dades**. S'entén per fitxer la implementació informàtica d'una taula, amb les dades estructurades en registres i camps.

Els fitxers s'implementen seguin aquestes consideracions:

- La implementació de cada entitat instància s'anomena **registre**, i equival a una fila de la representació tabular.
- La implementació de cada atribut s'anomena **camp**, i equival a una columna de la representació tabular.
- Cada intersecció d'un registre i d'un camp emmagatzema el **valor** que tingui el camp del registre de què es tracti.

Els fitxers s'han d'emmagatzemar en algun dispositiu de memòria externa de l'ordinador, típicament un disc dur, per tal de conservar les dades permanentment. L'emmagatzemament en la memòria interna no satisfà aquest objectiu perquè és volàtil.

En la taula 1.1, podem veure una representació tabular de l'entitat COTXES, que només consta de dos camps: Matrícula i Marca. Si augmentés el nombre de registres a emmagatzemar només caldria afegir més files. I si haguéssim de considerar més camps, només caldria afegir més columnes.

Però en cap cas no es comprometria la complexitat de l'estructura tabular, que seria, essencialment, la mateixa.



TAULA 1.1. Exemple de representació tabular

Cotxes	
Matrícula	Marca
QWE1234	Citroën
ASD0987	Ford
ZXC5678	Citroën
MNB5432	Renault

### 1.1.9 Les BD

Normalment, quan hàgim de representar informàticament certes informacions (corresponents, doncs, al món conceptual), no ens trobarem amb una sola entitat tipus, sinó amb unes quantes.

Intuïtivament, podem pressuposar que si partim d'un nombre concret d'entitats tipus necessitarem, com a mínim, el mateix nombre de taules per representar-les (i probablement algunes més). Ara bé, aquestes taules, o fitxers, no seran objectes inconnexos, sinó que hauran d'estar interrelacionats.

Les **interrelacions** són informacions que permeten associar les entitats entre elles.

Les interrelacions entre els registres de dues (o més) taules es fan mitjançant camps del mateix tipus de dada que emmagatzemin els mateixos valors.

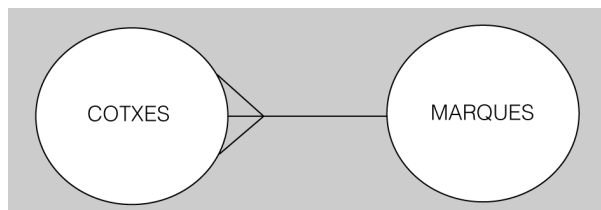
#### Exemple d'entitats interrelacionades

Imaginem ara que construïm una petita BD. Només hi ha dues entitats tipus del nostre interès: COTXES i MARQUES.

També tenim una altra informació complementària, sobre la interrelació entre ambdues entitats: cada cotxe serà d'una marca concreta, però hi podrà haver molts cotxes de cada marca.

En la figura 1.3, es mostra una representació de les dues entitats (COTXES i MARQUES) interrelacionades.

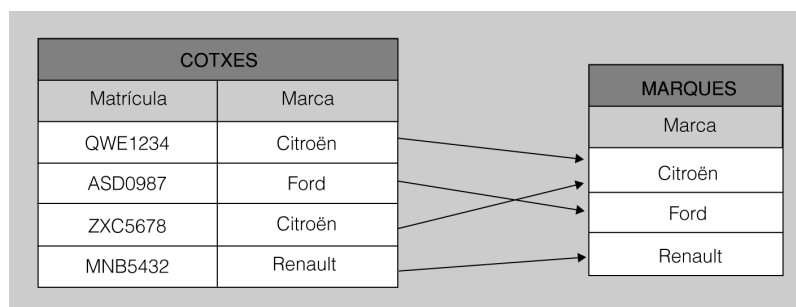
FIGURA 1.3



L'entitat MARQUES només té l'atribut Marca, i l'entitat COTXES només té l'atribut Matrícula. El fitxer per representar COTXES haurà d'afegir un camp addicional, per tal de permetre la interrelació amb el fitxer que representa l'entitat MARQUES.

La figura 1.4 mostra la interrelació entre els dos fitxers corresponents a l'entitat COTXES i a l'entitat MARQUES.

**FIGURA 1.4.** Exemple de fitxers interrelacionats



La interrelació entre fitxers implica que els canvis de valor dels camps que serveixen per interrelacionar-los (o, fins i tot, la seva supressió) han de quedar reflectits en tots els fitxers implicats, per tal de mantenir la coherència de les dades. Per tant, els programes que treballen amb fitxers de dades interrelacionats sempre tindran un plus de complexitat, derivat de l'exigència que acabem de comentar.

Una **BD** consisteix en un conjunt de fitxers de dades interrelacionats.

Un sistema gestor de bases de dades (SGBD) és un tipus de programari especialitzat en gestionar i administrar bases de dades.

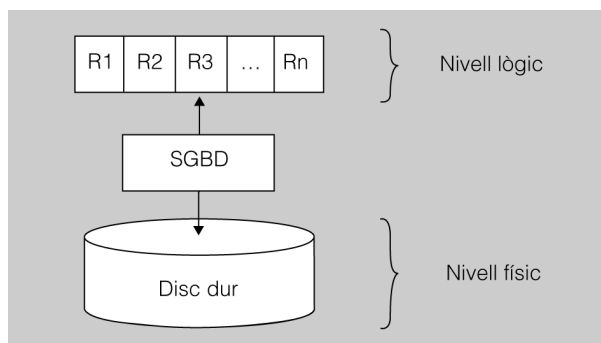
En aquest sentit, els **SGBD** s'han anat desenvolupant tenint com a un dels objectius principals facilitar la programació amb accés a dades persistents, i per gestionar l'accés simultani a les dades per part de diferents usuaris.

### 1.1.10 El nivell lògic i el nivell físic

L'organització de les dades, i el seu enregistrament i accés, es poden considerar des de dos punts de vista, més o menys propers a la implementació física de l'enregistrament de les dades:

- **Nivell lògic.** Permet treballar amb les dades de manera més senzilla, independentment de la implementació física concreta, que no cal conèixer. És la manera de treballar més productiva i, per tant, la més recomanable, sempre que les circumstàncies no ens obliguin a fer optimitzacions a nivells més baixos.
- **Nivell físic.** Implica un coneixement a baix nivell de la implementació física de l'organització de les dades i el seu accés.

En la figura 1.5, es mostra la doble perspectiva apuntada, la lògica i la física. En el disc dur es desen les dades, organitzades de determinada manera a nivell físic. L'SGBD ens permet accedir a les dades emmagatzemades en el disc dur considerant només aspectes de nivell lògic, com ara seqüències de registres.

**FIGURA 1.5.** Nivell lògic i nivell físic**Exemples de treball a nivell lògic i a nivell físic**

Nivell lògic: es treballa tenint en compte, fonamentalment, les taules, amb els camps i registres corresponents, i les seves interrelacions.

Nivell físic: es treballa considerant altres factors a més baix nivell, com ara l'encadenament dels registres físics, la compressió de dades, les tipologies d'índexs, etc.

## 1.2 Conceptes de fitxers i bases de dades

Les BD són conjunts estructurats de dades organitzades en entitats, les quals estan interrelacionades.

És important conèixer el concepte de BD i el seu origen. També és interessant establir una comparativa d'avantatges i d'inconvenients segons si s'utilitzen fitxers tradicionals o BD.

Hi ha diferents perspectives des de les quals es pot observar una BD, i diferents tasques i mètodes de treball sobre una BD, que tot informàtic ha de conèixer.

En el mercat hi ha diferents models de BD: el jeràrquic, en xarxa, relacional, orientat a objectes, etc. És interessant, doncs, conèixer les característiques de cadascun dels models.

### 1.2.1 Concepte i origen de les BD

Les BD no existeixen per casualitat. Van aparèixer per donar resposta a una sèrie de necessitats. La millor manera d'entendre tant aquestes circumstàncies com el concepte de BD que van originar és fer una petita aproximació històrica a la seva evolució.

Les aplicacions informàtiques dels anys seixanta del segle XX tenien, per regla general, les característiques següents:

- Normalment, consistien en processos per lots (en anglès, *batch processing*), amb les particularitats següents:
  - Un lot és un conjunt finit de feines que es volen tractar com un tot.
  - El seu processament, una vegada engegat, no necessita cap interacció amb l'usuari.
  - Normalment, aquest tipus d'execució es realitza en tasques repetitives sobre gran volums d'informació.
- Realitzaven tasques molt específiques, relacionades amb molt poques entitats tipus, com per exemple l'emissió de factures, la confecció de nòmines de personal, etc.
- Normalment, els programes treballaven de manera seqüencial sobre un sol fitxer mestre, que estava emmagatzemat en una cinta magnètica (encara no, en un disc dur), i generaven un altre fitxer com a resultat.
- Quan es detectava la necessitat d'implementar una nova aplicació que utilitzés parcialment les dades contingudes en un fitxer i, a més, algunes altres de noves, es dissenyava un nou fitxer amb tots els camps necessaris, i s'omplia amb les dades corresponents:
  - Les dades que ja eren en l'antic fitxer es podien copiar en el nou, justament, mitjançant un altre processament per lots (*batch*).
- D'aquesta manera s'aconseguia que el nou programa no hagués de treballar amb molts fitxers, la qual cosa en simplificava el codi, d'una banda, i d'una altra n'optimitzava el temps d'execució.
- Com a contrapartida, aquesta manera de treballar comportava la redundància d'algunes dades, que eren repetides en diferents fitxers. Aquest fet dificultava el manteniment de la coherència d'aquestes dades.

Posteriorment, l'evolució tecnològica va fer possible la implantació progressiva de tres nous elements:

- Els terminals. Dispositius de maquinari per introduir o mostrar dades de les computadores.
- Els discos durs. Dispositius d'emmagatzemament d'alt rendiment, que no estaven limitats *de facto* a l'accés seqüencial.
- Les xarxes de comunicació.

A partir d'aquestes innovacions, els programes informàtics van haver d'implementar la possibilitat de realitzar consultes i actualitzacions de les mateixes dades, simultàniament, per part de diferents usuaris.

Al mateix temps, es va anar produint un fenomen que consistia en la integració de les diferents aplicacions informàtiques que utilitzava cada organització (per exemple, gestions d'estocs, facturacions, proveïdors...). Aquesta tendència requeria les accions següents:

- Interrelacionar els fitxers de les antigues aplicacions.
- Eliminar la redundància, és a dir, la repetició innecessària de dades, que a més de resultar ineficient posa en risc la seva coherència.

Tant la interrelació dels fitxers com el fet que cada vegada hi accedien simultàniament més usuaris exigien unes estructures físiques que proporcionessin accessos raonablement ràpids, com ara índexs.

Al començament dels anys setanta, aquests conjunts de fitxers interrelacionats, compartits per diferents processos i usuaris simultàniament, i amb estructures complexes, van rebre inicialment el nom de *data bases*, o DB (*bases de dades*, o BD, en català).

I al final dels anys setanta van anar sortint al mercat programaris encara més sofisticats, que permetien gestionar més fàcilment les relacions entre fitxers, i ja estaven en condicions de garantir l'actualització simultània de dades per part de diferents usuaris, etc. Aquests programaris es van anomenar **sistemes gestors de bases de dades**, o **SGBD** (*data base management systems*, o **DBMS**, en anglès).

Amb aquesta perspectiva històrica, doncs, podem donar una definició de BD més completa.

Una **BD** és la representació informàtica dels conjunts d'entitats instància corresponents a diferents entitats tipus i de les relacions entre aquestes. Aquest conjunt estructurat de dades ha de poder ser utilitzat de manera compartida i simultània per una pluralitat d'usuaris de diferents tipus.

### 1.2.2 Fitxers i BD

Els fitxers tradicionals (i els programes necessaris per treballar-hi) s'han trobat amb serioses dificultats per satisfer les creixents necessitats dels usuaris en pràcticament tots els àmbits.

Per aquesta raó, les BD s'han anat implantant com a mecanisme per excel·lència d'emmagatzematge, processament i obtenció d'informació, tot desplaçant progressivament els fitxers de la seva posició preeminent anterior. La taula 1.2 conté una breu descripció de les principals diferències entre els sistemes basats en fitxers tradicionals i les BD.

TAULA 1.2. Fitxers i BD

	Fitxers	Bases de dades
<b>Entitats tipus</b>	Les entitats instància d'un fitxer pertanyen a una sola entitat tipus.	Les BD contenen entitats instància d'infinat d'entitats tipus interrelacionades.
<b>Interrelacions</b>	El sistema no interrelaciona fitxers.	El sistema té previstes eines per interrelacionar fitxers.
<b>Redundàncies</b>	És necessari crear fitxers a mida de cada aplicació, amb totes les dades necessàries, encara que estiguin repetides en altres fitxers.	Tècnicament, totes les aplicacions poden treballar amb la mateixa BD, la qual cosa evita la redundància de dades i els riscos que comporta.
<b>Inconsistències</b>	És possible que els valors d'unes mateixes dades en diferents fitxers no coincideixin, si els programadors no les han actualitzat degudament.	Si les interrelacions estan ben dissenyades, les dades només han d'estar emmagatzemades en la BD un sol cop. Per tant, no hi ha risc d'inconsistències.
<b>Obtenció de dades</b>	Si no hi ha una aplicació que obtingui les dades que volem, o bé s'ha de fer un programa a mida, o bé s'ha d'aprofitar la sortida d'un programa amb objectius similars, i fer els càlculs necessaris manualment.	Permeten obtenir qualsevol conjunt de dades, segons les necessitats, dels del seu propi entorn de treball, sense haver d'escriure, compilar i executar cap nou programa d'aplicació contra la BD.
<b>Aïllament de dades</b>	Les dades estan disperses i aïllades en diferents arxius, la qual cosa dificulta el desenvolupament de les aplicacions.	Totes les dades són en la mateixa BD, interconnectades, la qual cosa en facilita l'obtenció.
<b>Integritat de dades</b>	Els programes han d'implementar totes les restriccions sobre les dades, afegint el codi font corresponent. El manteniment és complicat quan la informació es conté en diferents fitxers utilitzats per diferents aplicacions.	La BD s'encarrega directament d'implementar les restriccions sobre les dades. Els programes no han d'incorporar codi font addicional per garantir-les.
<b>Atomicitat</b>	Alguns conjunts d'operacions sobre les dades s'han d'executar de manera indivisible (o tots o cap), independentment de les fallades que el sistema pugui presentar (com ara per un tall de subministrament elèctric). Però això és molt difícil de garantir amb un sistema d'informació basat en fitxers.	Les BD incorporen la tècnica de les transaccions per tal de garantir fàcilment l'execució atòmica d'una pluralitat de processos sobre les dades.
<b>Accés concurrent</b>	L'actualització simultània de dades d'un mateix fitxer per part de diferents usuaris o aplicacions en pot provocar fàcilment la inconsistència.	Amb la tècnica del bloqueig, les BD garanteixen automàticament la consistència de les dades, malgrat que més d'un usuari o més d'una aplicació les vulguin actualitzar simultàniament.
<b>Seguretat</b>	Habitualment, cada fitxer serveix per a un sol usuari o una sola aplicació (sobretot simultàniament), i ofereix una visió única del món real. Però no sempre tots els usuaris que utilitzen un fitxer haurien de tenir accés a totes les dades que conté.	Una BD pot ser compartida per molts usuaris de diferents tipus (fins i tot, simultàniament), els quals poden tenir diferents visions (vistes) del món real, en funció del seu perfil i dels permisos que s'hagin de concedir en cada cas.

Evidentment, les prestacions de les BD són molt superiors a les que proporcionen els sistemes de fitxers. Però això no vol dir que en alguns casos no sigui millor

utilitzar fitxers, com ara quan el volum de les dades a contenir és molt petit, o quan només hem de treballar amb una entitat instància i, per tant, no cal considerar interrelacions.

Algunes utilitzacions possibles dels fitxers en l'actualitat són les següents:

- Fitxers de configuració d'aplicacions.
- Fitxers de configuració de sistemes.
- Fitxers de registres d'esdeveniments (*logs*).

En casos com aquests, no compensaria carregar innecessàriament el sistema amb una BD (i amb el sistema gestor corresponent), ja que no s'aprofitarien els avantatges de les BD però, en canvi, empitjoraria el rendiment del sistema.

### 1.2.3 L'accés a les dades: tipologies

No serveix de res estructurar dades i emmagatzemar-les si després no s'hi pot d'accedir, per consultar-les, modificar-les o transmetre-les.

En general, hi ha dues maneres bàsiques d'accedir a les dades:

- **L'accés seqüencial** a un registre determinat, que implica l'accés previ a tots els registres anteriors.
- **L'accés directe** a un registre concret, que implica l'obtenció directa del registre.

A més, hi ha una altra classificació habitual de tipologies d'accessos:

- **L'accés per valor**, que permet obtenir el registre en funció del valor d'algun (o alguns) dels seus camps, sense considerar la posició que ocupa el registre.
- **L'accés per posició**, que obre l'accés a un registre que ocupa una posició determinada, sense considerar el contingut del registre.

Combinant les dues dicotomies anteriors, resulten quatre mètodes d'accés a les dades, tal com es mostra en la taula 1.3, que s'ajusten més a la realitat.

TAULA 1.3. Mètodes d'accés a les dades

	P - per posició	V - per valor
S - seqüencial	SP	SV
D - directe	DP	DV

Examinem, doncs, les quatre tipologies d'accés a dades més freqüents:

- **SP (accés seqüencial per posició).** Després d'haver accedit a un registre que es troba en una posició determinada, s'accedeix al registre que ocupa la posició immediatament posterior.
- **DP (accés directe per posició).** S'obté directament un registre pel fet d'ocupar una posició determinada.
- **SV (accés seqüencial per valor).** Després d'haver accedit a un registre que té un valor concret, s'accedeix al registre que ocupa la posició immediatament posterior, segons l'ordenació establerta a partir d'un camp determinat (o més). L'ordre serà creixent o decreixent, si es tracta d'un camp numèric, o alfabètic ascendent o descendent, si es tracta d'un camp de caràcters.
- **DV (accés directe per valor).** S'obté directament un registre pel fet de tenir un valor determinat en un dels seus atributs (o més).

#### Exemples de tipus d'accés a les dades

Imaginem que disposem un fitxer en què s'emmagatzema informació relativa als alumnes d'un centre docent: DNI, nom, cognoms, data de naixement, adreça, telèfon, etc. A continuació, es dona un exemple de cadascun del quatre mètodes d'accés estudiats.

SP (accés seqüencial per posició): la llista de tots els alumnes, sense establir cap ordenació.

DP (accés directe per posició): en l'àmbit de la programació, el cas més típic és el de les cerques dicotòmiques en vectors ordenats; en l'àmbit de les BD, aquest tipus d'accés es produeix en utilitzar un índex de tipus *hashing*.

SV (accés seqüencial per valor): la llista de tots els alumnes, seguint un ordre alfabètic ascendent, en primer lloc dels cognoms i després del nom.

DV (accés directe per valor): obtenció de les dades emmagatzemades en un registre corresponent a un alumne concret, que es digui, per exemple, Pere García Manent (és a dir, que el camp Nom conté el valor "Pere", i el camp Cognoms conté el valor "García Manent").

### 1.2.4 Les diferents visions de les dades

Un dels principals objectius de les BD és proporcionar, als usuaris, una **visió abstracta de les dades**. Amb aquesta finalitat, el sistema amaga als usuaris certs detalls relatius a l'emmagatzemament i manteniment de les dades, per facilitar-los la feina, d'una banda, però també per garantir certs aspectes en matèria de seguretat.

Perquè les BD resultin útils, han de ser mínimament eficients a l'hora de recuperar les dades. Per aquest motiu, els sistemes de BD tenen implementades, a baix nivell, estructures de dades bastant complexes.

S'utilitzen tres nivells d'abstracció -físic, lògic i de vistes- per tal d'amagar aquestes estructures complexes i simplificar, d'aquesta manera, la interacció dels usuaris amb el sistema.



1. **Nivell físic:** és el nivell d'abstracció més baix de tots els utilitzats.

- Descriu com s'emmagatzemen realment les dades a baix nivell, especificant en detall les complexes estructures que es necessiten.
- No és freqüent treballar a aquest nivell. Només es fa quan calen optimitzacions en l'estructuració de les dades a baix nivell.

2. **Nivell lògic:** és el nivell d'abstracció intermedi.

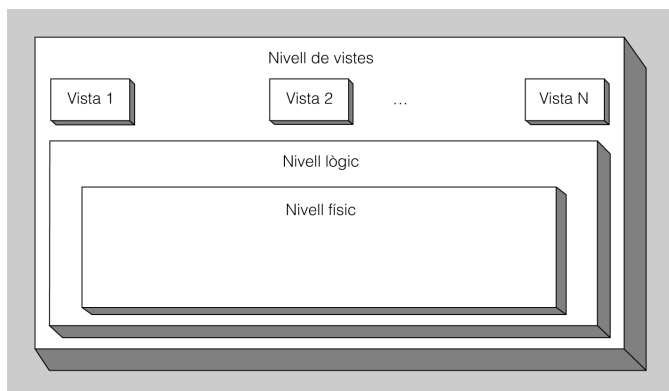
- Descriu totes les dades emmagatzemades en la BD i les seves interrelacions, mitjançant un nombre no gaire elevat d'estructures força simples (típicament, taules).
- La implementació d'aquestes estructures lògiques pot comportar la presència d'estructures molt més complexes a nivell físic. Però els usuaris del nivell lògic no s'han de preocupar d'aquesta complexitat. Ni tan sols necessiten conèixer-la.
- Els administradors de BD treballen habitualment amb aquest nivell d'abstracció.

3. **Nivell de vistes:** és el nivell d'abstracció més alt.

- La majoria dels usuaris no necessiten conèixer tota l'estructuració lògica de la BD amb què treballen. Tractant-se d'una BD gran, a més, conèixer tota la seva estructura pot comportar un esforç considerable.
- D'altra banda, sovint, i per motius tant de seguretat com de privacitat, no resulta convenient que els usuaris tinguin accés a totes les dades, sinó solament a la part que estrictament necessiten per realitzar la seva feina.
- Cada vista només descriu una part de la BD. L'establiment de vistes simplifica la interacció de l'usuari amb el sistema, el fa més segur, i proporciona més privacitat. Es poden establir diferents vistes, segons les necessitats, sobre la mateixa BD.

En la figura 1.6, es poden veure els diferents nivells d'abstracció utilitzats per facilitar la interacció dels usuaris amb les BD.

**FIGURA 1.6.** Nivells d'abstracció de dades



### 1.3 Els SGBD

Els **SGBD** són un tipus de programari que té com a finalitats la gestió i el control de les BD.

És interessant conèixer l'evolució d'aquest tipus de programari al llarg de la seva història, i els objectius que tots ells persegueixen amb més o menys encert. També cal destacar que hi ha nocions relatives tant a l'arquitectura dels SGBD com a les aplicacions que els fan servir. També s'ha de destacar que hi ha diferents tipologies d'usuaris i administradors de BD, i llenguatges que tots han d'utilitzar per comunicar-se amb els sistemes gestors.

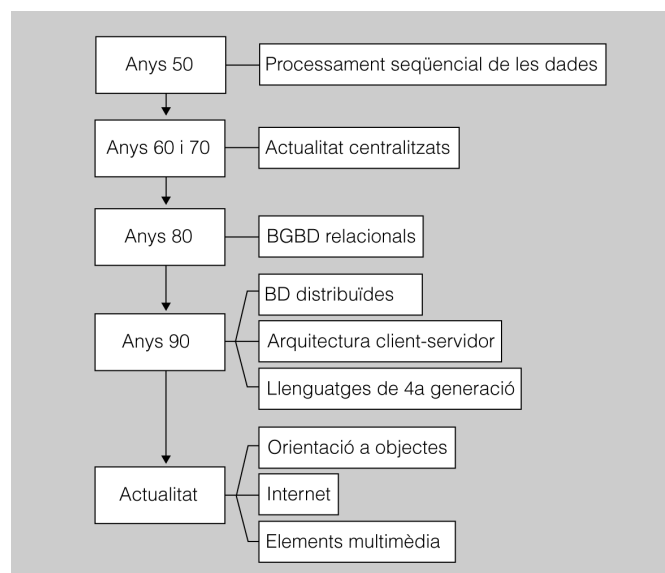
#### 1.3.1 Evolució dels SGBD

Per tal d'entendre millor per què els SGBD són avui dia tal com els coneixem, convé repassar breument la seva història.

Igual que en altres àmbits de programari (com, per exemple, en el dels sistemes operatius), l'evolució dels SGBD ha estat, sovint, intrínsecament lligada a l'evolució del maquinari.

La figura 1.7 mostra un esquema de les etapes evolutives per les quals han passat els SGBD, en el qual se n'indiquen les principals característiques.

**FIGURA 1.7.** Evolució històrica dels SGBD



## **Anys cinquanta: processament seqüencial**

Inicialment, l'únic maquinari disponible per emmagatzemar la informació consistia en paquets de cintes perforades i en cintes magnètiques. Aquests dispositius només es podien llegir de manera seqüencial i, per tant, el programari de l'època estava limitat per aquesta circumstància. Com que la grandària de les dades a processar era molt superior a la de la memòria principal de les computadores, els programes només podien realitzar processos per lots, de la manera següent:

- Obtenint les dades en un ordre determinat (des d'una o més cintes).
- Fent algun càlcul sobre les dades.
- Escrivint el resultat (en una nova cinta).

### **Exemple de processament seqüencial**

Imaginem que una empresa necessita actualitzar els preus dels productes que ofereix: en primer lloc, caldria gravar els increments dels preus en targetes perforades.

A continuació, s'aniria llegint el paquet de les cintes perforades anteriors, sincronitzadament amb la cinta mestra que continguéss totes les dades relatives als productes. Les targetes perforades haurien de respectar l'ordre dels registres del fitxer de productes contingut en la cinta.

Amb totes les dades relatives als productes contingudes en la cinta mestra, més els preus actualitzats en funció dels nous valors reflectits en les targetes perforades, es gravaria una nova cinta, que passaria a ser la nova cinta mestra dels productes de l'empresa.

## **Anys seixanta i setanta: sistemes centralitzats**

Durant la major part de les dues dècades dels anys seixanta i setanta, els SGBD van tenir una estructura centralitzada, com corresponia als sistemes informàtics d'aleshores: un gran ordinador per a cada organització que se'l pogués costejar, i una xarxa de terminals no intel·ligents, sense capacitat pròpia per processar dades.

Inicialment, només es feien servir per gestionar processos per lots amb grans volums de dades. Posteriorment, amb l'aparició dels terminals, de vegades connectats mitjançant la línia telefònica, es van anar elaborant aplicacions transaccionals, per exemple per reservar i comprar bitllets en línies de transports, o per realitzar operacions financeres.

Els programes encara estaven molt lligats al nivell físic, i s'havien de modificar sempre que es feien canvis en el disseny de la BD, ja que aquests canvis implicaven, al seu torn, modificacions en l'estructura física de la BD. El personal que realitzava aquestes tasques havia d'estar altament qualificat.

## **Anys vuitanta: SGBD relacionals**

Tot i que des del principi dels anys setanta ja s'havia definit el model relacional, i l'accés no procedimental a les dades organitzades seguint aquest model, no va ser fins als anys vuitanta quan van anar apareixent SGBD relacionals en el mercat.

La raó d'aquesta demora en l'ús dels sistemes relacionals va ser el pobre rendiment que oferien inicialment els productes relacionals en comparació amb les BD jeràrquiques i en xarxa. Però la innovació en el maquinari, primer amb els miniordinadors i posteriorment amb els microordinadors, va comportar un cert abaratiment de la informàtica i la seva extensió a moltes més organitzacions.

Fins aleshores, la feina dels programadors que treballaven amb BD prerelacionals havia estat massa feixuga, ja que, d'una banda, havien de codificar les seves consultes de manera procedimental, i d'una altra, havien d'estar pendents del seu rendiment i fer consideracions d'índole física a l'hora de codificar-les.

Però a causa de l'expansió de la informàtica que va tenir lloc durant la dècada que comentem, calia simplificar el desenvolupament de les aplicacions. Els SGBD ho van aconseguir, tot independitzant els programes dels aspectes físics de les dades.

### SQL

El 1986, l'Institut Nacional Nordamericà de Normalització (American National Standards Institute, o ANSI, en anglès) va publicar les primeres normes que enuncien la sintaxi i la semàntica de l'SQL.

A més, l'aparició del llenguatge de consulta estructurat (*structured query language*, o SQL, en anglès) i, sobretot, la seva estandardització a partir de l'any 1986 van facilitar enormement l'ús dels sistemes relacionals i, per tant, la seva implantació massiva.

Finalment, les BD relacionals van poder competir, fins i tot, en matèria de rendiment amb les jeràrquiques i amb les estructurades en xarxa, amb la qual cosa van acabar reemplaçant les seves competidores en la majoria dels casos.

## Anys noranta: BD distribuïdes, architectures client/servidor, i llenguatges de quarta generació

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 (*personal computers*, o PC, en anglès), 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).

En funció del nombre de SGBD utilitzats, els sistemes distribuïts poden ser de dos tipus:

- **Homogenis**, si tots els nodes utilitzen el mateix SGBD. Les interaccions entre els diferents nodes són més senzilles. Però les actualitzacions del sistema gestor implicaran necessàriament a tots els nodes.
- **Heterogenis**, si cada node utilitza un SGBD diferent. Les interaccions entre els diferents nodes poden ser més complicades. Però hi haurà més flexibilitat a l'hora d'actualitzar el sistema gestor de cada node.

Els punts a favor dels sistemes distribuïts són fonamentalment dos:

- **Rendiment**. Si el sistema està ben dissenyat, la majoria de les operacions es realitzaran amb dades emmagatzemades localment. D'aquesta manera

les respostes seran més ràpides, disminuirà la despesa en comunicacions, i s'evitarà la dependència d'un node central col·lapsat.

- **Disponibilitat.** Els sistemes distribuïts són més resistent a les aturades que no pas els centralitzats. En un sistema centralitzat, l'aturada del node central atura tot el sistema. En canvi, en un sistema distribuït, si un dels nodes queda temporalment fora de servei per qualsevol eventualitat, la resta continuarà funcionant normalment, i podrà donar servei sempre que no es necessitin les dades emmagatzemades en el node aturat. Però, a més, segons quin esquema de disseny s'hagi seguit en fer la distribució, si les dades del node aturat estan duplicades en un altre, continuaran estant disponibles.

Però, evidentment, no tot són avantatges. Per exemple, en el cas dels sistemes heterogenis, sovint és necessari realitzar conversions de dades per permetre la comunicació dels diferents nodes entre ells. A més, en general, la comunicació és més complexa i, per tant, la quantitat d'errors i de problemes derivats d'aquest fet que s'han de controlar és molt més gran que en un sistema centralitzat.

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.

El funcionament dels sistemes basats en aquest tipus d'arquitectura és, esquemàticament, el següent: dos processos s'executen en un mateix sistema o en dos de diferents, de tal manera que un fa de client (o peticionari d'un servei), i l'altre de servidor (o proveïdor del servei demanat).

La classificació dels processos en les categories de client i de servidor és de tipus lògic (i no físic) i, per tant, cal tenir en compte alguns aspectes que deriven d'aquesta circumstància, com ara els següents:

- Un client pot demanar serveis a diversos servidors.
- Un servidor pot rebre peticions de molts clients.
- El client i el servidor poden residir en un mateix sistema.

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.

Es tracta de llenguatges molt senzills, però al mateix temps molt potents, especialitzats en el desenvolupament d'aplicacions centrades en l'accés a BD. Ofereixen moltes facilitats per definir, generalment de manera visual, finestres des de les quals es poden consultar, introduir, modificar o esborrar dades, fins i tot en entorns client/servidor.

#### Arquitectura client/servidor

Una arquitectura client/servidor es caracteritza pel fet de disposar d'un sistema en xarxa on hi ha uns ordinadors que actuen com a servidors de peticions i uns altres (els clients) que demanen serveis.

#### 4GL

Aquests són alguns entorns actuals de desenvolupament que utilitzen llenguatges de quarta generació: eDeveloper, de Magic Software, Oracle Developer, d'Oracle Corporation, SAS System, de SAS Institute Inc., etc.

## Tendències actuals: orientació a objectes, Internet, i elements multimèdia

Actualment, els SGBD relacionals acaparen el mercat. Han evolucionat de tal manera que ja no tenen competidors en rendiment, fiabilitat o seguretat. D'altra banda, ja no necessiten habitualment tasques de manteniment planificades que en comportin l'aturada periòdica. Per tant, la disponibilitat que ofereixen és molt elevada, ja que s'acosta a les vint-i-quatre hores de tots els dies de l'any.

Però, al mateix temps, tots estan immersos en un complex procés de transformació per tal d'adaptar-se a les innovacions tecnològiques de més èxit:

**1. Les tecnologies multimèdia.** Els tipus de dades que tradicionalment admetien els SGBD ara es veuen incrementats per altres de nous que permeten emmagatzemar imatges i sons.

### Exemple d'incorporació de tecnologies multimèdia en un SGBD

D'aquesta manera, per exemple, la taula que emmagatzemi les dades personals dels empleats d'una empresa determinada podrà contenir la foto de cadascun, la qual cosa pot resultar especialment interessant si l'organització disposa d'un entorn virtual de treball (de tipus intranet, per exemple) en què els correus electrònics incorporin la foto del remitent, a fi de permetre la identificació visual.

**2. L'orientació a objectes.** Aquest paradigma de la programació ha acabat influint en l'orientació de molts SGBD, que segueixen alguna de les dues línies esbossades a continuació:

- SGBD relacionals amb objectes, els quals admeten tipus abstractes de dades (TAD), a més dels tipus tradicionals.
- SGBD orientats a objectes, que estructurin les dades en classes, les quals comprenen tant les dades (atributs) com les operacions sobre elles (mètodes). Les classes s'estructuren jeràrquicament, de tal manera que les de nivells inferiors (subclasses) hereten les propietats de les de nivells superiors (superclasses).

**3. Internet.** 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.

Hi ha dues maneres d'incorporar la tecnologia XML en els SGBD:

- Els **SGBD relacionals amb suport per a XML**, que en el fons continuen essent relacionals i que, per tant, emmagatzemen tota la informació de manera tabular. La seva utilitat principal és que les dades que retornen les consultes sobre la BD poden estar expressades en format XML, si així es demana al sistema gestor.
- Els **sistemes gestors per a BD natives XML**, que no són en realitat relacionals, sinó més aviat jeràrquics, i que no solament estan en condicions d'oferir els resultats de les consultes en format XML, sinó que també emmagatzemen la informació en documents XML. El llenguatge estàndard de consultes per a aquest tipus de SGBD s'anomena XQuery.

#### XQuery

L'XQuery és un llenguatge de consultes dissenyat per consultar col·leccions de dades XML, creat pel World Wide Web Consortium (conegut abreviadament com a W3C). El W3C és un consorci internacional que produeix estàndards per a la World Wide Web (coneguda abreviadament com a WWW).

La utilització de dades estructurades mitjançant l'estàndard XML resulta especialment interessant en l'intercanvi d'informació entre sistemes basats en plataformes poc compatibles entre elles.

### 1.3.2 Objectius i funcionalitats dels SGBD

Tots els SGBD del mercat volen assolir una sèrie d'objectius i oferir una sèrie de funcionalitats, amb més o menys encert, que actualment es consideren indispensables per al bon funcionament de qualsevol sistema d'informació:

- Possibilitar les consultes no predefinides de qualsevol complexitat.
- Garantir la independència física i la independència lògica de les dades.
- Evitar o solucionar els problemes derivats de la redundància.
- Protegir la integritat de les dades.
- Permetre la concurrència d'usuaris.
- Contribuir a la seguretat de les dades.

#### Possibilitar les consultes no predefinides de qualsevol complexitat

Els usuaris autoritzats d'un SGBD han de poder plantejar directament al sistema qualsevol consulta sobre les dades emmagatzemades, de la complexitat que sigui necessària, tot respectant, això sí, les regles sintàctiques perquè la sentència sigui correcta.

A continuació, el SGBD ha de ser capaç de respondre ell mateix a la consulta formulada, sense que sigui necessari recórrer a cap aplicació externa.

Quan no existien ni les BD ni els sistemes gestors, per tal d'obtenir un subconjunt de les dades emmagatzemades en un fitxer, hi havia dues possibilitats:

- Llançar un llistat seqüencial de totes les dades i, a continuació, seleccionar manualment les que interessessin.
- Escriure el codi font d'un programa específic per resoldre la consulta en qüestió, compilar-lo i executar-lo.

Els SGBD actuals, en canvi, estan perfectament capacitats per interpretar directament les consultes, expressades habitualment com a sentències formulades en el llenguatge de consulta SQL.

Evidentment, això no vol dir que no es puguin continuar produint programes que incorporin consultes mitjançant les quals accedeixen a BD. De fet, aquesta és l'opció més encertada i còmoda si es tracta de consultes que s'han de repetir al llarg del temps.

### **Garantir la independència física i la independència lògica de les dades**

Cal garantir la màxima independència física de les dades respecte als processos usuaris, en general (és a dir, tant pel que fa a les consultes interpretades pel SGBD com als programes externs que accedeixen a la BD), de tal manera que es puguin dur a terme tot tipus de canvis tecnològics d'índole física per millorar el rendiment (com ara afegir o treure un índex determinat), sense que això impliqui haver de modificar ni les consultes a la BD ni les aplicacions que hi accedeixen.

De manera similar, també és desitjable la independència lògica de les dades, la qual implica que les modificacions en la descripció lògica de la BD (per exemple, afegir un nou atribut o suprimir-ne un altre) no han d'impedir l'execució normal dels processos usuaris no afectats per aquelles.

I, pel que fa a la independència lògica de les dades, fins i tot pot interessar (i, de fet, aquesta és una opció freqüent) que convisquin diferents visions lògiques d'una mateixa BD, en funció de les característiques concretes dels diferents usuaris o grups d'usuaris.

### **Evitar o solucionar els problemes derivats de la redundància**

Tradicionalment, la repetició de les dades s'ha considerat una cosa negativa, ja que comporta un cost d'emmagatzematge innecessari. Avui en dia, però, aquesta característica, tot i ser certa, gairebé no es té en compte, a causa de l'abaratiment dels discos durs i de l'augment de la seva capacitat i rendiment.

Però hi ha un altre aspecte a considerar que no ha perdut vigència, i és el fet que la repetició de les dades és perillosa, ja que quan s'actualitzen poden perdre la integritat. Quan es modifica el valor d'una dada que està repetida, s'han de modificar simultàniament els valors de les seves repeticions perquè es mantingui la coherència entre totes.

---

Són dades íntegres les que es mantenen senceres i correctes.



La **redundància** consisteix en la repetició indesitjada de les dades, que incrementa els riscos de pèrdua d'integritat d'aquestes quan s'actualitzen.

Malgrat tot, els SGBD han de permetre al dissenyador de BD la definició de dades repetides, ja que de vegades (sobretot en matèria de fiabilitat, de disponibilitat o de costos de comunicació) és útil mantenir certes rèpliques de les dades.

Ara bé, en tots aquests casos, l'objectiu de l'SGBD ha de ser garantir l'actualització correcta de totes les dades allà on estiguin duplicades, de manera automàtica (és a dir, sense que l'usuari del SGBD s'hagi d'encarregar de res).

Un altre tipus de duplicitat admissible és la que constitueixen les anomenades **dades derivades**. Es tracta de dades emmagatzemades en la BD, que en realitat són el resultat de càlculs fets amb altres dades també presents en la mateixa BD.

Les dades derivades també poden ser acceptables, tot i que comporten una repetició evident d'algunes dades, si permeten fer consultes de caire global molt ràpidament, sense haver d'accedir a tots els registres implicats.

Però també aquí, el SGBD s'ha d'encarregar d'actualitzar degudament les dades derivades en funció dels canvis soferts per les dades primitives de les quals depenen.

### Protegir la integritat de les dades

A més de la redundància, hi ha molts altres motius que poden fer malbé la consistència de les dades, com ara els següents:

- Els errors humans.
- Les deficiències en la implementació dels algorismes de les aplicacions.
- Les avaries dels suports físics d'emmagatzematge.
- Les transaccions incompletes com a conseqüència de les interrupcions del subministrament elèctric.

Els SGBD han de protegir la integritat de les dades en tots aquests casos. Per a això disposen, d'una banda, de les regles d'integritat, també anomenades *restriccions*, i d'una altra, dels sistemes de restauració basats en còpies de seguretat.

Mitjançant les **regles d'integritat**, el sistema valida automàticament certes condicions en produir-se una actualització de dades, i l'autoritza si les compleix, o denega el permís en cas contrari.

### Exemple de restricció del model

Un SGBD relacional mai no acceptarà que una taula emmagatzemi registres amb una clau primària idèntica, perquè aleshores la clau no serviria per identificar inequívocament els registres entre ells.

Les regles d'integritat poden ser de dos tipus:

- **Restriccions del model.** Són regles inherents al model de dades que utilitza el SGBD (com ara el model relacional). El sistema les incorpora predefinides, i sempre s'acompleixen.
- **Restriccions de l'usuari.** Són regles definides pels usuaris (dissenyadors i administradors, fonamentalment) de la BD, que no incorpora *a priori* el SGBD, i que serveixen per modelitzar aspectes específics del món real.

### Exemple de restricció de l'usuari

En una taula que emmagatzema els alumnes d'un centre docent, es vol evitar que hi hagin alumnes matriculats en cicles formatius de grau superior que fossin menors d'edat, ja que la normativa vigent no ho admet. Aleshores, cal establir una restricció en aquest sentit per calcular si la diferència entre la data del sistema en el moment de la matrícula i la data de naixement de cada alumne és igual o superior als 18 anys. En aquest cas, el sistema permetria la matrícula, i en cas contrari la prohibiria.

Els SGBD també proporcionen eines per realitzar periòdicament **còpies de seguretat de les dades** (o *backups*, en anglès) que permeten restaurar les dades malmeses i retornar-les a un estat consistent.

Ara bé, els valors restaurats es correspondran amb els que hi havia en el moment de realitzar la còpia de seguretat, abans de l'incident que origina la restauració, i per tant mai no estaran del tot actualitzats, encara que siguin correctes.

### Permetre la concurrència d'usuaris

Un objectiu fonamental de tot SGBD és possibilitar de manera eficient l'accés simultani a la BD per part de molts usuaris. A més, aquesta necessitat ja no està circumscrita només a grans companyies o a administracions públiques amb molts usuaris, sinó que cada cop és més freqüent, a causa de l'expansió d'Internet i l'èxit de les pàgines dinàmiques, allotjades en servidors web que han d'incorporar un SGBD.

Hem de considerar dues tipologies d'accessos concurrents, amb problemàtiques ben diferenciades:

- **Accessos de consulta de dades.** Poden provocar problemes de rendiment, derivats sobretot de les limitacions dels suports físics disponibles (per exemple, si la lentitud de gir del disc dur que conté la BD, o del moviment del braç que porta incorporat el capçal, no permeten atendre degudament totes les peticions d'accés que rep el sistema).

---

La concurrència d'usuaris en una BD consisteix en l'accés simultani a la BD per part de més d'un usuari.

---

- **Accessos de modificació de dades.** Les peticions simultànies d'actualització d'unes mateixes dades poden originar problemes d'interferència que tinguin com a conseqüència l'obtenció de dades errònies i la pèrdua d'integritat de la BD.

Per tractar correctament els problemes derivats de la concurrència d'usuaris, els SGBD fan servir fonamentalment dues tècniques: les **transaccions** i els **bloquejos**.

Una **transacció** consisteix en un conjunt d'operacions simples que s'han d'executar com una unitat.

Les operacions incloses dins d'una transacció mai no s'han d'executar parcialment. Si per algun motiu no s'han pogut executar totes correctament, el SGBD ha de desfer automàticament els canvis produïts fins aleshores. D'aquesta manera, es podrà tornar a llançar l'execució de la mateixa transacció, sense haver de fer cap modificació en el codi de les diferents operacions que inclogui.

#### Exemple de transacció

Imaginem que el conveni col·lectiu d'una empresa determina que els salaris dels treballadors s'han d'apujar el mes de gener un 3%. La millor opció consistirà a actualitzar la taula d'empleats i, més concretament, els valors del camp que recull els sous dels empleats, mitjançant una consulta d'actualització que modifiqui totes aquestes dades i les incrementi en un 3%.

Si volem garantir que l'actualització del salaris no quedi a mitges, haurem de fer que totes les instruccions que impliqui aquest procés es comportin de manera transaccional, és a dir, que s'executin totes o bé que no se n'executi cap.

Però també es pot donar la situació en què diferents transaccions vulguin accedir a la BD simultàniament. En aquests casos, encara que cada transacció, individualment considerada, sigui correcta, no es podria garantir la consistència de les dades si no fos per l'ús de la tècnica del bloqueig.

Un **bloqueig** consisteix a impedir l'accés a determinades dades durant el temps en què siguin utilitzades per una transacció. Així s'aconsegueix que les transaccions s'executin com si estiguessin aïllades, de tal manera que no es produeixen interferències entre elles.

#### Exemple de bloqueig

Imaginem que el departament de recursos humans de l'empresa de l'exemple anterior disposa d'una aplicació que li proporciona certes dades de caire estadístic sobre de les remuneracions dels empleats, com ara el salari mitjà.

Si es vol executar de manera concurrent la transacció descrita, que incrementa els sous en un 3%, i al mateix temps es llança l'execució de l'aplicació que calcula el salari mitjà de tots els empleats de l'empresa, el resultat obtingut per aquest programa probablement serà erroni.

Per tal de garantir la correcció del càlcul, s'haurà de bloquejar una de les dues transaccions mentre l'altra s'executa.

#### COMMIT i ROLLBACK

La instrucció COMMIT indica, al SGBD, que un conjunt d'operacions determinat s'ha d'executar de manera transaccional. L'operació ROLLBACK desfà els canvis produïts en cas que les operacions d'una transacció s'hagin executat parcialment.

Si es bloqueja l'actualització de dades, el salari mitjà estarà calculat a partir dels sous antics (és a dir, abans de ser actualitzats).

En canvi, si es bloqueja el programari estadístic, en primer lloc s'actualitzaran totes les dades, i després es calcularan els resultats a partir dels nous valors del camp que emmagatzemi el salari.

Els bloquejos provoquen esperes i retencions, i per això les noves versions dels diferents SGBD del mercat s'esforcen a minimitzar aquests efectes negatius.

## Contribuir a la seguretat de les dades

### Exemple de dades confidencials

Resulta evident la necessitat de restringir l'accés als secrets militars o, fins i tot, comercials (com ara les dades comptables). Però també s'ha de respectar la privacitat, fins i tot per imperatiu legal, en altres vessants aparentment més modestos, però en realitat no menys importants, com són les dades personals.

L'expressió **seguretat de les dades** fa referència a la seva confidencialitat. Sovint, l'accés a les dades no ha de ser lliure o, com a mínim, no ho ha de ser totalment.

Els **SGBD** han de permetre definir autoritzacions d'accés a les BD, tot establint permisos diferents en funció de les característiques de l'usuari o del grup d'usuaris.

Actualment, els SGBD permeten definir autoritzacions a diferents nivells:

- Nivell global de tota la BD
- Nivell d'entitat
- Nivell d'atribut
- Nivell de tipus d'operació

### Exemples de drets d'accés

Els usuaris del departament de comptabilitat potser no haurien de tenir accés a l'entitat que emmagatzema les dades personals dels empleats de l'empresa, a diferència de l'usuari que ostenta el càrrec de director general, que les podrà consultar per tal d'optimitzar la ubicació dels treballadors el l'organigrama en funció del respectiu perfil, o també dels usuaris del departament de recursos humans, que haurien de poder fins i tot modificar-les.

En general, els usuaris no haurien de tenir accés als atributs que emmagatzemen l'adreça particular dels empleats, a no ser que es tracti del personal de recepció, si resulta que és l'encarregat d'enviar-los certa correspondència a domicili (com ara les nòmines o els certificats de retencions d'IRPF), i per tant hauria, si més no, de poder consultar-los.

Aquests mecanismes de seguretat requereixen que cada usuari es pugui identificar. El més freqüent és utilitzar un nom d'usuari i una contrasenya associada per a cada usuari. Però també hi ha qui fa servir mecanismes addicionals, com ara targetes magnètiques o de reconeixement de la veu. Actualment, s'investiga en altres direccions com, per exemple, en la identificació personal mitjançant el reconeixement de les empremtes dactilars.

Un altre aspecte a tenir en compte en parlar de la seguretat de les dades és la seva encriptació. Molts SGBD ofereixen aquesta possibilitat, en alguna mesura.

Les **tècniques d'encryptació** permeten emmagatzemar la informació utilitzant codis secrets que no permeten accedir a les dades a persones no autoritzades i que, per tant, no disposen dels codis esmentats.

L'encryptació pot fer disminuir el rendiment en l'accés a les dades, ja que comporta la utilització d'algoritmes addicionals en les operacions de consulta. Per això se n'ha de dosificar l'ús. Ara bé, sempre que sigui possible, és convenient encryptar les contrasenyes.

### 1.3.3 Llenguatges de SGBD

La comunicació entre els SGBD i els seus usuaris s'ha de realitzar mitjançant algun tipus de llenguatge. Els llenguatges de BD es poden classificar en dues grans tipologies segons la finalitat:

1. **Llenguatges de definició de dades** (*data definition languages*, en anglès, o DDL). Estan especialitzats en la definició de l'estructura de les BD mitjançant l'especificació d'esquemes.
2. **Llenguatges de manipulació de dades** (*data management languages*, en anglès, o DML). Possibiliten la consulta, modificació i eliminació, de les dades emmagatzemades, i també la inserció de noves informacions. Podem considerar l'existència de dos subtipus, bàsicament:
  - **Procedimentals**. Requereixen especificar no solament les dades que es necessiten, sinó també els procediments que s'han de seguir per obtenir-les. S'utilitzaven de manera exclusiva abans de l'arribada del model relacional. Actualment, es continuen utilitzant, però només quan cal optimitzar algun procés que no rendeix prou, pel fet d'estar implementat de manera declarativa. Són més eficients que els declaratius, però més complicats, ja que exigeixen tenir certs coneixements sobre el funcionament intern del SGBD utilitzat.
  - **Declaratius**. Només requereixen especificar quines dades es necessiten, sense que calgui especificar com s'han d'obtenir. Són més senzills d'aprendre que els procedimentals, però també menys eficients.

El llenguatge més utilitzat per interaccionar amb els SGBD relacionals és l'**SQL**.

L'SQL engloba les dues tipologies de llenguatges de BD descrites. Les seves operacions es poden classificar, doncs, en un dels dos tipus esmentats (DDL i DML) amb finalitats pedagògiques, però en realitat totes formen part d'un únic llenguatge.

### Exemples d'operacions DDL i DML del llenguatge SQL

Com a instruccions de tipus DML, podem esmentar SELECT (per fer consultes), i també INSERT, UPDATE i DELETE (per realitzar el manteniment de les dades).

I com a instruccions de tipus DDL, podem considerar CREATE TABLE (que ens permet definir les taules, les seves columnes i les restriccions que calgui).

En relació al component DML de l'SQL, cal dir que és fonamentalment declaratiu, tot i que té possibilitats procedimentals, que es poden explotar en diferents SGBD:

---

PostgreSQL és un SGBD  
distribuït amb llicència BSD  
(Berkeley Software  
Distribution)

---



Logotip de PostgreSQL

- **PL/SQL**, llenguatge procedimental per treballar amb els SGBD creats per Oracle.
- **PL/PgSQL**, similar al PL/SQL d'Oracle, però dissenyat per treballar amb PostgreSQL.

Cal dir que, a més del respectiu llenguatge nadiu de BD (habitualment, SQL), els SGBD ofereixen, des de ja fa molt de temps, dues possibilitats més per incrementar la productivitat en el treball amb BD, que són els **llenguatges de quarta generació** i les interfícies visuals, sovint proporcionades dins de l'entorn d'una sola eina.

Sovint, l'accés a les BD també es fa des d'aplicacions externes al SGBD, escrites en llenguatges de programació (com per exemple C, Java, etc.), els quals normalment no incorporen instruccions pròpies que permetin la connexió amb BD.

Per respondre a aquesta necessitat, hi ha dues opcions:

1. Realitzar, dins del programa, crides a diferents funcions que són en llibreries que implementen estàndards de connectivitat de BD amb programes escrits en certs llenguatges, com ara els següents:
  - **ODBC** (*open data base connectivity*), sistema creat per Microsoft i compatible amb molts sistemes com, per exemple, Informix, MS Access, MySQL, Oracle, PostgreSQL, SQL Server, etc.
  - **JDBC** (*Java data base connectivity*), per realitzar operacions amb BD des d'aplicacions escrites en Java.
2. Hostatjar les sentències del llenguatge de BD que siguin necessàries, dins d'un programa amfitrió escrit en el llenguatge de programació utilitzat. És imprescindible que el compilador utilitzat accepti la introducció de sentències escrites en el llenguatge de BD utilitzat (que normalment serà l'SQL).

### 1.3.4 Usuaris i administradors

Les persones que treballen amb SGBD es poden classificar com a usuaris en sentit estricte, els quals simplement interactuen amb el sistema (tot i que de diferents

maneres i amb diferents finalitats), o bé com a administradors, si a més realitzen tasques de gestió i control.

## Usuaris d'SGBD

Podem diferenciar tres categories d'usuaris de SGBD en funció de la manera en què interactuen amb el sistema: externs, sofisticats i programadors d'aplicacions.

**1. Usuaris externs.** Són usuaris no sofisticats, que no interactuen directament amb el sistema, sinó mitjançant alguna aplicació informàtica desenvolupada prèviament per altres persones amb aquesta finalitat.

### Exemple d'usuari extern

Qualsevol persona assumeix aquest rol quan treu diners d'un caixer automàtic, ja que accedeix a la BD de l'entitat financera identificant-se mitjançant una targeta magnètica i un número d'identificació personal secret (*personal identification number*, en anglès, o PIN).

Una vegada autoritzada a entrar en el sistema, podrà realitzar diferents operacions de consulta o, fins i tot, d'actualització. En el cas plantejat, després de treure diners, el saldo del compte corrent associat a la targeta patirà el decrement corresponent.

**2. Usuaris sofisticats.** Interactuen directament amb el sistema, sense utilitzar les interfícies proporcionades per programes intermediaris. Formulen les consultes en un llenguatge de BD (normalment, SQL), des de dins de l'entorn que el SGBD posa a la seva disposició. Tradicionalment, aquest entorn ha estat una consola en què es podien escriure les consultes, però cada vegada són més freqüents entorns que permeten construir les consultes de mode visual, com autèntiques eines CASE.

**3. Programadors d'aplicacions.** Són professionals informàtics que creen els programes que accedeixen als SGBD i que, posteriorment, són utilitzats pels usuaris que hem anomenat *externs*. Aquestes aplicacions es poden desenvolupar mitjançant diferents llenguatges de programació i eines externes al SGBD. Però molts SGBD comercials també inclouen entorns propis de desenvolupament i llenguatges de quarta generació que faciliten enormement la generació de formularis i informes que permeten visualitzar i modificar les dades.

### Eines CASE

Les eines CASE (acrònim de *computer aided software engineering*, o enginyeria del programari assistida per ordinador) són aplicacions informàtiques destinades a augmentar la productivitat en el desenvolupament de programari reduint el cost del desenvolupament en termes de temps i de diners.

## Administradors d'SGBD

Els **administradors** són uns usuaris especials que realitzen tasques d'administració i control centralitzat de les dades, i gestionen els permisos d'accés concedits als diferents usuaris i grups d'usuaris, per tal de garantir el funcionament correcte de la BD.

Els administradors han d'actuar, evidentment, per solucionar les eventuals aturades del sistema, però la seva responsabilitat fonamental consisteix, justament, a evitar que es produeixin incidents.

La feina dels administradors no és fàcil, tot i que els SGBD incorporen cada vegada més eines per facilitar-la, i en la majoria dels casos amb interfície visual. Es tracta, per exemple, d'eines de monitoratge de rendiment, d'eines de monitoratge de seguretat, de verificadors de consistència entre índexs i dades, de gestors de còpies de seguretat, etc.

Una llista no exhaustiva de les tasques dels administradors podria ser la següent:

- Crear i administrar els esquemes de la BD.
- Administrar la seguretat: autoritzacions d'accés, restriccions, etc.
- Realitzar còpies de seguretat periòdiques.
- Controlar l'espai de disc disponible.
- Vigilar la integritat de les dades.
- Observar l'evolució del rendiment del sistema i determinar quins processos consumeixen més recursos.
- Assessorar els programadors i els usuaris sobre la utilització de la BD.
- Fer canvis en el disseny físic per millorar el rendiment.
- Resoldre emergències.

### 1.3.5 Components funcionals dels SGBD

El programari que conforma els SGBD es divideix en diferents **mòduls**, encarregats de les respectives funcionalitats que ha de garantir el sistema.

El components funcionals dels SGBD més importants són el gestor d'emmagatzemament i el processador de consultes.

#### Gestor d'emmagatzemament

Les BD corporatives tenen enormes requeriments d'espai d'emmagatzematge en disc. Les dades es transfereixen entre el disc en què estan emmagatzemades i la memòria principal de l'ordinador només quan és necessari. I, com que la transferència de dades cap al disc o des del disc és lenta en comparació amb la velocitat de la unitat central de processament, és molt important que el SGBD estructuri les dades de tal manera que se'n minimitzi la necessitat de transferència entre el disc i la memòria principal.

El gestor d'emmagatzemament proporciona la interfície entre les dades, considerades a baix nivell, i les consultes i els programes que accedeixen a la BD. Les dades s'emmagatzemen en disc fent servir el sistema d'arxius que proporciona



el sistema operatiu utilitzat. I el gestor tradueix les instruccions DML a ordres comprensibles pel sistema d'arxius a baix nivell.

Els components del gestor d'emmagatzemament són els següents:

- **Gestor d'autoritzacions i d'integritat.** Comprova que se satisfacin tant les restriccions d'integritat com les autoritzacions dels usuaris per accedir a les dades.
- **Gestor de transaccions.** Assegura que la BD es mantingui en un estat de consistència malgrat les fallades del sistema, i també que les transaccions concurrents no s'interfereixin entre elles.
- **Gestor d'arxius.** Gestiona la reserva d'espai d'emmagatzemament en disc i les estructures de dades utilitzades per representar la informació emmagatzemada en disc.
- **Gestor de memòria intermèdia.** Transfereix les dades des del disc a la memòria principal, i decideix quines dades s'han de tractar en memòria cau. Permet al sistema tractar amb dades de grandària molt superior a la de la memòria principal.

D'altra banda, el gestor d'emmagatzemament utilitza certes estructures de dades que formen part de la implementació física del sistema:

- **Arxius de dades.** Emmagatzemen la BD pròpiament considerada.
- **Diccionari de dades.** Emmagatzema les metadades relatives a tota l'estructura de la BD.
- **Índexs.** Proporcionen un accés ràpid a certes dades en funció dels seus valors.

## Processador de consultes

El processador de consultes ajuda el SGBD a simplificar l'accés a les dades. Les vistes a alt nivell contribueixen a assolir aquest objectiu, ja que eviten que els usuaris hagin de conèixer detalls de la implementació física del sistema. Però això no treu que el sistema no hagi de perseguir l'eficàcia en el processament de les consultes i de les actualitzacions de dades. De fet, el sistema ha de traduir les instruccions escrites, a nivell lògic, en un llenguatge no procedimental (típicament, SQL), a una seqüència d'operacions a nivell físic, amb uns mínims d'eficiència.

Els components del processador de consultes són els següents:

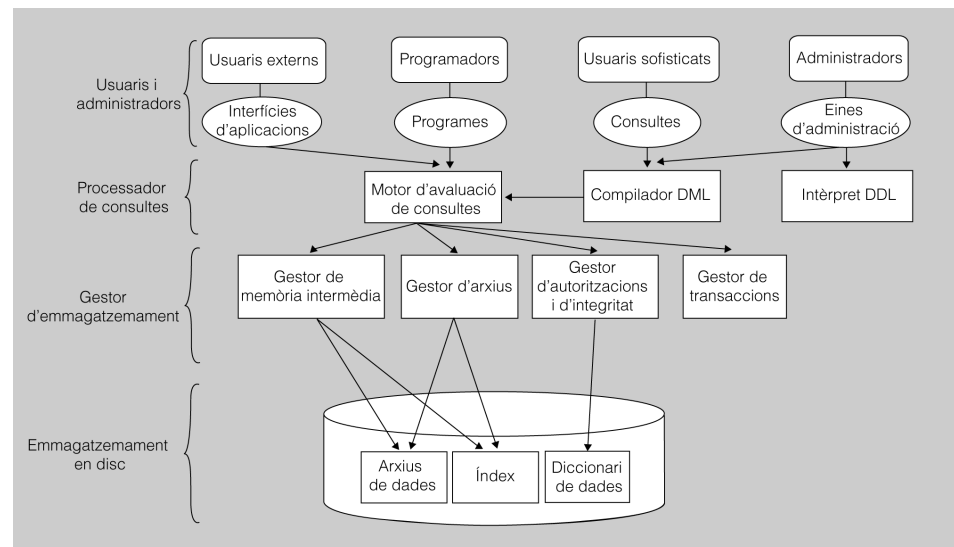
- **Intèrpret DDL.** Interpreta les instruccions de tipus DDL i registra les definicions en el diccionari de dades.
- **Compilador DML.** Tradueix les instruccions DML formulades en un llenguatge de consultes (normalment, SQL) a una sèrie d'instruccions a baix

nivell que pot interpretar el motor d'avaluació de consultes. En realitzar la traducció esmentada, un bon compilador DML també s'encarregarà de fer una optimització de consultes triant, entre totes les alternatives, la de menor cost.

- **Motor d'avaluació de consultes.** Executa les instruccions de baix nivell generades pel compilador DML.

La figura 1.8 mostra tots aquests components i les connexions entre ells.

**FIGURA 1.8.** Components funcionals dels SGBD



### 1.3.6 Diccionari de dades

Un diccionari de dades d'una base de dades és el conjunt de metadades que proporcionen informació sobre el contingut i l'organització de la base de dades.

Un **diccionari de dades**, segons *IBM Dictionary of Computing* es pot definir com un “repositori centralitzat d'informació sobre dades com ara el significat, relacions amb altres dades, origen, ús i format.”

De vegades el concepte de diccionari de dades o metadades també és conegut amb el nom de catàleg del sistema o, també, com a repositori de metadades.

Els diferents SGBD específics implementen de diverses formes el diccionari de dades, de forma que cada programari implementa amb un conjunt de taules i ofereix un conjunt de vistes a diferents tipus d'usuaris del sistema. Així doncs, normalment, un usuari amb privilegis d'administrador del sistema (DBA) pot visualitzar més informació i de tipus més específic que un usuari sense aquests privilegis.

#### DBA

DBA és l'acrònim de *Data Base Administrator* o Administrador de Bases de Dades, que és la persona encarregada de gestionar les BD, dintre del SGBD.

Els elements que es troben habitualment a un diccionari de dades inclouen:

- Definicions de l'esquema de la base de dades.
- Descripcions detallades de taules i camps, així com de tots els objectes de la base de dades (vistes, clusters, índexos, sinònims, funcions i procediments, triggers, etc.).
- Restriccions d'integritat referencial.
- Informació de control d'accés, com ara noms d'usuari, rols, i privilegis.
- Paràmetres d'ubicació de l'emmagatzemament.
- Estadístiques d'ús de la base de dades.

Tot aquest conjunt d'informació, s'emmagatzema en centenars de taules d'informació. Tot i que hi ha organismes que intenten estandarditzar l'organització d'aquesta meta-informació, la realitat és que cada distribuïdor de programari específic (cada SGBD) ofereix la seva pròpia estructura.

Un administrador del sistema o DBA haurà de conèixer com accedir i consultar tota aquesta informació consultant la guia de referència del sistema amb què treballi.

#### **El diccionari de dades en Oracle**

En Oracle és l'usuari SYS el propietari del diccionari de dades i l'únic que pot fer actualitzacions sobre la informació que conté. Tot i que, alterar o manipular el diccionari de dades és una operació d'administració que cal fer amb moltes precaucions, doncs pot provocar danys permanents.

En Oracle es creen tres tipus de vistes diferents, que permeten consultar informació sobre diferents tipus de dades. Així les vistes poden tenir un d'aquests prefixos:

- USER: Per a visualitzar informació sobre els objectes propietat de l'usuari.
- ALL: Per a consultar informació sobre els objectes on té accés l'usuari.
- DBA: Que dóna accés a tots els objectes del sistema, per a poder ser controlats i gestionats.

Així doncs, per exemple, es poden consultar el conjunt d'objectes a què es té accés des d'un usuari donat d'Oracle amb la següent sentència:

```
SELECT owner, object_name, object_type FROM ALL_OBJECTS;
```

#### **Diccionari de dades en MySQL**

En MySQL, en canvi, és la vista INFORMATION\_SCHEMA qui proporciona informació sobre les bases de dades que s'emmagatzemen en el sistema.

Per a obtenir certa informació sobre una base de dades concreta anomenada db\_name en un SGBD MySQL podríem executar una sentència com ara:

```
SELECT table_name, table_type, engine FROM information_schema.tables WHERE  
table_schema = 'db_name';
```



## 2. Models de bases de dades

Les bases de dades (BD) representen informàticament la part del món real del nostre interès, que prèviament hem conceptualitzat, mitjançant uns processos d'observació i d'abstracció.

Per tant, podem afirmar que les BD són models de la realitat. Però no hem de confondre aquesta característica de les BD amb el que s'entén per *model de dades* (o *model de base de dades*). L'estructura concreta de cada BD està construïda a partir del model de dades respectiu, triat pel dissenyador en funció de les necessitats i de les eines disponibles.

Els **models de dades** són uns conjunts d'eines lògiques per descriure les dades, les seves interrelacions, el seu significat i les restriccions a aplicar per tal de garantir-ne la coherència.

Tots els models de BD, en general, proporcionen tres tipus d'eines:

- **Estructures de dades.** Elements amb els quals es construeixen les BD, com ara taules, arbres, etc.
- **Regles d'integritat.** Restriccions que les dades hauran de respectar, com per exemple tipus de dada, dominis, claus, etc.
- **Operacions a realitzar amb les dades.** Altes, baixes, modificacions i consultes, com a mínim.

### 2.1 Arquitectura dels SGBD

El 1975, el comitè ANSI/X3/SPARC va proposar una arquitectura per als SGBD estructurada en tres nivells d'abstracció (intern, conceptual i extern), que resulta molt útil per separar els programes d'aplicació de la BD considerada des d'un punt de vista físic.

D'altra banda, també resulta interessant examinar l'arquitectura dels SGBD des d'un punt de vista funcional, ja que coneixent els diferents components, i els fluxos de dades i de control podem entendre el funcionament dels SGBD, i estarem en millors condicions d'utilitzar-los d'una manera òptima.

#### SGBD

Acronim de Sistema Gestor de Bases de Dades. És un programari especialitzat en la gestió de BD -bases de dades- (enteses, aquestes, com un conjunt estructurat d'informació).

### 2.1.1 Esquemes i nivells

Per gestionar les BD, els SGBD han de conèixer la seva estructura (és a dir, les entitats, els atributs i les interrelacions que conté, etc.). Els SGBD necessiten disposar d'una descripció de les BD que han de gestionar. Aquesta definició de l'estructura rep el nom d'esquema de la BD, i ha d'estar constantment a l'abast del SGBD perquè aquest pugui complir les seves funcions.

D'acord amb l'estàndard ANSI/X3/SPARC, hi hauria d'haver tres nivells d'esquemes:

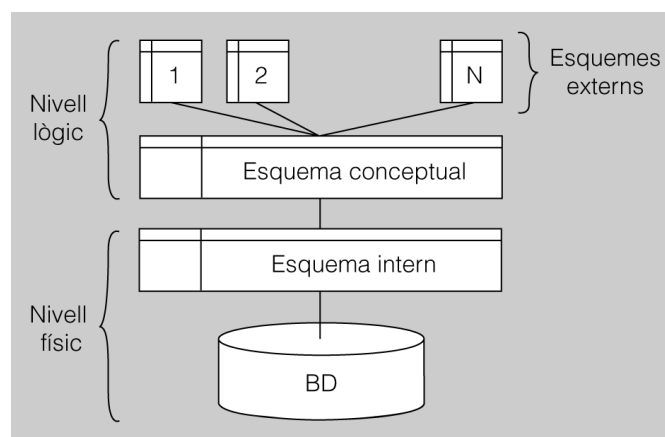
#### Comitè ANSI/X3/SPARC

És un grup d'estudi de l'*Standard Planning and Requirements Committee* (SPARC) de l'ANSI (*American National Standards Institute*), dins del comitè X3, que s'ocupa d'ordinadors i d'informàtica.

- En el **nivell extern** se situen les diferents visions lògiques que els processos usuaris (programes d'aplicació i usuaris directes) tenen de les parts de la BD que utilitzen. Aquestes visions s'anomenen *esquemes externs*.
- En el **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.
- En el **nivell físic** hi ha una única descripció física, que anomenem *esquema intern*.

La figura 2.1 representa la relació, d'una banda, entre el nivell físic i l'esquema intern; i, d'una altra banda, entre el nivell lògic i l'esquema conceptual, i els diferents esquemes externs (o vistes).

FIGURA 2.1. Esquemes i nivells



En definir un esquema extern, només s'inclouran els atributs i entitats que interessin, els podrem reanomenar, podrem definir dades derivades, podrem definir una entitat de tal manera que les aplicacions que utilitzen aquest esquema extern creguin que en són dues, o podrem definir combinacions d'entitats perquè en semblin una de sola, etc.

En l'esquema conceptual, es descriuran les entitats tipus, els seus atributs, les interrelacions i també les restriccions o regles d'integritat.

L'esquema intern o físic contindrà la descripció de l'organització física de la BD: camins d'accés (índexs, *hashing*, apuntadors...), codificació de les dades, gestió de l'espai, mida de la pàgina, etc.

## 2.2 Els models de bases de dades més comuns

Els models de dades més utilitzats al llarg del temps han estat els següents, exposats en ordre d'aparició:

1. Jeràrquic
2. En xarxa
3. Relacional
4. Relacional amb objectes / orientat a objectes

Actualment, hi ha algunes noves tendències incipients, gràcies al fenomen Internet, tot i que la informació en les empreses segueix utilitzant els models clàssics de BD.

### 2.2.1 Model jeràrquic

Les BD jeràrquiques es van concebre al principi del anys seixanta, i encara s'utilitzen gràcies al bon rendiment i la millor estabilitat que proporcionen amb grans volums d'informació.

Les **BD jeràrquiques** emmagatzemen la informació en una estructura jeràrquica que podem imaginar amb una forma d'arbre invertit, on cada node pare pot tenir diferents fills. El node superior, que no té pare, es coneix com a *arrel*. I els nodes que no tenen fills s'anomenen *fulles*.

Les dades s'emmagatzemen en forma de registres. Cada registre té un tuple de camps. Un conjunt de registres amb els mateixos camps forma un fitxer.

Però les BD jeràrquiques no ofereixen una perspectiva lògica per sobre de la física. Les relacions entre les dades s'estableixen sempre a nivell físic, mitjançant adreçaments físics al mitjà d'emmagatzemament utilitzat (és a dir, indicant sectors i pistes, en el cas més habitual dels discos durs).

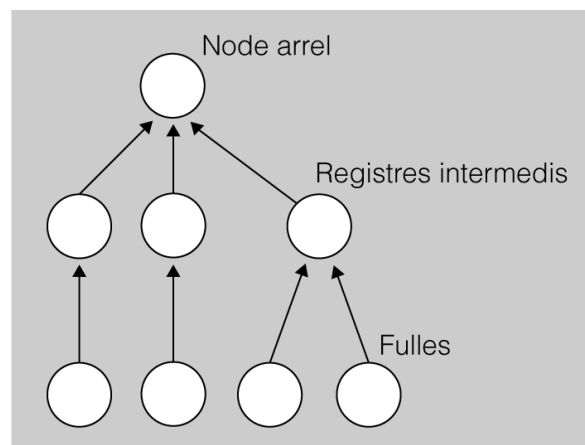
Així, doncs, les interrelacions s'estableixen mitjançant punters entre registres. Qualsevol registre conté l'adreça física del seu registre pare en el mitjà d'emmagatzemament utilitzat. Aquesta circumstància proporciona un rendiment molt bo: l'accés des d'un registre a un altre és pràcticament immediat.

Però, si bé el model jeràrquic optimitza les consultes de dades des dels nodes cap al node arrel, amb les consultes en sentit invers es produeix el fenomen contrari, ja que aleshores cal fer un recorregut seqüencial de tots els registres de la BD.

Altres limitacions d'aquest model consisteixen en la seva incapacitat per evitar la redundància (les repeticions indesitjades de les dades) o per garantir la integritat referencial (ja que els registres poden quedar "orfes" en esborrar-se el node pare respectiu). La responsabilitat per evitar aquests problemes queda a les mans de les aplicacions externes a la BD.

La figura 2.2 mostra l'estructura de nodes interrelacionats d'una BD jeràrquica.

FIGURA 2.2. Estructura d'una BD jeràrquica



Dos dels gestors de BD jeràrquiques que tenen més implantació són els següents:

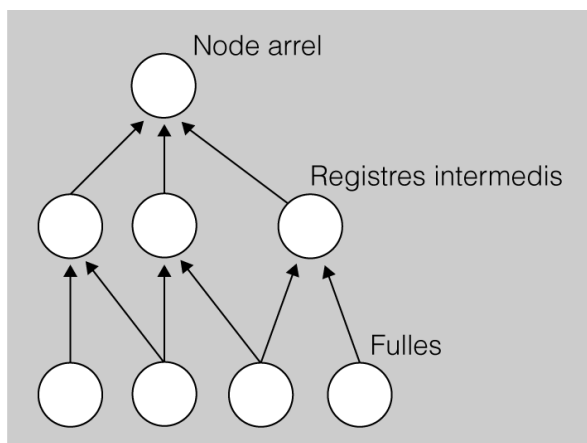
- IMS(*information management system*), de la multinacional nordamericana IBM.
- Adabas (*adaptable database system*), de l'empresa alemanya Software AG.

### 2.2.2 Model en xarxa

Al començament del anys setanta, en el mercat, van anar sorgint BD que segueixen un **model en xarxa**, semblant al model jeràrquic, amb registres interrelacionats mitjançant una estructura en forma d'arbre invertit, però més flexible, ja que permetia que els nodes tinguessin més d'un sol pare.

La figura 2.3 mostra l'estructura de nodes interrelacionats d'una BD en xarxa.



**FIGURA 2.3.** Estructura d'una BD en xarxa

El model en xarxa va comportar una millora respecte al model jeràrquic, perquè permetia controlar de manera més eficient el problema de la redundància de dades.

Malgrat aquests avantatges, el model en xarxa no ha tingut tanta fortuna com el seu predecessor, a causa de la complexitat que comporta l'administració de les BD que l'adopten.

El consorci de la indústria de les tecnologies de la informació CODASYL (acrònim de *Conference on Data Systems Languages*) va proposar un estàndard que van seguir la majoria de fabricants.

Un dels gestors de BD en xarxa més coneguts, i que segueix l'estàndard CODASYL, és l>IDMS (*integrated database management system*), de Computer Associates.

### 2.2.3 Model relacional

El **model relacional** es basa en la lògica de predicats i en la teoria de conjunts (àrees de la lògica i de les matemàtiques). Actualment, és el sistema més àmpliament utilitzat per modelitzar dades.

A partir dels anys vuitanta, es van començar a comercialitzar gran quantitat de BD que aplicaven aquest model.

Les dades s'estructuren en representacions tabulars, anomenades **taules**, que representen entitats tipus del món conceptual, i que estan formades per files i columnes. Les columnes formen els **camp**s, que implementen els atributs, és a dir, les característiques que ens interessen de les entitats. I les files són els **registres**, que implementen les entitats instància, constituïdes pels conjunts dels valors que presenten els camps corresponents a cada instància.

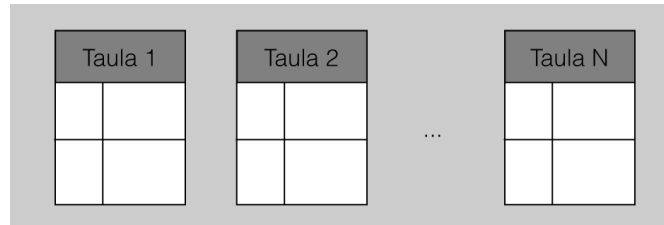
#### Model relacional

Va ser proposat formalment per E. F. Codd, l'any 1970, en el seu treball *A Relational Model of Data for Large Shared Data Banks* ('Un model relacional de dades per a grans bancs de dades compartides').

En els models de dades jeràrquic i en xarxa les dades s'estructuraven gràcies a dos elements: els registres i les interrelacions. Però el model relacional només consta d'un element: les **relacions** o **taules**.

La figura 2.4 mostra l'estructura de taules corresponent a una BD relacional.

**FIGURA 2.4.** Estructura d'una BD relacional



Les interrelacions s'han d'implementar utilitzant les taules: quan és necessari s'han d'afegir, a les taules, un o més camps que actuïn com a el que anomenarem clau forana i que, per tant, "apunten" al camp o camps referenciats d'una altra taula, els quals han de formar la seva clau primària. En coincidir els valors dels camps de la clau primària i de la clau forana, s'estableix la interrelació entre els registres.

El model relacional comporta certs avantatges respecte al model jeràrquic i al model en xarxa:

- Proporciona eines per evitar la duplicitat de registres, mitjançant claus primàries i foranes que permeten interrelacionar les taules.
- Vetlla per la integritat referencial: en eliminar-se un registre o en modificar-se el seu valor, o bé no permet fer-ho si hi ha registres interrelacionats en altres taules, o bé s'esborren o es modifiquen en cascada els registres interrelacionats, en funció de quina orientació hàgim seguit en administrar la BD.
- En no tenir importància la ubicació física de les dades, afavoreix la comprensibilitat. De fet, el model relacional, com a tal, es limita al nivell lògic, i deixa de banda el nivell físic. Per aquest motiu, es diu que el model relacional possibilita la independència física de les dades.

## 2.2.4 El paradigma de l'orientació a objectes

El **model de dades relacionals amb objectes** és una extensió del model relacional en sentit estricte.

### TAD

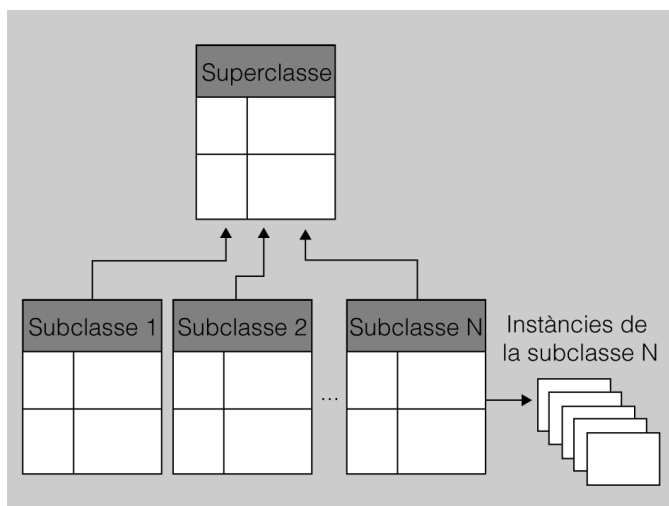
Un tipus abstracte de dades (TAD) és un concepte que defineix les dades juntament amb les seves operacions associades.

Aquest nou model admet la possibilitat que els tipus de dades siguin, a més dels tradicionals, **tipus abstractes de dades (TAD)**. Amb aquesta particularitat, s'acosten els sistemes de BD relacionals al paradigma de la programació orientada a objectes.

Els models estrictament orientats a objectes defineixen les BD en termes d'objectes, de les seves propietats i, el que és més innovador, de les seves operacions. Els objectes amb una mateixa estructura i comportament pertanyen a una classe, i les classes s'organitzen en jerarquies. Les operacions de cada classe s'especifiquen en termes de procediments predefinits, anomenats *mètodes*.

La figura 2.5 mostra l'estructura de classes, subclasses i instàncies, d'una BD orientada a objectes.

**FIGURA 2.5.** Estructura d'una BD orientada a objectes



#### Oracle

Alguns SGBD presents en el mercat des de fa molt de temps, van estenent el model relacional per tal d'incorporar conceptes relatius a l'orientació a objectes. Aquest és el cas de la coneguda firma Oracle, la qual està treballant en aquesta línia des de la versió 8 del seu producte.

## Nous models de bases de dades

Actualment, hi ha organitzacions que gestionen grans quantitats de dades i que aquestes dades les han de consultar habitualment i els cal que aquestes consultes siguin ràpides. Típicament aquesta necessitat es va crear en empreses que necessitaven consultar dades per a la presa de decisions.

Es pot pensar, per exemple, en una multinacional que disposa de BD relacionals on emmagatzema totes les seves factures, totes les línies de factures, que s'han anat fent al llarg de 15 anys d'història. Disposar d'un resum global de l'evolució de la facturació dels darrers 10 anys, pot suposar, en aquesta BD relacional la consulta de milions de registres. Es pot intuir, doncs, que resultarà un procés costós obtenir aquesta informació.

El model relacional, doncs, moltes vegades no dóna una resposta prou eficient per a gestionar aquests tipus d'informació i per això van començar a aparèixer els sistemes Datawarehouse. Els Datawarehouse són magatzems de dades que integren eines per a extreure transformar i carregar informació anomenada d'intel·ligència empresarial així com informació de metadades.

Més enllà dels Datawarehouse existeixen les BD multidimensionals.

Les **BD multidimensionals** són BD dissenyades per a optimitzar el processament analític en línia, conegut com OLAP (*On-Line Analytical Processing*). La característica més destacable del processament OLAP és l'estructuració de les dades en els anomenats cubs OLAP (OLAP-cube, és el terme anglès amb el que es coneix aquest concepte).

Un cub OLAP és una estructura de dades que permet accessos ràpids a la informació i que s'organitza aquesta informació en diverses perspectives o dimensions.

#### Exemple de cub OLAP

Un exemple típic de cub OLAP es pot donar en una empresa que requereixi analitzar informació financera des de diferents perspectives:

- Per productes
- Per períodes de temps
- Per poblacions
- Per comparació amb el pressupost
- Etc.

Disposar de forma eficient de la informació organitzada des de totes aquestes perspectives es pot veure com a un hipercub d'informació, on cada dimensió del cub (que pot ser de dues, tres, quatre, etc. dimensions) correspon a una forma (perspectiva) d'accedir a aquestes dades.

Un cas particular de BD multidimensional són les **BD multivalor**. Les BD multivalor solen dissenyar BD on els atributs emmagatzemen llistes de valors, a diferència de les BD relacionals on els atributs són monovalor. Moltes vegades es coneix a les BD multivalor com a BD post-relacionals.

Les BD multivalor proporcionen llenguatges d'accés a les dades molt més semblants al llenguatge natural i, per tant, més senzills que SQL. La dificultat actual resideix en què cada implementació de BD multivalor disposa del seu propi llenguatge, no existint, en l'actualitat un estàndard comú.

## 2.2.5 Modelització de dades amb l'UML

Últimament, i per influència de les tècniques de desenvolupament de programari orientades a l'objecte, han aparegut alguns models semàntics com a noves extensions del model ER originari, entre les que destaca especialment l'UML (*unified modeling language*).

La notació del llenguatge UML és diagramàtica, com la notació del model ER. Els diagrames UML es poden utilitzar per expressar els dissenys conceptuals, tal com es fa amb els diagrames ER.

Però cal tenir en compte que el disseny de BD s'encarrega, fonamentalment, de la part estàtica dels sistemes d'informació (és a dir, la que no hauria de canviar al llarg del temps, com ara, justament, l'estructuració de les dades).

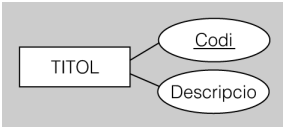

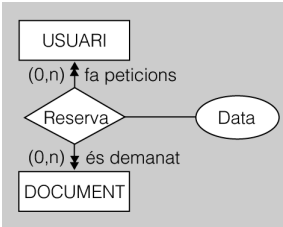

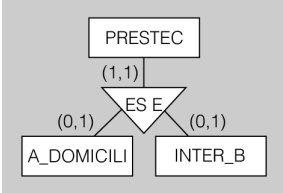
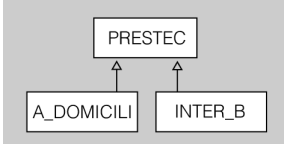
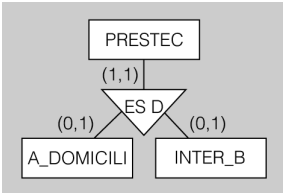
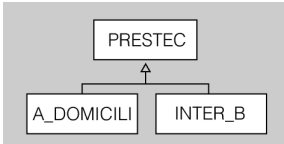
En l'UML, la part estàtica dels sistemes es representa mitjançant els anomenats **diagrames de classes** i, concretament, amb els components següents:

- Les classes i els atributs respectius (però no necessàriament, les operacions).
- Les relacions entre classes, com les associacions i les generalitzacions (però no tant, les dependències ni les realitzacions).

Per tant, aquí no considerarem ara altres aspectes de l'UML que s'ocupen de modelitzar més aviat la part dinàmica dels sistemes d'informació (és a dir, els aspectes que canvien amb el temps).

En la taula 2.1, podem veure les equivalències més importants entre la notació emprada en els diagrames ER, i l'emprada en els diagrames de classes UML.

**TAULA 2.1.** Taula d'equivalències entre els diagrames ER i els diagrames de classe UML

	Diagrama ER	Diagrama de classe UML
<b>Entitats: atributs</b>		
<b>Interrelacions: cardinalitats, rols i atributs</b>		
<b>Generalitzacions encavalcades</b>		
<b>Generalitzacions disjunts</b>		

### Classe

Una classe en orientació a objectes és una abstracció que agrupa un conjunt d'instàncies d'objectes. Per exemple, podem tenir la classe Cotxe que defineixi les característiques dels diferents objectes cotxe (instàncies).

## Entitats i atributs

L'UML considera les entitats tipus com a classes, i les especifica amb requadres dividits verticalment en tres seccions:

- La superior, en què es col·loca el nom de la classe.
- La intermèdia, en què apareixen els atributs respectius.
- La inferior, en què han de constar els noms de les operacions de la classe (no hem mostrat aquest apartat perquè ara mateix només ens interessen els aspectes estàtics de les dades).

Les classes permeten especificar molts altres detalls, com ara el tipus de dada de cada atribut, la seva visibilitat, etc.

## Interrelacions: cardinalitats, rols i atributs

L'UML considera les interrelacions com a associacions entre classes. Les interrelacions binàries es representen en els diagrames de classe UML simplement amb una línia contínua que connecta les dues classes implicades.

Aquesta línia pot ser dirigida, és a dir, que pot acabar amb una punta de fletxa (no tancada) en un dels extrems (o en tots dos).

A més, les associacions, poden incloure una etiqueta amb el nom que tinguin assignat o, fins i tot, dues, si preferim especificar el rol que adopta cadascuna de les dues classes.

Si, en un diagrama ER, una interrelació té atributs propis, el diagrama de classes UML equivalent haurà d'incorporar un requadre addicional, dividit verticalment en dues seccions. En la secció superior haurà de constar el nom de l'associació, i en la inferior s'hauran d'incloure els atributs. Finalment, aquest requadre haurà d'anar unit amb una línia discontinua amb l'associació.

Les restriccions de cardinalitat s'especifiquen en els diagrames de classe UML de manera molt semblant a com es fa en els diagrames ER. També s'utilitza la forma  $m\grave{a}x \dots m\grave{i}n$  que ja coneixem, per tal d'indicar el màxim i el mínim d'associacions en què pot participar cada instància de la classe. Ara bé, habitualment, la  $n$  es representa amb un asterisc (\*), i la ubicació de les etiquetes que indiquen les restriccions de cardinalitat és exactament la inversa de la que correspon en un diagrama ER.

Cal parar atenció en el fet que les interrelacions n-àries d'ordre superior a 2 no es poden representar directament en els diagrames de classe UML.

## Generalitzacions encavalcades i disjuntives

Els **diagrames de classe UML** poden representar explícitament generalitzacions i especialitzacions, i considerar-les genèricament a totes dues com a generalitzacions entre una superclasse i unes quantes subclasses.

Les generalitzacions es representen mitjançant línies contínues que parteixen de les subclasses i acaben en una punta de fletxa buida que apunta a la superclasse.

Si es tracta d'una generalització encavalcada (és a dir, quan les instàncies de la superclasse també poden ser instàncies, simultàniament, de més d'una subclasse), cal establir una línia i una punta de fletxa pròpia per a cada subclasse.

En canvi, quan es vol representar una generalització disjunta (és a dir, quan les instàncies de la superclasse només poden ser, també, instàncies d'una sola de les subclasses), les línies que parteixen de les subclasses han d'acabar confluint en una única punta de fletxa buida que apunti a la superclasse.

## 2.3 Bases de dades distribuïdes

Un dels sectors informàtics on més s'està evolucionant darrerament, tot integrant el desenvolupament tecnològic amb la innovació metodològica, és el relatiu als sistemes distribuïts d'informació.

Amb aquesta darrera expressió volem fer referència a la utilització de dades emmagatzemades en diferents ubicacions, de vegades molt distants entre si, però que al mateix temps estan connectades, mitjançant una xarxa de comunicacions.

Aquesta tendència es fa palesa en l'ús habitual d'Internet per part de qualsevol de nosaltres, però també hi ha nombroses empreses i institucions que necessiten que els sistemes informàtics (i per tant, també les BD) s'adaptin cada cop més a la seva estructura geogràfica o funcional.

Un cas concret d'aquests sistemes el constitueixen les bases de dades distribuïdes. És important conèixer les diferents arquitectures aplicades als sistemes de bases de dades, en general. Podem distingir entre les arquitectures centralitzades (incloent-hi els sistemes client-servidor) i les descentralitzades, en què s'ha de tenir en compte el funcionament dels sistemes paral·lels.

També és important conèixer les diferents metodologies per distribuir BD, en funció dels objectius plantejats en la fase de disseny, i també de si l'estratègia emprada té un caràcter ascendent o descendent. S'han de tenir en compte, però, les dues conseqüències més problemàtiques de la distribució de BD: la duplicació i la fragmentació de les dades, on hem de diferenciar entre fragmentacions horitzontals, verticals i mixtes.

### BD

*BD* és l'acrònim de *base de dades*, entesa com a conjunt estructurat de dades emmagatzemades que permeten obtenir informació.

No hem d'oblidar tampoc les problemàtiques específiques que representen per a les BD distribuïdes tant les transaccions com la concurrència, ni els diferents protocols amb els quals es dona resposta a aquestes eventualitats.

### 2.3.1 Arquitectures de sistemes de bases de dades: centralitzades, descentralitzades, client-servidor

L'arquitectura de tot sistema de BD està molt condicionada per les característiques del sistema informàtic sobre el qual s'executa, i en especial pels aspectes següents:

- La connexió en xarxa de diferents computadores.
- El processament paral·lel de consultes dins d'una mateixa computadora.
- La distribució de les dades en diferents computadores, fins i tot allunyades entre si.

Aquestes innovacions tecnològiques han permès, respectivament, el desenvolupament, a partir dels inicials sistemes totalment centralitzats en una sola computadora, de diferents arquitectures de sistemes de BD més evolucionades, i que permeten donar resposta a una gran varietat de necessitats dels usuaris i de les organitzacions en què aquests treballen, com ara:

- Sistemes client-servidor.
- Sistemes paral·lels.
- Sistemes distribuïts.

#### Arquitectures centralitzades i client-servidor

Inicialment els sistemes de BD eren de tipus estrictament centralitzat, en el sentit que s'executaven sobre un únic sistema informàtic, sense necessitat d'interaccionar amb cap altre.

Però l'abaratiment dels ordinadors personals i el desenvolupament vertiginós de les seves capacitats, juntament amb la implantació de les xarxes i d'Internet, ha fet evolucionar els sistemes centralitzats cap a arquitectures de tipus client-servidor.

#### Sistemes centralitzats en una sola computadora

##### Concurrència

Es parla de concurrència quan diversos processos s'executen paral·lelament, i, en aquest cas, fan ús de les mateixes dades.

En parlar de sistemes de BD centralitzats, es fa referència tant als petits sistemes monousuaris que s'executen en un únic ordinador personal, com als grans sistemes multiusuaris d'alt rendiment.



En els sistemes monousuaris, els sistemes de BD centralitzats són petits sistemes de BD pensats per a les tasques que pugui fer un sol usuari amb una estació de treball. Aquests sistemes no sempre ofereixen totes les possibilitats que sempre ha de garantir qualsevol sistema de BD multiusuari, per modest que sigui, com per exemple el control automàtic de la concurrència.

En els sistemes multiusuaris, en canvi, es tracta de grans sistemes que poden donar servei a un gran nombre d'usuaris. Aquests només disposen per interaccionar amb el sistema de terminals sense capacitat pròpia per emmagatzemar dades ni tampoc per processar consultes, ja que de la realització d'aquestes tasques s'encarrega l'únic sistema centralitzat existent.

## Sistemes client-servidor

Els sistemes client-servidor tenen les seves funcionalitats repartides entre el sistema servidor central i múltiples sistemes clients que li envien peticions.

### ODBC i JDBC

*ODBC* i *JDBC* són els acrònims d'*open database connectivity* i *Java database connectivity*, respectivament. Es tracta de dos dels protocols més utilitzats per a la interconnexió de les aplicacions de BD.

Gradualment, els antics terminals dels sistemes centralitzats han estat substituïts per ordinadors personals, igualment connectats als subsistents sistemes centrals. Com a conseqüència d'això, pràcticament tots els sistemes centralitzats actuen avui en dia com a sistemes servidors que satisfan les peticions que els envien els respectius sistemes clients.

Actualment, els estàndards ODBC i JDBC permeten que tot client que utilitzi qualsevol dels dos, es pugui connectar a qualsevol servidor que proporcioni la interfície respectiva.

Podem distingir dues tipologies de sistemes servidors:

- **Servidors de dades.** S'utilitzen en xarxes d'àrea local en què s'arriba a una alta velocitat de connexió entre els clients i el servidor, sempre que les estacions de treball siguin comparables al servidor quant a la capacitat de processament. En entorns així definits, pot tenir sentit enviar les dades als clients, fer allà totes les tasques de processament d'aquestes dades, i finalment reenviar, si cal, els resultats al servidor.
- **Servidors de consultes.** Proporcionen una interfície, mitjançant la qual els clients els envien peticions per tal que resolguin consultes, i els retornin els resultats obtinguts. Així doncs, les transaccions s'executen sobre el servidor, però les dades resultants es visualitzen en el client del qual provenia la petició, si escau.

---

Les architectures basades en servidors de dades s'han implantat especialment en les BD orientades a objectes.

---

---

Les architectures basades en servidors de consultes són les que tenen més implantació.

---

### Memòria principal vs. memòria persistent

Entenem per memòria principal la memòria volàtil, habitualment la RAM. Entenem per memòria persistent aquella que no és volàtil. Habitualment en forma de disc o altres dispositius d'emmagatzematge extern.

## Arquitectures descentralitzades

La descentralització de les arquitectures de BD pot consistir en el repartiment de la càrrega de feina entre diferents components físics del sistema (bàsicament pel que fa a processadors, memòria principal i memòria persistent) comunicats entre si mitjançant una xarxa d'interconnexió.

Però una arquitectura descentralitzada també pot consistir en la distribució de la mateixa BD en diferents computadores, seguint la metodologia més adient per assolir l'objectiu proposat.

## Sistemes paral·lels

L'objectiu principal dels sistemes paral·lels consisteix a augmentar la velocitat de processament i d'E/S mitjançant la utilització en paral·lel d'UCP, memòria i discos durs.

E/S i I/O són els acrònims d'entrada i sortida, i de l'original en anglès input/output.

UCP i CPU són els acrònims d'unitat central de procés, i de l'original en anglès central processing unit.

Els sistemes paral·lels són molt útils (o fins i tot són imprescindibles) en el treball quotidià amb BD molt grans (de l'ordre de terabytes), o que han de processar moltes transaccions (de l'ordre de milers per segon), ja que normalment els sistemes centralitzats i els sistemes client-servidor no tenen prou capacitat per donar resposta a aquest tipus de necessitats.

Avui en dia molts ordinadors de gamma alta ofereixen un cert grau de paral·lelisme, atès que incorporen dos o quatre processadors. Però hi ha computadores paral·leles que suporten centenars de processadors i discos durs.

Ara bé, una de les característiques que permet avaluar la utilitat d'un sistema paral·lel de BD és la seva ampliabletat, la qual ha de garantir el funcionament ulterior del sistema a una velocitat acceptable, encara que creixi la grandària de la BD o el nombre de transaccions.

Però la veritable ampliabletat dels sistemes paral·lels ve donada per la comunicació dels seus components mitjançant alguna xarxa que faci possible connectar-los entre si. Les tres tipologies de xarxa més utilitzades són les següents:

- **Bus:** es pot tractar d'una xarxa *ethernet* o una interconnexió paral·lela. En tot cas, aquesta estructura només és apta per al treball amb un petit nombre de processadors.
- **Malla:** cada component està connectat amb tots els nodes adjacents. (Si la malla és bidimensional els nodes adjacents seran 4, i si és tridimensional, 6).
- **Hipercub:** s'assigna a cada component un nombre binari exclusiu, de tal manera que dos components han de tenir una connexió directa entre si sempre que els nombres binaris respectius només difereixin en un sol

En una malla, un component pot arribar a estar a  $2(n - 1)$  nodes de distància d'altres components.

bit. Així doncs, cadascun dels  $n$  elements del sistema estarà directament connectat amb uns altres  $\log(n)$  components.

D'altra banda, hi ha diferents models d'arquitectures paral·leles de BD. Alguns dels models més importants són:

- **Memòria compartida.** Tots els processadors comparteixen una memòria comuna, habitualment mitjançant un bus. Les arquitectures de memòria compartida han de dotar cada processador de molta memòria cau, per tal d'evitar els accessos a la memòria compartida sempre que això sigui possible. Ara bé, el límit raonable de processadors treballant en paral·lel ve donat, justament, pel cost que representa el manteniment de la coherència de la memòria cau.
- **Discos compartits.** Tots els processadors comparteixen un conjunt de discos comú, mitjançant una xarxa d'interconnexió. Aquest model permet el treball en paral·lel d'un nombre de processadors més gran que amb el model de memòria compartida, però com a contrapartida la comunicació entre ells és més lenta, ja que utilitzen una xarxa en lloc d'un bus.
- **Sense compartició.** Els processadors no comparteixen ni memòria ni discos. Aquest model té unes potencialitats d'ampliació encara més grans que el de compartició de discos, però en canvi els costos de comunicació són superiors als del model esmentat.
- **Jeràrquic.** Combinació de les característiques dels models anteriors. Aquesta arquitectura s'estructura en diferents nivells. Al nivell més alt, els nodes, connectats mitjançant una xarxa d'interconnexió, no comparteixen ni memòria ni discos. Cadascun d'aquests nodes pot ser, internament, un sistema de memòria compartida entre diferents processadors. O bé cadascun d'aquests nodes pot ser, internament, un sistema de discos compartits que inclogui, a un tercer nivell, un sistema de memòria compartida.

#### Retard de les comunicacions en un hipercub

En un hipercub, un component pot estar, com a màxim, a  $\log(n)$  nodes de distància d'altres components. El retard de les comunicacions en un hipercub és menor que en una malla.

## Sistemes distribuïts

Una **BD distribuïda** està formada per un conjunt de BD parcialment independents, emmagatzemades en diferents computadores, que comparteixen un esquema comú i que coordinen el processament de les transaccions que accedeixen a dades remotes.

Els processadors de les diferents computadores que formen un sistema distribuït es comuniquen entre si mitjançant una xarxa de comunicacions, però no comparteixen ni memòria ni discos. Cadascuna d'aquestes computadores constitueix, per tant, un **node del sistema distribuït**.

Normalment, els nodes de les BD distribuïdes es troben en llocs geogràficament distants, s'administren parcialment de manera independent i tenen una interconnexió força lenta entre ells.

#### SGBD

*SGBD* és l'acrònim de *sistema gestor de bases de dades*. Es tracta d'un programari especialitzat en la gestió i l'emmagatzematge de BD.

Normalment, tot SGBD actual és capaç de treballar amb un altre d'idèntic. Però això no sempre és tan fàcil entre SGBD de diferents fabricants. En funció d'aquesta eventualitat, es distingeix entre dos tipus de sistemes distribuïts de BD:

- **Sistemes homogenis:** són sistemes fortament acoblats, en què tots els nodes utilitzen el mateix SGBD o, en el pitjor dels casos, diferents SGBD del mateix fabricant.
- **Sistemes heterogenis:** els SGBD que utilitzen els nodes són diferents, i, per tant, solen ser més difícils d'acoblar.

---

L'acoblament és el grau d'interacció i dependència que tenen dues parts d'un sistema.

---

En tot cas, els usuaris dels sistemes distribuïts de BD no han de conèixer els detalls d'emmagatzemament de les dades que utilitzi, com ara la seva ubicació concreta o la seva organització. D'aquesta característica se'n diu **transparència**. Evidentment, els administradors de la BD sí que hauran de ser conscients d'aquests aspectes.

### Avantatges i inconvenients de la distribució de BD

Hi ha bones raons per implementar BD distribuïdes, com ara la compartició de la informació, la disponibilitat de les dades o l'agilització del processament d'algunes consultes:

- **Compartició de la informació i autonomia local.** Un avantatge de compartir les dades mitjançant la distribució consisteix en el fet que des de cada node es pot controlar, fins a cert punt (de fet, fins allà on permeti l'administrador global de la BD), l'administració de les dades emmagatzemades localment. Aquesta característica es coneix com a *autonomia local*.
- **Fiabilitat i disponibilitat.** Si es produeix una fallada en algun node d'un sistema distribuït o en les comunicacions amb aquest, és possible que els altres nodes puguin continuar treballant, si les dades que conté el node caigut o incomunicat estan repetides en altres nodes del sistema.
- **Agilització del processament de consultes.** Quan una consulta necessita accedir a dades emmagatzemades en diferents nodes, pot ser possible dividir la consulta en diferents subconsultes que s'executin en els nodes respectius.

Però l'ús de BD distribuïdes també té els seus punts febles, com per exemple l'increment dels costos en desenvolupament de programari i l'augment de possibilitat d'errors, i el temps addicional a afegir al temps de processament:

- **Increment en els costos de desenvolupament de programari.** És més difícil estructurar un sistema distribuït de BD, i també són més complicades les aplicacions que han de treballar amb aquest sistema, que no pas si es tracta d'un sistema de BD centralitzat.
- **Més possibilitat d'errors.** Com que els diferents nodes del sistema distribuït operen en paral·lel, és més difícil garantir la correcció dels algorismes.

- **Temps extra que cal afegir al temps de processament.** L'intercanvi de missatges i els càlculs necessaris per garantir la integritat de les dades distribuïdes entre tots els nodes comporten un afegit extra de temps, inexistent en els sistemes centralitzats.

### 2.3.2 Disseny de bases de dades distribuïdes. Estratègies. Metodologies

El disseny de la distribució d'una BD implica adoptar certes decisions. La primera té a veure amb la mateixa idoneïtat d'utilitzar una BD distribuïda i no pas una altra arquitectura. Aquesta decisió s'ha de fonamentar en el rendiment esperat de cadascuna de les arquitectures disponibles, aplicades al mateix conjunt de necessitats.

A continuació, en el cas d'optar per una BD distribuïda, cal prendre altres decisions no menys importants sobre quina és la millor manera d'ubicar en els diferents nodes del sistema tant les dades com les aplicacions que hagin d'accedir a aquestes dades.

La ubicació de les aplicacions habitualment no comporta grans problemes. Depèn de les funcionalitats que aquestes han d'oferir i dels llocs des dels quals s'utilitzaran majoritàriament o exclusivament. Però també s'ha de tenir molt en compte si hauran de treballar amb un sistema homogeni o heterogeni, i els SGBD utilitzats en cada cas.

Però la distribució de les dades és més crítica i ha de perseguir la consecució de certs objectius: potenciació del processament local, distribució ideal de la càrrega de feina i reducció dels costos d'emmagatzemament. A més, cal seguir una estratègia general de disseny ascendent o descendent, tot i que tots dos enfocaments no són mútuament excloents i es poden emprar en un mateix projecte en diferents etapes d'aquest. Finalment, i com a conseqüència de tot això, s'ha d'adoptar una metodologia concreta de distribució de dades, és a dir, multiplicació, divisió, etc.

#### Objectius de la distribució

Hi ha un cert consens en alguns dels objectius bàsics que ha de perseguir tot sistema distribuït de BD, com ara:

- **Potenciació del processament local.** En un sistema distribuït hi ha dos tipus de transaccions: les locals i les globals. Les primeres únicament necessiten accedir al mateix node del qual parteix la petició. En canvi, les segones necessiten accedir a dades ubicades en altres nodes, la qual cosa comporta un cost addicional, ja que cal utilitzar la xarxa que els comunica. Si les dades són distribuïdes acostant-les a les aplicacions que més les utilitzen, es maximitza el processament local.

- **Distribució ideal de la càrrega de feina.** La distribució de les dades també ha de tenir en compte les característiques de les diferents computadores ubicades a cada node i els usos més adients per a cadascuna d'elles. D'aquesta manera es potencia el paral·lelisme en l'execució de les aplicacions. Ara bé, cal advertir que la persecució d'aquest objectiu pot afectar negativament la potenciació del processament local.
- **Reducció dels costos d'emmagatzemament.** La repetició de les dades en diferents nodes d'un sistema distribuït pot contribuir a la disponibilitat d'aquestes, ja que si es produeix una fallada en un dels nodes, es podrà continuar treballant amb les duplicacions existents en un altre. Això comporta, entre altres coses, un increment dels costos d'emmagatzematge que ha de ser tingut en compte, encara que últimament resulta irrellevant si es compara amb els costos derivats en matèria d'UCP, E/S i transmissions per la xarxa.

### Estratègies: dissenys ascendent i descendent

A l'hora de dissenyar una BD distribuïda podem optar, fonamentalment, per dues estratègies: disseny ascendent i disseny descendent.

El **disseny ascendent** és una estratègia que pot ser aplicada quan s'ha de dissenyar una nova BD a partir de petites BD preexistents que han de ser integrades en una de sola, però conservant en la mesura que es pugui la ubicació originària de les dades.

Bottom-up designa com s'anomena, en anglès, el disseny ascendent.

En el disseny ascendent de BD distribuïdes s'han de sintetitzar els esquemes lògics locals per arribar a construir l'esquema lògic global del sistema distribuït.

Els sistemes distribuïts resultants d'un disseny ascendent amb BD preexistents són amb certa freqüència heterogenis, llevat que el projecte prevengui la migració a un SGBD distribuït, comú a tots els nodes.

El **disseny descendent** de BD distribuïdes és l'estratègia més adient quan es tracta de dissenyar aplicacions i BD noves, o quan es pot prescindir de conservar les estructures de dades anteriors, en cas que n'hi hagi. Evidentment, en aquests casos en què el dissenyador té més capacitat decisòria, el més recomanable és optar per implantar un sistema homogeni.

Top-down designa com s'anomena, en anglès, el disseny descendent.

En el disseny descendent de BD distribuïdes s'ha de partir de l'anàlisi de requeriments inicials, per tal de definir en primer lloc el disseny conceptual i simultàniament el disseny de les vistes dels usuaris finals de la futura BD.

De fet, en l'àmbit de les BD distribuïdes, el disseny conceptual (que dona lloc a entitats i a interrelacions entre elles) es pot interpretar com una integració de les diferents vistes dels usuaris.

Posteriorment, el disseny lògic inclourà totes les decisions en matèria de distribució de les dades. En funció de la metodologia de distribució adoptada, les relacions s'ubicaran senceres en diferents nodes del sistema, o bé es dividiran en fragments per ser distribuïts entre els nodes d'aquesta manera.

Finalment, en els nodes en què es consideri oportú, es podrà fer el disseny físic, a nivell local.

## Metodologies de distribució

Hi ha diferents metodologies per orientar la distribució de les BD, cadascuna de les quals té els seus avantatges i els seus inconvenients:

- Multiplicació.
- Divisió.
- Distribució amb node principal.
- Distribució amb duplicacions en nodes seleccionats.

## Multiplicació

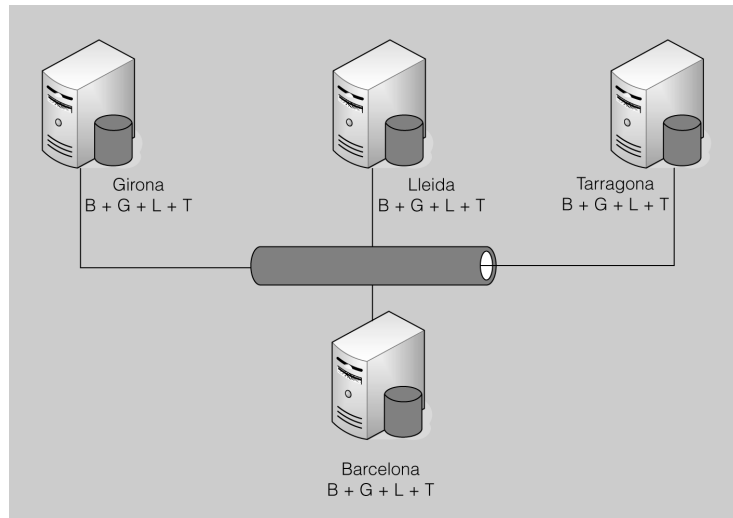
La multiplicació implica que la BD està replicada íntegrament a cada node del sistema. A la pràctica, aquest sistema és molt poc utilitzat.

L'avantatge principal de la multiplicació és evident, ja que les consultes es fan localment, sense haver d'accedir a la xarxa de comunicacions i, per tant, de manera molt ràpida. A més, en cas que caigui algun node, la resta pot continuar treballant.

Però no en falten de desavantatges, ja que les operacions d'actualització de dades s'han de fer en tots els nodes per tal de mantenir la coherència de la BD, la qual cosa implica un trànsit molt intens a la xarxa. I encara que actualment, en la majoria de casos, sigui un problema menor, cal dir que aquest model multiplica pel nombre total de nodes l'espai necessari per emmagatzemar la BD.

En la figura 2.6 es mostra un exemple de BD multiplicada, on cada node del sistema conté replicada completament la BD.

**FIGURA 2.6.** BD multiplicada, on es mostra la ubicació de les diferents parts (B, G, L i T) de les dades de la BD

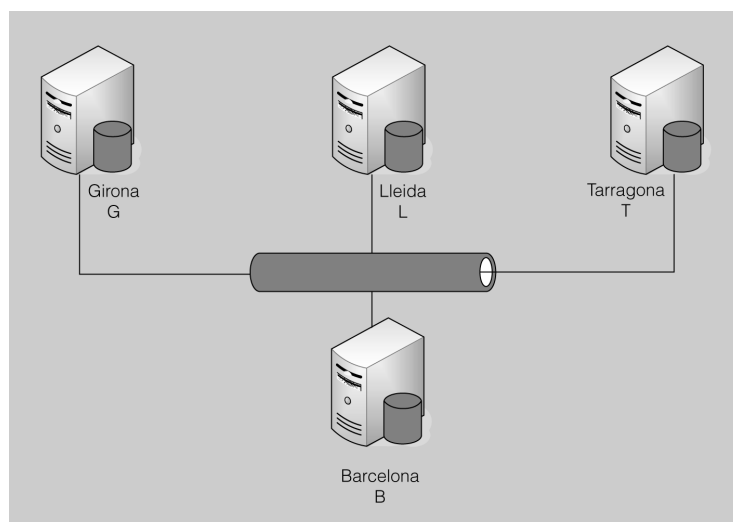


## Divisió

Amb el mètode de la divisió, la BD està distribuïda de tal manera que no hi ha cap part que estigui replicada en més d'un node.

L'avantatge principal de la divisió és que les operacions d'actualització són molt senzilles, ja que no es requereix actualitzar de manera transaccional les mateixes dades en diferents nodes. A més, tampoc no es necessita més espai d'emmagatzemament del que es necessitaria si es tractés d'una BD centralitzada.

**FIGURA 2.7.** BD dividida, on es mostra la ubicació de les diferents parts (B, G, L i T) de les dades de la BD



Com a contrapartides, podem dir que les operacions de consulta són sempre molt costoses, ja que la majoria són globals, la qual cosa comporta un trànsit molt intens a la xarxa. A més, la caiguda de qualsevol node implica la impossibilitat de poder accedir a aquella part de la BD emmagatzemada en ell.



En la figura 2.7 tenim un exemple de BD dividida, on cada node conté només les dades que li són pròpies, sense que estigui duplicada cap part de la BD en més d'un node.

### Distribució amb node principal

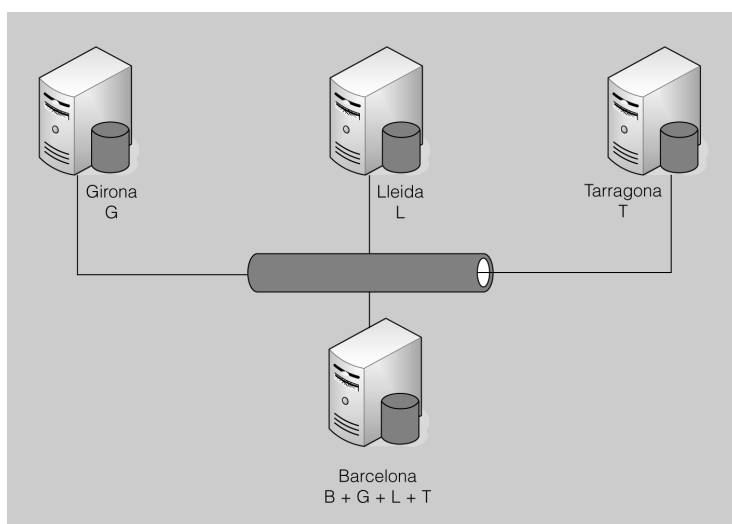
Quan s'utilitza el model de distribució amb node principal, un dels nodes, al qual es pot considerar principal, conté la BD sencera, i cadascun dels altres nodes conté replicada alguna part de la BD.

L'avantatge principal de la distribució amb node principal és que les dades s'acosten als nodes que més les utilitzen, la qual cosa potencia el processament local. A més, la disponibilitat és força bona, ja que totes les dades estan replicades com a mínim una vegada, pel fet d'estar emmagatzemades en algun dels nodes del sistema, a més d'estar-ho en el node principal.

Els desavantatges consisteixen fonamentalment en el fet que els costos de les operacions d'actualització i el d'emmagatzemament sempre seran més elevats (tot i que sense arribar als extrems de les BD multiplicades) que els produïts en sistemes centralitzats.

La figura 2.8 mostra un exemple de BD distribuïda amb un node principal (Barcelona) que conté tota la BD, mentre que la resta de nodes només contenen, duplicades, les dades que els són pròpies.

**FIGURA 2.8.** BD distribuïda, amb node principal on es mostra la ubicació de les diferents parts (B, G, L i T) de les dades de la BD



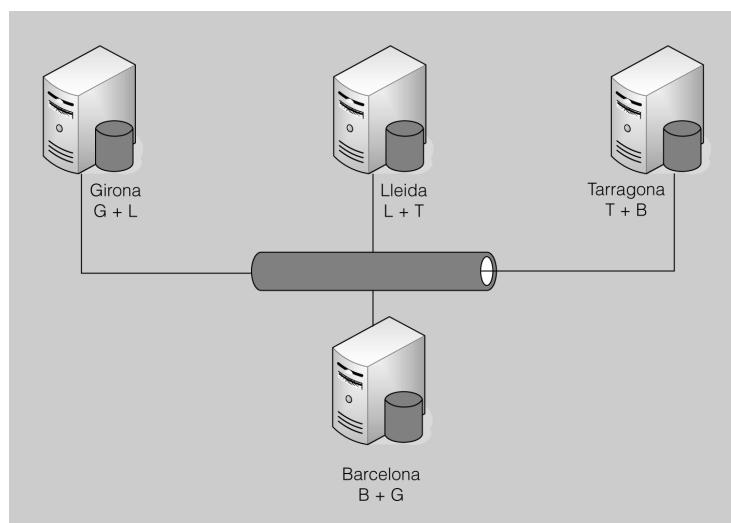
## Distribució amb duplicacions en nodes seleccionats

Seguint la metodologia de distribució amb duplicacions en nodes seleccionats, cap node conté la BD completa, però cada node replica alguna part de la BD, de tal manera que tota la BD, globalment considerada, està duplicada.

La distribució amb duplicacions en nodes seleccionats es tracta d'una solució amb uns avantatges i uns inconvenients similars als del model basat en la distribució amb node principal. L'avantatge respecte a aquell és que, com que no hi ha un node principal que contingui tota la BD, la disponibilitat és un xic més elevada.

La figura 2.9 proporciona un possible exemple de BD distribuïda amb duplicacions en nodes seleccionats. Cada node conté les dades que li són pròpies, i a més, conté replicada una altra part de la BD. No hi ha un node principal que contingui tota la BD, però la duplicació d'aquesta és completa si considerem les dades contingudes en tots els nodes.

**FIGURA 2.9.** BD distribuïda, amb duplicacions en nodes seleccionats on es mostra la ubicació de les diferents parts (B, G, L i T) de les dades de la BD



### 2.3.3 Conseqüències de la distribució de les dades: duplicació, fragmentació

Les BD distribuïdes emmagatzemen les relacions seguint principalment un dels dos esquemes referits a continuació:

#### Relacions en una BD

En el cas més habitual de BD relacionals, basades en el model relacional, una relació és la unitat lògica d'organització de les dades, que es correspon a una taula de BD.

- **Duplicació.** El sistema conserva còpies idèntiques (com a mínim una) de cada relació en diferents nodes.
- **Fragmentació.** Les relacions es divideixen en diferents fragments i s'emmagatzemen en diferents nodes. La fragmentació es realitza seguint alguna de les metodologies disponibles (horitzontal, vertical, etc.).

La duplicació i la fragmentació es poden utilitzar de manera combinada, fragmentant les relacions i distribuint còpies de cadascun dels fragments resultants entre els diferents nodes del sistema.

## Duplicació

La **duplicació** consisteix en l'emmagatzematge de relacions senceres, o de fragments d'aquestes, en diferents nodes de la xarxa.

La duplicació és un tipus d'esquema d'emmagatzematge de les BD distribuïdes que comporta tant avantatges com inconvenients.

D'una banda, la duplicació de les dades millora la seva disponibilitat respecte als sistemes centralitzats, ja que si falla un dels nodes que conté una relació o un fragment d'aquesta, es pot acudir (encara que només sigui temporalment) a un dels altres nodes que continguin les mateixes dades.

D'altra banda, la duplicació de les dades en diferents nodes incrementa el paral·lelisme, ja que fa augmentar les possibilitats de que les dades es trobin en el mateix node des del qual es llença la consulta.

Però, al mateix temps, la duplicació de dades implica un increment de la sobrecàrrega del sistema quan es produeixen actualitzacions de dades, ja que cal garantir la consistència de totes les rèpliques existents.

## Fragmentació

La **fragmentació** consisteix a dividir les relacions en diferents fragments. Aquests fragments han de contenir tota la informació necessària per tal de reconstruir les relacions originàries corresponents, en cas necessari.

El problema fonamental de la fragmentació inherent a les BD distribuïdes consisteix a trobar la unitat ideal de distribució de les dades. Normalment les relacions no són la millor opció de distribució per moltes raons.

D'una banda, les vistes que proporcionen les aplicacions habitualment són subconjunts de relacions. Per tant, pot ser molt més convenient considerar aquests subconjunts de relacions com les unitats desitjables de distribució.

Però, a més, la descomposició d'una relació en diferents fragments, allotjats en diferents nodes del sistema, pot contribuir a millorar el rendiment del sistema, ja que permet l'execució concurrent de transaccions, i provoca, en molts casos, l'execució paral·lela de les consultes, quan s'han de dividir en diferents subconsultes per tal d'operar sobre els diferents fragments.

## Fragmentació horitzontal

La **fragmentació horitzontal** consisteix a dividir els tuples d'una relació (és a dir, les files) en dos o més subconjunts en funció dels valors que aquelles tinguin en un o més atributs.

Aquests valors han de ser indicatius dels nodes que més consultes realitzaran sobre els respectius tuples, per tal d'acostar aquests als seus usuaris més habituals. Els tuples poden estar presents en més d'un fragment, però han d'estar com a mínim en un d'ells per tal que la fragmentació sigui correcta.

La taula 2.2 mostra una relació, anomenada PROVEIDOR, per tal d'exemplificar els mètodes principals de fragmentació.

TAULA 2.2. Relació PROVEIDOR

PROVEIDOR				
NIF*	Nom	Telefon	Adreça	Localitat
33333333K	L'abastadora, SL	902456456	Pol. Ind. Polièdric, s/n	Lleida
44444444L	Proveïdora Ibèrica, SA	906789789	C/ del pi, 3	Lleida
55555555M	Supplies & Co. Ltd.	900123123	C/ del call, 4	Girona
66666666N	Assortiments de l'Onyar, SCP	908852852	Pg. De la ribera, s/n	Girona

\*NIF és la clau principal de la taula proveïdor.

La taula 2.3 i taula 2.4 mostren dos possibles fragments horitzontals de la relació PROVEIDOR. Els tuples s'han dividit en funció de la localitat de cada proveïdor.

TAULA 2.3. Primer fragment horitzontal de la relació PROVEIDOR

PROVEIDOR				
NIF*	Nom	Telefon	Adreça	Localitat
33333333K	L'abastadora, SL	902456456	Pol. Ind. Polièdric, s/n	Lleida
44444444L	Proveïdora Ibèrica, SA	906789789	C/ del pi, 3	Lleida

\*NIF és la clau principal de la taula proveïdor.

TAULA 2.4. Segon fragment horitzontal de la relació PROVEIDOR

PROVEIDOR				
NIF*	Nom	Telefon	Adreça	Localitat
55555555M	Supplies & Co. Ltd.	900123123	C/ del call, 4	Girona
66666666N	Assortiments de l'Onyar, SCP	908852852	Pg. De la ribera, s/n	Girona

\*NIF és la clau principal de la taula proveïdor.

## Fragmentació vertical

La fragmentació vertical consisteix a dividir els atributs de la relació (és a dir, les columnes) en diferents fragments. Els fragments resultants han de contenir els atributs que utilitzaran més freqüentment els usuaris del node on seran respectivament emmagatzemats.

A més dels atributs seleccionats, cada fragment haurà de contenir la clau primària de la relació, per tal de poder associar els tuples de tots els fragments pertanyents a una mateixa relació, que estiguin allotjats en els diferents servidors del sistema.

Els atributs poden estar presents en més d'un fragment, però han d'estar com a mínim en un d'ells per tal que la fragmentació sigui correcta.

La taula 2.5 i taula 2.6 mostren dos possibles fragments verticals de la relació PROVEIDOR. Els fragments resulten de seleccionar només els atributs de cada proveïdor que utilitzaran amb més freqüència els nodes que respectivament els emmagatzemin.

**TAULA 2.5.** Primer fragment vertical de la relació PROVEIDOR

PROVEIDOR		
NIF*	Nom	Telefon
33333333K	L'abastadora, SL	902456456
44444444L	Proveïdora Ibèrica, SA	906789789
55555555M	Supplies & Co. Ltd.	900123123
66666666N	Assortiments de l'Onyar, SCP	908852852

\*NIF és la clau principal de la taula proveïdor.

**TAULA 2.6.** Segon fragment vertical de la relació PROVEIDOR

PROVEIDOR			
NIF*	Nom	Adreça	Localitat
33333333K	L'abastadora, SL	Pol. Ind. Polièdric, s/n	Lleida
44444444L	Proveïdora Ibèrica, SA	C/ del pi, 3	Lleida
55555555M	Supplies & Co. Ltd.	C/ del call, 4	Girona
66666666N	Assortiments de l'Onyar, SCP	Pg. De la ribera, s/n	Girona

\*NIF és la clau principal de la taula proveïdor.

## Fragmentacions mixtes

Les fragmentacions mixtes consisteixen a aplicar tant la fragmentació horitzontal com la vertical.

En funció de com es combinen les fragmentacions horitzontal i vertical, s'obtenen quatre tipologies mixtes:

1. **Fragmentació VH.** Es desenvolupa en primer lloc la fragmentació vertical, i a continuació l'horitzontal.
2. **Fragmentació HV.** S'aplica primer una divisió horitzontal i tot seguit es desenvolupa una altra de vertical sobre els fragments prèviament generats.
3. **Fragmentació semàntica.** La fragmentació de les relacions es fa alternant successivament fragmentacions horitzontals i verticals, però sempre tenint en compte el significat de les operacions més habituals que s'han de fer sobre les dades.
4. **Fragmentació simultània.** S'apliquen de manera simultània, i no pas seqüencial, la fragmentació horitzontal i la vertical, i la relació originària es transforma en una matriu, les cel·les de la qual són els fragments que s'han de distribuir. El nivell de fragmentació així obtingut normalment és molt elevat, la qual cosa no vol dir que sempre sigui més eficient.

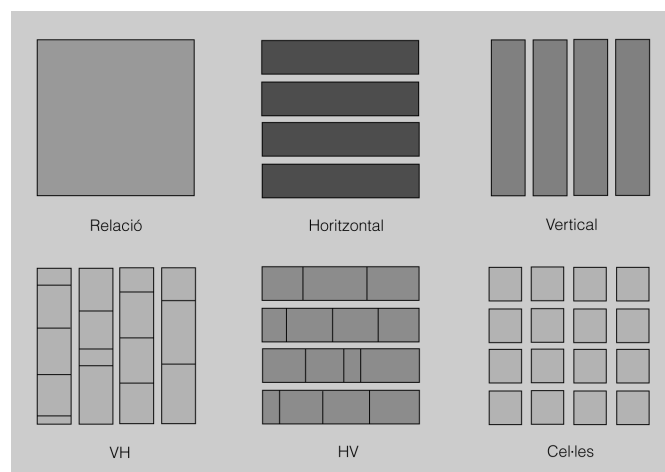
### Grau de fragmentació

En fragmentar una BD ha de ser valorat el grau de fragmentació que assolirà aquesta BD, ja que aquest paràmetre influirà notablement en el rendiment del sistema a l'hora d'executar consultes.

El grau de fragmentació serà igual a zero, en absència de fragmentació, és a dir, quan es prenguin les relacions com a unitats de fragmentació. I el grau serà màxim quan cada tuple (en la fragmentació horitzontal) o cada atribut (en la fragmentació vertical) de cada relació constitueixin un fragment. Davant d'aquest dos extrems, habitualment cal buscar solucions de compromís, tenint en compte la utilització que de la BD hagin de fer les aplicacions i els usuaris, des de tots els nodes de la xarxa.

La figura 2.10 mostra esquemàticament algunes possibilitats de fragmentació d'una relació.

**FIGURA 2.10.** Tipologies de fragmentació d'una relació



### 2.3.4 Transaccions i protocols de compromís

Els SGBD en general, i els distribuïts en particular, han de garantir la integritat de les dades, mitjançant el compliment d'aquestes característiques:

- **Atomicitat.** O bé es fan correctament en la BD totes les operacions incloses en la transacció, o bé no se'n fa cap.
- **Consistència.** L'execució aïllada de la transacció (és a dir, sense la concurrència de cap altra transacció) conserva la consistència de la BD.
- **Aïllament.** Davant la concurrència de transaccions, el sistema garanteix que l'execució de cadascuna d'elles pressuposa, o bé que l'execució de les altres encara no ha començat, o bé que ja ha finalitzat completament.
- **Durabilitat.** Després de la finalització amb èxit d'una transacció, els canvis produïts en la BD romanen, fins i tot, en cas de fallades del sistema.

---

ACID properties és l'acrònim que s'obté amb la primera lletra de les quatre propietats de les transaccions en anglès: *atomicity, consistency, isolation, durability*.

---

Les BD distribuïdes tenen els mateixos requisits transaccionals que les BD centralitzades. Però, evidentment, és més difícil garantir l'atomicitat i l'aïllament de les transaccions globals en un sistema distribuït que no pas les transaccions ocasionades en un altre de centralitzat.

Per tal de garantir la integritat de les dades en l'execució de transaccions, els SGBD distribuïts executen els anomenats protocols de compromís.

#### Compromís en dues fases

Els protocols de compromís serveixen per garantir que tots els nodes del sistema, implicats en l'execució d'una mateixa transacció, coincideixin en el resultat final obtingut.

El protocol de compromís més senzill, però que a la vegada és també un dels més utilitzats, s'executa en dues fases:

- **1a fase.** Un dels servidors dels nodes implicats en la transacció, que actua com a coordinador, pregunta als servidors dels altres nodes si estan en condicions de confirmar la transacció. Aleshores cadascun d'aquests servidors fa les comprovacions necessàries (per exemple, en matèria de restriccions d'integritat) per tal de donar una resposta. El servidor coordinador confirmarà la transacció si obté una resposta afirmativa per part de cadascun dels altres servidors implicats. En cas contrari, haurà de cancel·lar la transacció.
- **2a fase.** El servidor que actua com a coordinador envia un missatge a la resta de servidors implicats en la transacció comunicant-los si han de confirmar o cancel·lar la transacció.

Aquest protocol és molt sensible a la caiguda dels nodes implicats, a les fallades de la xarxa, o fins i tot, a la disminució de la seva velocitat, atès el nombre de missatges a intercanviar entre els nodes implicats en cada transacció.

Però el problema potencial més important d'aquest protocol és la possibilitat de bloqueig del sistema. Si el servidor que actua com a coordinador cau durant el procés, o s'interrompen les comunicacions amb ell, la transacció pot quedar activa en la resta de servidors, la qual cosa acabaria per produir un bloqueig del sistema, o d'alguns dels seus nodes, ja que aquests servidors no haurien de cancel·lar unilateralment l'operació, ja que no sabrien si la resta de servidors han confirmat ja els canvis i haurien d'esperar indefinidament fins al restabliment de les comunicacions amb el servidor coordinador de la transacció.

### **Compromís en tres fases**

Per a moltes aplicacions el problema latent del bloqueig durant l'execució del protocol de compromís en dues fases no és acceptable, per la qual cosa s'han anat desenvolupat protocols alternatius, que si bé comporten certs avantatges, tampoc no estan exempts de problemes.

El protocol de compromís en tres fases evita alguns inconvenients del protocol de compromís en dues fases, del qual es pot considerar una extensió. Però, al mateix temps, comporta uns increments de complexitat i de sobrecàrrega tals, que no és gaire utilitzat.

Aquest protocol pressuposa que no pot tenir lloc una fragmentació de la xarxa, i que mai no fallaran més d'un nombre predeterminat de nodes. Aleshores s'evita la possibilitat de bloqueig, afegint-hi una tercera fase addicional, en la qual s'impliquen uns quants nodes addicionals al que actua com a coordinador en la presa de decisió del compromís.