BASES 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.

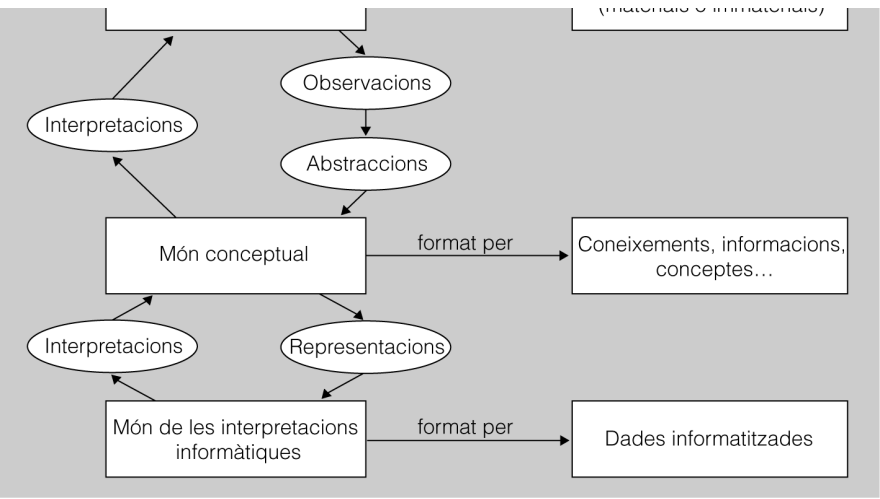
1.1. Les dades i les bases de dades

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

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 assoleixen.

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**.

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.

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.

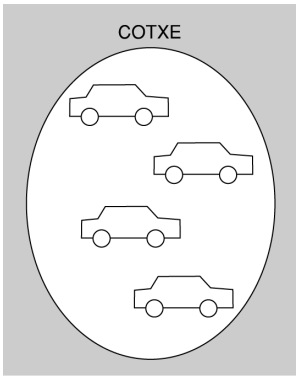
Si només coneixem dos d'aquests tres elements, no disposarem d'una veritable informació, ja que es produirà alguna de les mancances següents:

- 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.
- 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

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.



Entitats instància

> **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.

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, si 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.

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

dos atributs: un que designi l'alumne de què es tracti (duplicament, 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.

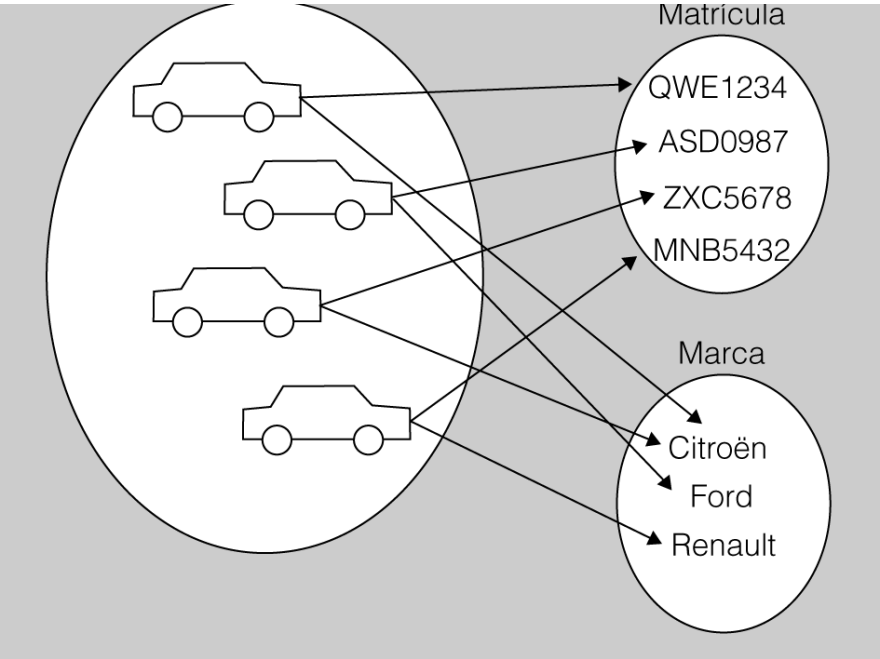
En el fons, les BD només són conjunts de fitxers interrelacionats (o, si es prefereix, que emmagatzemen dades que estan interrelacionades). Però no serveix de res emmagatzemar dades si, posteriorment, no hi accedim. Hi ha diferents tipus d'accés a les dades que convé examinar: seqüencial, directe, per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

BASES DE DADES



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 accepcions, 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

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.

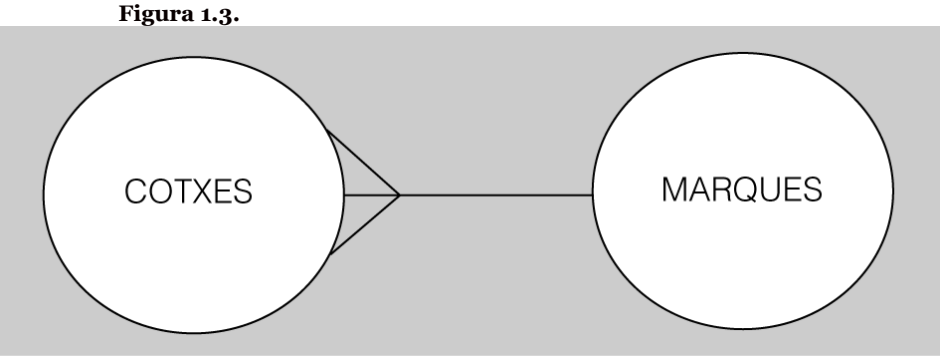
Les interrelacions entre els registres de dues (o més) taules es fan mitjançant camps del mateix tipus de dada que emmagatzemin els mateixos valors.

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.

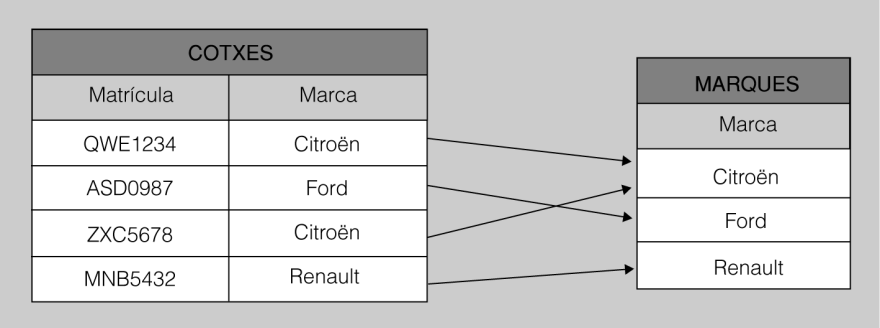
Figura 1.3.



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.

tani a les dades per part de diferents usuaris.

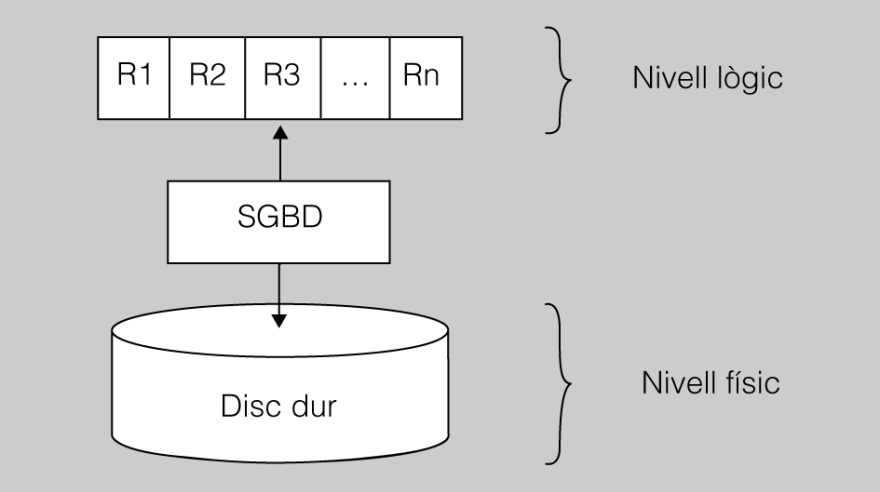
1.1.10. El nivell lògic i el nivell físic

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.

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,

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**
Entitats tipus	Les entitats instància d'un fitxer pertanyen a una sola entitat tipus.	Les BD contenen entitats instància d'infinitat d'entitats tipus interrelacionades.
Interrelacions	El sistema no interrelaciona fitxers.	El sistema té previstes eines per interrelacionar fitxers.

		dades i els riscos que comporta.
Inconsistències	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.
Obtenció de dades	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.
Aïllament de dades	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.
Integritat de dades	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.
Atomicitat	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.
Accés concurrent	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.
Seguretat	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

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 dóna 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.

(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.

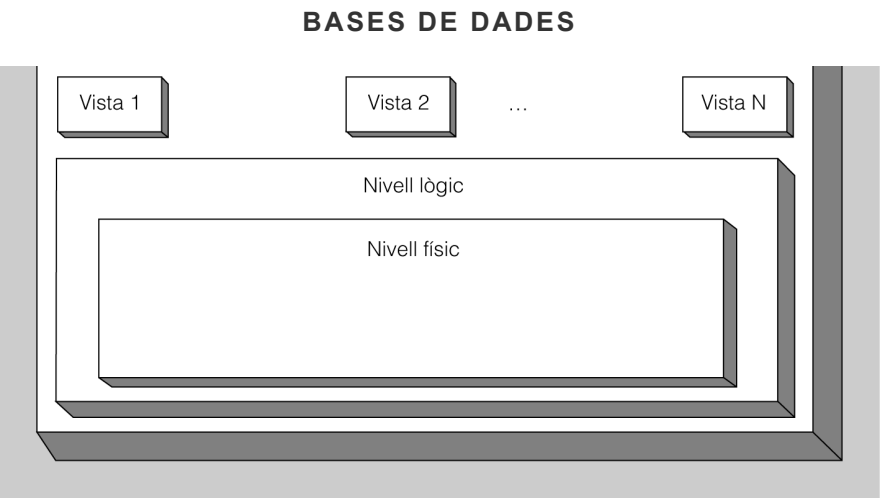
Perquè les BD resultin útils, han de ser mínimament eficients a l'hora de recuperar les dades. Per aquest motiu, els sistemes de BD tenen implementades, a baix nivell, estructures de dades bastant complexes.

> 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.

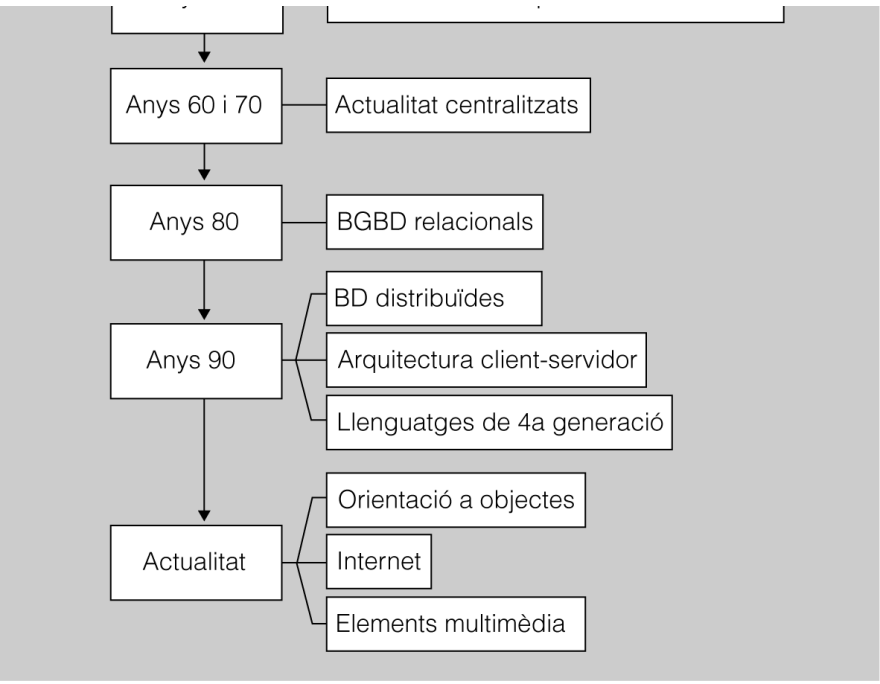
1.3.1. Evolució dels SGBD

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 contingués 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.

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 enunciaven 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, arquitectures 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:

han 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 resistents 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 diferent 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.

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.

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.

4GL

Aquests són alguns entorns actuals de desenvolupament que utilitzen llenguatges de quarta generació: eDeveloper, de Magic Software, Oracle Developer,

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.

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 estructuren 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).

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 abreujadament com a W3C). El W3C és un consorci internacional que produeix estàndards per a la World Wide Web (coneguda abreujadament 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.

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 algoritmes 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.

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.

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

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

tud 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).

- **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.

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.

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

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.

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

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'encriptació** 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'encriptació 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 encriptar 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.

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

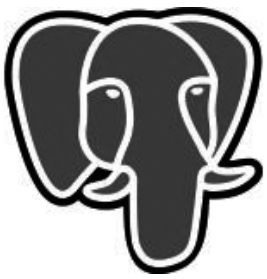
Com a instruccions de tipus DML, podem esmentar SELECT (per fer consultes), i també INSERT, UPDATE i DELETE (per realitzar el manteniment de les dades).

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)

- **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.



Logotip de 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:

guatges, 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.

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.

guatge 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.

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.

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 sistemes 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:

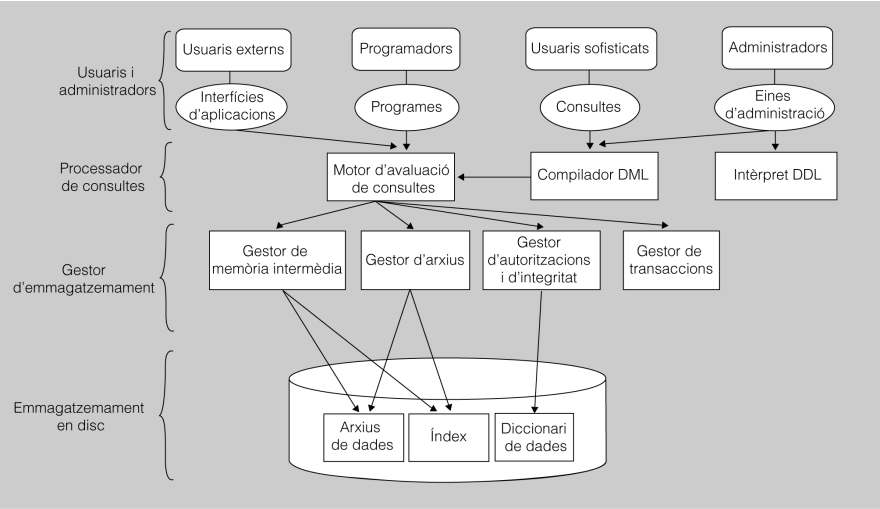
- **Intèrpret DDL**. Interpreta les instruccions de tipus DDL i registra les definicions en el diccionari de dades.

tal 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, orígen, 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.

- 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';
```

BASES DE DADES