Introducció a les bases de dades

Rafael Camps Paré

PID_00171643

Universitat Oberta de Catalunya

Índex

Introducció

Començarem aquest mòdul didàctic veient quins són els **objectius dels sistemes de gestió de les bases de dades (SGBD)** i, a continuació, donarem una visió general de l'**arquitectura**, el **funcionament** i l'**entorn** d'aquests sistemes.

Objectius

En els materials didàctics d'aquest mòdul trobareu les eines per a adquirir una visió global del món de les BD i dels SGBD, i assolir els objectius següents:

1. Conèixer a grans trets l'evolució dels SGBD des dels anys seixanta fins avui.

2. Distingir els principals objectius dels SGBD actuals i contrastar-los amb els sistemes tradicionals.

3. Saber explicar mitjançant exemples els problemes que intenta resoldre el concepte de *transacció*.

4. Relacionar la idea de flexibilitat en els canvis amb la independència lògica i física de les dades, i amb l'arquitectura de tres nivells.

5. Distingir els principals models de BD.

6. Conèixer a grans trets el funcionament d'un SGBD.

7. Saber relacionar els diferents tipus de llenguatges amb els diferents tipus d'usuaris.

1. Concepte i origen de les BD i dels SGBD

El concepte de *base de dades* (BD) es pot entendre com un conjunt de fitxers entre els quals hi ha lligams o interrelacions. Ara introduirem el concepte de BD des d'un punt de vista històric.

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

Aplicacions informàtiques dels anys seixanta

L'emissió de factures, el control de comandes pendents de servir, el manteniment del fitxer de productes, la nòmina del personal, eren algunes de les aplicacions informàtiques habituals als anys seixanta.

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

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

Integració d'aplicacions

Per exemple, s'integra l'aplicació de factures, la de comandes pendents i la gestió del fitxer de productes.

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

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

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

El **programari de gestió de fitxers** era massa elemental per a donar satisfacció a totes aquestes necessitats. Per exemple, el tractament de les interrelacions no hi estava previst; no era possible que diversos usuaris actualitzessin dades

simultàniament, etc. La utilització d'aquests conjunts de fitxers per part dels programes d'aplicació era excessivament complexa, de manera que, especialment durant la segona meitat dels anys setanta, van anar sortint al mercat programaris més sofisticats, els anomenats *Data Base Management Systems*, que aquí denominem **sistemes de gestió de BD (SGBD)**.

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

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

Els fitxers tradicionals i les BD

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

1) Entitats tipus:

- Fitxers: tenen registres d'una sola entitat tipus.
- BD: tenen dades de diverses entitats tipus.

2) Interrelacions:

- Fitxers: el sistema no interrelaciona fitxers.
- BD: el sistema té previstes eines per a interrelacionar entitats.

3) Redundància:

Fitxers: es creen fitxers a la mida de cada aplicació, amb totes les dades necessàries encara que algunes siguin redundants respecte d'altres fitxers.

BD: totes les aplicacions treballen amb la mateixa BD i la integració de les dades és bàsica, de manera que s'evita la redundància.

4) Usuaris:

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

2. Evolució dels SGBD

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

2.1. Els anys seixanta i setanta: sistemes centralitzats

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

Els **primers SGBD** –als anys seixanta encara no se'ls anomenava així– estaven orientats a facilitar la utilització de grans conjunts de dades en què les interrelacions eren complexes. L'arquetipus d'aplicació era el *Bill of materials* o *Parts explosion*, típica en les indústries de l'automòbil, en la construcció de naus espacials i en camps similars. Aquests sistemes treballaven exclusivament per lots (*batch*).

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

El *Data Base / Data Comunications*

IBM anomenava *Data Base / Data Comunications* (*DB/DC*) el programari de comunicacions i de gestió de transaccions i de dades. Les aplicacions típiques eren la reserva/compra de bitllets a les companyies aèries i de ferrocarrils i, una mica més tard, els comptes de clients en el món bancari.

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

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

2.2. Els anys vuitanta: SGBD relacionals

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

L'aparició dels SGBD relacionals[1] suposa un avenç important per a facilitar la programació d'aplicacions amb BD i per a aconseguir que els programes siguin independents dels aspectes físics de la BD.

(1)Oracle apareix l'any 1980.

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

Els ordinadors personals

Durant els anys vuitanta apareixen i s'estenen ràpidament els ordinadors personals. També surten programaris per a aquests equips monousuari (per exemple el dBase i els seus derivats), amb els quals és molt fàcil crear i utilitzar conjunts de dades, i que es denominen *personal data bases*. Noteu que el fet d'anomenar SGBD aquests primers sistemes per a PC és una mica forçat, ja que no acceptaven estructures complexes ni interrelacions, ni podien ser utilitzats en una xarxa que servís simultàniament a molts usuaris de tipus diferents. Però alguns, amb el temps, s'han anat convertint en autèntics SGBD.

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

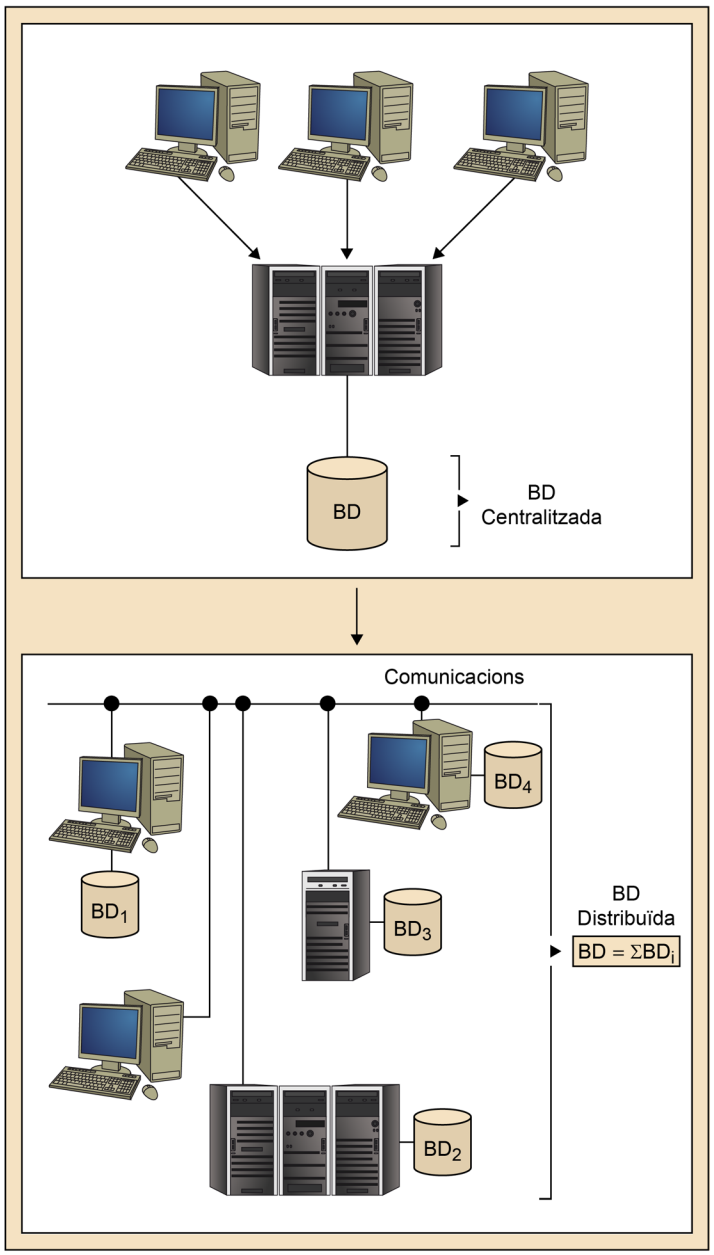
En acabar la dècada dels vuitanta, els SGBD relacionals ja eren utilitzats pràcticament a totes les empreses. Malgrat tot, fins a la meitat dels noranta, quan es necessitava un rendiment elevat se seguien utilitzant els SGBD prerelacionals.

Al final dels vuitanta i començament dels noranta, les empreses es trobaven que els seus departaments anaven comprant ordinadors departamentals i personals, i anaven fent aplicacions amb BD. Com a resultat, en el si de l'empresa hi havia nombroses BD i diversos SGBD de tipus o proveïdors diferents. Aquest fenomen de **multiplicació de les BD i dels SGBD** es va veure incrementat per la febre de les fusions d'empreses.

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

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

Figura 1



A més d'aquesta distribució "imposada", en voler tractar de manera integrada diverses BD preexistents, també es pot fer una distribució "volguda", dissenyant una BD distribuïda físicament, i amb certes parts reproduïdes en diferents sistemes. Les raons bàsiques per les quals interessa aquesta distribució són les se-güents:

1) Disponibilitat. La disponibilitat d'un sistema amb una BD distribuïda pot ser més alta, perquè si queda fora de servei un dels sistemes, seguiran funcionant els altres. Si les dades residents al sistema no disponible estan reproduïdes en un altre sistema, continuaran estant disponibles. En cas contrari, només estaran disponibles les dades dels altres sistemes.

2) Cost. Una BD distribuïda pot reduir el cost. En el cas d'un sistema centralitzat, tots els equips usuaris, que poden estar distribuïts per diverses àrees geogràfiques llunyanes, estan connectats per mitjà de línies de comunicació al sistema central. El cost total de les comunicacions es pot reduir fent que les dades que un usuari utilitza més sovint les tingui a prop; per exemple, en un ordinador de la seva pròpia oficina o, fins i tot, en el seu ordinador personal.

La tecnologia que es fa servir habitualment per a distribuir dades és la que es coneix com a **entorn (**o arquitectura) **client/servidor (C/S)**. Tots els SGBD relacionals del mercat hi han estat adaptats.

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

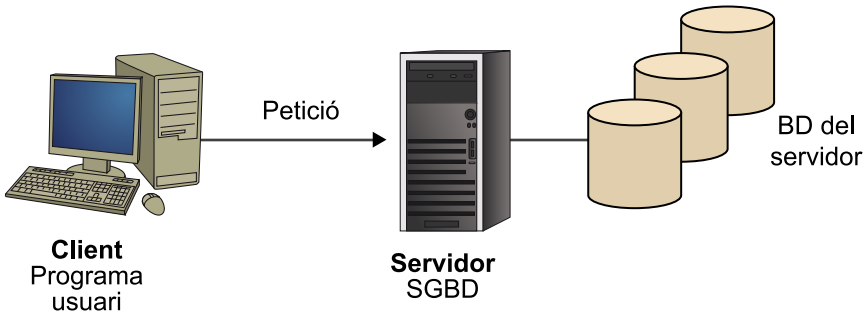
Per exemple, un programa d'aplicació que un usuari fa executar en el seu PC (que està connectat a una xarxa) demana certes dades d'una BD que resideix en un equip UNIX on s'executa l'SGBD relacional que la gestiona. El programa d'aplicació és el client i l'SGBD és el servidor.

Altres serveis

Noteu que el servei que dóna un servidor d'un sistema C/S no té per què estar relacionat amb les BD; pot ser un servei d'impressió, d'enviament d'un fax, etc., però aquí ens interessen els servidors que són SGBD.

Un procés client pot demanar serveis a diversos servidors. Un servidor pot rebre peticions de molts clients. En general, un procés *A* que fa de client demanant un servei a un altre procés *B* pot fer també de servidor d'un servei que li demani un al-tre procés *C* (o fins i tot el *B*, que en aquesta petició seria el client). Fins i tot el client i el servidor poden residir en un mateix sistema.

Figura 2



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

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

Exemple

C/S, SQL i 4GL són sigles de moda des del principi dels anys noranta en el món dels sistemes d'informació.

2.4. El nou mil·leni

Des que els sistemes de gestió de bases de dades relacionals van irrompre en el mercat, han estat els més utilitzats en la indústria, possiblement perquè s'acostaven a usuaris i aplicacions i s'allunyaven del maquinari. La prova és que encara en l'actualitat les bases de dades relacionals continuen essent les més utilitzades, tant pel que fa a sistemes de gestió de bases de dades com en les aplicacions que utilitzen els sistemes d'informació.

No obstant això, en el darrer mil·lenni, amb l'aparició d'Internet i els dispositius mòbils, la internalització i la compartició de la informació, el panorama ha canviat força i les bases de dades relacionals ja no donen solució a tots els problemes plantejats, ni són tan eficients com caldria esperar. Per superar aquestes limitacions i donar resposta a les noves necessitats, darrerament han aparegut altres models de bases de dades i extensions al model relacional, alguns dels quals encara són tendències.

3. Objectius i serveis dels SGBD

Els SGBD que actualment són al mercat pretenen satisfer tot un conjunt d'objectius directament deduïbles del que hem explicat fins ara. Tot seguit els comentarem.

3.1. Consultes no predefinides i complexes

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

Consultes que afecten més d'una entitat tipus

- Es vol conèixer el nombre d'estudiants de més de vint-i-cinc anys i amb nota mitjana superior a set que hi ha actualment matriculats a l'assignatura d'*Àlgebra*.
- De cada estudiant matriculat en menys de tres assignatures, se'n vol obtenir el nom, el número de matrícula, el nom de les assignatures i el nom dels professors d'aquestes assignatures.

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

Fitxers tradicionals

En els fitxers tradicionals, cada vegada que es volia fer una consulta s'havia d'escriure un programa a mida.

L'usuari ha de poder formular la consulta amb un **llenguatge senzill** (que es quedi, òbviament, en el nivell lògic) i el sistema l'ha d'interpretar directament. Això no vol dir, però, que no es puguin escriure programes amb consultes incorporades, com, per exemple, per a processos repetitius.

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

3.2. Flexibilitat i independència

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

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

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

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

En el món dels fitxers ja hi havia independència física en un cert grau, però en el món de les BD acostuma a ser molt més gran.

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

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

Independència lògica de les dades

Per exemple, el fet de suprimir l'atribut data de naixement de l'entitat estudiant i afegir-hi una entitat aula no hauria d'afectar cap dels programes existents que no utilitzin l'atribut data de naixement.

Vegeu també

Les diferents visions lògiques d'una BD es veuran en l'apartat 4 d'aquest mòdul didàctic.

Exemples d'independència lògica

1) El personal administratiu de secretaria podria tenir una visió de l'entitat *estudiant* com si no existís l'atribut *nota*. Però els usuaris professors (o els programes dirigits a ells) podrien tenir una visió en què existís l'atribut *nota* però no l'atribut *data de pagament*.

2) Decidim ampliar la longitud de l'atribut *nom* i l'augmentem de trenta a cinquanta caràcters, però els programes que ja tenim escrits no caldria modificar-los si no ens importa que els valors obtinguts tinguin només els primers trenta caràcters del nom.

Independència lògica

Els sistemes de gestió de fitxers tradicionals no donen gens d'independència lògica. Els SGBD sí que en donen; un dels seus objectius és aconseguir-ne la màxima possible, però en donen menys del que seria desitjable.

3.3. Problemes de la redundància

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

Segurament penseu que el problema de la redundància és l'espai perdut. Antigament, quan el preu del *byte* de disc era elevadíssim, això era un problema greu, però actualment gairebé mai no ho és. Quin problema hi ha, doncs? Simplement, allò que tots hem patit més d'una vegada; si tenim apuntada alguna

Activitat

Per què volem evitar la redundància? Quin problema ens porta?

cosa en dos llocs diferents no passarà gaire temps fins que les dues anotacions deixin de ser coherents, perquè haurem modificat l'anotació en un dels llocs i ens haurem oblidat de fer-ho en l'altre.

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

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

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

Vegeu també

Per a recordar el que s'ha dit sobre dades reproduïdes, vegeu el subapartat 2.3 d'aquest mòdul didàctic.

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

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

Dades derivades

És freqüent tenir dades numèriques acumulades o agregades: l'import total de totes les matrícules fetes fins avui, el nombre mitjà d'estudiants per assignatura, el saldo de la caixa de l'oficina, etc.

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

3.4. Integritat de les dades

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

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

Regles d'integritat

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

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

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

Regles d'integritat del model

Un SGBD relacional, per exemple, mai no acceptarà que una taula tingui files duplicades; un SGBD jeràrquic mai no acceptarà que una entitat tipus estigui definida com a filla de dues entitats tipus diferents, etc.

A part de les regles d'integritat que el dissenyador de la BD pot definir i que l'SGBD entendrà i farà complir, el mateix SGBD té regles d'integritat inherents al model de dades que utilitza i que sempre s'acompliran. Són les anomenades **regles d'integritat del model**. Les regles definibles per l'usuari són les **regles d'integritat de l'usuari**. El concepte d'*integritat de les dades* va més enllà de la prevenció que els programes usuaris emmagatzemin dades incorrectes. En casos d'errades o desastres, també podríem perdre la integritat de les dades. L'SGBD ens ha de donar les eines per a poder reconstruir o restaurar les dades malmeses.

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

3.5. Concurrència d'usuaris

Usuaris concurrents

Actualment ja no són rars els SI que tenen, en uninstant determinat, milers de sessions d'usuari obertessimultàniament. No calpensar en els típics sistemesdels consorcis de companyiesaèries o casos similars, n'hiha prou en pensar en elsservidors de pàgines web.

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

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

Quan un usuari o més d'un estan actualitzant les dades, es poden produir **problemes d'interferència** que tinguin com a conseqüència l'obtenció de dades errònies i la pèrdua d'integritat de la BD.

Per a tractar els accessos concurrents, els SGBD fan servir el concepte de transacció de BD, concepte d'especial utilitat per a tot allò que fa referència a la integritat de les dades, com veureu tot seguit.

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

Exemples de transaccions

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

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

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

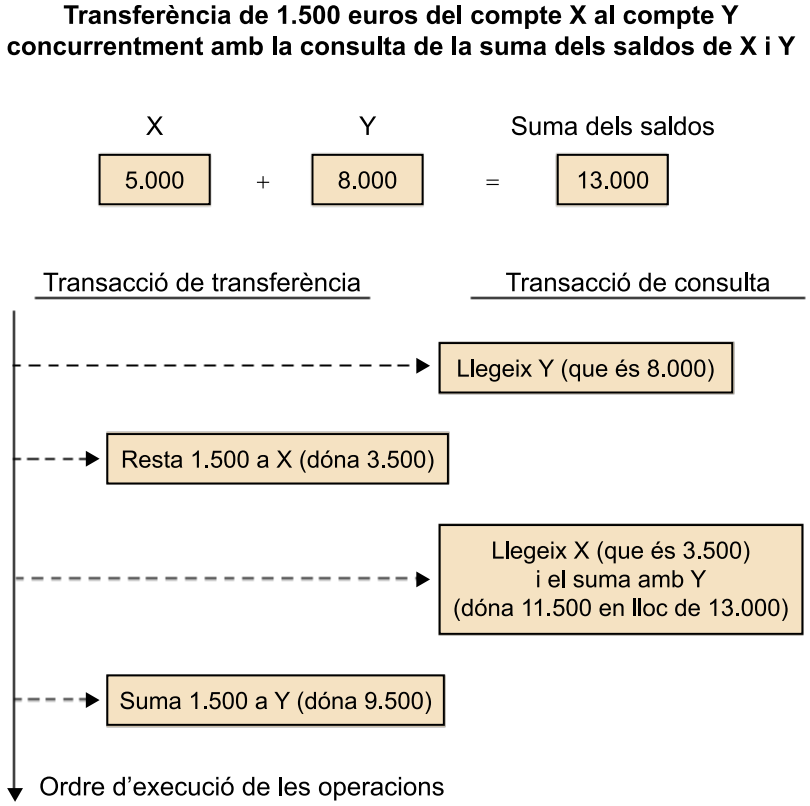
Acabem de veure la utilitat del concepte de *transacció* per al manteniment de la integritat de les dades en cas d'interrupció d'un conjunt d'operacions lògicament unitari. Però entre transaccions que s'executen concurrentment es poden produir problemes d'interferència que facin obtenir resultats erronis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

Conseqüències de la interferència entre transaccions

1) Imaginem que una transacció que transfereix diners de X a Y s'executa concurrentment amb una transacció que mira el saldo dels comptes Y i X, en aquest ordre, i en mostra la suma. Si l'execució concurrent de les dues transaccions casualment és tal que la transferència s'executa entre l'execució de les dues lectures de la transacció de suma, pot produir resultats incorrectes. A més, si els decideix escriure a la BD, aquesta quedarà inconsistent (vegeu la figura 3 a la pàgina següent).

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

Figura 3



Ens interessarà que l'SGBD executi les transaccions de manera que no s'interfereixin, és a dir, que quedin aïllades les unes de les altres. Per a aconseguir que les transaccions s'executin com si estiguessin aïllades, els SGBD utilitzen diverses tècniques, de les quals la més coneguda és el **bloqueig** (*lock*).

El bloqueig d'unes dades en benefici d'una transacció consisteix a posar limitacions als accessos que les altres transaccions podran fer a aquestes dades.

Bloqueigs en la transferència de diners

Per exemple, quan latransacció de transferència de diners modifica el saldo del compte X, provocarà que l'SGBD bloquegi aquest compte de manera que, quan la transacció de suma vulgui llegir-lo, hagi d'esperar que la transacció de transferència acabi, ja que quan una transacció acaba, quan s'efectuen les operacions `COMMIT` o `ROLLBACK`, s'alliberen els objectes que tenia bloquejats.

Quan es provoquen bloqueigs es produeixen esperes, retencions i, en conseqüència, el sistema és més lent. Els SGBD s'esforcen a minimitzar aquests efectes negatius.

3.6. Seguretat

El terme *seguretat* s'ha utilitzat en diferents sentits al llarg de la història de la informàtica.

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

Aquestes qüestions sempre han estat importants en els SI militars, les agències d'informació i en àmbits similars, però durant els anys noranta han anat adquirint importància en qualsevol SI on s'emmagatzemin dades sobre persones. Recordeu que a l'Estat espanyol tenim una llei[2] que exigeix la protecció de la confidencialitat d'aquestes dades.

[(2)]Llei orgànica 15/1999, de 13 de desembre, de Protecció de Dades de caràcter personal, (BOE núm. 298, de 12/12/1999, pp. 43088-43099).

Els SGBD permeten definir **autoritzacions** o drets d'accés a diferents nivells: al nivell global de tota la BD, al nivell entitat i al nivell atribut.

Drets d'accés

Per exemple, l'usuari SECRE3 podria tenir autorització per a consultar i modificar totes les entitats de la BD, excepte el valor de l'atribut nota dels estudiants, i no estar autoritzat a fer cap mena de supressió o inserció.

Aquests mecanismes de seguretat requereixen que l'usuari es pugui identificar. S'acostuma a utilitzar codis d'usuari (i grups d'usuaris) acompanyats de contrasenyes (*passwords*), però també s'utilitzen targetes magnètiques, identificació per reconeixement de la veu, etc.

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

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

4. Arquitectura dels SGBD

A continuació veurem algunes nocions bàsiques de l'arquitectura dels SGBD.

4.1. Esquemes i nivells

Per a treballar amb les nostres BD, els SGBD necessiten conèixer-ne l'estructura (quines entitats tipus hi haurà, quins atributs tindran, etc.).

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

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

Anteriorment, ja hem parlat de la distinció entre dos nivells de representació informàtica: el nivell lògic i el físic.

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

Per qüestions de rendiment, ens podrà interessar descriure elements de **nivell físic** com, per exemple, quins índexs tindrem i quines característiques presentaran, com i on (en quin espai físic) volem que s'agrupin físicament els registres, de quina mida han de ser les pàgines, etc.

En el període 1975-1982, ANSI intentava establir les bases per a crear estàndards en el camp de les BD. El comitè conegut com a ANSI/SPARC va recomanar que l'arquitectura dels SGBD preveiés tres nivells de descripció de la BD, no solament dos.[3]

(3)De fet, l'any 1971, el comitè CODASYL ja havia proposat els tres nivells d'esquemes.

D'acord amb l'arquitectura ANSI/SPARC, hi havia d'haver tres nivells d'esquemes, tres nivells d'abstracció. La idea bàsica d'ANSI/SPARC consistia a descompondre el nivell lògic en dos: el **nivell extern** i el **nivell conceptual**. Anomenaven **nivell intern** el que aquí hem anomenat *nivell físic*.

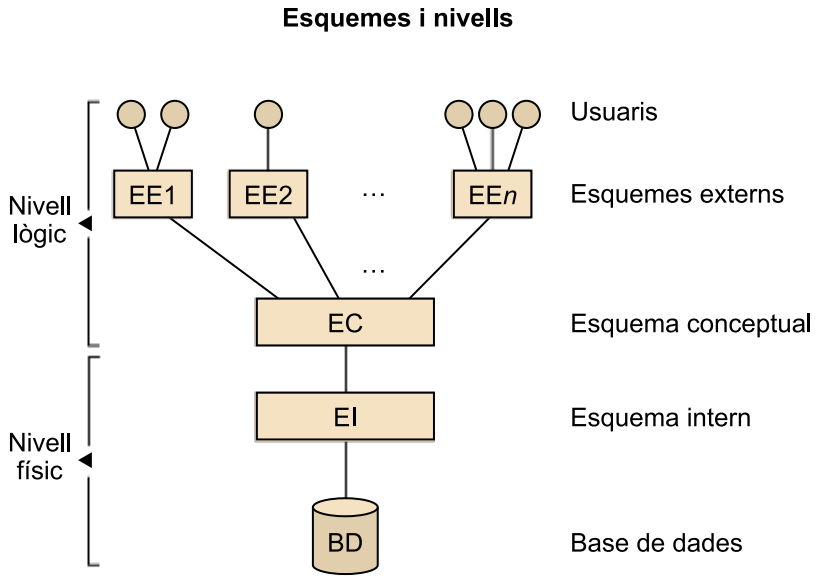
Així, doncs, d'acord amb ANSI/SPARC, hi haurien els tres nivells d'esquemes que esmentem a continuació:

a) Al nivell extern se situen les diferents visions lògiques que els processos usuaris (programes d'aplicació i usuaris directes) tindran de les parts de la BD que utilitzaran. Aquestes visions s'anomenen **esquemes externs**.

b) Al 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.

c) Al nivell físic hi ha una única descripció física, que anomenem **esquema intern**.

Figura 4



A l'**esquema conceptual**, s'hi descriuran les entitats tipus, els seus atributs, les interrelacions i també les restriccions o regles d'integritat.

L'esquema conceptual correspon a les necessitats del conjunt de l'empresa o del SI, per la qual cosa s'escriurà de manera centralitzada durant l'anomenat *disseny lògic* de la BD.

Però cada aplicació podrà tenir la seva visió particular, i segurament parcial, de l'esquema conceptual. Els usuaris (programes o usuaris directes) veuran la BD a través d'esquemes externs apropiats a les seves necessitats. Aquests esquemes es poden considerar com redefinicions de l'esquema conceptual, amb les parts i els termes que convinguin per a les necessitats de les aplicacions (o grups d'aplicacions). Alguns sistemes els anomenen *subesquemes*.

En definir un **esquema extern**, se citaran només aquells atributs i entitats que interessin, els podrem reanomenar, podrem definir dades derivades, podrem redefinir una entitat perquè les aplicacions que utilitzen aquest esquema extern creguin que en són dos, o podrem definir combinacions d'entitats perquè en semblin una de sola, etc.

Exemple d'esquema extern

Imaginem una BD que a l'esquema conceptual té definida, entre moltes altres, una entitat *estudiant* amb els atributs següents: *num_matri, nom, cognom, num_DNI, adreça, data_naix, telefon*. Però ens pot interessar que uns determinats programes o usuaris vegin la BD formada d'acord amb un esquema extern que tingui definides dues entitats, anomenades *estudiante* i *persona*.

a) L'entitat *estudiante* podria tenir definit l'atribut *número_matrícula* (definit com a derivable directament de *num_matri*), l'atribut *nombre_pila* (de *nom*), l'atribut *apellido* (només els primers quaranta caràcters de *cognom*) i l'atribut *DNI* (de *num_DNI*).

b) L'entitat *persona* podria tenir l'atribut *DNI* (obtingut de *num_DNI*), l'atribut *nombre* (format per la concatenació de *nom* i *cognom*), l'atribut *dirección* (obtingut d'*adreça*) i l'atribut *edad* (derivable automàticament de *data_naix*).

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.

L'esquema de nivell intern respon a les qüestions de rendiment (espai i temps) plantejades en fer el disseny físic de la BD i en ajustar-lo[4] posteriorment a les necessitats canviants.

[(4)]En anglès, l'ajust es coneix amb el nom de *tuning*.

Activitat

Què cal fer per a crear una BD?

D'acord amb l'arquitectura ANSI/SPARC, per a crear una BD cal definir-ne prèviament l'esquema conceptual, definir-ne com a mínim un esquema extern i, eventualment, definir-ne l'esquema intern. Si aquest últim no es defineix, el mateix SGBD haurà de decidir els detalls de l'organització física. L'SGBD s'encarregarà de fer les correspondències (*mappings*) entre els tres nivells d'esquemes.

Esquemes i nivells en els SGBD relacionals

En els SGBD relacionals, és a dir, en el món de l'SQL, s'utilitza una terminologia lleugerament diferent. No se separen clarament tres nivells de descripció. Es parla d'un sol esquema –***schema***–, però dintre d'aquest s'inclouen descripcions dels tres nivells. En l'*schema* es descriuen els elements del que en l'arquitectura ANSI/SPARC s'anomena *esquema conceptual* (entitats tipus, atributs i restriccions) més les vistes –*view*–, que corresponen aproximadament als esquemes externs.

El model relacional en què l'SQL està inspirat es limita al món lògic. Per això l'estàndard ANSI-ISO de l'SQL no parla en absolut del món físic o intern; ho deixa en mans dels SGBD relacionals del mercat. Però aquests donen la possibilitat d'incloure dintre l'*schema* descripcions d'estructures i característiques físiques (índex, *tablespace*, *cluster*, espais per a excessos, etc.).

4.2. Independència de les dades

En aquest subapartat veurem com l'arquitectura de tres nivells que acabem de presentar ens proporciona els dos tipus d'independència de les dades: la física i la lò-gica.

Vegeu també

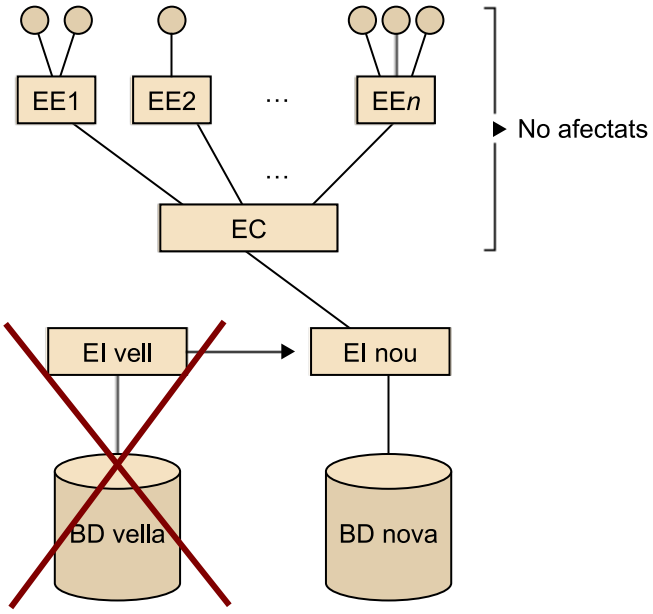
Tots dos tipus d'independència de les dades s'han explicat en el subapartat 3.2 d'aquest mòdul didàctic.

Hi ha **independència física** quan els canvis en l'organització física de la BD no afecten el món exterior, és a dir, els programes usuaris o els usuaris directes.

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

Figura 5

Independència física



És obvi que quan canviem unes dades d'un suport a un altre, o les canvien de lloc dintre d'un suport, no es veuran afectats ni els programes d'aplicació ni els usuaris directes, ja que no es modificarà l'esquema conceptual ni els externs. Però tampoc no s'haurien de veure afectats si canviéssim, per exemple, el mètode d'accés a uns registres determinats[5], el format o la codificació, etc. Cap d'aquests casos no hauria d'afectar el món exterior, només la BD física; és a dir, l'esquema intern.

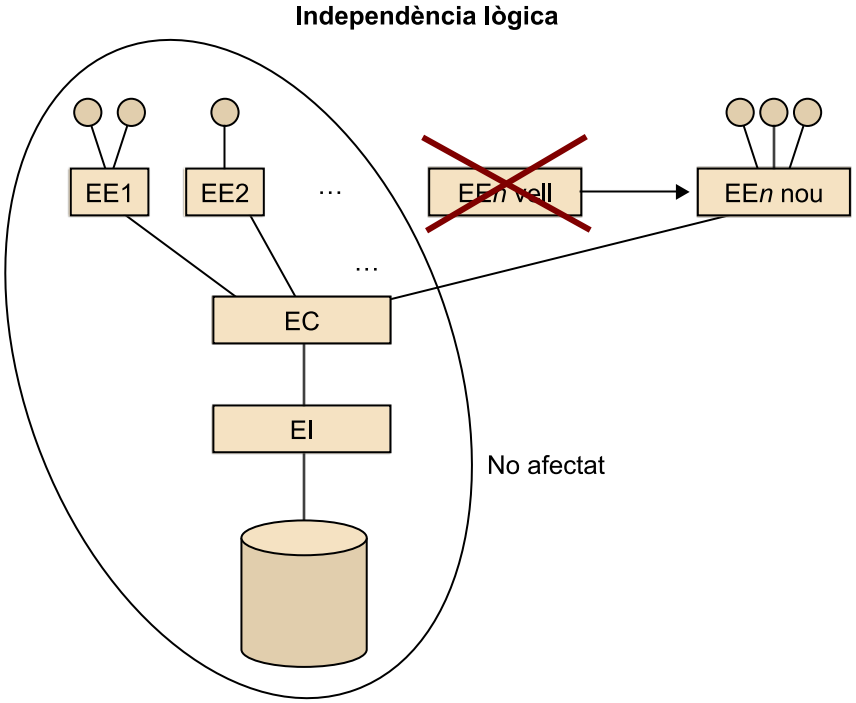
[5] Per exemple eliminant un índex o substituint-lo per un altre.

Si hi ha independència física de les dades, l'única cosa que variarà en canviar l'esquema intern són les correspondències entre l'esquema conceptual i l'intern. I, òbviament, la majoria dels canvis de l'esquema intern obligaran a refer la BD real, la física.

Hi ha **independència lògica** quan els usuaris[6] no es veuen afectats pels canvis en el nivell lògic.

[6] Programes d'aplicació o usuaris directes.

Figura 6



Donats els dos nivells lògics de l'arquitectura ANSI/SPARC, diferenciarem les dues situacions següents:

1) Canvis en l'esquema conceptual. Un canvi d'aquest tipus no afectarà els esquemes externs que no facin referència a les entitats o als atributs modificats.

Exemple

Si eliminem l'atribut *cognom*, per exemple, no es veuran afectats els esquemes externs (ni els usuaris) que no hi facin referència. Si s'allarga l'atribut *adreça* a l'esquema conceptual, no caldrà modificar l'esquema extern on s'ha definit la *dirección* ni, òbviament, resultaran afectats els programes i els usuaris que la utilitzen.

2) Canvis en els esquemes externs. Efectuar canvis en un esquema extern afectarà els usuaris que facin servir els elements modificats, però no hauria d'afectar els altres, ni l'esquema conceptual, ni, en conseqüència, l'esquema intern.

Usuaris no afectats pels canvis

Noteu que no tots els canvis d'elements d'un esquema extern afectaran els seus usuaris. Vegem-ne un exemple: abans hem vist que, quan eliminàvem l'atribut *cognom* de l'esquema conceptual, havíem de modificar l'esquema extern on definíem *nombre*, perquè allí estava definit com a concatenació de *nom* i *cognom*. Doncs bé, un programa que utilitzés l'atribut *nombre* no es veuria afectat si modifiquéssim l'esquema extern de manera que *nombre* fos la concatenació de *nom* i una cadena constant, per exemple tota en blanc. Com a resultat, hauria desaparegut el cognom de *nombre*, sense haver de modificar el programa.

Els SGBD actuals donen força independència lògica, però menys del que caldria, ja que les exigències de canvis constants en els SI demanen graus molt elevats de flexibilitat. Els sistemes de fitxers tradicionals, en canvi, no en donen gens.

4.3. Flux de dades i de control

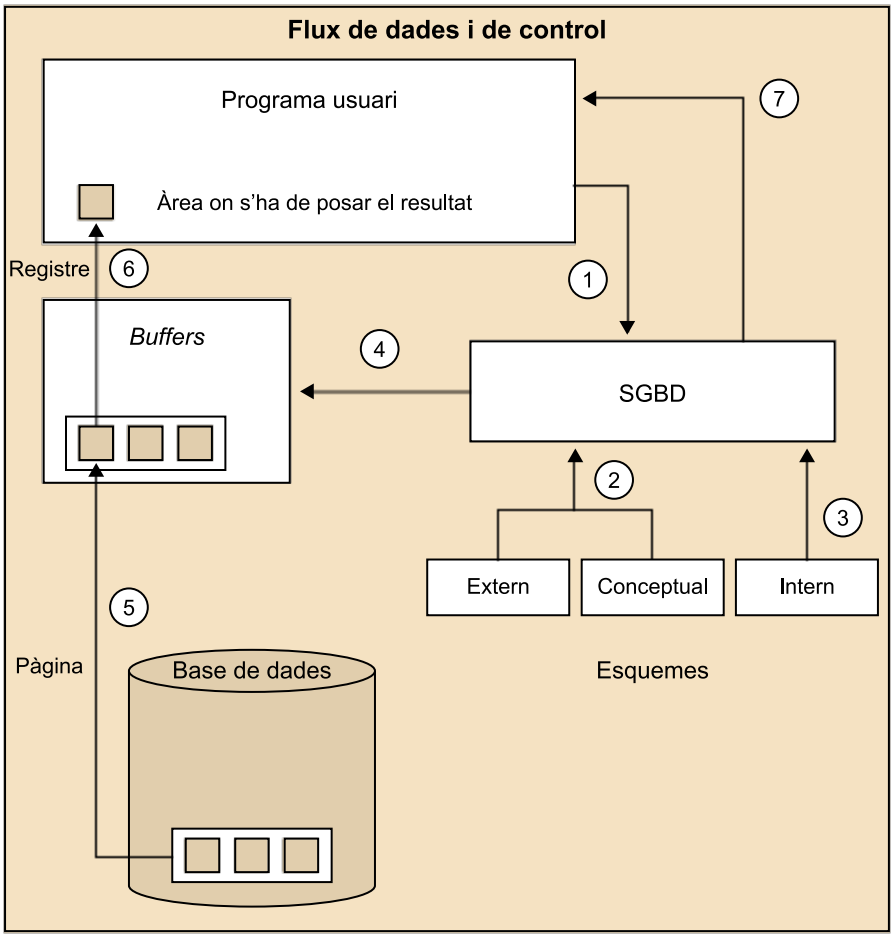
Per a entendre el funcionament d'un SGBD, a continuació veurem el principals passos de l'execució d'una consulta sotmesa a l'SGBD per un programa d'aplicació. Explicarem les línies generals del flux de dades i de control entre l'SGBD, els programes usuari i la BD.

Recordeu que l'SGBD, amb l'ajut del SO, llegeix pàgines (blocs) dels suports on està emmagatzemada la BD física, i les porta a una àrea de *buffers* o memòries cau en la memòria principal. Des dels *buffers* cap a l'àrea de treball del mateix programa, l'SGBD hi passa registres.

Suposem que la consulta demana les dades de l'estudiant que té un determinat DNI. Per tant, la resposta que el programa obtindrà serà un sol registre i el rebrà dintre d'una àrea de treball pròpia[7].

(7)Per exemple, una variable amb estructura de tupla.

Figura 7. Execució d'una consulta



A la figura veiem representada la BD, els tres nivells d'esquemes, l'àrea dels *buffers*, l'SGBD i el programa d'aplicació que li fa la consulta.

El procés que se segueix és el següent:

a) Comença amb una crida (1) del programa a l'SGBD, en la qual se li envia l'operació de consulta. L'SGBD ha de verificar que la sintaxi de l'operació rebuda sigui correcta, que l'usuari del programa estigui autoritzat a fer-la, etc. Per a poder fer tot això, l'SGBD mira (2) l'esquema extern amb el qual treballa el programa i l'esquema conceptual.

b) Si la consulta és vàlida, l'SGBD determina, consultant l'esquema intern (3), quin mecanisme ha de seguir per a respondre-la. Ja sabem que el programa usuari no diu res respecte a com s'ha de fer físicament la consulta. És l'SGBD qui ho ha de determinar. Quasi sempre hi ha diverses maneres, diferents camins, per a respondre una consulta[8]. Suposem que ha escollit aplicar un *hashing* al valor del *DNI* que és el paràmetre de la consulta, i el resultat és l'adreça de la pàgina on es troba (entre molts altres) el registre de l'estudiant buscat.

[8]Per exemple, sempre té la possibilitat de fer una cerca seqüencial.

c) Quan ja sap quina és la pàgina, l'SGBD mirarà (4) si per sort aquesta pàgina ja es troba en aquell moment a l'àrea dels *buffers* (potser com a resultat d'una consulta anterior d'aquest usuari o d'un altre). Si no hi és, l'SGBD, amb l'ajut del SO, la busca al disc i la carrega als *buffers* (5). Si ja hi és, s'estalvia l'accés al disc.

d) Ara, la pàgina desitjada ja és a la memòria principal. L'SGBD n'extreu, d'entre els diversos registres que la pàgina pot contenir, el registre buscat, i interpreta la codificació i el format segons el que digui l'esquema intern.

e) L'SGBD aplica a les dades les eventuals transformacions lògiques que implica l'esquema extern (potser tallant l'adreça per la dreta) i les porta a l'àrea de treball del programa (6).

f)A continuació l'SGBD retorna el control al programa (7) i dóna per acabada l'execució de la consulta.

Diferències entre SGBD

Encara que entre diferents SGBD pot haver-hi enormes diferències de funcionament, solen seguir l'esquema general que acabem d'explicar.

5. Models de BD

Una BD és una representació de la realitat (de la part de la realitat que ens interessa en el nostre SI). Dit d'una altra manera, una BD es pot considerar com un model de la realitat. El component fonamental per a modelar en un SGBD relacional són les taules (anomenades *relacions* en el món teòric). Però en altres tipus de SGBD s'utilitzen altres components.

El conjunt de components, o eines conceptuals, que un SGBD proporciona per a modelar rep el nom de **model de BD**. Alguns exemples de models de BD són el **model relacional**, el **model jeràrquic**, el **model en xarxa** i el **model relacional amb objectes**.

Compte amb les confusions!

Popularment, especialment en el camp de la informàtica personal, s'anomena BD allò que aquí anomenem SGBD. No s'ha de confondre, tampoc, la BD considerada com a model de la realitat amb el que aquí anomenem model de BD. El model de BD és el conjunt d'eines conceptuals (peces) que es fan servir per a construir el model de la realitat.

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

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

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

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

Evolució dels models de BD

Dels quatre models de BD que hem citat, el que primer va aparèixer, al principi dels anys seixanta, va ser el **model jeràrquic**. Les seves estructures són registres inter-relacionats en forma d'arbres. L'SGBD clàssic d'aquest model és l'IMS/DL1 d'IBM.

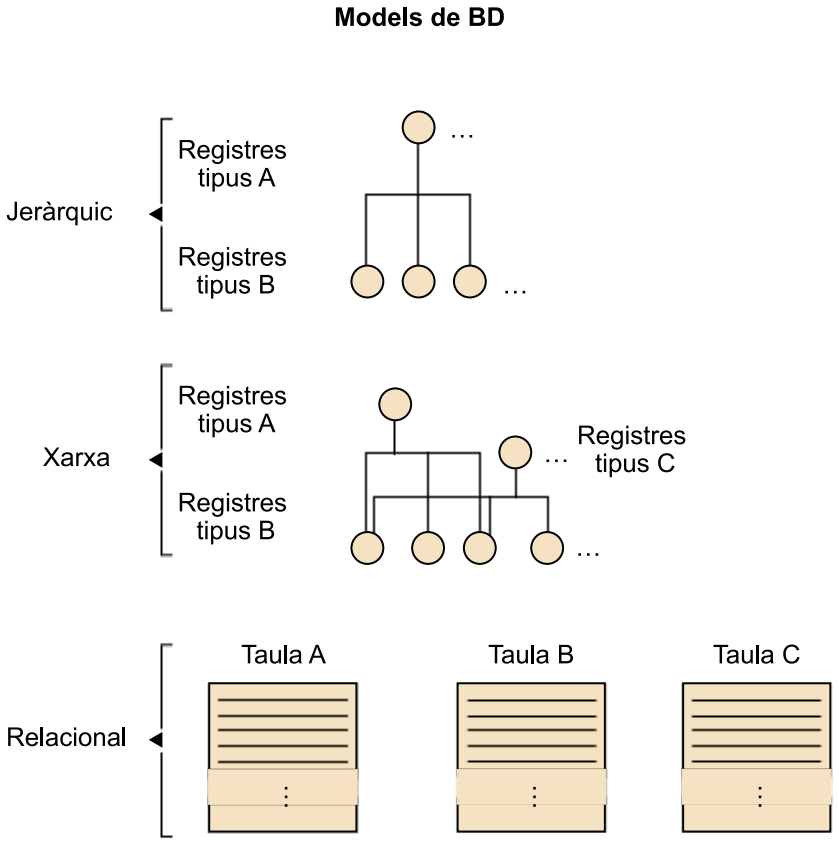
A començament dels setanta van sortir SGBD basats en un **model en xarxa**. Com en el model jeràrquic, hi ha registres i interrelacions, però un registre ja no està limitat a ser "fill" d'un sol registre tipus. El comitè CODASYL-DBTG va proposar un estàndard basat en aquest model que va ser adoptat per molts constructors de SGBD[9]. Però va trobar l'oposició d'IBM, l'empresa dominant aleshores. La proposta de CODASYL-DBTG ja definia tres nivells d'esquemes.

[9]Per exemple, l'IDS de Bull, el DMS d'Univac i el DBMS de Digital.

Des dels anys vuitanta han anat apareixent gran quantitat de SGBD basats en el **model relacional** proposat el 1969 per E.F. Codd, d'IBM, i gairebé tots utilitzen com a llenguatge nadiu l'SQL[10]. El model relacional es basa en el con-

[10]Per exemple, l'Oracle, el DB2d'IBM, l'Informix, el MySQL, PostgreSQL i l'SQL Server deMicrosoft.

cepte matemàtic de *relació*, que aquí de moment podem considerar equivalent al terme *taula* (formada per files i columnes). La major part dels SI que actualment estan en funcionament utilitzen SGBD relacionals.



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

Una altra diferència important entre els models prerelacionals i el model relacional és que aquest es limita al nivell lògic, no fa absolutament cap consideració sobre les representacions físiques. És a dir, ens dóna una independència física de dades total. Això és veritat si parlem del model teòric, però els SGBD del mercat ens donen una independència limitada.

El **model de BD relacional amb objectes** tracta d'ampliar el model relacional, afegint-hi la possibilitat que els tipus de dades siguin tipus abstractes de dades, TAD. Això acosta els sistemes relacionals al paradigma de la l'OO. Alguns SGBD relacionals (per exemple l'Oracle, el DB2 i PostgreSQL) han adoptat ampliacions d'aquest estil però no s'estan utilitzant gaire. Actualment, la visió de les BD relacionals des del paradigma OO s'acostuma a fer amb eines especialitzades com, per exemple, els ORM *frameworks* (Java EE/EJB, Hibernate, NET-EF, etc.), que permeten treballar amb classes persistents sense haver de tractar directament amb taules relacionals.

Parlem de models de BD, però de fet s'acostumen a anomenar *models de dades*, ja que permeten modelar les dades. Però hi ha models de dades que no són utilitzats pels SGBD del mercat, no s'utilitzen en les realitzacions, sinó únicament durant el procés d'anàlisi i disseny.

Els més coneguts d'aquests tipus de models de dades són els **models semàntics** i els **funcionals**. Aquests models ens donen eines molt potents per a descriure les estructures de la informació del món real, la semàntica i les interrelacions, però normalment no disposen d'operacions per a tractar-les. Es limiten a ser eines de descripció lògica. Són molt utilitzats en l'etapa del disseny de BD i en eines CASE. El més estès d'aquests models de dades és el model ER (entitat-relació o *entity-relationship*). En el món de l'OO és molt popular el diagrama de classes UML (llenguatge unificat de modelització o *unified modeling language*). Des del món de les BD s'acostuma a considerar que les classes persistents (i les seves associacions) d'un diagrama de classes UML representen el mateix que les entitats (i interrelacions) d'un model de dades ER. Existeixen algunes eines que permeten desenvolupar BD a nivell ER i UML sense haver de baixar a nivell de taula. Això és equivalent al que permeten fer les eines ORM esmentades anteriorment.

Més a prop del món lògic

L'evolució dels models al llarg dels anys els ha anat allunyant del món físic i els ha apropat al món lògic, és a dir, s'han allunyat de les màquines i s'han apropat a les aplicacions.

En aquesta assignatura parlem només de BD amb models de dades estructurats, que són els que normalment s'utilitzen en els SI empresarials. Però hi ha SGBD especialitzats en tipus d'aplicacions concretes que no segueixen cap d'aquests models.

6. Llenguatges i usuaris

Per a comunicar-se amb l'SGBD l'usuari, ja sigui un programa d'aplicació o un usuari directe, es val d'un llenguatge. Hi ha molts llenguatges diferents segons els tipus d'usuaris per als quals estan pensats i els tipus de coses que els usuaris han de poder expressar-hi:

a) Hi haurà usuaris informàtics molt experts que voldran escriure processos complexos i que necessitaran llenguatges complexos.

b) Però hi haurà usuaris finals no informàtics, casuals (esporàdics), que només faran consultes. Aquests necessitaran un llenguatge molt senzill, encara que doni un rendiment baix en temps de resposta.

c) També hi podrà haver usuaris finals no informàtics, dedicats o especialitzats. Són usuaris quotidians o, fins i tot, dedicats exclusivament a treballar amb la BD[11]. Aquests necessitaran llenguatges molt eficients i compactes, encara que no sigui fàcil aprendre'ls. Potser seran llenguatges especialitzats en tipus concrets de feines.

Què hauria de poder dir l'usuari a l'SGBD?

D'una banda, la persona que fa el disseny ha de poder descriure a l'SGBD la BD que ha dissenyat. D'altra banda, s'ha de poder demanar al sistema que ompli i actualitzi la BD amb les dades que se li donin. I, òbviament, l'usuari ha de disposar de mitjans per a fer-li consultes.

(11)Per exemple, persones dedicades a introduir-hi dades massivament.

Hi ha llenguatges especialitzats en l'escriptura d'esquemes, és a dir, en la descripció de la BD. Es coneixen genèricament com a **DDL** o *data definition language*. Fins i tot hi ha llenguatges específics per a esquemes interns, llenguatges per a esquemes conceptuals i llenguatges per a esquemes externs.

Altres llenguatges estan especialitzats en la utilització de la BD (consultes i manteniment). Es coneixen com a **DML** o *data management language*. Però el més freqüent és que el mateix llenguatge disposi de construccions per a les dues funcions, DDL i DML.

El **llenguatge SQL**, que és el més utilitzat en les BD relacionals, té verbs –instruccions– de tres tipus diferents:

1) Verbs del tipus DML, per exemple `SELECT` per a fer consultes, `INSERT`, `UPDATE` i `DELETE` per a fer el manteniment de les dades.

2) Verbs del tipus DDL, per exemple `CREATE TABLE` per a definir les taules, les seves columnes i les restriccions.

3) A més, l'SQL té **verbs de control de l'entorn**, com per exemple `COMMIT` i `ROLLBACK` per a delimitar transaccions.

Quant als **aspectes DML**, podem diferenciar dos tipus de llenguatges:

a) Llenguatges molt **declaratius** (o implícits), amb els quals s'especifica què es vol fer sense explicar com s'ha de fer.

b) Llenguatges més explícits o **procedimentals**, que ens demanen conèixer més qüestions del funcionament de l'SGBD per a poder detallar pas a pas com s'han de fer les operacions (el que s'anomena *navegar* per la BD).

Llenguatges declaratius i procedimentals

L'aprenentatge i la utilització dels llenguatges procedimentals acostumen a ser més difícils que els declaratius, i per això són utilitzats només per usuaris informàtics. Amb els procedimentals es poden escriure processos més eficients que amb els declaratius.

Òbviament, els **aspectes DDL**, les descripcions de les dades, són sempre declaratives per la seva pròpia naturalesa.

Els llenguatges utilitzats en els SGBD prerelacionals eren procedimentals. L'SQL és bàsicament declaratiu, però té possibilitats procedimentals.

Encara que quasi tots els SGBD del mercat tenen l'SQL com a llenguatge nadiu, ofereixen altres possibilitats, com per exemple 4GL i eines visuals:

1) Llenguatges 4GL (*4th Generation Languages*[12]) de molt alt nivell, que solen combinar elements procedimentals amb elements declaratius. Pretenen fer molt fàcil no tan sols el tractament de la BD, sinó també la definició de menús, pantalles i diàlegs.

2) Eines o **interfícies visuals**[13] molt fàcils de fer servir, que permeten utilitzar les BD seguint l'estil de diàlegs amb finestres, icones i ratolí, posat de moda per les aplicacions Windows. No solament són útils als usuaris no informàtics, sinó que faciliten molt la feina als usuaris informàtics: permeten consultar i actualitzar la BD, i també definir-la i actualitzar-ne la definició amb molta facilitat i claredat.

(12)Van proliferar cap als anys vuitanta.

(13)Van començar a proliferar als anys noranta; cada cop són més potents i sofisticats, i es poden integrar amb altres eines, com ara les eines CASE.

Tant els 4GL com les eines visuals (sovint unides com una sola eina) tradueixen el que fa l'usuari en instruccions SQL per vies diferents:

- En el cas dels 4GL la traducció s'acostuma a fer mitjançant la compilació.
- En el cas de les eines visuals s'efectua per mitjà de l'intèrpret de SQL integrat en l'SGBD.

Si volem escriure un programa d'aplicació que treballi amb BD, segurament voldrem utilitzar el nostre llenguatge habitual de programació[14]. Però aquests llenguatges generalment no tenen instruccions per a realitzar accesos a les BD. Llavors tenim les dues opcions següents:

[14]C, C++, C#, CoboL, Java, etc.

1) Les **crides a funcions**: al mercat hi ha llibreries de funcions especialitzades en BD (per exemple, les llibreries ODBC i JDBC). Només cal incloure crides a les funcions desitjades dintre el programa escrit amb el llenguatge habitual. Seran les funcions les que s'encarregaran d'enviar les instruccions (generalment en SQL) en temps d'execució a l'SGBD.

2) El **llenguatge hostatjat**: una altra possibilitat consisteix a incloure directament les instruccions del llenguatge de BD en el nostre programa. Però això exigeix utilitzar un precompilador especialitzat que accepti en el llenguatge de programació habitual les instruccions del llenguatge de BD. Llavors es diu que aquest llenguatge (quasi sempre l'SQL) és el llenguatge hostatjat o incorporat (*embedded*), i el llenguatge de programació (C, C++, C#, CoboL, Java, etc.) és el llenguatge amfitrió (*host*).

7. Administració de BD

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

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

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

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

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

3) Resolució d'emergències.

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

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

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

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

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

La feina de l'ABD no és senzilla

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

Resum

En aquest mòdul hem fet una introducció als conceptes fonamentals del món de les BD i dels SGBD. Hem explicat l'evolució dels SGBD, que va des d'una estructura centralitzada i poc flexible fins a una de distribuïda i flexible; des d'una utilització procedimental que requeria molts coneixements, fins a un ús declaratiu i senzill.

Hem revisat els **objectius dels SGBD actuals** i alguns dels **serveis** que ens donen per a aconseguir-los. És especialment important el concepte de **transacció** i la manera com és utilitzat per a vetllar per la integritat de les dades.

L'**arquitectura de tres nivells** aporta una gran flexibilitat als canvis, tant als físics com als lògics. Hem vist com un SGBD pot funcionar utilitzant els tres esquemes propis d'aquesta arquitectura.

Hem explicat que els **components d'un model de BD** són les estructures, les restriccions i les operacions. Els diferents models de BD es diferencien bàsicament per les seves estructures. Hem parlat dels models més coneguts, especialment del model relacional, que està basat en taules.

Cada tipus d'usuari de l'SGBD pot utilitzar un llenguatge apropiat per a la seva feina. Uns usuaris que tenen una feina important i difícil són els **administradors de les BD**.

Exercicis d'autoavaluació

1. Quins avantatges van aportar els SGBD relacionals respecte als prerelacionals?

2. Digueu, de les afirmacions següents, quines són certes i quines falses:
a) El model ER és més conegut com a *model relacional*.
b) Els SGBD no permeten la redundància.
c) El DML és un llenguatge declaratiu.
d) El DDL és un llenguatge pensat per a escriure programes de consulta i actualització de BD.
e) Quan un programa vol accedir a unes dades a través d'un índex, ho ha de dir a l'SGBD.

Solucionari

Exercicis d'autoavaluació

1. Els SGBD relacionals van aportar una programació més senzilla: els llenguatges són més senzills i no depenen tant de les característiques físiques de la BD. Hi ha més flexibilitat en els canvis (més independència física de les dades). El programador s'ha de preocupar molt menys de les qüestions de rendiment, ja se n'ocupa l'SGBD. Inclouen llenguatges declaratius de consulta per a usuaris no informàtics.

2.a) Falsa, **b)** Falsa, **c)** Falsa, **d)** Falsa, **e)** Falsa.

Glossari

administrador de BD (ABD) *m* Tipus d'usuari especial que realitza funcions centralitzades d'administració i control de la BD que asseguren que l'explotació de la BD és correcta.

base de dades (BD) *f* Conjunt estructurat de dades que representa entitats i les seves interrelacions. La representació serà única, integrada, malgrat que ha de permetre utilitzacions diverses i simultànies.

client/servidor (C/S) *m* Tecnologia habitual per a distribuir dades. La idea és que dos processos diferents, que es poden executar en un mateix sistema o en sistemes separats, actuen de manera que l'un fa declient o peticionari d'un servei i l'altre fa de servidor. Un procés client pot demanar serveis a diversos servidors. Un servidor pot rebre peticions de molts clients. En general, un procés que fa de client demanant un servei a un altre procés *B*, pot fer també de servidor d'un servei que li demani un altre procés *C*.

***Data Definition Language* (DDL)** *m* Llenguatge especialitzat en l'escriptura d'esquemes, és a dir, en la descripció de BD.

***Data Manipulation Language* (DML)** *m* Llenguatge especialitzat en la utilització de BD (consultes i manteniment).

esquema *m* Descripció o definició de la BD. Aquesta descripció està separada dels programes i és utilitzada per l'SGBD per a saber com és la BD amb la qual ha de treballar. L'arquitectura ANSI/SPARC recomana tres nivells d'esquemes: l'extern (visió dels usuaris), el conceptual (visió global) i el físic (descripció de característiques físiques). Però els SGBD habitualment no acostumen a tenir tres nivells d'esquemes, ho tenen tot barrejat en un únic esquema.

sistema de gestió de BD (SGBD) *m* Programari que gestiona i controla BD. Les seves principals funcions són facilitar-ne la utilització a molts usuaris simultanis i de tipus diferents, independitzar l'usuari del món físic i mantenir la integritat de les dades.

***Structured Query Language* (SQL)** *m* Llenguatge especialitzat en la descripció (DDL) i la utilització (DML) de BD relacionals. Creat per IBM al final dels anys setanta i estandarditzat per ANSI-ISO l'any 1985 (l'últim estàndard de SQL és de 2008). Actualment és utilitzat pràcticament per tots els SGBD del mercat.

transacció *f* Conjunt d'operacions (de BD) que volem que s'executin com un tot (totes o cap) i aïlladament (sense interferències) d'altres conjunts d'operacions que s'executin concurrentment.

Bibliografia

Bibliografia bàsica

Date, C. J. (2001). *Introducción a los sistemas de bases de datos* (7a ed.). Prentice-Hall.

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (2006). *Fundamentos de bases de datos* (5a ed.). Madrid: McGraw-Hill.
La introducció d'aquest llibre és similar a la que hem fet aquí.

Ullman, J. D.; Widom, J. (2008). *A fist course in database system* (3a. ed). Prentice Hall.

Bibliografia complementària

Per a actualitzar la vostra visió global dels SGBD del mercat, podeu consultar a Internet les pàgines de diferents SGBD, per exemple, l'Oracle, el DB2, l'Informix, l'SQL Server, el Sybase, el MySQL o PostgreSQL.