Bases de dades distribuïdes i client/servidor

Les bases de dades i les xarxes

Pablo Costa Vallés

P03/05053/02053

UOC
Universitat Oberta de Catalunya

Índex

Introducció 5

Objectius 7

1. Visió general de l'accés a bases de dades per xarxa 9
1.1. Formes de distribució 9
1.1.1. Funcions a distribuir 9
1.1.2. Distribució del processament 10
1.1.3. Distribució de les dades 12
1.1.4. Nivells de distribució 14
1.2. Aspectes crítics 15

2. Bases de dades client/servidor 18
2.1. Clients i servidors 18
2.2. Arquitectures 20

3. Bases de dades distribuïdes 22
3.1. Regles fonamentals de treball 22
3.2. Arquitectures 23
3.3. Accés als elements d'una base de dades 25
3.4. Transaccions distribuïdes 27
3.4.1. Protocol de confirmació en dues fases 27
3.4.2. Detecció distribuïda d'abraçades mortals 29
3.5. Optimització de consultes distribuïdes 29
3.6. Mecanismes de distribució de dades 31
3.6.1. Fragmentació 31
3.6.2. Reproduir 33
3.6.3. Caus persistents 34

4. Exemple d'aplicació client/servidor i distribuïda 35

Resum 38

Exercicis d'autoavaluació 41

Solucionari 42

Glossari 44

Bibliografia 45

Introducció

Una de les grans tendències actuals de la informàtica és la interconnexió d'ordinadors mitjançant xarxes de transmissió de dades. L'objectiu de la interconnexió és facilitar l'accés a la informació i als serveis informàtics.

Aquesta tendència es fa patent, en l'àmbit del gran consum i d'una manera global, en la utilització d'Internet. No obstant això, l'impacte de la interconnexió en xarxa també és molt important a l'interior de les organitzacions que utilitzen la informàtica per a dur a terme activitats quotidianes.

Cada vegada és més important que la informàtica s'adapti a l'estructura d'aquestes organitzacions, d'acord amb la seva distribució geogràfica i les divisions funcionals de què disposi, intentant, en la mesura que sigui possible, que no hi hagi limitacions no desitjades per a accedir a la informació i usar-la.

Aquest intent d'adaptació es tradueix habitualment en l'existència de conjunts d'ordinadors d'una mateixa unitat organitzativa (seu central, sucursal, delegació, departament, etc.) connectats mitjançant xarxes d'àrea local (LAN), com ara Ethernet o Token Ring. Aquestes LAN, al seu torn, se solen connectar entre si i amb altres organitzacions mitjançant xarxes d'àrea metropolitana (MAN) o d'àrea global (WAN), com l'X.25, Frame Relay o ATM.

En general, es poden arribar a connectar una gran varietat de tipus d'ordinadors, mitjançant diferents protocols de comunicació. Les màquines connectades poden ser tant ordinadors personals o estacions de treball com miniordinadors o grans ordinadors (*mainframes*). Poden estar equipats amb sistemes operatius tan diversos com els de la família Windows, UNIX, OS/2, OS/400 o MVS, i connectats per protocols de xarxa com ara NetBIOS, TCP/IP, SPX/IPX o SNA.

Els ordinadors connectats mitjançant una xarxa es poden diferenciar en les dues classes següents:

1) Els ordinadors que ofereixen serveis comuns, coneguts com a **servidors**. Els serveis que ofereixen poden ser tan variats com l'accés a fitxers compartits, el correu electrònic, la utilització d'impressores, la interconnexió amb altres xarxes, etc. Els anomenats *sistemes operatius en xarxa* ofereixen alguns d'aquests serveis, els més bàsics. Els més coneguts són els productes de la família Windows, Novell NetWare i els més propis del món UNIX. Altres serveis, menys bàsics, han de ser proporcionats per eines especialitzades.

2) Els ordinadors que utilitzen directament els usuaris, anomenats **ordinadors personals** o **clients**. Aquests ordinadors tenen una capacitat gràfica molt elevada i es poden usar amb més facilitat i intuïció. Permeten utilitzar, si és necessari,

aplicacions potents d'ofimàtica, com ara processadors de text, fulls de càlcul, eines gràfiques, etc. Aquestes eines permeten millorar la presentació de la informació i usar-la d'una manera més eficaç i eficient.

En particular, les aplicacions que treballen amb bases de dades també han de funcionar en aquest entorn de treball i aprofitar les possibilitats que ofereix:

a) Els usuaris d'una mateixa unitat organitzativa poden utilitzar bases de dades d'àmbit local per a les seves activitats, aprofitant les eines dels ordinadors personals per a treure més rendiment de la informació.

b) Aquests mateixos usuaris també poden compartir informació amb altres unitats organitzatives mitjançant altres bases de dades.

Normalment existiran els tres tipus d'informació següents:

- Informació de caràcter corporatiu: compartida per moltes unitats organitzatives.

- Informació de caràcter local: pròpia de departaments, delegacions o sucursals concretes.

- Informació personal: no compartida.

En aquest context, l'objectiu ha de ser aconseguir construir aplicacions de bases de dades que permetin compartir informació comuna i gestionar de manera flexible informació particular. Naturalment, és necessari atendre al mateix temps els ineludibles aspectes de rendiment, seguretat, integritat de les dades i administració. La construcció d'aquestes aplicacions es basarà, de manera natural, en la utilització de servidors que ofereixin als clients serveis d'emmagatzemament de bases de dades i d'accés a aquestes. Tal com veurem, l'ús de termes simplificadors, com ara *client/servidor* i *bases de dades distribuïdes*, amaga una realitat complexa i diversa. 

Objectius

Els materials didàctics d'aquest mòdul contenen les eines fonamentals perquè l'estudiant assoleixi els objectius següents:

1. Saber què s'entén per bases de dades client/servidor i bases de dades distribuïdes.

2. Conèixer les possibles arquitectures d'accés a bases de dades per mitjà d'una xarxa, d'acord amb les diferents maneres de repartir funcions entre els ordinadors implicats.

3. Identificar i comprendre les diferències que hi ha entre els diferents tipus de clients i servidors.

4. Saber quines diferències hi pot haver entre la programació d'aplicacions de bases de dades completament centralitzades i la d'aplicacions client/servidor i distribuïdes.

5. Tenir presents els mecanismes que ofereixen els sistemes de gestió de bases de dades per a garantir les propietats transaccionals en un marc distribuït.

6. Conèixer les eines disponibles per a construir aplicacions client/servidor i distribuïdes que siguin eficients, a pesar del cost que suposa la transmissió de dades per xarxes de comunicació.

1. Visió general de l'accés a bases de dades per xarxa

Les possibilitats que sorgeixen davant la construcció d'aplicacions d'accés a BD per mitjà de xarxes de comunicació són innombrables. Hi ha moltes maneres de permetre als usuaris accedir a informació emmagatzemada en un o més ordinadors remots. Les claus es troben en les diverses maneres de distribuir la informació i les funcions entre els ordinadors implicats. Per entendre millor aquestes possibilitats primer intentarem fer-ne un breu esbós i, a continuació, examinarem els aspectes crítics que cal tenir presents.

1.1. Formes de distribució

Una aplicació d'accés a bases de dades per mitjà de xarxes de comunicació ha de distribuir les funcions que han d'executar els diferents ordinadors que hi estan connectats. S'ha d'assignar a cada ordinador les responsabilitats necessàries per al funcionament global de l'aplicació. Tot seguit, veurem els diversos aspectes que cal tenir en compte.

1.1.1. Funcions a distribuir

Hi ha, almenys, tres tipus de funcions diferents que s'han de distribuir entre les màquines que intervenen en una aplicació d'accés a bases de dades en una xarxa:

1) L'**emmagatzemament de les dades**. En general, hi pot haver un o més ordinadors amb accés als dispositius físics d'emmagatzemament utilitzats. L'usuari final pot treballar directament contra aquests ordinadors o, de manera més comuna, treballar amb un ordinador propi que es connecta als primers per xarxa.

Accés als dispositius físics d'emmagatzemament

Normalment, cada dispositiu físic d'emmagatzemament està connectat a un únic ordinador, a través del qual es fan tots els accessos. No obstant això, és interessant saber que hi ha mecanismes que permeten accedir a un dispositiu físic des de diversos ordinadors. Aquesta possibilitat pot ser útil per a poder continuar treballant després d'una fallada en un dels ordinadors o per a augmentar la capacitat global de processament.

2) L'**execució del codi**. Això inclou tant el codi de l'aplicació com el codi dels SGBD. Quan l'usuari treballa amb un ordinador propi, almenys una part del codi de l'aplicació s'executa en el seu ordinador (el client), i una altra part s'executa en els servidors de dades. Ambdues parts del codi, per descomptat, s'han de comunicar. No obstant això, podem variar el percentatge del processament que du a terme cada participant i el format i el volum de les dades que s'inter-

canvien. A més, també hi pot haver servidors intermedis que executin altres parts del codi de l'aplicació.

3) L'**emmagatzemament del codi**. Normalment va lligat al lloc en què se n'executa cada part, per la qual cosa haurem de diferenciar els dos casos següents:

El llenguatge de programació Java,...

... inicialment dissenyat per a Internet, ha posat molt en voga la possibilitat de carregar el codi en el client des de qualsevol altre ordinador, sense necessitat d'utilitzar fitxers compartits.

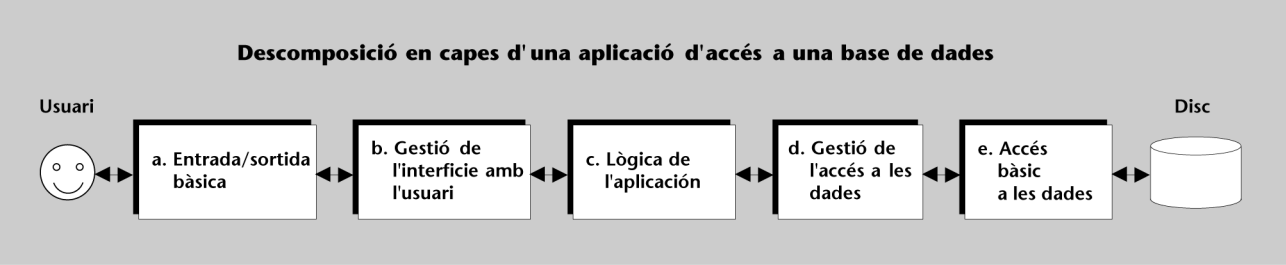
- El **codi executat en els servidors** està emmagatzemat habitualment en aquests, d'una manera o una altra.

- El **codi executat en els clients** de vegades està emmagatzemat en el mateix client, però també ho pot estar en un servidor. Aquest servidor no té per què coincidir amb el servidor de dades. Sol ser un ordinador de la LAN* del client que, simplement, conté fitxers compartits. En executar l'aplicació, el codi disponible en aquests fitxers es carrega en la memòria del client.

*** LAN és la sigla del terme anglès *Local Àrea Network*.**

1.1.2. Distribució del processament

De manera simplificada, una vegada una aplicació d'accés a BD s'està executant, se sol poder descompondre en les capes que il·lustra la figura següent:



El flux de control i la informació corren per aquestes capes en resposta a les peticions de l'usuari. Cada capa tindrà les atribucions següents:

a) L'**entrada/sortida bàsica** és responsable de controlar l'ús dels dispositius d'entrada/sortida (pantalla, teclat, ratolí...), a un nivell molt elemental.

b) La **gestió de la interfície amb l'usuari** és responsable de presentar i captar les dades de l'aplicació, mitjançant una interfície d'usuari adequada.

c) La **part de la lògica de l'aplicació** efectua les actualitzacions i consultes que siguin necessàries d'acord amb les peticions dels usuaris i amb la funcionalitat de l'aplicació.

d) La **gestió de l'accés a les dades** ofereix serveis d'alt nivell per a fer consultes i actualitzacions*.

*** Per exemple, mitjançant instruccions SQL.**

e) L'**accés bàsic a les dades** s'encarrega de llegir i escriure les dades en la memòria externa, d'acord amb el format d'emmagatzemament que tinguin.

Quan una aplicació treballa en xarxa, cal assignar l'execució de totes aquestes funcions als diferents ordinadors que hi estan connectats.

Parlarem d'**aplicacions de BD client/servidor** sempre que una part de l'aplicació s'executi en un ordinador client, usat per l'usuari final, i l'accés a les dades emmagatzemades en la memòria externa tingui lloc en un o més ordinadors servidor, amb els quals es comunica el client, directament o indirectament, mitjançant una xarxa de transmissió de dades.

Màquines client i màquines servidor

És corrent parlar d'ordinadors client i ordinadors servidor. No obstant això, és important que siguem conscients que el que realment hi ha són processos client i processos servidor. Res no impedeix, en principi, que una mateixa màquina executi simultàniament processos que són clients i processos que són servidors. Tot i amb això, és una pràctica habitual tenir unes màquines dedicades exclusivament a executar processos servidor i unes altres només per als usuaris, les quals principalment executen processos client. També és freqüent evitar posar molts serveis en la mateixa màquina, per a no sobrecarregar-la i no córrer el risc que una caiguda de la màquina deixi tots aquests serveis inaccessibles.

En el cas més senzill, quan només hi ha un ordinador client (el de l'usuari) i un servidor (el que emmagatzema les dades), en algun punt s'ha d'establir la utilització de la xarxa. D'acord amb la decisió que es prengui, el tipus d'informació que viatjarà per la xarxa serà diferent. Les possibilitats que hi ha, on la barra indica la posició de la xarxa, són almenys, les següents:

1) a/bcde. El client s'encarrega només de l'accés als dispositius d'interacció amb l'usuari. En el servidor s'executa la resta de l'aplicació*. Mitjançant la xarxa, el client i el servidor es comuniquen esdeveniments d'entrada/sortida, com ara pulsacions de tecles o peticions de visualització en pantalla.

*** Per exemple, les aplicacions basades en X-windows poden treballar d'aquesta manera.**

2) ab/cde. En el client es gestionen tots els aspectes d'interacció amb l'usuari. En el servidor s'executa la lògica de l'aplicació i l'accés a les dades. Aquesta pot ser la manera de treballar utilitzant eines com ara procediments de crida remota (RPC), monitors transaccionals* i procediments emmagatzemats en la BD. Per mitjà de la xarxa s'envien peticions d'execució de procediments o programes en el servidor i es reben les respostes corresponents.

*** Per exemple, CICS i Tuxedo.**

Accés a bases de dades en grans ordinadors

L'accés a les bases de dades emmagatzemades en els grans servidors, com els utilitzats per les organitzacions que tracten dades crítiques i abundants, es du a terme quasi sempre per mitjà de monitors transaccionals, no directament. Els monitors transaccionals permeten que, de manera fiable i eficient, les peticions dels usuaris siguin ateses principalment per programes que, gestionats pel monitor de transaccions, s'executen en els servidors (no en la màquina de l'usuari). Aquesta manera de treballar encaixa amb l'enfocament ab/cde.

3) abc/de. En el client es gestionen tots els aspectes d'interfície amb l'usuari i també la lògica de l'aplicació. El servidor atén peticions del client per a fer consultes i actualitzacions d'alt nivell. Aquest és l'esquema de treball de la majoria d'SGBD relacionals. A través de la xarxa viatgen peticions d'execució de sentències SQL i les respostes corresponents.

4) abcd/e. El client controla quasi tots els aspectes de l'aplicació, a excepció de l'accés físic a les dades emmagatzemades. El trànsit de xarxa el constitueixen, principalment, les peticions de lectura i escriptura de pàgines de disc.

La majoria d'aplicacions client/servidor encaixen bastant bé en alguna d'aquestes arquitectures d'aplicació. No obstant això, la realitat pot ser més complexa i es poden donar les situacions següents:

- En certs casos, la posició de la xarxa no separa tan nítidament el tipus de funcions que hem vist. De vegades, per exemple, una part de la lògica de l'aplicació es du a terme en el client, i l'altra, en el servidor.
- La comunicació per xarxa pot ser transparent al codi d'aplicació o no ser ho. Serà transparent si l'aplicació es construeix sobre un programari base que, configurat adequadament, s'encarrega de gestionar la comunicació per xarxa i amagar-la a l'aplicació.
- L'aplicació pot haver d'accedir a diversos servidors de dades, en lloc de fer-ho a un de sol.

Vegeu la distribució de dades al subapartat 1.1.3 d'aquest mòdul didàctic.

- De vegades hi ha servidors intermedis, que poden acomplir funcions pròpies o servir de pont per a la comunicació amb altres servidors.

Queda completament fora de l'àmbit d'aquest mòdul didàctic tractar totes les arquitectures possibles. Ens centrarem en arquitectures del tipus abc/de o abcd/e en què el codi d'aplicació sigui similar al d'una arquitectura centralitzada i l'SGBD amagui les comunicacions per mitjà de la xarxa, tal vegada amb la col·laboració d'un sistema operatiu en xarxa per a accedir als fitxers físics.

Vegeu les arquitectures del tipus abc/de amb més profunditat al subapartat 2.1 d'aquest mòdul didàctic.

1.1.3. Distribució de les dades

Així com el concepte de *client/servidor* està relacionat amb la distribució del processament entre els ordinadors implicats en l'execució de l'aplicació, el concepte de *base de dades distribuïda* està lligat a la distribució de les dades emmagatzemades.

Parlarem de **bases de dades distribuïdes**, en sentit ampli, sempre que les dades emmagatzemades a què accedeix una aplicació es trobin en més d'un ordinador. És l'oposat a la utilització de bases de dades centralitzades, que emmagatzemen en un mateix ordinador totes les dades necessàries per a una aplicació.

Fixeu-vos que el concepte de *distribució* és independent del de *client/servidor*, de manera que es poden produir les dues situacions següents:

- Una aplicació client/servidor pot no ser distribuïda, si les dades s'emmagatzemen en un únic servidor.

- Una aplicació distribuïda, tot i que cada vegada és menys corrent, pot no ser client/servidor, si els usuaris treballen directament amb els ordinadors que emmagatzemen les dades (amb terminals o sense).

No obstant això, aquesta definició del concepte de *BD distribuïdes* és probablement massa àmplia, ja que amaga una realitat molt diversa. En el pitjor dels casos, les aplicacions accedeixen a cada una de les BD de manera completament independent i s'encarreguen, sense cap tipus d'ajuda, de gestionar totes les relacions que hi ha entre elles. En el cas oposat, les aplicacions accedeixen a les dades exactament igual que si fossin en una única BD centralitzada. Així, la distribució de les dades es converteix, en certa manera, en un aspecte més del disseny físic de la BD. Se sol parlar d'**accés integrat** per a referir-se a aquesta situació ideal. Per a aconseguir-ho és necessària una col·laboració suficientment estreta entre els SGBD implicats.

Vegeu els tipus d'SGBD a l'apartat 3 d'aquest mòdul didàctic.

D'acord amb el grau de cooperació que s'assoleixi, els **tipus d'SGBD distribuïts** que se solen distingir són els següents:

- Els **SGBD distribuïts purs** o **fortament acoblats** són capaços de cooperar entre si per a oferir a les aplicacions un accés integrat a les dades.

- Els **SGBD federats** o **lleument acoblats**, per dificultats d'interoperabilitat, no poden oferir un accés totalment integrat a les dades.

La frontera entre els dos tipus de sistemes distribuïts no està massa ben definida. En particular, cal tenir present que la definició no es pot aplicar normalment a un SGBD concret, sinó a un conjunt d'SGBD utilitzats per a una aplicació concreta.

Un SGBD sol ser perfectament capaç de cooperar amb un altre d'idèntic, però no pot treballar amb tanta facilitat o li resulta completament impossible fer-ho amb algun d'un altre fabricant:

a) En general, els sistemes fortament acoblats són **sistemes homogenis**, en què tots els servidors de dades utilitzen el mateix SGBD o, en el pitjor dels casos, diferents SGBD del mateix fabricant.

b) En canvi, els **sistemes heterogenis**, en què els SGBD que s'utilitzen són diferents, solen ser més difícils d'acoblar.

L'origen dels diferents tipus d'SGBD distribuïts és, en cada cas, el següent:

1) Els sistemes fortament acoblats normalment són el resultat d'un **desenvolupament descendent***, mitjançant el qual l'aplicació es dissenya de manera distribuïda per decisió expressa dels desenvolupadors, des de zero o bé migrant acuradament una aplicació centralitzada. El disseny, després de determinar el model de dades que cal utilitzar i les funcions de l'aplicació, estableix la distribució més convenient de les dades.

*** En anglès, *top-down*.**

Els avantatges que s'esperen obtenir de la distribució de dades, partint del desig d'adaptar-se a l'estructura de les organitzacions, són la millora del rendiment, la fiabilitat i la possibilitat de compartir les mateixes dades entre usuaris d'àmbits més grans que els d'una aplicació centralitzada. També hi ha, però, inconvenients com ara l'augment de la complexitat de l'aplicació, més riscos de seguretat, l'increment de les tasques administratives, etc.

2) Per contra, els sistemes lleument acoblats solen sorgir arran d'un **desenvolupament ascendent***, mitjançant el qual es construeix una aplicació sobre un conjunt preexistent de BD. Cada una d'aquestes BD s'utilitza per aplicacions pròpies, implementades amb eines probablement distintes. Aquesta situació és d'allò més comuna. És freqüent que en una organització sorgeixi la necessitat de dur a terme funcions que comparteixin dades que es gestionaven per separat. Les dificultats d'interoperabilitat entre els sistemes utilitzats sol comportar importants limitacions en l'accés a les dades.

*** En anglès, *bottom-up*.**

1.1.4. Nivells de distribució

D'acord amb el que hem vist en els subapartats anteriors, generalment se solen distingir quatre nivells de distribució d'aplicacions de BD. Els nivells es diferencien entre si segons el tipus de transaccions que poden suportar:

a) En el **nivell de petició remota** cada petició efectuada al servidor de dades, per mitjà de la xarxa, es tracta com una única transacció.

b) Treballant amb el **nivell d'unitat de treball remota** és possible enviar al servidor diverses sentències, que es tracten com una única transacció. Es pot parlar també de *transacció remota*.

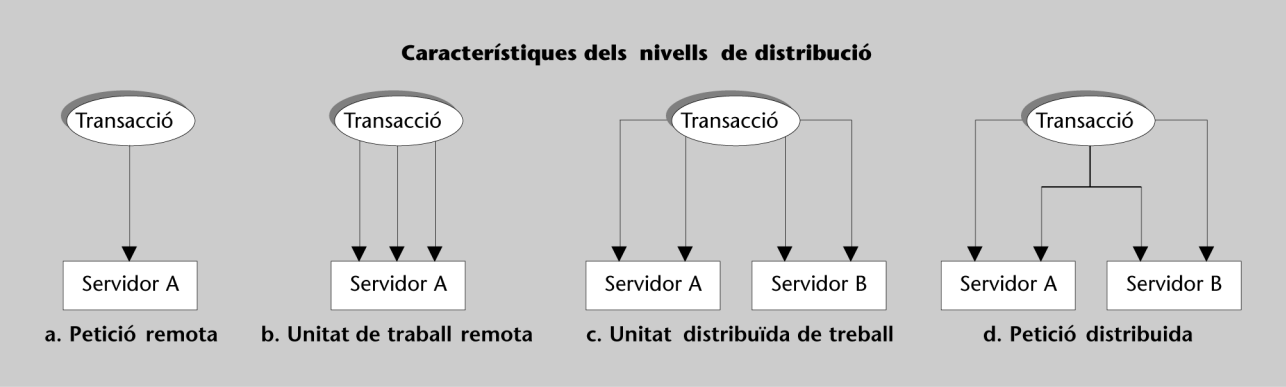
c) El **nivell d'unitat distribuïda de treball** permet accedir a dos o més servidors diferents en la mateixa transacció, tot i que cada petició s'ha d'executar exclusivament en un únic servidor. Es pot parlar també de *transacció distribuïda*. Això implica principalment, quan es duen a terme actualitzacions, la necessitat que tots els servidors implicats en una transacció es coordinin per a garantir-ne l'atomicitat.

Vegeu amb més detall com es garanteix l'atomicitat al subapartat 3.4 d'aquest mòdul didàctic.

d) En el **nivell de petició distribuïda**, una mateixa sentència pot ser executada alhora per diversos servidors, que han de col·laborar per a obtenir el resultat final. Per exemple, això és necessari per a poder executar una sentència `SELECT` d'SQL que combini dades de taules que es troben en servidors diferents.

Vegeu la utilització de taules ubicades a diversos servidors al subapartat 3.5 d'aquest mòdul didàctic.

La figura que presentem a continuació il·lustra les característiques de cada nivell de distribució:



Al llarg d'aquest mòdul didàctic veurem els mecanismes necessaris per a suportar cada un d'aquests nivells. No obstant això, heu de tenir en compte que la definició dels nivells de distribució, tot i que il·lustrativa, és una mica simplista. Alguns sistemes no encaixen correctament en cap dels nivells. Per exemple, és possible permetre executar consultes sobre taules de diversos servidors encara que no es pugui garantir l'atomicitat de les transaccions sobre diversos servidors.

Nivells pròpiament distribuïts

Els dos primers nivells, de petició remota i d'unitat de treball remota, no són pròpiament distribuïts, d'acord amb el que hem comentat al subapartat anterior. Corresponen a arquitectures client/servidor sense distribució de dades.

1.2. Aspectes crítics

En la construcció d'aplicacions en un entorn com el que hem examinat en els subapartats anteriors, cal considerar els aspectes crítics que descrivim a continuació.

1) Impacte del trànsit de xarxa en el rendiment

La distribució de funcions ha de tenir en compte el tipus de xarxa que s'utilitza. Cada tipus de xarxa té característiques diferents:

- Les LAN solen permetre la transmissió de dades a gran velocitat de manera fiable.
- Les WAN* solen ser molt menys ràpides i menys fiables.

*** WAN és la sigla del terme anglès *Wide Area Network*.**

Naturalment, la capacitat de la xarxa depèn, dintre el ventall de possibilitats que hi ha, del desemborsament econòmic que s'estigui disposat a efectuar. Actualment, és considerablement costós utilitzar WAN ràpides, mentre que les

LAN són un producte molt més assequible. Això comporta el fet que és molt més ràpid accedir als servidors de dades situats a la mateixa LAN que a aquells a què s'ha d'accedir mitjançant WAN. Les LAN solen enllaçar els ordinadors d'un mateix edifici o d'edificis contigus, mentre que per a comunicarse amb ordinadors més allunyats cal usar WAN.

L'accés a les dades per mitjà d'una xarxa es pot convertir en un coll d'ampolla per al rendiment general, sobretot si es treballa amb WAN, però també si es treballa amb LAN. La velocitat de transmissió de dades per xarxa acostuma a ser molt més baixa que la d'accés a dades en disc o la de processament, tot i que en el cas de les LAN la diferència és inferior. A més, atès que les xarxes normalment són compartides per usuaris i aplicacions, la seva capacitat s'ha de repartir.

Per tant, és fonamental tenir molt present el trànsit de xarxa que es produeix, d'acord amb la localització de les dades, la quantitat de missatges que s'intercanvien i les dimensions d'aquests.

2) Gestió de transaccions

Una aplicació distribuïda pot tenir els mateixos requeriments de gestió de transaccions que una aplicació centralitzada, tant respecte al control de concurrència com a la recuperació. En particular, és necessari evitar que fallades en les comunicacions o caigudes de servidors concrets puguin violar l'atomicitat i definitivitat de les transaccions.

3) Accés integrat a les dades

Perquè el desenvolupament i el manteniment de les aplicacions distribuïdes siguin còmodes i flexibles, és molt convenient que l'accés a dades distribuïdes s'assembli tan com sigui possible al de les dades centralitzades. Això, més enllà de la gestió de transaccions, afecta el fet de poder accedir a les taules, les vistes, els procediments, etc., de la manera habitual, sense que la seva localització física n'afecti la programació i l'ús.

Vegeu les possibilitats d'accés a les dades distribuïdes al subapartat 3.3 d'aquest mòdul didàctic.

4) Interoperabilitat

Hi ha dos problemes bàsics d'interoperabilitat:

- D'una banda, tal com ja hem comentat, l'accés integrat a BD distribuïdes requereix la cooperació dels SGBD que hi estiguin implicats.

- De l'altra, moltes vegades és important que un mateix programa client pugui treballar amb diversos SGBD diferents, sense cap canvi o amb els mínims possibles.

Queda completament fora de l'àmbit d'aquest mòdul didàctic aprofundir en aquesta qüestió. Només cal que sapigueu que les solucions passen sempre per l'establiment d'interfícies de programació estàndard, la utilització de mecanismes de comunicació estàndard (entre servidors, o entre clients i servidors) o l'ús de passarel·les més o menys complexes entre sistemes diferents. La funcionalitat que s'obté sol ser el resultat de la intersecció de les funcionalitats dels sistemes interconnectats.

5) Seguretat i integritat de les dades

El control i l'administració de les aplicacions en un entorn client/servidor i/o distribuït sempre són més complexos que en un entorn centralitzat clàssic. Independentment dels problemes d'interoperabilitat i de gestió de transaccions que hem comentat anteriorment, l'accés a les dades des de tots els ordinadors client és molt més difícil de controlar, bàsicament pels dos aspectes següents:

- És freqüent que els usuaris vulguin tenir llibertat per a construir o encarregar aplicacions pròpies, que els permetin resoldre amb agilitat situacions particulars. El comportament d'aquestes aplicacions no sempre és correcte.

- Els usuaris també solen disposar d'eines que permetin un accés interactiu i fàcil a les dades, cosa que també suposa un risc per a la integritat de les dades.

Hi ha d'haver eines perquè, sense imposar excessives limitacions als usuaris, es pugui controlar la integritat de les dades. Això es pot aconseguir usant restriccions d'integritat, disparadors, rols, vistes i procediments emmagatzemats.

Vegeu els mètodes de control de la integritat de les dades al mòdul didàctic "Components lògics d'una base de dades" d'aquesta mateixa assignatura.

6) Disponibilitat de les dades

Un objectiu de les aplicacions distribuïdes és que, en la mesura que sigui possible, les fallades de comunicació i dels servidors no impedeixin treballar als usuaris. En certs casos, es preveuran mecanismes alternatius de treball per quan sorgeixin fallades, per conservar una funcionalitat completa. No obstant això, normalment l'objectiu serà més modest, i es considerarà suficient que les possibles fallades afectin només un nombre limitat de funcions i usuaris.

7) Autonomia local

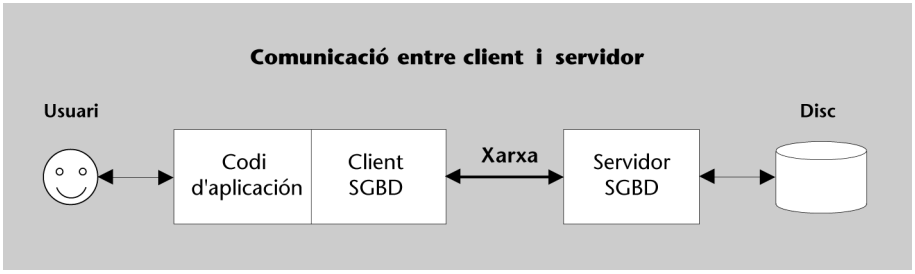
L'àmbit geogràfic i organitzatiu de les aplicacions distribuïdes generalment fa indispensable que els SGBD utilitzats en els servidors es puguin executar i administrar de la manera més autònoma possible, amb el mínim de dependències respecte a altres servidors. Això és necessari, fonamentalment, per motius de disponibilitat i de facilitat d'administració. De totes maneres, perquè sigui possible un accés integrat és necessari renunciar a un cert grau d'autonomia, a fi d'assolir un grau de cooperació suficient.

2. Bases de dades client/servidor

Després d'haver vist què entenem per aplicacions client/servidor, examinem ara més detalladament els tipus de clients i servidors que hi pot haver i les principals maneres de construir-hi aplicacions.

2.1. Clients i servidors

En una aplicació de BD client/servidor, hi ha una part client, que rep les peticions de l'aplicació, i una part servidor, que du a terme l'accés a les dades. Ambdues parts es comuniquen per mitjà de la xarxa, tal com il·lustra la figura següent:



Els dos **tipus bàsics d'arquitectures d'SGBD client/servidor** que hi ha són els següents:

Vegeu els esquemes de distribució de processament al subapartat 1.1.2 d'aquest mòdul didàctic.

1) Les **arquitectures de client lleuger / servidor pesant**, en què la major part de la funcionalitat de l'SGBD és al servidor. Si es treballa amb aquesta arquitectura, el client fa poca cosa més que enviar al servidor les peticions de l'aplicació* i retornar les respostes. La part servidor efectua la major part del processament. Això permet l'existència d'aplicacions que responguin a les arquitectures ab/cde i abc/de. La major part d'SGBD relacionals client/servidor s'ajusten a aquesta arquitectura.

*** Pot enviar les peticions, per exemple, en SQL.**

2) Les **arquitectures de client pesant / servidor lleuger**, en què contràriament al que succeïa en el cas anterior, tot el processament es fa en el client. En un SGBD amb aquesta arquitectura, la part servidor atén bàsicament peticions de lectura i actualització de pàgines de disc. A més, es pot encarregar de qüestions de seguretat, gestió de transaccions, etc. En tot cas, el processament de consultes i l'accés estructurat a les dades queden a les mans del client. Aquest enfocament, que encaixa amb l'esquema de distribució de processament abcd/e, el segueixen alguns SGBD relacionals senzills, pensats per a l'ús personal o de pocs usuaris, i alguns SGBD per a aplicacions amb requeriments especials*.

*** Com ara aplicacions de disseny o sistemes d'informació geogràfics.**

Tipus de servidors lleugers

Els SGBD personals més senzills (com Microsoft Access) utilitzen com a servidor de pàgines de disc els servidors de fitxers que proporcionen els sistemes operatius en xarxa. Això simplifica el producte, cosa que l'abarateix, i en facilita la instal·lació. Per contra, altres SGBD més potents (com ObjectStore o Smallworld) tenen servidors lleugers propis que ofereixen serveis més rics, relacionats normalment amb aspectes de concurrència i recuperació.

No és possible dir que una d'aquestes dues arquitectures sempre és millor que l'altra. La utilització d'un servidor pesant permet disminuir el trànsit de la xarxa, aconseguint que només s'enviï al client informació útil. Això és essencial quan es fan consultes que, per exemple, calculen el salari mitjà de tots els empleats. Un servidor lleuger enviaria les dades de tots els empleats per xarxa, i el salari mitjà es calcularia en el client. Per contra, un servidor pesant només enviaria el salari mitjà*.

*** En resposta, per exemple, a una petició `SELECT` del llenguatge SQL que incorporès la funció `average` (`avg`).**

No obstant això, els servidors lleugers poden ser efectius quan es treballa amb xarxes ràpides i les aplicacions es dediquen bàsicament a navegar per les dades per mitjà d'identificadors d'objecte (referències). En aquestes circumstàncies, i utilitzant mecanismes d'emmagatzenament de pàgines de disc en el client, hi haurà poc trànsit de xarxa inútil i es podrà compensar, fins i tot àmpliament, per la senzillesa i el cost baix de les comunicacions entre el client i el servidor. A més, la màquina servidor pot ser menys potent i es pot aprofitar millor la potència dels ordinadors client.

Transparència de la persistència de les dades

Els SGBD amb servidors lleugers més potents poden ser capaços de tractar les dades persistents, les de la BD, de manera molt similar a les dades volàtils. Per exemple, poden permetre que una referència, d'un registre de BD a un altre registre, es tracti en l'àmbit de la programació exactament igual que un apuntador. Fins i tot solen permetre definir disparadors que s'executin en el client, i amb capacitat per interaccionar amb l'usuari.

En el cas d'SGBD basats en SQL, la interfície de programació del client es pot basar tant en SQL hostatjat com en crides a llibreries de funcions. Totes dues possibilitats s'usen en productes comercials, però les segones s'utilitzen molt més en els entorns client/servidor pels motius següents:

Vegeu els SGBD basats en SQL a l'apartat 4 del mòdul didàctic "El llenguatge SQL" de l'assignatura *Bases de dades I*.

a) La compilació d'un programa d'SQL hostatjat requereix habitualment la presència de la BD que s'utilitzarà, cosa que resulta molesta per al desenvolupament i la instal·lació de les aplicacions. Els programes basats en llibreries de funcions es poden compilar sense connectar-los a cap BD.

b) Normalment, els precompiladors d'SQL hostatjat estan molt lligats a cada SGBD concret. Per tant, utilitzar-los significa lligar-se al fabricant corresponent i a les seves eines de desenvolupament. Per contra, és més fàcil trobar eines de desenvolupament ràpid d'aplicacions capaces de treballar, mitjançant crides a llibreries de funcions estàndard, amb la majoria d'SGBD que hi ha al mercat. Això en facilita el desenvolupament i la portabilitat.

L'ODBC

L'ODBC (*Open Database Connectivity*) és una interfície de programació estàndard, controlada per Microsoft, per a l'accés a BD mitjançant SQL. De fet, és la conjunció d'una llibreria de funcions estàndard i un conjunt de mecanismes que permeten instal·lar i configurar passarel·les d'aquesta interfície de programació estàndard a qualsevol SGBD que hi hagi. Fins i tot, és possible accedir a alguns tipus de fitxers senzills mitjançant SQL, sempre que s'instal·li la passarel·la corresponent (coneguda com a *controlador* en la terminologia d'ODBC).

Una característica essencial d'aquestes tècniques és que la compilació i l'optimització de les sentències SQL es du a terme dinàmicament, en temps d'execució. Això comporta potencialment una disminució de rendiment que, no obstant això, es pot combatre utilitzant procediments emmagatzemats.

Les interfícies de programació per a treballar en un entorn client/servidor solen ser quasi idèntiques a les que s'utilitzen en altres circumstàncies. No obstant això, a vegades disposen d'operacions especialment pensades per a l'accés per mitjà d'una xarxa. Per exemple, és corrent que hi hagi mecanismes per a recuperar múltiples registres de la BD en una única operació `FETCH`, fent aquesta operació mitjançant un únic missatge per xarxa (en lloc d'un per registre). Aquests mecanismes fins i tot poden ser transparents a la programació de l'aplicació.

2.2. Arquitectures

Els SGBD amb servidor pesant permeten dos **tipus d'arquitectures d'aplicació**:

- En les **aplicacions amb arquitectura de dues capes** el codi de l'aplicació del client treballa directament contra les estructures d'emmagatzemament de la BD.
- En les **aplicacions amb arquitectura de tres capes** el client fa peticions d'execució de procediments emmagatzemats en el servidor, i aquests accedeixen a les estructures de dades.

Les aplicacions amb arquitectura de tres capes aprofiten la capacitat de molts SGBD d'emmagatzemar i executar procediments emmagatzemats en el servidor. Aquests procediments emmagatzemats poden rebre paràmetres i retornar resultats, actuant com una capa intermèdia entre el codi en el client i les estructures de dades. La seva utilització està perfectament integrada amb la resta dels serveis de l'SQL. De fet, res no impedeix combinar l'accés directe a estructures de dades amb crides a procediments emmagatzemats. Per tant, una aplicació pot optar per una arquitectura híbrida o migrar esglaonadament a una arquitectura de tres capes.

Vegeu els procediments emmagatzemats al mòdul didàctic "Components lògics d'una base de dades" d'aquesta assignatura.

Els avantatges d'usar procediments emmagatzemats són múltiples, d'entre els quals podem destacar els següents:

Actualment,...

... la majoria d'SGBD client/servidor basats en SQL suporten procediments emmagatzemats. A més, aquests procediments formen part de l'estàndard d'SQL.

1) Pot permetre millorar ostensiblement el rendiment pels dos motius que exposem tot seguit:

a) Els procediments emmagatzemats es poden optimitzar estàticament, per la qual cosa no cal fer-ho durant l'execució.

b) No totes les consultes i actualitzacions que fa el procediment han de passar per la xarxa, per la qual cosa se'n redueix el trànsit.

2) Millora la seguretat i protegeix més la integritat de les dades. És possible fer que els usuaris només puguin consultar o actualitzar les dades per mitjà de procediments. D'aquesta manera, no podran fer operacions que no estiguin previstes.

3) El desenvolupament de les aplicacions pot aprofitar les possibilitats de modularització i reutilització de codi que ofereixen els procediments emmagatzemats.

4) Facilita la realització de canvis en l'aplicació. És possible canviar el comportament d'un procediment emmagatzemat en la BD sense haver de modificar els clients que el criden.

Els servidors capaços d'executar procediments emmagatzemats solen estar optimitzats per a suportar el màxim nombre d'usuaris possible. Per a això fan servir tècniques com ara la utilització de caus amb el codi dels procediments que s'han executat més recentment, l'aprofitament de les possibilitats de paral·lelisme del maquinari, una gestió òptima de processos, etc.

Processaments de transaccions lleugers i pesants

Se sol comparar la utilització de procediments emmagatzemats amb l'ús de monitors transaccionals. Els monitors transaccionals també executen programes en el servidor, però no estan emmagatzemats en la BD. Com que els monitors transaccionals estan pensats per a donar suport a un nombre més elevat d'usuaris, se'ls sol identificar amb el nom de *processament de transaccions pesant**, i es reserva el terme *processament de transaccions lleuger*** per a l'ús de procediments emmagatzemats.

*** En anglès, *TP-heavy*.**
**** En anglès, *TP-lite*.**

3. Bases de dades distribuïdes

A continuació veurem quin suport poden oferir els SGBD per a construir aplicacions que accedeixin a dades emmagatzemades en diversos servidors. Primer repassarem les condicions que, idealment, haurien de satisfer els SGBD que treballen amb BD distribuïdes. Tot seguit, farem un recorregut pels principals mecanismes concebuts amb aquest propòsit, i n'identificarem les limitacions de cada un.

3.1. Regles fonamentals de treball

Hi ha un cert consens pel que fa a les propietats que ha de satisfer un SGBD que permeti treballar amb dades distribuïdes.

Aquest consens es recull a les dotze regles fonamentals següents:

1) Màxima autonomia local: cada un dels servidors ha de tenir la màxima autonomia quant a seguretat, integritat de les dades, optimització de consultes, gestió de transaccions, etc.

2) Igualtat entre servidors: no hi ha d'haver cap servidor central que s'encarregui de funcions essencials, de les quals depengui de manera general el funcionament d'altres servidors.

3) Operació continuada: els servidors han d'estar disponibles; és a dir, les aplicacions hi han de poder accedir, encara que altres servidors hagin caigut o no sigui possible comunicar-s'hi.

4) Transparència de localització: les aplicacions han de poder accedir tant a les estructures de dades com a altres elements de la BD sense necessitat d'especificar-ne la localització.

5) Transparència de fragmentació: les aplicacions han de poder ignorar la fragmentació física que, per motius de rendiment, presentin les dades que des d'un punt de vista lògic són similars.

Vegeu la fragmentació de dades al subapartat 3.6.1 d'aquest mòdul didàctic.

6) Transparència de reproducció: les aplicacions han de funcionar de la mateixa manera amb independència que hi hagi o no dades reproduïdes per a millorar el rendiment.

Vegeu la reproducció de dades al subapartat 3.6.2 d'aquest mòdul didàctic.

7) Processament distribuït de consultes: l'SGBD ha de ser capaç de processar i optimitzar consultes sobre dades de més d'un servidor, tenint en compte la localització de les dades, l'existència de fragmentació i/o de reproducció, i el cost del trànsit de xarxa.

Vegeu la definició d'ACID al subapartat 1.2 del mòdul didàctic "Transaccions a les bases de dades".

8) Gestió distribuïda de transaccions: les transaccions han de complir les propietats ACID a pesar de la distribució.

Vegeu com es poden acomplir les propietats ACID al subapartat 3.4 d'aquest mòdul didàctic.

9) Independència del maquinari: el funcionament conjunt dels servidors no s'ha de veure afectat per les diferències de maquinari.

10) Independència del sistema operatiu: la utilització de sistemes operatius diferents en els servidors no ha ser causa de limitacions de funcionament.

11) Independència dels protocols de xarxa: la col·laboració entre diversos servidors no s'ha de restringir per l'existència de diferents protocols de comunicació per xarxa*.

*** Protocols com ara NetBIOS, TCP/IP, etc., s'han de poder fer servir sense problemes.**

12) Independència de l'SGBD: la cooperació entre servidors ha de ser possible encara que hi hagi nodes que treballin amb SGBD diferents, tant si són del mateix fabricant com de fabricants diferents.

Aquestes dotze regles es poden resumir en dues idees bàsiques:

- Un conjunt de BD distribuïdes s'hauria de veure, en l'àmbit conceptual i extern, com una BD centralitzada. L'existència de la xarxa i de diversos servidors només hauria de ser visible per als dissenyadors o administradors del sistema, en l'àmbit intern.
- És desitjable evitar limitacions no ineludibles a la feina de les aplicacions a causa de la caiguda d'un servidor o de fallades en les comunicacions.

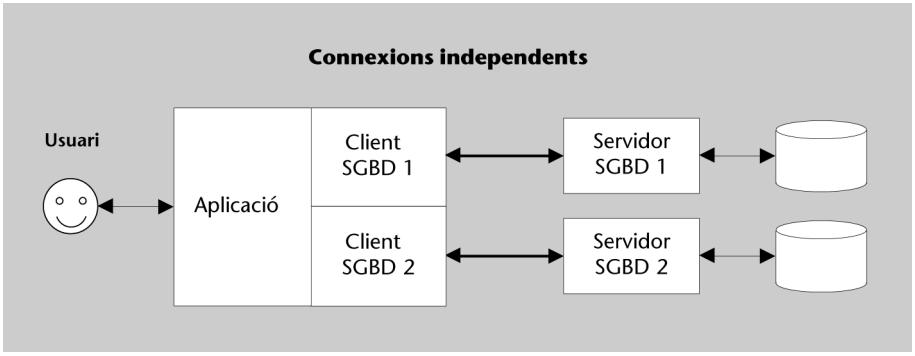
Els SGBD comercials compleixen aquestes regles en graus diversos. Els SGBD fortament acoblats assoleixen un nivell alt de satisfacció. Al llarg dels propers apartats veurem algunes de les coses que cal considerar per a aconseguir acomplir aquests regles. Els sistemes lleument acoblats poden tenir més o menys limitacions.

3.2. Arquitectures

Hi ha diferents arquitectures mitjançant les quals es pot aconseguir oferir un accés més o menys integrat a diversos servidors de BD. Vegem ara les principals alternatives que hi ha i quin nivell d'acoblament es pot esperar de cadascuna d'elles:

1) Connexions independents

La manera més senzilla d'accedir a dades distribuïdes és que l'aplicació estableixi una connexió separada per a cada un dels servidors a què vol accedir.

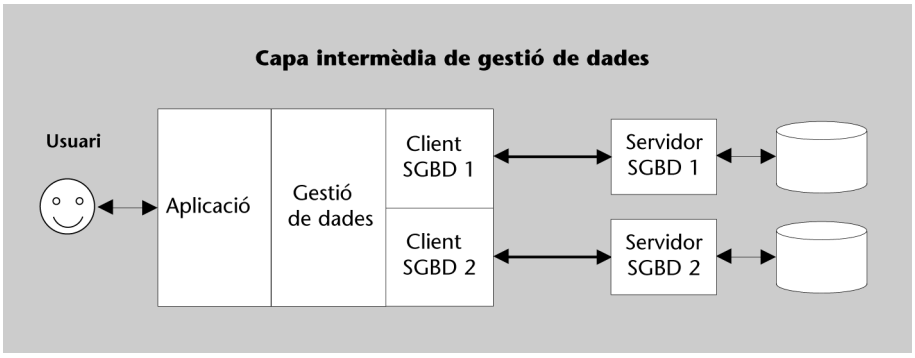


Aquesta arquitectura no permet cap nivell d'integració d'accés, ni peticions ni transaccions distribuïdes. L'aplicació ha de gestionar per si mateixa totes les relacions existents entre les dades que resideixen als diferents servidors. Això comporta dificultats importants de desenvolupament, rendiment i integritat de les dades.

2) Capa intermèdia de gestió de dades

Una manera d'intentar superar les limitacions de l'arquitectura de connexions independents és interposar una capa de gestió de dades entre l'aplicació i cada un dels servidors. Aquesta capa proporciona un accés més o menys integrat a les diverses fonts de dades independents*.

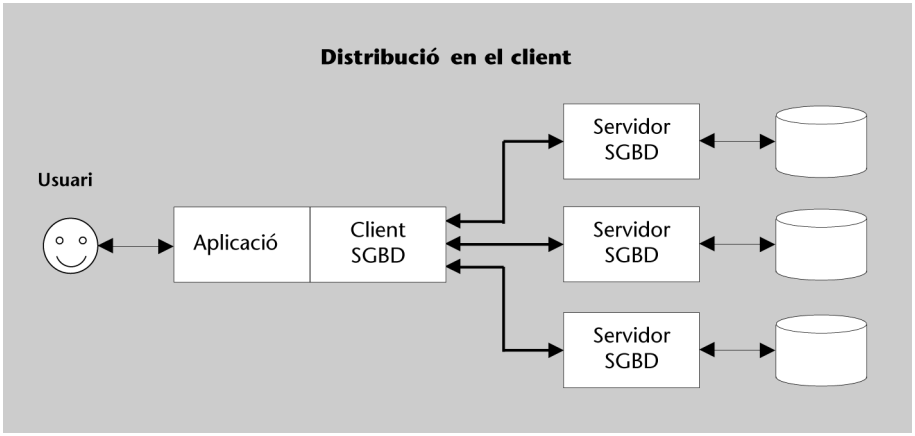
*** Per exemple, pot permetre fer una combinació (*join*) de taules de BD diferents.**



La capa de gestió de dades pot ser un senzill SGBD d'ordinador personal o una eina especialitzada. Pot disposar d'un catàleg propi per a emmagatzemar informació de les BD a què s'accedeix o dels objectes que hi permeten l'accés integrat. En general, aquesta manera de treballar sol ser viable per a aplicacions de consulta amb poques exigències de rendiment i integritat de les dades, però no per a aplicacions d'actualització que siguin crítiques. Normalment, pot suportar peticions distribuïdes, però no transaccions distribuïdes.

3) Distribució en el client

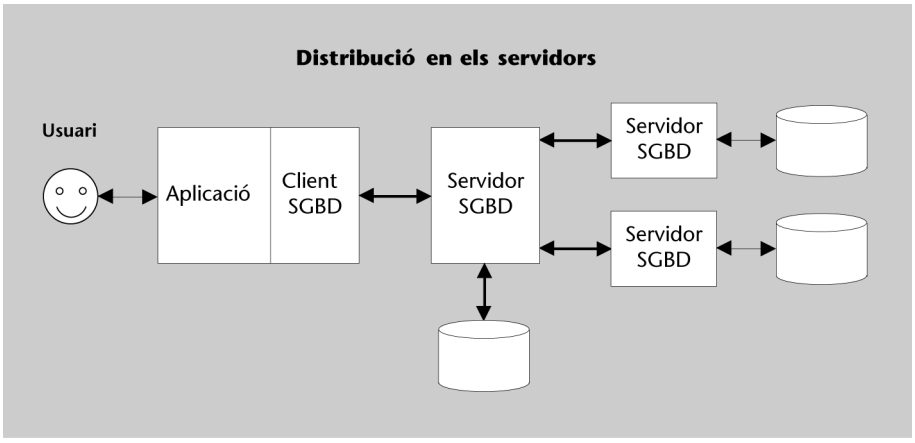
La connexió directa d'un únic client amb diversos servidors permet un accés més integrat que les anteriors, sempre que es disposi dels mecanismes adequats.



Aquesta manera de treballar pot oferir un rendiment acceptable i transaccions distribuïdes. No obstant això, si el client és lleuger s'ha d'accedir normalment a les dades de cada un dels servidors per separat, i sense poder executar peticions distribuïdes. Si el client és pesant, amb capacitat per a realitzar més funcions, podria donar suport a peticions distribuïdes

4) Distribució en el servidor

La solució més comuna, actualment, si es vol assolir el màxim grau d'acoblament entre diversos SGBD, és que el client es connecti a un únic servidor i que aquest permeti accedir a uns altres.



Aquesta arquitectura permet qualsevol nivell d'acoblament, i suporta tant les transaccions com les peticions distribuïdes. El client sol ser lleuger.

3.3. Accés als elements d'una base de dades

La manera que té una aplicació de referenciar els diversos tipus d'elements d'una BD i d'accedir-hi pot variar bastant segons l'arquitectura de distribució de dades que utilitzi. Els elements a què s'ha d'accedir són, fonamentalment, les taules que emmagatzemen les dades, tot i que també es poden efectuar altres ti-

pus d'accés, com ara les crides a procediments emmagatzemats. Idealment, la localització dels elements d'una BD és totalment transparent a les aplicacions.

Segons l'arquitectura de distribució de dades tenim tres possibilitats:

1) Quan es treballa mitjançant una distribució en el servidor, hi sol haver algun mecanisme per mitjà del qual és possible definir enllaços a bases de dades. Aquests enllaços tenen un nom i porten associada tota la informació necessària per a dur a terme la connexió amb l'altre servidor*. Els elements de la BD enllaçada podran ser identificats, a partir d'aquest moment, mitjançant **noms llargs**. Aquests noms llargs estan formats pel nom de l'enllaç a la BD més el nom de l'element en la BD (nom curt). Cada servidor té un catàleg per a descriure els seus propis elements de BD, però no els dels altres servidors, del quals només té la informació associada a l'enllaç.

*** Informació que inclou el protocol de xarxa que s'utilitza, la direcció del servidor, etc.**

Quan un client, connectat a un servidor local, demana l'accés a un element d'un servidor remot, mitjançant un nom llarg, el servidor local estableix la connexió amb el servidor remot i efectua l'accés. Normalment, les connexions amb altres servidors es mantenen obertes mentre no passa més d'un cert temps sense ser utilitzades. Això evita el cost de connectar-se i desconnectar-se constantment.

Aquesta estratègia ofereix un grau d'autonomia local alt, tot i que va en contra de la transparència de localització, ja que els objectes s'han de referenciar en part mitjançant el nom del seu servidor. No obstant això, aquest fet es pot alleujar en certa manera utilitzant definicions de **sinònims** per als nom llargs. En cas que un element es canviï de servidor, n'hi ha prou de canviar la definició del sinònim que s'utilitzi per a accedir-hi.

Sinònim

Suposem que amb un servidor SQL volem accedir a les dades de la taula `empleats` emmagatzemada en un altre servidor que pertany al departament de personal d'una organització.

Per a accedir al servidor podem definir un enllaç de base de dades de la forma següent:

```
CREATE DATABASE LINK deptpers USING <Info.Conexio>;
```

Una vegada definit l'enllaç, ja podem referenciar la taula mitjançant el nom llarg `empleats@deptpers`, i per a consultar totes les files de la taula, executem la sentència següent:

```
SELECT * FROM empleats@deptpers;
```

Per a estalviar-nos haver d'indicar la localització de la taula a totes les aplicacions i protegir-les de possibles canvis, definim el sinònim de la manera següent:

```
CREATE PUBLIC SYNONIM empleats FOR empleats@deptpers;
```

D'aquesta manera, la consulta anterior queda de la manera següent i continuarà essent vàlida en el cas que la taula `empleats` passi a un altre servidor:

```
SELECT * FROM empleats;
```

Cal tenir present que sempre hi ha limitacions respecte a què es pot fer amb els objectes d'altres servidors una vegada s'ha establert la connexió. Per exemple, pot ser que no es puguin definir disparadors sobre taules remotes o que no es puguin definir restriccions d'integritat referencial entre taules de diferents servidors.

Vegeu els disparadors al mòdul didàctic "Components d'emmagatzematge d'una base de dades" d'aquesta assignatura.

2) Quan s'utilitza una arquitectura de capa intermèdia de gestió de dades, els serveis de localització acostumen a ser similars als que acabem de veure, encara que amb més restriccions.

3) Quan un client lleuger es connecta directament als servidors, cada sentència es dirigeix a un únic servidor. Totes les referències a elements de BD que apareguin en aquesta sentència es referiran necessàriament a elements d'aquest servidor. Per tant, l'única cosa que cal fer és indicar a quin servidor s'ha d'enviar cada sentència. Si el client és pesant i suporta peticions distribuïdes, tornem a una situació semblant a les dels casos anteriors.

3.4. Transaccions distribuïdes

Les aplicacions de BD distribuïdes poden tenir els mateixos requeriments transaccionals que les aplicacions centralitzades. Els SGBD de BD distribuïdes han d'estar preparats per a participar en l'execució de transaccions distribuïdes, que no els pertanyin en exclusiva. Una vegada iniciada una transacció en un servidor, aquesta s'ha de poder propagar a altres servidors d'acord amb les peticions de l'aplicació. Per tant, els servidors han d'estar preparats per a treballar amb transaccions d'origen extern. Els missatges que s'envien entre si, per a atendre les peticions de les aplicacions, aniran marcats amb l'identificador de transacció corresponent.

De totes maneres, més enllà de la necessitat d'incorporar informació de gestió de transaccions als missatges entre servidors, és més difícil garantir l'atomicitat i l'aïllament de les transaccions treballant amb BD distribuïdes que amb una BD centralitzada. A continuació veurem les noves dificultats que sorgeixen i com es poden resoldre.

3.4.1. Protocol de confirmació en dues fases

Perquè una transacció abasti més de dos servidors de BD i se'n garanteixi l'atomicitat, és necessari que els servidors siguin capaços de posar-se d'acord a l'hora de determinar si aquesta transacció confirma o no els canvis realitzats. No

seria acceptable que un servidor donés els canvis per confirmats i un altre, a causa d'una caiguda o una fallada de xarxa, els cancel·lés.

El **protocol de confirmació en dues fases** és el mecanisme que segueixen els servidors implicats en una transacció per a garantir que, si una transacció es dóna per confirmada en un dels servidors que hi estan implicats, també es confirmi en tots els altres.

Tal com indica el seu nom, una vegada l'aplicació demana la confirmació de la transacció, el protocol s'executa en dues fases:

1) En la **primera fase**, un dels servidors implicats, el coordinador, pregunta als altres servidors si estan disposats a confirmar la transacció. Aquests servidors podran fer comprovacions de restriccions d'integritat o qualsevol altra tasca necessària. Els servidors que responguin afirmativament s'anotaran de manera segura la decisió. El coordinador decidirà confirmar la transacció si tots els participants responen afirmativament. En cas contrari, decidirà cancel·lar-la.

2) En la **segona fase** es fa efectiva la decisió que s'ha pres. El coordinador envia un missatge a tots els participants per a comunicar-los si han de confirmar o cancel·lar la transacció.

Una optimització senzilla del protocol és oblidar-se, amb vista a la segona fase, dels servidors als quals en la primera fase se sap que només s'han fet accessos de lectura. Aquests servidors es poden oblidar de la transacció en el moment en què responen que no tenen cap problema a l'hora de confirmar. En la segona fase, el coordinador ja no els ha de tenir en compte.

El servidor que actua com a coordinador es pot seleccionar segons diversos criteris. En el cas més senzill, podria ser sempre el primer servidor a què es connecti l'aplicació. L'ideal és que el coordinador sigui un servidor robust. Cal tenir en compte que, si cau durant el procés, la transacció podria quedar activa en la resta de servidors, per la qual cosa bloquejaria els recursos i les dades. Aquests servidors no podrien cancel·lar-la unilateralment, sense comunicar-se amb el coordinador, ja que no tindrien cap garantia que altres servidors no haguessin confirmat ja els canvis. Per consegüent, en situacions com aquestes cal esperar que es recuperi el coordinador o la comunicació que s'hi ha establert.

En general, el protocol és molt sensible a les caigudes dels servidors implicats o a les fallades de xarxa. A més, requereix un nombre de missatges considerable, que pot ser crític en xarxes lentes. Per aquests dos motius, el protocol de confirmació en dues fases només es pot utilitzar quan la velocitat de la xarxa, i la fiabilitat de la xarxa i dels servidors, són suficients.

En el cas contrari, i tal com veurem més endavant en parlar de reproducció, s'ha de renunciar a la protecció total que ofereixen les transaccions i utilitzar diferents tipus de mecanismes basats en cues de missatges. Les cues de missatges serveixen per a acumular en un servidor peticions que s'han d'executar en uns altres. Si és necessari, es pot esperar que aquests servidors estiguin disponibles o que s'hi restableixi la comunicació.

Vegeu la reproducció al subapartat 3.6.2 d'aquest mòdul didàctic.

3.4.2. Detecció distribuïda d'abraçades mortals

Les transaccions distribuïdes tenen els mateixos requisits d'aïllament que les no distribuïdes. El control de concurrència sol fer-se mitjançant reserves, de la mateixa manera que en les BD centralitzades. Una transacció distribuïda obtindrà les reserves necessàries en cada un dels servidors implicats, d'acord amb les dades a què s'accedeixi en cada un. La novetat més remarcable és la possible aparició d'abraçades mortals distribuïdes.

És possible que es produeixi una **abraçada mortal global**, produïda a causa d'un cicle d'espera entre diverses transaccions distribuïdes, sense que es produeixi cap abraçada mortal local, pròpia de només un dels servidors implicats.

Aparició d'una abraçada mortal global

En dos servidors, S1 i S2, s'executen dues transaccions, T1 i T2. En un determinat moment, la transacció T1 demana una reserva en S1, que no se li pot concedir a causa d'una reserva de T2. La transacció T2 continua l'execució i demana una reserva en S2, que no pot obtenir perquè T1 anteriorment ha obtingut una reserva incompatible.

En aquestes circumstàncies, T1 espera T2 en el servidor S1, mentre que T2 espera T1 en el servidor S2. Cap dels dos servidors no pot detectar per si mateix el cicle d'esperes. El cicle és d'àmbit global, no local.

Per a poder detectar abraçades mortals globals, els servidors s'han d'enviar periòdicament informació sobre les relacions d'espera entre les transaccions de què tenen coneixement. Aquesta informació es pot utilitzar per a detectar cicles globals. Això suposa, per descomptat, un grau de coordinació alt entre els servidors, i un cost significatiu en termes de trànsit de xarxa.

Una solució més senzilla i comuna consisteix a permetre només un temps màxim d'espera, després del qual es cancel·len les transaccions. Tot i que es corre el risc de cancel·lar transaccions innecessàriament, pot ser més eficient i molt més fàcil d'implementar.

3.5. Optimització de consultes distribuïdes

Per a suportar peticions distribuïdes, és necessari que l'SGBD sigui capaç de processar sentències que referenciïn dades de més d'un servidor. En particular, un SGBD relacional distribuït hauria de ser capaç de fer combinacions entre taules emmagatzemades en servidors diferents.

Per a això, l'SGBD ha d'incorporar als mecanismes d'execució de les sentències la capacitat de transmetre dades d'uns servidors a uns altres, tenint en compte el cost que comporta fer-ho. Aquest cost pot ser tant de rendiment com econòmic, si el preu d'utilització de la xarxa depèn de la quantitat de dades que es transmetin. Idealment, l'SGBD hauria de poder diferenciar els accessos que es duen a terme mitjançant una LAN ràpida dels que es duen a terme mitjançant una WAN. Segons la velocitat de la xarxa, seria convenient donar més o menys pes relatiu a la minimització del trànsit de xarxa, en comparació amb el que s'atorga als accessos a disc i al temps d'UCP.

Quan la minimització del trànsit per la xarxa resulta crítica, es pot recórrer a tècniques especialment orientades a aquesta finalitat. Una n'és la utilització de semicombinacions.

Una **semicombinació entre dues taules***, A i B, retorna les files d'A que, en cas que es dugui a terme la combinació equivalent entre A i B, apareixerien en el resultat (concatenades amb files de B).

*** El concepte de *semicombinació* es coneix també amb el terme *semi-join*.**

Utilització de semicombinacions per a disminuir l'ús de la xarxa

Suposeu que una taula, A, està emmagatzemada en un servidor SA i que una altra taula, B, està emmagatzemada en un altre servidor SB. Un usuari que es connecta a SA podria demanar el resultat de la consulta següent:

```
SELECT * FROM A, B WHERE A.atr = B.atr;
```

L'SGBD podria optar per diverses estratègies per a processar la consulta:

1) Portar totes les files de A a SB, resoldre la combinació a SB i retornar el resultat a SA, que al seu torn el retornaria a l'usuari. Es podria produir una gran quantitat de trànsit de xarxa: s'enviaria tota la taula A i el resultat de la consulta. El trànsit real dependria del volum d'aquesta informació.

2) Portar totes les files de B a SA, resoldre la combinació a SA i retornar el resultat a l'usuari. Possiblement, es produiria menys trànsit que en el cas anterior.

3) Per a evitar transmetre taules completes per xarxa, es podria intentar utilitzar el concepte de semicombinació. S'enviaria a SB una taula amb tots els valors diferents de la columna *atr* a A (és a dir, la projecció A[*atr*]). Aquesta taula s'utilitzaria a SB, mitjançant una semicombinació, per a calcular el subconjunt de files de B que es combinarien realment amb files de A. El resultat s'enviaria a SA, que faria la combinació i retornaria el resultat final a l'usuari. Tot i que depèn de les dimensions de les taules implicades i dels resultats, aquest algoritme en molts casos permetria reduir el trànsit de xarxa de manera considerable.

4) Una altra opció seria enviar, per a cada fila `F` de A, una consulta al servidor SB per a demanar-li les files de B tals que la columna `atr` valgués `F.atr`. Per a cada fila `F`, SB retornaria zero o més files, que es combinarien amb `F` a SA. Aquesta solució portaria només a SA les files de B indispensables, però potser a costa d'enviar molts missatges a SB que no retornarien informació útil.

Naturalment, l'SGBD hauria de tenir informació estadística sobre el contingut de les taules, per a poder prendre aquestes decisions correctament. Observeu, a més, que les possibilitats podrien ser encara més grans si l'usuari estigués connectat a un tercer servidor, SC, al qual no pertanyessin cap de les taules A i B. En aquest cas, l'SGBD hauria de seleccionar

el servidor per a fer la combinació, i portar les dades necessàries per la xarxa. L'opció més elemental consistiria a portar A i B completes a SC, per a fer la combinació. Es podria intentar evitar costos de transmissió per xarxa demanant als servidors SA o SB que resolguessin la consulta, d'acord amb la millor opció de les anteriors, i que després passessin el resultat a SC.

L'optimització de consultes distribuïdes pot ser considerablement complexa. A més, desafortunadament, sol ser difícil que l'SGBD disposi d'informació adequada sobre la xarxa*. Això té les dues conseqüències següents:

*** Informació sobre la velocitat, el nombre d'usuaris amb què es comparteix, la distribució del trànsit al llarg del temps, etc.**

1) Hi ha SGBD que han optat per no donar suport a les consultes distribuïdes, cosa que obliga les aplicacions a calcular per si mateixes el resultat a partir de consultes sobre cada un dels servidors. Això se sol justificar per la necessitat que els programadors siguin conscients del trànsit de xarxa que generen, davant la possibilitat que, sobretot utilitzant una WAN, el cost sigui inacceptable.

2) Si l'SGBD suporta consultes distribuïdes, el desenvolupador de les consultes haurà de considerar molt acuradament el rendiment de les que s'utilitzin.

3.6. Mecanismes de distribució de dades

Un dels principis bàsics de la construcció d'aplicacions de BD distribuïdes és acostar les dades als usuaris, per a disminuir el trànsit de xarxa. Naturalment, les possibilitats són diferents si es construeix una aplicació des de zero que si es treballa sobre BD preexistents. Ens centrarem en el primer cas, i ignorarem les dificultats addicionals del segon.

Una vegada establert el model de dades de l'aplicació, cal decidir en quins servidors se n'emmagatzemarà cada una de les parts. Per això cal saber a quina informació accedeix cada grup d'usuaris i amb quins altres usuaris l'ha de compartir. També és important considerar aspectes com la fiabilitat dels servidors, la fiabilitat dels enllaços de xarxa o la disponibilitat de personal d'administració de BD*, etc.

*** Per exemple, per a fer còpies de seguretat.**

Una solució senzilla, quan es treballa amb BD relacionals, és assignar l'emmagatzemament de cada taula de BD a un servidor concret. No obstant això, no sempre és la més adequada. A continuació veurem mecanismes de distribució de dades més sofisticats, basats en la fragmentació i la reproducció física de dades.

3.6.1. Fragmentació

Es pot donar el cas que per algun motiu hi hagi conjunts de files d'una taula a les quals els usuaris d'un lloc accedeixin molt més que els d'altres llocs. En

aquestes circumstàncies es pot intentar disminuir el trànsit de xarxa mitjançant la fragmentació horitzontal.

La **fragmentació horitzontal** consisteix a repartir l'emmagatzemament de les files d'una taula entre diversos servidors, col·locant a cada un aquelles a les quals accedeixin més freqüentment els usuaris més propers.

Hi ha d'haver un criteri clar* per a decidir en quin servidor s'emmagatzemarà una fila en inserir-la. Normalment, cal evitar que un canvi de la fila comporti, d'acord amb aquest criteri, un canvi de servidor d'emmagatzemament.

*** Normalment, el criteri que es fa servir és el valor d'una columna.**

Distribució de dades entre les delegacions d'una empresa

Assumint que les dades d'un client d'una empresa són utilitzades quasi sempre en la delegació que l'atén, es poden emmagatzemar les dades d'aquest client en un servidor propi d'aquesta delegació. S'accediria a les dades, per exemple, mitjançant una xarxa local, de manera ràpida i sense haver-se de comunicar amb una seu central que potser és molt lluny.

Altres vegades, les columnes d'una taula només són interessants per a usuaris que treballen en llocs concrets.

La **fragmentació vertical** reparteix l'emmagatzemament de les columnes d'una taula entre diversos servidors, col·locant en cada servidor aquelles columnes a les quals accediran més freqüentment els usuaris propers. Per a poder relacionar les dades d'un servidor amb les d'un altre, és necessari que en tots els fragments es guardi la clau primària.

Distribució de dades entre les diferents seus d'una empresa

Suposem que alguns dels camps dels productes d'una empresa només s'utilitzen per a la fabricació, mentre que d'altres s'utilitzen per a vendes i facturació. Si els grups d'usuaris de les diferents columnes són en llocs apartats, entre els quals la comunicació per xarxa és lenta, podria ser convenient col·locar les dades de fabricació en un servidor (a la fàbrica corresponent), i les dades de venda i facturació en un altre servidor (a les oficines en què hi hagi els usuaris que s'encarreguen d'aquestes tasques).

Transparència de fragmentació

Per desgràcia, la majoria d'SGBD no ofereixen transparència de fragmentació. Això significa que cada un dels fragments s'ha de tractar com una taula distinta, sense cap relació especial amb les taules dels altres fragments. En particular, per a fer una consulta que retorni registres de diversos fragments, caldrà executar la consulta sobre cadascun d'ells. Aquest problema es pot intentar alleugerir amb la utilització de vistes.

Si s'escau, aquests dos tipus diferents de fragmentació es poden combinar, fent fragmentacions horitzontals o verticals ja no de taules, sinó de fragments de taules.

La utilització de la fragmentació pot millorar el rendiment, però és important tenir en compte que pot ser inapropiada si es col·loquen dades en servidors que, per les característiques pròpies o les de la ubicació física que tenen, no ofereixen garanties de seguretat, no es poden administrar amb comoditat, etc. Aquest tipus de consideracions, a vegades, prevalen respecte a les de rendiment, cas en què se sol tendir a fer un emmagatzemament centralitzat, amb reproducció o sense.

3.6.2. Reproduir

La fragmentació permet que les dades siguin a prop dels usuaris que més les utilitzen, però no resulta apropiada quan diversos grups d'usuaris, situats en llocs allunyats, volen accedir de manera prou eficient a les mateixes dades.

La **reproducció** consisteix en l'emmagatzemament de còpies (reproduccions) de taules o de fragments de taules en diversos servidors, de manera que les actualitzacions que s'hi duguin a terme a la còpia d'un servidor es propaguin a totes les altres.

Evidentment, els usuaris han d'accedir sempre a les reproduccions més properes. Fer consultes mai no és un problema, però les actualitzacions requereixen un tractament especial, d'acord amb el qual es distingeixen, almenys, dos tipus de reproducció:

1) Reproducció síncrona. Usant la reproducció síncrona, les actualitzacions que s'efectuen en un servidor es propaguen immediatament a totes les reproduccions que hi ha en altres servidors.

La reproducció síncrona permet que tots els usuaris vegin en cada moment les mateixes dades. En canvi, requereix la utilització del protocol de confirmació en dues fases. Això comporta una certa pèrdua de rendiment i fa impossible dur a terme actualitzacions en un servidor si n'ha caigut un altre. Per consegüent, només s'ha d'utilitzar per a dades poc volàtils, amb servidors suficientment robustos i en entorns de xarxa fiables.

2) Reproducció asíncrona. Al contrari del que succeïa en el cas anterior, utilitzant la reproducció asíncrona, els canvis es propaguen periòdicament, a petició d'un administrador o per altres circumstàncies.

La reproducció asíncrona no requereix usar el protocol de confirmació en dues fases que requeria la reproducció síncrona. És més eficient, i més resistent a fallades dels servidors i de la xarxa. Per contra, té els dos inconvenients següents:

- Obliga a acceptar que alguns grups d'usuaris vegin amb un cert retard les actualitzacions que duen a terme d'altres. Normalment, els usuaris han de ser conscients d'aquest fet.

- Si les actualitzacions s'efectuen en més d'un servidor, hi ha la possibilitat que s'actualitzin les mateixes dades en llocs diferents. L'aplicació ha d'evitar aquesta possibilitat d'alguna manera o hi ha d'haver algun mecanisme per a resoldre les interferències quan es produeixin.

Reproducció en els SGBD comercials

La majoria d'SGBD comercials de cert nivell, tot i que suporten la reproducció síncrona, han centrat l'atenció en mecanismes de reproducció asíncrona potents i flexibles. Permeten fer propagacions incrementals de canvis (de manera consistent amb les transaccions efectuades), distingir reproduccions només de lectura de reproduccions actualitzables, establir criteris de propagació de canvis, seleccionar criteris de resolució d'interferències, etc.

Finalment, val a dir que també podem utilitzar **tècniques de reproducció en l'àmbit d'aplicació**, sense ajuda dels mecanismes que ofereixen els SGBD. Un cas particular d'aquest tipus de tècniques és la **desnormalització**; és a dir, l'existència de redundàncies controlades per l'aplicació.

Utilització de la redundància controlada de camps

En un model normalitzat podríem tenir una taula *CLIENTS*, amb un camp amb el codi postal de cada client, i una taula *CODISPOSTALS*, amb una llista de tots els codis postals i el nom de la població corresponent. En aquest model, per a mostrar el nom de la població del client s'hauria d'accedir a la taula *CODISPOSTALS*. Això podria ser molt ineficient si la taula *CODISPOSTALS* fos en un altre servidor. Per a evitar-ho, es podria introduir una redundància, afegint a la taula *CLIENTS* un camp amb el nom de la població. D'aquesta manera, aquest nom es podria obtenir sense accedir ni a un servidor remot i ni tan sols a una altra taula. Naturalment, si canviés el nom de la població d'un codi postal, caldria revisar els registres dels clients.

3.6.3. Caus persistents

El concepte de *cau** és molt conegut tant en el camp del maquinari, per a accelerar l'accés a la memòria dels ordinadors, com en el de l'accés a fitxers i BD, utilitzant memòries intermèdies per a evitar accessos als dispositius físics. La mateixa idea es pot aplicar a l'àmbit de les BD distribuïdes per a disminuir el trànsit de xarxa.

* La cau es coneix també amb el nom de *caché*.

El **mecanisme de cau persistent** consisteix a emmagatzemar en un servidor una còpia d'una part de les dades emmagatzemades en un altre servidor. La part de les dades que es guarda és la que sembla que té més possibilitats de tornar a ser accedida. Si un client consulta una dada que és a la cau persistent, no serà necessari demanar-la al servidor que té les dades completes.

Les caus persistents optimitzen l'accés per a lectura, però les actualitzacions s'han de continuar fent contra el servidor que emmagatzema totes les dades, que s'anomena *servidor mestre*. A més, ni tan sols no és possible fer consultes quan aquest servidor ha caigut o no és possible comunicar-s'hi. Això és degut al fet que la cau persistent no conté totes les dades i perquè la vigència de les que conté ha de ser validada constantment al servidor mestre, ja que poden ser actualitzades per altres vies.

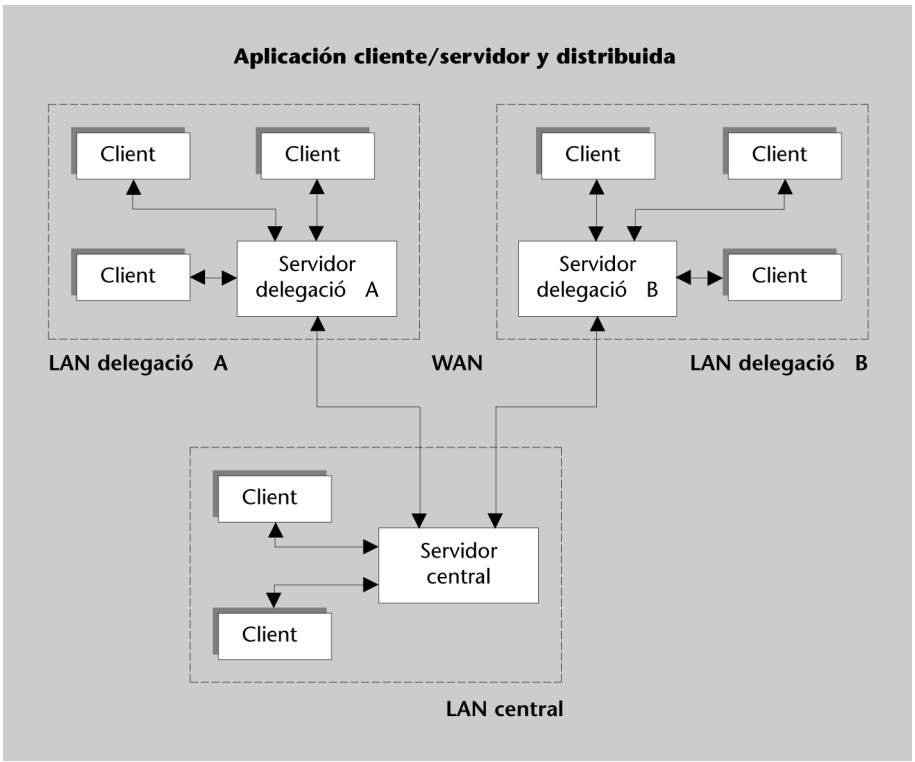
Pocs SGBD suporten la utilització de caus persistents. La majoria, especialment els relacionals, treballen amb arquitectures de servidor pesant i client lleuger, poc adequades per a la implementació eficient d'aquests mecanismes. Els SGBD basats en servidors lleugers sí que les tendeixen a usar.

4. Exemple d'aplicació client/servidor i distribuïda

Per a il·lustrar els conceptes que hem explicat al llarg d'aquest mòdul didàctic vegem en aquest apartat, de manera simplificada, un exemple concret i real d'una aplicació client/servidor i distribuïda. L'aplicació, construïda per a una empresa de distribució d'aigua, s'utilitza per a consultar i actualitzar les dades sobre la xarxa de transport i distribució*.

*** Dades com la situació geogràfica i les característiques de les canonades, les vàlvules que hi ha, els ramals de les finques, etc.**

L'empresa té una seu central i diverses delegacions. En tots aquests llocs es poden fer consultes i actualitzacions. Els ordinadors dels usuaris de la seu central estan connectats, mitjançant una LAN (LAN central), a un servidor central que conté una BD amb tota la informació de l'aplicació. Els usuaris de cada delegació també tenen la seva pròpia LAN i disposen d'un servidor local de BD. A més, les delegacions es poden comunicar amb la seu central mitjançant enllaços relativament lents. La figura següent il·lustra aquesta arquitectura:



L'SGBD que s'utilitza, especialment dissenyat per a aplicacions de gestió d'informació geogràfica, segueix un enfocament de client pesant i servidor lleuger. Per tant, la major part de les funcions es duen a terme en els clients. La informació que viatja per xarxa bàsicament consisteix en pàgines de disc, i les peticions de lectura i actualització corresponents.

Vegeu l'arquitectura client pesant / servidor lleuger al subapartat 2.1 d'aquest mòdul didàctic.

Els servidors locals, els de les delegacions, s'utilitzen per a evitar, en la mesura que sigui possible, la utilització dels enllaços amb la seu central. Perquè els clients puguin accedir a la major part de les dades que necessiten en el seu servidor local, s'ha adoptat l'estratègia següent:

- Hi ha dades que estan completament centralitzades, emmagatzemades només al servidor central. Això es fa, per simplicitat, per a les dades que s'usen molt poc a les delegacions, tant per a lectura com per a escriptura.

- Hi ha dades que s'actualitzen sempre directament a la seu central, però utilitzant caus persistents en els servidors locals. Això es fa amb dades que s'actualitzen poc, però que es consulten bastant*.

*** Per exemple, els catàlegs de materials i diàmetres de les canonades.**

- Les dades que més s'actualitzen a les delegacions també estan emmagatzemades a les caus persistents dels servidors locals, però utilitzant, a més, un mecanisme especial per a emmagatzemar canvis localment, de manera temporal*. D'aquesta manera, no és necessari accedir al servidor central ni per a consultar la major part de dades, que estan emmagatzemades a la memòria cau, ni per a actualitzar-les.

*** Això és el que s'ha fet amb la informació de la xarxa de transport i distribució d'aigua.**

- Durant la nit, es fan processos que propaguen els canvis que han tingut lloc en un servidor a tots els altres servidors. Aquest funcionament és possible perquè els usuaris accepten veure els canvis efectuats en altres delegacions amb aquest retard.

- Atès que totes les dades són al servidor central, independentment que estiguin o no també a les delegacions, n'hi ha prou de fer còpies de seguretat a la seu central, no es fan còpies de seguretat a les delegacions. Això no suposa un gran risc de pèrdua de dades, perquè les còpies de seguretat es fan després d'haver propagat els canvis de les delegacions al servidor central.

La realització dels canvis als servidors locals, que es propaguen asíncronament, introdueix la possibilitat que es produeixin conflictes. No obstant això, l'aplicació ho evita marcant les zones de xarxa que s'actualitzen en una delegació i impedint que s'actualitzin en una altra. Aquesta informació s'actualitza sempre directament al servidor central, ja que tots els usuaris l'han de veure de la mateixa manera.

Aquesta arquitectura suporta la caiguda del servidor local d'una delegació. Si passa això, només es veuen afectats els usuaris d'aquesta delegació. Això és possible gràcies al fet que no s'utilitza reproducció síncrona ni el protocol de confirmació en dues fases. Tot i amb això, la caiguda del servidor central sí que provoca una parada general. Per a reduir aquest risc, aquest servidor central és més robust i fiable que els de les delegacions.

Tot i que aquesta aplicació és un bon exemple d'aplicació client/servidor distribuïda, heu de tenir en compte que pocs SGBD disposen d'un mecanisme de caus persistents o de maneres d'emmagatzemar canvis localment. De fet, algunes particularitats de l'SGBD que s'utilitza faciliten la implementació d'aquests serveis. Si s'hagués utilitzat, per exemple, un SGBD relacional, cosa que no era possible per motius que ara no són rellevants, s'hauria d'haver optat per utilitzar mecanismes de fragmentació i/o reproducció asíncrona.

La fragmentació es podria haver utilitzat per a reproduir en cada servidor local només la informació que, geogràficament, pertanyés a la delegació corresponent. En el 99,9% dels casos, aquesta informació seria aquella a la qual s'accediria en cada delegació. No obstant això, caldria assumir problemes greus de rendiment en consultar informació d'altres delegacions, la qual cosa segurament seria inacceptable. Per tant, seria més adequat reproduir les dades de tot l'àmbit geogràfic. Utilitzant la reproducció asíncrona, els canvis que tinguessin lloc en una delegació no es veurien immediatament en les altres delegacions. En canvi, les actualitzacions serien més eficients i la caiguda d'un servidor local no afectaria els usuaris d'altres delegacions. Naturalment, les dades que s'usessin per a evitar conflictes, que s'haurien de veure exactament en el mateix estat per part de tots els usuaris, estarien emmagatzemades directament al servidor central.

Resum

La combinació de l'ús de les xarxes de comunicació de dades i els SGBD ofereixen la possibilitat que els usuaris d'una organització tinguin accés a informació compartida i personal de manera àgil i eficaç. Els SGBD que s'encarreguen de fer efectiva aquesta possibilitat són els dos tipus següents:

1) Els **SGBD client/servidor** fan possible que diversos usuaris accedeixin des dels seus ordinadors personals, per mitjà d'una xarxa, a dades emmagatzemades en altres ordinadors. Aquests SGBD es poden basar en dos tipus de servidors:

a) Els basats en servidors lleugers s'encarreguen bàsicament de llegir i escriure pàgines de disc, i deixen que el client s'encarregui de la resta de funcionalitats d'accés a dades.

b) Els basats en servidors pesants reben i executen sentències completes d'accés a les bases de dades. Per tant, ofereixen la possibilitat de construir aplicacions amb una arquitectura de tres capes, mitjançant la utilització de procediments emmagatzemats que s'executen al servidor, cosa que redueix el trànsit per xarxa.

2) Els **SGBD distribuïts** fan possible que un usuari accedeixi a dades emmagatzemades en diversos ordinadors comunicats per xarxa. Permeten, segons el grau en què s'entenguin els SGBD implicats, un accés a les dades més o menys semblant al d'un SGBD centralitzat. Idealment, les diferències de plataforma (sistema operatiu, protocols de xarxa, etc.) s'haurien de resoldre sense que n'afectessin el funcionament. La localització dels elements d'una BD pot ser més o menys transparent a les aplicacions, i hi ha més o menys limitacions a l'accés a objectes remots.

Per a poder executar de manera senzilla actualitzacions fiables que afecten diversos servidors, és necessari poder treballar amb transaccions distribuïdes. Les **transaccions distribuïdes** requereixen la utilització del protocol de confirmació en dues fases, per a garantir l'atomicitat dels canvis davant les caigudes de servidors i les fallades de xarxa. A més, quan s'utilitzen reserves, cal evitar d'alguna manera la possibilitat que es produeixin abraçades mortals globals. Per acabar, per a dur a terme consultes sobre dades en diversos servidors, l'SGBD ha de ser capaç d'optimitzar-les i executar-les, tenint en compte la necessitat i el cost de la transmissió de dades per xarxa.

Tot i que la utilització d'SGBD client/servidor i distribuïts aporta beneficis clars, també és cert que augmenta la complexitat de les aplicacions, cosa que en dificulta el desenvolupament i l'administració quotidiana. Sorgeixen així nous factors que cal tenir en compte, com ara l'efecte del trànsit de xarxa, la

interoperabilitat entre clients i servidors de diferents fabricants, les dificultats de seguretat i integritat de les dades davant la presència d'usuaris connectats per xarxa, les fallades dels servidors i de les connexions de xarxa, etc.

Sens dubte, l'impacte del trànsit de xarxa en el rendiment és un dels aspectes crucials, sobretot en usar WAN. Els diferents mecanismes que hem vist per a optimitzar aquest rendiment són els següents:

- La utilització d'arquitectures de tres capes, que pot ajudar en cas d'aplicacions client/servidor.

- L'acostament de les dades als usuaris que hi accedeixen, que és l'objectiu al qual aspirarem en el cas que fem servir BD distribuïdes. Per a aconseguir aquest acostament es pot recórrer a la fragmentació de dades (horitzontal o vertical) i a la reproducció (síncrona o asíncrona), i a l'ús de caus persistents, tot i que cal tenir sempre en compte factors com ara les necessitats d'administració i la fiabilitat de la xarxa que s'utilitzi.

Exercicis d'autoavaluació

1. Quines són les funcions bàsiques d'una aplicació de BD que es poden repartir entre els ordinadors d'una xarxa?

2. Expliqueu què s'entén per BD client/servidor i BD distribuïdes.

3. Quins són els aspectes principals que cal tenir en compte en construir aplicacions de BD client/servidor o distribuïdes?

4. Expliqueu en què es diferencien els distints nivells de distribució d'aplicacions de BD i quins mecanismes són necessaris per a assolir-ne cada un.

5. Indiqueu quines són les dues arquitectures principals d'SGBD client/servidor, segons els tipus de client i servidor que s'utilitzin.

6. Quines particularitats té el desenvolupament de la part client de les aplicacions SQL client/servidor?

7. Quines diferències hi ha entre les aplicacions de BD client/servidor amb arquitectura de dues i de tres capes?

8. Enumereu les regles fonamentals que, en condicions ideals, han de complir els sistemes de BD distribuïdes.

9. Quines són les possibles arquitectures d'accés a BD distribuïdes i quin nivell d'integració d'accés permeten suportar?

10. Descriviu de quines maneres poden les aplicacions de BD distribuïdes referenciar els elements de BD a què s'accedeix, tenint en compte que poden pertànyer a servidors distints.

11. Expliqueu el propòsit, el funcionament i les limitacions del protocol de confirmació en dues fases.

12. Té alguna dificultat especial la detecció d'abraçades mortals en treballar amb BD distribuïdes?

13. Quins nous aspectes s'han de tenir en compte en l'optimització de consultes distribuïdes?

14. Responeu les dues qüestions següents en relació amb les aplicacions distribuïdes:
a) Quines eines es poden utilitzar per a distribuir l'emmagatzemament de les dades?
b) Quins aspectes s'han de tenir en compte per a escollir entre les eines que es poden fer servir per a distribuir l'emmagatzemament de dades?

Solucionari

Exercicis d'autoavaluació

1. Les principals funcions d'una aplicació de BD que s'han de distribuir entre els ordinadors d'una xarxa són l'emmagatzematge de les dades, l'execució del codi (tant el de la matei xa aplicació com el del programari de base que es faci servir) i l'emmagatzematge d'aquest codi. Les decisions adoptades, especialment pel que fa als dos primers aspectes, determinen en gran mesura l'arquitectura que en resultarà.

2. En el camp de les BD parlarem de *client/servidor* per a referir-nos a l'accés a BD mitjançant una part client, executada per l'ordenador de l'usuari, i una o més parts servidor, que atenen les peticions del client i s'encarreguen de l'accés a les dades emmagatzemades.
D'altra banda, cal parlar de BD distribuïdes, en sentit ampli, sempre que les dades emmagatzemades a les quals accedeix una aplicació es troben a més d'un ordinador. És el contrari d'utilitzar les bases de dades centralitzades, que emmagatzemen en un mateix ordinador totes les dades necessàries per a una aplicació.
Així doncs, el concepte *client/servido*r està lligat a la distribució de l'execució del codi, mentre que el de *BD distribuïdes* s'associa a la distribució de l'emmagatzematge de les dades.

3. Els principals aspectes que cal tenir en comte en construir aplicacions client/servidor de BD són els següents: l'impacte del trànsit en el rendiment de la xarxa (sobretot si s'hi accedeix per mitjà d'una WAN), la protecció de les transaccions davant la possibilitat de fallades de xarxa o caigudes de servidors, la integració d'accés que faci possible accedir a les dades distribuïdes de la manera més semblant possible a les centralitzades, la interoperabilitat entre clients i servidors diferents (de diferents proveïdors i fins i tot del mateix), la seguretat i la integritat de les dades, la disponibilitat de l'aplicació i l'autonomia local dels servidors implicats.

4. Els nivells de distribució d'aplicacions de BD, en funció del tipus de transaccions suportades, són els següents: petició remota, transacció remota, transacció distribuïda i petició distribuïda. Els dos primers nivells són client/servidor, però no són pròpiament distribuïts.
- En el primer nivell només és necessari poder transmetre peticions i resultats entre el client i el servidor per mitjà de la xarxa.
- En el segon nivell ja és imprescindible establir una sessió de connexió entre client i servidor que permeti tractar diverses sentències com una unitat de transacció sense perdre el context.
- El nivell de transacció distribuïda requereix la implementació del protocol de confirmació en dues fases i algun mecanisme de detecció d'abraçades mortals globals, de manera que una transacció pugui accedir i actualitzar diversos servidors.
- El nivell de petició distribuïda permet processar consultes adreçades a dades de més d'un servidor mitjançant mecanismes d'optimització distribuïda.

5. Hi ha dos tipus bàsics d'arquitectures d'SGBD client/servidor:
- En les arquitectures de client lleuger / servidor pesant la major part de la funcionalitat de l'SGBD es troba en el servidor.
- En canvi, en les arquitectures de client pesant / servidor lleuger quasi tot el processament s'efectua en el client. Els servidors pesants reben i executen peticions d'alt nivell d'accés a BD (per exemple, peticions SQL). Els servidors lleugers es dediquen principalment a la lectura i escriptura de pàgines del disc: n'envien i en reben el contingut per mitjà de la xarxa.

6. El desenvolupament de la part client de les aplicacions client/servidor que treballen amb SQL pot ser quasi idèntic al de les aplicacions que s'executen amb un únic ordinador. Els SGBD s'encarreguen de gestionar l'ús de la xarxa de la manera més transparent possible. Tanmateix, en les aplicacions client/servidor és molt més habitual accedir a l'SGBD mitjançant crides a llibreries de funcions estàndard que amb SQL hostatjat. Les primeres permeten fer el desenvolupament sense la presència de la BD definitiva i emprar eines de desenvolupament independents de l'SGBD que s'utilitza.

7. Els SGBD basats en arquitectures de client lleuger / servidor pesant, com la majoria dels que treballen amb SQL, permeten executar en el servidor procediments emmagatzemats. D'aquesta manera els clients no accedeixen directament a les dades, sinó que ho fan per mitjà de les crides a procediments. Es parla d'aplicacions amb arquitectura de dues o tres capes per a distingir aquestes dues opcions, encara que res no impedeix combinar-les si es considera convenient. Fent servir procediments emmagatzemats millora el rendiment (gràcies a l'optimització estàtica i a la disminució del tràfic de la xarxa), la seguretat i la integritat de les dades, l'estructura del codi i la capacitat de modificació del codi del servidor.

8. Les regles fonamentals que haurien de complir els SGBD distribuïts són: màxima autonomia local, igualtat entre servidors, operació continuada (màxima disponibilitat), transparència de localització, transparència de fragmentació i de reproducció, processament distribuït

de consultes, gestió distribuïda de transaccions, independència respecte del maquinari, del sistema operatiu, dels protocols de la xarxa i de l'SGBD. L'objectiu general és que un sistema distribuït sigui com més semblant millor a un de centralitzat des del punt de vista de les aplicacions, sense que la possibilitat de fallades de xarxa o de caigudes d'alguns servidors alteri innecessàriament el funcionament.

9. Una aplicació pot accedir a les dades de diversos servidors de BD de quatre maneres bàsiques: amb diverses connexions client/servidor independents, per mitjà d'una capa intermèdia de gestió de dades (que amaga i coordina diverses connexions client/servidor), amb un client que gestiona la connexió de diversos servidors o amb gestió de la distribució en el servidor al qual es connecta el client directament. Les connexions independents no ofereixen cap nivell d'integració d'accés: no suporten ni transaccions ni peticions distribuïdes. Mitjançant una capa intermèdia de gestió de dades es poden suportar peticions distribuïdes, però difícilment se suporten transaccions distribuïdes. Ens trobem en el cas contrari quan treballen amb un client que gestiona les connexions a diversos servidors: aleshores és més fàcil suportar les transaccions distribuïdes que no les peticions distribuïdes, encara que és més fàcil en el cas dels clients pesants. La gestió de la distribució a nivell de servidor és la que fa més fàcil suportar tant transaccions com peticions distribuïdes.

10. Quan es treballa amb una arquitectura de distribució al servidor se sol accedir a altres servidors definint enllaços de BD. Aquests enllaços porten associada la informació necessària per a connectar-se a aquests altres servidors. Els objectes remots es poden referenciar mitjançant noms llargs, que inclouen el nom d'enllaç de la BD corresponent. A fi que la localització dels objectes sigui transparent a les aplicacions es poden definir sinònims. El funcionament pot ser similar quan es treballa amb una arquitectura de capa intermèdia de gestió de dades. En el cas que es treballi amb distribució en el client normalment n'hi ha prou que quedi clar a quin servidor es dirigeix cada sentència, si no se suporten peticions distribuïdes. Si el client és pesant i supera peticions distribuïdes tornem als casos anteriors.

11. L'objectiu del protocol de confirmació en dues fases és que tots els servidors de BD implicats en una transacció distribuïda es posin d'acord quan la confirmen, de manera que ho facin tots o cap.

- En la primera fase, un servidor coordinador pregunta a tots els altres si estan disposats a confirmar. Els que responen afirmativament anoten aquesta resposta i esperen la decisió final mantenint activa la transacció.
- En la segona fase es comunica als servidors si realment han de confirmar o, al contrari, si algun ha refusat de fer-ho, han de cancel·lar.

El protocol garanteix la màxima integritat de les dades, però requereix bastant trànsit de xarxa i és molt sensible a les caigudes de servidors o a les fallades de xarxa.

12. Treballant amb BD distribuïdes és possible que es produeixi una abraçada mortal global, que afecti diversos servidors, però que cap d'aquests servidors no la pugui detectar individualment. Això es pot resoldre mitjançant l'intercanvi d'informació d'esperes entre transaccions o utilitzant temps d'espera màxima, la solució més freqüent.

13. L'optimització de consultes distribuïdes requereix afegir el cost del trànsit de xarxa com un més dels factors que cal tenir en compte. Això comporta tenir en compte el cost de les transmissions i també plantejar-se l'ús de tècniques de processament orientades especialment a disminuir-les (per exemple, mitjançant semicombinacions). En general, els SGBD són bastant limitats en aquest aspecte.

14.

a) Els mecanismes especials que hi ha per a distribuir l'emmagatzematge de les dades són la fragmentació (horitzontal o vertical), la reproducció (síncrona o asíncrona) i les memòries cau persistents.

b) La fragmentació divideix les dades d'una taula o entitat entre diversos servidors. En el cas de la fragmentació horitzontal, les files es guarden en un servidor o en un altre segons un criteri determinat. En el cas de la vertical, es guarden columnes diferents en cada servidor. Totes dues opcions es poden combinar.

La reproducció permet duplicar en un servidor dades que també es troben en un altre servidor. Si és síncrona, les actualitzacions es propaguen immediatament entre les reproduccions, dins la mateixa transacció i utilitzant el protocol de confirmació en dues fases.

Si és asíncrona, la propagació es fa en diferit d'acord amb diferents criteris, sense haver d'utilitzar el protocol de confirmació en dues fases i sense necessitat que tots els servidors implicats estiguin accessibles per a fer l'actualització. En canvi, cal acceptar que alguns usuaris vegin certes actualitzacions amb retard i que es puguin produir actualitzacions contradictòries en reproduccions diferents.

Les cau persistents permeten emmagatzemar en un servidor una part de les dades d'un altre servidor, amb l'esperança que gran part dels accessos es puguin resoldre sense accedir al servidor remot o, almenys, amb una disminució significativa del trànsit de la xarxa. La majoria d'SGBD no ho suporten.

En general, l'objectiu és tenir a cada servidor les dades que necessiten els usuaris que hi poden accedir eficientment. Cal atansar la informació als usuaris que la proporcionen i la consumeixen, encara que també s'han de tenir en compte els requisits de seguretat, la integritat de les dades, les còpies de seguretat, l'administració, etc.

Glossari

abraçada mortal global *f* Abraçada mortal que es produeix en utilitzar reserves sobre dades distribuïdes a causa de cicles d'esperes que abasten diversos servidors.

abraçada mortal local *f* Abraçada mortal que es produeix en utilitzar reserves sobre dades distribuïdes a causa d'un cicle d'esperes produït en un únic servidor.

accés integrat *m* Accés a dades distribuïdes similar al que es pot dur a terme amb dades centralitzades, tenint en compte aspectes com ara la manera de referenciar els elements de BD, la gestió de transaccions i l'existència de consultes sobre dades distribuïdes.

arquitectura de dues capes *f* Arquitectura de les aplicacions, que utilitzant SGBD client/servidor, accedeixen directament a les dades des del client.

arquitectura de tres capes *f* Arquitectura de les aplicacions, que utilitzant SGBD client/servidor, accedeixen a les dades per mitjà de procediments emmagatzemats que s'executen en els servidors.

autonomia *f* Possibilitat, utilitzant SGBD distribuïts, que cada un dels SGBD que hi estan implicats es puguin administrar i executar el més lliurement possible, amb el mínim de dependències respecte a altres servidors.

BD *f* base de dades.

BD centralitzada *f* BD que conté totes les dades necessàries per a una aplicació, de manera que aquesta no hagi d'accedir a altres BD per mitjà d'una xarxa.

BD distribuïdes *f* Conjunt de BD emmagatzemades en ordinadors comunicats per xarxa, que poden ser accedides conjuntament per una o més aplicacions.

BD distribuïdes pures *f* Conjunt de BD distribuïdes a les quals s'accedeix de manera molt integrada, gràcies a l'alt grau d'acoblament dels servidors corresponents.

BD distribuïdes federades *f* Conjunt de BD distribuïdes que no poden assolir un alt nivell d'integració d'accés, a causa del baix nivell d'enteniment entre els servidors que les gestionen.

cau persistent *f* Mecanisme d'emmagatzemament per a BD distribuïdes que permet emmagatzemar en un servidor dades d'un altre servidor a les quals han accedit clients que estaven connectats al primer, amb el propòsit d'optimitzar l'accés per a lectura, evitant demanar al servidor mestre les dades que ja estan a la cau.

client *m* Part de l'aplicació que, en una arquitectura client/servidor, demana al servidor que dugui a terme els serveis. Per extensió, l'ordinador de l'usuari final, que executa bàsicament processos client.

client pesant *m* Part client d'un SGBD client/servidor basat en un servidor pesant.

client lleuger *m* Part client d'un SGBD client/servidor basat en un servidor lleuger.

client/servidor *f* Arquitectura d'accés a BD mitjançant una part client, executada en l'ordinador de l'usuari, i una part servidor, que atén les peticions del client i s'encarrega d'accedir a les dades emmagatzemades.

enllaç de BD *m* Definició, en un servidor de BD, d'una connexió a un altre servidor, que li assigna un nom i proporciona tota la informació necessària per a establir la connexió.

fragmentació horitzontal *f* Mecanisme, en SGBD distribuïts, de particionament físic de les dades. El seu propòsit és col·locar els fragments resultants el més a prop possible dels usuaris que hi accedeixen de manera que cada fragment es queda amb una part de les files d'una taula.

fragmentació vertical *f* Mecanisme, en SGBD distribuïts, de particionament físic de les dades. El seu propòsit és col·locar els fragments resultants el més a prop possible dels usuaris que hi accedeixen, de manera que cada fragment es quedi amb una part de les columnes d'una taula.

interoperabilitat *f* Capacitat de clients i servidors de treballar conjuntament, fins i tot a pesar d'haver estat desenvolupats per fabricants o plataformes diferents.

LAN *f* Xarxa d'àrea local destinada, habitualment, a connectar ordinadors situats en un mateix edifici i que habitualment ofereix una velocitat i una fiabilitat altes.
En.: *Local Area Network*

petició distribuïda *f* Petició d'accés a BD que afecta dades que estan emmagatzemades en diversos servidors.

petició remota *f* Nivell més bàsic d'accés a dades remotes, que permet a un client enviar cada petició a un únic servidor, sense possibilitat d'agrupar diverses peticions en una transacció.

protocol de confirmació en dues fases *m* Protocol mitjançant el qual els servidors implicats en una transacció distribuïda es posen d'acord perquè tots confirmin la transacció o no ho faci cap.

reproducció *f* Mecanisme d'emmagatzemament distribuït de dades que permet tenir en un servidor còpies o reproduccions de les dades que hi ha en un altre servidor amb el propòsit d'augmentar la velocitat d'accés als usuaris propers.

reproducció síncrona *f* Reproducció a la qual les actualitzacions d'una còpia es propaguen immediatament, en la mateixa transacció, a la resta de còpies mitjançant la utilització del protocol de confirmació en dues fases.

reproducció asíncrona *f* Reproducció en la qual els canvis en una còpia es propaguen en diferit, segons diversos criteris, a la resta de còpies.

servidor *m* Part de l'aplicació que, en una arquitectura client/servidor, atén les peticions de serveis dels clients. Per extensió, l'ordinador que, tot i no ser utilitzat per l'usuari final, executa processos servidor que ofereixen serveis comuns.

servidor pesant *m* Servidor que s'encarrega d'una gran part del processament d'una aplicació i deixa menys funcions al client. En l'àmbit de les BD, fa referència als servidors que reben i executen peticions d'alt nivell d'accés a BD.

servidor lleuger *m* Servidor que s'encarrega d'una part petita del processament d'una aplicació i en deixa la major part al client. En l'àmbit de les BD, fa referència als servidors que bàsicament es dediquen a la lectura i escriptura de pàgines de disc, enviant-ne i rebent-ne el contingut mitjançant la xarxa.

SGBD *m* Sistema de gestió de bases de dades.

SGBD fortament acoblats *m* SGBD amb un nivell d'entesa entre els servidors de BD corresponents que permet assolir un alt nivell d'integració d'accés a dades distribuïdes.

SGBD lleument acoblats *m* SGBD amb un nivell d'entesa que no és prou alt per a aconseguir un bon nivell d'accés a les dades distribuïdes que gestionen.

transacció distribuïda *f* Transacció amb peticions de lectura i escriptura que treballen sobre dades distribuïdes, sense deixar de satisfer les propietats ACID. Requereix la utilització del protocol de confirmació en dues fases i, si s'utilitzen reserves, la capacitat de detectar abraçades mortals globals.

unitat remota de treball *f* Transacció executada en un únic servidor remot, com a resposta a les peticions d'un client.

WAN *f* Xarxa de gran abast destinada normalment a connectar ordinadors situats en edificis, ciutats o, fins i tot, països i continents diferents, que sol tenir prestacions baixes, almenys en relació amb el seu preu.
En.: *Wide Area Network*

Bibliografia

Costa, M.; Quer, C. (1996). "Localización de datos para aplicaciones distribuidas". *Novática* (núm. 122, març/abril).

Orfali, R.; Harkey, D. (1994). *Client/Server Survival Guide with OS/2*. Nova York: Van Nostrand Reinhold.