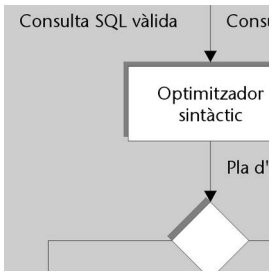
El component de processament de consultes i peticions SQL

Santiago Ortego Carazo

P01/11031/00003

Universitat Oberta de Catalunya



Índex

Introducció

Què fa un SGBD (sistema gestor de bases de dades) des que li plantegem una consulta fins que ens retorna la resposta? Què té en compte per donar-nos la resposta? Què passa si diferents usuaris volem treballar amb la mateixa informació alhora? Aquestes són la mena de preguntes a les quals es vol donar resposta amb aquest mòdul.

Si volem utilitzar un SGBD per a accedir a la informació continguda en una base de dades, el primer que caldrà comprovar és quines autoritzacions tenim sobre aquelles dades; d'això s'encarrega el component de seguretat de l'SGBD. Aquest component cada dia esdevé més important, atès que ara tots els ordinadors estan interconnectats i, per tant, qualsevol persona podria esdevenir usuari d'una base de dades. En moltes organitzacions la informació és un actiu intangible i de naturalesa sensible, i per això cal saber quins són les obligacions legals que tenim.

Les vistes havien estat durant molt de temps un simple mecanisme de simplificació de consultes, però actualment tenen una importància cabdal en diferents àrees: el disseny extern, el gestor de dades (Data Warehouse), la informàtica distribuïda. Els nous mecanismes que incorporen els SGBD comercials que permeten de fer actualitzable qualsevol vista, i els sistemes de gestor de dades que donen la facilitat de controlar l'actualització automàtica de vistes materialitzades són la raó d'aquest èxit.

Com és que alguns sistemes són molt més ràpids que d'altres? Què és el que fa el sistema a partir d'una consulta SQL? Pot ser que una consulta que ahir m'anava molt ràpida avui vagi lenta? A aquestes, i a d'altres preguntes semblants, intentarem de donar resposta, analitzant tot el procés que realitza l'SGBD des que es llença la consulta fins que s'executen les comandes al disc.

Quan diversos usuaris volen treballar alhora amb les mateixes dades, sempre és possible que surtin tota mena d'inconvenients; una funció important de l'SGBD serà assegurar que aquesta concurrència no produeixi problemes.

Tots els temes anteriors són ampliacions i aprofundiments de temes ja vistos en assignatures anteriors.

Objectius

Els materials didàctics associats a aquest mòdul pretenen de facilitar a l'estudiant l'assoliment dels objectius següents:

1. Conscienciar l'estudiant de la importància de la seguretat i donar-li eines per implementar-la.

2. Presentar les vistes com a elements de disseny extern, explicant nous mecanismes que sempre permeten de poder actualitzar-les.

3. Conèixer noves aplicacions de les vistes en els entorns del gestor de dades.

4. Saber quins són els mecanismes de processament i optimització d'una consulta per tal de poder, posteriorment, plantejar-la de la forma més eficient possible.

5. Relacionar les consultes SQL amb els mecanismes de lectura/escriptura en disc.

6. Completar la descripció dels diferents mecanismes de control de la concurrència estudiats fins aquest moment.

7. Presentar els mecanismes de control de concurrència que utilitzen els principals SGBD comercials.

1. La seguretat

En un sistema d'informació, les diferents aplicacions i usuaris de l'organització fan servir un únic conjunt de dades (base de dades corporativa) amb l'SGBD. D'una banda, això resol problemes de redundància, inconsistència i independència entre les dades i els programes i, de l'altra, fa que la seguretat esdevingui un dels problemes més importants en aquests entorns.

La importància de la seguretat

A mesura que els sistemes d'informació s'obren més al món –el paradigma és Internet–, la seguretat esdevé més important.

La paraula seguretat incorpora diferents conceptes. Els més importants són aquests:

a) Confidencialitat: cal protegir l'ús de la informació per part de persones no autoritzades. Això implica que un usuari només ha de poder llegir la informació per a la qual té autorització i que no podrà inferir informació secreta a partir de la informació a la qual té accés.

b) Integritat: la informació s'ha de protegir de modificacions no autoritzades; això inclou tant la inserció de dades falses com la destrucció de dades.

c) Disponibilitat: la informació ha d'estar disponible en el moment que faci falta a l'usuari.

Problemes de disponibilitat

Els problemes de disponibilitat inclouen la seguretat física (per problemes de maquinari), la no-saturació del sistema, la denegació de servei (DoS) per part de la xarxa, etc.

Per a aconseguir seguretat en un entorn de base de dades és necessari identificar les amenaces i triar les polítiques (què s'espera del sistema de seguretat) i els mecanismes (com farà el sistema de seguretat per a assolir els objectius) per a evitar que facin mal.

Una amenaça es pot definir com un agent hostil que, de manera casual o amb una tècnica especialitzada, pot revelar o modificar la informació gestionada pel sistema.

Les violacions de la base de dades consisteixen en lectures, modificacions o esborraments incorrectes de les dades. Les conseqüències d'aquestes violacions es poden agrupar en tres categories:

1) Alliberament incorrecte de la informació: causat per la lectura de dades per part d'usuaris impropis mitjançant un accés intencionat o accidental. En aquesta categoria s'inclouen les violacions del secret derivades de les deduccions d'informació secreta a partir de lectures d'informació autoritzada.

2) Modificació impròpia de les dades: correspon a totes les violacions de la integritat de les dades per tractaments o modificacions fraudulentes d'aques-

tes. Les modificacions impròpies no involucren necessàriament lectures no autoritzades, ja que les dades es poden falsificar sense ser llegides.

3) Denegació de serveis: correspon a accions que puguin impedir que els usuaris accedeixin a les dades o utilitzin recursos.

Les amenaces a la seguretat es poden classificar d'acord amb la manera en què poden ocórrer:

1) Amenaces no fraudulentes: són accidents casuals, entre els quals es poden distingir els següents:

- Desastres accidentals o naturals, com terratrèmols, inundacions o foc; aquests accidents danyen el maquinari del sistema.
- Errors en el maquinari o en el programari, els quals poden conduir a accessos no autoritzats.
- Errors humans, causats de forma no intencionada, en introduir dades o utilitzar aplicacions.

2) Les violacions intencionades o violacions fraudulentes: són causades per dos tipus d'usuaris diferents:

- Usuaris autoritzats, que abusen dels seus privilegis o autoritat.

- Agents hostils, usuaris impropis (interns o externs) que executen accions de vandalisme sobre el programari i/o el maquinari del sistema, o lectures o escriptures de dades; en tots dos casos els usos legals de les dades i les aplicacions poden emmascarar el propòsit fraudulent real.

Com a mecanismes bàsics de seguretat hi ha els següents:

a) Identificació i autenticació: mecanismes que identifiquen l'usuari i s'asseguren que és qui diu que és.

b) Control d'accés: mecanismes que s'asseguren que els usuaris accedeixin només als llocs als quals estan autoritzats amb l'objectiu de fer allò per què tenen permís.

c) Integritat i consistència: mecanismes perquè la base de dades resti sempre en un estat que compleixi totes les regles del negoci del model de dades, encara que es produeixin canvis.

d) Auditoria: mecanismes per a saber qui ha fet què, és a dir, portar un registre de qui fa tots els canvis i consultes a la base de dades. Més que un mecanisme per a donar seguretat, es tracta d'un mecanisme per a detectar el culpable.

1.1. Identificació i autenticació

Els sistemes d'informació i les dades que emmagatzemen i processen són recursos molt valuosos que cal protegir. Un dels primers passos cap a la seguretat en un sistema d'informació és la capacitat de verificar la identitat del usuaris. Aquest procés està format per dues parts: identificació (veure qui és) i autenticació (comprovar que és qui diu que és).

Identificar: veure qui és.

Autenticar: comprovar que és qui diu que és.

La identificació implica la manera en què un usuari proporciona la seva identitat al sistema.

La identitat ha de ser única perquè el sistema la pugui diferenciar entre els diferents usuaris. Segons els requeriments operacionals, una identitat pot descriure un individu, més d'un individu, o un o més individus només una part del temps.

L'autenticació és el procés d'associar un individu amb la seva identitat única, és a dir, la manera en què un individu estableix la validesa de la seva pretesa identitat.

Hi ha tres recursos d'identificació bàsics per a poder demostrar qui és realment un:

1) Una cosa que una persona coneix, com una contrasenya, un número d'identificació personal, etc.
2) Una cosa que un persona posseeix, com una targeta, una clau, etc.
3) Una cosa que caracteritza una persona, com l'empremta dactilar, la veu, etc.

Àmbit de la identitat

Una diferència important entre la identificació i l'autenticació és que les identitats són públiques, mentre que la informació d'autenticació es guarda en secret, i això proporciona el recurs pel qual una persona prova que és realment qui diu que és.

Aquests mètodes bàsics es poden emprar individualment, o es poden combinar per a obtenir un nivell de seguretat més alt.

Les contrasenyes són el mecanisme més clàssic d'autenticació. Les contrasenyes són paraules (o millor dit, combinacions de caràcters) que només coneix un usuari. La seguretat d'un esquema de contrasenyes depèn de la capacitat de mantenir-les en secret; una contrasenya s'ha de triar de manera que sigui fàcil de recordar i difícil d'endevinar.

Criteris per a la selecció de contrasenyes

A continuació presentem alguns criteris que convé tenir en compte alhora de triar una contrasenya:

a) Seleccionar contrasenyes llargues, 8 caràcters és una mida adequada.

b) Combinar diferents tipus de caràcters, majúscules, minúscules, nombres, espais en blanc i caràcters de puntuació.
c) No fer servir paraules amb significat.
d) Utilitzar contrasenyes diferents per a accedir a sistemes diferents.
e) Canviar la contrasenya de manera periòdica, com a mínim una vegada al mes.
f) No escriure la contrasenya en llocs a on un altre pugui accedir.
g) Que no sigui la mateixa que l'identificador d'usuari.
h) Quan es canvia, que difereixi de l'anterior en almenys 3 caràcters.
i) Canviar la contrasenya la primera vegada que s'iniciï una sessió.

Les targetes donen una seguretat més gran. Les tarjetes poden ser simples trossos de plàstic amb una banda magnètica o poden, fins i tot, incorporar xips, com fan les targetes intel·ligents. En tots dos casos, una contrasenya personal ha de coincidir amb la que hi ha escrita a la targeta, o bé la contrasenya més alguna informació de més que hi ha a la targeta han de coincidir amb la que hi ha a l'ordinador.

La tendència actual és anar cap a sistemes biomètrics. Els sistemes biomètrics són mètodes automatitzats de reconeixement d'una persona que es basen en una característica fisiològica o de comportament. Els sistemes biomètrics es poden fer servir com a mètode d'identificació, en què s'identifica una persona dintre una població sencera mitjançant una característica d'aquesta registrada en una base de dades i se'n cerca la coincidència. També es poden utilitzar en mode de verificació: el sistema autentica la identitat reclamada d'una persona amb el seu patró prèviament enregistrat.

Característiques per al reconeixement biomètric

Els trets humans més utilitzats en el reconeixement biomètric inclouen empremtes dactilars, veu, reconeixement facial, retina, iris, signatura escrita, geometria de la mà i les venes del canell.

Per a la validació d'un usuari d'un sistema de gestió de bases de dades es poden fer servir quatre mètodes:

1) Autenticació pel mateix SGBD: és la més utilitzada, ja que els comptes són més fàcils de controlar i gestionar, i el mateix SGBD té recursos que permeten l'administració de petites comunitats d'usuaris.

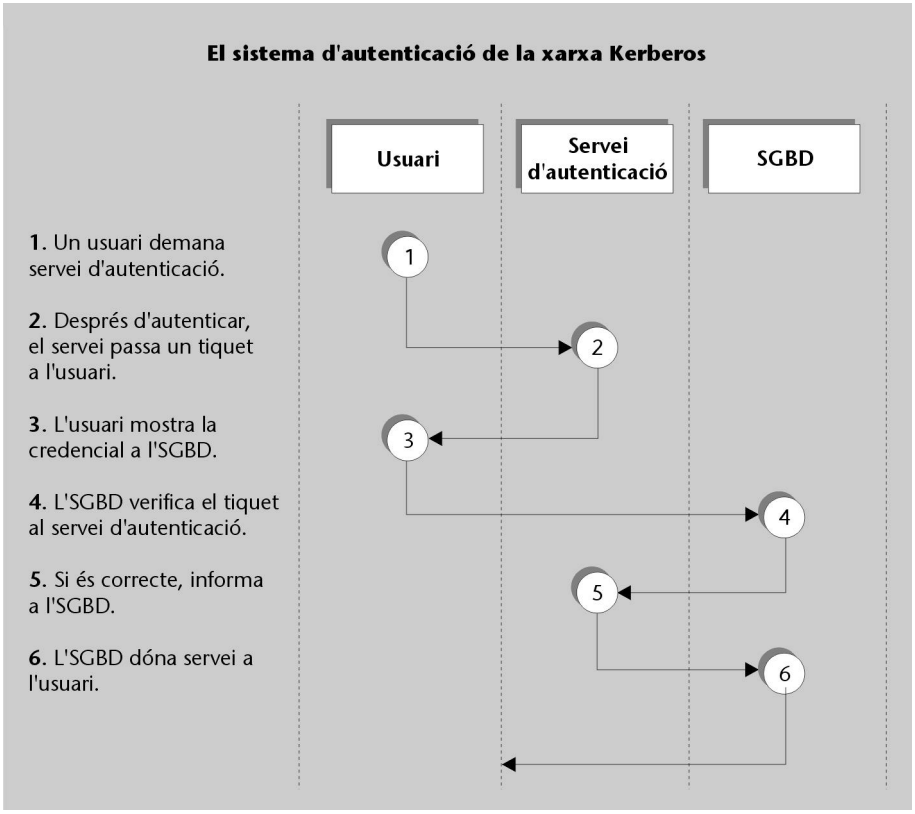
2) Autenticació pel sistema operatiu: és una forma de validació externa; només és possible en aquells sistemes que permetin la validació d'usuaris (UNIX, Windows NT...).

3) Autenticació pel servei de xarxa: fa servir productes especialitzats de xarxa, com Kerberos, CyberSafe, Identix, Radius o altres.

4) Autenticació per una capa intermèdia: en un sistema client/servidor de tres nivells, la capa intermèdia podria ser el programari intermediari* o l'aplicació mateix.

* En anglès, middleware.

En tots els casos la comunicació pot ser xifrada.



El sistema d'autenticació de la xarxa Kerberos

En sistemes d'alta seguretat és usual la utilització de sistemes externs (màquines diferents d'aquella a la qual ens volen connectar) com a serveis d'autenticació. El més conegut és Kerberos, el qual, utilitzant diferents tècniques de xifratge i emissió de certificats (tickets), dóna una seguretat més que adequada per a la majoria de sistemes. La figura mostra el sistema d'autenticació per a la xarxa Kerberos, que fa servir una màquina específica per a dur a terme l'autenticació i, d'aquesta manera, allibera l'SGBD d'aquesta tasca.

1.2. Control d'accés

S'anomena control d'accés aquella part de l'SGBD que té la funció d'assegurar que els accessos al sistema estan d'acord amb els models i regles fixades per la política de protecció.

El control d'accés controla la interacció (lectura, escriptura...) entre els subjectes (usuaris, aplicatius, processos...) i els objectes als qual accedeixen.

El control d'accés es pot considerar format per dos components:

1) Polítiques d'accés: defineixen els principis segons els quals s'autoritza a un usuari, es denega o es filtra un accés específic a un objecte.
2) Mecanismes de seguretat: procediments que s'apliquen a les consultes dels usuaris perquè compleixin les normes anteriors.

El filtratge retorna només una part de la informació demanada.

Les diferents implementacions de les polítiques d'accés es poden classificar en controls d'accés obligatori i control d'accés discrecional.

1.2.1. Control d'accés discrecional

El control d'accés discrecional (DAC) es basa en la identitat dels usuaris o grups d'usuaris per a restringir l'accés a objectes. El control discrecional és el

DAC és la sigla de Discretionary Access Controls.

mecanisme de control més comú que s'implementa en els sistemes d'informació actuals.

Ús del DAC

El control d'accés discrecional és el més utilitzat en els SGBD comercials.

Les polítiques discrecionals es fonamenten en el coneixement dels drets que cada usuari té sobre cada objecte. Aquestes polítiques podran ser definides per l'administrador de la base de dades o pel propietari de l'objecte.

Els drets de l'usuari poden incloure el d'administrar la seguretat de l'objecte.

La millor manera de representar l'estructura del control d'accés discrecional és la matricial:

	Objecte 1	Objecte 2	...	Objecte n
Usuari 1				
Usuari 2				Drets de l'usuari 2 sobre l'objecte n
...				
Usuari m				

Significat de la taula

La intersecció entre files i columnes indica els drets de cada usuari sobre cada objecte.

Una ampliació usual de la taula anterior consisteix a incorporar-hi els grups d'usuaris com si es tractés d'un altre usuari i els privilegis sobre el sistema com si es tractés d'un objecte.

La descentralització de les autoritzacions fa que apareguin problemes de propagació tant d'autoritzacions com de revocacions. Si un usuari M ha rebut una autorització d'un usuari N que tenia el dret d'administrar la seguretat sobre un objecte P, quins drets tindrà l'usuari M si a N se li revoquen tots els permisos? El sistema haurà de preveure polítiques d'autorització i revocació en cadena.

1.2.2. Control d'accés obligatori

El control d'accés obligatori (MAC) s'acostuma a fer servir en aquelles bases de dades en les quals les dades tenen una estructura de classificació molt rígida i estàtica, com per exemple, les bases de dades militars i governamentals.

MAC és la sigla de Mandatory Access Controls.

Ús del MAC

El control d'accés obligatori s'acostuma a utilitzar en els SGBD que treballen amb informació sensible (d'alta seguretat).

Les polítiques de control d'accés obligatori es basen en la idea que cada dada té un nivell de classificació (per exemple, molt secret, secret, confidencial...) i cada usuari té un nivell d'acreditació (amb les mateixes possibilitats que el nivell de classificació). Els diferents nivells estan ordenats de forma estricta: molt secret > secret > confidencial.

Les normes de funcionament del control d'acés obligatori són les següents:

- Un usuari pot veure un objecte solament si el seu nivell d'acreditació és més gran o igual que el nivell de classificació de l'objecte.
- Un usuari pot modificar un objecte només si el seu nivell d'acreditació és igual al nivell de classificació de l'objecte.

Nivell de classificació dels objectes

Els objectes escrits per un usuari adquireixen, com a mínim, un nivell de classificació igual al de l'usuari que els ha escrit.

Els usuaris hauran de tenir, primer, l'accés discrecional autoritzat i els privilegis adequats sobre l'objecte de la base de dades abans que es comprovi el sistema d'accés obligatori. El sistema de seguretat MAC es basa en el concepte d'etiqueta.

Una etiqueta indica el nivell de l'usuari (acreditació) o de l'objecte (classificació). Els valors baixos denoten informació no classificada o menys sensible; els valors alts denoten informació més restrictiva o assequible a menys usuaris.

Les etiquetes es poden refinar afegint-hi altres subnivells.

Exemple d'utilització d'etiquetes

A l'exercit es podria tenir una classificació com la que es presenta a la taula següent, en què a cada etiqueta hi correspon un valor:

Nom	Valor	Descripció
General	80	Alt secret, personal
Oficial	60	Alta seguretat, no distribuir
Suboficial	40	Moderadament segur
Militar	20	Nivell bàsic, no massa sensible
Civil	1	Públic coneixement, lliure distribució

El sistema de gestió de bases de dades comprova primer totes les autoritzacions d'accés discrecional; en cas que l'usuari compleixi tots els nivells de seguretat, el sistema d'accés obligatori compara les etiquetes de l'usuari i de l'objecte, per tal de decidir quin hi té accés.

1.2.3. Classificació dels sistemes de seguretat

Els sistemes de seguretat es classifiquen en quatre nivells de seguretat, que de més petita protecció a més gran són els següents:

1) Protecció mínima: la classe D no incorpora cap mecanisme de seguretat.

2) Protecció discrecional: la classe C suporta el control d'accés discrecional, té les subclasses C1 i C2. La classe C1 requereix separació entre les dades i els usuaris. La classe C2 requereix, a més, processos de registre, auditoria i aïllament de recursos.

Exemple de cobertura

Un exemple de cobertura podria ser inferir la resposta d'una consulta il·legal a partir d'una consulta legal.

3) Protecció estructurada: la classe B suporta el control d'accés obligatori i té les subclasses B1, B2 i B3. La classe B1 requereix protecció de seguretat etiquetada: tots el objectes estan etiquetats amb un nivell de seguretat. La classe B2

requereix, a més, una sentència formal per a cada cosa i també, que els canal de cobertura siguin identificats i eliminats. La classe B3 requereix, també, suports de recuperació i auditoria.

4) Protecció verificada: la classe A, la més segura, requereix una demostració matemàtica que els mecanismes de seguretat són consistents i adequats per a suportar la política de seguretat especificada.

Els sistemes de seguretat habituals

Encara que alguns productes comercials proporcionen un nivell de seguretat B1, el més normal és que només arribin al nivell C2.

Classificació dels sistemes de seguretat

La classificació que es presenta aquí va ser creada pel Pentàgon com a sistema de classificació estàndard. Aquesta classificació es troba definida en l'Orange Book, en el qual es defineixen els requeriments de seguretat per a qualsevol TCB (Trusted Computing Base), i en el Lavender Book, en el qual es defineix la interpretació dels requeriments per un sistema gestor de bases de dades específic.

1.3. Implementació del control d'accés discrecional a SQL:1999

L'estàndard actual defineix la sintaxi d'autoritzacions següent:

Varietat d'estratègies

Els diferents fabricants de SGBD han desenvolupat diverses estratègies per a implementar les polítiques de seguretat.

```
<sentència d'autorització> ::=
         <sentència autorització privilegis>
      |  <sentència autorització rols>

<sentència autorització privilegis> ::=
      GRANT <privilegis> TO <autoritzat> [ { , <autoritzat> } ]
         [ WITH HIERARCHY OPTION ]
         [ WITH GRANT OPTION ]
         [ GRANTED BY <autoritzador> ]

<privilegis> ::=
      <privilegis d'objecte> ON <nom objecte>

<privilegis d'objecte> ::=
      ALL PRIVILEGES | <acció> [ { , <acció> } ]

<acció> ::=
      DELETE |
      SELECT [ (<nom columna> [ , <nom columna> ] ... ] ) ] |
      INSERT [ (<nom columna> [ , <nom columna> ] ... ] ) ] |
      UPDATE [ (<nom columna> [ , <nom columna> ] ... ] ) ] |
      REFERENCES [ (<nom columna> [ , <nom columna> ] ... ] ) ] |
      TRIGGER |
      EXECUTE

<nom objecte> ::=
      [ TABLE ] <nom taula>

<autoritzat> ::=
         PUBLIC | <nom autoritzat>

<autoritzador> ::=
         CURRENT_USER | CURRENT_ROLE
```

El significat dels privilegis que es poden donar a un usuari sobre un objecte són els següents:

Clàusules d'autorització de privilegis

L'opció WITH GRANT OPTION permet de transmetre els privilegis que es té a un altre. La clàusula WITH HIERARCHY OPTION permet de traslladar els permisos a totes les subtaules.

Privilegis	Definició
INSERT [(*llista-noms-columnes*)]	Inserir valors en les columnes relacionades d'una fila.
UPDATE [(*llista-noms-columnes*)]	Actualitzar valors en les columnes relacionades d'una fila.
DELETE	Esborrar files.
SELECT [(*llista-noms-columnes*)]	Consultar valors en les columnes relacionades.
REFERENCES [(*llista-noms-columnes*)]	Fer referència a valors de les columnes relacionades.
TRIGGER	Crear un disparador sobre una taula.
EXECUTE	Executar un procediment incorporat.

Exemples d'autoritzacions

A continuació presentem un exemple de com s'implementen les autoritzacions tractades en aquest subapartat:

```
GRANT SELECT ON TABLE Empleat TO Auditor2;
GRANT USAGE ON DOMAIN d_Telefon TO PUBLIC;
GRANT ALL PRIVILEGES ON VIEW Vista1 TO Joan, Pere, Maria
```

És relativament freqüent associar els mateixos privilegis a diferents usuaris, per exemple, a tot el departament de comptabilitat. Per a facilitar aquesta mena d'autorització, es fa servir el rol. Un rol es pot definir com un conjunt de zero o més privilegis. Si s'autoritza un rol a un usuari, automàticament se li autoritzen tot els privilegis que incorpora el rol.

La sintaxi del rol és la següent:

```
<definició rol> ::=
   CREATE ROLE <nom rol>
      [ WITH ADMIN <autoritzador> ]

<sentència autorització rol> ::=
   GRANT <nom rol> [ { , <nom rol> } ]
         TO <autoritzat> [ { , <autoritzat> } ]
      [ WITH GRANT OPTION ]
      [ GRANTED BY <autoritzador> ]
```

La clàusula WITH ADMIN permet d'indicar qui administrarà el rol.

Exemples de creació i utilització de rols

Considereu els exemples següents de creació i utilització de rols:

```
CREATE ROLE Auditors WITH ADMIN CURRENT_USER;
GRANT SELECT, UPDATE (Salari, Comissio) TO Auditors;
GRANT Auditors TO Pere, Joan, Maria;
```

De la mateixa manera que es donen autoritzacions, es poden revocar. La sintaxi per a revocar rols és la següent:

```
<sentència revocar privilegis> ::=
      REVOKE [ { GRANT OPTION FOR | HIERARCHY OPTION FOR } ]
         <privilegis> FROM <autoritzat> [ { , <autoritzat>} ... ]
         [ FROM {CURRENT_USER | CURRENT_ROLE } ]
         { RESTRICT | CASCADE }

<sentència revocar rol> ::=
      REVOKE [ ADMIN OPTION FOR ] <nom rol> [ { , <nom rol> }... ]
         FROM <autoritzat> [ { , <autoritzat>} ... ]
         [ FROM {CURRENT_USER | CURRENT_ROLE } ]
         { RESTRICT | CASCADE }
```

Problemes de propagació de privilegis

Les opcions RESTRICT i CASCADE tenen una importància especial per a resoldre els problemes de propagació. Quan A autoritza un privilegi a B (amb l'opció d'administrar-lo) i B l'autoritza a C, si a A se li treuen els privilegis que tenia, B i C restaran amb els privilegis "abandonats". Si l'opció triada quan es revoquen els privilegis a A és CASCADE, els permisos de B i C també restaran revocats. Si l'opció és RESTRICT, només es podrà revocar els permisos de A, si prèviament s'han eliminat els permisos candidats a ser abandonats. Si no s'utilitza cap de les opcions, B i C restaran amb els permisos, i es correrà el risc que B torni a donar privilegis a A.

1.4. Auditoria

L'auditoria s'utilitza normalment per als casos següents:

La càrrega del treball d'auditoria

El treball d'auditoria pot representar una sobrecàrrega superior a la nivell del treball normal.

a) La investigació d'una activitat sospitosa.

Exemple d'investigació d'una activitat sospitosa

Si un usuari no autoritzat intenta d'esborrar dades de les taules, l'administrador de seguretat podrà decidir d'auditar totes les connexions de la base de dades i tots els intents (amb èxit o no) d'esborrar dades.

b) El monitoratge i la recollida d'activitats especifiques de la base de dades.

Exemple de monitoratge de la base de dades

L'administrador de la base de dades pot recollir dades sobre quines taules s'actualitzen o quants usuaris estan connectats en moments punta.

L'auditoria és el registre i monitoratge d'algunes accions especificades d'usuaris específics sobre la base de dades.

Auditar significa esbrinar qui és l'autor de qualsevol canvi a la base de dades.

El sistema d'auditoria ha de permetre diferents formes d'utilització:

- Auditar sentències. L'auditoria indicarà quan i qui ha utilitzat un tipus de sentència concreta. Per exemple, auditar totes les insercions o els esborrats.
- Auditar objectes. El sistema registrarà cada vegada que es realitzi alguna operació sobre un objecte determinat.
- Auditar sentències sobre objectes, una versió combinada de les dues anteriors.
- Auditar usuaris o grups.

La creació d'auditories

Alguns SGBD incorporen sentències SQL que permeten de generar una auditoria de manera declarativa; en altres casos caldrà crear disparadors (triggers) que vagin omplint les taules d'auditoria conforme s'hi produeixen esdeveniments.

També es pot decidir si es vol fer un únic registre per sessió, o per sentència o per accés; i si només es vol registrar quan té èxit, quan fracassa o en tots dos casos. La informació que acostuma a registrar l'auditoria és el nom de l'usuari, l'identificador de sessió, l'identificador de terminal, el nom de l'objecte al que s'ha accedit, l'operació executada o intentada, el codi complet de l'operació, la data i l'hora.

1.5. La Llei de protecció de dades de caràcter personal

La legislació espanyola recull una sèrie de drets i deures relatius a la utilització de les dades personals en l'àmbit dels sistemes d'informació. Les principals fonts de drets són les següents: la Llei orgànica 5/1992, de 29 d'octubre, de regulació del tractament automatitzat de les dades de caire personal (LORTAD) i el Reial decret 1332/1994, de 30 de juny, que desenvolupa la llei anterior. Aquesta llei ha estat derogada per la Llei orgànica 15/1999, de 13 de desembre, de protecció de dades de caràcter personal (LOPDCP).

Les lleis LORTAD i LOPDCP s'han fet en compliment del mandat constitucional contingut en l'article 18.4, que diu: "la llei limitarà l'ús de la informàtica per garantir l'honor i la intimitat personal i familiar dels ciutadans i el ple exercici dels seus drets".

Les normatives actuals també segueixen la directiva del Parlament europeu 95/46, de 24 d'octubre, relativa a la protecció de les persones físiques pel que fa al tractament de dades personals i a la lliure distribució d'aquestes dades.

1.5.1. Principis generals de protecció de dades

Finalitat del tractament de les dades

Les dades de caràcter personal objectes del tractament no es podran fer servir per a finalitats incompatibles amb aquelles per les quals s'haurien recollit.

A continuació es passen a descriure els principis generals de la protecció de les de dades:

a) Qualitat de dades: les dades de caràcter personal només es podran ser recollir per al seu tractament, i també es podran sotmetre a aquest tractament

quan sigui adequat, pertinent i no excessiu en relació amb l'àmbit i les finalitats per a les quals s'hagin obtingut.

b) Dret d'informació en la recollida de dades: s'haurà d'informar amb antelació als interessats a qui se sol·licitin les dades personals dels punts següents:

- De l'existència d'un fitxer o tractament de dades de caràcter personal.
- Del caràcter obligatori o facultatiu de la resposta donada a les preguntes que els siguin plantejades.
- De les conseqüències de l'obtenció de les dades o la negativa a subministrar-les.
- De la possibilitat d'exercitar els drets d'accés, rectificació, cancel·lació i oposició.
- De la identitat i direcció del responsable del tractament o, segons del cas, del seu representant.

c) Consentiment de l'afectat: llevat que la llei disposi una altra cosa, per al tractament de les dades de caràcter personal es requerirà el consentiment inequívoc dels afectats.

d) Dades especialment protegides: les dades de caràcter personal que revelin ideologia, afiliació sindical, religió i creences només podran ser objecte de tractament amb el consentiment, exprés i per escrit, de l'afectat.

Exemples de dades de caràcter personal

Les dades de caràcter personal que facin referència a l'origen racial, a la salut i a la vida sexual es podran tractar o cedides només quan així ho disposi la llei.

e) Dades relatives a la salut: les institucions i els centres sanitaris públics i privats i els professionals corresponents podran procedir al tractament de les dades de caràcter personal relatives a la salut de les persones sempre que siguin facilitades pel titular per motiu d'assistència sanitària.

f) Seguretat de les dades i deure secret: tant el responsable del fitxer com l'encarregat del tractament, en el seu cas, hauran d'adoptar les mesures tècniques i organitzatives necessàries que garanteixin la seguretat de les dades de caràcter personal i n'evitin l'alteració, la pèrdua, el tractament o l'accés no autoritzat.

Secret professional de les dades personals

Els responsables dels fitxers i els qui intervinguin en qualsevol fase del tractament dels mateixos estan obligats al secret professional respecte de les dades de caràcter personal i al deure de guardar-les, el responsable del fitxer.

g) Comunicació de dades: les dades de caràcter personal objecte de tractament només es podran ser comunicades a un tercer, amb l'anterior consentiment de l'interessat, per al compliment de finalitats directament relacionades amb les funcions legítimes del cedent i del cessionari.

h) Accés a les dades per terceres persones: els tractaments que realitzin terceres persones haurà d'estar regulat en un contracte per escrit o en alguna altra forma que permeti d'acreditar-ne la subscripció i el contingut. L'accés d'un tercer a aquestes dades no es considerarà comunicació de dades quan sigui necessari per a la prestació d'un servei al responsable del tractament.

Les persones tenen els drets següents:

Impugnació de valoracions

L'afectat podrà impugnar els actes administratius o decisions privades que impliquin una valoració del seu comportament.

a) Impugnació de valoracions: els ciutadans tenen dret a no veure's sotmesos a una decisió amb efectes jurídics que es fonamentin únicament en un tractament de dades destinades a avaluar aspectes de la seva personalitat.

Dret d'accés

El Registre General serà de consulta pública i gratuïta.

b) Dret d'accés: L'interessat tindrà dret a sol·licitar i obtenir gratuïtament informació de les seves dades de caràcter personal sotmeses a tractament, l'origen d'aquestes i les comunicacions efectuades o que se'n previnguin fer.

Tutela dels drets

Les actuacions contràries a la present Llei poden ser objectes de reclamació pels interessats davant l'Agència de Protecció de Dades.

c) Dret de rectificació i cancel·lació: quan les dades de caràcter personal resultin inexactes o incompletes, seran rectificades o cancel·lades. El responsable del tractament té l'obligació de fer efectiu aquest dret de rectificació o cancel·lació de l'interessat.

d) Dret a una indemnització: si el responsable o l'encarregat del tractament incompleixen la llei i si, com a conseqüència d'aquest incompliment, els interessats tenen algun perjudici o lesió en els seus béns o drets, aquests tindran dret a ser indemnitzats.

1.5.2. Els nivells de seguretat

Segons el Reial decret 994/1999, s'estableixen tres nivells de seguretat: el bàsic, el mitjà i l'alt. L'aplicació de l'un i l'altre dependrà de la naturalesa de la informació que emmagatzemin.

Per a l'aplicació de les mesures de seguretat dels fitxers amb dades de caràcter personal, s'ha d'observar el següent:

1) Tots els fitxers que continguin dades de caràcter personal han de seguir les mesures de seguretat del nivell bàsic.

El document de seguretat

El document de seguretat s'haurà de mantenir sempre actualitzat i s'haurà de revisar sempre que es produeixin canvis rellevants en el sistema d'informació o en l'organització d'aquest.

El document s'haurà d'adequar sempre a la normativa vigent en matèria de protecció de dades de caràcter personal.

Les condicions del nivell bàsic de seguretat es definiran en el document de seguretat que haurà d'elaborar i implementar el responsable del fitxer. Aquest document haurà d'incloure els punts següents:

a) L'àmbit d'aplicació del document amb especificació detallada dels recursos protegits.

b) Les mesures, normes, procediments, regles i estàndards encaminats a garantir el nivell de seguretat exigit pel Reial decret 994/1999.

c) Les funcions i obligacions del personal.

d) L'estructura dels fitxers que continguin dades de caràcter personal i descripció dels sistemes d'informació que les tracten.

e) El procediment de notificació, gestió i resposta de les incidències (tant tecnològiques com funcionals i d'accés).

Registre d'incidències

El procediment de notificació i gestió d'incidències ha d'incloure un registre en què faci constar com a mínim el tipus d'incidència, el moment en el qual s'ha produït la incidència, la persona que fa la notificació, la persona a qui es notifica la incidència i els efectes derivats d'aquesta.

f) Els procediments de realització de còpies de suport i de recuperació de dades.

g) La periodicitat amb la qual s'han de substituir les contrasenyes dels usuaris.

h) El personal autoritzat per a concedir, modificar i/o alterar els accessos autoritzats a les dades i/o recursos, i també els criteris per a realitzar aquestes accions.

i) El personal autoritzat per a accedir al lloc on s'emmagatzemen dades de caràcter personal.

2) Els fitxers que continguin dades relatives a la comissió d'infracció penals i/o administratives, la Hisenda Pública i serveis financers (solvència patrimonial i crèdit, compliment o incompliment de les obligacions monetàries) hauran de seguir les mesures del nivell mitjà.

El responsable de seguretat

Molt sovint l'administrador de la base de dades (ABD) és la persona responsable de la seguretat i de mantenir el document de seguretat.

Les mesures de seguretat de nivell mitjà inclouen les del nivell bàsic i, a més, exigeixen el compliment dels punts següents:

a) Identificació del responsable o responsables de seguretat.

b) Controls periòdics (auditoria) per a verificar el compliment del que està establert en el mateix document de seguretat.

Auditories

Els informes d'auditoria s'hauran de dipositar a l'Agència de Protecció de Dades.

c) Mesures que calgui adoptar quan un suport que contingui dades de caràcter personal estigui a punt de ser destruït o reutilitzat.

d) Personal autoritzat per a accedir als locals a on els sistemes d'informació amb dades de caràcter personal es trobin físicament ubicats.

3) Fitxers amb continguts de dades relacionades amb la ideologia, religió, creences, origen racial, salut o vida sexual seguiran unes mesures del nivell alt.

Les mesures de seguretat de nivell alt inclouen les del nivell mitjà i, a més, exigeixen el compliment dels punts següents:

a) Les dades de caràcter personal que s'hagin de distribuir en qualsevol tipus de suport es xifraran.

b) Per a cada accés es guardarà com a mínim la identificació de l'usuari, la data i l'hora en què es va fer l'accés, el nom del fitxer al qual s'ha accedit i el tipus d'accés.

c) S'haurà de guardar un còpia de suport i dels procediments de recuperació de les dades en un lloc diferent d'on es trobin físicament els sistemes d'informació.

d) Les dades de caràcter personal que s'hagin de transmetre per la xarxa de telecomunicacions es xifraran.

1.5.3. L'Agència de Protecció de Dades

L'Agència és un ens de dret públic, amb personalitat jurídica pròpia i plena capacitat pública i privada. Actua amb total independència de les administracions públiques en l'exercici de les seves funcions. La seva finalitat principal consisteix a vetllar pel compliment de la legislació sobre protecció de dades personals i controlar-ne l'aplicació, en especial en tot allò relatiu als drets d'informació, accés, oposició, rectificació i cancel·lació de dades.

L'agència de protecció de dades a Internet

Podeu trobar les darreres actualitzacions de totes les normatives, i fer notificacions i registres d'inscripció dels fitxers de dades personals a l'adreça d'Internet de l'Agència de Protecció de Dades:

https://www.agenciaprotecciondatos.org/

La pàgina també disposa de les FAQ (preguntes més freqüents) amb la finalitat que el ciutadà pugui exercir els seus drets i amb la d'aclarir com les organitzacions han de complir amb el seus deures.

2. El processament de vistes

Des d'un punt de vista teòric, una vista es pot definir com el resultat dinàmic d'una o més operacions relacionals sobre les taules relacionals bàsiques per a produir una altra relació.

Les vistes són relacions virtuals que només estan representades pel seu nom i la seva definició; no existeixen físicament, de forma definitiva, a la base de dades, si no que es crea (més endavant veurem que aquest segurament no és sempre el verb més adequat) en el moment que un usuari la fa servir.

Una versió reduïda de la sintaxi de creació de vistes segons l'estàndard SQL:1999 és la següent:

```
CREATE [ RECURSIVE] VIEW vista [ ( llista-columnes) ]
   AS subconsulta
   [WITH CHECK OPTION ]
```

La clàusula WITH CHECK OPTION assegura que mai no podrem actuar sobre parts d'una taula que no estan a la vista.

Exemple de definicions de vistes

Si es té la base de dades definida pel diagrama relacional següent:

```
DEPARTAMENT(codi, nom, localitat)
EMPLEAT(codi,nom,localitat,salari,categoria,departament)
   Departament fa referència a Departament(codi)
```

Algunes definicions de vistes són les següents:

```
CREATE VIEW Vista1 AS
   SELECT Nom, Dataalta, Salari, Categoria
   FROM Empleat
   WHERE Categoria NOT IN ('President', 'Directiu')

CREATE VIEW Vista2(Empleat, Departament) AS
   SELECT E.Nom, D.Nom
   FROM Empleat E, Departament D
   WHERE E.Departament = D.Codi
```

```
CREATE VIEW Vista3 AS
   SELECT *
   FROM Vista2
   WHERE Empleat IN
      (SELECT Nom
      FROM Vista1)
```

De manera opcional, es pot assignar nom a les diferents columnes de la vista; en aquest cas el nom de les columnes coincidirà amb el de la subconsulta; si no es defineixen noms, les columnes prendran el mateix nom que tenen a la subconsulta. En cas que la subconsulta inclogui columnes calculades o funcions, serà imprescindible definir un nom per a aquestes columnes.

Fixeu-vos que, tal com es pot veure en el tercer dels exemples, en la subconsulta d'una vista pot aparèixer una altra vista.

D'altra banda, què passaria si es fes l'actualització següent?

```
UPDATE Vista1
SET Categoria = 'Directiu'
WHERE Categoria = 'Analista'
```

Es donaria el cas inversemblant que es trauria una fila fora de la vista. Una situació semblant es donaria si es fes la inserció següent:

```
INSERT INTO Vista1
VALUES ('Mateu', '12/3/2001', 1500, 'Directiu')
```

Si la vista s'hagués creat amb la clàusula WITH CHECK OPTION, les operacions s'haurien comprovat prèviament, per a assegurar que les noves files inserides o actualitzades compleixen la condició de la vista.

Encara que una vista no ocupa gaire lloc a la base de dades, recordeu que només és una definició i, per tant, també es pot destruir. La clàusula opcional CASCADE destruirà tots els objectes creats a partir d'aquesta vista.

Exemple de destrucció d'una vista

Amb la sentència següent destruirem la primera de les vistes creada a l'exemple anterior:

```
DROP VIEW Vista1 CASCADE
```

2.1. Mecanismes d'implementació de vistes

Hi ha dues estratègies fonamentals per a implementar les vistes: reescriure la consulta o materialitzar la vista.

L'estratègia de reescriptura és la més utilitzada per l'SGBD. Segons aquesta estratègia, la consulta es refà de manera que referenciï les taules de la base de dades.

Reescriptura o càlcul inline

L'estratègia d'implementació de vistes per reescriptura també es coneix amb el nom de càlcul sota comandament o, en anglès, inline.

Exemple d'implementació de vistes per reescriptura

Considereu la base de dades definida pel diagrama relacional de l'exemple anterior i les vistes següents:

Vegeu el diagrama relacional presentat en l'"Exemple de definicions de vistes" vist en aquest mateix apartat.

```
CREATE VIEW Informacio1(Departament, Salari, Empleats) AS
   SELECT D.Nom, AVG(E.Salari), COUNT(*)
   FROM Empleat E, Departament D
   WHERE E.Departament = D.Codi
      AND E.Categoria NOT IN ('President', 'Directiu')
   GROUP BY D.Nom
```

Una consulta com la que presentem a continuació:

```
SELECT Departament, Salari
FROM Informacio1
WHERE Empleats > 4
```

es convertiria en el següent:

```
SELECT D.Nom, AVG(E.Salari)
FROM Empleat E, Departament D
WHERE E.Departament = D.Codi
   AND E.Categoria NOT IN ('President', 'Directiu')
GROUP BY D.Nom
HAVING COUNT(*) > 4
```

La consulta que realment s'executa amb aquesta estratègia és molt més complexa que la que es planteja inicialment i aquesta complexitat farà que les consultes requereixin molt temps d'execució. El súmmum de la ineficiència es dóna quan cal fer diferents consultes sobre una mateixa vista d'una forma seguida. També pot ser difícil la conversió en cas de consultes molt complexes.

Amb la segona estratègia, anomenada materialització de la vista, quan es fa la consulta es crea una taula temporal amb el contingut físic de la vista.

Duració de les vistes

Si durant un període de temps la vista no es consulta, el mateix sistema la destrueix i la torna a contruir de nou, si cal, en un altre moment.

Exemple d'implementació de vistes per materialització

Vegeu l'"Exemple de definicions de vistes".

Si s'apliqués l'estratègia de materialització de la vista a l'exemple de definicions de vistes, es crearia la taula temporal informacio1 amb les dades del resultat de la consulta de creació de la vista, i les consultes es farien físicament sobre aquesta taula.

Segons aquesta estratègia, el cas de mínima eficiència que presentava la primera estratègia es transformaria en màxima eficiència, ja que només hauria de calcular la taula una sola vegada.

En contrast amb aquest avantatge, l'estratègia de materialització de la vista presenta un inconvenient: si les taules bàsiques a partir de les quals es construeix la vista varien de contingut, caldrà modificar la taula temporal (actua-

lització incremental). Això pot resultar força costós, sobretot per a alguns casos de funcions agregades o en què apareguin clàusules GROUP BY.

Caldria esperar d'un bon SGBD que utilitzés la primera de les estratègies per a les consultes sobres vistes simples i la segona, per a les complexes.

2.2. Actualització de vistes

Les vistes sempre es poden consultar, però no sempre es poden actualitzar. Considerem, per exemple, una taula amb les dades dels empleats i una vista amb el nom de departament al qual estan assignats i el salari mitjà del departament. Quin significat tindria augmentar un 10% la mitjana dels salaris d'un departament? Aquest augment es pot aconseguir de moltes maneres, apujant i abaixant els salaris dels diferents membres del departament. Per tant, estem davant una ambigüitat.

D'una manera més formal, sigui D una base de dades, i V una vista que es construeix amb la funció V = F(D). Si es fa una operació d'actualització A sobre la vista A(V) = A(F(D)), caldrà trobar una operació d'actualització A', tal que A(V) = F(A'(V)). En molts casos hi haurà moltes operacions A' que compliran la igualtat anterior, encara que deixaran la base de dades en un estat diferent.

Només és possible assegurar que una vista serà actualitzable, si està construïda sobre una única taula i els atributs de la vista contenen una clau candidata de la taula original. Les vistes definides sobre més d'una taula acostumen a no ser actualitzables. Les vistes que inclouen clàusules GROUP BY o funcions agregades mai no són actualitzables.

El concepte actualitzable té una base semàntica, no sintàctica; es poden trobar definicions de vistes que a priori semblen no actualitzables, però sí que ho són.

Expressions actualitzables equivalents

La vista que presentem a continuació sembla no actualitzable:

```
CREATE VIEW Vista1 AS
   SELECT Codi
   FROM Empleat
   WHERE Salari > 30
      UNION
   SELECT Codi
   FROM Empleat
   WHERE Departament = '20'
```

Però sí que ho és, atès que és equivalent a aquesta altra:

```
SELECT Codi
FROM Empleat
WHERE Salari > 30 OR Departament = '20'
```

I aquesta expressió sí que és actualitzable.

2.2.1. Actualització amb disparadors de substitució

Hi ha diferents mecanismes per a trencar l'ambigüitat en l'actualització de vistes; tots es basen a guardar, conjuntament amb la definició de la vista, el criteri d'actualització adequat. Els disparadors de substitució* són un dels més utilitzats pels fabricants de SGBD. Actualment dos dels més grans fabricants de SGBD (Oracle8ir2 i Microsoft SQL Server 2000) els fan servir.

* En anglès, triggers instead of.

Els disparadors de substitució s'activen en lloc del comandament que ha provocat el disparament i sempre han d'estar definits a la fila. Aquesta mena de disparadors permeten d'efectuar modificacions sobre vistes que no són actualitzables. Només es poden definir sobre vistes.

Vegeu l'"Exemple d'implementació de vistes per reescriptura" en el subapartat 2.1 d'aquest mòdul.

Exemple d'actualització de vistes amb disparadors de substitució

Considerem l'exemple clàssic d'implementació de vistes. Suposem que s'hi defineix la vista següent:

```
CREATE VIEW Informacio2 (Nom, Salari) AS
            SELECT D.Nom, AVG(E.Salari)
            FROM Empleat E, Departament D
            WHERE E.Departament = D.Codi
            GROUP BY D.Nom
```

Com hem vist, aquesta vista és clarament no actualitzable. Si la política de l'organització és l'increment proporcional, un disparador que fes actualitzable aquesta taula podria ser el següent (la sintaxi utilitzada és Oracle8ir2):

```
CREATE TRIGGER ActualitzarInformacio2
INSTEAD OF UPDATE ON Informacio2
DECLARE
v_IncrementNUMBER;
BEGIN
   IF :NEW.Salari != 0 and :OLD.Salari != 0 THEN
      v_Increment := :NEW.Salari / :OLD.Salari;
      UPDATE Empleat
      SET Salari =v_Increment * Salari
      WHERE Departament = :OLD.Nom AND Salari IS NOT NULL;
   END IF;
   IF :NEW.Nom != :OLD.Nom THEN
      UPDATE Empleat
      SET Departament = :NEW.Nom
      WHERE Departament = :OLD.Nom;
   END IF;
END ActualitzarInformacio2;
```

En l'exemple es pot veure que per a actualitzar el salari a un valor nou, el salari s'incrementa proporcionalment a l'increment que tindria la mitjana de salaris en tots ells. Si la

política d'actualització de salaris de la nostra organització es proporcional, l'actualització funcionarà. De la mateixa manera, si es canvia el nom d'un departament, es canviarà en la taula corresponent.

Si la política de l'organització és que quan desapareix un departament tots els seus empleats queden en espera de destinació, dit d'una altra manera, amb el camp empleat a nul, el disparador corresponent fóra el següent:

```
CREATE TRIGGER EsborrarSalarisDepartament
INSTEAD OF DELETE ON SalarisDepartament
DECLARE
   v_Codi Empleat.Departament%Codi;
BEGIN
   SELECT Codi INTO v_Codi
   FROM Departament
   WHERE Nom = :OLD.Nom;
   UPDATE Empleat
   SET Departament = NULL
   WHERE Departament = v_Codi;
   DELETE FROM Departament
   WHERE Nom = :OLD.Nom;
END EsborrarSalarisDepartament;
```

L'ús d'aquesta mena de disparadors facilita el disseny intern, i fa més fàcil l'encapsulació de la informació .

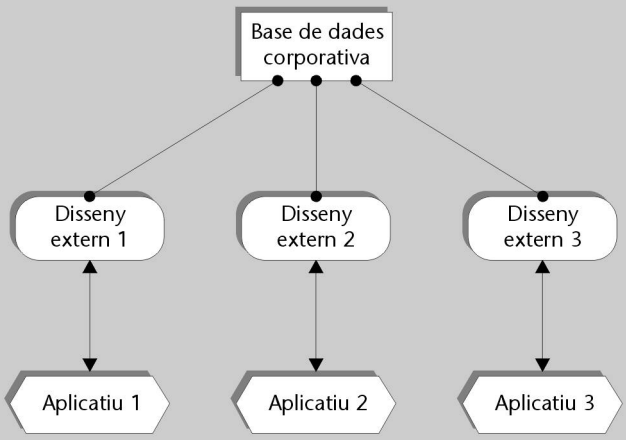
2.3. La vista com a element de disseny extern

Les bases de dades sempre s'haurien de dissenyar tenint en compte com és l'organització o el sistema, mai la utilització que es farà de les dades, si no és que cal per raons d'eficiència. Una base de dades corporativa serà utilitzada per múltiples aplicatius, i a tots els agradaria que la base de dades fos dissenyada a mida de les seves necessitats (si més no, als seus dissenyadors). La manera d'obtenir aquest doble objectiu rep el nom de disseny extern.

El disseny extern consisteix a crear una interfície entre la base de dades i l'aplicatiu.

En l'esquema següent es pot veure una representació gràfica d'aquesta situació:

Funció del disseny extern en una base de dades corporativa



Implementació del disseny extern

Per a implementar el disseny extern, es podran fer servir (segons les possibilitats de l'SGBD) vistes, consultes, vistes objecte, procediments, funcions, disparadors de substitució, mètodes, etc.

Una manera força senzilla i molt segura d'implementar el disseny extern és fer servir una col·lecció de vistes amb els respectius disparadors de substitució per a cadascuna de les aplicacions client; d'aquesta manera, les aplicacions es "faran la il·lusió" que tenen una base de dades dissenyada a mida, en la qual tota la informació es troba tal com els convé.

Operacions sobre vistes

Quan es treballa sobre les vistes (que les aplicacions creuen que són taules) es poden fer tota mena d'operacions. Aquestes operacions es filtren i transformen en operacions sobre les taules reals del sistema.

Exemple d'implementació de disseny extern amb vistes

Una aplicació podria tenir assignat en el seu esquema la vista Informacio1 que hem vist en els exemples anteriors sense cap mena de disparador de substitució perquè només haurà de fer consultes, mentre que una altra "veurà" la base de dades com si només hi hagués la taula Informacio2, sobre la qual pot esborrar i actualitzar gràcies als disparadors de substitució, però no inserir.

2.4. Les taules derivades

La major part dels sistemes gestors de bases de dades incorporen unes estructures molt semblants a les vistes, les taules derivades, també conegudes com instantànies (snapshots), agregats o vistes materialitzades.

Atenció

En aquest apartat es fa servir la notació i facilitats de l'SGBD comercial Oracle8ir2.

Les taules derivades són taules que tenen existència física, ocupen lloc a la base de dades, però es calculen (deriven) a partir de les taules bàsiques.

Exemple de taula derivada

En un supermercat es podria tenir una taula amb la suma de les vendes de cada secció. En aquesta taula no hi inseriríem cap fila, sinó que s'actualitzaria automàticament a mesura que s'anirien inserint files a la taula de vendes.

Per a poder crear la vista materialitzada, a més de la consulta de creació, caldrà indicar alguns elements més, el més important dels quals és la clàusula de refresc, és a dir, el mecanisme que farà servir l'SGBD per a mantenir la taula derivada actualitzada a mesura que vagin canviant les taules "mestres". El sistema de gestió de bases de dades Oracle8ir2 presenta les estratègies d'actualització que s'expliquen a continuació i que poden servir d'exemple per a altres SGBD de potencialitats semblants, encara que amb una sintaxi diferent.

Comparació entre vistes i taules derivades

Les vistes són simples definicions que s'utilitzen quan cal (per tal de materialitzar o reescriure una vista).

En canvi, les taules derivades existeixen físicament.

L'actualització es pot produir en dos moments diferents:

1) ON COMMIT ('a la confirmació'): el refresc esdevindrà quan la transacció que modifiqui alguna de les taules mestres de la taula derivada es confirmi.
2) ON DEMAND ('sota comandament'): el refresc ocorrerà quan l'usuari executi manualment algun procés de refresc.

L'actualització també es pot produir de maneres diferents:

a) COMPLETE ('complet'): es recalcula tota la taula derivada segons la consulta que la defineix.

b) FAST ('ràpid'): aquest mètode d'actualització especifica un mètode de refresc incremental; els canvis s'efectuaran afegint les noves dades que han estat afegides a les taules mestra.

c) FORCE ('forçat'): aplica, si és possible, el refresc ràpid; en un altre cas, aplica el refresc complet.

d) NEVER ('mai'): indica que la vista materialitzada mai no serà refrescada.

Exemple d'actualització amb taules derivades

Considerem el següent model relacional de la base de dades operativa d'un supermercat:

```
VENDA(numero, caixa, data, caixer)
SECCIÓ(codi, nom)
ARTICLE(codi, secció, descripció, preu)
   Seccio fa referència a Seccio(codi)
LÍNIA(venda, numero, article, quantitat)
   Numero fa referència a Venda(numero)
   Article fa referència a Article(numero)
```

La taula derivada corresponent a les vendes per secció es podria crear amb l'expressió següent:

```
CREATE MATERIALIZED VIEW Prova3
   BUILD IMMEDIATE
   REFRESH FAST ON DEMAND
   ENABLE QUERY REWRITE
   AS SELECT S.Nom, SUM(L.Quantitat * A.Preu)
      FROM Linia L, Article A, Seccio S
      WHERE L.Article = A.Codi
         AND A.Seccio = S.Codi
         GROUP BY S.Nom
```

Si les possibilitats de les vistes materialitzades només fossin aquestes, tindríem molt de treball per al SGBD, que només es faria servir per a algunes consultes molt concretes. Els sistemes de gestió de bases de dades més moderns presenten mecanismes de reescriptura (ENABLE QUERY REWRITE) de consultes que, davant d'una consulta, permeten de fer servir una vista materialitzada, si es més eficient, encara que no se'n faci referència.

Exemple de mecanismes de reescriptura de consultes

Seguim l'exemple anterior. Suposem que es fes la consulta següent:

```
SELECT S.Nom, SUM(L.Quantitat * A.Preu)
FROM Linia L, Article A, Seccio S
WHERE L.Article = A.Codi
   AND A.Seccio = S.Codi
   AND S.Nom IN ('Verdura', 'Basar', 'Carn')
   GROUP BY S.Nom
   HAVING SUM(L.Quantitat * A.Preu) > 2500000
```

Aleshores, el sistema aprofitaria la vista materialitzada que ja té calculada i reescriuria la consulta com:

```
SELECT S.Nom, SUM(L.Quantitat * A.Preu)
FROM Prova3
WHERE S.Nom IN ('Verdura', 'Basar', 'Carn')
   AND SUM(L.Quantitat * A.Preu) > 2500000
```

No posem noms a les diferents columnes perquè d'aquesta manera l'usuari fa aquesta conversió de forma transparent.

Les taules derivades es poden utilitzar en entorns de gestor de dades per a emmagatzemar i precalcular dades agregades; s'acostuma a conèixer aquests entorns com a resums.

Taules derivades en entorns distribuïts

En entorns distribuïts, les taules derivades es poden utilitzar per a replicar dades a les diferents seus, sincronitzant les actualitzacions sense crear cap mena de conflicte.

Les taules derivades sempre es calculen a partir de les taules bàsiques i l'única operació que s'hi pot fer és la consulta; no té sentit parlar d'actualització, esborrament o inserció.

2.5. Les taules temporals

Un altre tipus de taules especials són les taules temporals, o més exactament s'haurien d'anomenar taules de contingut temporal.

En les taules temporals el contingut té vigència dintre d'una sessió (o una transacció) i desapareix en el moment en què aquesta finalitza.

La sintaxi de creació d'una taula temporal és la següent:

```
CREATE {GLOBAL | LOCAL} TEMPORARY TABLE nom-taula
     definició-taula
ON COMMIT { PRESERVE ROWS | DELETE ROWS}
```

Aquestes taules són molt útils per a emmagatzemar resultats intermedis d'una transacció.

Exemple d'utilització d'una taula temporal

Considerem la creació de la taula següent:

```
CREATE LOCAL TEMPORARY TABLE Auxiliar (
   Nom         VARCHAR2(50),
   Nota        NUMBER(3,1))
ON COMMIT DELETE ROWS
```

Quan es faci la primera inserció la taula es materialitzarà i es podrà treballar amb aquesta fins al moment en què en faci la clàusula COMMIT; llavors, la taula es destruirà i només en restarà la definició en el diccionari de dades.

2.6. Avantatges i desavantatges de la utilització de vistes

Com en qualsevol decisió de disseny, la utilització de vistes presenta una sèrie d'avantatges i desavantatges. Els avantatges més importants són els següents:

a) Independència de les dades i les aplicacions: si es fan servir vistes com a interfície es pot arribar a una independència completa entre l'estructura real de les dades i la que veuen les aplicacions. En qualsevol moment es pot canviar el nom d'una taula, o afegir-hi noves columnes, o canviar-ne completament l'estructura. Tots aquests canvis resulten invisibles per a les aplicacions clients. Només caldria redefinir la interfície, és a dir, les vistes.

b) Simplificació de l'ús per a l'usuari: la utilització de les vistes permet de tenir emmagatzemades consultes força complexes, amb múltiples combinacions de taules, condicions i funcions agregades i d'utilitzar-les amb una consulta molt simple.

c) Millora de la seguretat: l'usuari no coneix les taules i les columnes que formen realment la base de dades, de manera que no pot ni tan sols intentar d'accedir-hi; només té noticia d'aquella informació i estructura que li donem a partir de les vistes.

d) Integritat de les dades: amb la clàusula WITH CHECK OPTION l'usuari no pot portar dades fora dels límits que té marcats; tots els canvis que faci sobre la vista (si és actualitzable) hauran de deixar la fila consistent amb les condicions de la vista.

e) Rendiment: si és possible determinar quina mena de consultes es faran, a partir de la vista també serà possible determinar camins d'accés ràpid per a millorar l'eficiència de la consulta.

Però no tot són avantatges; els desavantatges més importants són els següents:

a) Restriccions d'actualització: no totes les vistes són actualitzables (en realitat, la majoria no ho són); no és possible fer un disseny extern només amb vistes (llevat que es facin servir disparadors de substitució).

b) Restriccions d'estructura: alguns SGBD, seguint una norma ISO, no permeten de construir una vista a partir de qualsevol consulta: una vista creada a partir d'una sentència SELECT, no tindrà en compte les columnes noves que s'hagin pogut afegir a la taula original.

3. El processament de consultes

El processament de consultes fa referència a totes aquelles activitats que el sistema du a terme per a extreure informació de la base de dades.

Quan van aparèixer les bases de dades, una de les principals crítiques que van tenir era la ineficiència: fins aquell moment les consultes es feien en llenguatges procedimentals amb els quals era possible d'optimitzar els mecanismes d'accés; en canvi, amb un sistema gestor de bases de dades modern la consulta es fa amb un llenguatge no procedimental (usualment en SQL): es diu què es vol, però no com s'ha d'obtenir el resultat, i el sistema haurà de triar quines pàgines del disc haurà de llegir i com.

El procés de consulta està format per una sèrie de subprocessos d'anàlisi, traducció i optimització que s'executaran seqüencialment. Els analitzadors comproven la validesa de la consulta, els traductors transformen la sentència en una altra de més propera a les possibilitats del sistema operatiu i els optimitzadors transformen una sentència en una altra d'equivalent, però "millor". En diferents moments del procés el concepte de millora pot variar, encara que sempre té un caire d'eficiència.

Comparació dels SGBD7

No tots els SGBD utilitzen els mateixos algorismes d'optimització, ni les mateixes tècniques d'accés a les dades. Per això, quan es comparen diferents SGBD, un dels paràmetres més importants és la manera com es fa la consulta i amb quina eficiència.3

El fet que l'SGBD sigui l'encarregat de dur a terme l'optimització presenta una sèrie d'avantatges:

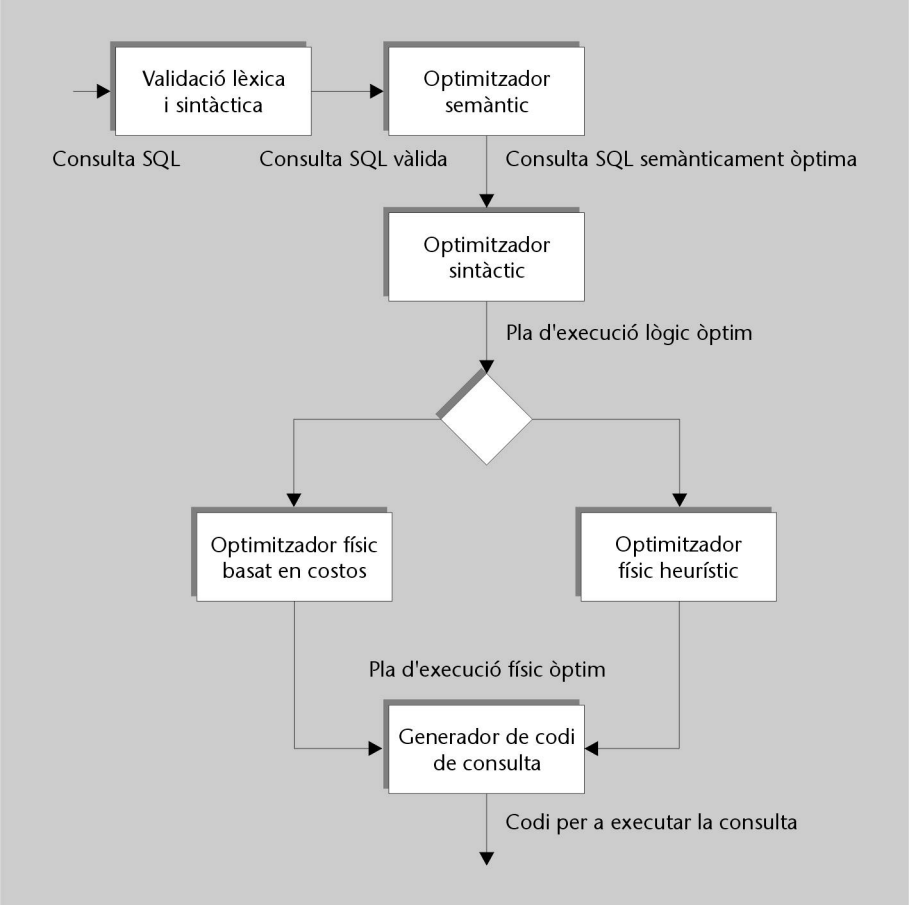
a) L'usuari no s'ha de preocupar de com formular les consultes; les poden fer usuaris no experts.

b) El sistema de gestió de bases de dades disposa d'informació constant (estadístiques) de com i on són les dades; a cada moment pot triar la millor estratègia.

c) Si l'optimitzador està ben construït, reuneix les habilitats i coneixements dels millors programadors de consultes, alhora que la paciència i la potència de càlcul (per a triar la millor opció) de l'ordinador.

En el procés de consulta primer se'n comprova la validesa lèxica, és a dir, si hi ha faltes d'ortografia. Seguidament, la validesa semàntica, o sigui, si els objectes dels quals es parla a la consulta són vàlids o no. Aleshores es pot fer una primera optimització semàntica. El resultat de l'optimització es transforma en l'expressió relacional corresponent, anomenada pla d'execució o arbre de consulta lògic. A partir del primer arbre de consulta lògic comença l'optimització sintàctica, que consisteix a transformar heurísticament l'expressió relacional original en una altra d'equivalent, però que sigui molt més eficient.

Per a implementar físicament les operacions anteriors, hi ha diferents algorismes. Triar quins són els millors rep el nom d'optimització física i amb aquest procés es transformarà l'arbre de consulta lògic inicial en un de físic que indicarà al sistema operatiu què ha de fer. L'esquema de la figura següent indica aquestes etapes:



3.1. Optimització semàntica

Quan es fa una consulta, es comprova en primer lloc si la sintaxi és correcta; si ho és, es passa a fer una validació semàntica: s'accedeix al diccionari de dades i es comprova que totes les columnes i taules que incorpora la consulta existeixen i que l'usuari hi té drets d'accés; si no hi ha cap problema, es passa a l'optimització.

L'optimització semàntica consisteix a transformar (reescriure) una sentència SQL en una altra de "millor"* mitjançant les lleis de la lògica i les restriccions d'integritat pròpies de les taules consultades.

* En aquest context, millor vol dir amb menys condicions en la sentència WHERE.

Exemples d'optimització semàntica

En els exemples següents s'apliquen les regles de la lògica. Considerem la base de dades següent:

```
Empleat(codi, nom, salari, departament), Departament(codi, nom)
```

amb Empleat(departament) que fa referència a Departament(codi). Considerem la consulta següent:

```
SELECT Nom, Salari
FROM Empleat
WHERE Salari > 1200 AND Salari > 2000
```

Aquesta consulta es pot simplificar, ja que la segona condició inclou la primera, de manera que resulta:

```
SELECT Nom, Salari
FROM Empleat
WHERE Salari > 2000
```

D'altra banda, no cal fer aquesta altra consulta:

```
SELECT Nom, Salari
FROM Empleat
WHERE Salari < 1200 AND Salari > 2000
```

atès que les dues condicions són incompatibles.

En altres casos s'hauran de tenir en compte les restriccions d'integritat: si hi ha una restricció que indica que el salari ha de ser superior a 500 euros, la consulta que presentem a continuació:

```
SELECT Nom, Salari
FROM Empleat
WHERE Salari > 400 AND Departament = 3
```

Passaria a ser aquesta altra:

```
SELECT Nom, Salari
FROM Empleat
WHERE Departament = 3
```

Així mateix, considerem aquesta altra consulta:

```
SELECT E.Departament
FROM Empleat E, Departament D
WHERE E.Departament = D.Codi
```

Atès que tots els valors d'E.Departament han d'aparèixer a D.Codi, la consulta que presentem a continuació equivalent a l'anterior:

```
SELECT E.Departament
FROM Empleat E
```

La implementació dels optimitzadors en l'SGBD permetrà d'incrementar-ne el rendiment, sobretot en aquells sistemes que tinguin una semàntica molt rica i la tinguin completament implementada, ja que no caldrà cercar segons totes aquelles condicions que no es compleixen per a cap fila de la taula.

3.2. Optimització sintàctica

Una vegada comprovat que la consulta no conté errors i un cop s'ha dut a terme l'optimització semàntica, caldrà traduir la consulta a algun llenguatge intern, que haurà de ser prou ric per a permetre de representar qualsevol mena de consulta i prou neutral perquè no predisposi a cap estratègia d'implementació. La millor elecció resulta ser l'àlgebra relacional.

El traductor transformarà la consulta en SQL (no procedimental) en una consulta d'àlgebra relacional (procedimental). Com ja s'ha sabeu, qualsevol expressió d'àlgebra relacional es pot presentar en forma d'un arbre que es coneix com a pla de consulta lògic.

Vegeu amb més detall el pla de consulta lògic en les assignatures Bases de dades I i Bases de dades II.

Normalment, el traductor genera un pla lògic canònic estàndard sense efectuar cap mena de transformació o millora, encara que alguns SGBD el transformen a una forma normal conjuntiva: una sèrie de "blocs" connectats per l'operador I; o disjuntiva, en la qual tots els "blocs" estan connectats per l'operador O.

Exemple d'optimització sintàctica

Considerem les taules que es defineixen a continuació:

```
ASSIGNATURA(codi, nom)

ESTUDIANT(codi, nom)

MATRICULA(numero, data, estudiant)

OBTENIR(matricula, assignatura, qualificacio, nota)
```

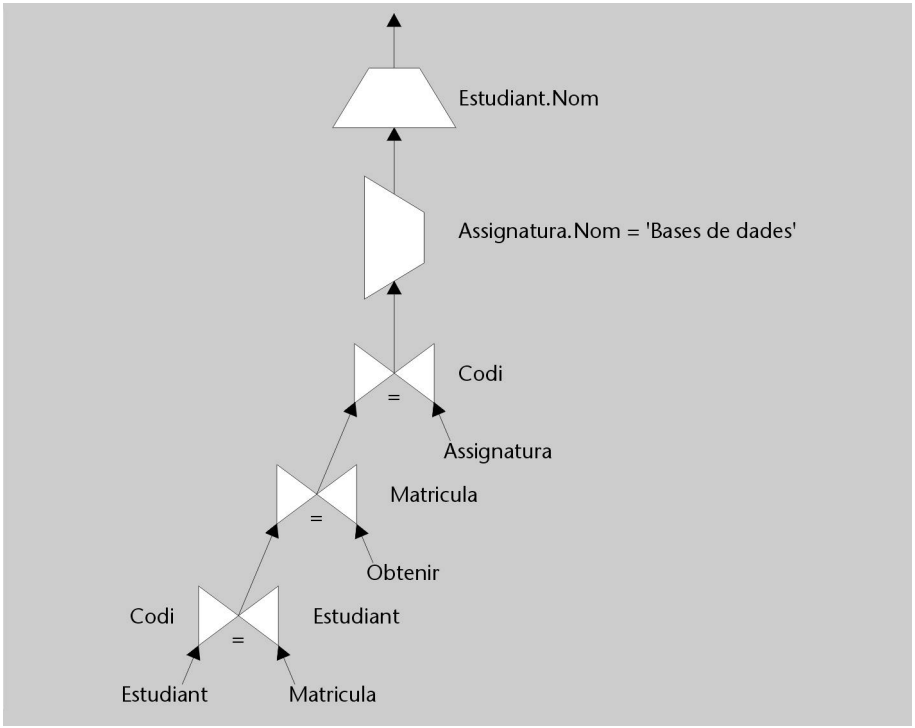
on estudiant referencia a estudiant(codi) matricula referencia a matricula(codi) i assignatura referencia a assignatura(codi).

Es fa una consulta sobre el nom dels estudiants que han obtingut un notable en l'assignatura de bases de dades. En SQL, aquesta consulta es farà de la manera següent:

```
SELECT E.Nom
FROM Estudiant E, Matricula A, Obtenir O, Assignatura A
WHERE A.Nom = 'Bases de Dades AND A.Codi = O.Assignatura
AND O.Matricula = M.Numero AND M.Estudiant = E.Codi
```

La mateixa consulta expressada en àlgebra relacional fóra (utilitzant resultats intermitjos i tenint en compte que els arbres de consulta s'executen de baix a dalt):

```
R1 := Estudiant [Codi = Estudiant] Matricula    Combinació (join)
R2 := R1 [Numero = Matricula] Obtenir           Combinació (join)
R3 := R2 [Assignatura = Codi ] Assignatura      Combinació (join)
R4 := R3 [Assignatura.Nom = 'Bases de Dades']   Selecció
R5 := R4 [Estudiant.Nom]                        Projecció
```



3.2.1. Plans de consulta lògics equivalents

Hi ha moltes regles per a transformar una consulta relacional en una altra d'equivalent. Les consultes equivalents són consultes que donen el mateix resultat quan s'apliquen sobre qualsevol extensió de les relacions participants. Les propietats de l'àlgebra que es fan servir per a transformar les expressions relacionals són les següents:

a) Una seqüència de seleccions sobre una relació T es pot transformar en una sola restricció: $T(S_1 \wedge S_2 \wedge ... \wedge S_n) = ((T(S_1))(S_2)) ...)(S_n)$.

b) Una selecció d'una projecció es pot transformar en una projecció d'una selecció: $(T(S_1))(S_2) = (T(S_2))(S_1)$.

c) En una seqüència de projeccions sobre una relació R es poden ignorar totes menys l'última, si cadascun dels atributs indicats a la darrera també apareix a totes les altres: $((R[P_1])[P_2]) ... [P_n] = R[P_n]$.

d) La selecció és distributiva respecte a la unió, la intersecció i la diferència; i la selecció ho és respecte a la unió i la diferència. La projecció és distributiva respecte a la combinació si tots els atributs de la condició s'inclouen en la projecció.

e) Les operacions d'unió, intersecció i combinació són commutatives, associatives i idempotents.

Heurístics

Heurístic vol dir 'basat en allò que acostuma a passar'.

Els heurístic acostumen a donar bons resultats en la majoria dels casos, però en alguns poden donar resultats funestos.

3.2.2. Heurístics d'optimització sintàctica

Hi ha una sèrie de regles heurístiques que permeten de trobar una bona expressió equivalent (moltes vegades, la millor) a partir de l'aplicació de les pro-

pietats anteriors. Els heurístics normalment acceptats són els que presentem a continuació:

a) Fer les operacions de selecció tan aviat com sigui possible: les seleccions sempre redueixen la cardinalitat del resultat intermedi i, per tant, els elements que cal operar en les etapes posteriors. Si sobre una relació s'ha d'aplicar més d'una selecció es fan servir les diferents propietats commutatives en les quals intervé la selecció per a aconseguir de fer-les totes alhora. En l'arbre del pla de consulta lògic hauran d'estar al més baix possible.

b) Fer les operacions de projecció tan aviat com sigui possible, si escau, immediatament després de les seleccions amb les quals estiguin relacionades. Si sobre una taula cal aplicar més d'una projecció, es fan servir les diferents propietats commutatives en les quals intervé la projecció per a fer-les totes alhora.

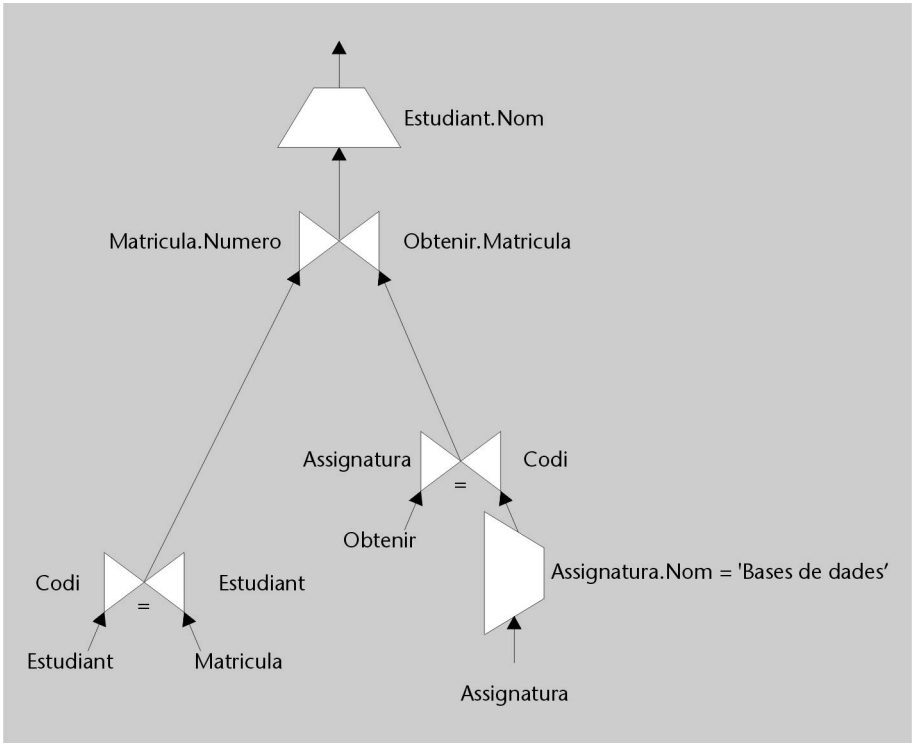
c) Utilitzar l'associativitat de les operacions binàries (combinacions, unions i interseccions), per a fer primer l'operació que doni el resultat de cardinalitat més petita de manera que, després, sigui més fàcil de dur a terme l'operació següent, amb el primer operador de la mida més petita possible.

d) Si hi ha una expressió comuna en diferents parts de l'arbre, fer l'operació una única vegada.

Exemple d'aplicació dels heurístics

Vegeu l'"Exemple d'optimització sintàctica" al principi del subapartat 3.2 d'aquest mòdul.

Considerem l'exemple d'optimització sintàctica presentat i intentem d'aplicar-hi els heurístics que hem explicat. Es pot comprovar que com a primera operació es pot fer la selecció; d'aquesta manera restringirem la cardinalitat del resultat a una única fila (si, com resulta lògic, el nom de l'assignatura és una clau alternativa). Després es pot operar amb la relació Obtenir. Aleshores es pot aplicar l'heurístic de l'associativitat i combinar Estudiant i Matricula i després de fer la combinació entre tots dos resultats, ja es pot projectar sobre el nom de l'estudiant. El resultat final és el pla d'execució lògic següent:



3.3. Implementació física dels operadors relacionals

El sistema de gestió de bases de dades disposa de diversos algorismes per tal de poder implementar físicament els diferents operadors relacionals, és a dir, formes de transformar les diferents fases del pla de consulta lògic en lectures en disc (la frase també es podria utilitzar per a les modificacions de dades). El sistema gestor de bases de dades haurà de triar quina és la millor estratègia física d'implementació per a cada consulta; dit d'una altra manera, haurà de transformar un pla de consulta lògic en un pla de consulta físic.

Per a poder estimar el cost de cadascun dels algoritmes, cal disposar d'alguns estadístics de les taules que intervenen en la consulta, la majoria dels quals son valors mitjans que es mantenen calculats (o es recalculen de tant en tant) en el diccionari de dades. Els estadístics més importants són els següents:

a) Per a cada taula R:

- n(R), nombre de files de la taula R.
- f(R), factor de blocatge de R (el nombre de files de R que caben en una pàgina).
- b(R), nombre de pàgines requerides per a guardar R: b(R) = n(R)/f(R).
- t(R), mida en bytes d'una fila de R.

b) Per a cada atribut A de la taula R:

- V(A,R), nombre de valors diferents de l'atribut A que apareixen a R.
- min(A,R), valor més petit de l'atribut A en la taula R.
- max(A,R), valor més gran de l'atribut A en la taula R.
- CS(A,R), cardinalitat de la selecció de l'atribut. És el nombre mitjà de files que compleixen una condició d'igualtat sobre l'atribut A. Aquest valor serà igual a 1 si A és una clau candidata i igual a n(R)/V(A,R), si els valors estan distribuïts uniformement.

c) Una altra dada important és el nombre de pàgines de memòria, M, que es poden utilitzar com a memòria intermèdia*.

* En anglès, buffer.

d) També és possible, suposant una distribució uniforme, calcular la cardinalitat de selecció per a altres casos diferents de la igualtat:

- Per a $A > c$, $n(r) \times (\max(A,R) - c) / (\max(A,R) - \min(A,R))$.
- Per a $A < c$, $n(r) \times (c - \max(A,R)) / (\max(A,R) - \min(A,R))$.
- Per a $A \subset (c_1, ..., c_n)$, $n(R)/V(A,R)$.
- Per a $A \wedge B$, $SC(A,R) \times SC(B,R)$.
- Per a $A \vee B$, $SC(A,R) + SC(B,R) - SC(A,R) \times SC(B,R)$.

e) Per a cada índex multinivell I d'un atribut A:

- n(A,I), el nombre de nivells en I.
- nfb(A,I), el nombre de pàgines de fulls en I.

A cada operació física se li assigna un cost. Atès que actualment el maquinari no representa cap problema, és fàcil i econòmic incrementar la memòria o les prestacions de la CPU i el cost d'això haurà d'estar relacionat amb el temps de resposta. El temps de resposta inclou molts factors: lectura/escriptura en memòria, temps de procés, lectura del disc, escriptura en disc de resultats intermedis, entre d'altres. Com que aquest valor es fa servir només de forma relativa per a comparar diferents estratègies, s'ha triat el factor que té més pes, el nombre de pàgines de disc llegides en el pitjor dels casos. Quan es considera que no hi ha cap informació en memòria intermèdia, aquest valor s'acostuma a representar per E.

Valoració intuïtiva dels costos

Per a fer una valoració aproximada dels costos , es pot suposar un temps de 150 ns per a la lectura/escriptura de pàgines físicament adjacents, i de 5 ms si s'ha cercar físicament en el disc.

3.3.1. Operacions d'ordenació

Moltes vegades es fa necessari l'ordenació de les dades d'una taula, sigui perquè es vol el resultat ordenat o perquè és una etapa intermèdia dintre algun altre algorisme.

L'ordenació es pot aconseguir amb la creació d'un índex sobre la columna (o columnes) que es vol ordenar i la lectura segons aquest índex. Això presenta el problema que si es vol fer una lectura seqüencial de tota la taula, caldrà fer una lectura en disc per cada fila (ja que aquestes no són consecutives); per tant, moltes vegades interessa més crear una taula temporal amb les files ordenades. Es poden donar les dues situacions següents:

El mètode d'ordenació clàssic més habitual

El mètode clàssic d'ordenació habitual és el d'ordenació ràpida, o quicksort, amb un cost de b(R)).

a) Si tota la taula cap a la memòria primària, el cost és mínim, només cal llegir-la i l'ordenació es fa per qualsevol dels mètodes clàssics.

b) Quan la taula no cap a la memòria primària, s'acostuma a fer servir l'algorisme d'ordenació externa*. Si es disposa de M pàgines de memòria primària disponible per a l'operació, l'algorisme trenca la taula en M taules, ordenades cadascuna d'aquestes amb b(R)/M pàgines, i a continuació va agafant elements de forma ordenada de cada subtaula temporal i redueix el nombre de taules temporals. El cost total d'aquest algorisme d'ordenació vindrà donat per l'expressió següent:

* En anglès, external merge sort.

$$2 \times b(R) \times [(\log_{M-1}(b(R)/M) + 1].$$

Càlcul dels costos amb l'algorisme d'ordenació externa

Considerem una taula de n(R) = 6.000.000 files que ocupa b(R) = 806.250 pàgines en el disc; per tant, f(R) = 6.000.000 / 806.250 = 7,44. Si es disposa de 8 MB de memòria (M = 4.096 pàgines), el cost de fer l'ordenació externa seria de 2.636.584 operacions de lectura/

escriptura. Si se suposa un temps de lectura/escriptura de pàgina d'uns 150 ns, s'haurà trigat un total de $2.636.584 \times 0,00015 = 395$ s (6,6 minuts). Si repetiu el càlcul per a 256 kB de mèmoria, podreu comprovar que el temps és d'uns 11,2 minuts.

3.3.2. Operacions de selecció

Hi ha molts algorismes per a la selecció de valors en una taula. A continuació fem un repàs dels més importants i n'indiquem el cost:

a) La cerca lineal sobre fitxers no ordenats: amb aquest algorisme s'exploren totes les pàgines que formen la taula i es comprova cadascuna de les files per a veure si compta el valor adequat de l'atribut:

- Si es tracta d'un atribut que no és una clau candidata, el cost serà $E = b(R)$.

- Si l'atribut sobre el qual es fa la cerca és una clau candidata, en mitjana el trobarem per la meitat de la taula (de vegades abans i de vegades després); per tant, el cost serà $E = b(R)/2$.

b) La cerca binària sobre fitxers ordenats: consisteix a examinar primer la posició central de la taula, de manera que s'elimina la meitat que no pertoca i es repeteix el procediment fins que es troba el valor cercat:

- Per a trobar un segon atribut que sigui clau, el cost serà $E = \log_2(b(R))$.

- Si es tracta d'un atribut no clau i se suposa una distribució uniforme dels valors, es tindrà $E = \log_2(b(R)) + (n(R)/(V(A,R) \times f(R))) - 1$.

Si la cerca es fa sobre la igualtat d'una clau primària sobre la qual s'ha definit un índex amb estructura de B^+-arbre, el nombre de pàgines llegides serà igual al nombre de nivells de l'arbre més la pàgina de les dades, és a dir: $n(A,I) + 1$. Si es tracta d'un índex que no representa una clau candidata i, per tant, poden existir diferents valors que compleixen la igualtat, per tant, s'hauran de llegir tots les pàgines que els continguin i el cost serà: $n(A,I) + (CS(A,R)/f(R))$.

Ordres de magnitud del cost de les operacions de selecció

Per a tenir una idea de l'ordre de magnitud del cost de les operacions de selecció de valors, se suposarà la taula Ciutada(codi,nom,barri,carrer,dataNaixement) emmagatzemada amb pàgines de 2 Kb, amb els estadístics següents:

- $n(\text{ciutada}) = 2.452.369$
- $f(R) = 13,55$
- $b(R) = 180.987$
- $t(\text{ciutada}) = 115$
- $V(\text{barri,ciutada}) = 26$
- $V(\text{dataNaixement,ciutada}) = 33.864$
- $CS(\text{barri,ciutada}) = 90.828$
- $CS(\text{datanaixement,ciutada}) = 73$
- $n(\text{ciutada},I_{codi}) = 4$
- $nfb(\text{ciutada},I_{codi}) = 18.323$

El nombre de pàgines llegides per a trobar el ciutadà de codi 1.854.347 serà:

- Cerca lineal, E = 180.987/2 = 90.494 pàgines
- Cerca binària, E = $\log_2(180.987)$ = 18 pàgines
- Índex primari, E = 4 + 1 = 5 pàgines

El nombre de pàgines llegides per a trobar els ciutadans del barri B1 serà

- Cerca lineal, E = 180.987 pàgines
- Cerca binària, E = $\log_2(180.987)$ + (2.452.369 / (26 × 13,55)) – 1 = 6.978 pàgines
- Índex secundari, E = 4 + 90.828 / 13,55 = 6.707 pàgines

El nombre de pàgines llegides per a trobar els ciutadans que van néixer a la data D1 serà:

- Cerca lineal, E = 180.987 pàgines
- Cerca binària, E = $\log_2(180.987)$ + (2.452.369 / (73 × 13,55)) – 1 = 2.497 pàgines
- Índex secundari, E = 4 + 73 /13,55 = 10 pàgines

Es pot comprovar que fins i tot per a trobar valors que no siguin clau, el pitjor cost de la utilització de la cerca binària és 25 vegades més petit que el de la lineal; la utilització d'índexs serà molt eficient per a cerques per a clau primària o per a valors que tinguin una cardinalitat de selecció petita.

3.3.3. Operacions de projecció

Una projecció implica dues operacions: eliminar els atributs no desitjats i eliminar (si escau, segons la clàusula DISTINCT) les files repetides que s'han generat en el procés anterior. Els SGBD acostumen a fer aquesta eliminació de repetits en tres fases:

L'eliminació de repetits en SQL

El fet que el cost de l'ordenació sigui molt elevat, ens permet comprendre per què les consultes SQL, per defecte, no eliminen repetits.

1) Lectura seqüencial de la taula, amb un cost b(R), i escriptura en una taula temporal de les files amb només els atributs triats, amb un cost c(R). El cost c(R) serà menor que b(R) i el seu valor dependrà del nombre i de la mida dels camps que es vulguin retenir.

2) Ordenació segons algun algorisme extern, amb el cost previst en l'apartat anterior.

3) Lectura seqüencial de la taula ordenada comparant files contigües i eliminant les duplicades amb un cost de c(R).

Considerem la taula R(nom,dataNaixement,ciutat,sexe,adreça) i els estadístics n(r) = 6.100.000, t(R) = 95, b(R) = 375.810. Si es fa la consulta següent:

```
SELECT DISTINCT Ciutat
FROM R
```

Amb c(R) = 46.162. El cost total serà:

E = 421.972 + 119.210 + 46.162 = 587.344

3.3.4. Operacions de combinació

Les operacions de combinació acostumen a ser les que més temps consumeixen durant el processament de consultes i, per tant, marcaran de manera important l'eficiència global de l'SGBD. Els algorismes més importants d'implementació d'aquesta operació relacional són la combinació de cicle imbricat (nested loop join), la combinació de cicle imbricat indexat (index nested loop join), la combinació per ordenació per fusió (sort-merge join), la combinació per dispersió (hash join) i la combinació per agrupació (cluster).

La combinació de cicle imbricat consisteix en dos cicles imbricats, per a cada fila de la taula exterior R s'obtenen totes les files de la taula interna T i es comprova quines compleixen la condició. El nombre total de pàgines llegides serà $b(R) + b(R) \times f(R) \times b(T)$.

Cost d'aplicació de la combinació de cicle imbricat

Suposem que $b(R) = 500$, $f(R) = 14$, $b(T) = 14.000$ i $f(T) = 11$. Si el llaç intern es fa sobre T:

$$E = 500 + 500 \times 14 \times 14.000 = 98.000.500 \text{ pàgines.}$$

Si es fa sobre R:

$$E = 14.000 + 14.000 \times 11 \times 500 = 77.014.000 \text{ pàgines.}$$

Per tant, sempre serà millor, des d'aquest punt de vista, que el llaç exterior es faci sobre la taula més petita. Observeu la no commutativitat de la fórmula.

En cas que es pugui guardar tota la taula del llaç intern a la memòria primària, o una part, el resultat és molt diferent. En aquest cas, el cost seria:

$$b(R) + b(T).$$

Si a la memòria intermèdia es poden guardar M pàgines, el cost serà:

$$b(R) + b(R) \times b(T) / (M - 2).$$

Exemples de costos amb l'algorisme de combinació de cicle imbricat

Si en l'exemple on s'ha calculat el cost de l'aplicació de la combinació de cicle imbricat supodem que es pot guardar tota la taula en memòria primària, el cost que en resulta és:

$$E = 14.000 + 500 = 14.500 \text{ pàgines.}$$

Si, d'altra banda, només es disposa de memòria intermèdia per a 300 pàgines i el llaç intern es fa sobre T, el cost és el següent:

$$E = 500 + 500 \times 14.000 / (300 - 2) = 24.806 \text{ pàgines.}$$

Si es fa sobre R, resulta un cost de:

$$E = 14.000 + 500 \times 14.000 / (300 - 2) = 38.306 \text{ pàgines.}$$

El segon algoritme d'implementació és la combinació de cicle imbricat indexat. En aquest cas, en el llaç interior no es fa un recorregut sobre totes les pàgines de la taula, sinó que es va directament (mitjançant l'índex o una altra estructura d'accés) a la pàgina adequada.

Per a trobar el cost de la consulta, caldrà conèixer: n(B,I), el nombre de nivells del B^+-arbre de l'índex I sobre la columna B; i CS(B,T), la cardinalitat de selecció de l'atribut B de la taula T i del nombre de files que caben en una pàgina f(T). Si es tracta d'un índex sobre una clau candidata, el cost serà:

$$E = b(R) + b(R) \times f(R) \times (n(B, I) + 1).$$

Si l'índex no correspon a una clau candidata:

$$E = b(R) + b(R) \times f(R) \times (n(B,I) + (CS(B,T) / f(T)).$$

Càlcul de costos amb la combinació de cicle imbricat indexat

Per a comparar amb els mètodes anteriors, es poden fer les suposicions següents, n(A,I) = 2, n(B,I) = 3. Si T és la taula que té l'índex:

$$E = 500 + 500 \times 14 \times (3 + 1) = 28.500 \text{ pàgines.}$$

Si la taula que té l'índex és R:

$$E = 14.000 + 14.000 \times 11 \times (2 + 1) = 476.000 \text{ pàgines}$$

Adoneu-vos que el primer supòsit correspon a un clàssic mestre-detall lligat per una clau forana.

El tercer algoritme, combinació per ordenació per fusió (o combinació per ordenació-barreja), és el més eficient sempre que les dues taules es trobin ordenades; en cas contrari, caldrà un pas previ per a ordenar-les. Una vegada ordenades, només cal anar a llegir seqüencialment cadascuna de les taules i afegir als resultats aquells valors que coincideixen.

Aquest algoritme serà molt eficient, ja que només ha de llegir una vegada totes les pàgines que formen la taula, per tant $E = n(R) + n(T)$. Si s'han d'ordenar prèviament, caldrà sumar -hi el cost d'ordenació.

El quart dels algoritmes de combinació rep el nom de combinació per dispersió i fa servir una funció d'associació (hash) per a dividir les files de totes dues taules en conjunts que tinguin el mateix valor de la funció d'associació, i després es fa la combinació de les files de cada combinació.

El cost d'aquest mena de combinació es pot estimar en:

$$E = 3 \times (n(R) + n(T)) + 2 \times \text{Màxim de particions.}$$

Una estratègia molt utilitzada pels SGBD per a augmentar l'eficiència en les consultes que impliquen combinacions és l'agrupació (clustering); en aquest cas les files de dues taules es guarden agrupades dintre les mateixes pàgines, de manera que quan s'efectua la lectura es va accedint als valors comuns (segons l'índex de l'agrupació (cluster) de les dues taules.

Estratègies de combinació en els SGBD comercials

Els SGBD comercials no suporten tots els algoritmes de combinació: Oracle8ir2 suporta cicle imbricat i ordenació-barreja i dispersió; DB2 i SQL Server suporten cicle indexat, cicle imbricat, ordenació-barreja i dispersió; Informix suporta cicle indexat, cicle imbricat i ordenació per dispersió.

3.4. Optimització física

L'optimitzador físic de consultes és el component de l'SGBD que es dedica a fer que els plans de consulta lògics es duguin a terme de la millor manera possible.

En primer lloc indicarem què vol dir "la millor manera possible". Hi ha diverses possibilitats: que tota la resposta es tingui en el mínim temps; que la primera fila s'obtingui en el mínim temps; que el cost de CPU sigui mínim; que la necessitat de memòria sigui mínima; etc.

Actualment, com que la potència de càlcul és relativament barata, s'acostuma a definir en termes de resposta més ràpida, i molts SGBD permeten de decidir si tot es considera la resposta o només la primera fila. Com ja s'ha indicat, la raó entre les operacions a la memòria i els accessos a disc és d'un ordre físic de magnitud (de mil·lisegons a nanosegons); per tant, minimitzar el temps serà equivalent a minimitzar el nombre d'accessos al disc.

Els algoritmes d'optimització física es poden classificar en dues grans famílies:

1) Optimització heurística: també es coneix com optimització basada en regles. Consisteix a triar un pla d'execució físic amb aquells algoritmes que normalment són més eficients.

2) Optimització basada en costos: consisteix a trobar totes les implementacions físiques; una vegada que es té tot l'espai de possibles realitzacions de la consulta, es calcula per a cada pla el cost (temps, nombre d'accessos a disc, etc.) que tindria i es tria la de cost més petit.

Els SGBD comercials acostumen a permetre de decidir entre una implementació o una altra, encara que cap d'aquests no fa una optimització basada en costos pura, ja que no troben tots els plans possibles, si no que de manera heurística eliminen alguns que, a priori, acostumen a ser poc eficients. La tendència és oferir només optimització basada en costos, ja que encara que l'heu-

rística és molt ràpida de càlcul, pot triar plans d'execució molt poc eficients. Només cal pensar que si augmentem la memòria del nostre servidor, s'aconseguirà que molts del processos de combinació es puguin fer completament a la memòria. Hauran canviat tots el temps d'execució, mentre que els plans d'execució es mantindran.

3.4.1. Optimització física heurística

Es coneix com optimització heurística l'aplicació d'una sèrie de "receptes" que normalment donen bons resultats i que en la majoria dels casos permeten d'obtenir resultats prou bons.

Com tota decisió de disseny, l'optimització heurística té virtuts i defectes: la principal virtut és que no cal efectuar cap mena de càlcul, l'obtenció d'un bon pla d'execució es fa de manera immediata; el principal defecte és que, de vegades, els plans proposats poden diferir molt dels òptims.

Els processos d'optimització

Qualsevol procés d'optimització ha d'estalviar més temps que el que triga el procés en si.

Ja hem vist que hi ha diferents algoritmes per a efectuar les operacions relacionals i que tots aquests tenen costos diferents, lligats tant a les característiques de la informació de les taules i les files que ens demanen (cardinalitat, selectivitat, etc.) com a les característiques físiques de l'equip, sobretot la disponibilitat de memòria intermèdia. El sistema de gestió de bases de dades té definit un ordre d'eficiència per a les diferents operacions lògiques i tria aquell que acostuma a tenir un cost més baix.

Ordre d'eficiència en l'Oracle8ir2

A tall d'exemple es presenta l'ordre de d'eficiència que té definit Oracle8ir2 per a les diferents operacions:

1. Fila única utilitzant RowId
2. Fila única utilitzant Cluster Join
3. Fila única utilitzant Hash Cluster Join amb clau primària o alternativa
4. Fila única per a clau primària o alternativa
5. Clustered Join
6. Clau Hash Cluster
7. Clau de cluster indexat
8. Índexs de columna única
9. Cerca per rang afitat sobre columnes amb índex
10. Cerca per rang sense afitar sobre columnes amb índex
11. Combinació d'ordenació-barreja
12. MAX o MIN en columna indexada
13. ORDER BY de columna indexada
14. Exploració de tota la taula

Per a cada operació, l'SGBD cerca de dalt a baix la primera operació física que pot fer i l'aplica en el pla físic d'execució.

3.4.2. Optimització basada en costos

La tendència actual dels SGBD és utilitzar una política d'optimització basada en costos.

A priori, el procés d'optimització basada en costos és molt fàcil, només cal construir un conjunt amb tots els possibles plans d'execució físics, calcular el cost de cadascun d'aquests i triar el que tingui el cost més reduït.

Però la facilitat aparent d'aquest mètode amaga un problema: per a calcular els costos dels diferents plans, cal tenir informació sobre el contingut de les taules, cardinalitats, mides, nombre de valors, etc. I aquests valors varien contínuament, de manera que es carrega l'SGBD amb la feina afegida de mantenir estadístiques sempre actualitzades sobre el contingut de les taules.

Les estadístiques registren la distribució de les dades i les característiques d'emmagatzematge de les taules, columnes, índexs i particions, a partir de les quals l'optimitzador estima quina quantitat d'accessos a disc i memòria cal per a executar un pla físic particular.

Els paràmetres estadístics que els SGBD guarden més sovint són els següents:

a) Taula: nombre de files; nombre de pàgines; nombre de pàgines buides; llargada mitjana d'una fila.
b) Columna: nombre de valors diferents en la columna; nombre de nuls en la columna; histograma de freqüència d'aparició de cadascun dels valors.
c) Índex: nombre de pàgines fulles; nivells; factor d'agrupament.

Emmagatzematge de les estadístiques

Les estadístiques s'emmagatzemen en el diccionari de dades i des d'aquí es poden traspassar d'una base de dades a una altra; això és útil, per exemple, per a simular en un entorn de proves una situació real.

Un manteniment exhaustiu de les estadístiques és molt feixuc, de manera que només es recullen de tant en tant, i per això s'utilitza la sentència d'SQL ANALYCE, que permet de gestionar les estadístiques sobre qualsevol objecte de la base de dades. Una opció força interessant és la de prendre estadístiques a partir d'una mostra de la taula i no de totes les files.

3.5. L'encarrilament enfront de la materialització

Una estratègia que de vegades es fa servir per a millorar el temps de resposta de les consultes és la de l'encarrilament*.

* En anglès, pipelining.

L'encarrilament consisteix a aprofitar la sortida d'una operació com a entrada de la següent; d'aquesta manera ens estalviem la creació de resultats intermedis en disc (materialització).

Amb l'encarrilament també es pot aconseguir que diverses operacions s'executin alhora, de manera que es guanya velocitat a canvi (com sempre) de necessitar més memòria per a dur a terme els diferents processos.

Amb l'abaratiment de la memòria, una altra opció que fan servir molts optimitzadors és la de materialitzar les taules intermèdies a la memòria; amb això es pot aconseguir una eficiència semblant a la de l'encarrilament.

4. El control de concurrència

A l'actualitat no es pot concebre una base de dades que no permeti la utilització concurrent, és a dir, per diferents usuaris alhora. De la mateixa manera que dues persones poden llegir un mateix diari al mateix temps, no representa cap mena de problema fer diverses consultes alhora a la base de dades; ara bé, si algú ha intentat d'escriure sobre un text quan un altre el llegeix o d'escriure sobre un mateix paper en la mateixa línia mentre que un altre hi escriu, els errors seran inevitables.

Utilització concurrent

S'entén per utilització concurrent l'ús de la mateixa informació per diferents usuaris alhora.

Per a entendre els problemes que dóna la concurrència, es pot comparar el treball d'un SGBD amb el treball en una oficina en què diferents empleats treballen sobre els mateixos documents:

- El problema de l'actualització perduda: dos empleats editen el mateix document amb l'ordinador i cadascun guarda el seu treball de manera independent; el darrer empleat que fa la còpia es "carrega" els canvis que ha fet l'altre, sense que aquest en tingui noticia.

- El problema de la lectura no confirmada: un empleat fa canvis sobre un document i un altre en fa una còpia i la distribueix. El primer empleat, però, no està content amb el resultat del document i n'elimina tots els canvis; la còpia distribuïda conté informació que mai no hauria d'haver sortit.

- El problema de la lectura no repetible: un empleat llegeix dues vegades un document i entre les dues lectures s'hi fan canvis. La segona vegada que l'empleat llegeix el document, el troba diferent.

- El problema de la lectura fantasma: després que el director hagi donat el vistiplau a un document, un empleat hi afegeix informació nova. Pot ser que el document es reparteixi sense que estigui completament supervisat.

Si es canvia l'oficina per un SGBD i els empleats per les diferents transaccions, es tenen els quatre tipus possibles de problemes que presenta el processament simultani.

Transacció

Per tal de poder permetre la utilització concurrent de la base de dades per diferents usuaris, els SGBD implementen diferents mecanismes i estratègies. L'estratègia més comuna és el treball amb transaccions. A Bases de dades II es definia transacció com el conjunt d'operacions de lectura i/o actualització de la base de dades que acaba amb la confirmació o cancel·lació dels canvis que s'han dut a terme.

En afegir l'estratègia de les transaccions al problema de la concurrència, el podem replantejar de la manera següent: les diferents transaccions s'han de poden executar concurrentment, però el seu efecte final sobre la base de dades (afegir, modificar i esborrar dades) haurà de ser el mateix que si s'executessin en sèrie. Aquest criteri rep el nom de seriabilitat.

Exemple de planificació de transaccions

A continuació es presenta un exemple de planificació de dues transaccions. La primera transfereix 1.000 ptes. del compte A al compte B:

```
T0: llegir (A)
    A := A – 1.000
    escriure(A)
    llegir(B)
    B := B + 1.000
    escriure(B)
```

La segona transfereix el 10% del saldo del compte A al compte B:

```
T1: llegir(A)
    temp := 0,1 × A
    A := A –temp
    escriure(A)llegir(B)
    B := B + temp
    escriure(B)
```

Si se suposa que inicialment el compte A disposava de 25.000 ptes. i el compte B de 15.000, el resultat de la planificació en sèrie T_0 seguit de T_1 es pot veure a la taula següent:

T_0	T_1	A	B
llegir(A)		25.000	15.000
A := A – 1000		25.000	15.000
escriure (A,)		24.000	15.000
llegir (B,)		24.000	15.000
B := B + 10.00		24.000	15.000
escriure (B)		24.000	16.000
	llegir(A)	24.000	16.000
	temp := 0.1 × A	24.000	16.000
	A := A – temp	21.600	16.000
	escriure (A)	21.600	16.000
	llegir (B)	21.600	16.000
	B := B + temp	21.600	16.000
	escriure(B)	21.600	18.400

Planificació consistent

Hi ha les mateixes 40.000 ptes. del començament.

En cas que totes dues transaccions s'executin concurrentment, una planificació podria ser, per exemple, la que es veu a la taula següent:

T_0	T_1	A	B
llegir (A)		25.000	15.000
A := A – 1000		25.000	15.000
	llegir(A)	25.000	15.000
	temp := 0.1 × A	25.000	15.000
	A := A – temp	25.000	15.000
	escriure(A)	22.500	15.000
escriure(A)		24.000	15.000
llegir(B)		24.000	15.000
B := B + 1000		24.000	15.000
escriure(B)		24.000	16.000
	llegir(B)	24.000	16.000
	B := B + temp	24.000	16.000
	escriure(B)	24.000	18.500

Planificació inconsistent

Al final hi ha 42.500 ptes., 2.500 ptes. més que al començament.

Hi ha dues grans menes de tècniques per a implementar la seriabilitat: les pessimistes, que es basen a suposar que sempre és possible tenir problemes de concurrència i que, per tant, cal preveure'ls; i les optimistes, que parteixen de la base que, com que hi ha moltes dades, no és usual que es produeixin problemes de concurrència i només caldrà controlar-los després que s'hagi comprovat que se n'ha produït un.

Pessimisme o optimisme?

Com a la vida real, el pessimisme i l'optimisme són apreciacions subjectives.

Les tècniques de control de concurrència amb l'ús de reserves, que ja coneixeu, són de tipus pessimista. Es basen en fer una reserva sobre la informació (grànul) abans de poder-hi operar. Les reserves poden ser de tipus compartit (molts usuaris poden fer aquest tipus de reserva sobre un grànul alhora) o exclusiu (només un usuari la pot fer). Abans de llegir, una transacció haurà d'haver efectuat una reserva compartida i, abans d'escriure, una d'exclusiva. Quan ja no necessita els recursos, els allibera.

Vegeu les tècniques de control de concurrència a l'assignatura Bases de dades II.

El planificador pot fer servir diferents protocols per a assegurar la seriabilitat de les transaccions que utilitzen reserves. El més comú es c`oneix com protocol de reserves de dues fases, que requereix que cada transacció faci les reserves i l'alliberament en dues fases: una de creixement, en la qual pot fer reserves però no alliberaments; i una de decreixement, en la que es poden fer alliberaments, però no reserves.

Exemple d'aplicació del protocol de reserves de dues fases

En l'exemple de planificació de transaccions presentat abans, l'alicació del protocol de reserves de dues fases dóna el resultat següent:

T_0	T_1	A	B
Comença T_0		25.000	15.000
lock(A,X)		25.000	15.000
llegir(A)		25.000	15.000
	Comença T_1	25.000	15.000
	lock(A,X)	25.000	15.000
A := A − 1000	ESPERA	25.000	15.000
escriure(A)	ESPERA	24.000	15.000
lock(B,X)	ESPERA	24.000	15.000
unlock(A)	ESPERA	24.000	15.000
	llegir (A)	24.000	15.000
	temp := 0.1 × A	24.000	15.000
	A := A − temp	24.000	15.000
	escriure(A)	24.000	15.000
llegir(B)		21.600	15.000
B := B + 1000		21.600	15.000
escriure(B)	lock(B,X)	21.600	15.000
unlock(B)	ESPERA	21.600	16.000
	llegir(B)	21.600	16.000
	unlock(A)	21.600	16.000
	B := B + temp	21.600	18.400
	escriure(B)	21.600	18.400
	unlock(B)	21.600	18.400

La funció lock(A,X) indica una reserva exclusiva del recurs A, i la funció unlock(A) elimina la reserva sobre A.

Amb aquest resultat es pot comprovar com es manté la base de dades en un estat consistent, al final continua havent-hi les mateixes 40.000 ptes. del principi.

4.1. Control de concurrència basat en marques temporals

Un mètode de tipus optimista, és a dir, que pressuposa que hi haurà pocs conflictes entre transaccions, és la utilització de marques temporals.

El mètode de les marques temporals consisteix a assignar una marca temporal MT(T) abans que es comenci a executar una transacció T. Aquestes marques són úniques i s'assignen en ordre ascendent, de manera que qualsevol transacció posterior T' es complirà:

$$MT(T') > MT(T).$$

A més, a cada element d'informació (grànul) X se li assignaran dues marques temporals i un bit:

Assignació de marques temporals

Hi ha dos mecanismes per a assignar marques temporals:

- Utilitzar el rellotge del sistema assignant com a marca de cada transacció l'hora en què entra al sistema.
- Utilitzar un comptador lògic, que s'incrementi d'un en un cada vegada que s'assigni una marca a una transacció.

- TL(X): temps de lectura, que correspon a la marca temporal més gran (la darrera) de totes les transaccions que ha llegit X.
- TE(X): temps d'escriptura, que correspon a la marca temporal més gran (la darrera) de totes les transacció que han escrit en X;
- C(X): bit de confirmació, que serà cert (1) només si la darrera transacció que ha escrit en X ja està confirmada o ha avortat i, per tant, compta com si no hagués escrit.

També caldrà mantenir una llista amb tots els elements escrits per X.

A continuació presentem els problemes, causats per la concurrència, que es poden produir.

a) Llegir massa tard: la transacció T_i prova de llegir l'element X, però aquest ha estat modificat per una transacció T_j que ha començat després de X.

Problema per llegir massa tard

Considerem el cas següent:

$$MT(T_i) = 22.515,\ MT(T_j) = 22.623$$

T_J escriu en l'element X, TE(X) = 22.623. T_i vol llegir X, però troba que:

$$MT(T_i) = 22.515 < TE(X) = 22.623.$$

Com que teòricament T_i s'ha de fer abans que T_j, no pot llegir el valor i s'haurà de desfer.

b) Escriure massa tard: la transacció T_i prova d'escriure l'element X, però aquest ja ha estat llegit per una transacció T_j que ha començat després de X.

Problema per escriure massa tard

Considerem l'exemple següent:

$$MT(T_i) = 22.800,\ MT(T_j) = 22.945$$

T_J llegeix l'element X, TL(X) = 22.945. T_i vol escriure X, però troba que:

$$MT(T_i) = 22.800 < TL(X) = 22.945$$

Com que teòricament T_i s'ha de fer abans que T_j, no pot escriure el valor i s'haurà de desfer.

c) Lectura no confirmada: la transacció T_i escriu a l'element X, després de l'escriptura comença la transacció T_j, que llegeix l'element X; després d'aquesta lectura, la transacció T_i és avortada i el valor llegit per T_j ja no té sentit.

Perquè no es produeixin els problemes anteriors, el planificador podrà prendre davant de qualsevol petició de lectura o escriptura per part d'una transacció T una de les accions següents:

- Autoritzar la petició.
- Avortar T i recomençar T amb una nova marca temporal.
- Aturar T i més tard decidir si avorta T o si autoritza la petició.

Les normes per a prendre aquesta decisió són les següents:

- Quan el planificador rep una petició de lectura L(T,X) per part de la transacció T sobre l'element X:

```
si MT(T) ≥ TE(X) llavors
    si C(X) := cert llavors
        Autoritzar la petició
        si MT(T) > TL(X) llavors
            TL(X) := MT(T)
        fsi
    sino
        Aturar T fins que C(X) esdevingui cert
    fsi
sino
    Avortar T
    Recomençar T amb una nova marca temporal
fsi
```

- Quan el planificador rep una petició d'escriptura, E(T,X), per part de la transacció T sobre l'element X:

```
si MT(T) ≥ TL(X)  i MT(T) ≥ TE(X) llavors
    si C(X) = cert
        Autoritzar la petició
        TE(X) := MT(T)
        C(X) := fals
    sino
        Aturar T fins que C(X) esdevingui cert
    fsi
sino
    Avortar T
    Recomençar T amb una nova marca temporal
fsi
```

- Quan el planificador rep una petició per a confirmar T:

```
per tots els elements X escrits per T C(X) := cert fer
    continuar les transaccions que estaven esperant
fper
```

- Si el planificador rep una ordre d'avortar, o la pren com a efecte de les regles anteriors:

```
per tots els elements X escrits per T fer
    continuar les transaccions que estaven esperant
fper
```

Aplicació de l'estratègia de les marques temporals

Si s'aplica l'estratègia de les marques temporals per al control de la concurrència a les dues transaccions de l'exemple de planificació de transaccions vist abans, per simplificar se suposarà que des que ha començat el planificador, no s'ha efectuat cap lectura ni escriptura sobre A i B i, per tant, no tenen assignades marques temporals.

Vegeu l'exemple de planificació de transferències al principi d'aquest apartat.

Marques temporals d'inici de transacció:

MT(T0) = 23.548, MT(T1) = 23.687.

T_0	T_1	A	B	TL(A)	TE(A)	C(A)	TL(B)	TE(B)	C(B)
Comença T_0		25.000	15.000	0	0	1	0	0	1
llegir(A)		25.000	15.000	23.548	0	1	0	0	1
	Comença T_1	25.000	15.000	23.548	0	1	0	0	1
A := A – 1000	ESPERA	25.000	15.000	23.548	0	0	0	0	1
escriure(A)	ESPERA	24.000	15.000	23.548	23.548	0	0	0	1
llegir(B)	ESPERA	24.000	15.000	23.548	23.548	0	23.548	0	1
B := B + 1000	ESPERA	24.000	15.000	23.548	23.548	0	23.548	0	1
escriure(B)	ESPERA	24.000	16.000	23.548	23.548	0	23.548	23.548	0
COMMIT	ESPERA	24.000	16.000	23.548	23.548	1	23.548	23.548	1
	llegir (A)	24.000	16.000	23.687	23.548	1	23.548	23.548	1
	temp := 0.1 × A	24.000	16.000	23.687	23.548	1	23.548	23.548	1
	A := A – temp	21.600	16.000	23.687	23.548	1	23.548	23.548	1
	escriure(A)	21.600	16.000	23.687	23.687	0	23.548	23.548	1
	llegir(B)	21.600	16.000	23.687	23.687	0	23.687	23.548	1
	B := B + temp	21.600	16.000	23.687	23.687	0	23.687	23.548	1
	escriure(B)	21.600	18.400	23.687	23.687	0	23.687	23.687	0
	COMMIT	21.600	18.400	23.687	23.687	1	23.687	23.687	1

Es pot veure com la segona transacció està en espera fins que ha finalitzat la primera i ja no hi ha possibles problemes de concurrència.

Hi ha una variant molt emprada de l'algorisme anterior, coneguda com la regla d'escriptura de Thomas. Aquest algorisme modifica lleugerament l'escriptura, de manera que en cas que es vulgui escriure un valor que una transacció posterior ja ha escrit, l'execució de la transacció no s'atura, sinó que s'ignora l'operació d'escriptura i es continua amb la transacció. Això es pot fer perquè el valor resta obsolet quan una transacció posterior el modifica.

4.2. Control de concurrència per validacions

En molts entorns, els conflictes de concurrència entre transaccions són molt estranys i resulta molt feixuc i poc eficient mantenir-hi un control de concurrència. Els mètodes optimistes només actuen sobre aquelles transaccions que potser poden entrar en conflicte amb d'altres.

Transaccions amb els mètodes optimistes

Els mètodes optimistes, en lloc d'anar fent comprovacions a l'inici i al llarg de la transacció, només fan una comprovació al final, just abans d'escriure. Si no hi ha problemes, s'escriuen els resultats; si passa altrament, es reinicia la transacció.

Problemes per conflictes de concurrència

Reprenguem l'exemple presentat per a il·lustrar la planificació de transferències, quantes vegades en un banc dues transaccions (bancàries o informàtiques) intentaran de llegir o escriure de manera concurrent les mateixes dades? Penseu com hauria de ser un procediment bancari que provoqués problemes d'aquest tipus. També podeu pensar en els problemes que es podrien crear si no es controlés la concurrència.

Els mètodes optimistes no són la panacea universal, ja que les estratègies que fan servir per a validar les transaccions de vegades fan desfer (ROLLBACK) tran-

saccions que no presenten cap mena de problema. S'acostumen a utilitzar en entorns en què el perill de concurrència és petit i les transaccions són curtes.

El mètode de les validacions és una implementació especial de les marques temporals en la qual el planificador manté informació sobre el que fan les transaccions. Per a cadascuna de les transaccions T manté un conjunt CL(T), un conjunt de lectura de transaccions amb totes les dades de la base de dades que haurà de llegir, i un altre CE(T), conjunt d'escriptura de transaccions amb totes les que haurà d'escriure.

Les transaccions s'hauran d'executar de manera seqüencial seguint les tres fases següents:

1) Fase de lectura: durant aquesta fase la transacció llegeix de la base de dades tota la informació que hi falta, fa tots els càlculs i en guarda els resultats a la memòria.

2) Fase de validació: en aquesta fase el planificador comprova si pot copiar en la base de dades els resultats que ha calculat a la fase anterior sense produir problemes de violació de la seriabilitat. Per això compara els conjunts de lectura i escriptura amb els de les altres transaccions: si no provoca problemes es passa a la següent fase, en cas contrari, es reinicia la transacció.

Transaccions amb una sola fase

Pot ser que hi hagi transaccions tals que CE(T) = ∅, és a dir, que no han d'escriure res a la base de dades. Aquestes transaccions només tindran la primera fase.

3) Fase d'escriptura: en aquesta fase els resultats que es guardaven en memòria s'escriuen a la base de dades.

Per a poder fer la validació, el planificador associa tres marques temporals a cada transacció T:

- I(T), moment en què s'inicia l'execució de la transacció T.
- V(T), moment en què finalitza la fase de lectura de T i comença la de validació.
- F(T), moment en què finalitza la fase d'escriptura de T.

A partir dels valors anteriors, el planificador manté tres conjunts de transaccions:

La marca V(T)

La marca V(T) indica al planificador la seqüencialitat de les transaccions, és a dir, que l'estat resultant de la base de dades és com si s'haguessin realitzat seqüencialment segons l'ordre de V(T).

1) INIC: conjunt de totes les transaccions que han estat iniciades, que només tenen la marca temporal I(T).

2) VAL: conjunt de totes aquelles transaccions que ja s'han validat, però que encara no han finalitzat, que només tenen les marques temporals I(T) i V(T).

3) FIN: conjunt de totes les transaccions que ja han finalitzat la fase d'escriptura, que tenen les tres marques temporals I(T), V(T) i F(T).

A partir d'aquesta informació és possible assegurar la seriabilitat de les transaccions; les regles que haurà de seguir el planificador són les següents:

1) Per a validar una transacció T_n s'ha de complir, al menys una de les condicions següents:

- $\forall T \in$ FIN es compleix que $F(T) < I(T_n)$. La transacció T ja ha finalitzat quan ha començat T_n.
- $\forall T \in$ VAL es compleix que $CL(T_n) \cap CE(T) = \varnothing$. Si una transacció encara no ha escrit, no ha d'escriure res que T_n hagi llegit.
- $\forall T \in$ INIC es compleix que $CL(T_n) \cap CE(T) = \varnothing$ i $CE(T_n) \cap CE(T) = \varnothing$. Si una transacció encara no ha llegit, no ha de poder crear cap problema de concurrència.

2) Si no es compleix cap de les condicions anteriors, la transacció T_n es reiniciarà.

El conjunt FIN

El conjunt FIN hauria d'anar creixent indefinidament si no existís algun mecanisme de purga, ja que només interessa mantenir informació sobre aquelles transaccions que encara poden donar problemes. Si es defineix MIN_IT com la més petita de les marques temporals d'iniciació de transacció del conjunt VAL, això ens donarà el moment d'iniciació de la transacció més "antiga" que encara no han superat la fase d'escriptura. Totes aquelles transaccions tals que F(T) < MIN_IT, es podran eliminar del conjunt FIN.

4.3. Control de concurrència multiversió

Totes les estratègies presentades fins ara es basen a aturar o desfer l'execució d'una transacció per tal d'aconseguir que els resultat siguin equivalents al de l'execució seqüencial de les transaccions.

Una manera diferent d'aconseguir el mateix és la tècnica de mantenir diverses versions de les dades, una de nova cada vegada que s'escriu. Quan una transacció vol llegir les dades, el planificador haurà de saber quina és la versió adequada. Aquestes estratègies reben el nom de multiversió.

Sobre aquest sistema es poden aplicar diferents mecanismes i els més usuals són els de marques temporals i els de reserva de doble fase.

4.3.1. Control per marques temporals

En la tècnica de control per marques temporals, com en el cas del control per marques temporals normal s'assigna una marca temporal MT(T) a cada transacció T. A cada element d'informació X se li assigna una seqüència de versions $<V_1(X), ..., V_n(X)>$, cadascuna formada per tres elements:

1) $Cont(X_j)$ = Valor de la versió j del gránul X.

- $TL(X_j)$ = Marca temporal de la transacció que hagi creat la versió.

- $TE(X_j)$ = La més gran de les marques temporals de totes les transaccions que han llegit amb èxit la versió j.

El planificador utilitza l'estratègia següent per a una transacció T:

```
sigui TE(Xk) la més gran de les MT que compleix que TE(Xk) ≤ MT(T)
retornar Cont(Xk)
fsi

si una transacció T vol escriure X  llavors
    si MT(T) < TL(Xk) llavors
        ROLLBACK T
    fsi
    si MT(T) = TL(Xk) llavors
        sobreescriure el contingut de la versió
    fsi
    si MT(T) > TL(Xk) llavors
        crear una nova versió de X
    fsi
fsi
```

Es pot comprovar...

... que en cap cas fallarà una lectura. Si una transacció T vol llegir X, cal desfer la transacció en comptes de mantenir-la en espera.

Per a eliminar les versions que no siguin necessàries es fa servir la regla següent: només cal mantenir la darrera de les versions que sigui més antiga que la més antiga de les transaccions del sistema.

Aquest mecanisme és molt eficient per a aquells sistemes en els quals les lectures són molt més abundants que les escriptures.

4.3.2. Control per reserva de doble fase

En aquest esquema, cada grànul d'informació pot estar sotmès a tres tipus de reserves diferents: a part de les dues reserves clàssiques (compartida i exclusiva), hi ha un tercer tipus de reserva anomenada reserva de certificació. La taula de compatibilitat de reserves es presenta a continuació:

	Lectura	Escriptura	Certificació
Lectura	Sí	Sí	No
Escriptura	Sí	No	No
Certificació	No	No	No

Conmpatibilitat de la reserva de certificació

Sorprenentment, es pot comprovar que una reserva de lectura és compatible amb una d'escriptura, mentre que el nou tipus de reserva, la de certificació, no és compatible amb cap altra.

Per a permetre l'aplicació del mètode de reserves de doble fase, es fa necessari mantenir dues versions de cada grànul: una única versió pot tenir una reserva de lectura sobre un grànul X, mentre que moltes altres el poden llegir. Per-

què no es produeixin problemes de concurrència, hi ha d'haver dues versions de X: una, X_c –que haurà d'haver estat escrita per una transacció confirmada– i una altra, X_p –escrita per la transacció que hagi fet la reserva d'escriptura; qualsevol de les transaccions que vulgui fer una lectura, amb reserva prèvia, llegirà el valor confirmat X_c. La transacció que ha fet la reserva d'escriptura podrà efectuar tots els canvis que desitgi sobre la seva versió de la informació sense que això provoqui cap mena de problema a les altres transaccions.

Quan la transacció que té la reserva d'escriptura vulgui fer la confirmació (COMMIT), prèviament haurà d'aconseguir una reserva de certificació, la qual no és compatible amb la de lectura; per tant, haurà d'esperar que totes les transaccions que fan lectures alliberin el recurs. En aquest moment, i de manera incompatible amb cap altre tipus de reserva, es copiarà el valor S_p en S_c, en el qual s'eliminarà, i ja es podrà alliberar la reserva de certificació.

Aquesta variant de l'estratègia de la reserva de doble fase aconsegueix que diverses transaccions puguin llegir un valor que ja ha estat reservat per a escriptura per una altra transacció, al cost que aquesta darrera ha d'esperar per a fer la confirmació.

4.4. Control de concurrència en SGBD comercials

A continuació es presenta de manera resumida els mecanismes que fan servir els principals sistemes gestors de bases de dades comercials. Encara que en alguns casos no coincideix exactament amb la nomenclatura utilitzada fins aquest moment, s'ha preferit conservar la notació de cada fabricant.

Podreu comprovar com cada SGBD utilitza diferents i múltiples estratègies amb la finalitat d'assolir el més gran rendiment possible.

4.4.1. Oracle8i

La versió 8.1.6 de l'SGBD Oracle8i pot fer diferents tipus de reserva; les més usuals són les següents:

a) Reserva de taula amb fila compartida(RS): és la menys restrictiva, permet que altres transaccions modifiquin altres files de la taula i que qualsevol transacció llegeixi tota la taula, al temps que impedeix modificacions en l'estructura de la taula.

b) Reserva de taula amb fila exclusiva (RX): manté la reserva absoluta sobre les files i així impedeix altres lectures i escriptures i assegura que no es canviarà l'estructura de la taula. També fa altres tipus de reserva en els nivells de diccionari, de fitxer, d'espai de taula, de segment de rollback.

c) Reserva de taula compartida (S): fa una reserva compartida per tota la taula.

d) Reserva de taula compartida amb fila exclusiva (SRX): fa una reserva exclusiva per a la fila i compartida per a la taula.

e) Reserva exclusiva (X): fa una reserva exclusiva per a tota la taula.

Encara que la forma normal de reserva és la implícita (l'SGBD la du a terme d'acord amb l'operació que fa), també permet d'efectuar diferents tipus de reserves explícites.

Oracle efectua el control de la concurrència amb mecanismes de reserva de doble fase multiversió: manté en els seus segments de rollback la versió provisional de les dades fins que es confirma. Quan una transacció vol llegir un gränul i es troba que aquest ha estat creat per una transacció més jove, el sistema cerca pels segments de rollback una versió més antiga de la informació que ja estigui confirmada.

Nivells d'aïllament amb Oracle

Oracle permet d'assignar tres dels quatre nivells d'aïllament definits per SQL:1999: READ COMMITED, READ UNCOMMITED i SERIALIZABLE.

4.4.2. DB2

DB2 permet de fer diferents tipus de reserves sobre diferents recursos:

a) No-intenció (IN): es pot aplicar sobre espai de taules i taules; el propietari de la reserva pot llegir qualsevol dada de la taula, incloent-hi les dades no confirmades, però no les pot actualitzar. Altres transaccions concurrents poden llegir o actualitzar les dades.

b) Intenció compartida (IS): es pot aplicar sobre espais de taules i taules; el propietari de la reserva pot llegir totes les dades de la taula reservada, però no pot efectuar cap actualització. La transacció acostuma a fer una reserva de tipus S o NS sobre cada fila llegida, sobre les quals una altra transacció no podrà escriure.

c) Següent clau compartida (NS): s'aplica sobre files; el propietari de la reserva i totes les altres transaccions poden llegir la fila reservada, però no la poden actualitzar. Aquesta reserva s'efectua sobre les files d'una taula, en comptes d'un de tipus S, quan el nivell d'aïllament és RS o CS.

d) Compartida (S): s'aplica sobre files i taules; el propietari de la reserva i totes les altres transaccions concurrents poden llegir el gränul reservat, però no el poden actualitzar.

e) Intenció exclusiva (IX): es pot aplicar sobre espais de taules i taules; el propietari de la reserva i les altres transaccions concurrents poden llegir i actualitzar les dades de la taula. Quan una transacció llegeix una fila, fa una reserva del tipus S, NS, X o U; si vol fer una actualització, durà a terme una reserva del tipus X.

f) Compartida amb intenció exclusiva (SIX): s'aplica només sobre taules; la transacció que fa la reserva pot efectuar lectures i actualitzacions sobre la taula amb una reserva X prèvia; les altres transaccions podran llegir la taula.

g) Actualització (U): s'aplica sobre taules i files; la transacció que fa la reserva pot actualitzar el recurs reservat, sobre el qual prèviament tindrà una reserva X; les altres transaccions no podran llegir les dades.

h) Següent clau exclusiva (NX): s'aplica només sobre files; la transacció que fa la reserva podrà llegir la fila reservad, però nola podrà actualitzar.

i) Següent clau exclusiva feble (NW): s'aplica només sobre files; la reserva es fa sobre la fila següent quan una fila s'insereix a l'índex d'una taula. La transacció que la fa pot llegir la fila reservada, però no la pot actualitzar.

j) Exclusiva (X): s'aplica sobre taules i files; la transacció que fa la reserva pot efectuar lectures i actualitzacions sobre la taula; només les transaccions UR poden tenir accés a les files reservades.

k) Exclusiva feble (W): s'aplica només sobre files; la reserva es fa quan s'insereix una fila. Només les transaccions UR poden tenir accés a les files reservades.

l) Superexclusiva (Z): s'aplica sobre espais de taules i taules; tipus de reserva especial efectuat sobre una taula quan aquesta s'ha d'eliminar, reorganitzar o se n'ha de crear o eliminar un índex.

Nivells d'aïllament amb DB2

DB2 permet de controlar els quatre nivells d'aïllament definits per SQL:1999: *READ COMMITED*, *READ UNCOMMITED*, *REPEATIBLE READ* i *SERIALIZABLE*.

La taula següent presenta la compatibilitat que hi ha entre els diferents tipus de reserva:

	IN	IS	NS	S	IX	SIX	U	NX	X	Z	NW	W
IN	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí
IS	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No	No	No	No	No
NS	Sí	Sí	Sí	Sí	No	No	Sí	Sí	No	No	Sí	No
S	Sí	Sí	Sí	Sí	No	No	Sí	No	No	No	No	No
IX	Sí	Sí	No	No	Sí	No	No	No	No	No	No	No
SIX	Sí	Sí	No	No	No	No	No	No	No	No	No	No
U	Sí	Sí	Sí	Sí	No	No	No	No	No	No	No	No
NX	Sí	No	Sí	No`	No	No	No	No	No	No	No	No
X	Sí	No	No	No	No	No	No	No	No	No	No	No
Z	No	No	No	No	No	No	No	No	No	No	No	No
NW	Sí	No	Sí	No	No	No	No	No	No	No	No	Sí
W	Sí	No	No	No	No	No	No	No	No	No	Sí	No

DB2 fa servir com a estratègia el mètode de les reserves de doble fase combinant i escalant el diferents tipus de reserva.

4.4.3. SQL Server

La versió 2000 de l'SGBD Microsoft SQL Server permet de fer reserves sobre diferents recursos:

a) RID: identificador de fila, reserva una única fila d'una taula.

b) Clau: reserva de fila dintre d'un índex, es fa servir per a protegir intervals de claus en transaccions serialitzables.

c) Full: full de dades o d'índexs de 8 kB.

d) Extensió: grup contingut de fulls de dades o índex, es fa servir durant l'assignació d'espai.

e) Taula: taula completa, incloent-hi les dades i l'índex.

f) Base de dades: tota la base de dades, es fa servir durant la restauració d'una base de dades.

SQL Server fa servir els dos tipus de reserves bàsiques: compartit (S) i exclusiu (X), més uns altres quatre tipus per a situacions especials:

a) Reserva per intenció: es fa servir per a establir una jerarquia de reserves; si una transacció té una reserva exclusiva per a fila, impedeix que una altra l'adquireixi per a full. Les reserves d'intenció poden ser: compartida (IS), exclusiva (IX) i compartida amb reserva exclusiva d'intenció (SIX):

b) Reserva d'actualització (U): la fa servir quan vol modificar un full de la base de dades per a impedir problemes d'abraçada mortal.

c) Reserva d'esquema: impedeix que s'elimini una taula o un índex, o que es canviï l'estructura de la taula. Hi ha dues versions: estabilitat de l'esquema (Sch-S), que assegura que no s'eliminarà la taula; modificació de l'esquema (Sch-M), que assegura que no es modificarà la taula.

d) Reserva d'actualització massiva (BU): s'utilitza per a fer còpies massives de dades sobre una taula.

A continuació es presenta la matriu de compatibilitat entre els tipus de reserves més importants:

	IS	S	U	IX	SIX	X
IS	Sí	Sí	Sí	Sí	Sí	No
S	Sí	Sí	Sí	No	No	No
U	Sí	Sí	No	No	No	No
IX	Sí	No	No	Sí	No	No
SIX	Sí	No	No	No	No	No
X	No	No	No	No	No	No

Nivells d'aïllament amb SQL Server

SQL Server permet de controlar els quatre nivells d'aïllament definits per SQL:1999 (READ COMMITED, READ UNCOMMITED, REPEATIBLE READ i SERIALIZABLE) i el temps màxim que una transacció pot esperar per a alliberar un recurs.

Per a controlar els problemes de concurrència, l'SGBD disposa de diferents estratègies, tant pessimistes (reserva de doble fase) com optimistes (multiversió per marques temporals i per valors; aquest mètode compara el valor actual d'una dada amb el que tenia la darrera vegada que es va llegir); l'estratègia triada dependrà del nivell d'aïllament i de la simultaneïtat que triï l'usuari.

4.4.4. Informix

La versió 9.2 de l'SGBD Informix Dynamic Server:2000 (actualment propietat d'IBM) permet de fer reserves sobre una única fila o índex, sobre un full de dades o índex, sobre una taula o sobre tota la base de dades. També permet de determinar el temps que una transacció que no pot efectuar una reserva espera abans d'enviar un error a l'aplicació.

Amb Informix es poden fer els següents tipus de reserves sobre una taula:

a) Compartida (S): permet de llegir a la transacció que fa la reserva i a qualsevol altra.
b) Exclusiva (X): només permet de llegir i d'actualitzar la transacció que fa la reserva.
c) Actualització (U): permet de fer la lectura de la resta de la taula i actualitzar algunes files.
d) De byte (B): s'aplica quan es vol reduir la mida d'una dada del tipus cadena de caràcters de llargada variable.

Nivells d'aïllament amb SQL Server

SQL Server 2000 permet de controlar els quatre nivells d'aïllament definits per SQL: 1999: SERIALIZABLE, READ COMMITED, READ UNCOMMITED i REPEATIBLE READ.

Resum

Els mecanismes bàsics de seguretat són la identificació i autenticació, el control d'accés, la integritat i consistència, i l'auditoria. El control d'accés es pot implementar seguint dues polítiques diferents: discrecional –cada usuari té uns permisos sobre cada recurs–, que és el sistema que utilitzen la major part dels SGBD comercials; i obligatori –els recursos i els usuaris estan jerarquitzats. S'ha de tenir una cura especial amb totes aquelles dades de caràcter personal que estan protegides per la llei.

Els SGBD, a part de les taules bàsiques, mantenen i/o utilitzen diferents tipus de taules derivades, el més conegut dels quals és la vista. Altres tipus són els snapshots i les taules temporals. Una vista es pot considerar com el resultat d'una consulta i es pot implementar amb materialitzacions o amb reescriptura de consultes. Amb l'ús de disparadors de substitució es pot fer que totes les vistes siguin actualitzables.

L'SGBD haurà de passar d'una consulta no procedimental (normalment en SQL) a una sèrie de lectura/escriptures en disc, procés que rep el nom de processament. La consulta SQL, una vegada comprovada lèxicament i feta, si escau, l'optimització semàntica, es transforma en un pla d'execució lògic basat en l'àlgebra relacional, el qual caldrà optimitzar. El pas següent serà decidir quina és la millor estratègia d'implementació física(la de cost més petit), a partir dels diferents algoritmes d'accés als fitxers que permeti l'SGBD.

La utilització concurrent de la informació és una de les funcions més importants de l'SGBD. A part de les polítiques pessimistes ja estudiades (sempre cal preveure que es produeixin conflictes), hi ha altres mètodes que només es preocupen dels possibles problemes per concurrència després que es produeixin. Els més utilitzats es basen en marques temporals o en l'existència de diferents versions de la informació.

Activitats

En els aspectes tractats en aquest mòdul, les diferents implementacions comercials dels SGBD presenten força diferències. Aneu a les webs dels principals fabricants i llegiu-ne la documentació específica de cadascun.

a) Oracle8ir2 (cal donar-se d'alta, és gratuït):

http://otn.oracle.com/docs/products/oracle8i/doc_index.htm

b) SQL Server 2000

http://msdn.microsoft.com/sqlserver/

Cerqueu l'apartat de documentació.

c) DB2 7.1

http://www-4.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/

support/v7pubs.d2w/en_main

d) Informix Dynamic Server 2000. Darrerament, aquest SGBD ha estat venut a IBM, i la seva tecnologia s'incorporà a DB2:

http://www.informix.com/answers/english/pids92.htm

Exercicis d'autoavaluació

1. Suposeu que sou l'administrador de la base de dades i creeu un rol nou. Doneu permís als membres d'aquest rol per tal de poder consultar la taula Empresa de l'esquema Facturació, i control total sobre les taules Factura i Albara de l'esquema Comptabilitat. Després, creeu dos usuaris i assigneu-los el rol anterior.

2. Donades les taules següents, creeu una vista que inclogui el DNI, el nom, la qualificació i la nota de tots els estudiants de l'SGBD. Creeu un disparador de substitució que a partir del DNI permeti de modificar la qualificació i la nota d'un estudiant.

Estudiant(DNI, nom, localitat)

Assignatura(codi,nom)

Obtenir(estudiant,assignatura,qualificacio,nota)

On Estudiant referencia a la taula Estudiant(DNI) i Assignatura referencia a Assignatura(codi)

3. Donades les taules anteriors i amb els estadístics següents (el sistema fa servir pàgines de 2 kB):

	Estudiant	Obtenir	Assignatura
n(R)	75000	2625000	1400
f(R)	35	70	88
b(r)	2142	37500	16

		V(A, R)
Estudiant	Dni	75000
	Nom	73000
	Localitat	2000
Obtenir	Estudiant	7500
	Assignatura	1400
	Qualificació	6
Assignatura	Nota	21
	Codi	1400
	Nom	1400

a) Trobeu el pla de consulta lògic:

```
SELECT DISTINCT A.Nom
FROM Estudiant E, Obtenir O, Assignatura A
WHERE O.Estudiant = E.Dni
   AND O.Assignatura = A.Codi
   AND E.Localitat = 'Valls'
   AND O.Qualificació = 'Notable'
   AND O.Nota = 7,5
```

b) Si la memòria interna (RAM) que pot utilitzar per a aquesta consulta és de 5 pàgines, quina serà l'estratègia física d'implementació de la combinació entre la taula Obtenir i la taula Estudiant?

4. Presenteu una planificació de les dues transaccions descrites en l'apartat 4 utilitzant l'estratègia multiversió de doble fase.

Solucionari

1. Creem el rol Comptable:

```
CREATE ROLE Comptable;
```

Li donem els permisos:

```
GRANT SELECT ON Facturacio.Empresa TO Comptable;
GRANT ALL PRIVILEGES ON Comptabilitat.Factura TO Comptable;
GRANT ALL PRIVILEGES ON Comptabilitat.Albara TO Comptable;
```

Creem usuaris

```
CREATE SCHEMA Usuari1 AUTHORIZATION Usxxxi1;
CREATE SCHEMA Usuari2 AUTHORIZATION Usyyyi1;
```

Els assignem el rol:

```
GRANT Comptable TO Usuari1, Usuari2;
```

2. Creem la vista:

```
CREATE VIEW Resultats(Dni, Nom, Qualificació, Nota) AS
SELECT E.Dni, E.Nom, O.Qualificacio, O.Nota
FROM Estudiant AS E, Assignatura AS A, Obtenir AS O
WHERE O.Estudiant = E.Dni
AND O.Assignatura = A.Codi
AND A.Nom = 'SGBD'
```

Creem el disparador:

```
CREATE TRIGGER ActualitzarResultats
INSTEAD OF UPDATE ON Resultats
DECLARE
v_Assig     VARCHAR2(50);

BEGIN
SELECT Codi INTO v_Assig
FROM Assignatura
WHERE Nom = 'SGBD';

UPDATE Obtenir
SET Qualificacio = :NEW.Qualificacio
WHERE Estudiant = :OLD.Dni
AND Assignatura = v_Assig;

UPDATE Obtenir
SET Nota = :NEW.Nota
WHERE Estudiant = :OLD.Dni
AND Assignatura = v_Assig;

END ActualitzarResultats;
```

3.

a) Donada la consulta següent:

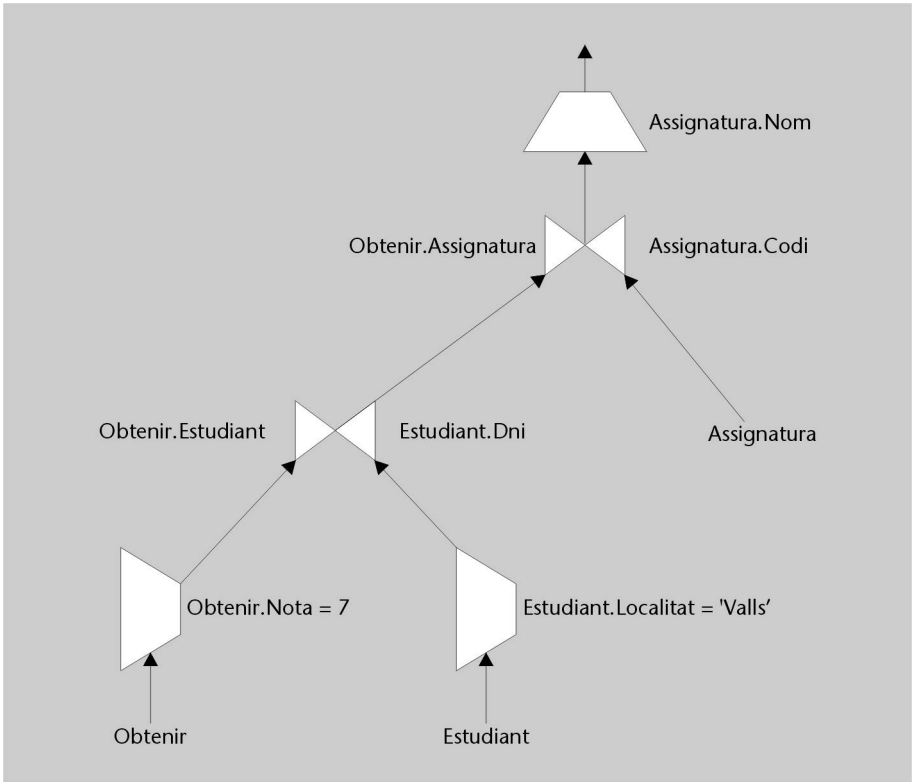
```
SELECT DISTINCT A.Nom
FROM Estudiant E, Obtenir O, Assignatura A
WHERE O.Estudiant = E.Dni
   AND O.Assignatura = A.Codi
   AND E.Localitat = 'Valls'
   AND O.Qualificació = 'Notable'
   AND O.Nota = 7.5
```

Fem l'optimització semàntica; totes les notes 7,5 són notables, per tant, es pot millorar la consulta de la manera següent:

```
SELECT DISTINCT A.Nom
FROM Estudiant E, Obtenir O, Assignatura A
WHERE O.Estudiant = E.Dni
   AND O.Assignatura = A.Codi
   AND E.Localitat = 'Valls'
   AND O.Nota = 7.5
```

Per a reduir la cardinalitat, caldrà fer primer les seleccions: suposem una distribució uniforme (suposició que segur que no és certa en el nostre cas, segur que hi ha més estudiants de Barcelona que de Valls, i més amb un 5,5 que amb un 7,5), haurem de reduir els estudiants a 75.000 / 2.000 ≈ 38 i les files d'Obtenir a 2.625.000 / 21 ≈ 125.000.

El pas següent serà combinar la taula resultant d'Obtenir amb les altres. Si primer es combina amb Assignatura, el resultat tindrà 125.000 files; en canvi, si es combina amb la selecció d'Estudiant, tindrà (suposem que cada estudiant té 2.625.000 / 75.000 = 35 notes) $35 \times 38 = 1.330$ files. Aquest resultat combinat amb les assignatures donarà 1.330 files (l'únic que fem és posar el nom a l'assignatura). Després caldrà fer la selecció i eliminar els repetits. Per tant, el pla (arbre) lògic d'execució serà:



b) Després de la selecció sobre la taula Obtenir tenim 125.000 files que ocuparan 125.000 / 70 = 1.786 pàgines i de la taula Estudiant en tindrem 38 files que ocuparan 38 / 35 = 2 pàgines. Com que es pot guardar tota la taula Estudiant en memòria, una bona opció és utilitzar un algoritme de combinació de cicle imbricat amb la taula Estudiant en el bucle intern; el cost serà de 1.786 + 2 = 1.788 pàgines.

4. A partir de la taula següent es pot comprovar com ha estat possible que la transacció 1 hagi llegit el valor de A, encara que la transacció 0 havia fet prèviament una reserva d'escriptura; després la 1 no ha pogut fer la reserva d'escriptura i ha restat a l'espera, fins que la transacció 0 ha demanat una reserva de certificació i en aquest moment s'ha produït una abraçada mortal.

T_0	T_1	A_0	B_0	A_1	B_1
Comença T_0		25.000	15.000	25.000	15.000
lock (A, Lect)		25.000	15.000	25.000	15.000
llegir (A)		25.000	15.000	25.000	15.000
A := A - 1000	Comença T_1	25.000	15.000	25.000	15.000
lock (A, Esc)	lock (A, Lect)	25.000	15.000	25.000	15.000
escriure (A,)	llegir (A)	24.000	15.000	25.000	15.000
lock (B, Lect)	temp := 0.1 * A	24.000	15.000	25.000	15.000
llegir (B,)	A := A - temp	24.000	15.000	25.000	15.000
B := B + 1000	ESPERA	24.000	15.000	25.000	15.000
lock (B, Esc)	ESPERA	24.000	15.000	25.000	15.000
escriure (B)	ESPERA	24.000	16.000	25.500	15.000
COMMIT	ESPERA	24.000	16.000	25.000	15.000
DEAD LOCK	DEAD LOCK				
COMMIT	ROLLBACK	24.000	16.000		
	Comença T_1			24.000	16.000
	lock (A, Lect)			24.000	16.000
	llegir (A)			24.000	16.000
	temp := 0.1 * A			24.000	16.000
	A := A - temp			24.000	16.000
	escriure (A)	24.000	16.000	22.600	16.000
	lock (B, Lect)	21.600	15.000	22.600	16.000
	llegir (B)	21.600	15.000	22.600	16.000
	B := B + temp	21.600	15.000	22.600	16.000
	lock (B, Esc)	21.600	16.000	22.600	16.000
	escriure (B)	21.600	16.000	22.600	18.400
	COMMIT	21.600	16.000	22.600	18.400

S'ha resolt l'abraçada mortal, desfent i tornant a iniciar la transacció 1, que en aquest moment ja ha treballat amb les dades que havia escrit la 0.

Glossari

auditoria
Registre i monitoratge d'algunes accions especificades d'usuaris específics sobre la base de dades.

autenticació
Mecanisme que assegura que un usuari és qui diu que és.

control d'accés
Mecanisme que té la funció d'assegurar que els accessos al sistema estan d'acord amb el models i regles fixades per la política de protecció.

disparador de substitució
Programa que s'executa "en consta de" realitzar-se alguna operació de modificació de la base de dades.

heurístic -a
Dit dels mètodes que utilitzen el raonament i les experiències passades per a trobar la millor solució a un problema.

identificació
Procés que permet a l'usuari de proporcionar-ne la identitat al sistema.

optimització
Procés pel qual es transforma una consulta en una altra d'equivalent, però més eficient. L'optimització es pot realitzar a nivell semàntic, sintàctic i físic.

pla d'execució
Seguit d'operacions (lògiques o físiques) que cal realitzar per a obtenir el resultat d'una consulta.

planificador
Part de l'SGBD, que assigna el temps de procés a les diferents transaccions, seguint alguna política que impedeixi els diferents problemes de la concurrència.

marca temporal
Valor numèric que es pot assignar a qualsevol transacció o grànul d'informació i pot ser utilitzat per a implementar polítiques de control de concurrència.

Bibliografia

Bibliografia bàsica

Connolly, T.; Begg, C.; Strachan, A. (1999). Database Systems. Harlow: Addison-Wesley.

Date, C.J. (2000). An Introduction to Database Systems. Reading: Addison-Wesley.

Ramakrishnan, R; Gehrke, J. (2000). Database Management Systems. Singapore: McGraw-Hill International Editions.

Bibliografia complementària

Castano, S.; Fugini, M.; Martella, G.; Samarati, P. (1994). Database Security. Workingham: Addison-Wesley.

Elmasri, R.; Navathe, S.B. (2001). Sistemas de bases de datos. Wilmigton: Addison-Wesley Iberoamericana.

Garcia-Molina, H.; Ullman, J.D.; Widom, J. (2000). Database Systems Implementation. Upper Saddle River: Prentice Hall.

Gulutzan, P.; Pelzer, T. (1999). SQL-99 Complete, Really. Lawrence: R&D Books.

Silberschatz, A.; Korth, H.F.; Sudarshan, S. (1998). Fundamentos de bases de datos. Madrid: McGraw-Hill.