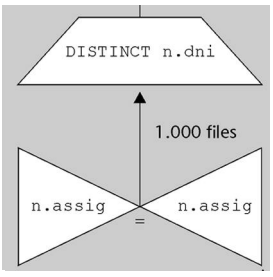
Disseny físic de bases de dades

Blai Cabré i Segarra

P01/11031/00007

Universitat Oberta de Catalunya



Índex

Introducció 5

Objectius 6

1. Adaptació a un SGBD concret 7
   1.1. Taula 7
   1.2. Espai per a taules 9
   1.3. Base de dades 12
   1.4. Índex 14
   1.5. Clau primària i clau alternativa 17

2. Ajust i millora 18
   2.1. Metodologia d'afinament d'un subsistema de bases de dades 18
      2.1.1. Aplicacions en línia 20
      2.1.2. Aplicacions per lots 21
      2.1.3. Consultes d'usuari 22
   2.2. Índexs 23
      2.2.1. Conveniència dels índexs 25
   2.3. Components físics 29
      2.3.1. Fitxers de dades 29
      2.3.2. Fitxers de control 34
      2.3.3. Fitxers d'auditoria 36
      2.3.4. Diaris 38
   2.4. Paràmetres del sistema 39

3. Gestió del rendiment 41
   3.1. Pla de les consultes 41
   3.2. Monitors 46
   3.3. Benchmarks 49
   3.4. Elecció d'un SGBD 50

4. Informe del dissenyador de la base de dades 54

Resum 56

Activitats 57

Exercicis d'autoavaluació 57

Solucionari 59

Glossari 59

Bibliografia 60

Introducció

Després del disseny lògic de la base de dades s'arriba al disseny físic tenint en compte les característiques de cada sistema gestor de base de dades (SGBD).

Els components físics de cada SGBD són específics, i el constructor els ha dissenyat pensant en el sistema en què ha de funcionar, per tal de treure'n el màxim profit. El disseny físic de la base de dades parteix del disseny lògic, i ha de tenir present les peculiaritats de cada gestor i adaptar-s'hi convenientment.

També formen part del disseny físic els aspectes relacionats amb el rendiment de les aplicacions, i la manera d'ajustar el disseny de la base de dades per a millorar-lo.

Objectius

En aquest mòdul revisarem el pas del disseny lògic de la base de dades al disseny físic, adaptat a les característiques de cada sistema de gestió de bases de dades i alhora ajustat per a tenir un bon rendiment. Els objectius principals que pretenem d'assolir són els següents:

1. Aprendre a fer el disseny físic de la base de dades a partir del disseny lògic, adaptat a les característiques d'un SGBD concret.

2. Definir els índexs necessaris i convenients en cada taula, per tal que les aplicacions tinguin un bon rendiment quan accedeixen a la base de dades.

3. Comprendre la influència dels paràmetres del sistema en el funcionament i rendiment del gestor.

4. Entendre la informació obtinguda del pla de les consultes i predir els accessos del gestor.

5. Saber interpretar la informació subministrada pels monitors de rendiment.

1. Adaptació a un SGBD concret

A partir del disseny lògic d'una base de dades hem de passar al disseny físic d'aquesta, passant pel nivell virtual. La forma d'implementar un disseny lògic en un SGBD concret depèn de les característiques pròpies de cada un d'aquests. I les característiques pròpies de cada SGBD són un reflex de l'entorn:

Vegeu la descripció del nivell virtual en l'apartat 4 del mòdul "Components d'emmagatzematge d'una base de dades" de l'assignatura Disseny de bases de dades II.

- Característiques del maquinari.
- Sistema operatiu i programari bàsic.
- Disseny del SGBD.

Cada SGBD ha desenvolupat un llenguatge propi, fet a mida pel mateix constructor, per a implementar el disseny físic de la base de dades, d'acord amb les característiques de l'entorn, i per a obtenir el màxim rendiment del maquinari, del sistema operatiu i del mateix gestor. De fet, es podria considerar com una ampliació del llenguatge SQL estàndard, amb aquelles clàusules pròpies, que cada gestor necessita per a definir els components del disseny físic. No obstant això, hi ha una gran similitud o equivalència entre gran part dels components, dels diferents gestors.

L'estàndard SQL incorpora la definició de tots els components del disseny lògic de la base de dades. En canvi, no incorpora cap element del disseny físic.

Per a adaptar el disseny de la base de dades a un SGBD concret, partim dels elements següents:

1) Les definicions de taules amb les corresponents claus primàries, claus foranes, claus alternatives, que hem obtingut en el disseny lògic de la base de dades.
2) A continuació relacionem cada taula amb un espai per a taules, i cada índex amb un espai per a índex. És el nivell virtual.
3) Per acabar, relacionem cada espai virtual amb un fitxer físic, i en definim les característiques. Això constitueix el disseny físic de la base de dades.

Vegem el pas al disseny físic amb alguns exemples d'SGBD concrets.

1.1. Taula

La taula és un component del disseny lògic de la base de dades. I com a tal està perfectament definit a l'estàndard SQL.

La sentència CREATE TABLE de tots els SGBD incorporen les clàusules de l'estàndard SQL i, a més a més, l'enllaç amb els elements físics propis de cada un d'aquests.

A continuació presentem alguns exemples de creació de taules amb tres SGBD diferents:

1) Amb SQL Server:

```
CREATE TABLE table_name
             ( column_definition )
             < unique_constraint >
             < referential_constraint >
             < check_constraint >

ON filegroup
```

2) Amb Oracle:

```
CREATE TABLE table_name
             ( column_definition )
             < unique_constraint >
             < referential_constraint >
             < check_constraint >

< extent_specs >
TABLESPACE table_space_name
```

3) Amb DB2:

```
CREATE TABLE table_name
             ( column_definition )
             < unique_constraint >
             < referential_constraint >
             < check_constraint >

IN table_space_name
```

Les clàusules següents de la sentència CREATE TABLE pertanyen a l'SQL estàndard: column_definition, unique_constraint, referential_constraint, check_constraint. Aquestes clàusules són definicions del disseny lògic. Tots els SGBD les incorporen amb una sintaxi quasi idèntica.

D'altra banda, les clàusules ON de l'SQL Server, < extent_specs > d'Oracle, TABLESPACE d'Oracle i IN de DB2, són específiques de cada gestor. Serveixen per a relacionar la definició de la taula (nivell lògic) amb l'espai per a taules (nivell virtual) que en cada gestor s'anomena Filegroup a l'SQL Server, Tablespace a Oracle i Tablespace a DB2. A més a més, la clàusula < extent_specs > d'Oracle incorpora paràmetres de percentatge d'ocupació de les pàgines de l'espai per a taules: PCTFREE, PCTUSED, INITRANS, MAXTRANS, etc.

1.2. Espai per a taules

L'espai per a taules és un component del nivell virtual. No pertany al disseny lògic de la base de dades i, per tant, no està inclòs a l'estàndard SQL.

La sentència de creació d'un espai per a taules és diferent a cada SGBD i incorpora les clàusules d'enllaç amb els elements físics propis de cada un d'aquests (els fitxers). Vegem-ne alguns exemples:

1) Amb SQL Server:

```
< filegroup > ::=

     FILEGROUP filegroup_name < filespec > [ ,...n ]

< filespec > ::=

   [ PRIMARY ]
   ( [ NAME = logical_file_name , ]
     FILENAME = 'os_file_name'
     [ , SIZE = size ]
     [ , MAXSIZE = { max_size | UNLIMITED } ]
     [ , FILEGROWTH = growth_increment ] ) [ ,...n ]
```

- Filegroup_name és el nom de l'espai per a taules que es relaciona a la clàusula ON del CREATE TABLE. De fet, l'espai per a taules es defineix dins la sentència CREATE DATABASE que veurem en el subapartat següent.

- Cada filegroup s'associa a un, o més d'un, fitxer físic (filespec).

- La definició del fitxer físic ve donat pel seu nom lògic logical_file_name.

- os_file_name és el nom extern del fitxer, que està format pel dispositiu, directori, tota l'estructura jeràrquica de subdirectoris, nom del fitxer i extensió. Exemple: 'c:\program files\sql server\saledat.mdf'.

- SIZE és la mida inicial assignada al fitxer.

- FILEGROWTH és el percentatge de creixement incremental del fitxer, a partir de la mida inicial. L'increment es pren cada cop que el fitxer està ple. En cap cas es pot sobrepassar el màxim permès.

- MAXSIZE és la mida màxima del fitxer a què es permet arribar després dels successius creixements.

2) Amb Oracle:

```
CREATE TABLESPACE table_space_name
       DATAFILE < filespec > [ ,...n ]
       DEFAULT STORAGE < storage_clause >


< filespec > ::=

     'file_name'
     [ SIZE nn ]


< storage_clause > ::=

     INITIAL   nn
     [ NEXT    nn ]
     [ MINEXTENTS   nn ]
     [ MAXEXTENTS   nn ]
     [ PCTINCREASE    nn ]
     [ OPTIMAL      nn ]
     [ ..... ]
```

- Table_space_name és el nom de l'espai per a taules, que es defineix amb aquesta sentència, i que està relacionada amb la clàusula TABLESPACE de la sentència CREATE TABLE.

- Cada espai per a taules s'associa a un, o més d'un, fitxer físic (filespec).

- La definició del fitxer físic ve donat pel seu nom extern file_name, la ubicació en disc, i la seva mida size.

- La clàusula storage_clause defineix les característiques de l'emmagatzematge, mida inicial, mida incremental, extensions mínimes i màximes, etc.

3) Amb DB2:

```
CREATE TABLESPACE table_space_name
       IN         database_name
       [ USING STOGROUP < storage_clause > ]
       [ NUMPARTS nn ]
       [ ..... ]


< storage_clause > ::=

        stogroup_name
        [ PRIQTY  nn ]
        [ SECQTY  nn ]
        [ FREEPAGE   nn ]
        [ PCTFREE    nn ]
```

- Table_space_name és el nom de l'espai per a taules. Es defineix en aquesta sentència. Aquest nom d'espai per a taules és assignat des de la clàusula IN de la sentència CREATE TABLE.

- Cada espai per a taules també es relaciona amb una base de dades de nom database_name, i que s'explica en el subapartat següent.

- Cada espai per a taules s'associa a un, o més d'un, fitxer físic en funció de la clàusula NUMPARTS.

- La definició del fitxer físic ve donat pel seu nom extern, que està format pel conjunt dels noms de la database_name, table_space_name i stogroup_name.

- La clàusula storage_clause defineix les característiques de l'emmagatzematge, com ara:

– L'espai per a taules s'associa a un stogroup_name, el qual agrupa unes determinades unitats de disc, on s'ubicarà el fitxer físic.

– PRIQTY és la mida inicial del fitxer.

– SECQTY és la mida incremental, per quan s'esgota l'extensió inicial del fitxer.

– FREEPAGE indica la reserva de pàgines buides, entremig de pàgines ocupades.

– PCTFREE és el percentatge de reserva d'espai buit dins cada pàgina.

Com podem comprovar, la definició de l'espai virtual és diferent a cada SGBD, per bé que hi ha molts components equivalents. En aquests exemples hem vist les clàusules principals de la definició de l'espai per a taules i dels fitxers físics. Per a més detall, cal consultar la documentació pròpia de cada gestor.

1.3. Base de dades

No pertany al disseny lògic de la base de dades i, per tant, no està inclòs a l'estàndard SQL.

La sentència CREATE DATABASE és diferent a cada SGBD i incorpora clàusules de definició d'elements físics propis de cada un d'aquests (diari, catàleg, fitxers, etc.). Vegem-ne alguns exemples:

1) Amb SQL Server:

```
CREATE DATABASE database_name
[ ON
   [ < filespec > [ ,...n ] ]
   [ , < filegroup > [ ,...n ] ]
]
[ LOG ON { < filespec > [ ,...n ] } ]
[ COLLATE collation_name ]
[ FOR LOAD | FOR ATTACH ]
```

- database_name és el nom de la base de dades que s'associa a un conjunt de fitxers que s'especifiquen a la clàusula ON filespec, i que contenen els espais per a taules que hem vist anteriorment.

- La clàusula LOG ON especifica el nom dels diaris que es defineixen perquè el gestor registri totes les actualitzacions de les taules d'aquesta base de dades i així possibilitar de recuperar-les en cas necessari.

- El detall de la definició dels fitxers físics s'ha explicat en el subapartat dels espais per a taules.

2) Amb Oracle:

```
CREATE DATABASE database_name
     [ CONTROLFILE REUSE ]
     [ LOGFILE < filespec > [ ,...n ] ]
     [ MAXLOGFILES nn ]
     [ MAXLOGMEMBERS nn ]
     [ MAXLOGHISTORY nn ]
     [ DATAFILE < filespec > [ ,...n ] ]
     [ MAXDATAFILES nn ]
     [ MAXINSTANCES nn ]
     [ CHARACTER SET charset ]
     [ ..... ]
```

- database_name és el nom de la base de dades que s'associa a un conjunt de fitxers físics que contenen els espais per a taules que hem explicat en el subapartat anterior.

- Els fitxers físics que contenen els espais per a taules es relacionen a la clàusula DATAFILE < filespec >.

- La clàusula LOGFILE < filespec > especifica el nom dels diaris que es defineixen perquè el gestor registri totes les actualitzacions de les taules d'aquesta base de dades i així possibilitar de recuperar-les en cas necessari.

- El detall de la definició dels fitxers físics < filespec > s'ha explicat en el subapartat dels espais per a taules.

- Altres paràmetres limiten el nombre màxim de fitxers de cada tipus, MAXLOGFILES, MAXLOGMEMBERS, MAXDATAFILES, etc.

3) Amb DB2:

```
CREATE DATABASE database_name
     [ STOGROUP stogroup_name ]
     [ ..... ]
```

- La bases de dades database_name s'associa amb un nom de grup d'emmagatzematge stogroup_name que s'utilitza per omissió quan en la definició dels espais per a taules no se n'especifica cap.

- En aquest cas, l'associació entre base de dades i espais per a taules es fa des de la definició d'aquest últim (CREATE TABLESPACE).

- No hi ha associació amb cap diari. El diari és únic per a totes les bases de dades del gestor.

1.4. Índex

Els índexs són uns elements del disseny físic de la base de dades que tenen com a finalitat millorar el rendiment de les aplicacions quan accedeixen a les taules. Els índexs no formen part del disseny lògic de la base de dades i, per tant, no estan inclosos a l'estàndard SQL.

La definició de la clau primària i de les claus foranes formen part del disseny lògic de la base de dades, i així consten a l'estàndard SQL. En canvi, els índexs que es defineixen per a millorar l'accés a les taules, responen a requeriments de rendiment de les aplicacions. Per tant, correspon al disseny físic de la base de dades.

La sentència CREATE INDEX és present a tots els SGBD amb opcions molt similars. A més a més, incorpora clàusules per a definir l'espai per a índex i clàusules d'enllaç amb els elements físics propis (els fitxers d'índexs). Vegem-ne alguns exemples:

1) Amb SQL Server:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]
   INDEX index_name
ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
   [ WITH < index_option > [ ,...n] ]
   [ ON filegroup ]
```

- Index_name és el nom lògic de l'índex, definit sobre la taula especificada a la clàusula ON.

- UNIQUE i CLUSTERED són característiques de l'índex.

- Filegroup és el nom de l'espai per a índex, i que es defineix dins la sentència CREATE DATABASE que hem vist anteriorment.

- Com ja sabem, cada filegroup s'associa a un, o més d'un, fitxer físic amb característiques pròpies de nom, grandària, ubicació física en disc, etc., tal com hem vist en la definició dels espais per a taules.

2) Amb Oracle:

```
CREATE INDEX index_name
       ON table_name ( column [ ASC | DESC ] [ ,..n ])
       [ CLUSTER cluster_name ]
       [ < extent_specs > ]
       [ TABLESPACE table_space_name ]
       [ ..... ]


< extent_specs > ::=

       [ PCTFREE nn ]
       [ PCTUSED nn ]
       [ INITRANSnn ]
       [ MAXTRANSnn ]
       [ STORAGE < storage_clause > ]


< storage_clause > ::=

       INITIAL   nn
       [ NEXT    nn ]
       [ MINEXTENTS   nn ]
       [ MAXEXTENTS   nn ]
       [ PCTINCREASE  nn ]
       [ OPTIMAL      nn ]
       [ ..... ]
```

- Index_name és el nom lògic de l'índex, definit sobre la taula especificada a la clàusula ON.
- UNIQUE i CLUSTER són característiques de l'índex.
- Table_space_name és el nom de l'espai per a índex, i que es defineix amb la sentència CREATE TABLESPACE que hem vist anteriorment.
- Quan es crea l'espai per a índex (CREATE TABLESPACE) s'associa a un, fitxer físic amb característiques pròpies de nom, grandària, ubicació física en disc, etc.
- La clàusula < extent_specs > especifica condicions de percentatge d'ocupació de les pàgines de l'espai per a índex: PCTFREE, PCTUSED.
- La clàusula storage_clause defineix característiques d'emmagatzematge, mida inicial, mida incremental, extensions mínimes i màximes, etc.

3) Amb DB2:

```
CREATE [ UNIQUE ] INDEX index_name
       ON table_name ( column [ ASC | DESC ] [ ,..n ])
       [ CLUSTER ]
       [ USING STOGROUP < storage_clause > ]
       [ ..... ]


< storage_clause > ::=

       stogroup_name
       [ PRIQTY    nn ]
       [ SECQTY    nn ]
       [ FREEPAGE  nn ]
       [ PCTFREE   nn ]
```

- Index_name és el nom lògic de l'índex, definit sobre la taula especificada a la clàusula ON.

- UNIQUE i CLUSTER són característiques de l'índex.

- Aquesta sentència CREATE defineix l'espai per a índex i l'associa a un fitxer físic amb característiques pròpies, com són el nom del fitxer, la grandària, la ubicació física en disc, etc.

- La definició del fitxer físic ve donat pel seu nom extern, que està format pel conjunt dels noms de la database_name, index_name i stogroup_name.

- La clàusula storage_clause defineix característiques d'emmagatzematge, igual que s'ha explicat amb l'espai per a taules, i amb idèntic significat. Recordem-ho:

– L'espai per a índex s'associa a un stogroup_name, el qual agrupa unes determinades unitats de disc, on s'ubicarà el fitxer físic de l'índex.
– PRIQTY és la mida inicial del fitxer.
– SECQTY és la mida incremental per quan s'esgota l'extensió inicial del fitxer.
– FREEPAGE indica la reserva de pàgines buides, entremig de pàgines ocupades.
– PCTFREE és el percentatge de reserva d'espai buit dins de cada pàgina.

La forma de definir l'espai per a índex és diferent a cada SGBD, per bé que hi ha molts components equivalents. En aquests exemples hem vist les clàusules principals de la definició de l'espai per a índex i dels fitxers físics que els contenen. Per a més detall, cal consultar la documentació pròpia de cada gestor.

1.5. Clau primària i clau alternativa

Les definicions de claus primàries, foranes i alternatives, actualment formen part de l'estàndard SQL. Però no sempre ha estat així. L'any 1986 encara no s'havien definit, i l'estàndard SQL86 es va publicar sense aquestes. A l'estàndard SQL89 s'esmenten les claus primàries, per primera vegada. I no va ser fins al 1992 que es defineixen i s'incorporen de fet a l'estàndard SQL92.

Com ja sabem, la clau primària ha de ser, obligatòriament, una clau única i que no admet valors nuls. La teoria del model de bases de dades relacionals suggereix que cada taula hauria de tenir una clau primària que identifiqui de forma unívoca l'entitat que descriu. Malgrat això, l'estàndard SQL no obliga l'existència d'una clau primària a cada taula, sinó que és opcional. Lògicament, però, cada taula pot tenir com a màxim una clau primària.

La resta de claus úniques i que no admeten valors nuls són claus alternatives a la clau primària. Tampoc això no està legislat a l'estàndard SQL.

2. Ajust i millora

El disseny físic de la base de dades inclou també l'ajust i millora d'una sèrie de components del sistema de gestió de base de dades, sense els quals no obtindríem un rendiment global satisfactori. Aquests components són els següents:

a) Els índexs que es defineixen per la seva gran influència en el rendiment.
b) Els components físics com ara els fitxers que contenen la base de dades, i també altres fitxers de control del mateix gestor.
c) Paràmetres del sistema que tenen repercussió en el rendiment.

Comencem, però, veient una metodologia que convé seguir per a afinar el sistema de bases de dades des d'un punt de vista global.

2.1. Metodologia d'afinament d'un subsistema de bases de dades

Un sistema de bases de dades que s'ha dissenyat per a explotar-lo i treure'n un bon rendiment necessita que tots els seus components principals tinguin un bon disseny, una bona integració i un bon afinament. Els components més importants són:

– el maquinari,
– el sistema operatiu,
– l'SGBD (sistema gestor de base de dades),
– les aplicacions i
– la base de dades.

El mal funcionament o el pobre rendiment d'algun d'aquests components dóna com a resultat un funcionament defectuós del sistema global de bases de dades. Per aquest motiu, és important la bona "posada al punt" de tots els components que el configuren.

El conjunt de màquines (unitats centrals, perifèrics d'emmagatzematge –discos–, perifèrics de comunicacions, etc.) que intervenen en el funcionament d'un sistema de bases de dades requereixen un afinament propi que desenvolupa habitualment l'especialista de maquinari. Les tasques d'afinament depenen de la configuració pròpia de màquines de les quals es disposa, i de les característiques de cada una d'aquestes. És el mateix constructor qui estableix les tasques que cal fer per a afinar el conjunt de màquines.

L'afinament del sistema operatiu també té unes característiques pròpies i requereix la intervenció de l'especialista corresponent. L'afinament acostuma a

ser bastant independent dels altres components del sistema, encara que hi ha alguns elements que estan relacionats amb l'SGBD, que comentarem més endavant.

En aquest mòdul, tractarem alguns aspectes de rendiment de l'SGBD i, amb més detall, aspectes de disseny d'aplicacions i de disseny de bases de dades, que són els que tenen més importància, per dues raons fonamentals:

1) En primer lloc, per la gran quantitat de recursos emprats, habitualment, en el desenvolupament d'aplicacions i en el disseny de bases de dades.

2) En segon lloc, per l'elevat cost (temps, personal, recursos dedicats) i la gran dificultat que representa fer canvis en el funcionament de les aplicacions i en el disseny de les bases de dades, a posteriori, quan el sistema ja està funcionant en producció real. Canvis en les aplicacions i en les bases de dades que són deguts a la necessitat de millorar aspectes de rendiment.

El disseny i l'afinament dels programes d'aplicació, i el disseny i l'afinament de les bases de dades poden millorar de manera extraordinària el rendiment del sistema. Els aspectes més importants que cal considerar en la fase de disseny en relació amb el rendiment són els següents:

a) Consum de recursos: cicles de CPU, memòria central, memòria externa, entrada/sortida, etc.

b) Concurrència de diferents processos sobre les mateixes dades. Integritat de la informació.

c) Contenció entre processos d'aplicació que competeixen pels mateixos recursos i les mateixes dades. Poden arribar fins a situacions d'interbloqueig (abraçades mortals*) o de temporització (cancel·lacions per temps**).

* En anglès, deadlocks.
** En anglès, timeouts.

d) Temps de resposta i durada de cada procés.

Una de les condicions importants en el disseny físic de bases de dades és aconseguir un disseny que permeti que les aplicacions tinguin un bon rendiment quan accedeixen a les dades.

Ni el disseny físic de la base de dades es pot fer sense tenir en compte el tipus d'aplicacions que hi accedirà, ni el disseny de les aplicacions es pot fer a esquena del disseny de les bases de dades que faran servir.

Malgrat que aquesta assignatura és de disseny de bases de dades, hem de conèixer alguns aspectes del disseny d'aplicacions que ens permetin de fer un disseny físic de base de dades adequat i obtinguem un rendiment acceptable.

Ja sabeu com podem classificar les diferents formes d'utilitzar el llenguatge SQL. Recordem-ho:

Vegeu amb més detall les diferents formes d'utilitzar el llenguatge SQL en el mòdul "Programació amb SQL" de l'assignatura Disseny de bases de dades II.

a) SQL interactiu o SQL directe: s'utilitzen les sentències en llenguatge SQL que són interpretades directament. Es tracta d'SQL dinàmic.

b) SQL programat: s'incorporen les sentències d'accés a la base de dades en un programa d'aplicació. Hi ha les dues tècniques següents:

- SQL hostatjat: les sentències SQL es troben incorporades directament dins l'aplicació, barrejades amb el llenguatge amfitrió. Les sentències SQL es compilen i obtenim SQL estàtic.

- SQL/CLI: l'aplicació es comunica amb l'SGBD mitjançant unes interfícies que permeten de fer crides a subrutines que es troben disponibles en llibreries. El llenguatge SQL és interpretat i, per tant, estem parlant d'SQL dinàmic.

D'altra banda, podríem classificar la majoria dels processos aplicatius vers bases de dades en els tres grans grups següents:

– processos en línia*,
– processos per lots** i
– consultes d'usuari***.

* En anglès, online.
** En anglès, batch.
*** En anglès, queries.

A més a més, podem esmentar les utilitats que tenen una altra consideració, ja que no es tracta de processos aplicatius.

Utilitats

Recordeu que les utilitats són programes propis de l'SGBD, i que fan funcions auxiliars com ara: càrregues, còpies, reorganitzacions, restauracions, recuperacions, etc.

Vegem les característiques de cada una d'aquestes.

2.1.1. Aplicacions en línia

Un entorn transaccional té uns requeriments de disponibilitat de dades, temps de resposta i rendiment, que condicionen el disseny de les aplicacions.

Les característiques comunes dels programes transaccionals amb accés a bases de dades relacionals són les següents:

a) Són processos d'alta prioritat. Quan s'inicia una transacció és perquè es necessita fer el procés en aquest moment. No es pot planificar més tard.

b) Hi ha una gran concurrència d'accés a les dades. Moltes transaccions competeixen entre si per accedir a les mateixes dades.

c) Els temps de resposta han de ser relativament curts. Darrere d'una transacció hi ha un usuari que espera la resposta.

d) El temps d'entrada/sortida determina un gran percentatge del temps de resposta.

e) S'utilitzen, majoritàriament, instruccions d'SQL hostatjat (SQL estàtic). En menys ocasions es fa servir SQL/CLI (SQL dinàmic).

f) La complexitat de les sentències SQL utilitzades no acostuma a ser molt gran.

g) Generalment, l'accés a dades es fa per mitjà d'índexs.

h) Cada transacció accedeix a un nombre reduït de dades (poques files).

Els aspectes més importants que cal tenir en compte en el disseny transaccional són la concurrència d'accessos i l'ús eficient dels recursos:

1) Un disseny orientat a la concurrència permet que diversos processos s'executin en paral·lel sobre les mateixes dades, sense que les interferències entre si comprometin seriosament el rendiment total. Si el disseny és correcte, el rendiment total es podria millorar simplement afegint més recursos de màquina al sistema.

2) Un disseny orientat a l'ús eficient dels recursos possibilita reduir els requeriments de recursos mantenint la mateixa funcionalitat de les aplicacions.

2.1.2. Aplicacions per lots

Les característiques comunes dels programes d'aplicació que s'executen en un entorn per lots*, amb accés a bases de dades relacionals, són les següents:

* En anglès, batch.

a) Són processos de baixa prioritat i que s'acostumen a planificar en hores de poca activitat. Per exemple, durant la nit, que hi ha poca activitat en línia. Molts processos per lots es planifiquen a determinades hores de la tarda o de la nit, sense la presència de l'usuari i només amb la supervisió de l'operador de la màquina.

b) Majoritàriament no hi ha concurrència d'accés a les dades entre processos per lots. Però sí que n'hi ha entre processos per lots i processos en línia. Pensem que, cada cop més, les aplicacions han d'estar en marxa les vint-i-quatre hores del dia. En conseqüència, hi ha concurrència entre processos per lots i processos en línia.

c) Sovint es tracta de processos de llarga durada. Processos i càlculs d'una certa complexitat i que afecten un gran volum de dades.

d) És molt important encavalcar el temps d'entrada/sortida amb el temps de CPU, per a reduir la durada del procés.

e) Es fan servir, preferentment, instruccions d'SQL hostatjat (SQL estàtic).

f) Els principals accessos a dades són seqüencials a la base de dades. Els altres són mitjançant índexs.

g) Cada programa accedeix a un nombre elevat de dades (moltes files).

Aspectes més rellevants amb vista al rendiment del sistema:

1) En sistemes grans, els SGBD acostumen a ser molt eficients en processos seqüencials d'accés a la base de dades. A més a més, les operacions d'entrada/sortida es poden fer de forma asíncrona i encavalcada amb el procés de la CPU.

2) S'han de dissenyar els processos per lots pensant en la concurrència de dades, conjuntament amb els processos en línia. Cal limitar la durada del bloqueig de dades perquè no interfereixi amb altres processos.

2.1.3. Consultes d'usuari

Les consultes* directes són un conjunt d'instruccions SQL que s'executen per separat o de forma encadenada, i que l'usuari demana en un moment determinat. Les coneixem amb el nom d'SQL directe o interactiu.

* En anglès, queries.

Les característiques principals de les consultes directes són les següents:

a) Són processos en línia. Majoritàriament de lectura (actualitzacions poc freqüents).

b) L'usuari utilitza consultes planificades prèviament, és a dir, consultes que s'han escrit, i provat amb anterioritat. També l'usuari crea consultes ad hoc o imprevistes, que són consultes no planificades i que escriu en el mateix moment en què sol·licita la seva execució. Per tant, es tracta de consultes que no s'han provat i de les quals es desconeix el resultat.

c) Els temps de resposta és molt variable. Hi ha consultes curtes, llargues, i de durada desconeguda quan es tracta d'una nova consulta i que resulta difícil d'estimar.

d) Són instruccions d'SQL interactiu o directe (SQL dinàmic).

e) La complexitat de les sentències SQL utilitzades acostuma a ser alta.

f) Hi ha tot tipus d'accés: seqüencial de dades i per mitjà d'índexs.

g) Cada transacció accedeix a un nombre relativament gran de dades (moltes files).

h) El temps de resposta ve determinat per la complexitat de les sentències SQL utilitzades.

Aquests són els aspectes més rellevants:

1) Les consultes són processos de cost elevat i no planificats. Se'ls assigna baixa prioritat perquè no interfereixin en els processos en línia que tenen alta prioritat.

2) Per la mateixa raó, el seu disseny ha de permetre la concurrència amb altres processos quan es comparteixen les mateixes dades.

2.2. Índexs

La definició d'índexs és una part important del disseny físic de bases de dades i té una influència cabdal en el rendiment dels processos aplicatius que accedeixen a les bases de dades.

Com hem vist en els apartats anteriors la majoria dels processos aplicatius en línia d'alta prioritat accedeixen a les dades per mitjà dels índexs.

En cada un dels accessos a taules que té programat una aplicació, l'optimitzador de l'SGBD analitza quin és el millor camí d'accés a la base de dades. L'anàlisi es fa durant el procés de construcció del pla d'accés*. L'optimitzador fa l'anàlisi tenint en compte els elements següents:

* En anglès, BIND plan.

- Els índexs que s'han definit a cada taula.
- Les dades estadístiques sobre el nombre de files, i el nombre de claus de cada índex. Aquestes dades estadístiques, cal generar-les periòdicament i emmagatzemar-les al catàleg de l'SGBD.
- La cardinalitat de les claus de cada índex.
- El nombre de files que l'SGBD preveu que haurà d'explorar per satisfer la instrucció SQL.
- El nombre resultant de files afectades per l'accés SQL.
- Etc.

Cardinalitat

Nombre d'ocurrències diferents per a una clau determinada.

L'optimitzador de l'SGBD tria quin creu que és el camí òptim d'accés a la taula. L'optimitzador entén com a òptim aquell accés que preveu que tindrà un cost inferior, quant a consum de recursos (temps de CPU) i un millor temps de resposta. En molts casos, aquesta tria comporta usar algun dels índexs definits per a la taula.

A la pràctica, la diferència entre un accés a base de dades utilitzant o no un índex pot ser enorme.

Exemple d'accés a base de dades amb índex o sense

Tenim una taula de clients d'una empresa. Els atributs de cada client són: el nom i cognoms, el DNI o NIF, l'adreça, el telèfon, etc.

```
CREATE TABLE   clients
               (dni              ......
               nom               ......
               cognom            ......
               adreca            ......
               poblacio          ......
               telefon           ......
               sexe              ......
               data_naixement    ......
               saldo             ......
               ............      ......
```

Tenim un programa de consulta de la taula de clients. La consulta es fa a partir del DNI o NIF del client.

```
SELECT      .........
FROM        clients
WHERE       dni = :hv1
```

En cas de disposar d'un índex definit sobre la columna DNI, l'optimitzador de l'SGBD molt probablement l'haurà triat com a camí d'accés òptim. En aquest cas, l'SGBD farà uns quants accessos al fitxer d'índex per a seleccionar la clau i la localització de les dades. A continuació n'hi ha prou amb un accés directe a les dades per a obtenir tota la informació del client sol·licitada pel programa de consulta.

Ara pensem què hauria passat si no disposéssim de l'índex esmentat. L'SGBD hauria hagut de llegir seqüencialment tota la taula fins a localitzar el client amb el DNI sol·licitat.

Òbviament, el fet d'accedir a la base de dades, amb índex o sense, no té importància quan es tracta de taules amb molt poques files; en canvi, el tema s'agreuja a mesura que la taula creix (pensem que hi ha taules que contenen milions de files).

Per tant, una de les raons importants per a crear un índex a la taula és la de reduir el nombre d'accessos que l'SGBD haurà de fer per a localitzar les dades.

Quan hem de definir un índex? Quan i com l'hem de definir des del punt de vista de rendiment? Quines característiques ha de tenir?

Ja heu après a definir índexs i en quins casos és convenient de fer-ho. En aquest subapartat, plantejarem una sèrie de consideracions que cal tenir en compte des del punt de vista del rendiment.

2.2.1. Conveniència dels índexs

Vegeu la manera de definir els índexs en el mòdul "Introducció al disseny de bases de dades" de l'assignatura Disseny de bases de dades I.

És convenient definir un índex en els casos següents:

1) Volem garantir la unicitat d'alguns atributs. Definint un índex únic, l'SGBD garanteix en tot moment que no hi ha duplicitat de valors en l'atribut.

La definició dels índexs

En la majoria dels SGBD, la definició d'una columna de la taula amb l'atribut únic porta implícita la definició de l'índex. En altres SGBD, cal definir l'índex de manera explícita.

2) Volem garantir l'ordenació de les files dins la taula. Quan es defineix un índex agrupat*, l'SGBD organitza la taula seguint l'ordre de l'índex esmentat. Com hem vist en els apartats anteriors, sovint els processos aplicatius per lots, accedeixen a les taules grans de forma seqüencial. Per tant, l'elecció de l'índex agrupat té una importància cabdal. Cal elegir com a índex agrupat aquell que afavoreix els processos per lots crítics de més durada. En els sistemes grans, l'SGBD és molt eficient en processos seqüencials.

* En anglès, cluster index.

3) Tenim relacions d'integritat referencial entre dues o més taules. La clau primària constitueix, obligatòriament, un índex de la taula pare. Les claus foranes formen, en la majoria dels casos, un altre índex. Pensem que entre dues taules relacionades amb integritat referencial, quan s'insereix o s'esborra una fila, en molts casos, l'SGBD ha d'anar a comprovar l'existència de la clau relacionada a l'altra taula. En aquests casos, és molt recomanable de disposar d'índex sobre les claus foranes per tal d'optimitzar els accessos de comprovació del mateix SGBD.

La definició de les claus primàries

En la majoria dels SGBD, la definició d'una clau primària porta implícita la definició d'un índex únic i no nul (NOT NULL). En altres SGBD, cal definir l'índex de manera explícita.

4) Cal analitzar la clàusula WHERE de les sentències SQL d'accés a base de dades dels programes més importants. Si no es disposa del codi, n'hi ha prou d'analitzar el pseudocodi del programa. S'ha d'estudiar la possibilitat de millorar cada accés a la taula si es defineix un índex amb les columnes de la clàusula WHERE. D'aquest estudi surten els índexs candidats a ser definits per a millorar els accessos. Cal seleccionar els índexs candidats que proporcionen més avantatges i crear-los.

5) Hi ha processos aplicatius que utilitzen les clàusules ORDER BY o GROUP BY. Definint un índex amb les columnes que formen part de l'ORDER BY o GROUP BY, eliminem la necessitat que l'SGBD classifiqui les files, ja que l'índex proporciona l'ordre que es vol.

6) Combinació* entre dues o més taules. En un procés de combinació entre dues taules, el punt més crític és la forma en què s'estableix l'enllaç entre les

* En anglès, join.

dues taules, a través dels predicats de la clàusula WHERE. Si definim un índex a cada taula amb les columnes que formen els predicats d'enllaç, tant de la primera taula com de la segona, aconseguirem, en la majoria dels casos, que l'optimitzador triï els índexs per a fer l'enllaç i, per tant, es millora el temps d'execució.

7) Més difícil resulta preveure quins són els índexs més útils per a les consultes* no planificades. No obstant això, d'acord amb els requeriments dels usuaris, es poden preveure, inicialment, els possibles accessos i criteris de classificació més habituals. D'aquests requeriments es dedueix la necessitat d'utilitzar índexs, que cal definir si encara no existeixen. Més endavant, amb el pas del temps, aquests requeriments solen canviar i n'apareixen de nous, per la qual cosa convé replantejar la necessitat dels índexs i fer els canvis oportuns.

* En anglès, queries.

No és convenient definir índex en els casos següents:

1) Si la taula té poques files. O millor dit, si les files de la taula ocupen poques pàgines. L'estructura del fitxer d'índex podria arribar a ser tant o més gran que el fitxer de dades. En aquest cas, resulta més eficient llegir unes quantes pàgines de dades en lloc de llegir l'arbre de l'índex i a continuació la pàgina de dades corresponent. Per tant, és millor, en aquest cas, que el dissenyador de la base de dades no defineixi aquest índex, perquè l'optimitzador de l'SGBD, quan fa l'anàlisi del camí òptim d'accés a la taula petita, no s'enganyi i no triï l'accés per mitjà de l'índex pensant que pot ser més eficient. Per tant, moltes vegades no cal definir índexs en taules petites. Exemples: taules de codis (taula de codis de província, taula de codis postals, etc.).

2) Si el nombre de valors diferents que pren és molt petit. És a dir, si es tracta d'un índex que discrimina molt poc. Per exemple, un índex sobre la columna sexe de la taula de clients. Es tracta d'un índex que discrimina molt poc perquè té molts duplicats. Per tant, és millor no definir aquest índex per no induir l'optimitzador a error. En algunes versions d'SGBD més avançades, és el mateix optimitzador que s'adona que l'índex discrimina molt poc, i decideix no utilitzar-lo.

3) Si els programes d'aplicació pretenen d'utilitzar l'índex amb predicats no indexables, és a dir, que l'SGBD és incapaç de suportar.

Predicats indexables

Recordem que un predicat NOT EQUAL no és indexable, mentre que el predicat EQUAL, sí que ho és.

Exemple d'índex amb predicat no indexable

Volem seleccionar tots els clients de fora de la ciutat de Barcelona.

```
SELECT   ......
FROM     clients
WHERE    poblacio    NOT EQUAL      'BARCELONA'
```

Suposem que hem definit un índex sobre la columna població. En cas d'aquesta SELECT, l'SGBD no pot fer servir l'índex perquè el predicat NOT EQUAL no és indexable. Per tant, en aquest cas l'índex no té utilitat.

4) Si les aplicacions fan tractaments massius de la taula.

Sequential scan

Un sequential scan és un procés de lectura seqüencial i massiu de la taula que el mateix SGDB desencadena automàticament.

Exemple de tractament massiu de la taula

Volem tractar tots els clients que tenen un saldo superior a un determinat valor:

```
SELECT   ......
FROM      clients
WHERE     saldo   >  :hv9
```

La taula de clients pot arribar a ser molt gran. El nombre de clients amb un saldo superior al valor especificat també pot arribar a ser molt elevat. Sabem que els grans SGBD són molt eficients en tractaments seqüencials de la base de dades. Per tant, en aquest cas, és més ràpid llegir tota la taula de clients seqüencialment que no llegir l'arbre d'un hipotètic índex sobre la columna saldo, per acabar llegint un nombre elevat de files de la taula de clients.

En casos com aquest, el dissenyador de la base de dades no hauria de definir cap índex sobre la columna saldo. D'aquesta forma, l'optimitzador de l'SGBD triarà la lectura seqüencial (sequential scan) de la taula.

Altres consideracions per a definir índexs eficients. El nombre d'índexs definits en cada taula té un cert impacte en el rendiment. Quan s'insereix una nova fila a la taula, l'SGBD actualitza automàticament els fitxers d'índexs amb la nova entrada. El mateix passa en cas d'actualització d'un camp que forma part d'algun índex. El cost de mantenir els fitxers d'índexs és directament proporcional al seu nombre. Per tant, convé eliminar índexs superflus i que no són estrictament necessaris. Vegem-ne alguns casos amb detall:

1) Índex sobre columnes molt actualitzades. Cal evitar que els índexs tinguin columnes que s'actualitzen molt freqüentment. La raó no és altra que la de reduir el treball que fa l'SGBD quan es modifica una columna de la taula que forma part d'un índex. L'SGBD actualitza automàticament el contingut de la clau al fitxer d'índex, amb les corresponents actualitzacions de l'arbre d'índex, si s'escau.

Exemple d'índex sobre columnes molt actualitzades

Definir un índex sobre la columna saldo de la taula de comptes corrents d'una entitat bancària. Moltes de les operacions bancàries actualitzen la columna saldo. Es tracta d'operacions en línia, molt utilitzades, i que requereixen un bon temps de resposta. Per tant, cal analitzar cada un dels accessos a la base de dades i reduir-los al mínim imprescindible. Disposar d'un índex sobre la columna saldo implica que totes aquestes transaccions que actualitzen la columna saldo farien una doble actualització, a la taula i al fitxer d'índex. En molts casos resulta prohibitiu, per l'elevat cost, mantenir l'índex esmentat sobre la columna saldo.

2) Índexs ascendents o descendents. La majoria dels índexs es defineixen sobre claus en ordre ascendent (opció per omissió). Determinats processos poden necessitar seqüències de claus descendents.

Exemples d'índexs ascendents i descendents

Tenim la taula de clients amb un índex definit per a fer recerques segons l'edat:

```
CREATE  INDEX  ind3  ON    clients
               (data_naixement   ASC)
               .......
```

Suposem que hi ha programes d'aplicació amb els accessos següents:

```
SELECT   .......
FROM     clients
WHERE    data_naixement = :hv8
```

I d'altres programes amb accessos del tipus:

```
SELECT   .......
FROM     clients
WHERE    ..................
ORDER BY data_naixement DESC
```

El primer programa farà servir l'índex ind3 perquè qualifica la columna data naixement amb un predicat indexable. El segon programa no pot fer servir l'índex ind3 perquè demana una ordenació decreixent per data naixement i l'índex ind3 és creixent.

Fixem-nos que en aquest cas n'hi ha prou que modifiqui la definició de l'índex ind3 i ferlo decreixent. Serviria per als dos processos anteriors. En el primer procés li resulta indiferent el fet que l'índex sigui creixent o decreixent, mentre que en el segon procés li és indispensable que l'índex sigui decreixent.

En aquest exemple, si no hi ha cap altra raó que justifiqui que ind3 sigui creixent, el podem transformar en decreixent per a solucionar la problemàtica anterior.

3) Índexs que no es fan servir per a cap programa d'aplicació. Si es detecta que un determinat índex no és utilitzat per cap programa d'aplicació, es pot esborrar? Anteriorment hem vist que el manteniment de qualsevol índex representa un cert cost per l'SGBD, per tant, cal reduir el nombre d'índexs als estrictament necessaris. No obstant això, hem de pensar que hi ha determinats índexs que són necessaris, malgrat que no s'utilitzin directament per a les aplicacions. Vegem-ne alguns casos:

a) Un índex únic garanteix la unicitat de les claus. Per exemple, en un índex únic definit sobre la columna DNI de la taula de clients, l'SGBD ens garanteix que a la base de dades no hi ha cap client repetit. Suposem en aquest cas que no hi ha cap programa d'aplicació que accedeixi a la taula de clients a partir del DNI. Resulta temptador pensar a eliminar aquest índex que ningú no fa servir. A més a més, es podria pensar que el programa d'aplicació que dóna d'alta els clients a la taula, abans d'inserir un nou client, fes la comprovació per veure si el client ja estava donat d'alta a la taula, i actuar en conseqüència. Malgrat tot aquest raonament, no hem d'oblidar mai que l'SGBD fa el control d'unicitat de claus amb una fiabilitat total, mentre que, si el control es fa des dels programes d'aplicació, resulta més costós, perquè cal desenvolupar un codi nou per a cada cas que es vol controlar, i hi ha la possibilitat de cometre errors.

b) Índexs d'ús futur. Determinats índexs es defineixen d'acord amb els requeriments expressats pels usuaris. Si un determinat índex no es fa servir avui en dia, pot ser perfectament que es faci servir en el futur, si així s'ha previst en

els requeriments dels usuaris. El disseny d'una base de dades no es fa només pensant en les necessitats actuals sinó preveient les futures.

2.3. Components físics

En els sistemes de gestió de bases de dades podem distingir una colla d'elements, que podem anomenar components físics de l'SGBD, que passem a explicar a continuació.

2.3.1. Fitxers de dades

Els fitxers de dades són els fitxers físics que contenen les dades, és a dir, les files de les taules i les claus dels índexs.

El fitxer físic* que ha de contenir les dades de la taula, o sigui, les pàgines que contenen les files de la taula, o les pàgines que contenen les claus, es genera tenint en compte la definició de l'espai per a taules.

* En anglès, data file.

En la definició dels espais per a taules hi ha una sèrie de paràmetres que poden tenir influència en el rendiment de les aplicacions que hi accedeixen.

Vegem les característiques dels diferents tipus d'espais per a taules i d'espais per a índexs, que tenen relació amb aspectes de rendiment:

1) Grup d'emmagatzematge. Ens determina la ubicació física del fitxer. Més concretament ens determina en quin disc o grup de discos es defineix el fitxer que ha de contenir la taula o els índexs.

En sistemes mitjans i grans que disposen de diverses unitats de disc, hem de repartir els espais per a taules d'acord amb els criteris de rendiment següents:

a) Convé ubicar les taules crítiques en discos ràpids si disposem d'unitats de discos de diferents velocitats.

b) Triar les unitats de disc que durant l'horari punta tenen poca activitat per a ubicar les taules crítiques. En especial, les taules petites i a les quals s'accedeix molt.

c) Ubicar els fitxers d'índexs d'una determinada taula en una unitat de disc diferent de la que conté les dades. Penseu que la majoria dels processos crítics accedeixen a les dades per mitjà dels índexs. La raó fonamental d'ubicar el fitxer de dades en un disc diferent del fitxer d'índex és la d'evitar interferències d'uns accessos amb els altres (contenció de disc). Diem que es produeix con-

tenció de disc quan un programa determinat necessita llegir dades que estan ubicades en una unitat de disc, i el braç del lector està ocupat atenent una petició d'un altre programa. Es produeix el fenomen de seriació. Una petició espera que acabi l'altra.

2) Extensió primària. Quantitat d'espai en disc que s'assigna inicialment al fitxer de dades o d'índex. A mesura que s'introdueixen files a la taula es va ocupant l'espai assignat com a extensió primària. Quan s'omple (s'arriba al límit de l'extensió primària) l'SGBD sol·licita una extensió secundària al sistema operatiu, i d'aquesta manera s'amplia l'espai disponible. Cal preveure una quantitat d'espai suficient per a les necessitats de creixement d'aquesta taula durant un període de temps que es consideri oportú.

3) Extensió secundària. Quantitat d'espai en disc que s'assigna quan l'SGBD detecta que s'ha esgotat l'espai de l'extensió primària i en necessita més. L'assignació d'una extensió secundària significa que es crea un altre tros de fitxer de dades o d'índexs, segons el cas, que s'ubica en un disc del mateix grup d'emmagatzematge que l'extensió primària. A nivell lògic pertany al mateix espai per a taules. Això vol dir que l'SGBD ho tracta com una unitat i és el mateix sistema operatiu que s'encarrega de localitzar els trossos del fitxer.

Quan s'esgota l'espai de l'extensió secundària, el sistema agafa una altra extensió secundària amb les mateixes característiques, i així successivament fins a un nombre màxim de vegades que depèn del sistema operatiu.

Algunes vegades el dissenyador de la base de dades defineix la mida de l'extensió secundària com un percentatge de l'extensió inicial. El sistema operatiu fa el càlcul i agafa les extensions dinàmicament.

Des del punt de vista de rendiment, l'excés d'extensions secundàries pot tenir efectes negatius en els elements següents:

- Processos per lots o consultes d'usuari que exploren gran quantitat de files que poden estar ubicades en moltes extensions diferents.

- El fitxer d'índex desorganitzat, amb moltes extensions. L'accés a una clau pot representar llegir diverses extensions del fitxer per a seguir l'arbre de l'índex fins a localitzar la clau que es vol.

4) Percentatge d'espai lliure. Després d'un procés de reorganització de l'espai per a taules, l'SGBD deixa aquest percentatge d'espai lliure en cada pàgina, en previsió d'insercions futures. Aquest percentatge té dos efectes contraposats en relació amb el rendiment:

a) Un percentatge alt afavoreix els processos d'insercions de files. Més important és en el cas del fitxer d'índexs, quan s'insereixen claus intermèdies.

b)Un percentatge alt perjudica els processos seqüencials d'un gran nombre de files. El total de pàgines que cal llegir és superior a causa de l'espai buit que s'ha deixat en cada una d'aquestes.

5) Nivell de bloqueig. En cada espai per a taules es tria la unitat de bloqueig, és a dir, quina part de l'espai per a taules queda bloquejat mentre un procés està llegint o actualitzant una dada d'una fila. La pàgina és la unitat de bloqueig més habitual. Això vol dir que totes les files d'una pàgina estan bloquejades mentre el procés està llegint o actualitzant una fila. La fila és una altra unitat de bloqueig (només es bloqueja la fila afectada). El bloqueig de taula implica que totes les dades de la taula estan bloquejades per aquest procés.

És clar que aquest és un paràmetre que té una gran influència en la possibilitat de concurrència de diferents processos sobre les mateixes dades. El bloqueig més habitual és el de pàgina. Si en el funcionament normal de les aplicacions es detecten problemes de concurrència, es pot plantejar el canvi del bloqueig de taula al de fila.

Exemple de bloqueig de fila

Un programa vol llegir una fila i s'ha d'esperar que un altre procés alliberi la pàgina que la conté. Resulta que el segon programa té retinguda la pàgina perquè està actualitzant una altra fila de la mateixa pàgina. Aquest exemple es podria resoldre canviant el tipus de bloqueig de pàgina a fila.

6) Bloqueig d'índexs. En cada espai per a índex es tria la unitat de bloqueig, és a dir, quina part de l'espai per a índex queda bloquejat mentre un procés està llegint o actualitzant una clau. La pàgina és la unitat de bloqueig més habitual. Això vol dir que totes les claus d'una pàgina estan bloquejades mentre el procés n'està tractant una. Alguns SGBD tenen l'opció de no bloquejar l'índex en cap cas, la qual cosa dóna uns grans avantatges a l'hora d'afavorir la concurrència de processos.

7) Tipus d'espai per a taules. Alguns SGBD incorporen la possibilitat de triar el tipus d'espai per a taules amb característiques diferents. A continuació veurem els diferents tipus d'espais per a taules que troben en les darreres versions de DB2 i que amb noms similars podem trobar en d'altres SGBD:

a) Espai per a taules simple. Pot contenir una o més taules. L'espai està compost per pàgines, i cada pàgina pot contenir files de diverses taules. En altres paraules, un mateix fitxer físic pot contenir files barrejades de diferents taules.

b) Espai per a taules segmentat. Pot contenir una o més taules. L'espai per a taules està dividit en segments. Cada segment conté un nombre fix de pàgines, totes de la mateixa taula. És a dir, el fitxer físic pot contenir files de diferents taules, però sense barrejar.

Pàgines per segment

Els segments més habituals estan formats per 4, 8, 16, 32, 64 pàgines segons les característiques de l'SGBD.

c) Espai per a taules de partició o espai per a taules fragmentat. Conté una única taula amb un índex d'agrupació que fem servir per a definir cada partició

segons un rang de claus. El nombre de particions es tria segons les necessitats, i el nombre màxim ve limitat per les característiques de l'SGBD. Cada partició constitueix un fitxer de dades amb nom propi i característiques diferents (grup d'emmagatzematge, extensió primària i secundària, percentatge d'espai lliure, compressió, etc.).

L'elecció del tipus d'espai per a taules ve donat per la grandària de les taules i les previsions de concurrència:

- Les taules grans són candidates a ocupar un espai per a taules fragmentat. Es milloren les tasques d'administració. La majoria de les utilitats treballen sobre la unitat partició, i no sobre el total de la taula. A més a més, afavorim el paral·lelisme dels processos. Hi ha determinats SGBD, en combinació amb els sistemes operatius, que són capaços d'executar una determinada instrucció SQL en paral·lel entre diferents parts de la taula (les particions afavoreixen aquest paral·lelisme).

- Les taules mitjanes es poden definir cada una en un espai per a taules segmentat. La segmentació és una propietat que, en general, afavoreix el rendiment de les utilitats i no perjudica els programes d'aplicació.

- Les taules petites es poden agrupar en un espai per a taules segmentat. Es produeix un estalvi en recursos del sistema (manejament de fitxers). No convé, però, que l'espai per a taules resultant sigui massa gran (superior als espais de les taules mitjanes, per exemple). Per contra, cal tenir molt present que la majoria dels processos utilitaris actuen en l'àmbit de l'espai per a taules i, per tant, afecten, a la vegada, totes les taules agrupades en l'espai de taules. En conseqüència, no s'han d'agrupar taules de diferents aplicacions i que no tinguin els mateixos requeriments de disponibilitat (mateix horari de servei, mateixes necessitats de reorganització, etc.). Davant del dubte, sempre és millor definir taules petites en espais per a taules independents, que no agrupar-les quan no s'està molt convençut que tenen exactament els mateixos requeriments de disponibilitat. La segmentació és una manera de separar les pàgines de cada taula per a evitar contenció entre si i també per a millorar els processos seqüencials d'una taula, en què es llegeixen els segments propis de la taula i se salten la resta.

- Excepcionalment, determinades taules petites, amb molt poques insercions, i de les quals interessa mantenir un cert ordre de files de diferents taules, es poden agrupar en un mateix espai per a taules simples. Per exemple, suposem que tenim dues taules petites. La taula d'empleats i la taula de departaments d'una empresa. Cada empleat pertany a un únic departament. Suposem que el procés aplicatiu principal ha de tractar seqüencialment cada departament amb els seus empleats. Si carreguem les dues taules en un espai per a taules simple, i un programa insereix cada departament i a continuació els empleats que hi pertanyen, aconseguirem afavorir el procés principal de lectura, que trobarà les files de les dues taules en l'ordre que es vol.

8) Compressió de dades. És un paràmetre més en la definició de l'espai per a taules, que possibilita l'emmagatzematge de les dades en format comprimit.

D'entrada, la compressió de dades comporta, en la majoria dels casos, un estalvi en l'espai ocupat pel fitxer de dades.

L'SGBD, quan ha de comprimir les dades d'un espai per a taules, fa la compressió, fila a fila, a partir d'un diccionari de compressió. L'SGBD ha construït el diccionari amb anterioritat. La compressió de cada fila comporta que cada una d'aquestes ocupa menys espai i, per tant, en cada pàgina es podran emmagatzemar més files.

Com a conseqüència de l'increment del nombre de files per pàgina i, per tant, la reducció del nombre total de pàgines, s'aconsegueix una utilització inferior de la memòria intermèdia. Això comporta un millor aprofitament de les dades quan estan a la memòria intermèdia, tal com hem explicat anteriorment en el subapartat de paràmetres del sistema.

A més a més, la transferència de dades entre el fitxer i la memòria es redueix sensiblement, sobretot en processos per lots i massius, ja que les taules són més petites. També l'activitat que es grava en el diari (actualitzacions de la base de dades) es redueix considerablement. La informació que es grava en el diari és una imatge de la taula i, per tant, també va comprimida. En conseqüència, es redueix l'activitat d'entrada/sortida al diari.

Per contra, la compressió només té un efecte negatiu, que és el cost addicional que li representa a l'SGBD cada cop que ha de descompactar una fila llegida, i el cost de compactar una fila cada cop que l'ha de gravar a la taula. Aquest cost és un treball extra de CPU, que en els grans SGBD representa un petit percentatge d'increment respecte al cost dels mateixos accessos a la taula sense compressió. Malgrat això, no oblidem que el benefici total de la compressió en taules mitjanes i grans és molt superior.

Només es pot comprimir l'espai per a taules, i no l'espai per a índex. La compressió és rendible en taules mitjanes i grans. Als índexs s'accedeix més que a les taules, i tenen un volum molt més reduït.

Alguns SGBD proporcionen utilitats que són capaces de fer una anàlisi de les dades d'una taula, construir el diccionari de compressió i fer una estimació de la reducció d'espai que s'obtindrà si es tria l'opció de comprimir-la. Aquesta informació és molt útil a l'hora de decidir sobre la conveniència de comprimir un determinat espai per a taules.

No oblidem que el guany en la compressió depèn del contingut de les dades. Les taules amb predomini de columnes amb contingut alfabètic acostumen a tenir un millor guany de compressió que les taules amb predomini de camps

amb contingut binari. Per tant, l'anàlisi del programa utilitari és bàsic a l'hora de prendre la decisió.

A la pràctica, és fàcil d'aconseguir reduccions d'espai de l'ordre del 50% o 70% en taules mitjanes i grans.

9) Xifratge de dades. Per raons de confidencialitat de la informació emmagatzemada en una taula es pot triar l'opció de xifratge. Això vol dir que, quan un programa d'aplicació vol inserir o actualitzar una fila, crida una determinada rutina de xifratge, que codifica la informació, i l'escriu a la taula totalment codificada. Quan el programa vol llegir la fila, crida la mateixa rutina de xifratge i descodifica la informació.

D'aquesta manera, s'aconsegueix emmagatzemar a la base de dades la informació codificada de forma inintel·ligible per a qualsevol usuari que hi accedeix directament i que no està autoritzat a fer servir les rutines de xifratge.

Des del punt de vista de rendiment les rutines de xifratge representen una petita sobrecàrrega similar a les rutines de compressió.

Podem trobar exemples d'utilització de xifratge en bases de dades de sanitat, personal, etc.

2.3.2. Fitxers de control

Els fitxers de control* són els fitxers que contenen informació bàsica de control de funcionament de l'SGBD. Es tracta d'informació creada i mantinguda automàticament pel mateix gestor.

* En anglès, control files.

Distingirem entre dos tipus de fitxers de control, el catàleg i els fitxers de treball.

Catàleg

El catàleg* consisteix en un conjunt de taules que gestiona el mateix SGBD per a enregistrar totes les descripcions dels components del nostre sistema.

* En anglès, catalog.

Entre els components del sistema que es descriuen en el catàleg trobem els elements següents:

- Base de dades, espais per a taules, taules, vistes, índexs, columnes, etc.
- Pla d'accessos a les bases de dades per a cada programa.
- Autoritzacions d'accés a taules i d'ús de recursos del gestor.
- Control i gestió de les còpies de seguretat de les bases de dades.
- Altres.

Des del punt de vista de rendiment, hi ha poques consideracions a fer en relació amb el catàleg, ja que es tracta d'un disseny propi de l'SGBD i que en general no es pot tocar. L'únic aspecte que cal considerar és l'assignació de memòria intermèdia per a tot aquest conjunt de taules. Aquest tema ja s'ha tractat en el subapartat de paràmetres del sistema.

Fitxers de treball

Els fitxers de treball* són un conjunt de fitxers que l'SGBD gestiona i utilitza en diverses operacions en què necessita una àrea de treball per a resoldre determinades peticions (instruccions SQL).

* En anglès, work files.

Cada fitxer de treball es defineix com un espai per a taules genèric i l'SGBD el farà servir per al següent:

- Classificar files de taules. Pensem en una instrucció que conté un ORDER BY, o GROUP BY, etc., i no es disposa d'un índex que doni l'ordre que vol. L'SGBD llegeix les files de la taula, les traspassa al fitxer de treball i les classifica.

- Contenir la taula resultant d'una classificació, mentre el programa d'aplicació llegeix fila a fila.

- Contenir la taula resultant de la definició d'una vista, que requereix una composició de files que s'ha de materialitzar en una àrea de treball.

- Determinades combinacions* de dues o més taules s'han de resoldre en un espai per a taules de treball.

* En anglès, join.

- Altres.

El nombre de fitxers de treball depèn de l'estimació dels processos aplicatius que en un moment determinat poden necessitar fer servir l'espai per a taules de treball, en paral·lel.

Si es disposa de diferents unitats de discos, per raons de rendiment, convé repartir els fitxers de treball en diferents unitats per a evitar possibles contencions en el disc. Pensem que una mateixa petició pot necessitar més d'un fitxer de treball (per exemple, les classificacions).

Algunes transaccions poden necessitar classificar algunes files (lògicament en una transacció no s'haurien de fer classificacions molt grans). Si el nombre de files és molt reduït, alguns SGBD són capaços de fer la classificació directament a la memòria sense necessitat d'utilitzar els fitxers de treball. Evidentment, aquestes classificacions són molt més ràpides que utilitzant el disc.

Els processos per lots requereixen espais de treball superiors. Llavors l'espai de treball ha de ser prou gran per a encabir la taula més gran resultant del treball de tots els processos. Convé considerar els punts següents:

- La taula més gran classificada. Es recomana no seleccionar columnes innecessàries.
- La taula més gran resultat d'una vista.
- La taula més gran resultat d'una combinació, o unió de dues o més taules, etc.

Sempre cal establir un límit màxim segons els requeriments i les disponibilitats de recursos del nostre sistema.

Els fitxers de treball també es poden ajustar progressivament a partir de la informació de funcionament de l'SGBD obtinguda amb el monitor.

2.3.3. Fitxers d'auditoria

Els sistemes de seguretat i d'auditoria en un SGBD estan molt relacionats, perquè ambdós controlen els accessos als recursos i les dades de l'SGBD, però des d'òptiques diferents.

El sistema de seguretat cobreix el control d'accés al mateix SGBD, les bases de dades, els recursos del gestor, etc. El pla de seguretat determina quins usuaris poden accedir a determinats recursos, i en quines circumstàncies.

Els controls d'auditoria verifiquen el funcionament correcte del pla de seguretat i, a més, registra qui accedeix a determinades dades, o bé qui intenta accessos no autoritzats.

El control d'auditoria pretén de respondre les preguntes següents: Quins usuaris tenen privilegis per a accedir a determinats recursos? Quins usuaris accedeixen a determinades dades? Quins usuaris han fet intents d'accés a recursos no autoritzats?

La resposta a la primera pregunta es troba en el mateix catàleg de l'SGBD. Moltes taules del catàleg descriuen els objectes de l'SGBD (bases de dades, espais per a taules, taules, vistes, columnes, índexs, programes, etc.). D'altres taules contenen informació sobre les autoritzacions d'accés a cada un dels objectes anteriors, atorgades a diferents usuaris. Cada registre conté, no solament el nom de l'objecte autoritzat i l'usuari que ha rebut l'autorització, sinó l'usuari que l'ha atorgat, la data i hora, i el tipus de privilegi (lectura i/o actualització d'una taula o vista, d'una determinada columna, autorització a executar un determinat programa, etc.).

Les respostes a la segona i tercera pregunta es troben en els mateixos fitxers d'auditoria*. Es tracta de fitxers específics en què es grava tota la informació d'auditoria dels objectes que es volen controlar. Posteriorment aquesta informació es consulta i es tracta per si calen accions correctives.

* En anglès, audit files.

Com se selecciona i s'enregistra la informació d'auditoria?

En primer lloc, cal seleccionar què es vol auditar. Es pot triar el següent:

- Usuaris. Totes les operacions realitzades per un determinat usuari.
- Programes. Accessos efectuats per un determinat programa.
- Taules. Accessos sobre una determinada taula realitzats per qualsevol programa o usuari.

En segon lloc, cal activar l'enregistrament de la informació. Mitjançant un comandament s'inicia el període de temps en què l'SGBD està preparat per a gravar la informació d'auditoria seleccionada, a mesura que es va generant. L'interval de temps s'acaba amb un altre comandament.

En tercer lloc, el tipus d'informació enregistrada segons les opcions triades és la següent:

1) Intents d'accés denegats per manca d'autorització. Per exemple, un usuari ha intentat executar una instrucció d'actualització sobre una taula que només està autoritzada per a accessos de lectura. O bé un usuari ha intentat executar un programa no autoritzat.

2) Instruccions explícites d'autorització (GRANT i REVOKE) i el corresponent resultat. Queden enregistrades totes aquestes instruccions d'autorització amb informació dels usuaris que reben les autoritzacions i de l'usuari que les atorga.

3) Creació, modificació i esborrament (CREATE, ALTER i DROP) de taules seleccionades per al control d'auditoria. Queden enregistrades aquestes instruccions sobre determinades taules seleccionades.

4) Instruccions de modificació (INSERT, UPDATE i DELETE) del contingut de taules seleccionades per al control d'auditoria.

5) Accessos de lectura (SELECT) del contingut de taules seleccionades per al control d'auditoria.

6) Processos de construcció del pla d'accés a taules seleccionades per al control d'auditoria. Queden enregistrats els programes que accedeixen a les taules seleccionades.

7) Programes utilitaris sobre taules seleccionades.

Des del punt de vista del rendiment, el control d'auditoria té un cost que pot arribar a ser molt alt depenent del següent:

1) La quantitat d'informació que cal controlar. Pensem que no és el mateix controlar una taula específica, que controlar-les totes. Cada cop que un procés o usuari accedeix a una taula controlada, es grava informació en el fitxer d'auditoria. Això constitueix una operació d'entrada/sortida al disc, addicional a les del mateix procés.

2) La freqüència amb què s'accedeix a informació seleccionada per al control d'auditoria. Això és funció de l'activitat del sistema i del tipus d'informació seleccionada. Per exemple, si tenim una taula a la qual s'accedeix molt en lectura i poc actualitzada, no és el mateix controlar-la per a qualsevol tipus d'accés (AUDIT ALL) que controlar-la només per a actualitzar-la (AUDIT CHANGE).

En resum, el control d'auditoria implica incrementar el cost dels processos. En el cas del procés en línia, aquest cost és considerable en relació amb el cost total de la transacció. Recordem que un programa en línia fa molt pocs accessos al disc (o cap), mentre que cada control d'auditoria és un accés a fitxer.

Per aquesta raó, no és recomanable de fer un ús extensiu de les funcions d'auditoria. En canvi, es poden fer controls selectius de determinats objectes, o bé durant un període de temps reduït.

2.3.4. Diaris

L'SGBD registra totes les actualitzacions fetes a les bases de dades, i alguns fets rellevants del mateix gestor, a mesura que passen, van al diari*.

* En anglès, log o journal.

En el diari es registra una imatge de la pàgina de la base de dades, abans i després de cada actualització.

La informació continguda en el diari és imprescindible perquè l'SGBD pugui recuperar el contingut d'una taula en els casos següents:

- Acabament anormal d'una aplicació o d'una utilitat.
- Caiguda del mateix SGBD.
- Error físic d'accés a la base de dades.

L'SGBD grava informació en el diari en el moment en què l'aplicació pren un punt de sincronisme.

Per totes aquestes raons, el diari és un fitxer crític que es defineix al disc. Si el sistema disposa de diverses unitats de disc, convé definir el diari dual* (dues

* En anglès, dual log.

còpies del mateix diari, ubicades en unitats de disc diferents). Així, en cas de no disponibilitat d'un fitxer del diari, es pot treballar amb la còpia sense aturar cap procés.

Des del punt de vista del rendiment, el diari és un fitxer al qual s'accedeix molt per l'SGBD i que necessita un bon temps de resposta. Per aquesta raó, cal posar el diari en unitats de disc amb poca activitat. En sistemes grans, els diaris poden ocupar unitats de disc en exclusiva, per raons de volum i de temps de resposta.

Quan el diari s'omple, l'SGBD agafa un nou fitxer diari, i manté l'anterior per si necessita informació per a recuperar alguna taula. Els SGBD més grans tenen una funció addicional que els permet de copiar el contingut dels diaris plens a un fitxer diari arxivat* en una altra unitat d'emmagatzematge magnètic de menys cost. Per exemple, cartutxos magnètics.

* En anglès, archived log.

L'SGBD fa el canvi d'un diari al següent de manera automàtica. No obstant això, aquest canvi representa un cert cost per l'SGBD a l'hora de tancar un fitxer i obrir-ne un altre de nou. Aquesta activitat la fa l'SGBD enmig de l'activitat normal de les aplicacions. Per tant, convé no repetir massa sovint el canvi de diari. Llavors cada fitxer de diari ha de tenir prou grandària perquè pugui registrar les operacions durant un cert període de màxima activitat.

En sistemes grans amb molta activitat en línia, el període de temps que ha de cobrir un diari no és aconsellable que sigui inferior a trenta minuts, per exemple. Molt millor és ampliar aquest interval a diverses hores, segons la disponibilitat d'espai en disc per a ubicar diaris més grans.

2.4. Paràmetres del sistema

En el disseny físic de la base de dades hi ha un conjunt de variables que serveixen per a dimensionar els components físics, i també dimensionar el sistema.

La parametrització de la base de dades consisteix a assignar valor a cadascuna de les variables o paràmetres de definició dels components físics.

Com ja hem vist, cada SGBD concret té els seus propis paràmetres. A continuació esmentem els més importats i els que tenen significats semblants als diferents gestors:

- Mida de les pàgines que contenen les files de la taula, o bé mida de les pàgines que contenen l'arbre de l'índex.
- Nombre de pàgines assignades a la base de dades.
- Reserva d'espai lliure a cada pàgina, tant per a dades com per a índex.

- Mida de les extensions dels fitxers físics. Tant l'inicial o primària com la següent o secundària.
- Nombre màxim d'extensions del fitxer.
- Memòria intermèdia assignada per a les operacions de lectura i escriptura a la base de dades.
- Mida de les unitats de memòria intermèdia.
- Grau màxim de paral·lelisme permès en els sistemes que tenen paral·lelisme.
- Nombre màxim de processos concurrents.
- Nombre màxim de bloquejos concurrents a la base de dades, de diferents processos i usuaris.
- Ubicació en disc dels fitxers de dades (taules i índexs), fitxers de control (catàleg i fitxers de treball), fitxers d'auditoria i diaris.
- Nombre de diaris, mides, i característiques.

La llista de paràmetres és molt extensa, pròpia de cada SGBD concret, i la forma d'assignar valors ha de ser la següent:

1) Consulteu els manuals de cada SGBD el significat i els valors que pot prendre cada paràmetre.

2) Arrenqueu la base de dades amb els valors per defecte que subministra el constructor.

3) Posteriorment ajusteu els valors dels paràmetres per tal de millorar el rendiment de la base de dades, tenint en compte els punts següents:

- L'experiència del dissenyador de la base de dades.
- L'anàlisi dels resultats de rendiment que s'obtenen en els jocs de proves.

3. Gestió del rendiment

En aquest apartat, explicarem diversos elements que cal tenir en compte, relacionats amb el control i la gestió del rendiment d'un sistema de gestió de bases de dades.

3.1. Pla de les consultes

En el procés d'optimització d'una consulta, l'SGBD fa l'anàlisi de com estan emmagatzemades les dades a les quals ha d'accedir, de quins camins d'accés (índexs) disposa, i determina en quin ordre s'han d'executar les operacions elementals en què s'ha desglossat la consulta SQL. El conjunt d'operacions elementals tenen un cost que l'optimitzador s'encarrega d'estimar, segons les dades estadístiques de què disposa en el catàleg del gestor. Normalment l'SGBD disposa de diferents alternatives per a desglossar la instrucció SQL en operacions elementals. L'optimitzador s'encarrega d'estimar el cost de cada una d'aquestes i de triar l'alternativa de cost mínim.

Vegeu amb detall el procés d'optimització de les consultes en el mòdul "Altres temes de bases de dades" de l'assignatura Bases de dades II.

El pla d'accés de la consulta és el conjunt d'operacions elementals que ha triat l'SGBD per a resoldre la consulta (instrucció SQL).

La majoria dels SGBD moderns disposen de sentències o d'eines gràfiques per tal de visualitzar quin és el pla d'accés de la consulta. Això permet que el dissenyador de la base de dades comprovi quin és el camí d'accés a les dades (pla d'accés de la consulta) que ha triat l'optimitzador de l'SGBD. Si el dissenyador creu que hi ha algun altre camí d'accés millor, pot forçar l'optimitzador que el triï, per exemple, canviant l'ordre de les sentències dins la instrucció SQL, afegint un nou índex, traient un índex innecessari, etc.

Alguns gestors com l'SQL Server 2000 incorporen una eina gràfica, senzilla d'utilitzar, per a visualitzar el pla d'accés de la consulta, la qual de forma gràfica ens indica les operacions elementals en què s'ha desglossat la consulta, i també el nombre estimat de files a les quals s'ha accedit, la mida de les files, el cost estimat d'entrada/sortida, el cos estimat de CPU, etc.

D'altres gestors com l'Oracle 8i i el DB2 7.1, incorporen la sentència EXPLAIN PLAN, que genera la informació del pla d'accés de la consulta i l'emmagatzema en una taula d'usuari anomenada plan_table. A continuació es pot visualitzar aquesta informació de manera estructurada. La sentència EXPLAIN PLAN subministra informació del pla d'accés de la consulta sense que calgui executar la instrucció SQL que es vol analitzar.

Exemple de sentència EXPLAIN PLAN

Volem obtenir el pla d'accés de la consulta de la taula de clients a partir del DNI o NIF. En cas del DB2 tindríem:

```
EXPLAIN PLAN
SET QUERYNO = nn
FOR   SELECT nom, cognom, adreca, .........
      FROM clients
   WHERE dni = :hv1
```

A la taula plan_table queda emmagatzemada la informació del pla d'accés de la consulta sense necessitat d'executar la instrucció SELECT.

La visualització del pla d'accés de la consulta dóna informació estructurada o en format gràfic sobre els elements següents:

1) Les operacions físiques elementals en què es desglossa la consulta. S'especifica en quin ordre i de quina manera es resol cada operació elemental.

2) Els costos estimats de cada operació elemental, que inclou tant el cost de CPU de l'operació en si, com el cost d'entrada/sortida de llegir i escriure els resultats intermedis. El cost estimat total de la consulta és la suma dels costos estimats de les operacions elementals.

El pla d'accés de la consulta ajuda molt quan es tracta de fer l'anàlisi d'instruccions complexes. A la plan_table la sentència EXPLAIN PLAN insereix una fila per cada operació física elemental. El contingut i significat de cada columna depèn de cada gestor, per tant, cal consultar els manuals propis de l'SGBD per a tenir la informació detallada. Malgrat tot, els conceptes que manegen la majoria d'aquests són molt semblants. El procés de visualització del pla d'accés de la consulta també subministra informació de cada operació elemental en format gràfic o estructurat.

Exemple de procés de visualització del pla d'accés de la consulta amb Oracle

Vegem com a exemple alguns dels missatges que ens subministra Oracle en el procés de visualització del pla d'accés de la consulta:

a) TABLE ACCESS:
- FULL: recorre totes les files, generalment de forma seqüencial.
- CLUSTER: accés per mitjà de l'índex d'agrupació.
- HASH: ús de l'índex de dispersió.
- BY ROWID: accés per posició física de la fila.

b) INDEX:
- UNIQUE SCAN: accés a una única fila, per mitjà de l'índex, per posició.
- RANGE SCAN: accés a diverses files, per mitjà de l'índex.

c) FILTER: eliminació de files segons les condicions de selecció.

d) SORT:
- UNIQUE: classificació per a eliminar repetits.

- GROUP BY: classificació per a agrupar.
- JOIN: classificació de la primera part del MERGE JOIN.
- ORDER BY: classificació per la clàusula en no disposar d'índex.

e) NESTED LOOP: combinació per bucles imbricats.

f) MERGE JOIN: segona part de la combinació per intercalació.

Vegem a continuació amb uns exemples concrets com Oracle visualitza el pla d'accés de la consulta: Suposem les taules d'alumnes, assignatures i notes, amb els atributs següents.

```
CREATE TABLE   alumnes
               (dni              ......
               nom               ......
               cognom            ......
               adreca            ......
               poblacio          ......
               ............      ......
               ............      ......

CREATE TABLE assignatures
               (assig            ......
               nom               ......
               credits           ......
               ............      ......
               ............      ......

CREATE TABLE notes
               (assig            ......
               dni               ......
               nota              ......
               data              ......
               ............      ......
               ............      ......
```

Volem analitzar la consulta següent: relació d'alumnes amb una nota superior a 8.

```
SELECT DISTINCT n.dni
FROM assignatures a, notes n
WHERE a.assig = n.assig
   AND n.nota > 8
```

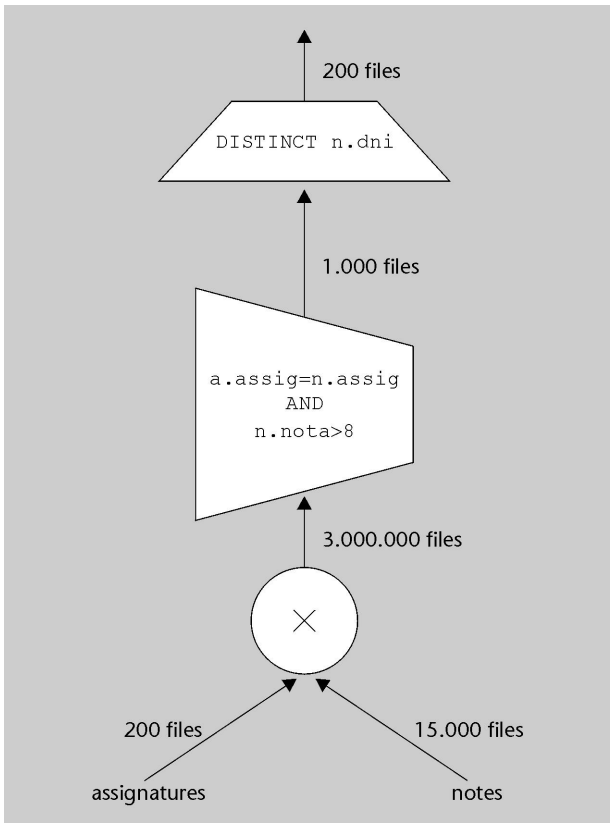
Suposem que el nombre de files que tenim a cada taula és de 3.000 alumnes, 200 assignatures i 15.000 notes. Suposem també que, de les 15.000 notes, només 1.000 són superiors al valor 8, i corresponen a 200 alumnes diferents.

Per a resoldre aquesta consulta, l'optimitzador té diverses alternatives. Vegem els arbres sintàctics de tres possibles solucions i a continuació analitzem els resultats obtinguts segons les tres alternatives:

Vegeu el concepte i la representació de l'arbre sintàctic en l'apartat "Optimització de consultes" del mòdul "Altres temes de bases de dades" de l'assignatura Bases de dades II.

1) Producte cartesià

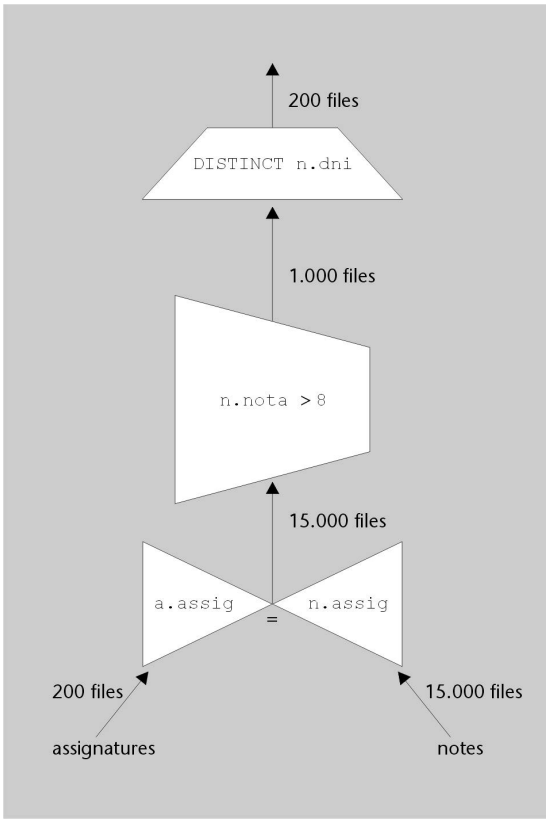
En la figura tenim el producte cartesià de les dues taules que dóna com a resultat una relació intermèdia de tres milions de files (15.000 notes multiplicat per 200 assignatures). A continuació la selecció de les notes superiors al valor 8, que com sabem són 1.000 notes. I, per últim, la projecció que ens donen els 200 alumnes diferents que han obtingut puntuacions superiors al 8.



Aquesta alternativa és més teòrica que no pas real, ja que, en l'actualitat, no hi ha cap SGBD que no sigui capaç de fer una combinació natural en lloc del producte cartesià.

2) Combinació no filtrada

La segona alternativa, que podem veure en la figura següent, correspon a la combinació natural de les 15.000 notes amb les 200 assignatures, que donen com a resultat 15.000 noves files, que un cop filtrades per la selecció de les notes superiors a 8 queden en 1.000 notes. Després de la projecció resulten els 200 alumnes diferents buscats.



Si visualitzem el pla d'accés de la consulta d'aquesta segona alternativa, Oracle ens donaria la informació estructurada de la forma següent:

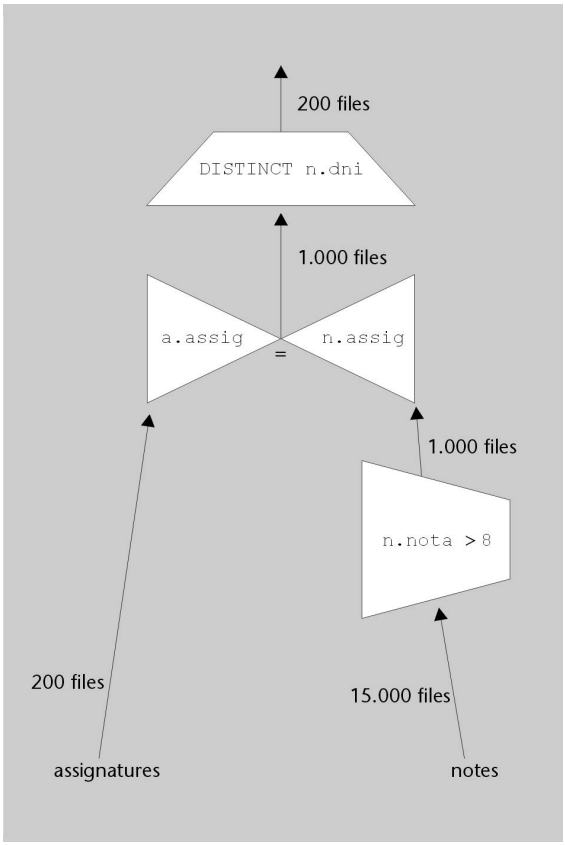
```
•     SORT UNIQUE
      o     MERGE JOIN
            •     SORT JOIN
                  •     TABLE ACCESS FULL assignatures
            •     SORT JOIN
                  •     TABLE ACCESS FULL notes
```

La forma de llegir aquesta estructura és de dins a fora.

Aquest cas es donaria en el supòsit que les taules no tinguessin cap clau primària, ni disposessin de cap índex. L'SGBD comença llegint seqüencialment totes les files de la taula de notes (TABLE ACCESS FULL) i a continuació les classifica (SORT JOIN) per l'atribut de combinació (assig). Fa el mateix amb la taula d'assignatures. A continuació fa la combinació intercalada (MERGE JOIN) de les dues taules i, per últim, la classificació (SORT UNIQUE) per a eliminar els duplicats.

3) Combinació filtrada

La tercera alternativa, que podem veure en la figura següent, consisteix en el filtratge per a selecció de les notes superiors a 8 que, com sabem, són 1.000 files. Després la combinació natural d'aquestes 1.000 files amb les 200 assignatures donen com a resultat unes altres 1.000 files, que després de la projecció final quedaran els 200 alumnes.



Si visualitzem el pla d'accés de la consulta d'aquesta tercera alternativa, Oracle ens donaria la informació estructurada de la forma següent:

```
•     SORT UNIQUE
      o     NESTED LOOP
            ▪     TABLE ACCESS FULL assignatures
            ▪     TABLE ACCESS BY ROWID notes
                  •     INDEX RANGE SCAN notes
```

Aquest cas es dóna quan les taules tenen definides les claus primàries, i un índex addicional a la taula de notes per la columna nota. L'SGBD comença llegint el rang de claus supe-

riors al valor 8 de l'índex (INDEX RANGE SCAN). A continuació accedeix per posició (BY ROWID) a les files que compleixen aquesta condició de la taula de notes. Llegeix seqüencialment totes les files de la taula d'assignatures (TABLE ACCESS FULL). Després fa la combinació natural (NESTED LOOP) de les dues taules resultants i, per últim, la classificació (SORT UNIQUE) per a eliminar els duplicats.

Com podem veure, cada alternativa té un cost diferent, que ve donat, entre altres coses, pel nombre de files que l'SGBD manipula en les diverses operacions elementals en què ha desglossat la consulta. En els arbres sintàctics podem veure el nombre de files que manipula en les relacions intermèdies.

Com ja sabem, l'optimitzador triarà l'alternativa que tingui un cost més petit. En aquest cas, és la tercera alternativa (combinació filtrada).

Podem repetir el mateix exemple que hem vist amb Oracle, amb altres gestors com SQL Server i DB2, i veurem com obtenim el mateix tipus d'informació, però amb paraules semblants o de forma gràfica.

Per acabar, podem esmentar unes regles heurístiques (sintàctiques) que podem aplicar per a millorar el procés d'optimització de les consultes, que són les següents:

1) Separar la consulta en condicions simples.

2) Dins l'arbre sintàctic, reordenar les operacions, de manera que les seleccions es puguin baixar al màxim possible. Això vol dir que s'executaran al més aviat possible i, per tant, disminuirà la cardinalitat de les operacions intermèdies. Aquesta regla, precisament, és la que ens ha permès, en l'exemple anterior, filtrar la combinació segons la figura 3 i obtenir el millor arbre sintàctic.

3) També cal baixar, tant com es puguin, les projeccions. Això també fa disminuir la cardinalitat de les operacions intermèdies.

3.2. Monitors

Els monitors són els components dels SGBD que tenen com a missió informar del funcionament del gestor des del punt de vista de consum de recursos (CPU, entrada/sortida, memòria), treball realitzat per les aplicacions, temps de resposta, accessos al disc, etc.

L'administrador de la base de dades és la persona responsable de vigilar el funcionament de l'SGBD. La informació subministrada pels monitors li serveix per a analitzar el funcionament de les aplicacions i de les bases de dades. De l'anàlisi, se'n pot deduir algunes correccions en el disseny de la base de dades i en el funcionament de les aplicacions per a millorar el rendiment global.

Els monitors solen subministrar informació dels tipus següents:

- En línia: en el mateix moment en què es produeix la informació. La majoria dels monitors despleguen un conjunt de menús de pantalla per a facilitar la

recerca de la informació. Amb diferents graus de detall (per exemple, de transacció, programa...) i la possibilitat d'agrupar la informació a diferents nivells (per exemple, dades sumades per grups de programes que s'executen en un moment determinat). A més a més, hi ha informacions concretes que es poden visualitzar de forma gràfica, per mitjà d'histogrames, etc.

- Per lots: informació recollida durant un període de temps i que s'explota en processos per lots que permeten d'obtenir dades estadístiques i comparatives entre diferents intervals de temps, diferents aplicacions, agrupada per diferents conceptes com ara, transaccions, programes, etc.

La informació que cal analitzar i que es pot recollir en cada execució de programa d'aplicació pot ser del tipus següent:

1) Temps de resposta: interval de temps que el programa ha invertit en accessos a la base de dades. És a dir, és la suma del temps de consum de CPU, temps d'entrada/sortida a la base de dades, temps d'espera per contenció amb altres processos, temps d'espera del mateix gestor, etc. Cal diferenciar clarament de la resta de temps que el programa dedica al procés mateix de l'aplicació no relacionat amb la base de dades, com ara càlculs, comunicacions amb altres sistemes, accés a fitxers externs, etc. El temps de resposta total de bases de dades es desglossa de la manera següent:

a) Consum de recursos: consum de CPU, és a dir, el temps que la unitat central ha dedicat a processar el conjunt de les instruccions SQL del programa.

b) Entrada/sortida: temps que el programa està esperant que es faci una operació d'entrada/sortida. Per tant, només cal comptabilitzar les lectures i les escriptures síncrones, és a dir, el temps que el programa està esperant. No s'han de comptabilitzar les lectures i les escriptures asíncrones que fa l'SGBD en paral·lel amb les tasques de l'aplicació.

Exemples de temps d'entrada/sortida

1) Lectura de DNI d'una fila de la taula de clients. El programa emet la instrucció de lectura i es queda esperant fins que el gestor li subministra la fila que volia. En aquest cas concret, ha de llegir unes pàgines de l'arbre de l'índex i la pàgina de dades. Si prèviament no hi ha cap d'aquestes pàgines en memòria, cal anar-les a buscar al disc. Per tant, es faran dos, tres, quatre, etc. lectures al disc. Si alguna d'aquestes pàgines es troba en la memòria, no caldrà llegir-la del disc. El cas òptim en què totes les pàgines necessàries estan en memòria, no hi ha operació d'entrada/sortida.

2) El programa actualitza dades d'un client. La pàgina que conté la fila actualitzada s'ha d'escriure al disc. Alguns SGBD han de fer l'escriptura de forma síncrona abans de continuar amb el programa d'aplicació. Altres gestors són capaços de guardar l'actualització a la memòria i continuar el programa, mentre es fa l'escriptura al disc de forma asíncrona. En aquest darrer cas no hi ha temps d'espera per l'entrada/sortida síncrona.

3) El programa vol llegir seqüencialment tots els clients de la taula. Quan el programa envia la instrucció per a llegir la primera fila, l'SGBD ha d'iniciar una lectura síncrona al disc (si la pàgina no està a la memòria). Al mateix temps, el gestor inicia la lectura seqüencial de grups de pàgines per endavant en previsió de les demandes que li farà el programa d'aplicació. Quan el programa sol·licita la segona fila, el gestor ja l'ha llegida i la

té a la memòria. I així successivament. En conseqüència, la resta de les lectures són asíncrones i, per tant, no hi ha temps d'espera.

c) Contencions: temps en què l'aplicació està esperant un recurs de base de dades que en aquest moment està bloquejat per un altre procés. El primer procés espera fins que el segon allibera el recurs, i el pot agafar. El recurs pot ser de diferents tipus. Per exemple:

- Taula. El programa intenta llegir files d'una taula que està retinguda, en exclusiva, per un altre procés, com pot ser una reorganització.
- Pàgina. El programa intenta llegir una fila que pertany a una pàgina que està essent actualitzada per un altre procés.
- Fila. El mateix cas anterior en què ambdós programes tracten la mateixa fila.
- Índex. El programa intenta llegir una pàgina d'índex que està retinguda en exclusiva per un programa utilitari, o bé que un altre procés l'està fent servir.

2) Activitat de les aplicacions: El monitor subministra informació del nombre d'instruccions SQL de cada tipus que executa cada programa d'aplicació, per exemple, del tipus SELECT, UPDATE, INSERT, ..., o bé CREATE, DROP, ALTER, ..., o bé GRANT, REVOKE, o COMMIT, ROLLBACK.

Aquesta informació ens dóna una primera aproximació de l'activitat sobre la base de dades. Per comparació amb la resta d'informació del monitor (temps de resposta, consum, contenció) podem detectar fàcilment els programes que s'aparten de la mitjana. Per exemple, més consum de CPU per instrucció SQL, més temps d'espera per contenció, etc. Aquesta informació és el punt de partida per a seleccionar els programes o consultes d'usuari sospitosos d'un consum o temps d'espera excessiu.

A més a més, el monitor en línia ens pot donar la instrucció SQL que s'executa en cada moment de cada programa.

3) Memòria intermèdia*: el monitor subministra informació d'ús de la memòria intermèdia en les operacions de lectura i escriptura a la base de dades. En concret, el monitor comptabilitza el nombre de pàgines llegides de la memòria intermèdia.

* En anglès, buffer pool.

D'altra banda, el monitor comptabilitza el nombre de pàgines llegides dels fitxers quan no es troben a la memòria. Recordem que un cop llegides del fitxer passen a la memòria intermèdia.

Recordem, també, que per a les aplicacions, com més pàgines trobin a la memòria intermèdia i menys n'hagin de llegir dels fitxers, millor serà el temps de resposta.

Un primer valor de referència sobre l'ús de la memòria intermèdia consisteix a calcular la relació entre el nombre de pàgines llegides dels fitxers i el nombre

de pàgines llegides de la memòria intermèdia. Estadísticament, aquest valor es manté constant en períodes de temps d'activitat similar. Quan es produeixen variacions d'aquest valor, cal analitzar-ne el motiu. Normalment és perquè ha augmentat l'activitat de les aplicacions, i necessiten més informació de la base de dades. Aquest pot ser un indici de la necessitat d'incrementar el nombre d'unitats de memòria intermèdia assignats, per tal que no empitjori el temps de resposta.

3.3. Benchmarks

Els benchmarks són un conjunt de proves que s'estableixen per a veure el comportament de diferents sistemes, la seva funcionalitat, prendre mesures de rendiment i comparar els resultats obtinguts.

Alguns benchmarks són famosos. Es tracta de proves estandarditzades per a veure la potència de cada sistema, i el seu comportament davant de situacions de màxima activitat*. D'altres proves es fan a mida segons les comparacions que es pretenen de fer.

* En anglès, stress.

Dissenyar un benchmark és sempre difícil. En el cas dels SGBD hem de tenir en compte una sèrie de característiques específiques, que són les següents:

a) Els SGBD accedeixen a la base de dades fent servir el llenguatge SQL que és estàndard. Però malgrat això, cada gestor té diferències de llenguatge pròpies, que segons el joc de proves triat, pot tenir més o menys importància amb vista a les comparacions.

b) Cada SGBD té un optimitzador propi que té un paper clau a l'hora de triar els camins d'accés a la base de dades més adients en cada cas.

c) Els sistemes operatius sobre els que treballen els SGBD poden ser iguals, semblants o diferents, segons els casos. Per tant, quan es fan proves de gestors que corren sobre plataformes no equivalents, cal considerar que el benchmark és del conjunt i no solament del gestor. Normalment cada SGBD s'ha dissenyat tenint en compte la plataforma que fa servir per a treure'n el màxim avantatge.

d) També el maquinari que s'utilitza per a fer les proves del benchmark pot ser el mateix per a tots els gestors; pot ser semblant o diferent.

e) Les eines d'afinament i de mesura del rendiment acostumen a ser pròpies de cada plataforma, i moltes vegades els resultats que subministren no són fàcils de comparar.

Totes aquestes circumstàncies poden dificultar la comparació dels resultats obtinguts en les proves de rendiment. Per tant, és molt important fer un bon disseny del benchmark i una implantació de les proves de rendiment ajustades als requeriments que han de cobrir tots els gestors.

Les condicions per a fer un bon disseny del benchmark són les següents:

1) Preparar un joc de proves amb taules carregades amb un nombre significatiu de files i amb continguts i distribucions de claus similars o comparables a la vida real. En cas de no disposar de dades reals, generar files amb continguts aleatoris, però coherents.

2) Seleccionar programes que siguin representatius de l'activitat sobre la base de dades que es preveu que l'aplicació farà a la vida real. Seleccionar els accessos més importants pel seu cost, durada o ús repetitiu.

3) Seleccionar consultes d'usuari que siguin representatives de l'activitat més important prevista sobre la base de dades.

4) Òbviament carregar les mateixes dades a tots els SGBD, i provar amb els mateixos programes i les mateixes consultes.

5) Comprovar que els optimitzadors de tots els SGBD seleccionen els camins d'accés previstos teòricament. Si algun optimitzador tria un camí d'accés no previst, cal analitzar el cas a fons abans de donar com a bons els resultats obtinguts.

6) Provar cada gestor amb la plataforma definitiva (maquinari i sistema operatiu). Recordem que si els gestors corren sobre la mateixa plataforma, les proves de rendiment mostren les diferències dels mateixos gestors. Però si les plataformes no són semblants, els resultats posen de manifest les diferències de rendiment dels conjunts: gestor + plataforma.

7) Comparar resultats homogenis en cada prova:
- Consum de CPU.
- Temps de resposta.
- Ocupació de recursos, espai en disc, ocupació de memòria, etc.

8) Fer els ajustos i les millores pertinents a cada gestor, i repetir el cicle de proves i comparació de resultats.

3.4. Elecció d'un SGBD

En determinades situacions ens podem trobar davant la disjuntiva d'haver de triar entre diferents SGBD per a implementar el nostre sistema de base de dades.

La tria és important ja que condiciona el desenvolupament i explotació d'aplicacions futures i, d'altra banda, la tria és difícil, ja que hi intervenen moltes circumstàncies de caire tècnic, i no tècnic, que són de difícil avaluació.

No hi ha un SGBD que sigui "el millor", sinó que cada un té unes característiques pròpies que el fa ser "el millor" en el seu entorn, és a dir, per a un determinat maquinari, per a un determinat sistema operatiu, per a un determinat volum de base de dades i per a unes aplicacions d'unes característiques determinades.

A continuació exposem una metodologia d'ajuda a la tria de l'SGBD més adient a cada cas concret, basada en la determinació dels requeriments que s'han de cobrir, i l'anàlisi de les qualitats dels diferents gestors de base de dades.

La metodologia consta dels passos següents:

1) En primer lloc, cal determinar les característiques de les aplicacions en relació amb els accessos a les bases de dades. Dit d'una altra manera, quins són els requeriments d'aplicació, d'usuari, de seguretat, d'integritat, de concurrència, etc., respecte a la base de dades. Vegem algunes característiques que cal tenir en compte:

- Volum d'informació que s'ha d'emmagatzemar a la base de dades. Necessitat de distribuir entre diferents centres, o necessitat de concentrar les dades en un sol gestor.

- Nombre de transaccions en hora punta.

- Nombre d'accessos a base de dades per transacció.

- Temps mitjà de resposta per transacció.

- Nombre d'usuaris concurrents fent consultes no planificades.

- Necessitats de protegir una determinada informació considerada confidencial, com per exemple, dades sanitàries, de nòmina, de comptabilitat, etc.

- Requeriments de concurrència de diferents processos (transaccions, programes per lots, consultes d'usuari) sobre les mateixes dades, amb garantia d'integritat.

- Etc.

2) Determinar els criteris que s'han de fer servir per a avaluar els SGBD, i les condicions mínimes que han de complir. Els criteris d'avaluació són una con-

seqüència de les característiques de les aplicacions que hem determinat. Podem agrupar els criteris en els tres grans grups següents:

a) Relatius al desenvolupament i manteniment d'aplicacions:

- Esforç necessari per a desenvolupar el sistema.
- Esforç necessari per a mantenir les aplicacions quan es produeixen canvis a la base de dades (reestructuracions).
- Possibilitat de construir prototipus del sistema, fiables i de forma fàcil, que ens permeti de desenvolupar-lo més còmodament i més ràpidament.
- Esforç necessari per a fer consultes no previstes (consultes ad hoc).
- Possibilitat que ofereix el diccionari en el desenvolupament, manteniment i documentació d'aplicacions.

b) Relatius a l'operació de la base de dades:

- Eficiència global d'utilització dels recursos del sistema (consum de CPU, temps de resposta, gestió de memòria, etc.).
- Possibilitat de control de concurrència de diferents processos (en línia, per lots, consultes d'usuari) sobre la mateixa base de dades, compartint taules.
- Garantia d'integritat de les dades.
- Possibilitat de control de la seguretat d'accés a la base de dades i possibilitat de protegir la confidencialitat de la informació emmagatzemada.
- Mitjans que s'ofereixen per a les tasques habituals d'administració de la base de dades, com ara còpies de seguretat, reorganitzacions, recuperacions i restauracions. Els mitjans que s'ofereixen poden ser del tipus d'eines interactives d'administració, eines d'ajuda a la gestió, programes utilitaris, etc.
- Facilitat d'operació de l'SGBD. Arrencar, aturar el sistema, una base de dades en concret, etc. Arrencar després de la caiguda del sistema.
- Fiabilitat i disponibilitat de servei de l'SGBD.
- Facilitat de distribució de les dades, de connexió amb altres bases de dades, i d'accés i tractament en l'ordinador personal.

c) Relatius a la comercialització del producte:

- Consolidació i dimensió de les empreses que desenvolupen, comercialitzen i distribueixen el producte.

Grau de difusió del producte (local, mundial). Cal considerar especialment aquelles aplicacions que funcionen en altres empreses i que són similars al nostre sistema.

- Plataformes i configuracions de maquinari i de sistema operatiu en què pot treballar el gestor.

- Freqüència de millora del producte. Aparició de noves versions.

- Possibilitat de formació. Assistència a cursos, seminaris, conferències d'aprenentatge del gestor.

- Disponibilitat de suport tècnic i de formació dels proveïdors. Suport presencial a escala local, regional, nacional, internacional. Suport telefònic. Suport electrònic, reparació via mòdem.

- Qualitat de la documentació del producte. Manuals, llibres, CD, etc.

- Cost del producte i garanties.

3) Identificar els SGBD disponibles en el mercat i seleccionar aquells que compleixen els criteris anteriors.

4) Analitzar i avaluar cada SGBD seleccionat en relació amb cada un dels criteris anteriors. Podem triar entre dos tipus d'avaluacions diferents: quantitativa i qualitativa.

a) Avaluació quantitativa. Consisteix a assignar un pes numèric a cada un dels criteris fixats per a avaluar els SGBD. El pes numèric s'assigna segons la importància relativa que es dóna a cada criteri. A continuació es puntua el grau de compliment de cada criteri a cada SGBD.

b) Avaluació qualitativa. La persona que fa les avaluacions qualifica el grau de compliment de cada criteri de la forma més objectiva possible.

5) Resum comparatiu de l'anàlisi anterior. Si s'ha fet l'avaluació quantitativa n'hi ha prou de sumar les puntuacions obtingudes per cada SGBD. Si s'ha fet l'avaluació qualitativa, cal comparar el conjunt dels criteris fixats.

6) Recomanar l'opció més ben avaluada, que ha obtingut més puntuació o més ben avaluada qualitativament. Per acabar, cal proposar un possible pla per a implementar el sistema.

4. Informe del dissenyador de la base de dades

Una etapa important del disseny de la base de dades és la documentació. No es pot considerar acabat el disseny si no s'han documentat, convenientment, totes i cada una de les seves etapes: el disseny conceptual, el disseny lògic i el disseny físic.

En aquest apartat, farem esment del contingut de la documentació apropiada per a l'etapa del disseny físic. És la documentació que prepara el dissenyador de la base de dades per a lliurar a l'administrador de la base de dades.

Òbviament, partim de la documentació de les etapes anteriors (disseny conceptual i disseny lògic). En aquesta etapa es completa la documentació anterior i s'amplia amb els aspectes propis de l'etapa de disseny físic. Aquestes etapes són les següents:

1) Disseny de taules

a) Definició de columnes: nom, tipus, precisió, escala, valor per defecte, indicador de si admet nul.

b) Clau primària i claus alternatives.

c) Claus foranes i taules a les quals fan referència.

d) Restriccions d'integritat.

2) Índexs que les diferents aplicacions han creat per a millorar els accessos a les taules

a) Columnes que formen cada índex.

b) Característiques: ascendent, descendent, únic, no únic, índex d'agrupació.

3) Espai per a índex

a) Volum previst. Nombre de claus.

b) Creixement previsible.

c) Grau de volatilitat de les claus. Nombre d'altes i baixes de claus, i distribució física. Previsió d'espai lliure per a afavorir el creixement.

d) Necessitats de reorganització de cada índex.

4) Espais per a taules

a) Tipus d'espai per a taules: espai d'agrupació, espai de taula simple, espai fragmentat, espai d'objectes grans.

b) Volum previst. Nombre de files. Nombre d'extensions del fitxer.

c) Creixement previsible.

d) Grau de volatilitat. Nombre d'altes i baixes. Previsió d'espai lliure per a afavorir el creixement dins de cada pàgina, entre pàgines.

e) Necessitat de reorganització de l'espai per a taules, per a recuperar espais lliures i per a recuperar l'ordenació de les files segons l'índex d'agrupació (si n'hi ha).

f) Necessitat de distribució del fitxer físic en diferents unitats de disc, unitats especials amb millor temps d'accés, unitats de disc amb dispositiu de mirall, etc.

5) Peculiaritats específiques d'aquest disseny. És a dir, totes aquelles característiques del disseny físic que s'aparten de les consideracions normals i que tenen la seva raó de ser, com ara:

a) Desnormalització de la base de dades per tal de millorar el rendiment de les aplicacions. Cal documentar, amb detall, quins fets no estan normalitzats, com s'han introduït les desnormalitzacions i per quina raó (normalment per qüestions de rendiment). També cal documentar de manera molt acurada com s'ha de controlar la coherència de les dades desnormalitzades, i quins processos faran el control.

b) Redundàncies. És la informació que està repetida a més d'un lloc. Per exemple, una columna pròpia d'una taula, es duplica en una altra taula, únicament i exclusivament perquè els processos que accedeixen a la segona taula disposin de la informació de la columna sense necessitat d'accedir a la primera taula. Es tracta de casos especials amb fortes exigències de rendiment. També en aquest cas cal documentar quins processos són els encarregats de duplicar la informació, i quins processos han de fer el control de la coherència de les dades.

c) Alternatives de disseny. Moltes vegades, quan es fa el disseny físic de la base de dades trobem més d'una alternativa o solució per a cada part, i s'implementa la que es considera millor, sense menysprear la resta de solucions. També és convenient documentar les diferents alternatives estudiades, amb vista a possibles replantejaments i canvis en el futur.

Resum

En aquest mòdul hem tractat la forma d'implementar el disseny físic de la base de dades:

1) Hem vist la forma d'adaptar-lo a un SGBD concret, amb les seves característiques pròpies.

2) Hem vist en quins casos és convenient definir índexs per a millorar els accessos a les taules, i en quins casos no és convenient de fer-ho. Hem repassat la seva influència en el rendiment de les aplicacions.

3) Hem repassat el funcionament dels components físics de l'SGBD, com ara els fitxers de dades, el catàleg, els fitxers de treball, d'auditoria, els diaris, i com influeixen en el rendiment del gestor.

4) Hem revisat la parametrització de l'SGBD, i la seva repercussió en el rendiment del gestor.

5) Hem interpretat la informació obtinguda del pla de les consultes per tal de predir els accessos a la base de dades en cada instrucció SQL.

6) També hem interpretat la informació que ens subministren la majoria dels monitors en relació amb els temes de rendiment.

7) Hem establert les idees bàsiques per a construir un benchmark entre diferents SGBD, i una metodologia per a seleccionar el més adequat als nostres requeriments.

8) Per acabar, hem vist la necessitat o conveniència que el dissenyador expliqui a l'administrador de la base de dades tot el procés de disseny de la base de dades, i ho documenti convenientment.

Activitats

1) Amb l'SGBD que tingueu a l'ordinador de casa vostra, descobriu alguns dels aspectes relacionats amb el disseny físic i el rendiment que hem tractat en aquest mòdul. Descobriu com el vostre SGBD implementa algunes d'aquestes funcions, i amb quines eines d'ajuda (si en té):
a) Eines de gestió d'esquemes de la base de dades. Faciliten les tasques d'administració (creació i manteniment de taules, índexs, espais per a taules, etc.).
b) Eines de gestió d'espais. Control de l'estat dels espais d'emmagatzematge. Volum ocupat, espai lliure, factors de compressió, etc.
c) Control dels camins d'accés a les dades. Consulta del pla de les consultes. Eines d'ajuda.
d) Monitors de rendiment. Informació que subministra l'ajust i millora del rendiment.
e) Etc.

2) Consulteu a Internet les pàgines web de cada fabricant d'SGBD i descobriu els aspectes que remarquen cada un d'aquests en relació amb les excel·lències de cada producte quant als temes d'ajust i gestió del rendiment.

Exercicis d'autoavaluació

1. El departament d'informàtica d'un important ajuntament ha dissenyat una base de dades de vehicles i contribuents per a la gestió de l'impost municipal de vehicles de motor. Ha definit dues taules de la manera següent:

```
CREATE TABLE   contribuents
               (dni              ......
               nom               ......
               cognom            ......
               adreca            ......
               poblacio          ......
               telefon           ......
               sexe              ......
               data_naixement    ......
               ............      ......
               ............      ......
               PRIMARY KEY (dni)
               )
               ............
               ............

CREATE TABLE vehicles
               (matricula        ......
               tipus_vehicle     ......
               marca             ......
               model             ......
               numero_bastidor   ......
               data_matriculacio ......
               data_alta         ......
               data_baixa        ......
               dni_propietari    ......
               ............      ......
               ............      ......
               PRIMARY KEY (matricula)
               FOREIGN KEY (dni_propietari)
                       REFERENCES contribuents
               )
               ............
               ............
```

Els funcionaris de l'ajuntament fan diferents consultes sobre la base de dades. Es vol saber quin tipus d'accés farà l'SGBD a cada taula. Dit d'una altra manera, què ens dirà el pla de les consultes (taula plan_table) de cada un dels accessos següents?

a) Recuperar les dades del vehicle a partir de la matrícula:

```
SELECT   ...........
FROM     vehicles
WHERE    matricula   =  :hv1
```

Quin tipus d'accés s'empra?

b) La guàrdia urbana ha localitzat un vehicle abandonat sense les plaques de la matrícula, però ha pogut llegir el número del bastidor. Per a recuperar les dades del vehicle a partir del número de bastidor:

```
SELECT   ...........
FROM     vehicles
WHERE    numero_bastidor = :hv2
```

Quin tipus d'accés s'empra?
Com es pot millorar?

c) Emissió dels rebuts anuals de l'impost de vehicles de motor:

```
SELECT   nom, cognom, adreca, poblacio, matricula, ...
FROM     contribuents a,
         vehicles b
WHERE    a.dni = b.dni_propietari
   AND   data_baixa IS NULL
```

Tipus d'accés a la taula de contribuents?
Tipus d'accés a la taula de vehicles?
Tipus de combinació?
Com es podria millorar?

d) Suposem ara que afegim l'índex següent sobre la clau forana de la taula de vehicles:

```
CREATE   INDEX ind5
         ON vehicles
         (dni_propietari ASC)
         ...........
         ...........
```

Com influeix la mateixa consulta anterior?
Tipus d'accés a la taula de contribuents?
Tipus d'accés a la taula de vehicles?
Tipus de combinació?

2. Esmenteu en quins casos és convenient de definir un índex.

3. Quins són els fitxers auxiliars (o components físics) que necessita l'SGBD per a treballar? Definiu-los tots.

Solucionari

1. Recordem que la definició de la clau primària a cada taula implica l'existència d'un índex. Mentre que la definició de la clau forana no implica l'existència de cap índex a menys que es defineixi de manera explícita.

a) L'accés és de tipus accés directe per índex. En aquest cas, la clau primària de la taula de vehicles (segons Oracle: INDEX UNIQUE SCAN vehicles).

b) Accés és seqüencial. En aquest cas, no hi ha cap índex definit sobre la columna numero_bastidor (segons Oracle: TABLE ACCESS FULL vehicles).
Per a millorar l'accés es pot definir un índex únic sobre aquesta columna, i l'accés seqüencial canviaria a accés directe per índex.

c) L'accés és directe per índex a la taula de contribuents.
Accés seqüencial a la taula de vehicles. No hi ha cap índex definit sobre la columna de la clau forana.
Respecte del tipus de combinació, atès que no es disposa d'índex sobre la segona taula i el nombre de files qualificades pot ser molt gran, el mètode escollit per l'optimitzador serà el merge join. Això vol dir el següent:
- Emmagatzemar les files llegides de la primera taula en un fitxer de treball.
- Classificar les files de la segona taula segons la clau forana. Emmagatzemar el resultat en un fitxer de treball.
- Combinar les files de la primera taula amb les de la segona segons el predicat de combinació.
- Aplicar la resta de predicats.

La manera de millorar els accessos consisteix a definir un índex sobre la clau forana, tal com ens proposa la pregunta següent (d).

d) L'accés és directe per índex a la taula de contribuents i directe per índex a la taula de vehicles. Ara es disposa de l'índex ind5 sobre la clau forana.
Respecte del tipus de combinació, es disposa d'índexs sobre els predicats de combinació de les dues taules. Per tant, el mètode triat serà el nested loop join. Això vol dir el següent:
- Seleccionar cada una de les files de la primera taula.
- Per cada una d'aquestes es busquen totes les files coincidents de la segona taula.
- S'apliquen la resta de predicats.

El resultat final és que no es fa cap tipus de classificació i, per tant, en aquest cas, el segon mètode de combinació és molt més eficient que el primer.

2. És convenient de definir un índex en els casos següents quan:
- Volem garantir la unicitat d'alguns atributs definint un índex únic.
- Volem garantir l'ordenació de les files dins la taula definint un índex agrupat.
- Volem millorar la relació d'integritat referencial entre dues taules, definint un índex sobre la clau forana.
- Volem millorar determinats accessos a la base de dades definint un índex sobre les columnes que formen la clàusula WHERE.
- Volem millorar determinades instruccions SQL definint índexs sobre columnes de les clàusules ORDER BY o GROUP BY.
- Volem millorar combinacions de dues o més taules definint índexs sobre les columnes que formen els predicats de combinació.

3. Components físics que necessita l'SGBD:
- Els fitxers de dades són els fitxers físics que contenen les files de les taules i les claus dels índexs.
- El catàleg és el conjunt de taules que gestiona el mateix SGBD per a enregistrar les descripcions dels components del sistema.
- Els fitxers de treball que gestiona l'SGBD com a àrea de treball per a resoldre determinades instruccions SQL.
- Els fitxers d'auditoria que fa servir l'SGBD per a gravar informació d'auditoria dels objectes que vol controlar.
- Els diaris són els fitxers que l'SGBD fa servir per a gravar cronològicament les modificacions fetes a la base de dades, i que calen en les tasques de recuperació.

Glossari

catàleg
Conjunt de taules que contenen les metadades de la base de dades.

diari
Fitxer que fa servir l'SGBD per a gravar cronològicament les modificacions fetes a la base de dades, i que cal en les tasques de recuperació.

extensió
Unitat d'assignació d'espai en un dispositiu perifèric. Cada extensió és un nombre enter de pàgines consecutives, i està continguda dins un fitxer.

fitxer
Unitat de gestió de l'espai en els dispositius perifèrics. Normalment el sistema operatiu gestiona els fitxers en lloc de l'SGBD.

fitxer d'auditoria
Fitxer cronològic dels fets (accessos a la base de dades i intents) que l'SGBD selecciona per a fer el control d'auditoria.

fitxer de treball
Fitxers que l'SGBD fa servir com a àrea de treball per a resoldre determinades instruccions SQL.

grup d'emmagatzematge
Espai d'emmagatzematge en dispositius perifèrics que es distribueix entre diferents grups per a fer la seva gestió més flexible.

Bibliografia

Hansen, G.W.; Hansen, J.V. (1997). Diseño y administración de bases de datos (2a. ed.). Prentice Hall.

Howe, D.R. (1989). Data Analysis for Data Base Design (2a. ed.). Edward Arnold.

Kirkwood, J. (1993). High Performance Relational Database Design. Ellis Horwood.

Shasha, D.E. (1992). Databasse Tuning. A Principled Approach. Prentice Hall.

Teorey, T.J. (1994). Database Modeling & Design. The Fundamental Principles (2a. ed.). San Francisco: Morgan Kaufmann Publishers, Inc.