

ESPAI FÍSIC - OPTIMITZACIÓ D'UNA CÀRREGA DE TREBALL

Aquesta sessió té dues parts, tal com indica el títol que hi hem posat. La primera dóna una descripció bàsica de l'organització interna dels objectes a Oracle i a la segona veurem quins elements influeixen en el temps que triga una consulta, entre els quals hi ha els índexs. Acabarem veient com podem seleccionar els índexs per aconseguir un bon rendiment global per a l'execució d'una determinada càrrega de treball (un conjunt de consultes).

Espai físic. A la diapositiva 5 hi teniu, a la dreta, els tres nivells o espais que ens trobem en un SGBD:

- Lògic. Correspon al pas bàsic de disseny físic, la implementació de les relacions del disseny lògic en taules del SGBD, tenint en compte les particularitats que té com ara el repertori de tipus.
- Virtual. En aquest espai hi tenim els objectes que l'usuari del SGBD pot crear. Naturalment, taules però també índexos, vistes i moltes més coses, algunes de les quals veurem ara mateix.
- Físic. Aquí hi tenim els fitxers i altres recursos del SO sobre els quals el SGBD implementa els objectes de l'espai virtual.

Les altres dues columnes de la diapositiva 5 situen els nivells de dues arquitectures de referència que ja hem vist anteriorment: la d'ANSI/SPARC que vam veure al capítol de vistes i la de passos de desenvolupament que vam veure a la introducció. L'esquema indica el nivell d'abstracció més o menys elevat segons l'altura en què es troba cada element.

La diapositiva 6 presenta un exemple de distribució en els tres espais d'Oracle, on veiem els objectes més rellevants que poden crear en l'espai virtual: taules, particions, índexos, vistes (materialitzades o no), clusters i tablespaces. El concepte de tablespace és el que ens dóna la possibilitat de definir objectes tant voluminosos com volguem, que es poden emmagatzemar fent servir tants dispositius com calgui, superant la limitació de volum a la capacitat del dispositiu, limitació que tindríem si un objecte hagués de cabre en un de sol. La diapositiva 7 explica com s'allotgen els objectes (que, genèricament, s'anomenen segments) en Oracle:

- Per una banda, podem crear tablespaces, cadascun dels quals amb una capacitat il·limitada (mentre anem afegint els dispositius necessaris) i està repartit en diversos fitxers de diversos dispositius.


- Els fitxers es creen amb una extensió inicial i poden anar creixent afegint-hi extensions. Una extensió és un conjunt de blocs de disc consecutius (es poden llegir seqüencialment fent una sola operació de posicionament cap al primer). Les extensions no són consecutives entre elles, això permet que un fitxer creixi sense haver de moure les dades que ja té. La gestió de les extensions la fa el SGBD.
- Per altra banda, tot segment estarà assignat a un tablespace, ocupant extensions d'un o més fitxers del tablespace. Una extensió és ocupada per un únic segment.

Els tablespaces, a més, fixen una sèrie de paràmetres relacionats amb la gestió de l'espai que afectaran tots els segments que hi resideixin. Cada administrador pot decidir quins tablespaces vol tenir per una determinada BD; un conjunt típic de tablespaces és el que teniu a la diapositiva 9.

Optimització d'una càrrega de treball. A partir de la diapositiva 10 passem a considerar persones i eines que intervenen en el rendiment d'una BD. N'esmentarem uns quants i ens centrarem en el procés de decisió dels índexs a crear en funció d'un conjunt de consultes que volem optimitzar globalment.

Els rols que tenen responsabilitat en el rendiment són:

- L'administrador de la BD. És qui decideix la configuració i els paràmetres de la BD.
- El dissenyador. És qui decideix quins objectes es crearan (taules, índexs, vistes, ...) amb l'ús de sentències de DDL (data definition language - create table, ... -).
- El programador. És qui decideix quines sentències de DML (data manipulation language - select, insert, ...) s'executaran per mantenir i consultar la informació que hi ha a la BD.

Les dues eines més importants que ajuden a prendre aquestes decisions són les estadístiques que guarda el catàleg i el pla d'accés que el SGBD genera per executar una consulta. Observant-lo, es poden fer canvis en el disseny, consultes o paràmetres per millorar el rendiment. Recordeu que podeu veure estadístiques consultant algunes vistes del catàleg i que amb el client DBeaver podeu demanar el pla d'accés situant el cursor de l'editor sobre la consulta i, dins l'opció de menú "SQL Editor", seleccionar "Explain Execution Plan" o bé fer clic al botó .

Hi ha circumstàncies, alienes al pla d'accés, que també afecten al temps d'execució de les consultes, com ara les que s'enumeren a la diapositiva 13 que no tractarem en aquesta sessió. Si ens concentrem en el pla d'accés, com a dissenyadors podem millorar el rendiment duent a terme el procés que es mostra a la diapositiva 14: a partir de l'espai disponible i els requisits d'optimització (quin conjunt de consultes, que

anomenem càrrega de treball -workload- i quin objectiu -global o per cada consulta-), prendre decisions de disseny físic com ara la creació d'índexs i altres que s'enumeren. En aquesta sessió limitem el procés a la creació d'índexs per fer una optimització global del workload. A la diapositiva 15 veiem que trobar la millor solució requereix un esforç enorme si pretenem analitzar totes les possibilitats, perquè creixen exponencialment en nombre. Per això ens marcarem com a objectiu trobar una solució raonablement bona, cosa que podem fer de dues maneres (o combinant-les totes dues):

- Tenint presents les regles que s'enumeren a les diapositives 16 i 17.
- Fent servir l'algoritme greedy (semblant al greedy que vam estudiar per seleccionar vistes materialitzades) que s'exposa a la diapositiva 19 i s'aplica a un exemple a partir de la 20.

Per acabar, fixem-nos en aquest exemple. La diapositiva 20 presenta el cas que volem optimitzar. Hi ha les taules, el workload i totes les estadístiques i paràmetres que necessitem. Construïrem una taula on les files són els índexs candidats a ser creats i a les columnes hi tenim, primer, l'espai extra requerit per cada candidat, i, després, el cost de cada consulta. A la darrera columna hi posem el cost mitjà calculat tenint en compte la freqüència d'execució de cada consulta. A la diapositiva 21 teniu aquesta taula a la part inferior.

A la part superior de la diapositiva 21 hi ha l'espai ocupat si no creem cap índex (10000 blocs d'una taula i 5000 de l'altra) i el temps d'execució de les consultes en aquestes condicions:

- Q1: cal fer un recorregut complet de la taula (selecció de diverses files)
- Q2: cal fer un recorregut, en mitja, de la meitat de la taula (selecció d'una única fila)
- Q3: els algoritmes de join disponibles ens diuen que són:
 - Hash Join. Com que la mida de la taula petita (5000 blocs) és menor que el quadrat de la memòria (100000) podem fer servir aquest algoritme i el cost serà $(10000 + 5000) * 3$.
 - Sort Match. S'hauran d'ordenar les dues taules i després fer-ne el recorregut en paral·lel. El cost serà de 40000 per ordenar una taula, 20000 per ordenar l'altra i 15000 del recorregut de les taules.

El cost de Q3 és, doncs, 45000.

Abans de seguir fent càlculs hem de determinar quins són els índexs candidats:

- Q1: com que la condició és una igualtat, ens podem plantejar qualsevol de les alternatives (B+, cluster i hash). Hem de veure, però, que el hash no és factible en les condicions que el vam definir: si l'ordre dels arbres és 75, en un bucket ple

del tot hi caben 150 entrades, però la taula té 100000 files i l'atribut té 100 valors diferents. Això vol dir que hi ha 1000 repeticions per valor i que els buckets haurien de poder tenir 1000 entrades.

- Q2: igual que abans, tots tres índexs poden anar bé i ara no hi ha problema per fer un hash.
- Q3: per al hash join no cal definir cap índex i per al sort match pot anar bé tenir les taules ordenades. Els candidats són, doncs, fer una estructura cluster o posar un cluster Book(author) i un Author(name).

Vegem ara d'on surten els números de la taula:

- Espai extra
 - B+ Book. $u = 100$, $h = 2$, 1000 fulles, 10 nodes sobre les fulles, 1 rel. L'arbre ocupa 1011 blocs.
 - Cluster Book. L'arbre com abans i, a més, la taula amb espais buits ocupa 5000 blocs més.
 - B+ Author. $u = 100$, $h = 2$, 200 fulles, 2 nodes sobre les fulles, 1 rel. L'arbre ocupa 203 blocs.
 - Cluster Author. L'arbre com abans i, a més, la taula amb espais buits ocupa 2500 blocs més.
 - Hash Author. $150 * 0.8 = 120$ entrades/bucket. $\lceil 20000/120 \rceil = 167$. Més 1 bloc per a la gestió del hash.
 - Estructura cluster. L'espai buit que cal deixar és la meitat del que ocupen les taules compactes, o sigui, 7500 blocs.
- Cost Q1:
 - Amb B+ o cluster per theme. S'apliquen les fórmules corresponents.
 - Amb cluster per author. L'índex no serveix i s'ha de recórrer la taula que, com que té un cluster, ocupa 15000 blocs, no 10000.
 - Amb índexos a la taula Author. No serveixen i recorrem la taula que té 10000 blocs.
 - Amb estructura cluster. Hem de recórrer tota l'estructura que ocupa els blocs de les dues taules i, a més, amb espais buits: $(10000 + 5000) * 1.5 = 22500$.
- Cost Q2:

- Amb índexos a la taula Books. No serveixen i recorrem la meitat de la taula que té 5000 blocs.
- Amb B+, cluster o hash per name, s'apliquen les fórmules corresponents.
- Amb estructura cluster. Hem de recórrer la meitat de l'estructura que, com ja hem dit, ocupa 22500 blocs.
- Cost Q3:
 - Amb Hash Join:
 - Amb B+ o Hash. Aquests índexs no afecten el volum de la taula i, com ja hem vist, el cost és 45000
 - Amb cluster a Book. La primera lectura d'aquesta taula s'incrementa pels espais buits, però no l'escriptura de la taula reorganitzada ni la segona lectura. I la taula Author no es veu afectada. Cost: $15000 + 10000 + 10000 + 5000 + 5000 + 5000 = 50000$.
 - Amb cluster a Author. La primera lectura d'aquesta taula s'incrementa pels espais buits, però no l'escriptura de la taula reorganitzada ni la segona lectura. I la taula Book no es veu afectada. Cost: $10000 + 10000 + 10000 + 7500 + 5000 + 5000 = 47500$.
 - Amb Sort Match:
 - Si no hi ha cap cluster, s'ha de fer com ja hem vist si no hi ha cap index, amb un cost de 75000.
 - Si hi ha el cluster Book(theme), la taula Book s'ha d'ordenar igualment per author, però amb un cost major perquè hi ha els espais buits i ocupa 15000 blocs, no 10000. La primera lectura de l'ordenació tindrà aquest cost extra de 5000, però no les posteriors escriptures i lectures. Així, el cost del Sort Math és de 45000 per ordenar Book, 20000 per ordenar Author i 15000 de recórrer en paral.lel les taules ordenades.
 - Si hi ha cluster Book(author), s'ha d'ordenar Author (20000) i recórrer en paral.le Author ordenat (5000) i Book en cluster (15000).
 - Si hi ha cluster Author(name), s'ha d'ordenar Book (40000) i recórrer en paral.le Book ordenat (10000) i Author en cluster (7500).

El primer que podem veure és que l'estructura cluster no hi cap. Necessitaríem els 15000 blocs de les taules i 7500 blocs extra, però només en tenim 22000.

Dels candidats que sí que hi caben, el que ens dona cost millor és el cluster Books(theme) i el seleccionem. Això invalida les altres opcions sobre la taula Books (no pot tenir un altre cluster i ja té un B+ per theme com a part del cluster). Si actualitzem la taula de costos ens queda com veieu a la diapositiva 22. Ens adonem que el cluster Author(name) ja no hi cap i de les dues alternatives que queden triem la millor.