OPTIMITZACIÓ FÍSICA

Selecció, Ordenació, Projecció

Aquesta sessió continua l'anterior completant el procés d'optimització física amb les operacions de selecció, ordenació i projecció. Feu-hi un repàs si ho creieu convenient abans de començar.

Selecció. Tot i que ja coneixem com es poden resoldre (i amb quin cost) seleccions simples (que involucren un sola condició que es pot resoldre amb algun índex o que cal fer recorrent una taula) ens falta considerar com resoldre una selecció que combina diverses condicions (amb AND/OR) en un context en què podem tenir índexs per resoldre algunes de les condicions simples combinades.

A la diapositiva 4 hi teniu l'algoritme que segueixen els SGBD per decidir quins índexs es fan servir i de quina manera es combinen els resultats obtinguts de cada índex per trobar les files que compleixen una condició complexa:

1. Posar la condició en Forma Normal Conjuntiva. Podeu trobar a internet fàcilment més detalls sobre la definició de FNC i propietats de la lògica per transformar una condició en una d'equivalent en FNC. Per exemple, a l'apartat 2.2 de la sessió 5 de la guia d'estudi http://www.salleurl.edu/semipresencial/ebooks/ebooks/ebook_logica_matematica.pdf
   Una condició en FNC, doncs, té la forma
   $F_1 \wedge F_2 \wedge ...F_n$ on cada F_i té la forma $l_1 \vee l_2 \vee ...l_m$
   De les *F* en diem clàusules i dels *l* literals, que en el nostre cas són condicions simples com ara A=V, A>V, A≠V, etc. essent *A* un atribut de la taula i *V* un valor.

2. Per a cada clàusula es decideix si es pot resoldre amb els índexs disponibles. Això serà possible si hi ha un índex per a cadascun dels literals de la clàusula. En cas afirmatiu, el SGBD obtindrà, de cada índex, la llista de RID (llista d'adreces de fila) de les files que compleixen el literal corresponent. La unió d'aquestes llistes correspon a la llista de RID de les files que compleixen la clàusula.

3. De les llistes de RID que hem obtingut en el pas anterior (una per a cada clàusula que es podia resoldre amb els índexs), en fem la intersecció per obtenir la llista de RID de les files que compleixen la conjunció de les clàusules corresponents.

4. S'accedeix ara a les files adreçades per la la llista obtinguda en el pas anterior i, per a cadascuna, es comprova si també compleix les clàusules que no s'han resolt a través dels índexs.

Les diferents versions de la diapositiva 6 il·lustren aquest procés amb un exemple. Abans, però, la diapositiva 5 remarca algunes coses relatives a l'algoritme de selecció:

- Per resoldre una clàusula amb índexs cal que hi hagi un índex per a cada literal de la clàusula, només que en falti un ja no en farem servir cap per a aquella clàusula.

- Lògicament, si no es pot resoldre cap clàusula a través dels índexs, la selecció es resoldrà recorrent la taula i sense fer servir cap índex.

- Les condicions simples que es poden resoldre amb els diferents tipus d'índex són aquestes:

	Arbre	Hash
$A = V$	Sí	Sí
$A \neq V$	No	No
$A > V$, $A < V$, $A \geq V$, $A \leq V$	Sí	No

Vegem amb un exemple com funciona aquest procés. Imaginem que tenim una taula

Vehicles				
origen	marca	model	any	@
Sildavia	ACME	Claudio	2004	A1
Bordúria	ACME	Coyote	2003	A2
Itaca	Ponte	RRunner	2002	A3
Bordúria	ACME	Coyote	1995	A4
Bordúria	Ponte	RRunner	1990	A5
Itaca	PKT	BBunny	1991	A6

amb un índex B+ per cadascun dels atributs i que hem d'executar la consulta

```
SELECT * FROM Vehicles WHERE
      (marca = 'ACME' OR any <> 1990) AND
      origen = 'Bordúria' AND
      (model = 'Claudio' OR any < 2000)
```

La condició ja està en FNC i tenim una primera clàusula (`marca = 'ACME' OR any <> 1990`) que no es pot resoldre amb els índexs degut a la condició "`any <> 1990`" i dues clàusules més que sí que es poden fer amb els índexs. L'execució farà aquests passos:

- S'accedeix al B+ per origen amb "`origen = 'Bordúria'`" i s'obté la llista d'adreces {A2, A4, A5}, que és el resultat de la segona clàusula.

- S'accedeix al B+ per model amb "`model = 'Claudio'`" i obtenim {A1} i al B+ per any amb "`any < 2000`" i obtenim {A4, A5, A6}. Fem la unió d'aquestes llistes i tenim {A1, A4, A5, A6} com a resultat de la tercera clàusula.

- Fem la intersecció de la llista obtinguda per a les clàusules segona i tercera i tenim {A4, A5} com a resultat de la conjunció d'aquestes clàusules, o sigui `origen = 'Bordúria' AND (model = 'Claudio' OR any < 2000)`

- Finalment, accedim a les files amb adreça A4 i A5 i comprovem si compleixen "`(marca = 'ACME' OR any <> 1990)`" i només ho fa la A4, que és el resultat de la selecció.

Ordenació. Quan necessitem obtenir les files d'una taula ordenades segons el valor d'algun atribut, pot ser que disposem de la taula indexada per aquell atribut o que no. Les opcions que se'ns presenten a la diapositiva 7 són:

- No hi ha índex. Caldrà fer servir algun algoritme d'ordenació.

- Hi ha un índex B+. Com sabem, les fulles de l'arbre estan ordenades, de manera que podem recórrer les fulles i anar accedint a la taula amb les adreces que anem trobant. Farem els accessos a disc necessaris per a la lectura de les fulles i, per a cada entrada de l'índex (que, segons vam decidir en el seu moment - a la sessió d'estructures d'accés -, equival al nombre de files de la taula) un accés a la taula. En funció de diverses variables, sobretot del factor de bloqueig (nombre de files per bloc) sortirà més a compte el procés que acabem d'explicar (si hi ha poques tuples per bloc) o fer servir un algoritme d'ordenació.

- Hi ha un índex cluster. Això vol dir que la taula està ordenada i només cal recórrer-la.

- Hi ha un índex hash. En aquest cas l'índex no ens ajuda i haurem de fer servir un algoritme d'ordenació.

L'algoritme d'ordenació que es fa servir habitualment és el merge sort, que fa un total de $2B\lceil Log_M B\rceil - B$ accessos a disc per ordenar una taula de B blocs fent servir M+1 pàgines de memòria. Les diapositives 8, 9 i 10 el mostren de forma esquemàtica.

Si us hi fixeu, a cada crida recursiva la mida de les taules a ordenar es divideix per M i el cas trivial es dóna quan la mida ja és menor o igual que M. D'aquí ve el factor logarítmic (base M) del cost. Podeu observar també com la fórmula de cost es correspon amb unes quantes suposicions de la mida de la taula a ordenar respecte de M:

- Si B <= M, la taula cap a memòria i només s'ha de llegir, ordenar i escriure. Es fan, doncs, B accessos de lectura i B d'escriptura, amb un total de 1* 2B.

- Si B > M, es faran M crides recursives, cadascuna amb una mida de B/M. Si $B<=M^2$, B/M<=M, per tant les crides recursives cauen en el cas anterior i faran 1*2(B/M) accessos cadascuna; en total, doncs, M*2(B/M) = 2B. Aquí hi hem d'afegir els accessos que fa la fusió, que seran $2(B_1 + ... + B_M) = 2B$. El nombre total d'accessos és, doncs, 2B + 2B = 2 * 2B.

- Si $B > M^2$, es faran M crides recursives, cadascuna amb una mida de B/M. Si $B<=M^3$, $B/M<=M^2$, per tant les crides recursives cauen en el cas anterior i faran 2*2(B/M) accessos cadascuna; en total, doncs, M*2*2(B/M) = 2*2B. Aquí hi hem d'afegir els accessos que fa la fusió, que seran $2(B_1 + ... + B_M) = 2B$. El nombre total d'accessos és, doncs, 2*2B + 2B = 3 * 2B.

- Si $B > M^3$, es faran M crides recursives, cadascuna amb una mida de B/M. Si $B<=M^4$, $B/M<=M^3$, per tant les crides recursives cauen en el cas anterior i faran 3*2(B/M) accessos cadascuna; en total, doncs, M*3*2(B/M) = 3*2B. Aquí hi hem d'afegir els accessos que fa la fusió, que seran $2(B_1 + ... + B_M) = 2B$. El nombre total d'accessos és, doncs, 3*2B + 2B = 4 * 2B.

Les diferents versions de la diapositiva 11 mostren l'evolució de l'algoritme per un cas concret molt senzill, amb 4 blocs i M = 2. Finalment, a la diapositiva 12 s'enumeren les situacions en què es fa servir l'ordenació:

- Ens demanen explícitament un resultat ordenat.

- Eliminació de duplicats. Una manera fàcil d'eliminar duplicats consisteix en ordenar i a l'hora d'escriure el resultat de l'ordenació saltar els valors repetits, sabent que totes les repeticions d'un valor vindran seguides.

- Per la mateixa raó, l'ordenació ens permet agrupar totes les aparicions d'un mateix valor.

- Com vam veure a la sessió anterior, l'algoritme Sort-Match fa servir l'ordenació per obtenir, a través d'un recorregut en paral.lel de taules ordenades, la Join.

- De la mateixa manera es pot obtenir la diferència.

- Com veurem en una sessió posterior (Tècniques avançades d'indexació). l'ordenació pot intervenir en la inserció de moltes files de cop en una taula indexada.

Projecció. La darrera operació de les que poden aparèixer en un arbre de procés que considerem és la projecció, que consisteix a seleccionar totes les files però conservant només un subconjunt dels atributs. Aquest procés de simple supressió d'atributs no té cap complicació però pot generar duplicats que pot ser que vulguem eliminar. En aquest cas, tenim les mateixes opcions que hem comentat a l'inici de l'apartat d'ordenació segons l'existència d'un índex definit sobre els atributs a projectar:

- Si no tenim cap índex, procedirem a ordenar i obtenir els valors repetits seguits, com hem explicat una mica més amunt.

- Hi ha un índex B+ definit. Com sabem, les fulles de l'arbre estan ordenades, de manera que podem recórrer les fulles i descartar els duplicats, que estaran seguits. Cas que no tots els atributs a projectar estiguin recollits a l'índex, caldrà

també accedir a la taula. Farem, doncs, els accessos a disc necessaris per a la lectura de les fulles i, potser, un accés a la taula per a cada fila.

- Hi ha un índex cluster. Això vol dir que la taula està ordenada i només cal recórrer-la descartant les repeticions, que van seguides.
- Hi ha un índex hash. En aquest cas, recorrent els buckets no trobem els valors ordenats però sí agrupades totes les repeticions. Podem fer el mateix procés que amb l'arbre B+.