INDEXACIÓ AVANÇADA

Aquesta sessió complementa la que ja vam dedicar a estructures d'accés i hi tractarem diverses qüestions relacionades pròpies dels índexs: càrrega massiva d'índexs, consideracions generals sobre la utilitat (o no) dels índexs, índexs multiatribut, join-indexes, bitmaps i resolució de consultes amb ús exclusiu d'índexs.

Càrrega massiva. Quan ens trobem davant la necessitat d'inserir moltes entrades de cop a un índex (per exemple, quan creem un índex d'una taula que ja conté dades o quan hem d'afegir moltes files a una taula indexada) tenim dues alternatives: fer les insercions d'una en una o crear un índex nou de cop a partir de la taula ja actualitzada.

La diapositiva 5 enumera els passos que cal seguir per construir un índex a partir de la taula ja plena i la 6 ho il.lustra amb un exemple.

La diapositiva 7 compara les accions que cal fer per efectuar la inserció massiva d'una manera o del'altra:

- D'una en una: per cada fila, l'hem d'afegir a la taula (això també ho haurem de fer si optem per l'altra alternativa) hem de cercar a quina fulla li toca anar, modificar-la i escriure-la. Cas que la fulla ja estigués plena, s'haurà de fer el corresponent split.

- Inserció massiva: a l'esquerra de la diapositiva, es mostren els passos tal com s'ha enumerat abans. Oblidarem l'índex vell (si n'hi havia) - això no té cap cost -, afegirem les files a la taula - això també ho hem de fer si optem per l'altra alternativa -, llegirem totes les files per generar les entrades, escriurem les entrades, les ordenarem i construirem l'arbre. A la dreta de la diapositiva es descriu una petita millora, que aprofita que les entrades corresponents a les files que ja eren a la taula les podem obtenir llegint les fulles de l'índex vell. Això permet no haver de llegir la taula per generar les entrades.

Situacions apropiades (o no) a l'ús d'índexs. A la diapositiva 8 hi teniu algunes situacions en què típicament és una bona decisió la definició d'un índex:

- Un B+ si hem de resoldre una selecció amb factor de selectivitat petit. Si el FS no és petit (que vol dir que la condició és poc selectiva), pot resultar menys costós un recorregut de la taula que accedir a través del B+ perquè farem una lectura per cada fila que hem de recuperar.

- Un hash en les mateixes condicions i pel mateix raonament. A més, el hash només serveix per a condicions d'igualtat, degenera si es van fer insercions i supressions i requereix moltes entrades per tal que la distribució de la funció sigui uniforme.

- Un cluster quan el factor de selectivitat és gran perquè en aquest cas a cada lectura recuperem totes les files del bloc llegit. Això sí, també pot degenerar amb insercions i supressions atès que s'ha de mantenir l'ordre a la taula.

També podem identificar situacions en què, generalment, un índex no aporta millora en el cost. Les teniu a la diapositiva 9:

- El processament de la taula es fa sempre de forma massiva.
- La taula és petita (i, per tant, llegir-la sencera costa poc)
- L'atribut de cerca té pocs valors, la qual cosa implica factors de selectivitat grans.
- L'atribut de cerca apareix en una expressió (per exemple, preu * 0.85 < 100 no es pot resoldre amb un índex sobre preu). A no ser que el SGBD admeti índexs sobre expressions.

Per als casos de factor de selectivitat gran, veurem més endavant que els índexs bitmap sí que són apropiats.

Índexs multiatribut. En el seu moment vam presentar els índexs definits sobre un sol atribut, però també els podem definir sobre més d'un atribut. En el cas del hash, això no té grans implicacions i funciona aproximadament igual i permet trobar les files que compleixen una igualtat en cada atribut indexat. En el cas dels arbres cal analitzar amb més deteniment la funcionalitat que ofereixen com a conseqüència de la funció d'ordenació de les entrades d'índex: es compara el primer atribut i, en cas d'igualtat, el segon i així successivament. La diapositiva 10 explica tot això i la diapositiva 11 conté un exemple.

Join-indexes. Ens fixem ara en el cas particular de les consultes multidimensionals. A la diapositiva 12 hi teniu un exemple per refrescar la memòria sobre aquestes consultes on veiem que generalment s'han de seleccionar les files de les taules de dimensions (place, time, product) i després fer joins que combinen totes aquestes seleccions i la taula de fet (sales). Pel que sabem fins ara, haurem d'executar les seleccions i a continuació les joins.

La diapositiva 13 ens ensenya què és un join-index, que pot ser molt útil en consultes de l'estil que acabem de descriure. La particularitat d'aquests índex rau en què les entrades de l'índex - recordeu, les parelles (valor, adreça) o (valor, llista d'adreces) - tenen el valor d'un atribut d'una taula però l'adreça de les files d'una altra taula. Per exemple, l'entrada (Cat, <1, 4, ...>) ens diu que les files 1, 4, ..., de la taula sales fan join amb la fila de la taula place que té el valor Cat a region. Fixeu-vos que aquest índex ens permet resoldre consultes com ara

```
SELECT ... FROM PLACE JOIN SALES ON (...) WHERE region ='Cat'
```

cercant 'Cat' a l'índex i accedint a la taula sales amb les adreces trobades, sense necessitat de fer la join! A més, un join-index pot ser, també, multiatribut i podríem tenir un índex que ens permetés substituir totes les seleccions i joins d'una consulta multidimensional per una cerca en aquest índex

Sabem, però, que si volem tenir índexs multiatribut que puguin resoldre totes les condicions (region = 'Cat' AND year > 2000 però també year = 2010) haurem de tenir molts índexs amb moltes combinacions/permutacions d'atributs. A més, si les condicions són poc selectives (entre altres raons, perquè el nombre de valors diferents dels atributs de les taules de dimensions és petit - com sol passar - i el nombre de files de la taula de fets és molt gran - com també sol passar -), els arbres tendeixen a tenir poca alçària i la cerca tendeix a assemblar-se a una cerca en una llista que retorna moltes adreces. Aquests problemes són els que les diapositives 15 i 16 volen posar de manifest i els que els bitmaps mitiguen.

Bitmaps. Un índex bitmap està format per tantes columnes de bits com valors tingui l'atribut que volem indexar. Cada columna té tants bits com files té la taula indexada. El bit de la posició *i* de la columna del valor *v* és un 1 si la fila *i* de la taula té el valor *v* en l'atribut indexat i 0 en cas contrari. A la diapositiva 17 hi teniu un exemple amb dos bitmaps que ens diuen que la primera fila de la taula té el valor 'Ballpoint' en el primer atribut i el valor 'Catalunya' en el segon. En canvi, la tercera té els valors 'Pencil' i 'Andalusia' respectivament.

Els bitmaps ens permeten resoldre combinacions de igualtats amb ANDs i ORs llegint les columnes de bits corresponents i fent operacions de bits a memòria, com podeu veure a la diapositiva 18. Per facilitar aquest procés, els bitmaps es guarden a disc per columnes. A la diapositiva 19 podeu veure com es fan les insercions i supressions en un bitmap.

Si passem a considerar el cost d'accés a les dades a través de bitmap, hem de tenir en compte que el SGBD es comporta de manera diferent que quan fa servir els altres índexs. Ara cada bloc es llegirà un sol cop; és a dir, a partir de la columna resultat de consultar el(s) bitmap(s), el SGBD identifica tots els 1s que corresponen a files que estan al mateix bloc de la taula i les llegeix totes amb un sol accés. Per això el càlcul de costos ara no es basa directament en la probabilitat que una fila compleixi la condició sinó en la probabilitat que un bloc contingui alguna fila que compleix la condició. Aquesta probabilitat, que està en funció del factor de selectivitat de la condició i del nombre de files per bloc de la taula, la teniu a la diapositiva 20, on SF és el factor de selectivitat i R el nombre de tuples per bloc.

Basant-nos en aquesta probabilitat ja podem donar el cost del repertori d'accessos que vam analitzar en el seu moment per als altres índexs (diapositiva 21). En aquestes fórmules tingueu present que $\lceil |T|/bits \rceil$ (nombre de files de la taula, que equival al nombre de bits d'una columna, dividit pel nombre de bits que caben en un bloc) són els

blocs que ocupa una columna de bits. La fórmula general de cost d'una consulta (des de una simple igualtat a una condició complexa amb ANDs i ORs) és

$$v\lceil |T|/bits \rceil + B(1-(1-SF)^R)$$

on $v\lceil |T|/bits \rceil$ són els blocs que cal llegir per portar a memòria la columna de bits de cada valor que intervé en la condició i $B(1-(1-SF)^R)$ el nombre de blocs de la taula que caldrà llegir (nombre de blocs total multiplicat per la probabilitat que un bloc contingui alguna fila que compleix la condició).

La inserció d'una fila que no genera un nou valor al bitmap requereix llegir i escriure el darrer bloc (per afegir un 0 o un 1) de cada columna (i hi ha *ndist* columnes, una per cada valor de l'atribut indexat). Si genera un valor nou, a més d'afegir un 0 a cadascuna de les columnes que ja existien, s'ha de generar una nova columna. En tots dos casos, a més, s'ha de llegir i escriure el darrer bloc de la taula per afegir-hi la fila nova.

Resolució de consultes amb ús exclusiu d'índexs. Per acabar aquest tema teniu a la diapositiva 25 diverses tipologies de consultes que es poden executar accedint només a algun índex, sense necessitat de procedir després a recuperar files d'una taula amb adreces obtingudes de l'índex:

- Els exemples de projecció i agregats es veu sense dificultat que es poden resoldre accedint a les fulles de l'arbre o buckets del hash si la taula People està indexada per edat. Recordeu que les entrades a les fulles estan ordenades per edat i que totes les repeticions d'un valor d'edat estan en el mateix bucket de hash. Les fórmules ens indiquen el nombre de fulles de l'arbre i de buckets del hash, que solen ser força menors que el nombre de blocs de la taula.

- En l'exemple de join, la idea és que si People està indexada per id i Departaments ho està per boss, podem fer la join dels indexs (que, en ser més petits que les taules, serà més ràpida que la join de les taules) i després accedir a la taula people per obtenir els name i resoldre la consulta. Per fer la join entre índexs podem fer servir els mateixos algoritmes que vam veure per fer la join entre taules.

- Fixem-nos que, en el cas de la join, si la taula people estigués indexada amb un multiatribut (id, name) podríem resoldre la consulta sense accedir a la taula. I que consultes com ara

  ```
  SELECT name FROM people WHERE edat BETWEEN 20 AND 25
  ```

  també es poden resoldre només amb un índex multiatribut (edat, name).