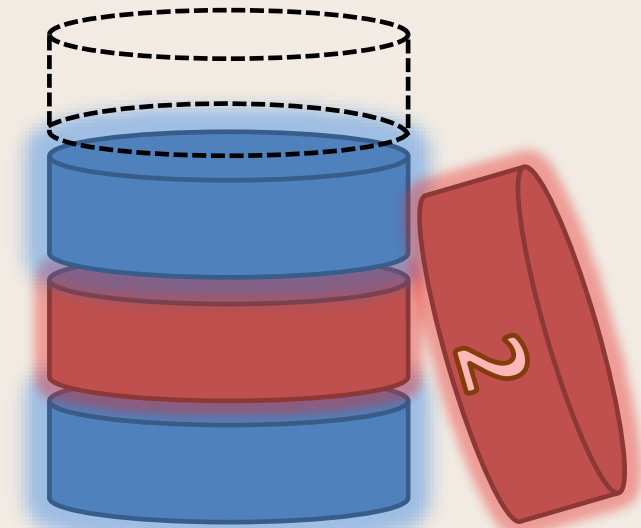


Evaluarea Operatorilor Relaționali 2



Condiții Join generale

- Egalități cu mai multe câmpuri
(ex., *R.sid=S.sid AND R.rname=S.sname*):
 - Pentru *Index NL*, putem construi un index compus *<sid, sname>* (dacă S e tabela internă); sau se pot utiliza doi indecși pe *sid* sau *sname*.
 - Pentru *Sort-Merge* și *Hash Join*, ordonarea/partiția se realizează pe combinația ambelor câmpuri.

Condiții Join generale

- Inegalități
(ex., *R.rname < S.sname*):
 - Pentru *Index NL*, e necesar un B-arbore (clusterizat!).
 - numărul de “potriviri” este de obicei mult mai mare decât în cazul egalităților.
 - *Hash Join*, *Sort Merge Join* nu sunt aplicabile.
 - *Block NL* este cea mai potrivită metodă în acest caz.

Statistici și cataloage

- *Catalogul* unei baze de date conține cel puțin următoarele informații despre tabele și indecși:
 - numărul de înregistrări (NTuples) și numărul de pagini (NPages) ale fiecărei tabele.
 - numărul de valori distincte ale cheilor de indexare (NKeys) și numărul de pagini (Npages) pentru fiecare index.
 - Înălțimea și valorile minime și maxime ale cheilor (Height /Low /High) pentru fiecare index cu structură de arbore.

Statistici și cataloage

- Cataloagele se actualizează periodic
 - Actualizarea la fiecare modificare e foarte costisitoare; dar fiind vorba (oricum) de aproximare acest lucru nu reprezintă un dezavantaj considerabil.
- Uneori se stochează informații mai detaliate (ex. histograme ale valorilor unui câmp)

Estimarea dimensiunii și factorii de reducere

- Fie interogarea:

```
SELECT attribute list  
FROM relation list  
WHERE term1 AND ... AND termk
```

- Numărul maxim de înregistrări din rezultat este produsul cardinalităților relațiilor din clauza FROM
- *Factorul de reducere (FR)* asociat fiecărei *term* reflectă impactul lui *term* în reducerea dimensiunii rezultatului. *Cardinalitatea rezultatului* = Nr maxim de înreg.* produsul tuturor FR.
 - Presupunem implicit că *termenii* sunt independenți!
 - *col=val* are FR: $1/NKeys(I)$, pentru indexul *I* pe *col*
 - *col₁=col₂* are FR: $1/MAX(NKeys(I_1), NKeys(I_2))$
 - *col>val* are FR: $(High(I) - val)/(High(I) - Low(I))$

Selecție simplă

```
SELECT *  
FROM   Students S  
WHERE  S.sname < 'C%'
```

- Are forma $\sigma_{R.c\acute{a}mp \text{ OP } val} (S)$
- Dimensiunea rezultatului aproximată de *dimensiunea lui S * factor de reducție*.
- Fără index, nesortat: trebuie scanată întreaga tabelă; costul este N (număr de pagini în S)
- Fără index, sortat : căutare binară pt. localizarea primei înregistrări ce satisface condiția **cost**= $\log_2 N$
- Cu un index pentru atributul de selecție:
Folosește indexul pentru determinarea înregistrărilor din rezultat, apoi returnează înregistrările corespunzătoare.

Utilizarea unui index pentru selecții

- Costul depinde de numărul de înregistrări returnate și de clusterizare.
 - Costul găsirii înregistrărilor (de obicei mic) plus costul returnării înregistrărilor (poate fi mare fără clusterizare).
 - În exemplu, presupunând distribuirea uniformă a numelor, aprox. 10% dintre înregistrări este returnat (50 pagini, 4000 înregistrări). Cu un index clusterizat, costul e mai mic de 50 I/Os; dacă e neclusterizat, costul e până la 4000 I/Os!

Utilizarea unui index pentru selecții

■ *Rafinare importantă a indecșilor ne-clusterizați:*

1. Găsirea înregistrărilor.
2. Sortarea acestora după *rid* (adresa/identificatorul fizic al înregistrărilor).
3. Se citesc *rid* în ordine. Se asigură că fiecare pagină de date este adusă în memoria internă o singură dată.

Condiții de selecție generale

(day<8/9/94 AND grade=10) OR cid=5 OR sid=3

- Fiecare condiție de selecție este prima dată convertită la forma normală conjunctivă (CNF):

*(day<8/9/94 OR cid=5 OR sid=3) AND
(grade=10 OR cid=5 OR sid=3)*

- Vom discuta doar cazul fără OR-uri.
- Un index se potrivește unei (conjuncții de) termeni dacă implică doar câmpuri dintr-un *prefix* al cheii de căutare.
 - Un index pe $\langle a, b, c \rangle$ se potrivește cu $a=5$ AND $b=3$, dar nu și $b=3$.

Abordări ale selecțiilor generale

1. Găsirea *cele mai selective căi de acces*, returnarea înregistrărilor folosind această cale și aplicarea tuturor termenilor ce nu au fost acoperiți de index:

- *Cea mai selectivă cale de acces*: parcurgerea unui index sau a unei tabele ce necesită cele mai puține citiri/salvări de pagini de memorie.
- Termenii care sunt acoperiți de index reduc numărul de înregistrări *returnate*; ceilalți termeni sunt folosiți pentru a invalida anumit înregistrări, dar nu afectează numărul de înregistrări/pagini citite.
- Exemplu *day<8/9/94 AND cid=5 AND sid=3*. Se poate utiliza un index B-arbore pe *day*; apoi, *cid=5* și *sid=3* trebuie verificate pentru fiecare înregistrare *returnată*. Similar, poate fi folosit un index pe *<cid, sid>*; trebuie apoi verificat *day<8/9/94*.

Abordări ale selecțiilor generale

2. (dacă sunt 2 sau mai mulți indecși):

- Se obține lista de *rid* ale înregistrărilor folosind fiecare index.
- Se *intersectează* listele de *rid*
- Pe înregistrările obținute se aplică toți termenii rămași.
- Fie *day*<8/9/94 AND *cid*=5 AND *sid*=3. Dacă e definit un index B arbore pe *day* și un alt index pe *sid*, se pot obține codurile *rid* ale înregistrărilor ce satisfac *day*<8/9/94 folosind primul index, și codurile *rid* ale înregistrărilor ce satisfac *sid*=3 folosind cel de-al doilea index. Aceste rezultate se vor intersecta și se va verifica și condiția *cid*=5.

Operatorul proiecție

■ Proiecția : $\pi_{\text{cid, sid}}$ Evaluations

```
SELECT  DISTINCT  
        E.sid, E.cid  
FROM    Evaluations E
```

■ Pentru implementarea proiecției

- Se elimină câmpurile nedorite
- Se elimină toate înregistrările duplicate

■ Abordări:

- Proiecție bazată pe sortare
- Proiecție bazată pe funcție de dispersie

Proiecție bazată pe sortare

- Pas 1 - Scanare E pentru a obține înregistrările având doar câmpurile dorite
 - $\text{Cost} = N$ I/O pentru scanare E (N = număr de pagini din E) + T I/Os pentru salvarea tabelii temporare E' (T = număr de pagini din E')
- Pas 2 - Sortează înregistrările folosind o combinație a câmpurilor ca și cheie de sortare
 - $\text{Cost} = O(T \log_2 T)$
- Pas 3 - Scanare rezultate sortate, se compară înregistrările adiacente și se elimină duplicările
 - $\text{Cost} = T$

Proiecție bazată pe sortare - îmbunătățire

- **Modificare pas 0 al sortării externe pentru a elimina câmpurile nedorite.** Se produc sub-șiruri inițiale sortate de lungime $2B$ pagini, având înregistrări de dimensiune mai mică decât înregistrările inițiale. (în funcție de numărul și dimensiunea câmpurilor eliminate)
- **Modificare pas de interclasare pentru a elimina duplicatele.** Numărul înregistrărilor rezultate este mai mic (Diferența depinde de numărul duplicatelor.)
- **Cost:** La pasul 0, se citește tabele inițială (dim. M), și este salvat temporar același număr de înregistrări de dimensiune mai mică. La pasul de interclasare rezultă mai puține înregistrări. Folosind *Evaluations*, cele 1000 pagini se reduc la 250 la pasul 0 dacă câmpurile rămase reprezintă 25 % din dimensiunea unei înregistrări

Proiecție bazată pe sortare - exemplu

- Proiecția tablei *Evaluations*
- Proiecția bazată pe sortare
 - Pas 1:
 - Scanează *Evaluations* cu 1000 I/Os
 - Dacă o înregistrare din E' e 10 octeți, se vor salva în tabela temporară E' 250 pagini
 - Pas 2:
 - Având 20 pagini în *buffer*, se sortează E' în doi pași la costul de $2*2*250$ I/O
 - Pas 3:
 - 250 I/O cost la scanarea de găsierea a duplicatelor
 - Cost total: 2500 I/O

Proiecție bazată pe sortare - exemplu

- Varianta îmbunătățită a proiecției tabeli *Evaluations*
 - Pas 1:
 - Scanare *Evaluations* cu 1000 I/O
 - Salvează E' cu 250 I/O
 - Având 20 pagini în *buffer*, 250 pagini sunt salvate ca 7 subșiruri sortate, fiecare având 40 pagini
 - Se folosește varianta optimizată a sortării externe,
 - Pas 2:
 - Se citesc subșirurile sortate (250 I/O) și se interclasează
 - Cost total: 1500 I/O

Proiecție bazată pe funcție de dispersie

- *Faza de partiționare*: Se citește tabela R folosind o singură pagină de input. Pentru fiecare înregistrare se elimină câmpurile nedorite și se aplică o funcție de dispersie $h1$ pentru a stoca înregistrarea într-unul dintre cele $B-1$ pagini rămase.
 - Rezultatul e format din $B-1$ partiții. Evident 2 înregistrări din 2 partiții diferite sunt distincte.
- *Faza de eliminare a duplicatelor*: Pentru fiecare partiție se aplică o funcție de dispersie $h2$, ($\neq h1$) pe toate câmpurile rămase, cu eliminarea duplicatelor.
 - Dacă partiția nu încapă în memorie se va aplica algoritmul de proiecție , recursiv.

Proiecție bazată pe funcție de dispersie - Cost

- Partiționare

- Citire $E = N \text{ I/O}$

- Salvare $E' = T \text{ I/O}$

- Eliminarea duplicatelor

- Citirea partițiilor = $T \text{ I/Os}$

- Cost total = $N + 2T \text{ I/Os}$

- Exemplu *Evaluations* = $1000 + 2*250 = 1500 \text{ I/Os}$

Discuție asupra proiecției

- Abordarea bazată pe sortare este standard; se aplică mai bine tabelelor cu dimensiune variabilă iar rezultatul este sortat.
- Dacă un index al relației conține toate câmpurile necesare în cheia de căutare, atunci tabela se poate scana folosind doar indexul(*index-only* scan)
 - Proiecția se aplică intrărilor indexului (dim. redusă!)
- Mai eficient este dacă un index al tabelului conține toate câmpurile necesare ca *prefix* al cheii de căutare:
 - Returnează intrările în ordine, renunțându-se la câmpurile nedorite și comparând înregistrările adiacente pentru determinare duplicărilor la o singură trecere.