Base de Données

Line POUVARET

2015-2016

Bilan sur "les performances"

- Card(R) =
  - nb de tuples de la relation
  - Soit e le nb de tuuples de R par page

$$NP(R) = \lceil \frac{Card(R)}{e} \rceil$$

- Card(I) =
  - nb valeur $\neq$ de l'attribut indexé
  - Soit d le nb de tuples de I par page (d $\gg$ e)

$$NP(I) = \lceil \frac{Card(I)}{d} \rceil$$

Exemple

Card(R) = 21
e = 5
NP(R) = 5

Remarque

Card(I) = Card(R)
si l'attribut indexé est clé de R.

	Balayage "select * from R"	Recherche sur égalité "X=..."	Plage de valeur X > '...'	Insertion	delete
Sequentiel	NP(R)	$\frac{NP(R)}{2}$ en moyenne	NP(R)	Cst	NP(R)
Hashé	NP(R)	1	NP(R)	Cst	NP(R)
Index ou B-arbre	NP(R)	$\log_d$ (Card(R))	$\log_d$ (Card(R)) + nb de pages où il y a des tuples vérifiant la propriété	Insertion dans le B^+ $\log_d$ (Card(R))	Insertion dans le B^+ $\log_d$ (Card(R))

→ Organisation ϕ des données est un paramètre important
→ Indexation :

- rapidité d'accès
- mise à jour coûteuse

$\rightarrow$ Grouper les mises à jour (en présence d'index) pour éviter de perturber les lectures.
$\rightarrow$ Compromis lectures/mises à jour.

Implantation des opérateurs d'accès aux données

$\rightarrow$ Opérations algébriques
$\rightarrow$ Opérateurs de tri (Order by/Group by)
$\rightarrow$ Agrégation (max, sum, count, min)

- Selection : σ_F(R)

  clause "where" du SQL

  Soit par accès séquentiel

  Soit par un index

- Projection : Π_y(R)

  clause "select" de SQL

  $\rightarrow$ Elimination des "doublons"

  - $\theta(n^2)$ méthode naïve
  - Utilisation d'une fonction de hashage $\rightarrow$ θ(n)

- Jointure naturelle : R $\bowtie$ S = S $\bowtie$ R

  - Pour une série de n jointures, on a (n-1)! séquence de jointures possibles
  - Comment choisir la bonne séquence ?

$\rightarrow$ Influence de stockage
$\rightarrow$ Influence des algos

Algo boucles imbriquées :

R $\bowtie$ S
(R.a = S.a)

T $\leftarrow \varnothing$ **pour** $r \in R$ **faire**
    pour $s \in S$ **faire**
        si $r.a = s.a$ **alors**
            ajoute(r,s) à T ;
        fin
    fin
fin

Remarque

- Le résultat est dans l'ordre de la relation externe
- Coût nb de comp $\theta(n^2)$
  $\rightarrow$ Coût en E/S
  Considérons une mémoire pouvant contenir b+2 pages de R
  Coût en nb d'E/S

$$NP(R) + \lceil \frac{NP(R)}{b} \rceil * NP(S)$$

Remarque

La jointure par boucles imbriquées n'est pas "symétrique".
$\rightarrow$ l'ordre du résultat $\rightarrow$ Coût E/S

Algo Tri-Fusion

Procéder en 2 étapes :

1. Trier R et S sur les attributs de jointure
2. Fusion des tables triées (évaluation de la condition de jointure)

Tri externe de R :

NP(R) * $\log_b$(NP(R))
où b est le nb de pages de R en mémoire.
Pour la jointure Tri fusion :

- Tri de R : NP(R) * $\log_b$(NP(R))
- Tri de S : NP(S) * $\log_b$(NP(S))
- Fusion : NP(R) + NP(S)

Jointure par Hash Code

(θ condition soit une "égalité")
R $\bowtie$ S
(R.a = S.a)

pour *chaque tuple* $r \in R$ **faire**
| ajouter r à Table(h(r)) ;
fin
pour *chaque tuple* $s \in S$ **faire**
| $E_r \leftarrow$ Table(h(s)) ;
| **pour** *chaque* $r \in E_r$ **faire**
| | Ajouter (r, s) à T ;
| **fin**
fin

Si les 2 tables sont déjà hashé :
NP(R) + NP(S)
S oui mais pas R
NP(R) + Card(S)

Estimation des coûts des $\neq$ chemins d'accès

- balayage séquentiel
- dichotomique (si triée)
- index

Pour une même requête, pour les différentes options d'évaluation (différents plans d'exécution), on estime un coût et on choisit l'option qui a le coût le plus faible.

En général, le coût est fonction de :

- Coût en E/S
  $\rightarrow$ 1
- Coût CPU
  $\rightarrow \frac{1}{100}$
- Coût de "com" si la BD est répartie
  $\rightarrow$ Coût CPU < Coût Com < Coût E/S

$\rightarrow$ Evaluer les cardinalités (nb de tuples) des relations initiales et intermédiaires

$\rightarrow$ On utilise des estimations basées sur des informations (statistique) que le SGBD maintient dans son catalogue.

CA = $\underbrace{\sigma * NbP}_{E/S} + \underbrace{\mu * NbT}_{calcul} + \underbrace{\nu * NbM}_{comm}$

Info "stat" que le SGBD maintient dans un "catalogue".

Informations statistiques

$\rightarrow$ Pour chaque relation R :

- Card(R) : la cardinalité de R est le nombre de tuples de la relation.
- NP(R) : le nombre de pages nécessaires pour contenir R
- FacP(R) = le facteur de mise en "blocs" de R. Le nb de tuples de R dans une page.

NP(R) = $\lceil \frac{Card(R)}{FacP(R)} \rceil$

$\rightarrow$ Pour chaque attribut A de la relation R :

- $Card_R$(A) = la cardinalité de A dans R. Le nombre de valeurs distinctes qui apparaissent pour A dans la relation R.
- Min_R(A), Max_R(A) $\rightarrow$ les valeurs Min et Max pour l'attribut A dans la relation R.

$\rightarrow$ Pour chaque index I

- NP(I) : le nombre de pages de l'index
- Prof(I) : le nombre de niveaux de I

Estimation des cardinalités

- Sélection σ_F(R)

  On note SF(F) le facteur de sélectivité de F :

  Card(σ_F(R)) = SF(F) * Card(R)

  SF(F) $\leq$ 1 et Card(σ_F(R)) $\leq$ Card(R)

  Pour un attribut A, on suppose que les valeurs de A sont équi-probables dans R et qu'il y a au moins une valeur qui satisfait la condition.

F	SF(F)	Card(σ_F(R))
A=x	$\frac{1}{Card_R(A)}$	1
	$\frac{1}{Card_R(A)}$	$\frac{1}{Card_R(A)} * Card(R)$
A > c	$\frac{Max_R(A)-C}{Max_R(A)-Min_R(A)}$	SF(F) * Card(R)
A < c	$\frac{c-Min_R(A)}{Max_R(A)-MinR(A)}$	–
c_1 < A c_2	$\frac{c_1-c_2}{Max_R(A)-Min_R(A)}$	–
$F_1 \wedge F_2$	$SF(F_1) * SF(F_2)$	–
$F_1 \vee F_2$	$SF(F_1) + SF(F_2) - SF(F_1) * SF(F_2)$	–
$\neg F$	1-SF(F)	–
$A_1 = A_2$	$\frac{1}{Max(Card(A_1),Card(A_2))}$	–

- Produit cartésien

  R x S

  Card(R x S) = Card(R) * Card(S)

- Jointure

  T = R $\bowtie$ S $A_1 = A_2$

  où $A_1 \in Att(R)$

  $A_2 \in Att(S)$

  On a Card(T) $\leq$ Card(R) * Card(S)

  - Si la condition de jointure porte sur une clé de l'une des relations (par exemple R) alors on a :
    Card(T) $\leq$ Card(S)
    (1 tuple de S peut joindre au max 1 tuple de R)
  - Dans le cas général, la cardinalité de la jointure est la cardinalité du produit cartésien suivi d'une sélection
    $Card(T) = \frac{1}{Max(Card(A_1),Card(A_2))} * Card(R)$

- Union ensembliste

  T = R $\cup$ S

  Card(T) $\neq$ Card(R) + Card(S)

  Max(Card(R), Card(S)) $\leq$ Card(T) $\leq$ Card(R) + Card(S)

- Intersection

  T = R $\cap$ S

  0 $\leq$ Card(T) $\leq$ Min(Card(R), Card(S))

- Différence

  T = R - S

  0 $\leq$ Card(T) $\leq$ Card(R)

Estimation des coûts

On se limite au coût d'accès (nb de pages)

- Sélection $\sigma_F(R)$
  → Recherche linéaire : NP(R) (Full scan)

  Remarque : Si F est du type A = x où A est une clé alors : $\frac{NR(R)}{2}$ coût moyen

  → Si F est de type (A = x) où A est une clé de hachage CA = 1 (Coût d'accès)
  → Egalité avec Index multiniveau
  (A = x)
  Prof(I) + 1 (Si A est une clé de R)
  Prof(I) + SF(F) * Card(R)

  → Inégalité avec Index :
  Prof(I) + $\frac{NP(R)}{2}$ (en moyenne, si F porte sur la clé)
  Prof(I) + SF(F) * Card(R)

  Exemple :

  Card(R) = 3000
  FacP(R) = 30
  → NP(R) = 100

  I est un index sur A $\in$ Att(R)
  Prof(I) = 2
  $Card_R(A) = 500$
  $Min_R(A) = 10000$
  $Max_R(A) = 50000$

  R' = $\sigma_{A=20000}(R)$
  Card(R') = $\underbrace{SF(A = 20000)}_{\frac{1}{500}} * \underbrace{Card(R)}_{3000} = 6$

  - Recherche linéaire : Ca = NP(R) = 100
  - Index : CA = Prof(I) + SF(F) * Card(R) = 2 + 6 = 8

  R" : $\sigma_{A>20000}(R)$
  Card(R") = SF(F) * Card(R)
  Card(R") = $\frac{Max_R(A)-20000}{Max_R(A)-Min_R(A)} * Card(R)$
  Card(R") = $\frac{50000-20000}{50000-10000} * 3000$
  Card(R") = 2250

  - Recherche linéaire : CA = NP(R) = 100
  - Index : CA = Prof(I) + SF(F) * Card(R) = 2 + 2250
    CA = 2252

SGBD (Oracle)

2 modes d'optimisation :

- orienté règles (RULE)
- orienté coût : Paramètre OPTIMIZER_MODE (CHOOSE)
  DBMS_STATS (PL/SQL)
  L'administrateur choisit à quelle fréquence les statistiques doivent être calculées.

→ EXPLAIN_PLAN : permet de connaître le plan d'exécution d'une requête
→ "hints" :

```sql
SELECT /* + Index(genreIndex) */ nom, prenom
FROM Etudiant
WHERE genre='M';
```