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

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

Implantation des opérateurs d'accès aux données

→ Opérations algébriques
→ Opérateurs de tri (Order by/Group by)
→ 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

  → Elimination des "doublons"

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

- Jointure naturelle : R ⋈ S = S ⋈ R

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

→ Influence de stockage
→ Influence des algos

Algo boucles imbriquées :

R ⋈ S
(R.a = S.a)

```
T ← ∅ pour r ∈ R faire
|   pour s ∈ 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 $\theta(n^2)$
  → 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".
→ l'ordre du résultat → 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.