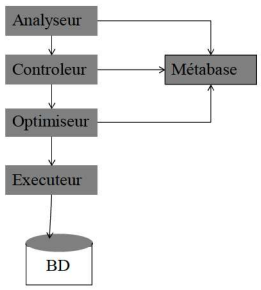
Evaluation et optimisation de requêtes relationnelles pour des SGBD mono-processeur

1. Principe d'évaluation d'une requête
2. Introduction à l'optimisation
3. Opérateurs physiques
4. Optimisation logique
5. Optimisation physique

Franck Morvan, Toulouse III, Laboratoire IRIT

Architecture fonctionnelle d'un SGBD



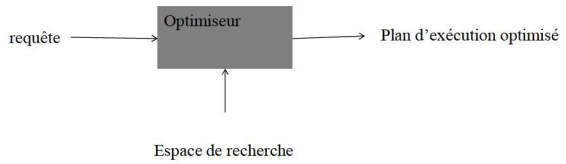
- Analyseur :
  - Analyse syntaxique : conformité à la grammaire
  - Analyse sémantique : conformité à la vue référencée ou au schéma
  - Traduction en format interne : noms remplacés par des références internes
- Controleur :
  - Vues remplacées par une ou plusieurs requêtes
  - Contrôle des droits d'accès
  - Contrôle de l'intégrité lors de MAJ

- Optimiseur :
  - Elaboration un plan d'accès optimisé
    - Décomposition en opérations d'accès élémentaire
    - Ordonnancement optimal ou proche de l'optimal des opérations
- Executeur :
  - Execution du plan
    - Gestion de la concurrence d'accès
    - Atomicité des transactions

4

Introduction à l'optimisation de requêtes relationnelles

- Position du problème



5

- q ∈ Q, p ∈ {Plans d'exécution} , $Coût_p$ (q) :
  - Trouver p calculant q tel que $Coût_p$ (q) est minimal
  - Objectif : trouver le meilleur compromis entre Min(temps de réponse) et Min(coût optimisation)
- Optimiseur <Strat., Espace, MC>
  - Strat. : stratégie de recherche
  - Espace : Espace de recherche
  - MC : Modèle de coûts

6

Opérateurs physiques : fichier & SGF

- Article
  - Enregistrement etu {nom, prénom, âge, @}
- Manipulation d'un fichier
  1. Type d'organisation : séquentiel, séquentiel indexé, organisation aléatoire relative
  2. Opérations : création, lecture, écriture

7

Opérations sur les fichiers (la syntaxe)

(1) Assignation : **assignation** (<nom_variable_logique>, <fichier_physique>, organisation [, attribut])
Si l'organisation est séquentielle indexée, on doit indiquer le nom d'attribut sur lequel le fichier est indexé.

(2) Ouverture : **ouverture** (<nom_variable_logique>, <lecture/ecriture>, <mode_accès>)
Mode_accès peut être séquentiel ou direct (utilisant l'index).

(3) Lecture : **lecture** (<nom_variable_logique>, <variable_article> [, clé])
Si l'accès est direct, on doit indiquer la valeur de la clé.

(4) Fin de fichier : **fdf** (<nom_variable_logique>)
La fonction retourne un booléen : vrai pour fin de fichier.

(5) Fermeture : **fermeture** (<nom_variable_logique>)

8

Fichier sequentiel

```
Début
        enregistrement etu {   nom : chaîne(20),
                               prenom : chaîne(20),  age entier,
                                   adresse : chaîne(20)} v ;
        assignation (fe, « ...etu.txt », séquentielle) ;
        ouverture (fe, lecture, séquentiel) ;
        Tant que non fdf(fe) faire
                Début
                lecture (fe, e) ;
                Si e.nom= « Dupont » alors afficher (e.adresse) ;
                Fin
        fermeture (fv) ;
Fin
```

9

Fichier sequentiel indexé

```
Début
        enregistrement etu {    nom : chaîne(20),
                                prenom : chaîne(20),    age entier,
                                adresse : chaîne(20)} v ;
        assignation (fe, « ...etu.txt », direct, nom) ;
        ouverture (fe, lecture, direct) ;
        lecture (fe, e, « Dupont ») ;
        afficher (e.adresse) ;
        fermeture (fv) ;
Fin
```

10

Fichier à adressage relatif

```
Début
        enregistrement etu {   nom : chaîne(20),
                               prenom : chaîne(20),  age entier,
                               adresse : chaîne(20)} v ;
        assignation (fe, « ...etu.txt », relatif) ;
        ouverture (fe, lecture, direct) ;
        lecture (fe, e, 123) ;
        afficher (e.adresse) ;
        fermeture (fv) ;
Fin
```

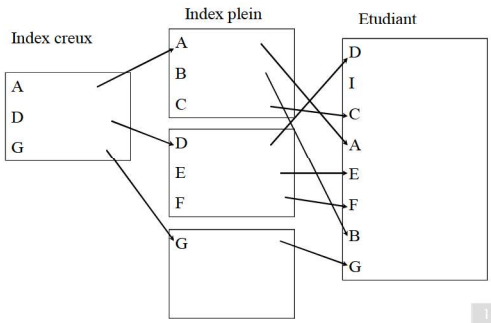
11

Les opérateurs physiques

- Algorithme de sélection
  - Sequentiel
  - Indéxé
- Algorithme de jointure

12

Index en B-arbre

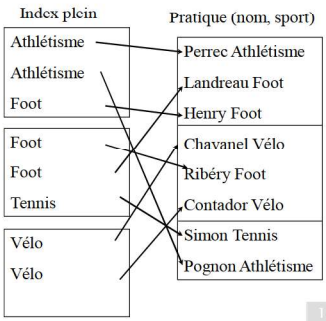


13

Index bitmap

1ière idée d'optimisation

Trier (Sport, @)



14

Index bitmap

rang	Athlé	Foot	Tennis	Vélo
0	1	0	0	0
1	0	1	0	0
2	0	1	0	0
3	0	0	0	1
4	0	1	0	0
5	0	0	0	1
6	0	0	1	0
7	1	0	0	0

Pratique (nom, sport)

Perrec Athlétisme	0
Landreau Foot	
Henry Foot	
Chavanel Vélo	1
Ribéry Foot	
Contador Vélo	
Simon Tennis	2
Pognon Athlétisme	

Rang div p → page

Rang modulo P → déplacement dans la page

15

Algorithme de jointure

- Jointure par produit cartésien
- Jointure par produit cartésien par bloc
- Jointure par produit cartésien indexé
- Jointure par hachage
- Jointure par tri-fusion

16

Jointure par Produit Cartésien

- Procédure JPC (R,S,T : relation; cond : conditionJointure)
  - Pour chaque tuple r ϵ R faire
    - Pour chaque tuple s ϵ S faire
      - Si cond alors écrire résultat dans T

17

Jointure par produit Cartésien par bloc

- b pages en mémoire
- Objectif : organiser les pages pour réduire les E/S
  - b-2 pages pour R
  - 1 page pour S
  - 1 page pour T

18

Jointure par produit Cartésien par bloc

- Procédure JPC_bloc (R,S,T : relation; cond : conditionJointure)
- Tq nonFin (R) faire
  - Lire (b−2) pages de R;
  - Tq nonFin (S) faire
    - Lire une page de S;
    - Pour chaque tuple de r ∈ R en mémoire faire
      - Si cond alors écrire
        - Écrire résultat dans tamponT
        - Si tamponT est plein alors écrire tamponT dans T
- écrire tamponT dans T

19

Jointure par produit Cartésien indéxé

- Procédure JPC_indexé (R,S,T : relation; cond : conditionJointure)
- Tq nonFin (R) faire
  - Lire 1 page de R;
  - Pour chaque tuple de r ∈ R en mémoire faire
    - chercherEntreeIndex (r.a, tableIndex, ptrIndex)
    - Tq nonFin (tableIndex) et ptrIndex.a = r.a faire
      - Charger la page corresponsant ptrIndex.ref;
      - Écrire résultat dans tamponT;
      - Si tamponT est plein alors écrire tamponT dans T;
      - Suivant (ptrIndex);
- écrire tamponT dans T

20

Jointure par hachage simple

- E1 : construire une table de hachage avec la plus petite des relations
- E2 : sonder la table de hachage

- La construction et le sondage se font avec une fonction de hachage appliquée sur l'attribut de jointure

21

Jointure par tri-fusion

- Trier R sur l'attribut de jointure;
- Trier S sur l'attribut de jointure;
- Utiliser la propriété de tri pour fusionner R et S

22

Optimisation Logique

- Objectif : diminuer le volume de données manipulées
- Règles de transformation
  - Éclatement des sélections (pour préparer la descente)
  - Regroupement des projections
  - Inversion projection sélection
  - Inversion sélection jointure (union, différence, …)
  - Inversion projection jointure (union, différence, …)

23

Optimisation physique

- Objectif :
- Stratégie de recherche
  - Énumérative :
    - Enumère l'ensemble des alternatives pour une requête
    - Peut entrainer des pbs de gestion d'un espace de recherche trop important
  - Aléatoire :
    - Applique un ensemble de transformation aléatoire à un plan d'exécution donnée en paramètre
    - Réduit l'espace de recherche
    - Meilleure solution pas forcément atteinte

24

Espace de recherche

Nombre de relations	Espace linéaire			Espace ramifié		
	Chaine	étoile	clique	Chaine	étoile	clique
3	4	4	6	8	8	12
5	16	48	120	224	384	1 680
7	64	1 440	5 040	8 448	46 080	665 280
10	512	725 760	3 628 800	2 489 344	185 794 560	17 643 225 600

25

Algorithme générique

- Arbre = initialiser();
- Tq non conditionArret() faire
  - Courant=selectionner(Arbre);
  - Arbre=Arbre-courant;
  - si courant.arbreComplet() alors res = res U courant;
  - Sinon
    - Succ = etendre(courant);
    - Succ=reduire(succ);
    - Arbre=Arbre U succ;
- Retourner resultatOptimal(res);

26

Desciption des primitives génériques

	Recherche en largeur d'abord	HA	HA*
Initialiser()	Toutes les relations de base	La plus petite des relations de base	Toutes les relations de base
conditionArret()	Arbre vide	Arbre vide	Arbre vide
Selectionner(Arbre)	Sélection du nœud le moins récent	Sélection du nœud le plus récent	Sélection du le plus récent s'il contient au moins 2 relations le plus petit sinon
reduire	Si plusieurs succ sont équivalent, le moins couteux est conservé	Le succ de plus petite cardinalité	Le succ de plus petite cardinalité

Optimisation physique

Select
From R1, R2, R3
Where R1.a=R2.b
And R2.c=R3.d;

Trace d'exécution de HA
Hypothèses R2 est la plus petite relation et
$|R2 \infty R1| < |R2 \infty R3|$

28

HA

Arbre = {R2}
Courant = R2; Arbre={}
Succ = {(R2 ∞ R1); (R2 ∞ R3)}
Arbre = {(R2 ∞ R1)}
Courant = (R2 ∞ R1); Arbre={}
Succ = {(R2 ∞ R1) ∞ R3}
Arbre = {(R2 ∞ R1) ∞ R3}
Courant = (R2 ∞ R1) ∞ R3; Arbre={}
Resultat = (R2 ∞ R1) ∞ R3

29