BDR 4I803 2016

Cours 3 : Traitement et optimisation de requêtes

1

Problème

EMP(ENO, ENAME, TITLE)
PROJECT(PNO, PNAME, BUDGET)
WORKS(ENO, PNO, RESP, DUR)

Soit la requête

pour chaque projet de budget > 250 qui emploie plus de 2 employés, donner le nom et le titre des employés

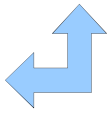
`Comment l'exprimer en SQL ?`

2

Un plan d'exécution possible

```
SELECT DISTINCT Ename, Title
       FROM   Emp, Project, Works
       WHERE  Budget > 250
       AND    Emp.Eno=Works.Eno
       AND    Project.Pno=Works.Pno
       AND    Project.Pno IN
              (SELECT Pno
               FROM   Works
               GROUP BY Pno
               HAVING COUNT(*) > 2)
```

T_1 ← *Lire* la table `Project` et *sélectionner* les tuples de `Budget` > 250
T_2 ← *Joindre* T_1 avec la relation `Works`
T_3 ← *Joindre* T_2 avec la relation `Emp`
T_4 ← *Grouper* les tuples de `Works` sur `Pno` et pour les groupes qui ont plus de 2 tuples, *projeter* sur `Pno`
T_5 ← *Joindre* T_3 avec T_4
T_6 ← *Projeter T5* sur `Ename`, `Title`
Résultat ← *Éliminer doublons* dans T_6



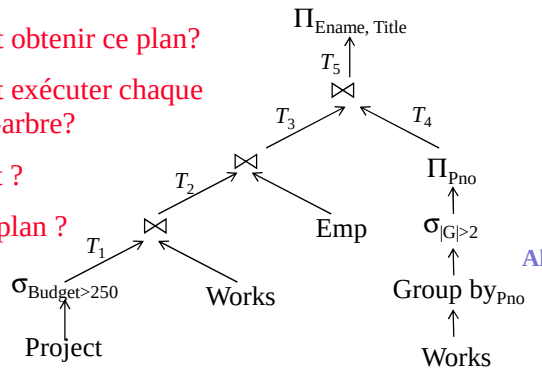
Représentation algébrique

1.Comment obtenir ce plan?

2.Comment exécuter chaque nœud/sous-arbre?

3.Quel coût ?

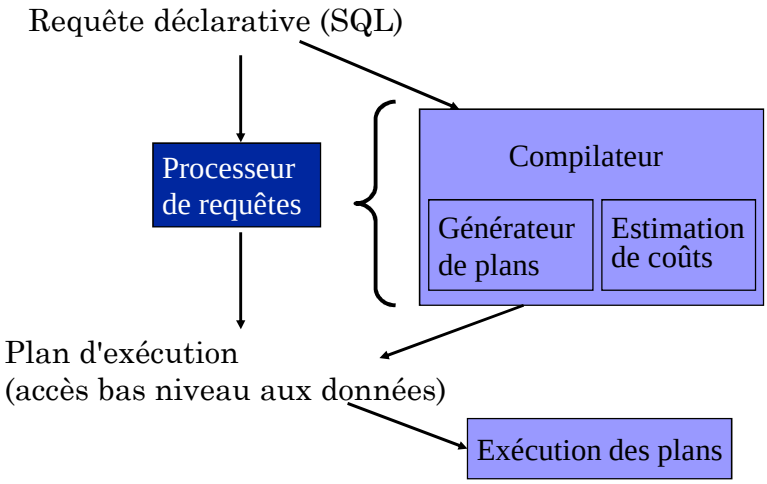
4.Meilleur plan ?



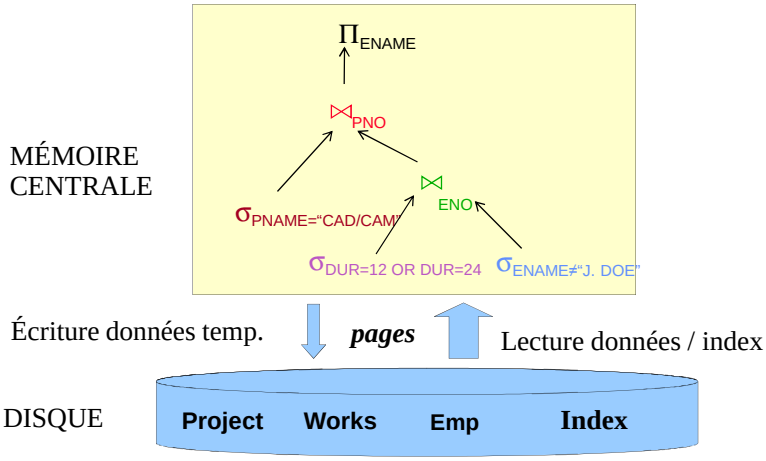
Algèbre étendue

$\Pi_{Ename, Title}(\Pi_{Pno}(\sigma_{|G|>2} Group_{Pno}(Works)) \bowtie (Emp \bowtie ((\sigma_{Budget>250000} Project) \bowtie Works)))$

•Traitement des requêtes

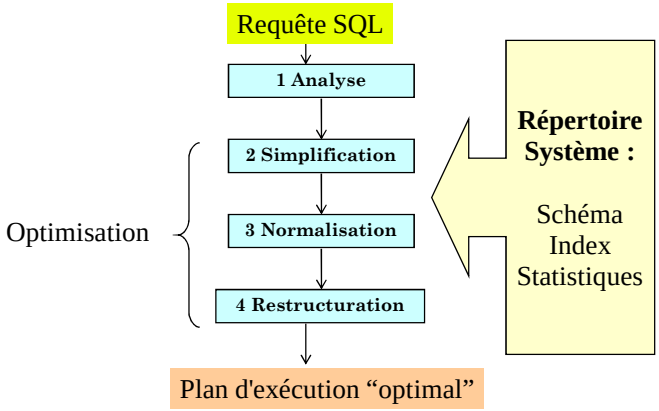


Exécution d'un plan



6

Étapes du traitement d'une requête



7

Normalisation de requête

- Analyse lexicale et syntaxique
  - vérification de la validité de la requête
  - vérification des attributs et relations
  - vérification du typage de la qualification
- Mise de la requête en forme normale
  - forme normale conjonctive
    $(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$
  - forme normale disjonctive
    $(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$
  - OR devient union
  - AND devient jointure ou sélection

8

Simplification

- Pourquoi simplifier?
  - plus une requête est simple, plus son exécution peut être efficace
- Comment? en appliquant des transformations
  - élimination de la redondance
    - règles d'idempotence

      $p_1 \wedge \neg(p_1) \equiv$ faux

      $p_1 \wedge (p_1 \vee p_2) \equiv p_1$

      $p_1 \vee$ faux $\equiv p_1$

      …
  - application de la transitivité (att1=att2 ,att2=att3)
- Éliminer des opérations redondantes : élagage
  - ex. : pas besoin de distinct après une projection sur une clé
- utilisation des règles d'intégrité
  - CI : att1 <100 Q: … where att1 > 1000…

9

Exemple de simplification

```
SELECT      Title
FROM        Emp
WHERE       Ename = 'J. Doe'          P1
OR            (NOT(Title = 'Programmer')   ¬P2
AND         (Title = 'Programmer'     P2
OR            Title = 'Elect. Eng.')  P3
AND         NOT(Title = 'Elect. Eng.'))   ¬P3
```

⇓ P1 ∨ (¬P2 ∧(P2 ∨ P3) ∧ ¬P3)

```
SELECT      Title
FROM        Emp
WHERE       Ename = 'J. Doe'
```

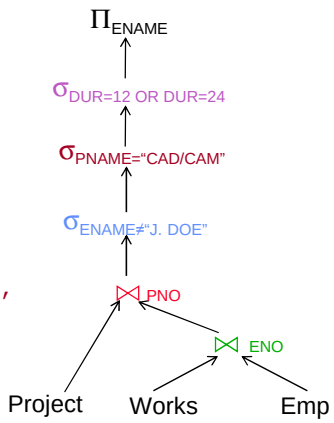
10

Traduction en algèbre

Conversion en arbre algébrique

Exemple :

```
SELECT   Ename
FROM     Emp, Works, Project
WHERE    Emp.Eno = Works.Eno
AND      Works.Pno = Project.Pno
AND      Emp.Ename <>'J.Doe'
AND      Project.name = 'CAD/CAM'
AND      (Works.Dur=12 OR
          Works.Dur=24)
```



11

Heuristiques

Observation : les opérations manipulant moins de données sont plus rapides (sélectivité corrélée à la performance)

Objectif : déterminer un ordre pour les opérations, censé être efficace.

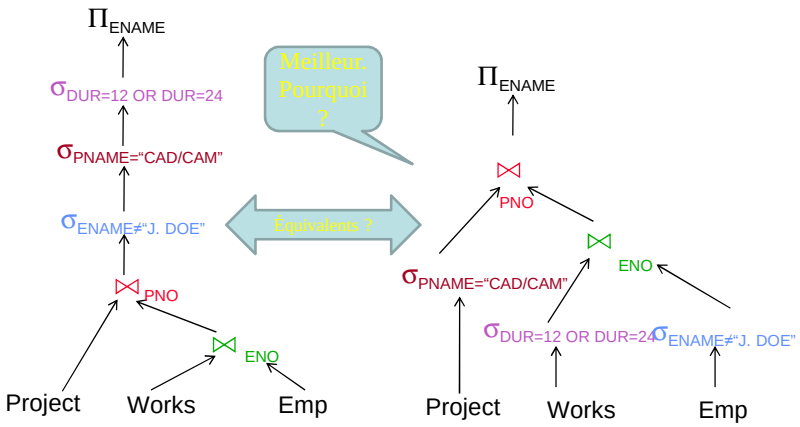
Méthode:

Commencer par traiter les opérations les plus sélectives (projection, sélection) et de manière à réduire la taille des données d'entrée pour les opérateurs suivants (jointures).

La place en mémoire est un facteur primordial pour l'efficacité d'une jointure (cf. dans 2 semaines)

12

Exemple



Optimisation basée sur le coût

- Elaborer des plans
  - arbre algébrique, restructuration, ordre d'évaluation
- Estimer leurs coûts
  - fonctions de coût
    - en terme de temps d'exécution
    - coût I/O + coût CPU
    - poids très différents
      - par ex. coût I/O = 1000 * coût CPU
- Choisir le meilleur plan
  - Espace de recherche : ensemble des expressions algébriques équivalentes pour une même requête
  - algorithmes de recherche:
    - parcourir l'espace de recherche
    - algorithmes d'optimisation combinatoire

14

Restructuration

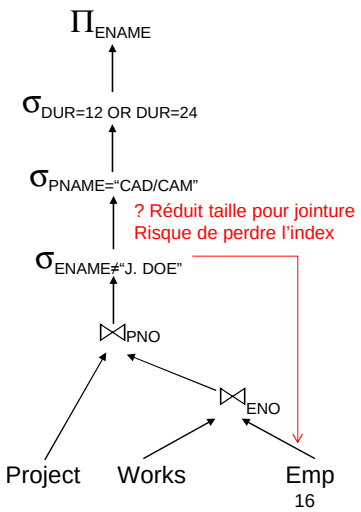
- Objectif : choisir l'ordre d'exécution des opérations algébriques (élaboration du plan logique).
- Conversion en arbre algébrique
- Transformation de l'arbre (optimisation)
  - règles de transformation (équivalence algébriques),
  - estimation du coût des opérations en fonction de la taille
  - Estimation du résultat intermédiaire (taille et ordre?)
  - En déduire l'ordre des jointures

15

Restructuration

- Conversion en arbre algébrique
- Exemple

```
SELECT  Ename
FROM  Emp, Works, Project
WHERE Emp.Eno=Works.Eno
AND Works.Pno=Project.Pno
AND   Ename NOT='J.Doe'
AND   Pname = 'CAD/CAM'
AND   (Dur=12 OR Dur=24)
```



16

Coût d'un plan

- Fonction de coût = estimation du temps écoulé pour
  - Accéder aux données :
    - temps I/O exprimé en nombre de pages lues ou écrites
  - Calculer le résultat d'une opération à partir des données lues
    - Temps CPU dépend du nombre d'instructions
- Estimation du coût d'exécution de chaque noeud de l'arbre algébrique
  - utilisation de pipelines ou de relations temporaires importante
    - Pipeline : les tuples sont passés directement à l'opérateur suivant.
      - Pas de relations intermédiaires (petites mémoires, ex. carte à puce).
      - Permet de paralléliser (BD réparties, parallèle)
      - Intéressant même pour cas simples : $\sigma_{F \wedge F'}(R)$, index sur F' $\rightarrow \sigma_F(\sigma_{F'}(R))$
    - Relation temporaire : permet de trier mais coût de l'écriture

17

Estimer la cardinalité d'une opération

- Estimation du nombre de nuplets résultant de chaque nœud par rapport à ses entrées
  - Permet d'estimer le coût de l'opération suivante
  - sélectivité des opérations – "facteur de réduction"
  - propagation d'erreur possible
  - basé sur les statistiques maintenues par le SGBD

18

Statistiques

- Relation
  - cardinalité : card(R)
  - taille d'un tuple : largeur(R)
  - fraction de tuples participant une jointure / attribut
  - …
- Attribut
  - cardinalité du domaine
  - nombre de valeurs distinctes distinct(A,R) = $\Pi_A(R)$
  - Valeur max, valeur min
- Hypothèses
  - indépendance entre différentes valeurs d'attributs
  - distribution uniforme des valeurs d'attribut dans leur domaine
  - Sinon, il faut maintenir des histogrammes
    - Equilarge : plages de valeurs de même taille
    - Equiprofond: plages de valeurs contenant le même nombre d'occurrence
    - Equiprofond meilleur pour les valeurs fréquentes (plus précis) voir diapo suivante
- Stockage :
  - Les statistiques sont des métadonnées, stockées sous forme relationnelle (cf. TME)
  - Rafraîchies périodiquement, pas à chaque fois.

Le fameux compromis L/E

Cardinalité des opérations

Sélection :

$taille(R) = card(R) * largeur(R)$

$card(\sigma_F(R)) = SF_\sigma(F) * card(R)$

où SF_σ est une *estimation* de la **sélectivité du prédicat** , dont la forme générale est "nombre d'éléments sélectionnés / nombre d'éléments possibles"

Si A continue. Sinon, remplacer par Card

$SF_\sigma(A = valeur) = \frac{1}{card(\prod_A(R))}$	
$SF_\sigma(A > valeur) = \frac{max(A) - valeur}{max(A) - min(A)}$	$SF_\sigma(A < valeur) = \frac{valeur - min(A)}{max(A) - min(A)}$
$SF_\sigma(p(A_i) \wedge p(A_j)) = SF_\sigma(p(A_i)) \bullet SF_\sigma(p(A_j))$	
$SF_\sigma(p(A_i) \vee p(A_j)) = SF_\sigma(p(A_i)) + SF_\sigma(p(A_j)) - (SF_\sigma(p(A_i)) \bullet SF_\sigma(p(A_j)))$	
$SF_\sigma(A \in \text{ens\_valeurs}) = SF_\sigma(A{=}valeur) \bullet card(\text{ens\_valeurs})$	

20

Cardinalité des opérations

Projection

$card(\Pi_A(R)) <= card(R)$ (égalité si A est *unique*)

Produit cartésien

$card(R \times S) = card(R) \cdot card(S)$

Union

borne sup. : $card(R \cup S) = card(R) + card(S)$

borne inf. : $card(R \cup S) = max\{card(R), card(S)\}$

Différence

borne sup. : $card(R–S) = card(R)$

borne inf. : 0

21

Cardinalité des opérations

Jointure :

- cas particulier: *A* est clé de *R* et *B est clé étrangère dans S vers R :*

$$card(R \bowtie_{A=B} S) = card(S)$$

- plus généralement

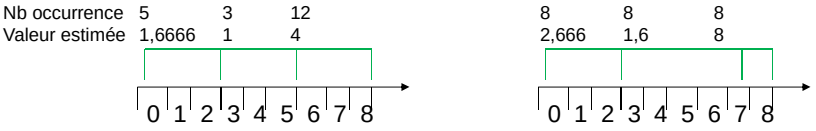
$$card(R \bowtie S) = SF_J \cdot card(R) \cdot card(S)$$

Comment l'obtenir ? Il faut des infos supplémentaire (SFj peut être stocké)

22

Histogramme

Select * from R where A = 8



Equilarge
Valeur estimée 4
Valeur réelle dans [0,12]
(faire l'hypothèse d'uniformité dans l'intervalle)

Equiprofond
Valeur estimée 8
Valeur réelle 8

Espace de recherche

- Caractérisé par les plans "équivalents" pour une même requête
  - ceux qui donnent le même résultat
  - générés en appliquant les règles de transformation vues précédement
- Le coût de chaque plan est en général différent
- L'ordre des jointures est important

Règles de transformation

- Commutativité des opérations binaires
  - $R \times S \equiv S \times R$
  - $R \bowtie S \equiv S \bowtie R$
  - $R \cup S \equiv S \cup R$
- Associativité des opérations binaires
  - $(R \times S) \times T \equiv R \times (S \times T)$
  - $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
- Idempotence des opérations unaires
  - $\Pi_{A'}(\Pi_{A''}(R)) \equiv \Pi_{A'}(R)$
  - $\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) \equiv \sigma_{p_1(A_1) \wedge p_2(A_2)}(R)$
    - où $R[A]$ et $A' \subseteq A$, $A'' \subseteq A$ et $A' \subseteq A''$

25

Règles de transformation

- Commutativité de la sélection et de la projection (si proj. des attr. sél.)
- Commutativité de la sélection avec les opérations binaires

$\sigma_{p(A)}(R \times S) \equiv (\sigma_{p(A)}(R)) \times S$

$\sigma_{p(A_i)}(R \bowtie_{(A_j,B_k)} S) \equiv (\sigma_{p(A_i)}(R)) \bowtie_{(A_j,B_k)} S$

$\sigma_{p(A_i)}(R \cup T) \equiv \sigma_{p(A_i)}(R) \cup \sigma_{p(A_i)}(T)$

où A_i appartient à R et T

- Commutativité de la projection avec les opérations binaires

$\Pi_C(R \times S) \equiv \Pi_{A'}(R) \times \Pi_{B'}(S)$

$\Pi_C(R \bowtie_{(A_j,B_k)} S) \equiv \Pi_{A'}(R) \bowtie_{(A_j,B_k)} \Pi_{B'}(S)$

$\Pi_C(R \cup S) \equiv \Pi_C(R) \cup \Pi_C(S)$

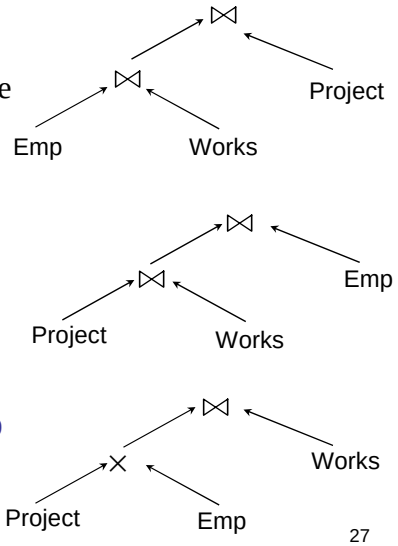
où $R[A]$ et $S[B]$; $C = A' \cup B'$ où $A' \subseteq A$, $B' \subseteq B$, $A_j \subseteq A'$, $B_k \subseteq B'$

26

Ordre des jointures

- Avec *N* relations, il y a O(*N*!) ordres de jointures équivalents qui peuvent être obtenus en appliquant les règles de *commutativité* et d'*associativité*

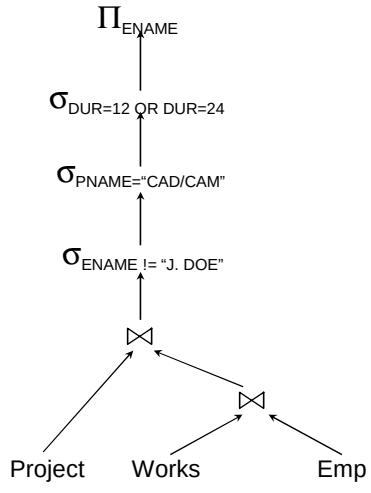
```
SELECT   Ename,Resp
FROM     Emp, Works,Project
WHERE    Emp.Eno=Works.Eno
AND      Works.PNO=Project.PNO
```



27

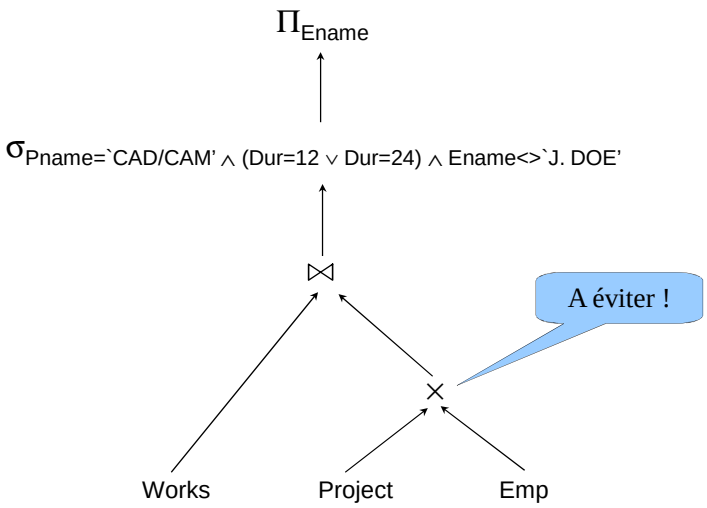
Exemple

```
SELECT Ename
FROM Project p, Works w,
     Emp e
WHERE w.Eno=e.Eno
AND  w.Pno=p.Pno
AND  Ename<>`J. Doe'
AND  p.Pname=`CAD/CAM'
AND  (Dur=12 OR Dur=24)
```



28

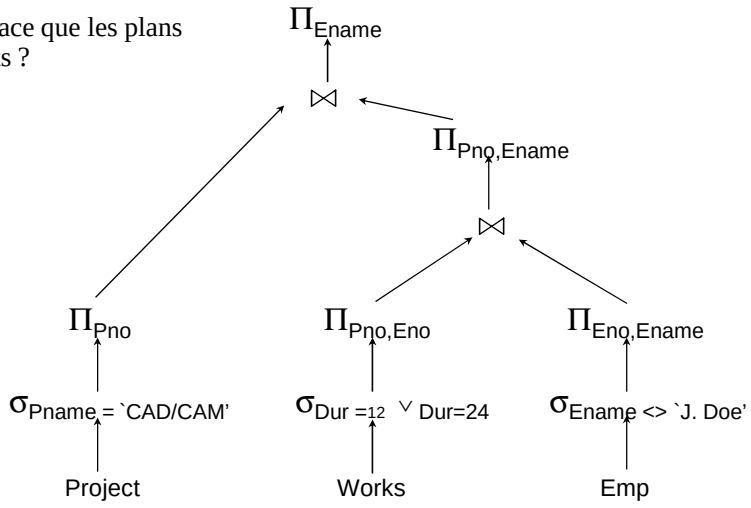
Plan équivalent



29

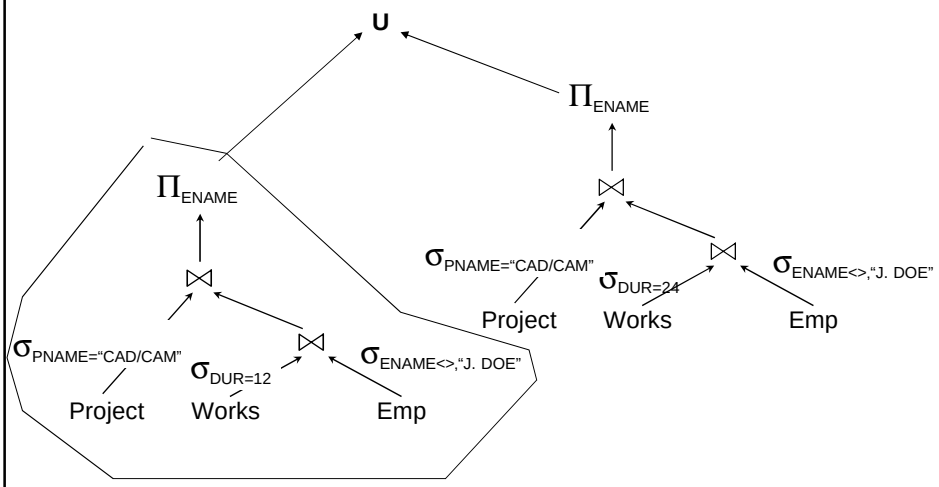
Autre plan équivalent

Plus efficace que les plans précédents ?



30

Encore un autre plan équivalent



31

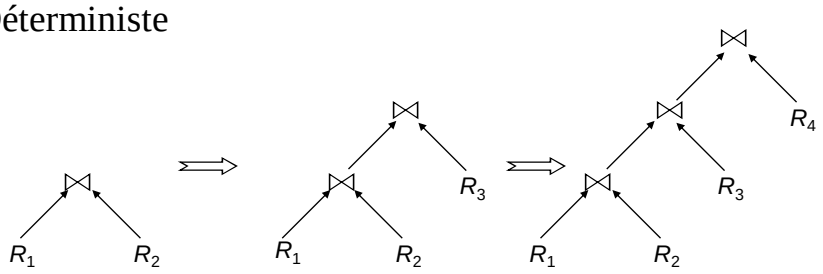
Stratégie de recherche

- Il est en général trop coûteux de faire une recherche exhaustive
- Déterministe
  - part des relations de base et construit les plans en ajoutant une relation à chaque étape
  - programmation dynamique: largeur-d'abord
  - excellent jusqu'à 5-6 relations
- Aléatoire
  - recherche l'optimalité autour d'un point de départ particulier
  - réduit le temps d'optimisation (au profit du temps d'exécution)
  - meilleur avec > 5-6 relations
    - **recuit simulé (simulated annealing)**
    - **amélioration itérative (iterative improvement)**

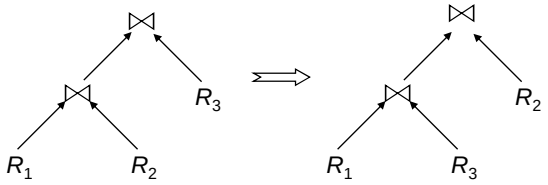
32

Stratégies de recherche

- Déterministe



- Aléatoire

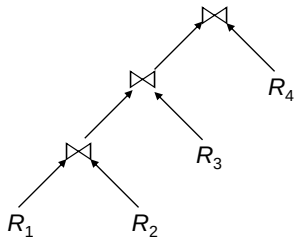


33

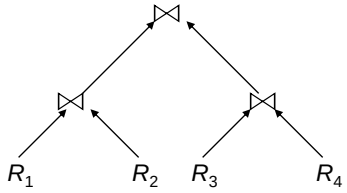
Algorithmes de recherche

- Limiter l'espace de recherche
  - heuristiques
    - par ex. appliquer les opérations unaires avant les autres
    - Ne marche pas toujours (perte d'index, d'ordre)
  - limiter la forme des arbres

Arbre linéaire



Arbre touffu



34

Génération de plan physique

- Sélection :
  - Commencer par les conditions d'égalité avec un index sur l'attribut
  - Filtrer sur cet ensemble de n-uplets ceux qui correspondent aux autres conditions
- Jointure
  - Utilisation des index, des relations déjà triées sur l'attribut de jointure, présence de plusieurs jointures sur le même attribut
- Pipelines ou matérialisation

35

Conclusion

- Point fondamental dans les SGBD
- Importance des métadonnées, des statistiques sur les relations et les index, du choix des structures d'accès.
- L'administrateur de bases de données peut améliorer les performances en créant de nouveaux index, en réglant certains paramètres de l'optimiseur de requêtes (voir TME)

36

Conclusion

- Un SGBD doit transformer une requête déclarative en un programme impératif :
  - Plan d'exécution
  - Algèbre
- Calculer les tailles des résultats intermédiaire donne une idée du coût d'un plan mais..
  - Comment mettre en œuvre les opérateurs ?
  - Comment enchaîner les opérateurs ?
  - Comment trouver le meilleur plan en fonction de ce qui précède

Réponses dans les prochains cours

37