

# Administration de Bases de Données

## - Optimisation des requêtes -

SKLAB Youcef

May 21, 2017

# Optimisation des requêtes

- 1 Généralités sur l'optimisation des requêtes
- 2 Optimisation Syntaxique
- 3 Traduction de la requête SQL en algèbre relationnelle
- 4 Tracer l'arbre algébrique
- 5 Dédire les plans d'exécutions pour l'arbre algébrique
- 6 Calculer les couts des plans
- 7 Sélectionner un plan

# Optimisation des requêtes

# Objectif de l'optimisation

Soit la requête:

**SELECT** Nom, Bureau

**FROM** Emp, Dep

**WHERE** Emp.numoD = Dep.numoD **AND** Emp.salaire > 1000

⇒ **Comment elle est exécutée par le SGBD?**

# Objectif de l'optimisation

Soit la requête:

**SELECT** Nom, Bureau

**FROM** Emp, Dep

**WHERE** Emp.numoD = Dep.numoD **AND** Emp.salaire > 1000

⇒ **Comment elle est exécutée par le SGBD?**

⇒ **Plusieurs possibilités:**

- ➊ Plusieurs façons d'exécuter une même requête
- ➋ Plusieurs variantes de requêtes équivalentes

# Objectif de l'optimisation

Soit la requête:

**SELECT** Nom, Bureau

**FROM** Emp, Dep

**WHERE** Emp.numoD = Dep.numoD **AND** Emp.salaire > 1000

⇒ **Comment elle est exécutée par le SGBD?**

⇒ **Plusieurs possibilités:**

- ➊ Plusieurs façons d'exécuter une même requête
- ➋ Plusieurs variantes de requêtes équivalentes

⇒ **Objectif de l'optimisation:**

- ➊ Trouver un plan d'exécution permettant d'améliorer le temps de traitement de la requête.

# Approches d'optimisation

- ① **Optimisation sémantique:** Rechercher les contradictions dans les restrictions qui conduirait à obtenir 0 tuple en réponse à la requête (Ex:  $\text{age} > 13$  AND  $\text{age} < 10$ ).

# Approches d'optimisation

- 1 **Optimisation sémantique:** Rechercher les contradictions dans les restrictions qui conduirait à obtenir 0 tuple en réponse à la requête (Ex:  $\text{age} > 13$  AND  $\text{age} < 10$ ).
- 2 **Optimisation syntaxique:** Réorganiser les différentes opérations relationnelles élémentaires d'une requête.



# Approches d'optimisation

- ❶ **Optimisation sémantique:** Rechercher les contradictions dans les restrictions qui conduirait à obtenir 0 tuple en réponse à la requête (Ex:  $\text{age} > 13$  AND  $\text{age} < 10$ ).
- ❷ **Optimisation syntaxique:** Réorganiser les différentes opérations relationnelles élémentaires d'une requête.
- ❸ **Optimisation physique:** Prend en compte les index, les statistiques (taille des tables et sélectivité des restrictions), la gestion de la mémoire cache et les principes algorithmiques classiques de l'accès au données.

# Plan d'exécution

- 1 Un plan d'exécution de requête est un arbre dont les nœuds sont des opérateurs de l'algèbre relationnelle.

# Plan d'exécution

- ① Un plan d'exécution de requête est un arbre dont les nœuds sont des opérateurs de l'algèbre relationnelle.
- ② Une requête peut donner lieu à plusieurs plans d'exécution.

# Plan d'exécution

- ① Un plan d'exécution de requête est un arbre dont les nœuds sont des opérateurs de l'algèbre relationnelle.
- ② Une requête peut donner lieu à plusieurs plans d'exécution.
- ③ Pour une requête donnée, quels plans doit-on considérer ? Comment peut-on estimer le coût total d'un plan ?

# Plan d'exécution

- ① Un plan d'exécution de requête est un arbre dont les nœuds sont des opérateurs de l'algèbre relationnelle.
- ② Une requête peut donner lieu à plusieurs plans d'exécution.
- ③ Pour une requête donnée, quels plans doit-on considérer ? Comment peut-on estimer le coût total d'un plan ?
- ④ Idéalement, on veut trouver le meilleur plan. En pratique, on choisira le moins pire!

# Bilan de l'optimisation : le plan d'exécution choisi et calcul de coût

- ① Le choix du plan d'exécution optimal se fera par un calcul de coût, c'est-à-dire de la performance.
- ② Le calcul de coût consiste à calculer le temps de traitement maximum des opérations élémentaires et de leur succession.
- ③ Dans le calcul de coût (donc dans la performance de la requête), interviennent :
  - Le nombre d'entrée-sortie (accès disque)
  - Le temps de calcul des opérations élémentaires
  - La taille des buffers requis

# Optimisation Syntaxique

# Les Étapes de l'optimisation de requêtes

- 1 Traduction de la requête SQL en algèbre relationnelle



# Les Étapes de l'optimisation de requêtes

- 1 Traduction de la requête SQL en algèbre relationnelle
- 2 Tracer l'arbre algébrique

# Les Étapes de l'optimisation de requêtes

- 1 Traduction de la requête SQL en algèbre relationnelle
- 2 Tracer l'arbre algébrique
- 3 Dédurre les plans d'exécutions pour l'arbre algébrique

# Les Étapes de l'optimisation de requêtes

- 1 Traduction de la requête SQL en algèbre relationnelle
- 2 Tracer l'arbre algébrique
- 3 Dédurre les plans d'exécutions pour l'arbre algébrique
- 4 Calculer les coûts des plans

# Les Étapes de l'optimisation de requêtes

- 1 Traduction de la requête SQL en algèbre relationnelle
- 2 Tracer l'arbre algébrique
- 3 Dédire les plans d'exécutions pour l'arbre algébrique
- 4 Calculer les coûts des plans
- 5 Sélectionner un plan.

# Exemple

Soit la requête:

**SELECT** Nom, Bureau

**FROM** Employe, Departement

**WHERE** Employe.numoD = Departement.numoD **AND** Employe.salaire  
> 1000

# Étape 01: Traduction de la requête SQL en algèbre relationnelle

# Rappel de Vocabulaire: Algèbre relationnelle

① La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$

# Rappel de Vocabulaire: Algèbre relationnelle

- 1 La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- 2 La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$



# Rappel de Vocabulaire: Algèbre relationnelle

- 1 La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- 2 La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$
- 3 Le produit cartésien  $\times$  :  $R \times S$

# Rappel de Vocabulaire: Algèbre relationnelle

- 1 La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- 2 La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$
- 3 Le produit cartésien  $\times$  :  $R \times S$
- 4 La jointure  $\bowtie$  :  $Client \bowtie_{numero=no\_client} Vente$

# Rappel de Vocabulaire: Algèbre relationnelle

- ① La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- ② La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$
- ③ Le produit cartésien  $\times$  :  $R \times S$
- ④ La jointure  $\bowtie$  :  $Client \bowtie_{numero=no\_client} Vente$
- ⑤ L'union  $\cup$  :  $R \cup S$

# Rappel de Vocabulaire: Algèbre relationnelle

- ① La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- ② La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$
- ③ Le produit cartésien  $\times$  :  $R \times S$
- ④ La jointure  $\bowtie$  :  $Client \bowtie_{numero=no\_client} Vente$
- ⑤ L'union  $\cup$  :  $R \cup S$
- ⑥ La différence  $-$  :  $R - S$

# Rappel de Vocabulaire: Algèbre relationnelle

- ① La projection  $\pi$  :  $\pi_{Nom,adresse}(Client)$
- ② La sélection  $\sigma$  :  $\sigma_{adresse=Bejaia}(Client)$
- ③ Le produit cartésien  $\times$  :  $R \times S$
- ④ La jointure  $\bowtie$  :  $Client \bowtie_{numero=no\_client} Vente$
- ⑤ L'union  $\cup$  :  $R \cup S$
- ⑥ La différence  $-$  :  $R - S$
- ⑦ Intersection, division, ...

# Étape 01: Traduction

## ■ Requête SQL

```
SELECT Nom, Bureau  
FROM Employe, Departement  
WHERE Employe.numoD = Departement.numoD AND Employe.salaire  
> 1000
```

## ■ Traduction en algèbre relationnelle

---

# Étape 01: Traduction

## ■ Requête SQL

```
SELECT Nom, Bureau
FROM Employe, Departement
WHERE Employe.numoD = Departement.numoD AND Employe.salaire
> 1000
```

## ■ Traduction en algèbre relationnelle

---


$$\pi_{Nom, Bureau} (\sigma_{E.salaire > 1000} (Employe \bowtie_{E.numD = D.numD} Departement))$$


---

# Étape 02: Tracer l'arbre algébrique



## Étape 02: Arbre algébrique

### 1 Structure de données arborescente

## Étape 02: Arbre algébrique

- 2 Visualiser graphiquement une requête relationnelle traduite en une expression équivalente de l'algèbre relationnelle.

## Étape 02: Arbre algébrique

- ③ Feuilles: tables utilisées dans la requête

## Étape 02: Arbre algébrique

- ④ Nœuds intermédiaires: opérations algébriques

## Étape 02: Arbre algébrique

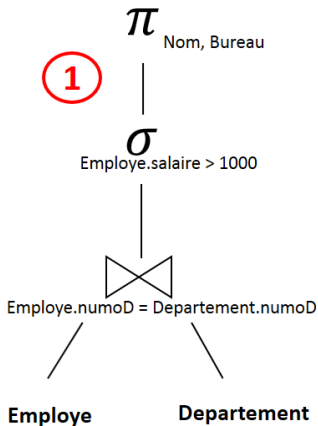
- 5 **Nœud racine:** dernière opération algébrique avant le retour du résultat

## Étape 02: Arbre algébrique

- 1 Structure de données arborescente
- 2 Visualiser graphiquement une requête relationnelle traduite en une expression équivalente de l'algèbre relationnelle.
- 3 **Feuilles:** tables utilisées dans la requête
- 4 **Nœuds intermédiaires:** opérations algébriques
- 5 **Nœud racine:** dernière opération algébrique avant le retour du résultat
- 6 L'opération d'un nœud intermédiaire est déclenchée quand ses opérandes sont disponibles. On remplace alors le nœud par la relation produite par l'opération.

## Étape 02: Arbre algébrique - Exemple

$\pi_{Nom, Bureau} (\sigma_{E.salaire > 1000} (Employee \bowtie_{E.numD=D.numD} Departement))$



# Étape 03: Déduire les plans d'exécutions pour l'arbre algébrique



## Étape 03: Principe intuitif de l'optimisation intuitive

Le principe général de l'optimisation repose sur le constat suivant :

- ❶ Les opérations unaires produisent des tables plus petites que la table d'origine.
- ❷ Les opérations binaires produisent des tables plus grandes que la table d'origine. Autrement dit, ce sont les produits cartésiens des jointures qui accroissent la taille des tables intermédiaires.

⇒ **Idée:** Supprimer un maximum de lignes et de colonnes avant de faire les jointures et faire les jointures avant les produits cartésiens.

## Étape 03: Règles de transformation

■ Règle 1: Commutativité et associativité de  $\bowtie$  et  $\times$

$$\begin{aligned} E_1 \bowtie E_2 &= E_2 \bowtie E_1, \\ (E_1 \bowtie E_2) \bowtie E_3 &= E_1 \bowtie (E_2 \bowtie E_3). \end{aligned}$$

$\Rightarrow$  Si  $(E_1 \bowtie E_2)$  est plus grand que  $(E_2 \bowtie E_3)$ , choisir  $E_1 \bowtie (E_2 \bowtie E_3)$

$$\begin{aligned} E_1 \times E_2 &= E_2 \times E_1, \\ (E_1 \times E_2) \times E_3 &= E_1 \times (E_2 \times E_3). \end{aligned}$$

## Étape 03: Règles de transformation

### ■ Règle 2: Cascade de projections

$$\pi_{A_1, \dots, A_n}(\pi_{B_1, \dots, B_m}(E)) = \pi_{A_1, \dots, A_n}(E)$$

### ■ Règle 3: Cascade de sélections

$$\sigma_{F_1}(\sigma_{F_2}(E)) = \sigma_{F_1 \wedge F_2}(E)$$

## Étape 03: Règles de transformation

### ■ Règle 4: Commutativité de la sélection avec la projection

Si  $F$  ne porte que sur  $A_1, \dots, A_n$ ,

$$\pi_{A_1, \dots, A_n}(\sigma_F(E)) = \sigma_F(\pi_{A_1, \dots, A_n}(E))$$

Si  $F$  porte aussi sur  $B_1, \dots, B_m$ ,

$$\pi_{A_1, \dots, A_n}(\sigma_F(E)) = \pi_{A_1, \dots, A_n}(\sigma_F(\pi_{A_1, \dots, A_n, B_1, \dots, B_m}(E)))$$

## Étape 03: Règles de transformation

- Règle 5: Commutativité de la sélection avec  $\times$ ,  $\cup$ ,  $-$ ,  $\bowtie$

$$\sigma_F(E_1 OPE_2) = \sigma_F(E_1)OP\sigma_F(E_2)$$

- Règle 6: Commutativité de la projection avec  $\cup$ ,  $\times$

$$\pi_{A_1, \dots, A_n}(E_1 OPE_2) = \pi_{A_1, \dots, A_n}(E_1)OP\pi_{A_1, \dots, A_n}(E_2)$$

## Étape 03: Les Étapes de l'optimisation intuitive

L'optimisation intuitive se résume à :

- 1 Faire toutes les restrictions spécifiques pour limiter le nombre de tuples.

## Étape 03: Les Étapes de l'optimisation intuitive

L'optimisation intuitive se résume à :

- 1 Faire toutes les restrictions spécifiques pour limiter le nombre de tuples.
- 2 Faire toutes les projections mono-tables possibles pour limiter le nombres d'attributs.

## Étape 03: Les Étapes de l'optimisation intuitive

L'optimisation intuitive se résume à :

- ① Faire toutes les restrictions spécifiques pour limiter le nombre de tuples.
- ② Faire toutes les projections mono-tables possibles pour limiter le nombres d'attributs.
- ③ Faire les jointures et après chaque jointure, les projections possibles.



## Étape 03: Les Étapes de l'optimisation intuitive

L'optimisation intuitive se résume à :

- 1 Faire toutes les restrictions spécifiques pour limiter le nombre de tuples.
- 2 Faire toutes les projections mono-tables possibles pour limiter le nombres d'attributs.
- 3 Faire les jointures et après chaque jointure, les projections possibles.
- 4 Faire les distinct, les tris, les group by, les fonctions de groupe.

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

- 1 Séparer les restrictions comportant plusieurs prédicats (R3).

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

- 1 Séparer les restrictions comportant plusieurs prédicats (R3).
- 2 Faire descendre les restrictions le plus bas possible (R4, R5)

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

- 1 Séparer les restrictions comportant plusieurs prédicats (R3).
- 2 Faire descendre les restrictions le plus bas possible (R4, R5)
- 3 Regrouper les restrictions successives portant sur une même relation (R3)

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

- 1 Séparer les restrictions comportant plusieurs prédicats (R3).
- 2 Faire descendre les restrictions le plus bas possible (R4, R5)
- 3 Regrouper les restrictions successives portant sur une même relation (R3)
- 4 Séparer les projections comportant plusieurs prédicats (R2).

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

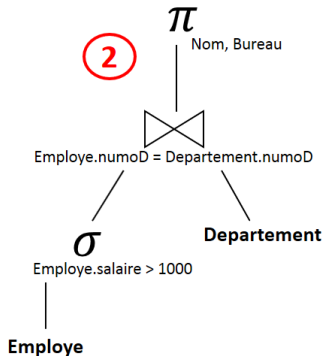
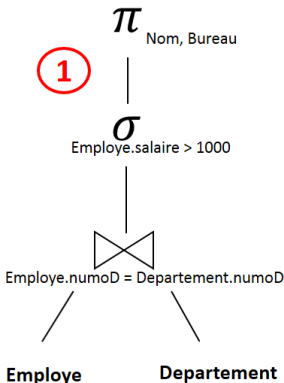
- 1 Séparer les restrictions comportant plusieurs prédicats (R3).
- 2 Faire descendre les restrictions le plus bas possible (R4, R5)
- 3 Regrouper les restrictions successives portant sur une même relation (R3)
- 4 Séparer les projections comportant plusieurs prédicats (R2).
- 5 Faire descendre les projections le plus bas possibles (R4, R6)

## Étape 03: L'optimisation d'après les règles de transformation

En utilisant les règles de transformations précédentes, on arrive à l'algorithme suivant:

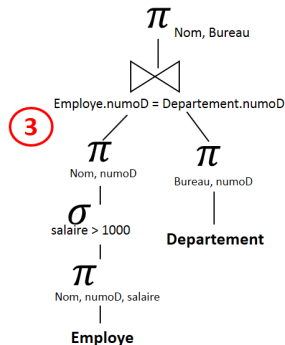
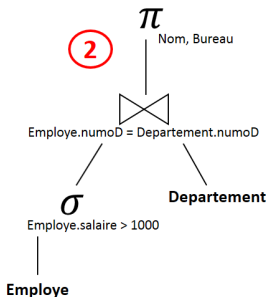
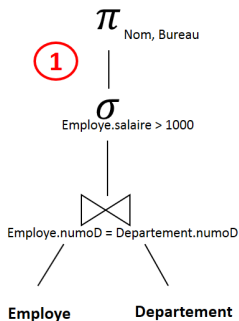
- 1 Séparer les restrictions comportant plusieurs prédicats (R3).
- 2 Faire descendre les restrictions le plus bas possible (R4, R5)
- 3 Regrouper les restrictions successives portant sur une même relation (R3)
- 4 Séparer les projections comportant plusieurs prédicats (R2).
- 5 Faire descendre les projections le plus bas possibles (R4, R6)
- 6 Regrouper les projections successives (R2)

## Plan d'exécution - Arbre 2

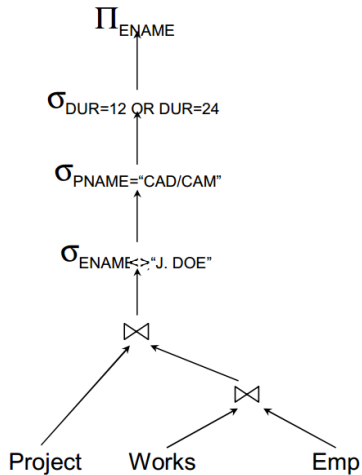
$$\pi_{Nom, Bureau} (\sigma_{E.salaire > 1000} (Employee \bowtie_{E.numD=D.numD} Departement))$$




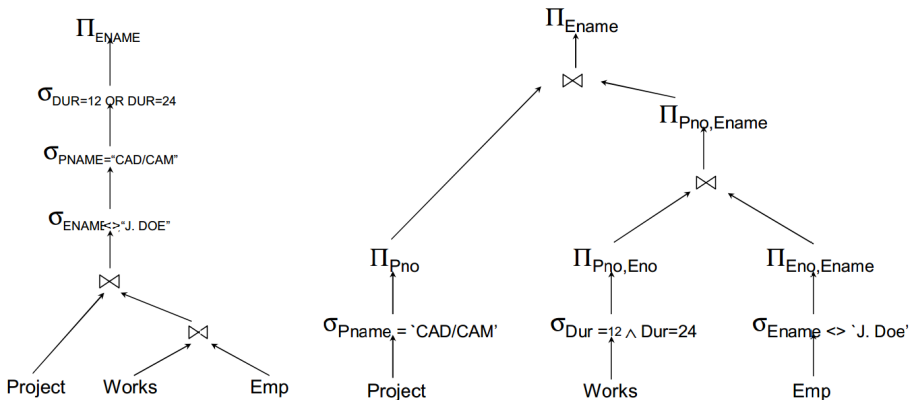
## Plan d'exécution - Arbre 3

$$\pi_{Nom, Bureau} (\sigma_{E.salaire > 1000} (Employee \bowtie_{E.numD=D.numD} Departement))$$


# Exemple

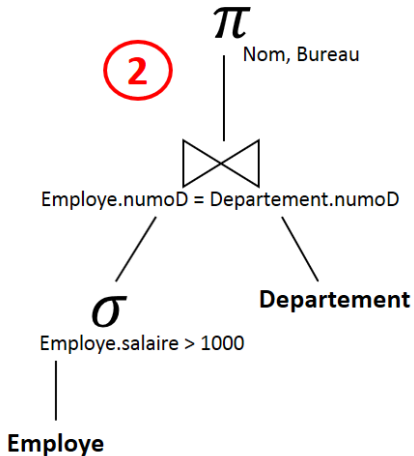
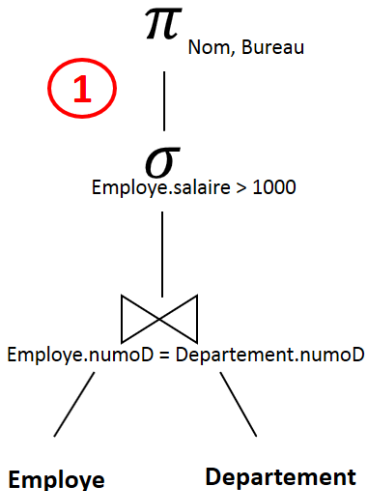


## Exemple (suite)



# Étape 04: Calculer les couts des plans

# Étape 04: Calcul du coût



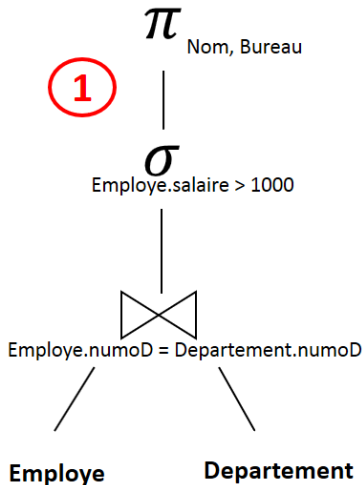
## Étape 04: Calcul du coût

Hypothèses:

- ① Il y a 300 lignes dans Employe
- ② Il ya 12 lignes dans Departement
- ③ On suppose qu'il n'y a que 20% d'employés ayant un salaire  $> 1000$ .

## Étape 04: Calcul du coût

⇒ Inscrire dans l'arbre algébrique le nombre de tuples récupérés dans chaque table.



## Étape 04: Calcul du coût

- 1 Le coût d'un plan dépend des algorithmes du calcul du coût des opérateurs algébriques.
- 2 Il existe plusieurs algorithmes, donc, le coût peut changer d'un algorithme à l'autre.



## Étape 04: Calcul du coût

### Algorithme:

- ① On s'intéresse au coût des opérateurs  $\bowtie$  et  $\sigma$  en terme d'Entrées/Sortie
- ② Algorithme:
  - Coût de la sélection  $\sigma_{condition}(R)$ :
    - Coût(Entrée) = Lecture de tous les tuples - Tuples(R).
    - Coût(Sortie) = Les tuples qui vérifient la condition.
  - Coût de la jointure:  $R1 \bowtie R2$ 
    - Coût(Entrée) = (Chargement) Lecture de R1, puis lecture de R2 = Tuples(R1) + (Tuples(R1)  $\times$  Tuples(R2))
    - Coût(Sortie) = les tuples qui vérifient la condition:  $\leq$  Tuples(R1)  $\times$  Tuples(R2)
  - Coût Total =  $\sum_{i=1}^{i=n} Op_i$

## Étape 04: Calcul du coût

$\pi_{Nom, Bureau}(\sigma_{E.salaire > 1000}(Employee \bowtie_{E.numD=D.numD} Departement))$

Plan initial:

- ① Coût de la jointure:  $(Employee \bowtie_{E.numD=D.numD} Departement)$ 
  - Coût(Entrée) =  $300E + 300 \times 12E = 3900E$
  - Coût(Sortie) =  $300S$  (pour chaque employé, un seul département)
  - total =  $4200E/S$
- ② Coût de la sélection  $\sigma_{condition}(R)$ :
  - Coût(Entrée) =  $300E$
  - Coût(Sortie) =  $300 \times 0.2 = 60S$
  - total =  $360E/S$
- ③ Coût Total =  $4200E/S + 360E/S = 4560E/S$

## Étape 04: Calcul du coût

$\pi_{Nom, Bureau} (\sigma_{E.salaire > 1000} (Employee \bowtie_{E.numD=D.numD} Departement))$

Plan optimisé:

- ① Coût de la sélection  $\sigma_{condition}(R)$ :
  - Coût(Entrée) = 300E
  - Coût(Sortie) =  $300 \times 0.2 = 60S$
  - total = 360E/S
- ② Coût de la jointure:  $(Employee \bowtie_{E.numD=D.numD} Departement)$ 
  - Coût(Entrée) =  $60E + 60 \times 12E = 780E$
  - Coût(Sortie) = 60S (pour chaque employé, un seul département)
  - total = 840E/S
- ③ Coût Total =  $360E/S + 840E/S = 1200E/S$

# Étape 05: Sélectionner un plan

## Étape 05: Sélectionner un plan

### ■ Principe

Le principe général pour sélectionner un plan parmi d'autres consiste à sélectionner celui qui permet de limiter le nombre d'accès disques, soit, en première approximation, le nombre de tuples parcourus.

Fin...!

Fin ...!

# Exercice Série TD

$$\pi_{m\_id, m\_nom, age}(\sigma_{age < 30}(\sigma_{couleur=rouge}(R \bowtie_{R.b\_id=B.b\_id \wedge R.m\_id=M.m\_id} (B \bowtie M))))$$

$$\pi_{m\_id, m\_nom, age}(\sigma_{age < 30}(\sigma_{couleur=rouge}(\sigma_{R.b\_id=B.b\_id}(B \times \sigma_{R.m\_id=M.m\_id}(B \times M))))))$$