

3I009 Licence d'informatique

Cours 3 : Traitement et optimisation  
de requêtes

**EMP(ENO, ENAME, TITLE)**

**PROJECT(PNO, PNAME, BUDGET)**

**WORKS(ENO, PNO, RESP, DUR)**

# • Problème

Soit la requête

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

Comment l'exprimer en SQL ?

- Avec count : traduire en algèbre ?
- Sans count : plus complexe (variables supplémentaires)

## Avec count :

```
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);
```

## Sans count :

```
SELECT DISTINCT E1.Ename, E1.Title
FROM Emp E1, Project, Works W1, Emp E2, Works W2
WHERE Budget > 250
AND E1.Eno=W1.Eno AND E2.Eno = W2.Eno
AND Project.Pno=W1.Pno AND W2.Pno = W1.Pno
AND E1.Eno <> E2.Eno;
```

# Un plan d'exécution possible

```
SELECT DISTINCT E1.Ename, E1.Title
FROM Emp E1, Project, Works W1, Emp E2, Works W2
WHERE Budget > 250
AND E1.Eno=W1.Eno AND E2.Eno = W2.Eno
AND Project.Pno=W1.Pno AND W2.Pno = W1.Pno
AND E1.Eno <> E2.Eno
)
```

$T_1 \leftarrow$  Lire la table Project et sélectionner  
les tuples de Budget > 250

$T_2 \leftarrow$  Joindre  $T_1$  avec la relation Works

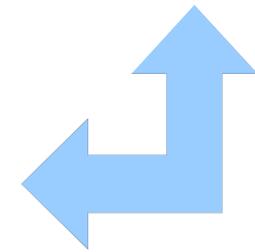
$T_3 \leftarrow$  Joindre  $T_2$  avec la relation Emp

$T_4 \leftarrow$  Joindre Works avec la relation Emp

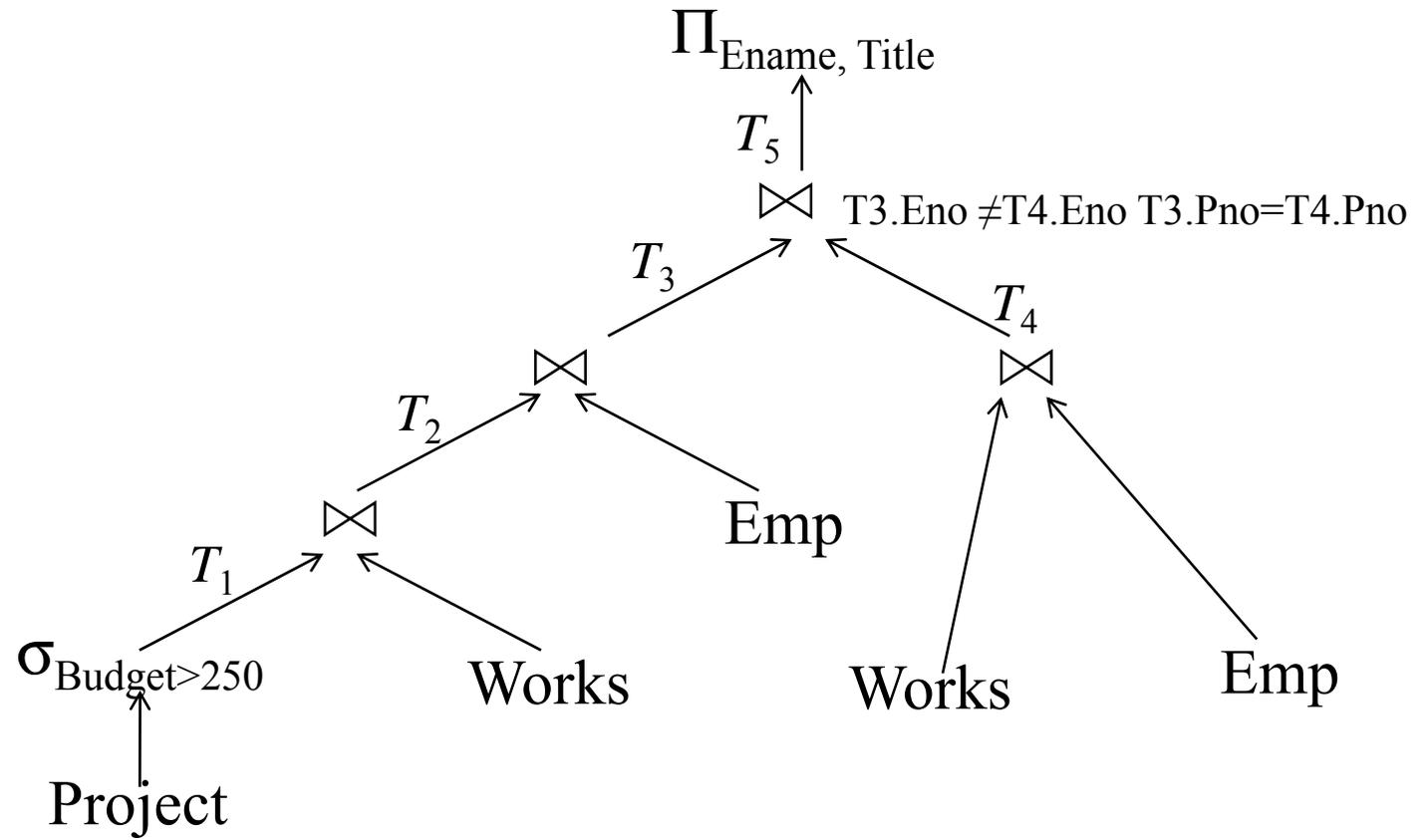
$T_5 \leftarrow$  Joindre  $T_3$  avec  $T_4$  sur Pno en vérifiant  $T_3.Eno \neq T_4.Eno$

$T_6 \leftarrow$  Projeter  $T_5$  sur Ename, Title

Résultat  $\leftarrow$  Éliminer doublons dans  $T_6$



# Représentation algébrique



# Un plan d'exécution possible (algèbre étendue)

```
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)
```

$T_1 \leftarrow$  Lire la table Project et sélectionner les tuples de Budget > 250

$T_2 \leftarrow$  Joindre  $T_1$  avec la relation Works

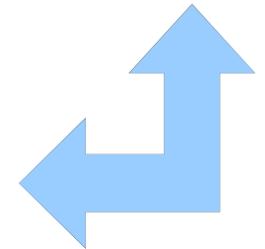
$T_3 \leftarrow$  Joindre  $T_2$  avec la relation Emp

$T_4 \leftarrow$  Grouper les tuples de Works sur Pno et pour les groupes qui ont plus de 2 tuples, projeter sur Pno

$T_5 \leftarrow$  Joindre  $T_3$  avec  $T_4$

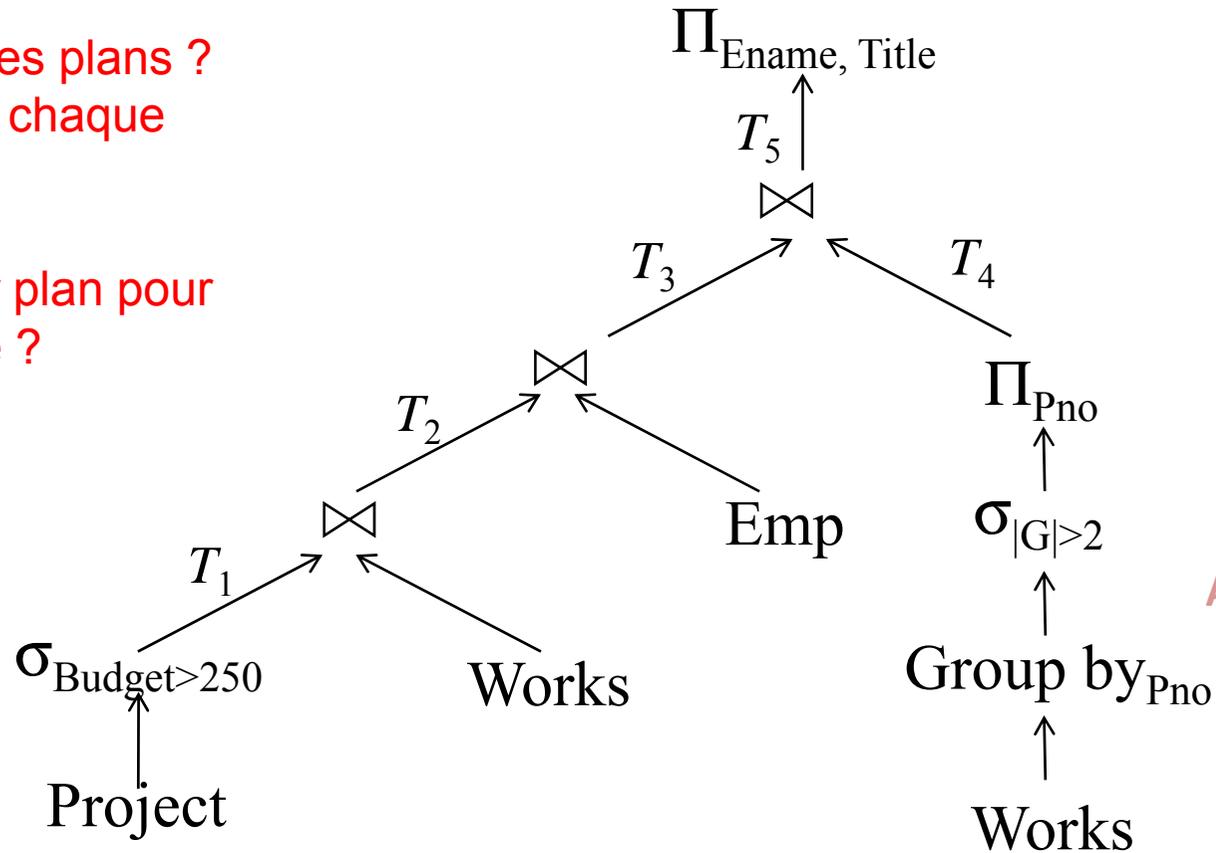
$T_6 \leftarrow$  Projeter  $T_5$  sur Ename, Title

Résultat  $\leftarrow$  Éliminer doublons dans  $T_6$



# Représentation algébrique

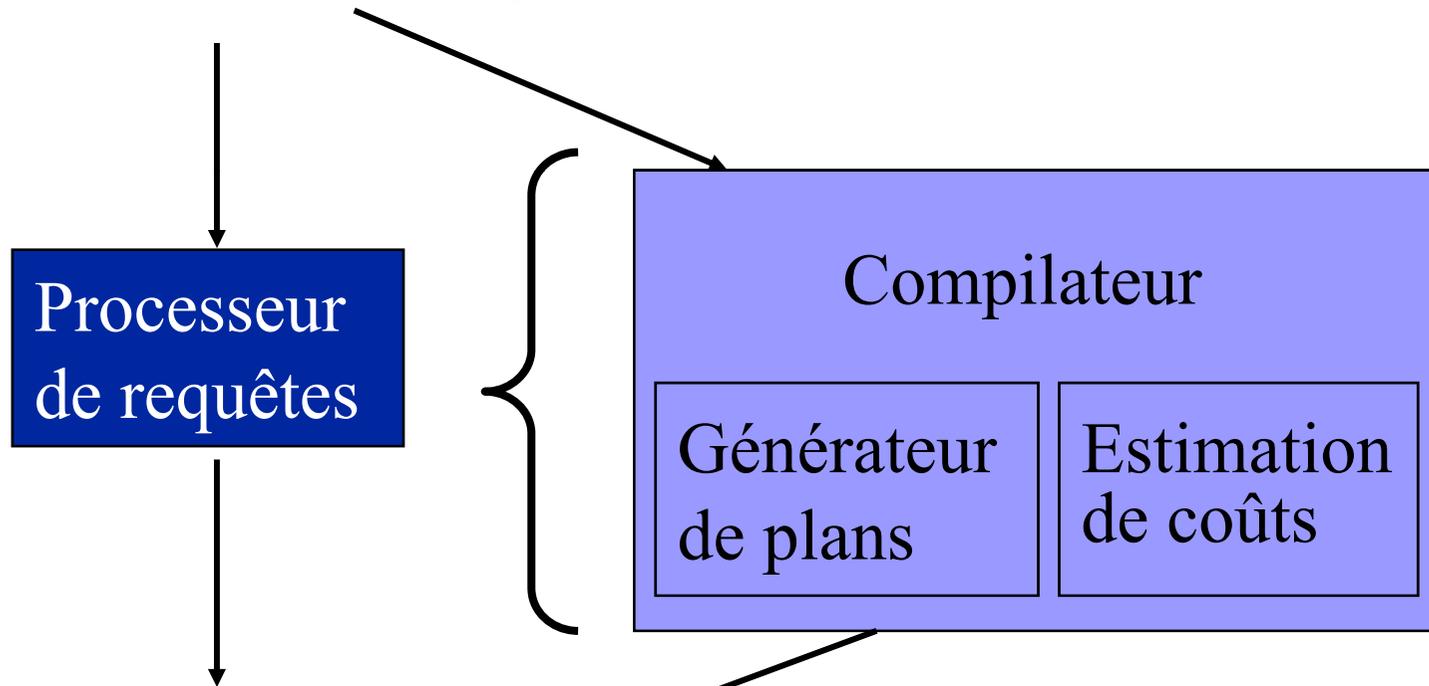
- Comment obtenir ces plans ?
- Comment exécuter chaque nœud / sous-arbre
- Quel est le coût ?
- Quel est le meilleur plan pour une requête donnée ?



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

# • Traitement des requêtes

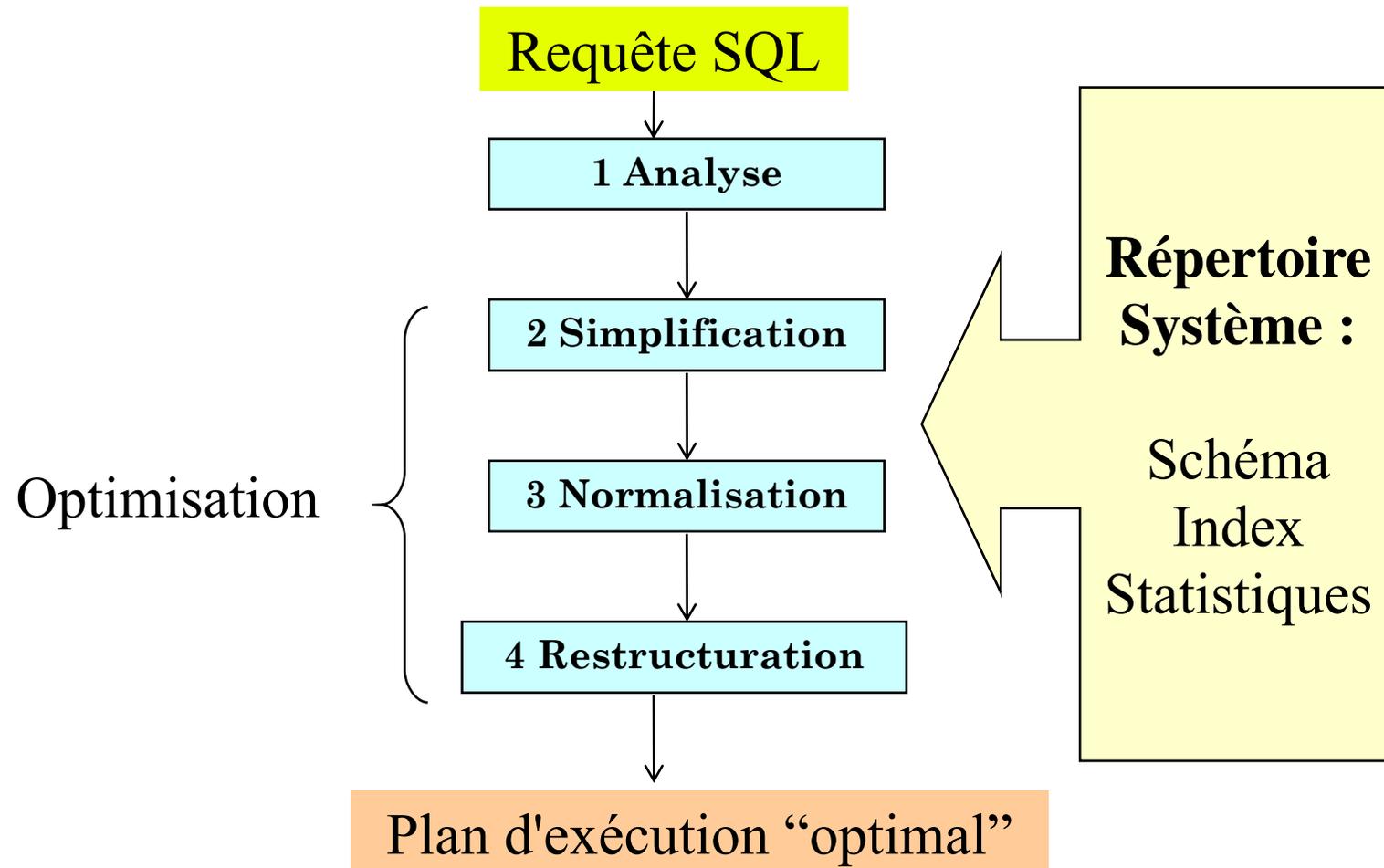
Requête déclarative (SQL)



Plan d'exécution  
(accès bas niveau aux données)

Exécution des plans

# Étapes du traitement d'une requête



# • 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  
 $(p_{11} \vee p_{12} \vee \dots \vee p_{1n}) \wedge \dots \wedge (p_{m1} \vee p_{m2} \vee \dots \vee p_{mn})$
  - forme normale disjonctive  
 $(p_{11} \wedge p_{12} \wedge \dots \wedge p_{1n}) \vee \dots \vee (p_{m1} \wedge p_{m2} \wedge \dots \wedge p_{mn})$
  - OR devient union
  - AND devient jointure ou sélection

# • 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
$$p_1 \wedge \neg(p_1) \equiv \text{faux}$$
$$p_1 \wedge (p_1 \vee p_2) \equiv p_1$$
$$p_1 \vee \text{faux} \equiv p_1$$

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

élagage

# 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)
```



$$P1 \vee (-P2 \wedge (P2 \vee P3) \wedge -P3)$$

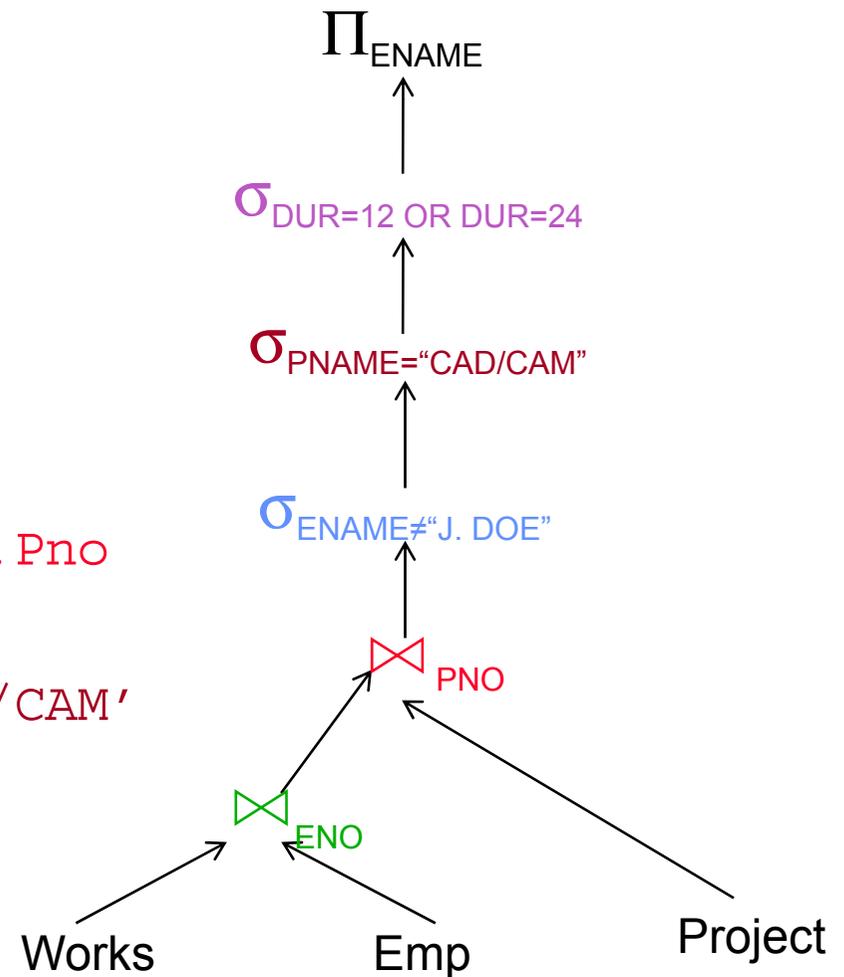
```
SELECT      Title
FROM        Emp
WHERE       Ename = 'J. Doe'
```

# Traduction en algèbre

Conversion en arbre algébrique

Exemple (ordre de la clause where) :

```
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)
```



# Alternatives de traduction

```
SELECT      Ename
FROM        Emp e, Works w
WHERE       e.Eno = w.Eno
AND         w.Dur > 37
```

Stratégie 1:

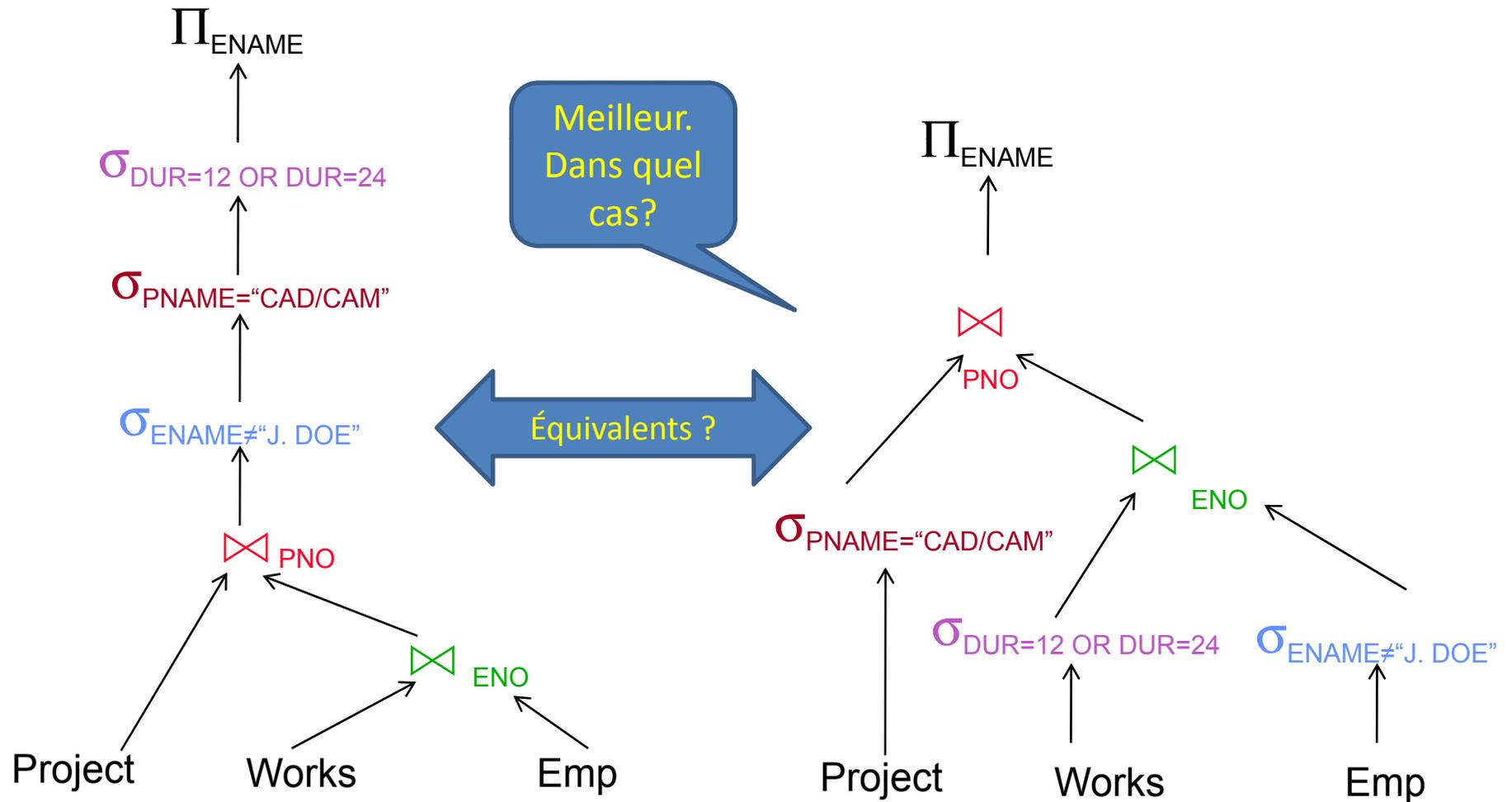
$$\Pi_{ENAME}(\sigma_{DUR>37 \wedge EMP.ENO=WORKS.ENO} (Emp \times Works))$$

Stratégie 2:

$$\Pi_{ENAME}(Emp \bowtie_{ENO} (\sigma_{DUR>37} (Works)))$$

- La stratégie 2 semble “meilleure” car elle évite un produit cartésien et sélectionne un sous-ensemble de Works avant la jointure
- Problème : Comment mesurer la qualité d’une stratégie ?

# Alternatives de traduction



# Optimisation de requête

**Objectif** : trouver le plan d'exécution le moins « coûteux »

**Fonction de coût** : donne une *estimation* du coût total réel d'un plan d'exécution

**coût total** = **coût I/O** (entrées/sorties) + coût CPU

- coût(I/O)  $\sim 1000 \cdot$  coût(CPU) : on peut *négliger le coût CPU*
- I/O se calcule en *nombre de page* (même coût de transférer une page vide ou pleine)

• **Problème 1** : Définition d'une bonne **fonction de coût**

*Solution* : statistiques (à maintenir !) et fonctions d'estimations

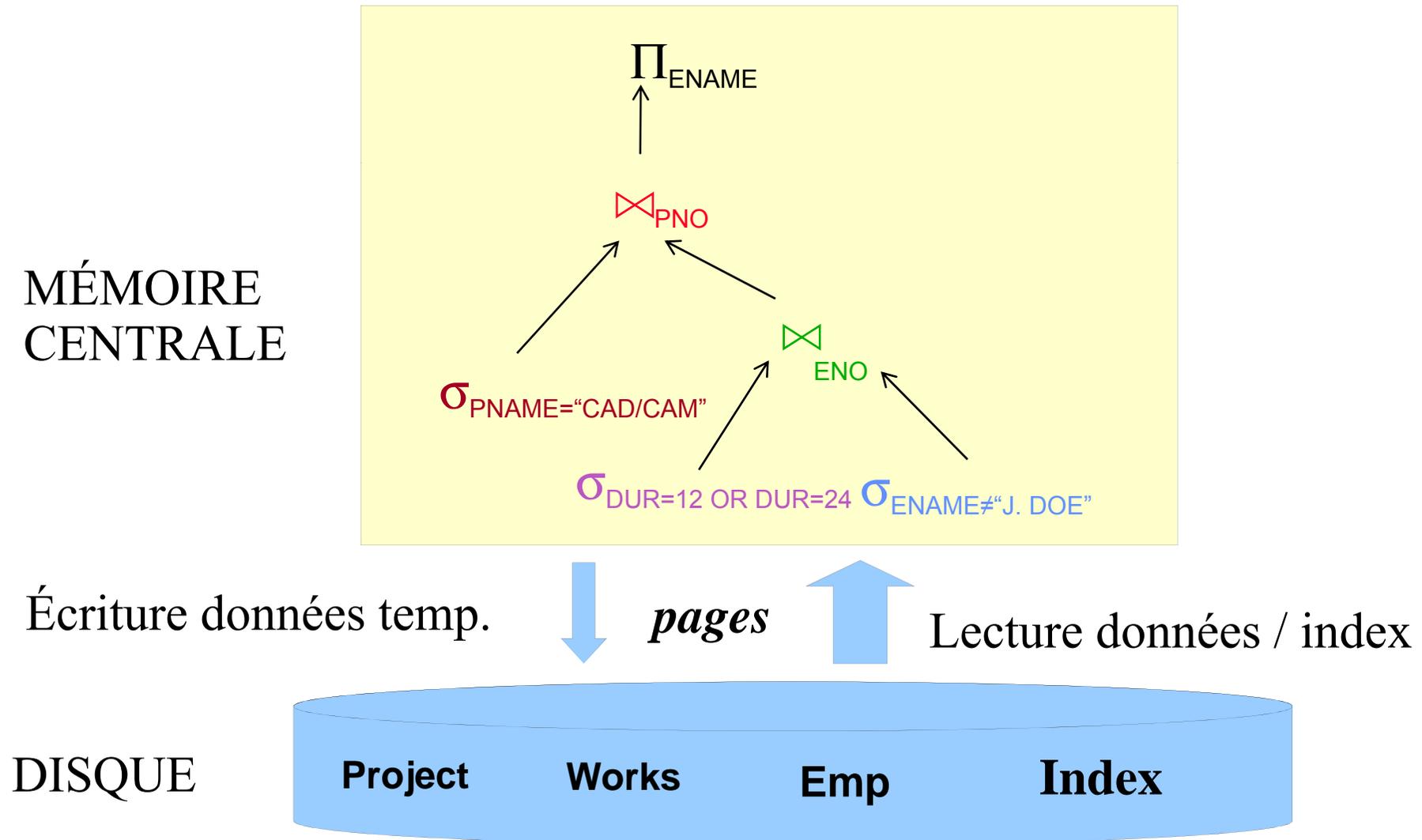
• **Problème 2** : Taille de l'**espace de recherche**

Espace de recherche = ensemble des expressions algébriques équivalentes pour une même requête.

Peut-être très grand. Optimisation en temps borné si non-compilé

*Solution* : *recherche non-exhaustive* d'une bonne solution (pas forcément la meilleure) en limitant l'espace de recherche, ou utilisation d'*heuristiques*

# Coût d'un plan d'exécution



# Estimer le coût d'un plan

La fonction de coût donne une estimation des temps I/O et CPU

nombre instructions et accès disques (écriture/lecture en nb pages)

## 1. Estimation du *nombre d'accès disque* pendant l'évaluation de chaque nœud de l'arbre algébrique

Dépend entre autres de la place mémoire disponible/taille des opérandes (principalement pour jointure) et de l'algo. utilisé pour mettre en œuvre l'opérateur (cf. semaine prochaine)

## 2. Estimation de la *taille du résultat* de chaque nœud par rapport à ses entrées :

**sélectivité des opérations** – « facteur de réduction »

influe sur la taille du résultat = opérande du prochain opérateur

basé sur les statistiques maintenues par le SGBD

# Estimer le coût d'un plan

Deux hypothèses (fortes) :

1. Hypothèse d' *uniformité* : les différentes valeurs d'un attribut ont la même probabilité.

Hypothèse plausible dans certains cas, pas dans d'autres (ex. âge)

2. Hypothèse d'*indépendance* des attributs : la probabilité d'un attribut ne dépend pas de la proba. d'un autre.

plausible : taille et couleur des cheveux

peu plausible : taille et âge

Ces hypothèses sont trop forte mais

- On ne sait pas faire mieux sinon il faut stocker des histogrammes ...
  - Coûteux
  - Compromis lecture / écriture
- L'erreur n'est pas fatale : au pire on choisit une solution un peu lente

# Tailles des relations intermédiaires

## Sélection :

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

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

où  $SF_\sigma$  est une *estimation* de la **sélectivité du prédicat**, dont la forme générale est "taille des sélectionnés / taille des possibles (domaine)"

$$SF_\sigma(A = valeur) = \frac{1}{card(\Pi_A(R))}$$

$$SF_\sigma(A > valeur) = \frac{max(A) - valeur}{max(A) - min(A)}$$

$$SF_\sigma(A < valeur) = \frac{valeur - min(A)}{max(A) - min(A)}$$

$$SF_\sigma(p(A_i) \wedge p(A_j)) = SF_\sigma(p(A_i)) \cdot SF_\sigma(p(A_j))$$

$$SF_\sigma(p(A_i) \vee p(A_j)) = SF_\sigma(p(A_i)) + SF_\sigma(p(A_j)) - (SF_\sigma(p(A_i)) \cdot SF_\sigma(p(A_j)))$$

$$SF_\sigma(A \in ens\_valeurs) = SF_\sigma(A=valeur) \cdot card(ens\_valeurs)$$

Si A continue.  
Sinon, remplacer  
par Card

# Tailles des relations intermédiaires

## Projection

$$\text{card}(\Pi_A(R)) \leq \text{card}(R) \quad (\text{égalité si } A \text{ est } \textit{unique})$$

## Produit cartésien

$$\text{card}(R \times S) = \text{card}(R) \cdot \text{card}(S)$$

## Union

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

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

## Différence

$$\text{borne sup. : } \text{card}(R - S) = \text{card}(R) \quad /* \ R \cap S = \emptyset$$

$$\text{borne inf. : } 0 \quad /* \ R \subset S$$

# Tailles des relations intermédiaires

## Jointure :

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

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

- plus généralement

$$\text{card}(R \bowtie S) = SF_J \cdot \text{card}(R) \cdot \text{card}(S)$$

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

# 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  $A' \subseteq A''$

# 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  $A' \subseteq A$ ,  $B' \subseteq B$ ,  $A_j \subseteq A'$ ,  $B_k \subseteq B'$

# Règles de transformation (suite)

- Distributivité de la jointure par rapport à l'union

$$(R \cup S) \bowtie T \equiv (R \bowtie T) \cup (S \bowtie T)$$

Ces règles de transformation permettent de passer d'une expression à une autre expression équivalente. Ceci permet d'explorer l'espace des plans d'exécution possibles pour une requête donnée

# Heuristiques

*Observation* : opérations plus ou moins *coûteuses* et plus ou moins *sélectives*

Idée : réordonner les opérations :

faire les opérateurs les moins coûteux (projection, sélection) et les plus sélectives en premier, de manière à réduire la taille des données d'entrée pour les opérateurs les plus coûteux (jointure). La place en mémoire est un facteur primordial pour l'efficacité d'une jointure (cf. dans 2 semaines)

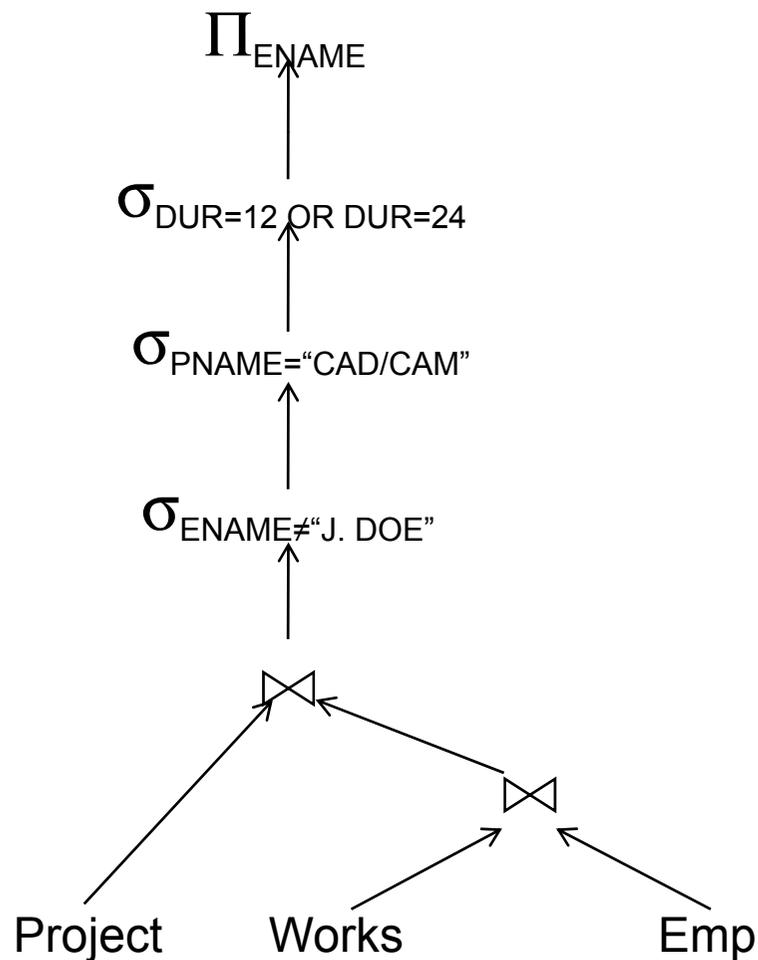
Méthode *heuristique* :

*descendre* les sélections, puis les projections au maximum grâce aux règles de transformation.

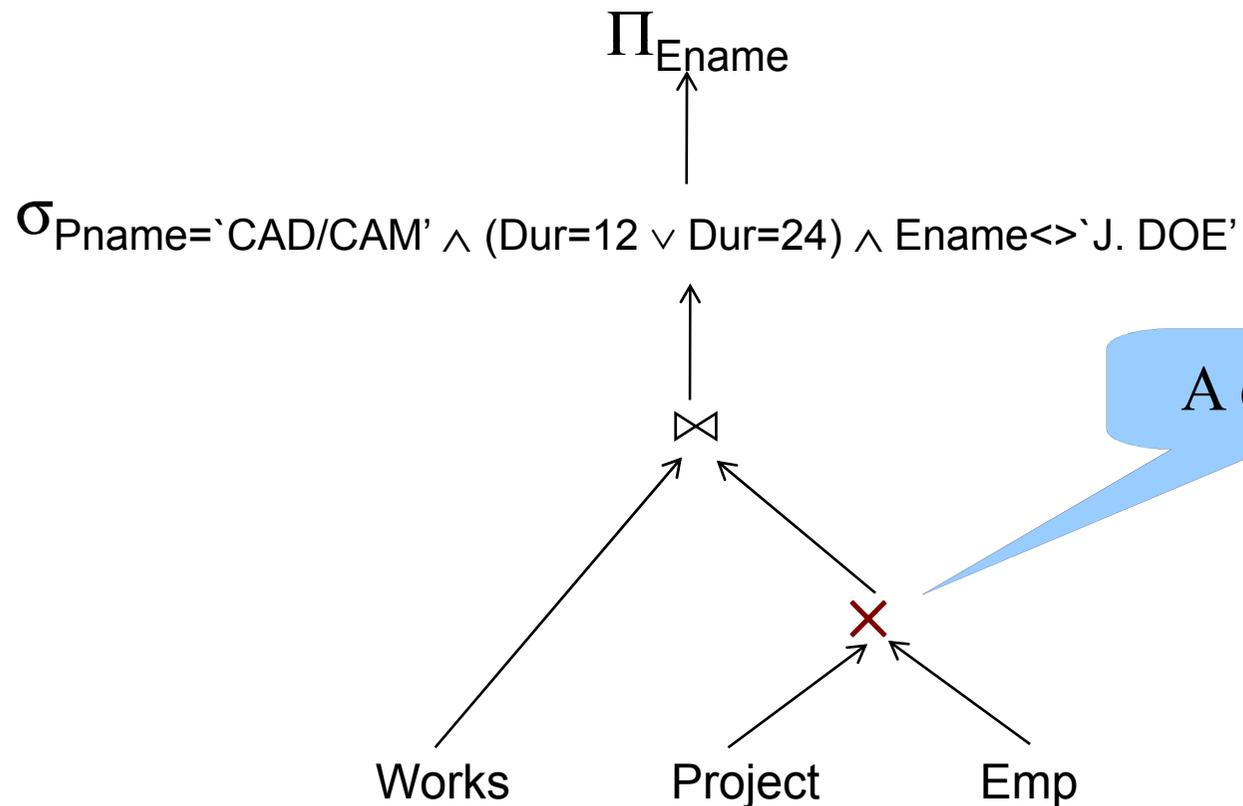
N'est pas toujours meilleur, car dépend de la présence d'index, de la nécessité d'écrire des relations temporaires...

# 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)
```



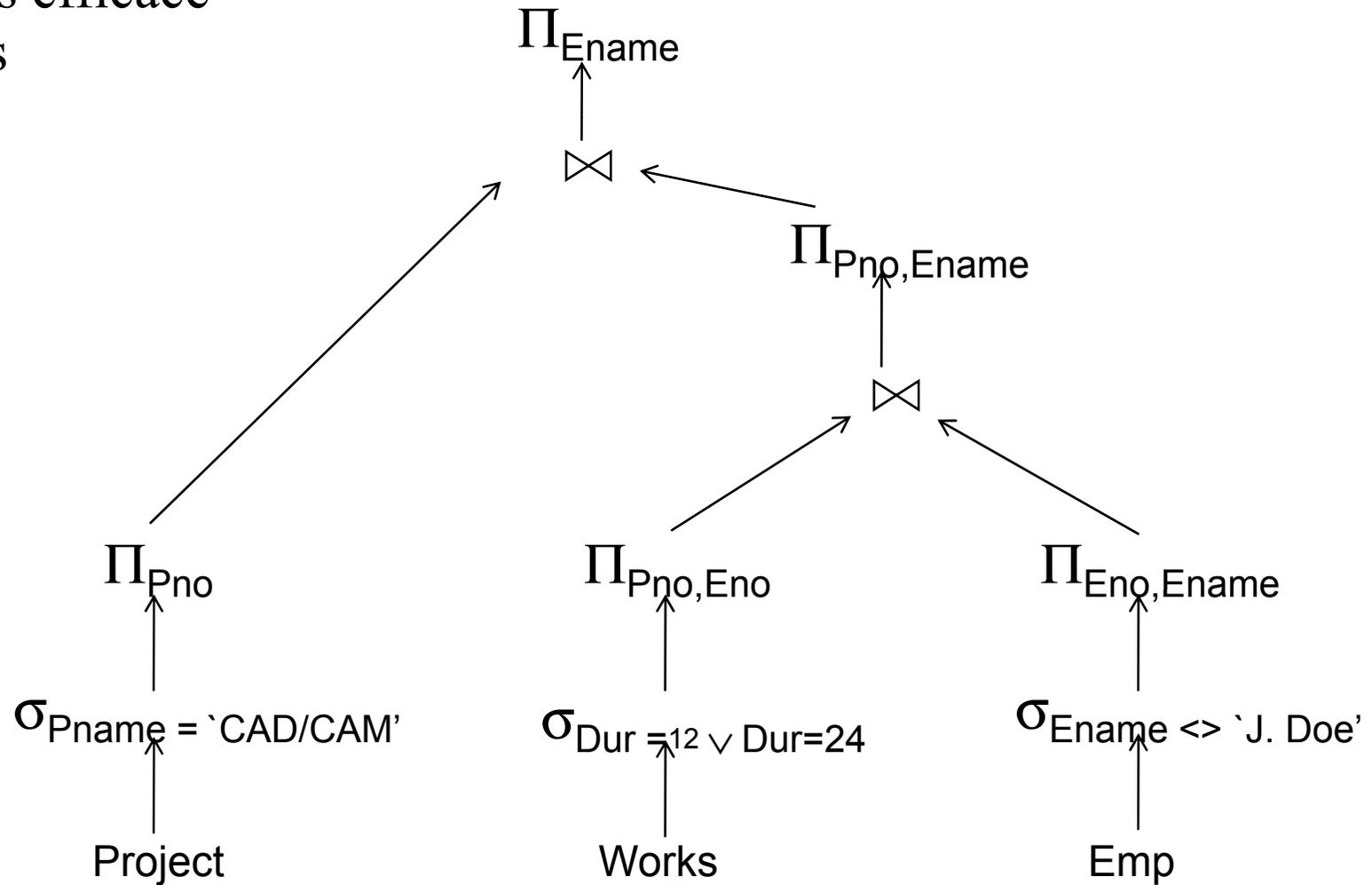
# Requête équivalente



Appliquons l'heuristique décrite précédemment...

# Autre requête équivalente

En principe plus efficace  
que les requêtes  
précédentes.



# 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 accéder aux données ?
  - Comment enchaîner les opérateurs ?
  - Comment trouver le meilleur plan en fonction de ce qui précède

Réponses dans les deux prochains cours