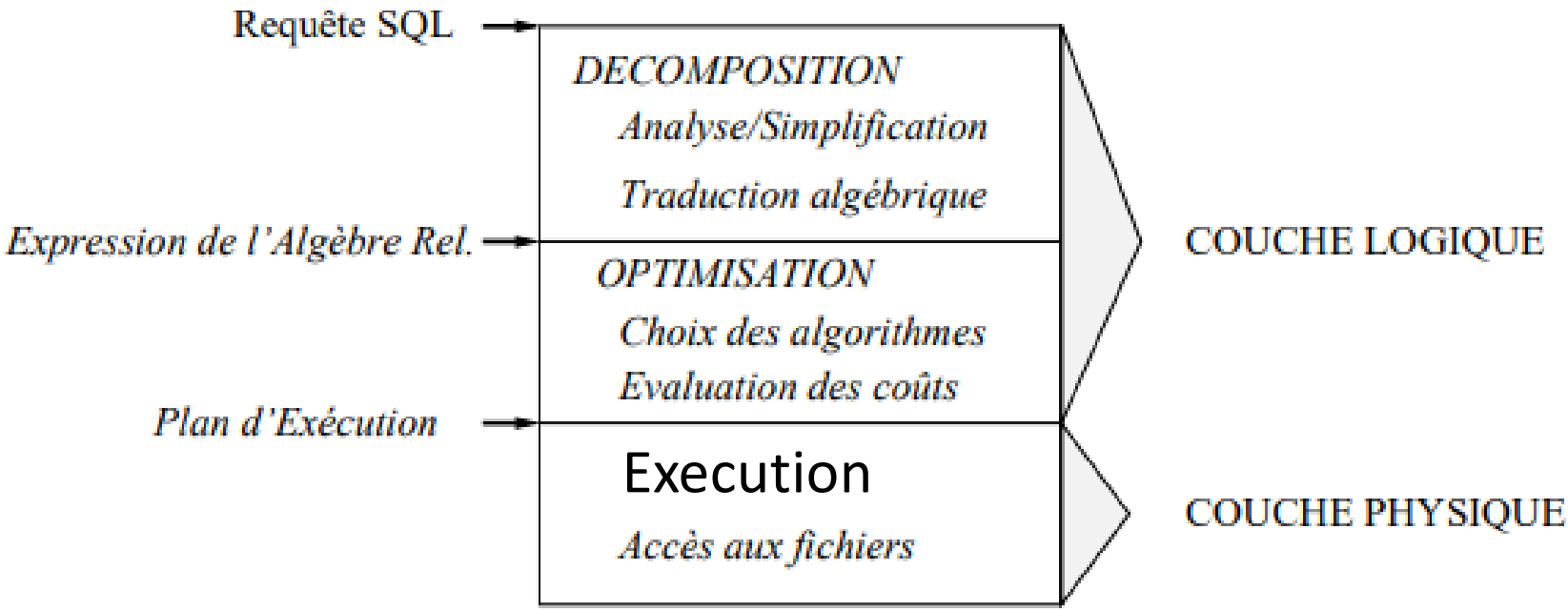
Traitement et Optimisation des Requêtes

2024

Contenu

Traitement d'une requête SQL



I. Étape de décomposition

Décomposition d'une requête

permet de transformer une requête de **haut niveau** en une requête **en algèbre relationnelle** et de vérifier que la requête est correcte sur les plans **syntaxique** et **sémantique**.

I. Étape de décomposition

a. Analyse lexicale et syntaxique

- Vérification de la validité de la requête (par rapport à la syntaxe SQL)

- Vérification des types
  - présence des attributs et relations dans le schéma
  - compatibilité des types dans les prédicats

- Décomposition en requêtes/sous-requêtes

I. Étape de décomposition

b. Normalisation

- convertir la requête en une forme normale plus facile à manipuler

❖ La forme normale conjonctive

(p11 v p12 v,,, v p1n) ^ ,,, ^ (pm1 v pm2 v ,,, v pmn)

❖ La forme normale disjonctive

(p11 ^ p12v,,, ^ p1n) v ,,, v (pm1 ^ pm2 ^,,, ^ pmn)

c. Analyse sémantique

rejeter les requêtes normalisées incorrectement formulées ou contradictoire.

I. Étape de décomposition

Graphe de requête:

- Nœud : résultat ou opérande
- Arcs entre nœuds non resultats : Jointure
- Arcs d' extrémité nœud résultat : Projection
- Nœud non résultat : peut être labellé par une expression de sélection ou une jointure

Exemple

Schéma Relationnel

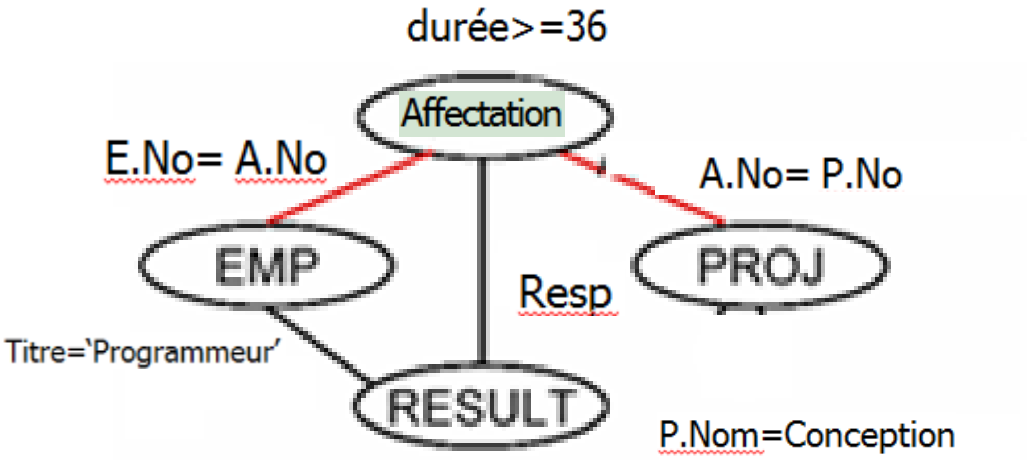
Employe (Eid, Enom, Titre, Ville)

Projet (Pid, Pnom, Budget, Ville)

Affectation (#Eid, #Pid, Resp,Durée)

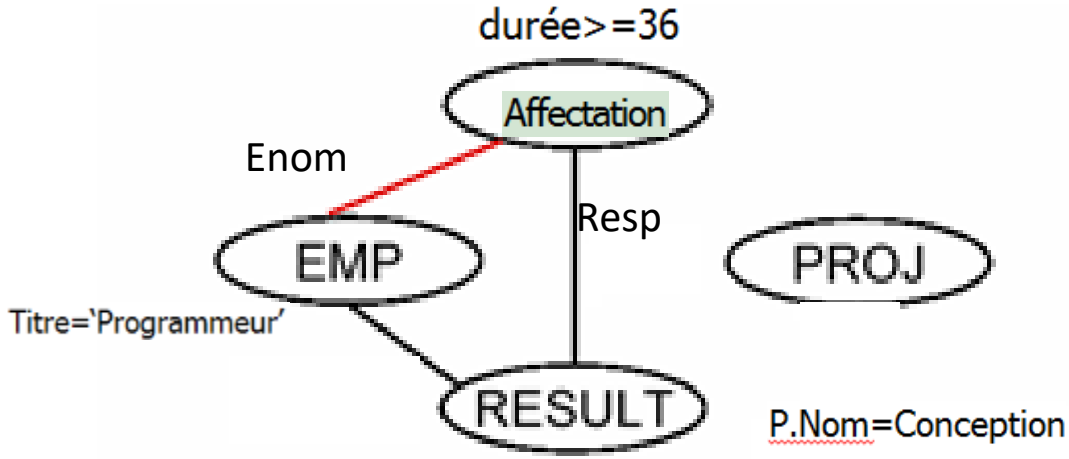
Exemple

SELECT Enom, Resp
FROM Employe E, Projet P, Affectation A
WHERE E.No= A.No
AND A.No= P.No
AND P.Nom=Conception
AND durée>=36
AND Titre='Programmeur' ;



Exemple

SELECT Enom, Resp
FROM Employe E, Projet P, Affectation A
WHERE E.No= A.No
AND P.Nom=Conception
AND durée>=36
AND Titre='Programmeur' ;



I. Étape de décomposition

d. Simplification

- Transformer la requête en une forme calculée sémantiquement équivalente, mais plus facile à calculer et plus efficace.

d.1. Élimination de la redondance : pour ce faire, on applique les règles d'idempotence :

- $p \wedge p \longleftrightarrow p; p \vee p \longleftrightarrow p$
- $p \wedge Vrai \longleftrightarrow p; p \wedge Faux \longleftrightarrow Faux$
- $p \vee Vrai \longleftrightarrow Vrai; p \vee Faux \longleftrightarrow p$
- $p \wedge \neg p \longleftrightarrow Faux; p \vee \neg p \longleftrightarrow Vrai$
- $p_1 \wedge (p_1 \vee p_2) \longleftrightarrow p_1$
- $p_1 \vee (p_1 \wedge p_2) \longleftrightarrow p_1$

I. Étape de décomposition

d.2. Éliminer des opérations redondantes

pas besoin de distinct après une projection sur une clé

d.3. Utilisation des règles d'intégrité

CI : att1 <100 Q: … where att1 > 1000…

d.4. Application de la transitivité.

e. Traduction Algébrique

La requête de **haut niveau** est transformée en une **représentation interne** sous forme d'arborescence

Règles de passage d'une requête SQL en AR :

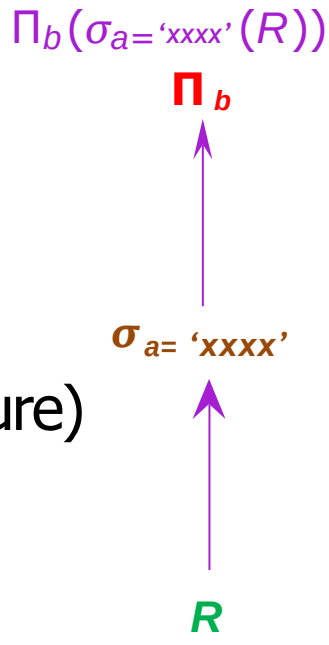
- `select` pour définir une **projection**
- `from` pour définir les **relations** (feuilles de l'arbre) et **jointures / produits cartésiens**
- `where` pour définir :
  - avec les comparaisons *attribut/valeur* des **sélections**
  - avec les comparaisons *attribut/attribut* des **jointures**

I. Etape de Décomposition

Arbre Algébrique

Une requête en algèbre relationnelle peut se représenter sous forme d'arbre algébrique

- **Racine de l'arbre** = résultat de la requête
- **Noeuds intermédiaires de l'arbre** = un opérateur algébrique (ex., sélection, union, jointure)
- **Feuilles de l'arbre** = relations



Arbre algébrique ≈ **plan d'exécution** d'une requête

Espace de recherche

Espace de recherche = l'ensemble des plans "équivalents" pour une même requête :

- Ceux qui donnent le même résultat
- Générés en appliquant des règles de transformation

Caractéristiques des plans d'un espace de recherche :

- Le coût de chaque plan est en général différent
- L'ordre des jointures est important

Si l'espace de recherche est très grand, l'optimiseur peut chercher un plan raisonnable, mais pas forcément optimal (par exemple sous PostgreSQL, dès qu'il y a plus de 12 jointures)

Exemple d'espace de recherche

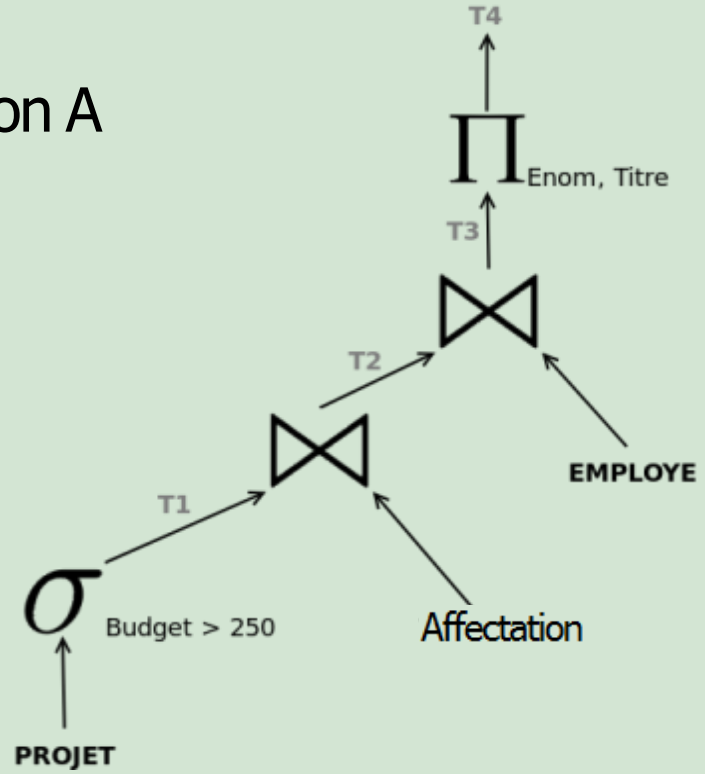
employe (Eid, Enom, Titre, Ville)
projet (Pid, Pnom, Budget, Ville)
Affectation (#Eid, #Pid, Respid,Durée)

Le nom et le titre des employés qui travaillent dans des projets avec un budget supérieur à 250

```
SELECT DISTINCT Enom, Titre
FROM Employe E, Projet P, Affectation A
WHERE P.Budget > 250
    AND E.Eid=A.Eida
    AND P.Pid=A.Pid ;
```

Exemple d'espace de recherche - plan 1

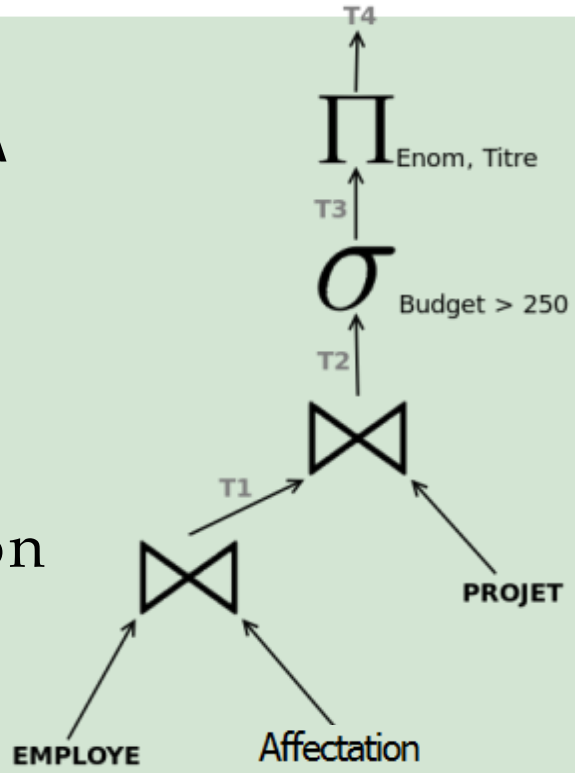
```
SELECT DISTINCT Enom, Titre
FROM Employe E, Projet P, Affectation A
WHERE P.Budget > 250
        AND E.Eid=A.Eid
        AND P.Pid=A.Pid ;
```



- $T_1 \leftarrow$ Lire la table `projet` et sélectionner les tuples de *Budget* > 250
- $T_2 \leftarrow$ Joindre T_1 avec la relation `Affectation`
- $T_3 \leftarrow$ Joindre T_2 avec la relation `employe`
- $T_4 \leftarrow$ Projeter T_3 sur *Enom*, *Titre*

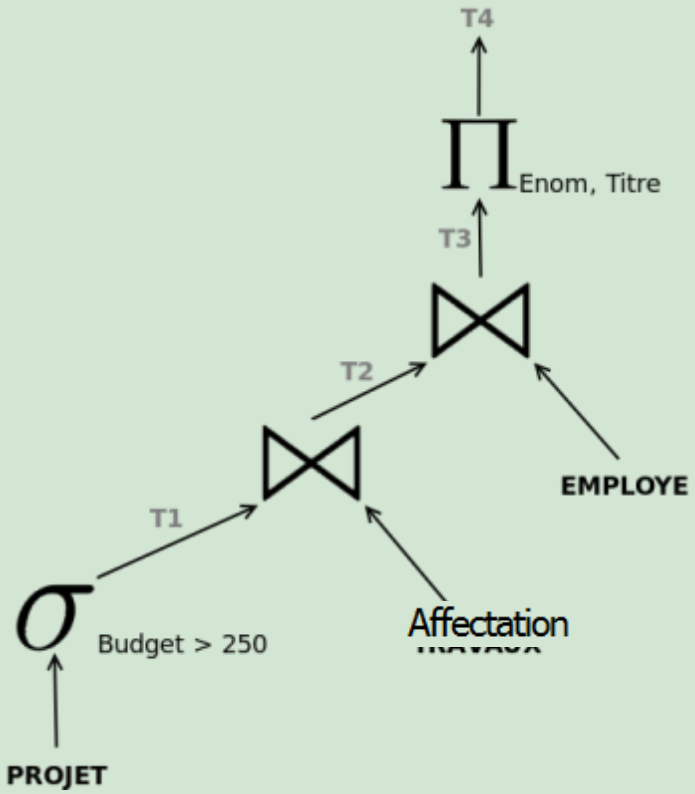
Exemple d'espace de recherche - plan 2

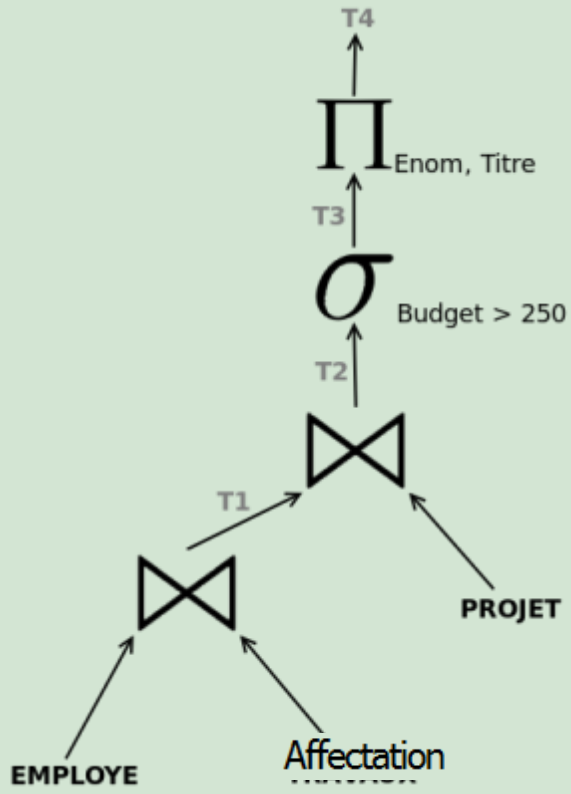
SELECT DISTINCT Enom, Titre
FROM Employe E, Projet P, Affectation A
WHERE P.Budget > 250
AND E.Eid=A.Eid
AND P.Pid=A.Pid ;



- $T_1 \leftarrow$ Joindre la relation $\mathrm{Affectation}$ avec la relation $\mathrm{employe}$
- $T_2 \leftarrow$ Joindre T_1 avec la relation projet
- $T_3 \leftarrow$ Lire T_2 et sélectionner les tuples de *Budget* >250
- $T_4 \leftarrow$ Projeter T_3 sur *Enom*, *Titre*

Exemple d'espace de recherche - choix de plan





Espace de recherche contenant deux plans d'exécution. Quel est le meilleur plan (sans considérer d'autres informations) ?

II. Optimisation d'une requête

Optimiser, c'est trouver le plan d'exécution d'une requête dont le coût soit minimal (i.e., le temps de réponse à la requête soit le plus rapide possible)

L'objectif est de trouver une **stratégie d'évaluation** permettant d'accélérer l'évaluation :

- Par manipulations algébriques, indépendante de la manière dont sont stockées les informations
- En utilisant une stratégie dépendante du stockage (clés, index)

Dans ce cours, manipulations algébriques (à base de règles)

II.1. Optimisation à base de règles

Application de règles pour transformer une expression de l'algèbre relationnelle en plan d'exécution optimisé :

- Entrée : arbre représentant l'expression à évaluer
- Sortie : plan d'exécution pour la requête

- Le **plan d'exécution** d'une requête est généralement **optimisé** selon trois opérateurs (projection, sélection et jointure)
- Les **autres opérateurs** (ex., regroupement, tri) sont réalisés ensuite (ajoutés au plan optimisé des trois opérateurs)
- Un **plan d'exécution** indique quel algorithme est appliqué pour chaque **opérateur**

Lois sur les jointures et les produits

Commutativité :

$$E_1 \infty_C E_2 \equiv E_2 \infty_C E_1 \quad (1)$$

$$E_1 \infty E_2 \equiv E\ 2 \infty E_1 \quad (2)$$

$$E_1 \times E_2 \equiv E_2 \times E_1 \quad (3)$$

Associativité :

$$E_1 \infty_C (E_2 \infty_D E_3) \equiv (E_1 \infty_C E_2) \infty_D E_3 \quad (4)$$

$$E_1 \infty (E_2 \infty E_3) \equiv (E_1 \infty E_2) \infty_C E_3 \quad (5)$$

$$E_1 \times (E_2 \times E_3) \equiv (E_1 \times E_2) \times E_3 \quad (6)$$

Lois sur les projections et les séléctions

Cascade de projections:

$$\pi_{A_1,\ldots,A_n}(\pi_{B_1,\ldots,B_k}(E)) \equiv \pi_{A_1,\ldots,A_n}(E) \quad (7)$$

Cascade de sélections:

$$\sigma_C(\sigma_D(E)) \equiv \sigma_{C\wedge D}(E) \quad (8)$$
$$\sigma_C(\sigma_D(E)) \equiv \sigma_D(\sigma_C(E)) \quad (9)$$

Permutations de projections et sélections :

- Si C ne porte que sur des attributs parmi $A_1, \ldots, A_n$:

$$\pi_{A_1,\ldots,A_n}(\sigma_C(E)) \equiv \sigma_C(\pi_{A_1,\ldots,A_n}(E)) \quad (10)$$

Lois sur les séléctions et les autres opérations

Sélections et produits cartésiens/jointures :

- Si C ne porte que sur les attributs de E_1 :

$$\sigma_C(E_1 \infty E_2) \equiv \sigma_C(E_1) \infty E_2 \quad (11)$$

$$\sigma_{C \wedge D}(E_1 \infty E_2) \equiv \sigma_D(\sigma_C(E_1) \infty E_2) \quad (12)$$

-

- Si C ne porte que sur les attributs de E_1 et D sur les attributs de E_2 :

$$\sigma_{C \wedge D}(E_1 \infty E_2) \equiv \sigma_C(E_1) \infty \sigma_D(E_2) \quad (13)$$

Sélections et union / différence :

$$\sigma_C(E_1 \cup E_2) \equiv \sigma_C(E_1) \cup \sigma_C(E_2) \quad (14)$$

$$\sigma_C(E_1 - E_2) \equiv \sigma_C(E_1) - \sigma_C(E_2) \quad (15)$$

Lois sur les projections et les autres opérations

- Jointure

$$\pi_{A_1,\ldots,A_i,\ldots,A_n}(E_1 \ \infty_c E_2) \equiv \pi_{A_1,\ldots,A_i}(E_1) \infty_c \pi_{A_{i+1},\ldots,A_n}(E_2) \quad (16)$$

- attributs c= Attributs Ai
- $A_1,\ldots,A_i$ *porte sur* $E1$ *et* $A_{i+1},\ldots,A_n$ *porte sur* $E2$

- Union :

$$\pi_{A_1,\ldots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,\ldots,A_n}(E_1) \cup \pi_{A_1,\ldots,A_n}(E_2) \quad (17)$$

1. Utiliser (8) pour transformer les sélections $\sigma_{C_1 \wedge \ldots \wedge C_n}(E)$ en cascades $\sigma_{C_1}(\ldots \sigma_{C_n}(E))$.

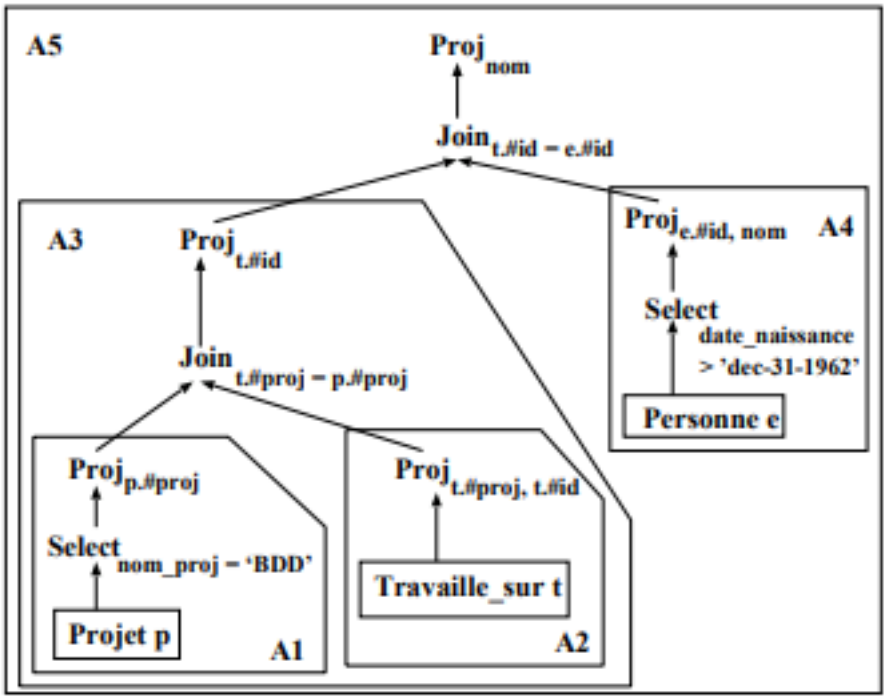
2. Pour chaque sélection, utiliser (9), (10), (11), (12), (13), (14) et (15) pour pousser les sélections en bas de l'arbre.

3. Pour chaque projection, utiliser les règles (7), (16), (17) et (10) pour pousser la projection au plus bas dans l'arbre. Éliminer une projection qui conserve tous les attributs.

4. Utiliser les règles (7), (8) et (10) pour combiner les cascades de sélections et projections en une sélection unique, une projection unique, ou une sélection suivie d'une projection.

5. Partitionner l'arbre résultant en groupes comme suit :
   - Créer un groupe par noeud binaire ($\times$, $\cup$, $\cap$, $\setminus$, ∞)
   - Le groupe inclut tous les ancêtres unaires intermédiaires
   - Le groupe inclut toute branche étiquetée par des opérateurs unaires et se terminant à une feuille (table).

6. Chaque groupe correspond à une étape. Les étapes sont à évaluer dans n'importe quel ordre tant qu'un groupe est évalué après ses descendants.

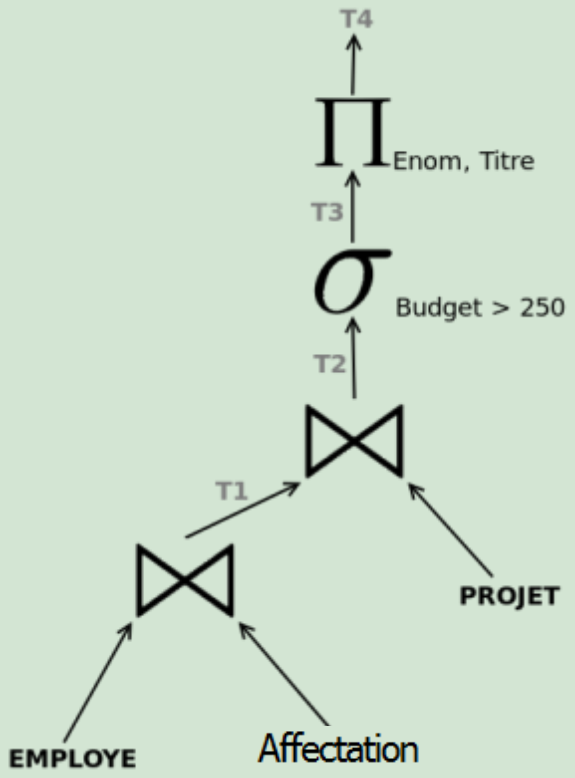


Exemple

Le nom et le titre des employés qui travaillent dans des projets avec un budget supérieur à 250

```
SELECT DISTINCT Enom, Titre
FROM Employe E, Projet P, Affectation A
WHERE P.Budget > 250
AND E.Eid=A.Eid  AND
P.Pid=A.Pid ;
```

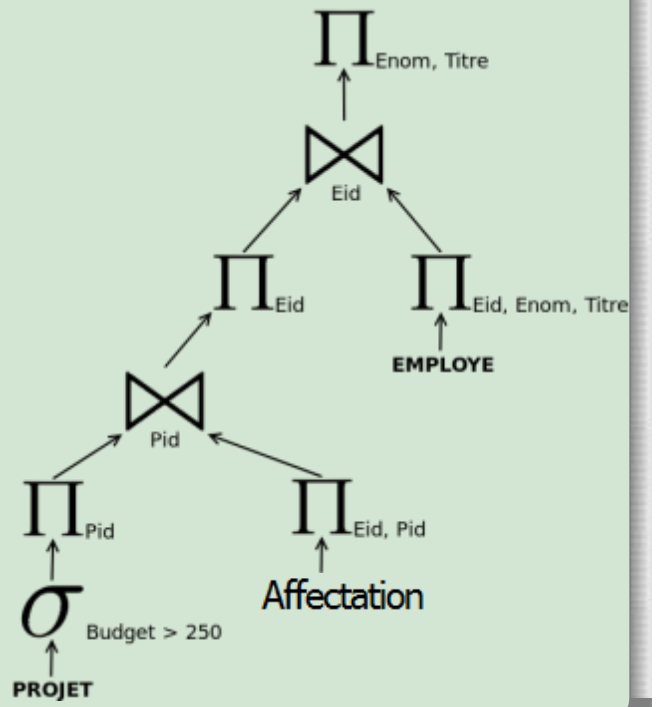
⇒ Plan non optimisé !



Exemple

Le nom et le titre des employés qui travaillent dans des projets avec un budget supérieur à 250

```
SELECT DISTINCT Enom, Titre
FROM Employe E, Projet P, Affectation A
WHERE P.Budget > 250
AND E.Eid=A.Eid  AND
P.Pid=A.Pid ;
```

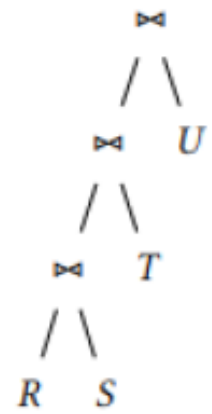


⇒ Plan optimisé par les règles :
(R11), (R10), (R16)

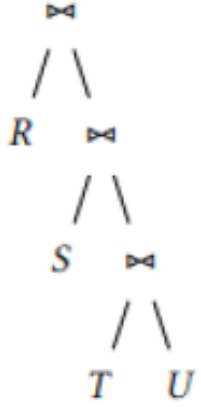
II.1. Optimisation à base de règles

Problème d'ordre de jointure

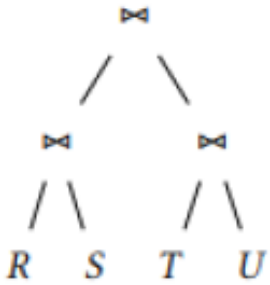
Les règles de transformation algébriques pour l'opération de jointure peuvent produire de nombreuses expressions de jointure équivalentes. En conséquence, le nombre d'arborescences de requêtes alternatives croît très rapidement avec l'augmentation du nombre de jointures dans une requête. En général, pour un bloc de requête comportant **n** relations, il existe **n!** ordres de jointure. Un arbre d'une requête ayant plus de deux jointures peut être exprimé en trois structures :



Arbre de jointure linéaire gauche



b. Arbre de jointure linéaire droit.

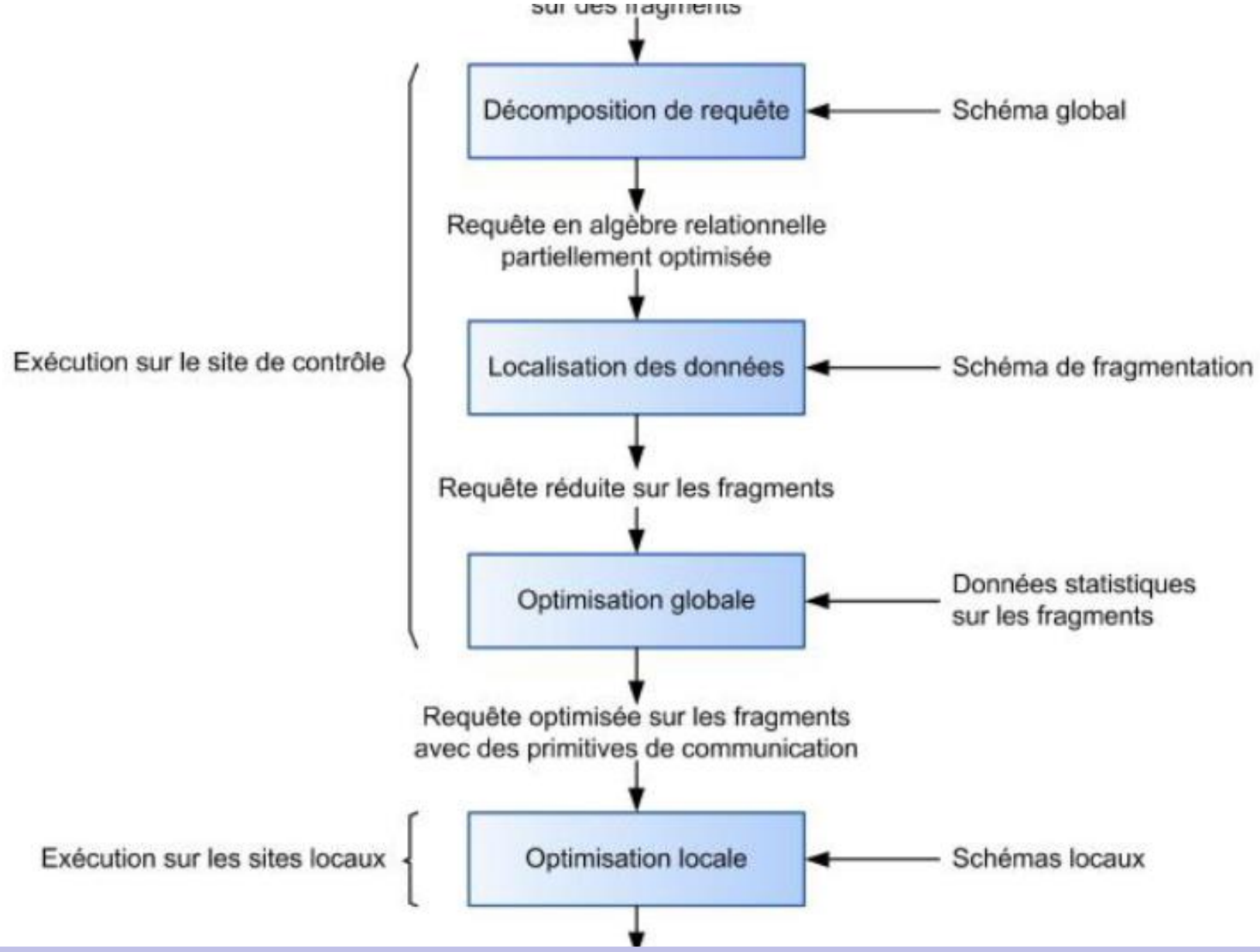


c. Arbre de jointure ramifié

En résumé

- Optimisation à base de règles = application de règles de transformation entre les opérateurs de l'AR
- Une quinzaine de règles et un algorithme à suivre
- Idée générale : placer les sélections et les projections en bas de l'arbre (le plus tôt possible)

Optimisation dans les BDRs



Localisation des données

Prendre en considération la **répartition des données** pour effectuer certaines **optimisations** plus poussées à l'aide de **règles heuristiques**.

- remplacer les **relations globales** des feuilles de l'arbre par leurs **algorithmes de reconstruction**. L'**arbre d'algèbre relationnelle** obtenu est désigné d'***arbre générique d'algèbre*** *relationnelle*.

- Utiliser ensuite des techniques de réduction pour générer une requête plus simple et **optimisée** (**arbre réduite**).

N.B. La **technique de réduction** employée dépend du **type de fragmentation** mise en place.

Types d'algorithmes

Réduction dans le cas de la fragmentation horizontale primaire
Deux cas sont envisagés :

Réduction par l'opération de sélection

- si le **prédicat de sélection** **contredit** la définition du fragment, alors il donne comme **résultat** une **relation intermédiaire vide** et les opérations peuvent être éliminées.

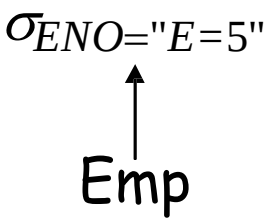
Exemple
- Supposons que: EMP est fragmentée en EMP1, EMP2 et EMP3 comme suit:

EMP1= σENO≤"E3"(EMP)
EMP2= σ"E3"<ENO≤"E6"(EMP)
EMP3= σENO≥"E6"(EMP)

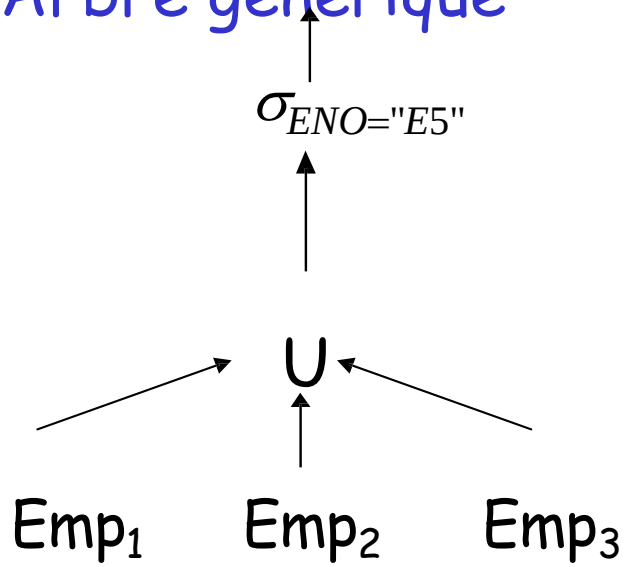
```
SELECT   *
FROM     EMP
WHERE    ENO="E5"
```

Réduction (Cas de FHP)

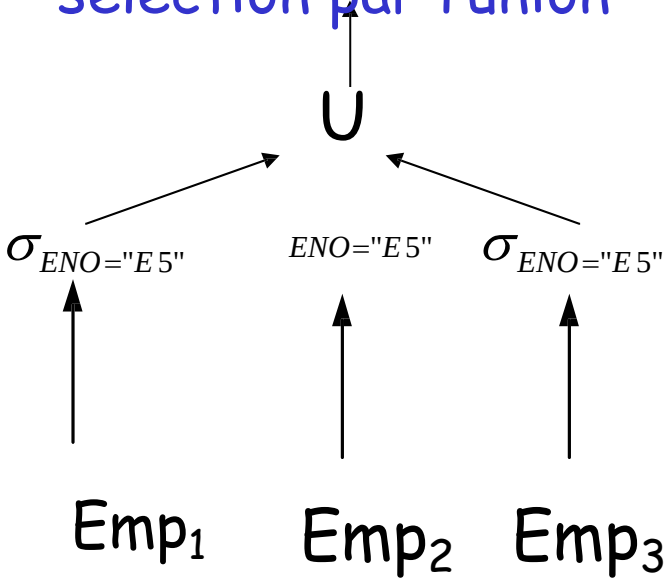
1.Arbre original



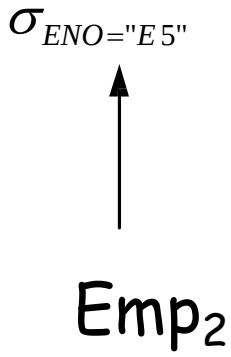
2.Arbre générique



3.Commutation de la selection par l'union



4.Arbre final



Types d'algorithmes

Réduction pour la fragmentation horizontale dérivée

La réduction dans le cas de la fragmentation horizontale dérivée fait également appel à la règle de transformation de la commutation des opérations de jointure et d'union. Dans ce cas, nous savons que la fragmentation d'une relation est basée sur l'autre relation et que, lors de la commutation, certaines jointures partielles sont probablement redondantes.

Règle:

Distribuer les jointures sur les unions

$$(R_1 \cup R_2) \bowtie S \Leftrightarrow (R_1 \bowtie S) \cup (R_2 \bowtie S)$$

- Appliquer la réduction pour la jointure de la FH.

$$R_i \bowtie R_j = \varnothing \text{ if } \forall x \text{ in } R_i,\ \forall y \text{ in } R_j: \neg(p_i(x) \wedge p_j(y))$$

Réduction (Cas de FHD)

EMP(ENO, ENom, Titre)
Affec (ENO, PNO, RESP, DUR)

Schema de Fragmentation
- EMP1: σTitre="Programmeur" (EMP)
- EMP2: σTitre<> "Programmeur" (EMP)

- Affec1: Affec $\ltimes$ENO EMP1
- Affec2: Affec $\ltimes$ENO EMP2

```
SELECT *
FROM
EMP,Affec
WHERE
Affec.ENO=EMP.ENO
AND Titre="Programmeur"
```

Réduction (Cas de FV)

Réduction pour la fragmentation verticale

La réduction dans le cadre de la fragmentation verticale suppose la suppression des fragments verticaux qui n'ont aucun attribut en commun avec les attributs de projection, à l'exception de la clé de la relation.

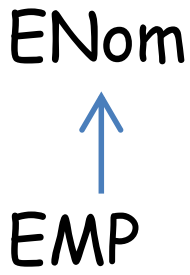
- Une relation R définie sur les attributs A = {A1, ..., An}, verticalement fragmentée par Ri = $\Pi_{A'}(R)$ où A' $\subseteq$ A: $\Pi_{D,C}(Ri)$ est inutile si l'ensemble des attributs de projection D n'est pas en A'

Réduction (Cas de FV)

Requête

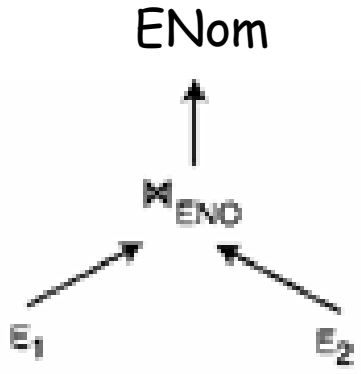
```
SELECT ENom
FROM EMP
```

1. Arbre Algébrique de la requête:

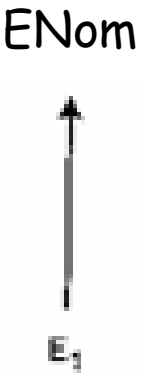


Schema Fragmentation

$$E_1 = \Pi_{ENO,ENOM}(EMP)$$
$$E_2 = \Pi_{ENO,ETITRE}(EMP)$$



2.Arbre générique



3.Arbre final

Réduction (Cas de FHybride)

1. Supprimer les relations vides générées par des sélections contradictoires sur des fragments horizontaux.
2. Supprimer les relations inutiles générées par les projections sur des fragments verticaux.
3. Distribuer les jointures sur les unions afin d'isoler et supprimer les jointures inutiles.

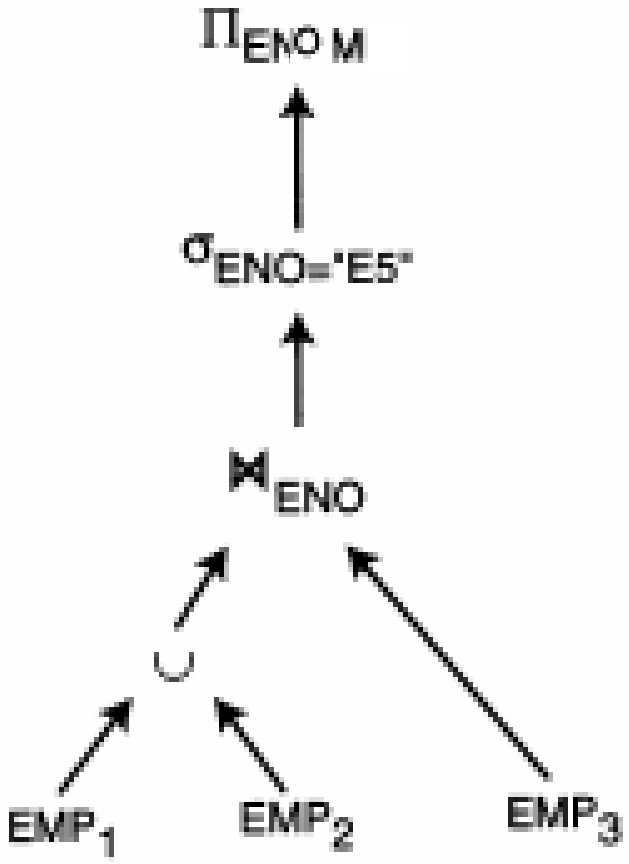
$$EMP_1 = \sigma_{ENO \leq "E4"}(\Pi_{ENO,ENOM}(EMP))$$

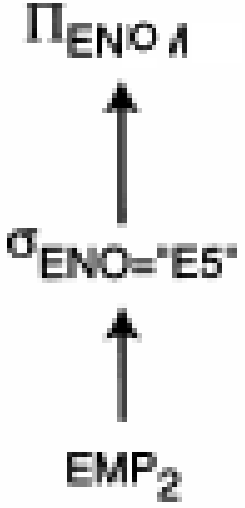
$$EMP_2 = \sigma_{ENO > "E4"}(\Pi_{ENO,ENOM}(EMP))$$

$$EMP_3 = \Pi_{ENO,TITRE}(EMP)$$

```
SELECT ENOM
FROM EMP
WHERE ENO="E5";
```

Réduction (Cas de FHybride)





Jointures distribuées

La jointure est un aspect important dans un SGBD centralisé, et elle est encore plus importante dans une BDR puisque les jointures entre des fragments stockés sur des sites différents peuvent augmenter le temps de communication.

Deux approches existent:

- Optimiser l'ordre des jointures : INGRES et INGRES distribué et Système R et Système R*

- Remplacer les jointures par des combinaisons de semijointures afin de minimiser les coûts de communication : Hill Climbing et SDD-1.

Jointures distribuées

a. Ordre de jointure l'ordre de jointure de deux relation / fragments situés à différents des sites consiste à déplacer la plus petite relation vers l'autre site, alors nous devons estimer la taille de R et S

Définir l'ordre direct de jointure des requêtes impliquant plus de deux relations est plus complexe

Jointures distribuées

Exemple : La requête peut être évaluée dans au moins 5 différentes façons.

Plan 1:	**Plan 2:**	**Plan 3:**	**Plan 4:**	**Plan 5:**
EMP → Site 2 Site2 Emp'=EMP ⋈ Affec EMP' → Site 3 Site 3: Emp' ⋈ j	Affec → Site 1 Site 1: Emp'=Emp ⋈ affec EMP' → Site 3 Site 3: Emp' ⋈Proj	Affec Site 3 Site 3: AFFEC'=Affec ⋈ oj AFFE' → Site 1 Site 1: AFFEC' ⋈MP	Proj → Site 2 Site 2: Proj'=Proj ⋈ fec PROJ' → ite 1 Site 1: PROJ' ⋈MP	EMP Site 2 PROJ Site 2 Site 2: EMP ⋈ PROJ ⋈ AFFEC

Pour choisir un plan, beaucoup d'informations sont nécessaires, y compris

- Taille (EMP), Taille (Affec), Taille (PROJ)
- Taille (EMP ⋈ Affec),T (Affec ⋈ PROJ)
- Possibilités d'exécution parallèle si le temps de réponse est utilisé

Jointures distribuées

b. Utilisation de la semi jointure

Exemple Supposons par exemple que nous souhaitions évaluer l'expression de jointure R1 ⋈ R2 sur le site S2, où R1 et R2 sont des fragments stockés respectivement sur les sites S1 et S2, R1 et R2 sont définis respectivement sur les attributs $A = (x, a1, a2, \ldots, an)$ et $B = (x, b1, b2, \ldots, bm)$. Nous pouvons changer cela de manière à utiliser plutôt l'opération de semi-jointure.

Nous pouvons dès lors évaluer l'opération de jointure comme suit :

1) Évaluer R' = Π x (R2) dans S2 (ne nécessite que les attributs de jointure du site S1).

2) Transférer R' au site S1.

3) Évaluer R" = R1 ⋈ R' au site S1.

4) Transférer R" dans le site S2.

5) Évaluer R" ⋈ R2 au site S2.

Optimisation

Optimisation Globale cette couche prend en considération des informations statistiques pour trouver un plan d'éxécution proche de l'optimum. Le résultat de cette couche est une stratégie d'éxécution basée sur des fragments, où viennent s'ajouter des primitives de communication qui envoient les parties de la requête aux SGBD locaux pour qu'ils les exécutent, et qui permettent ensuite de recevoir les résultats

Optimisation Locale tandis que les trois premières couches s'éxécutent sur le site de contrôle (généralement le site qui a lancé la requête), cette couche particulière s'éxécute sue chacun des sites locaux impliqués dans la requête, Chaque SGBD local effectue ses propres optimisation.