Evaluation des requêtes

Contenu de cette partie :

- **Introduction**
- **Algorithmes pour l'évaluation d'une requête**
- **Optimisation d'une requête**

Optimisation

SQL est **déclaratif**. L'utilisateur :

- indique ce qu'il veut obtenir.
- n'indique pas **comment** l'obtenir.

Donc le système fait le reste :

- comprendre la requête: il traduit la requête en algèbre relationnelle
- Choisir la meilleure stratégie d'évaluation. On obtient un plan d'exécution physique
- Evaluer le plan d'exécution

L'optimisation sur un exemple

Considérons le schéma :

$CINEMA(Cinéma, Adresse, Gérant)$
$SALLE(Cinéma, NoSalle, Capacité)$

et la requête :

$$Adresse\ des\ cinémas\ ayant\ des\ salles\ de\ plus\ de\ 150\ places$$

```
SELECT Adresse
FROM   CINEMA, SALLE
WHERE  capacité > 150
AND    CINEMA.cinéma = SALLE.cinéma
```

En algèbre relationnelle

Traduit en algèbre, on a plusieurs possibilités. En voici deux :

1. $\pi_{Cinéma}(\sigma_{Capacité>150}(CINEMA \bowtie SALLE))$
2. $\pi_{Cinéma}(CINEMA \bowtie \sigma_{Capacité>150}(SALLE))$

La deuxième est meilleure.

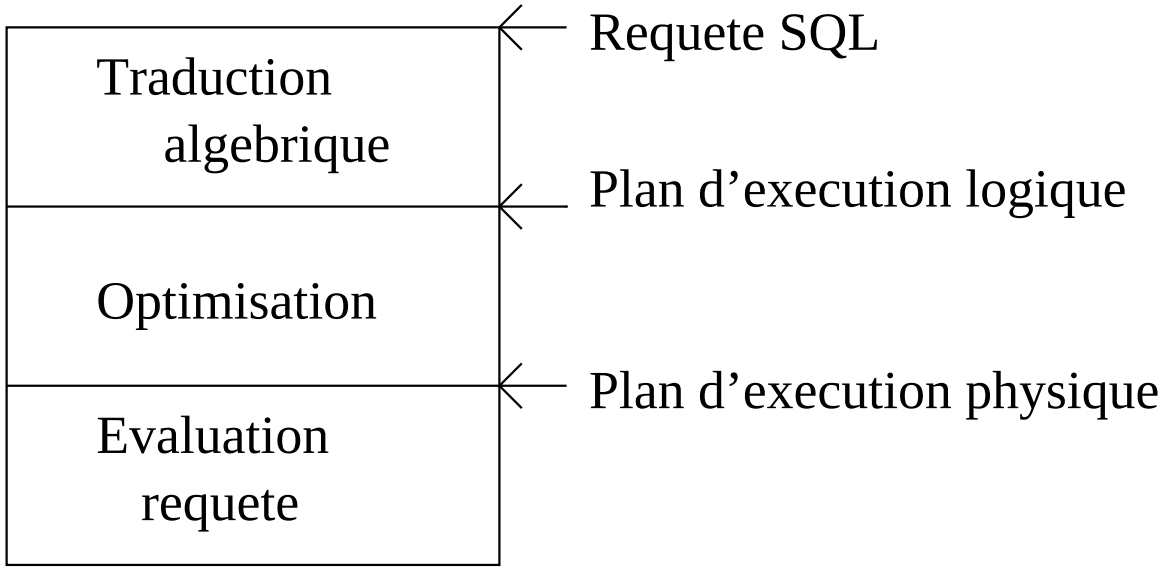
Choix d’un plan d’exécution physique

Choix d’un ensemble d’opérateurs physiques (en gras ci-dessous) et choix d’un ordre d’exécution

- **Accès séquentiel** de SALLE : faire la sélection sur SALLE et projeter sur Cinéma
- **trier** le résultat
- **Accès séquentiel** de CINEMA : projeter la table CINEMA sur Cinéma
- **trier** le résultat
- **Fusionner** les deux listes triées de cinémas

Il ne reste qu’à évaluer ce plan.

Les étapes de traitement d'une requête



Evaluation de requête efficace

Minimiser le temps

- **d'évaluation** : temps pour traiter entièrement la requête
- **de réponse** : temps pour donner à l'utilisateur le 1er nuplet

on s'intéresse au temps d'évaluation.

Mesure du temps

Temps d'évaluation = fonction du temps d'évaluation des opérations élémentaires (tri, fusion, accès séquentiel, etc.)

Temps d'évaluation d'une opération= nombre de pages lues/écrites;
on ne compte pas l'écriture du résultat qui ne dépend que

- de la taille du résultat
- mais pas de l'algorithme choisi

Evaluation du plan d'exécution physique (PEP)

Le résultat de l'optimisation est un PEP qu'il faut évaluer.

On utilise le pipelinage entre opérations successives du plan: on n'attend pas qu'une opération soit terminée avant de commencer la suivante sauf pour le tri. Intérêts:

- Besoin de moins de mémoire: pas besoin de stocker le résultat intermédiaire
- temps de réponse meilleur

On regarde les algorithmes pour implanter chacune des opérations du PEP

Algorithmes pour les principales opérations physiques

- tri
- sélection
- projection (on fait l'impasse par manque de temps)
- jointure

TRI

Le tri est nécessaire pour

- éliminer des doublons (projection)
- fonctions agrégat et order by
- algorithme de jointure

Les 2 phases du tri

On a M tampons en mémoire, de taille un bloc. Soit B le nombre de blocs de la relation.

- tri: on lit la relation M blocs par M blocs. On trie les Mblocs et on les écrit sur disque. On obtient $\lceil B/M \rceil$ partitions triées.
- on fusionne les partitions jusqu‘a ce qu’on obtienne une seule partition triée

C’est la 2e phase qui coûte cher: $Blog_{M-1}B$

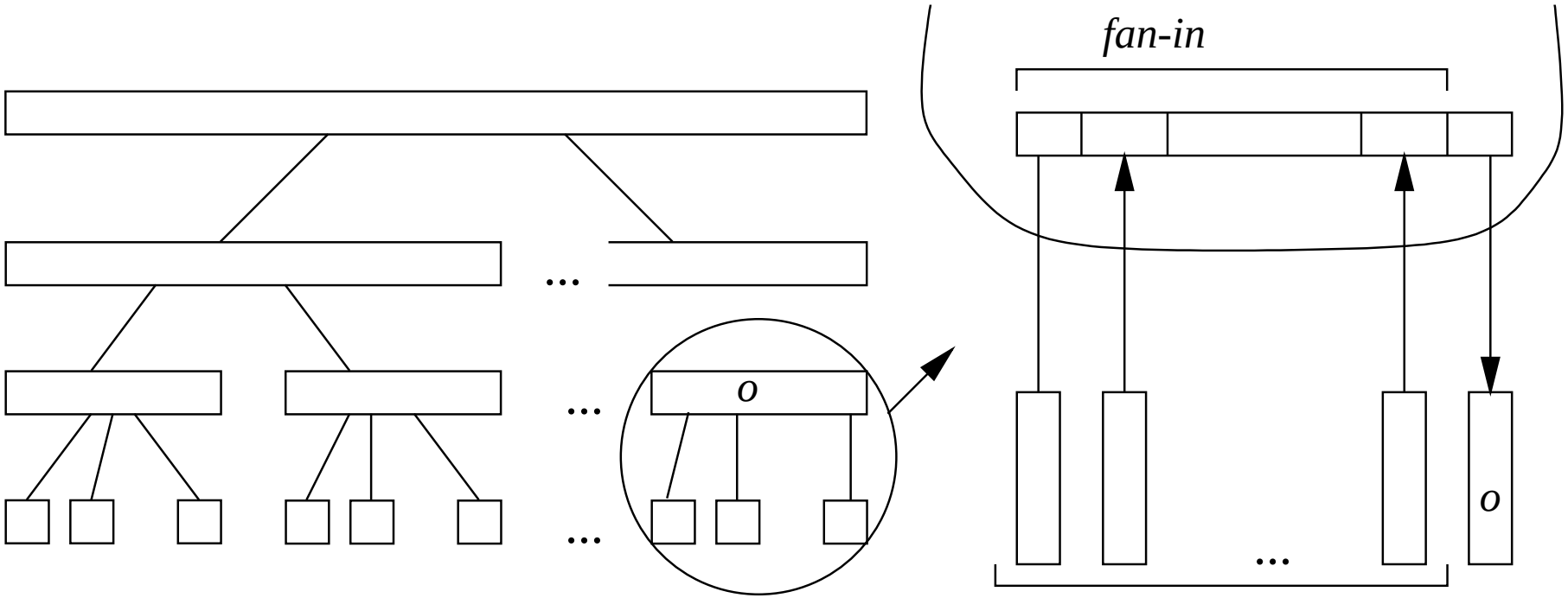
phase de fusion dans le tri

A chaque étape

- on fusionne les partitions $M-1$ par $M-1$
- on se sert du Me tampon pour le résultat
- on obtient $M-1$ fois moins de partitions triées $M-1$ fois plus grandes.
- on lit et on écrit B blocs

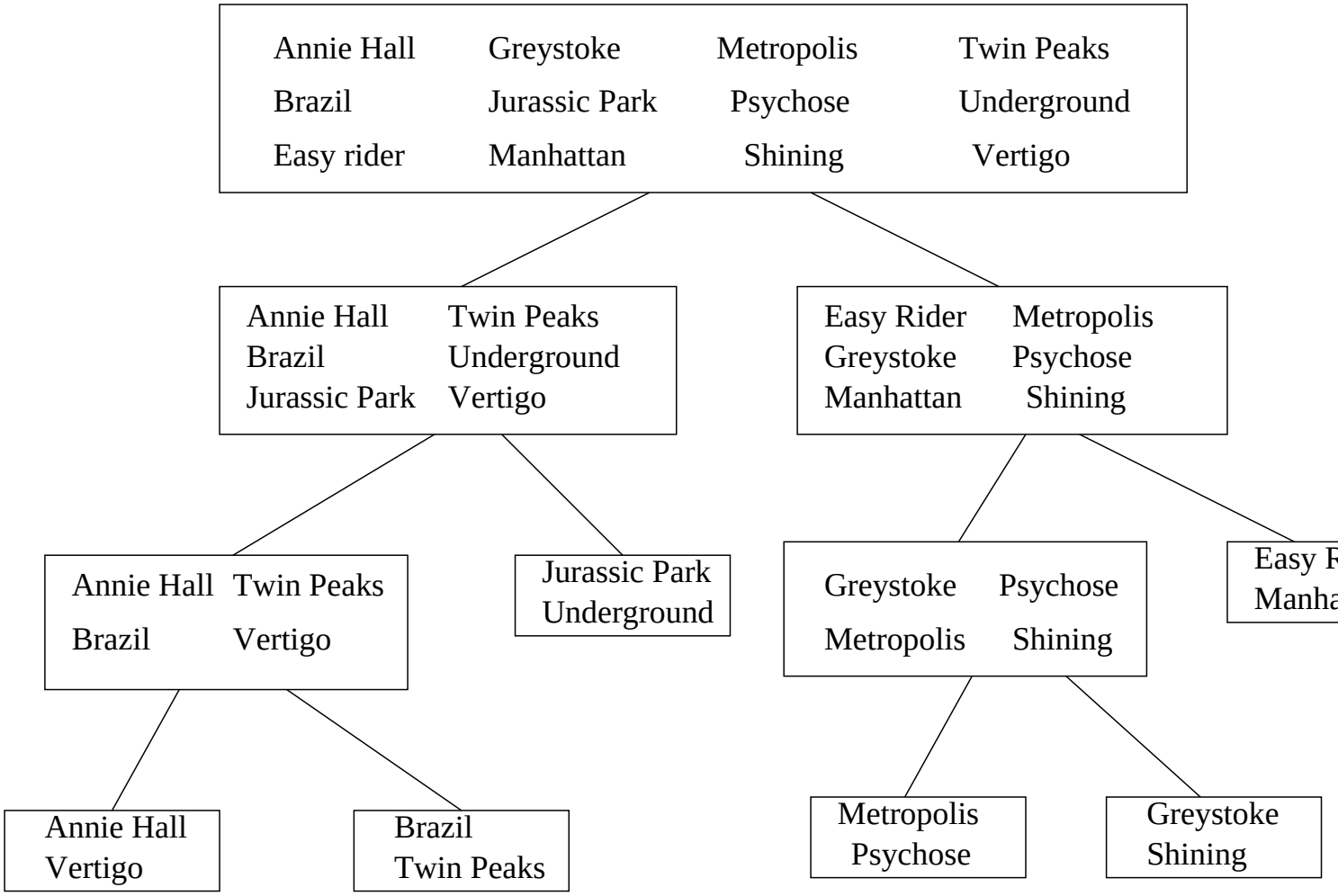
Le nombre d'étapes est $log_{M-1}B$

phase de fusion



tri: un exemple

Vertigo	Shining
Annie Hall	Greystoke
Twin Peaks	Underground
Brazil	Jurassic Park
Psychose	Manhattan
Metropolis	Easy rider



algorithmes pour la sélection

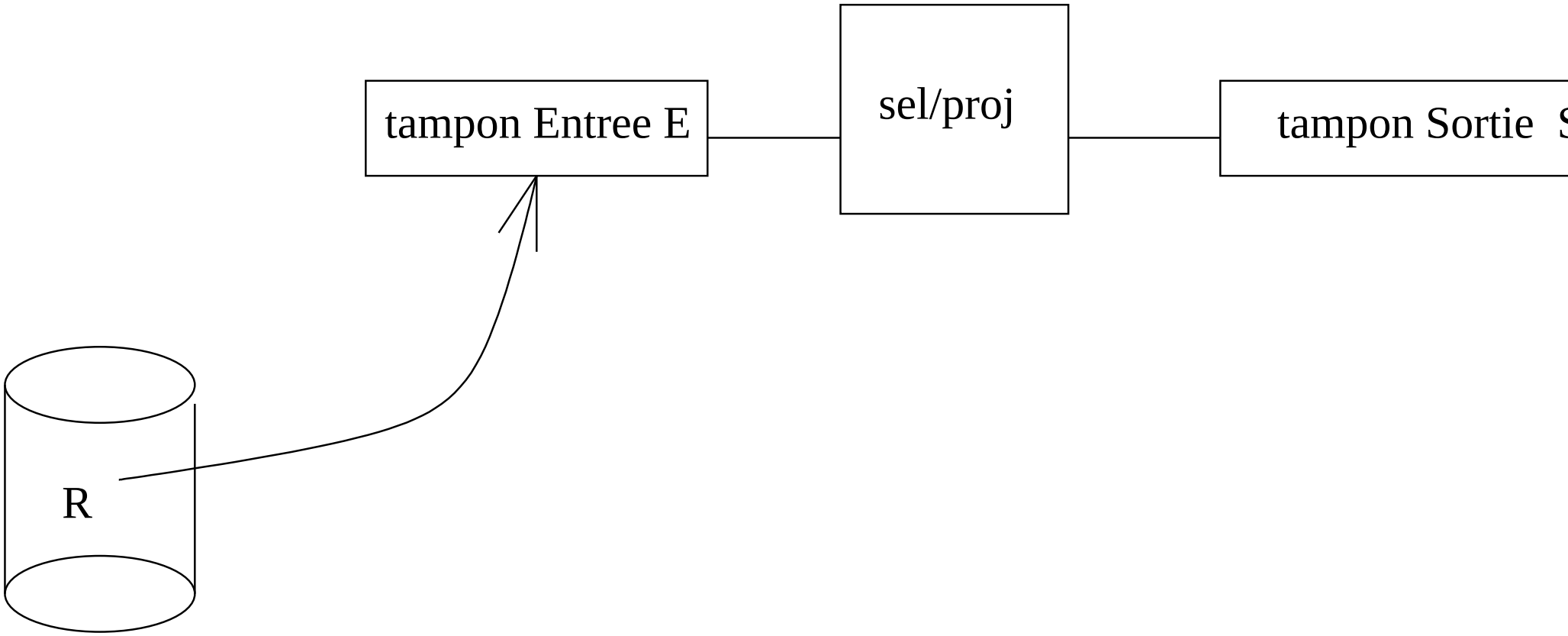
On s'intéresse aux algorithmes pour faire simultanément une conjonction de sélections et une projection. Il y a deux algorithmes:

- par balayage séquentiel
- par traversée d'index

sélection par accès séquentiel

- un tampon de lecture et un tampon d'écriture,
- on parcourt tout le fichier : pour chaque bloc lu, on fait les sélections/projections,
- quand le tampon de sortie est plein on l'écrit sur disque.

Coût: B lectures

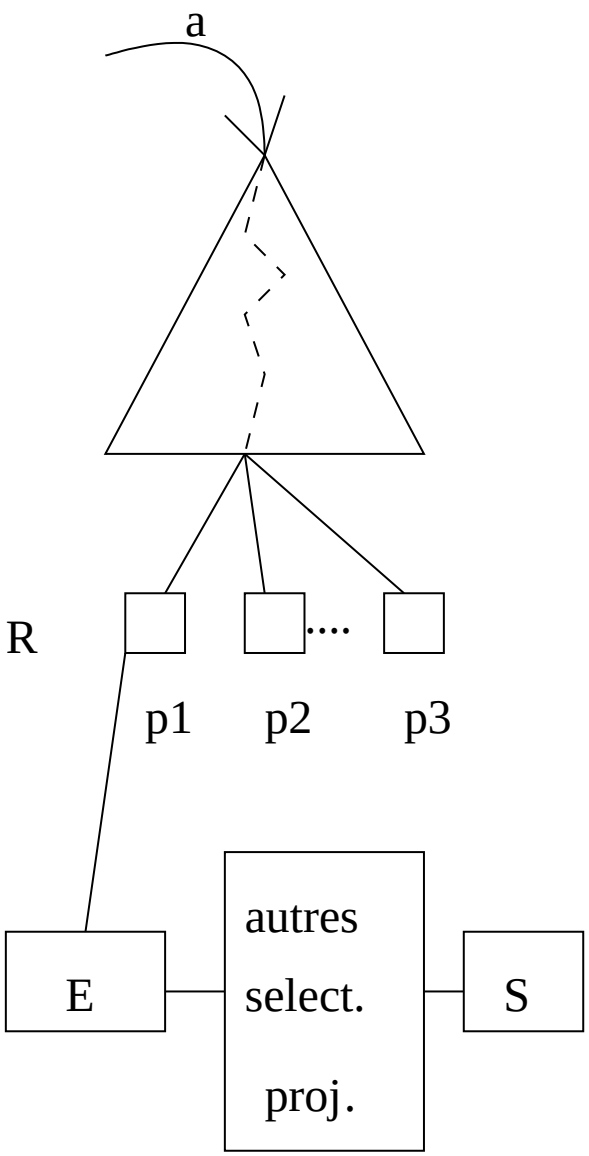

tampon Entree E
sel/proj
tampon Sortie S
R

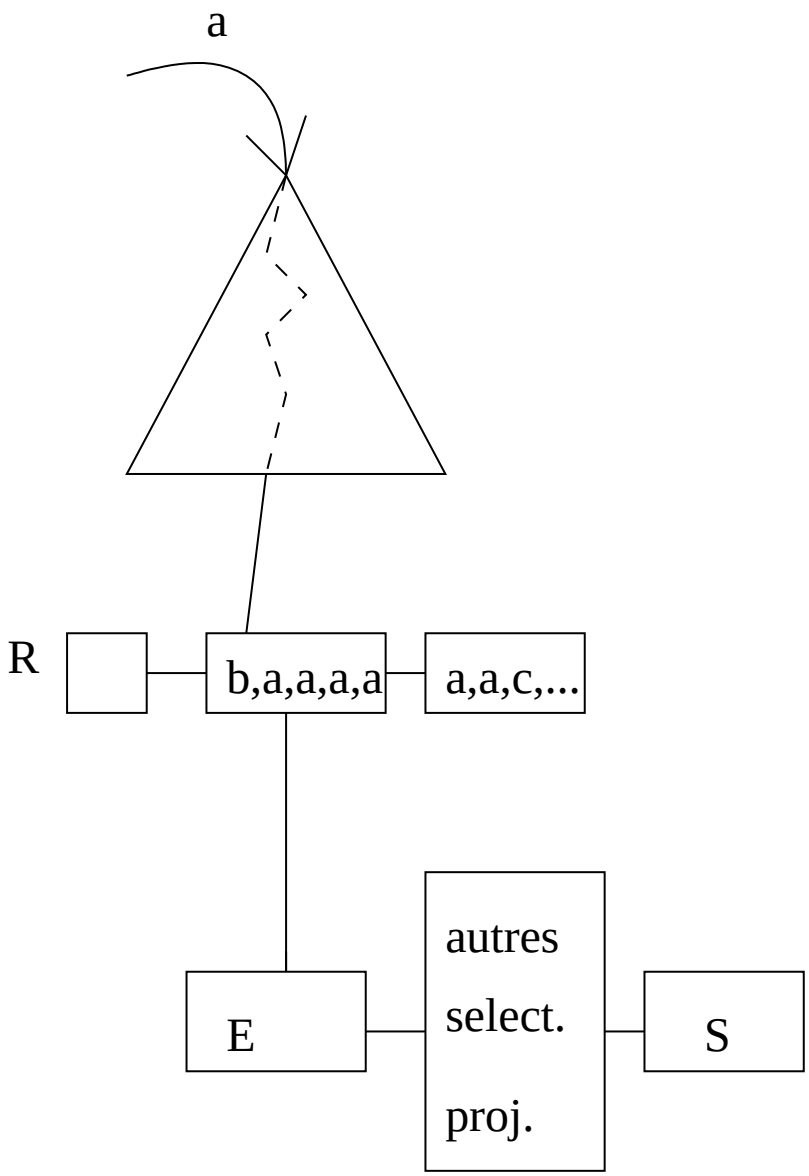
Sélection par traversée d'index

Dans le cas où la table R est indexée sur A qui apparait dans un des critères de sélection, $A = a$, 2 étapes:

- $TRAV(I, a)$ traverse l'index I avec pour clé a et donne les adresses de nuplets telles que $A = a$
- pour chaque adr dans $TRAV(I, a)$
  - **Accès par adresse** au nuplet n de R d'adresse adr
  - faire les autres sélections et les projections éventuelles et copier le nuplet résultant s'il y a lieu dans le tampon de sortie

Traversée d’index


a
R
....
p1
p2
p3
E
autres
select.
proj.
S


a
R
b,a,a,a,a
a,a,c,...
E
autres
select.
proj.
S

Estimation du Coût: $I + N \times S$,

- I est le coût de traversée d'index
- N est le nombre de nuplets
- dans le cas d'un index dense, la sélectivité S d'un attribut $A = 1/|\pi_A(R)|$

Algorithmes de jointure

Les algorithmes les plus utilisés sont:

- Boucles imbriquées
- Tri-fusion (aucune des 2 tables n'est indexée sur l'attribut de jointure)
- Jointure par hachage

On ne regardera que les deux premiers.

Certains algorithmes ne marchent que pour une équijointure

Jointure par boucles imbriquées

- Le plus simple mais cher : $B_R \times B_S$.
- Utilisé quand l'une des tables est petite.
- 2 tampons en lecture, un T en écriture.
- S'adapte aux jointures avec inégalité.

On lit tous les blocs d'une relation R *directrice*. Pour chaque bloc lu, on lit **tous** les blocs de l'autre relation S. Pour chaque couple de blocs lus, on fait la jointure des nuplets en mémoire (procédure BIM).

Jointure par boucles imbriquées(suite)

```
Pour chaque p dans pages(R) {
     lire p
     Pour chaque q dans pages(S) {
          lire q
          BIM(p,q)}
}
```

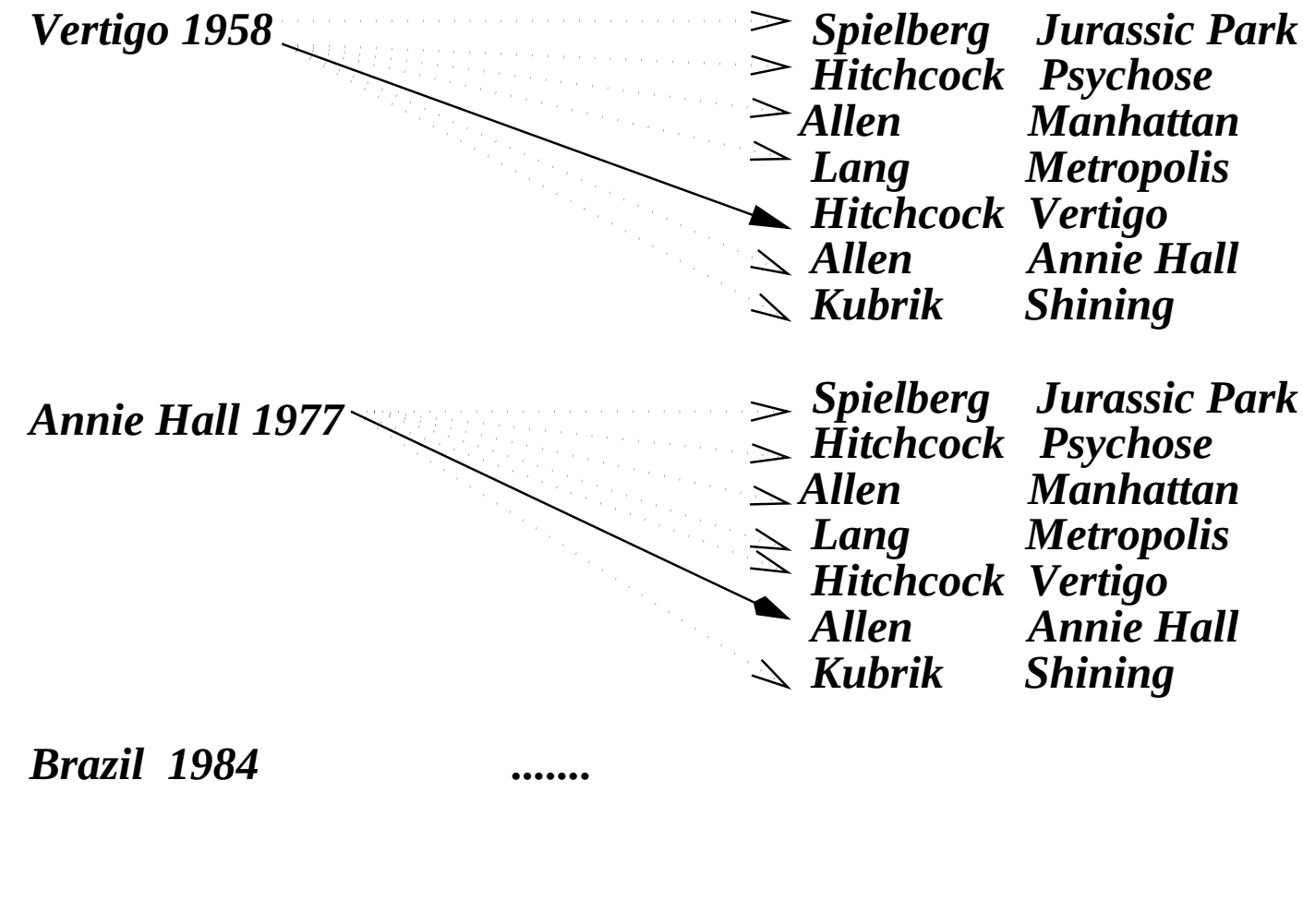
Algorithme BIS: jointure par boucles imbriquées simples

Jointure par boucles imbriquées(fin)

```
Pour chaque r dans R {
     Pour chaque s dans S {
          si r et s sont joignables pour donner t, alors
          si T est plein vider T sur disque,
          sinon ajouter t à T
     }
}
```

Procédure BIM (R,S): boucles imbriquées en mémoire

Boucles imbriquées: un exemple


Vertigo 1958
Spielberg Jurassic Park
Hitchcock Psychose
Allen Manhattan
Lang Metropolis
Hitchcock Vertigo
Allen Annie Hall
Kubrik Shining
Annie Hall 1977
Spielberg Jurassic Park
Hitchcock Psychose
Allen Manhattan
Lang Metropolis
Hitchcock Vertigo
Allen Annie Hall
Kubrik Shining
Brazil 1984 .....
Comparaison
Association

Boucles imbriquées et traversée d'index

Si l'une des deux tables S est indexée sur l'attribut de jointure, on utilise une variante efficace de coût $B_R \times O(log(B_S))$. Soit A l'attribut d ejointure de R.

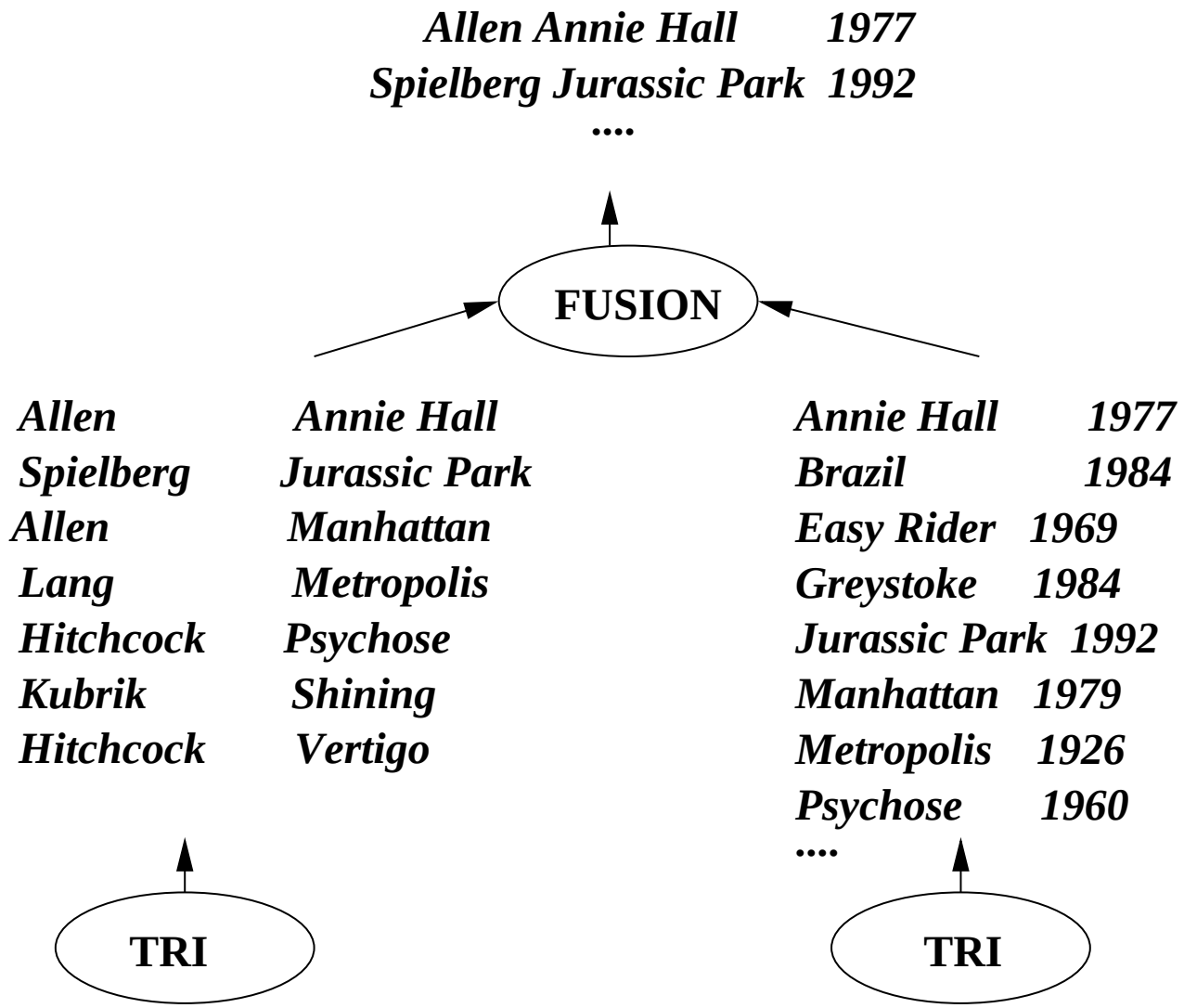
```
Pour chaque p dans pages(R) {
     lire p dans tampon E
     Pour chaque nuplet r dans p {
          adresses = TRAV (I,r.A)
          pour chaque a dans adresses {
               acces par adresse au nuplet s
               t= jointure de r et s
               si tampon resultat T plein {
                  vider T}
               ecrire t dans T}}}
```

Jointure par tri-fusion

Soit l'expression $\pi_{R.Ap,S.Bq}(R \bowtie_{A_i=B_j} S)$.

Projeter R sur $\{A_p, A_i\}$
Trier R sur A_i
Projeter S sur $\{B_q, B_j\}$
Trier S sur B_j
Fusionner les deux listes triées.

On les parcourt en parallèle en joignant les n-uplets ayant même valeur pour A_i et B_j.


Allen Annie Hall 1977
Spielberg Jurassic Park 1992
....
FUSION
Allen Annie Hall
Spielberg Jurassic Park
Allen Manhattan
Lang Metropolis
Hitchcock Psychose
Kubrik Shining
Hitchcock Vertigo
Annie Hall 1977
Brazil 1984
Easy Rider 1969
Greystoke 1984
Jurassic Park 1992
Manhattan 1979
Metropolis 1926
Psychose 1960
....
TRI
TRI
Fichiers Artistes
Fichier films

Tri-fusion : performance

Le coût est dominé par la phase de tri :

$$O(B_R \times \log(B_R) + B_S \times \log(B_S)).$$

Dans la seconde phase, un simple parcours en parallèle suffit.

Cet algorithme est particulièrement intéressant quand les données sont déjà triées en entrée.

Pour des grandes relations et en l'absence d'index, la jointure par tri-fusion présente les avantages suivants :

- **Efficacité** : bien meilleure que les boucles imbriquées.
- **Manipulation de données triées** : facilite l'élimination de dupliqués ou l'affichage ordonné.

Optimisation

- Une requête est décomposée en *blocs*
- un bloc a une seule clause select-from-where, une seule clause Group By, une seule clause having
- On se concentre sur l'optimisation d'un bloc

Toute requête ayant des imbrications peut être décomposée en une collection de blocs

Exemple de requête imbriquée

```
Employes (nom, departement, salaire)
Departement(dept-id, directeur)

SELECT directeur
FROM Departement,Employes
WHERE Departement.dept-id = Employes.departement
AND salaire = (SELECT MAX (salaire)
               FROM Employes)
```

et sa décomposition en blocs

```
SELECT directeur
FROM Departement,Employes
WHERE Departement.dept-id = Employes.departement
AND salaire = référence au bloc imbriqué
```

Requête sous FNC

- Toute requête peut être mise sous Forme normale conjonctive (FNC)
- Sratégies d'optimisation pour une disjonction $A = a \cup B = b$
  - Si index à la fois sur A et sur B alors traverser les index et faire l'union
  - sinon balayage séquentiel de la relation

Traduction en un PEL

La première étape de la compilation d'une requête comporte deux phases

- Analyse syntaxique
- Génération d'un plan d'exécution logique (PEL)

Analyse syntaxique

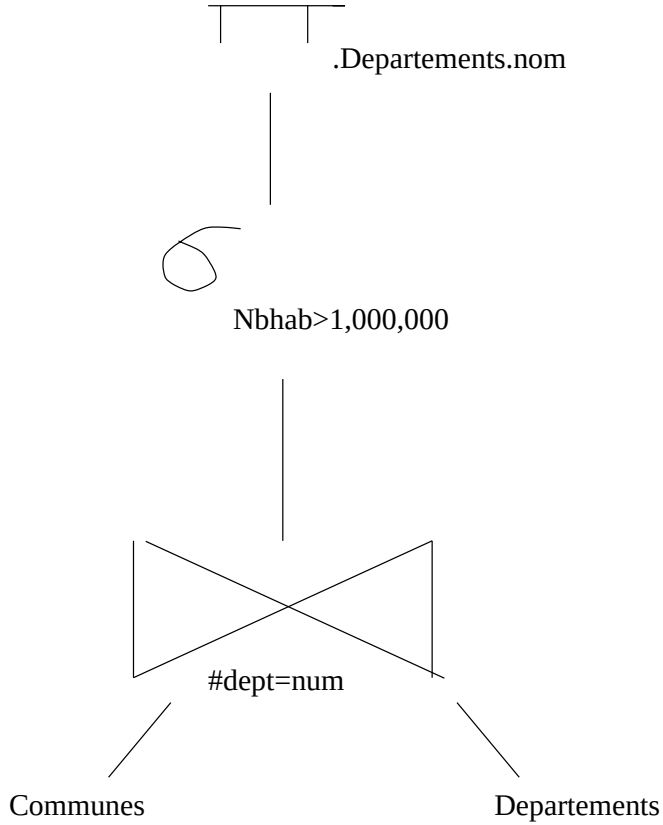
- Vérification de l'existence des relations et des attributs
- Détection de contradictions, comme $Nbhab = 1,000,000 \cap Nbhab = 500,000$
- Simplifications telles que: $(A \cup notB) \cap B)$ est remplacé par $A \cap B$.

Traduction en un PEL

Soit la requête

```
Select Dept.nom
From Departements, Communes
Where Nbhab > 1,000,000
And Communes.numdept=Departements.num
```

L’arbre suivant représentant une expression algébrique est un Plan d’exécution équivalent


.Departements.nom
Nbhab>1,000,000
#dept=num
Communes
Departements

Règles de réécriture

Les propriétés de l'algèbre permettent de transformer un PEL en un PEL équivalent:

1. **Commutativité des jointures** :
   $R \bowtie S \equiv S \bowtie R$
2. **Associativité des jointures** :
   $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
3. **Regroupement des sélections** :
   $\sigma_{A='a' \wedge B='b'}(R) \equiv \sigma_{A='a'}(\sigma_{B='b'}(R))$
4. **Commutativité de la sélection et de la projection**
   $\pi_{A_1,A_2,...A_p}(\sigma_{A_i='a}(R)) \equiv \sigma_{A_i='a}(\pi_{A_1,A_2,...A_p}(R)), i \in \{1,...,p\}$
5. **Commutativité de la sélection et de la jointure**.
   $\sigma_{A='a'}(R(...A...) \bowtie S) \equiv \sigma_{A='a'}(R) \bowtie S$
6. **Distributivité de la sélection sur l'union**.

$\sigma_{A='a'}(R \cup S) \equiv \sigma_{A='a'}(R) \cup \sigma_{A='a'}(S)$

NB : valable aussi pour la différence.

7. **Commutativité de la projection et de la jointure**

   $\pi_{A_1 \ldots A_p B_1 \ldots B_q}(R \bowtie_{A_i=B_j} S) \equiv$

   $\pi_{A_1 \ldots A_p}(R) \bowtie_{A_i=B_j} \pi_{B_1 \ldots B_q}(S))\ i \in \{1, .., p\}, j \in \{1, ..., q\}$

8. **Distributivité de la projection sur l'union**

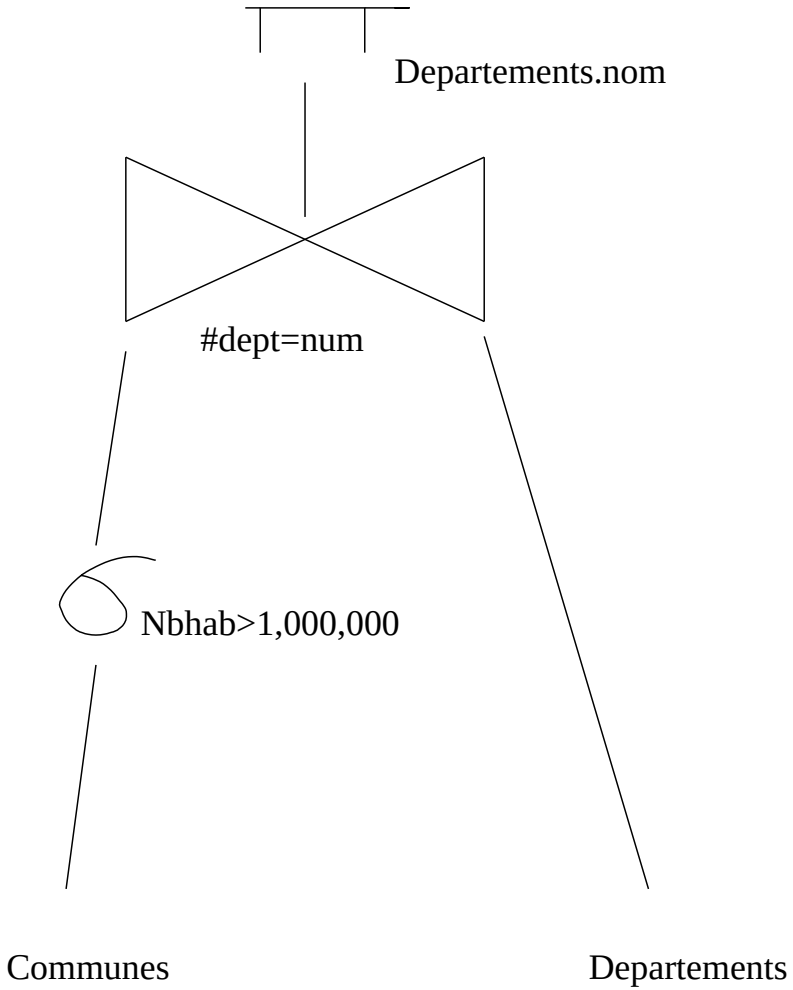
   $\pi_{A_1 A_2 \ldots A_p}(R \cup S) \equiv \pi_{A_1 A_2 \ldots A_p}(R) \cup \pi_{A_1 A_2 \ldots A_p}(S)$

Amélioration d'un PEL

On transforme grâce aux règles ci-dessus un PEL en un PEL "meilleur"

- faire les sélections, projections le plus tôt possible permet de réduire les tailles de relations et donc de réduire la complexité des opérations suivantes (règles 4,5 et 7),
- décomposer une sélection en une composition de sélections permet de tirer avantage de l'existence d'un index (règle 3),
- etc.

Le PEL ci-dessous est meilleur:



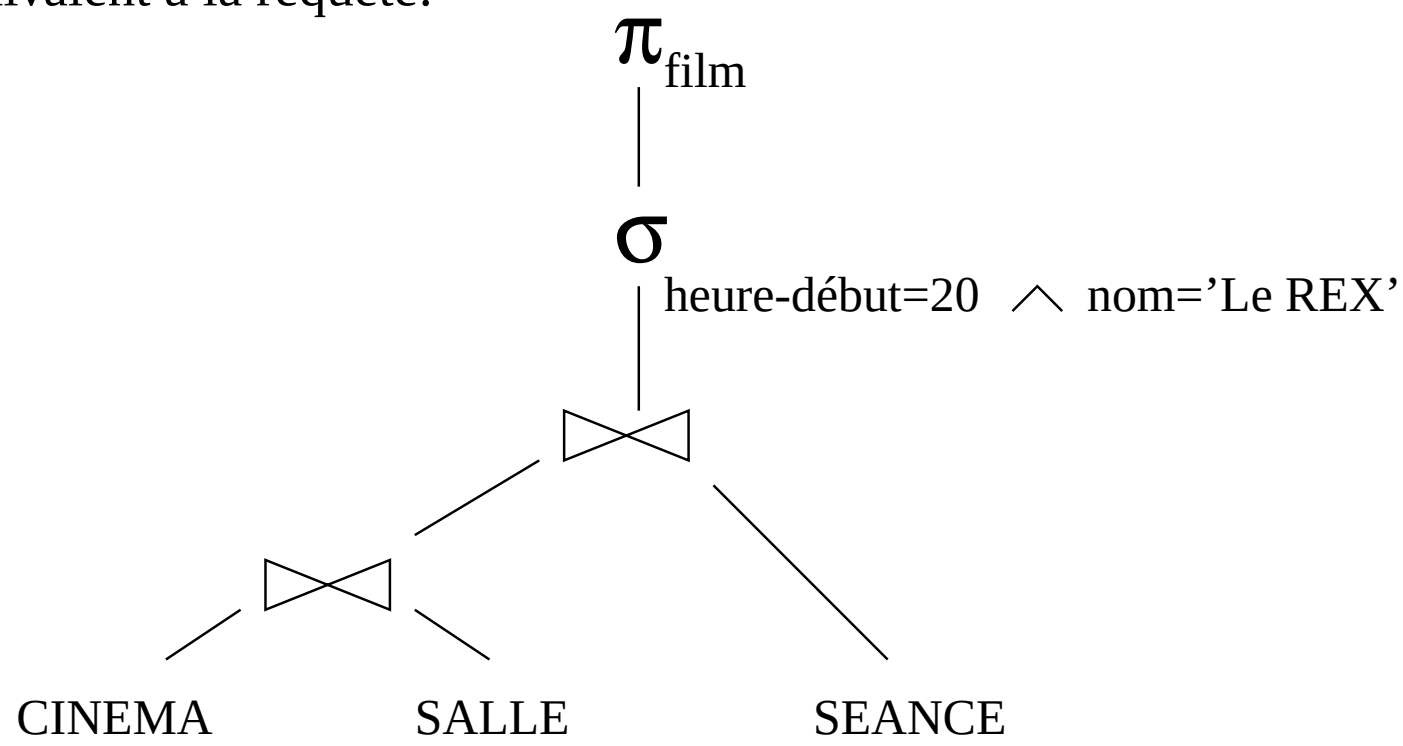
Un autre exemple:

```
Cinéma (nom-cinéma, adresse)
Salle (salle,nom-cinéma)
Séance (salle, film, heure-début)
```

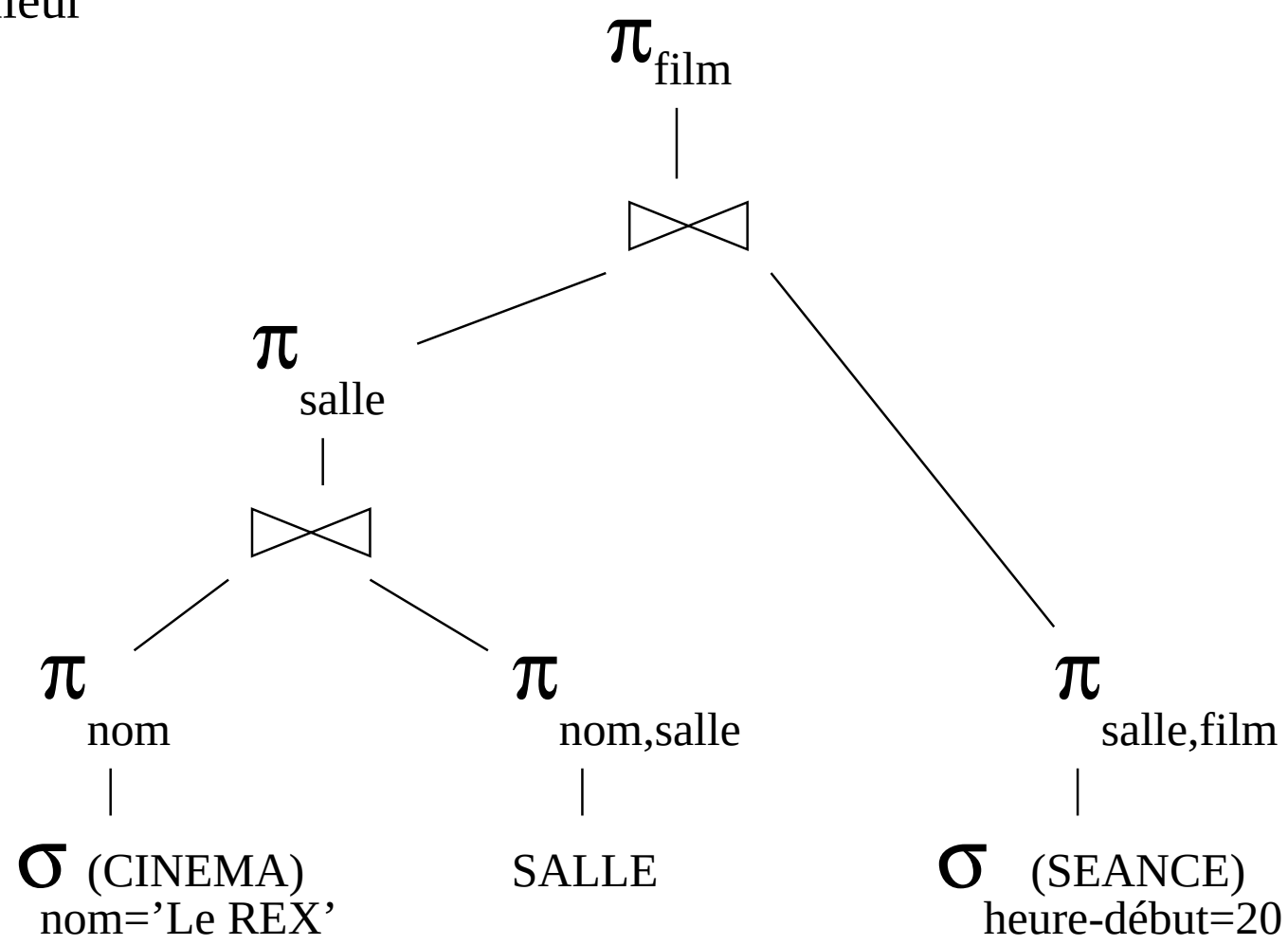
Soit la requête “Quels films passent au REX a 20 heures ?” exprimée par l’ordre SQL suivant.

```
SELECT film
FROM   Cinéma, Salle, Séance
WHERE  Cinéma.nom-cinéma = ’Le Rex’
AND    Séance.heure-début = 20
AND    Cinéma.nom-cinéma = Salle.nom-cinéma
AND    Salle.salle = Séance.salle
```

Un PEL équivalent à la requête:



Un PEL meilleur



Remarques:

- Sélection avant jointure est une bonne règle mais il peut y avoir des exceptions.
- Ce réarrangement des PEL est nécessaire mais pas suffisant: il faut ensuite choisir le meilleur algorithme pour chaque opération du PEL: le PEL est transformé en Plan d'exécution physique (PEP).

Estimation des coûts et choix d'un PEP

Un PEP contient

- les opérateurs physiques choisis pour évaluer la requête
- ainsi que l'ordre dans lequel exécuter la requête

Un PEP est représenté sous forme d'arbre:

- les feuilles sont les fichiers où sont stockés les tables et index
- les noeuds internes sont les opérations physiques,
- les arcs représentent les flots de données produits par une opération et consommés par l'opération suivante dans l'arbre

liste d'opérations physiques

CHEMINS D'ACCES

Sequentiel

TABLE

Parcours sequentiel

Adresse

TABLE

Acces par adresse

Attribut(s)

INDEX

Parcours d'index

OPERATIONS PHYSIQUES

Critere

Selection

Selection selon un critere

Attribut(s)

Tri

Tri sur un attribut

Critere

Fusion

Fusion de deux ensembles tries

Critere

Filtre

Filtre d'un ensemble en fonction d'un autre

Critere

Jointure

Jointure selon un critere

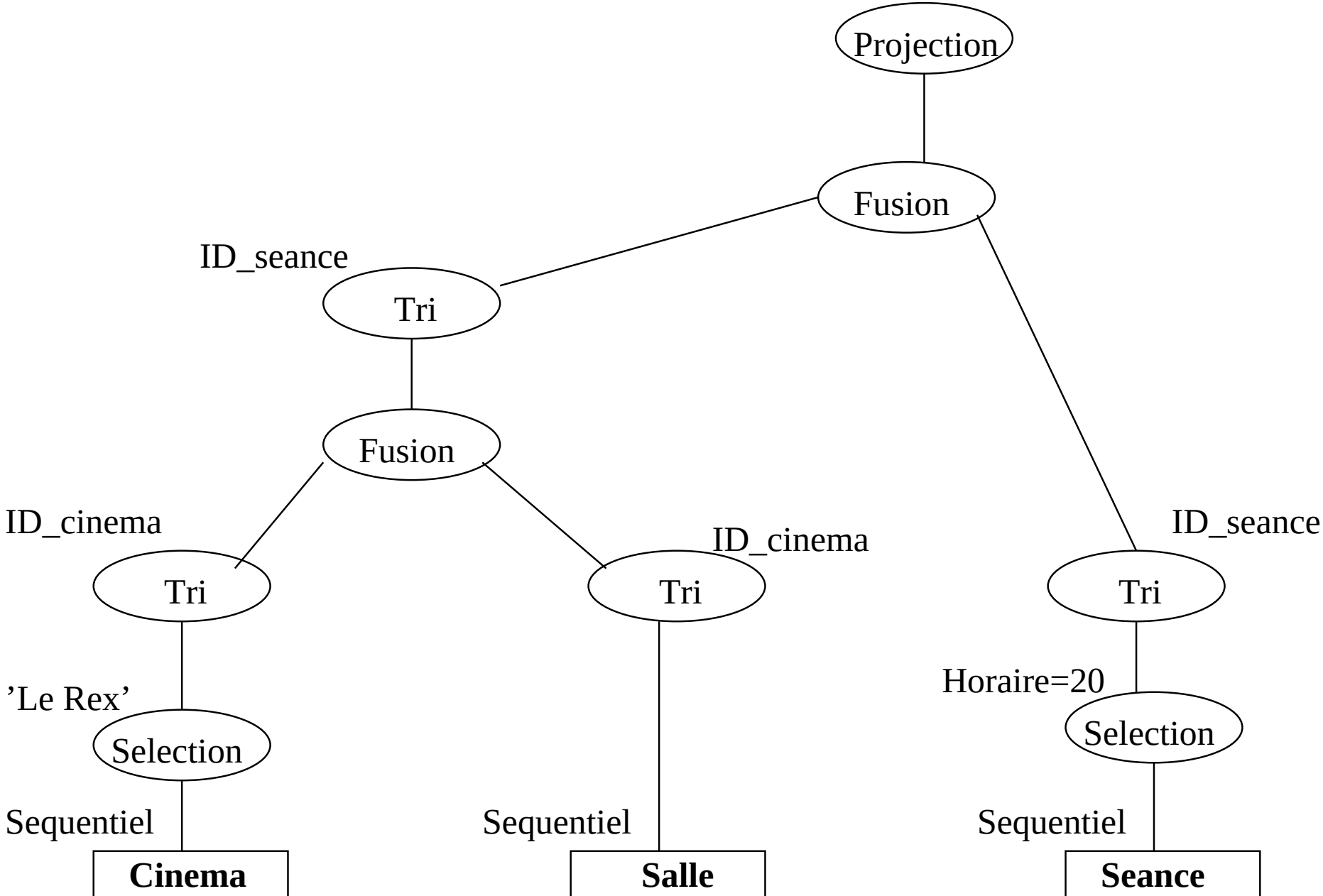
Attribut(s)

Projection

Projection sur des attributs

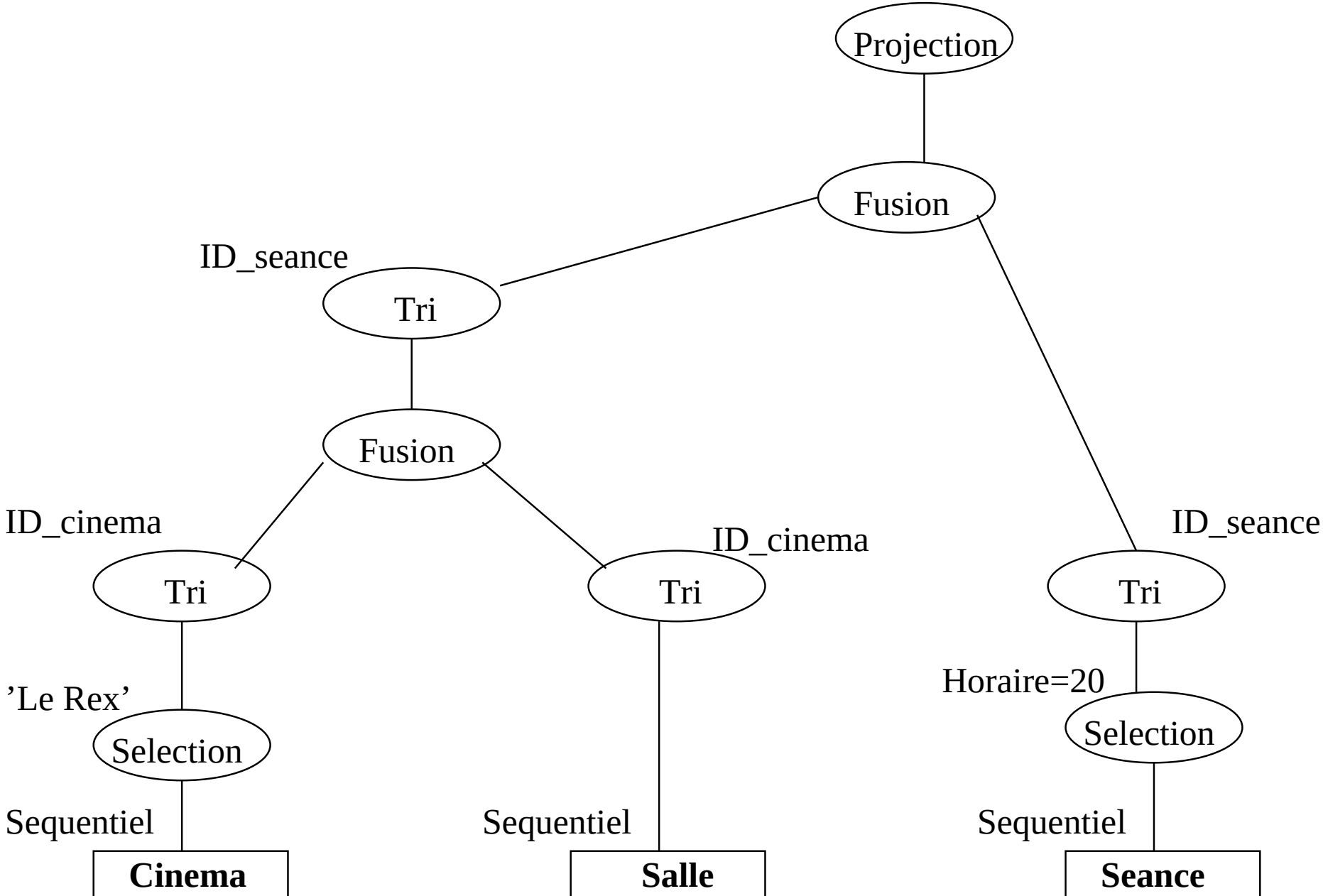
Un PEP

Pas d’index sur les attributs de jointure

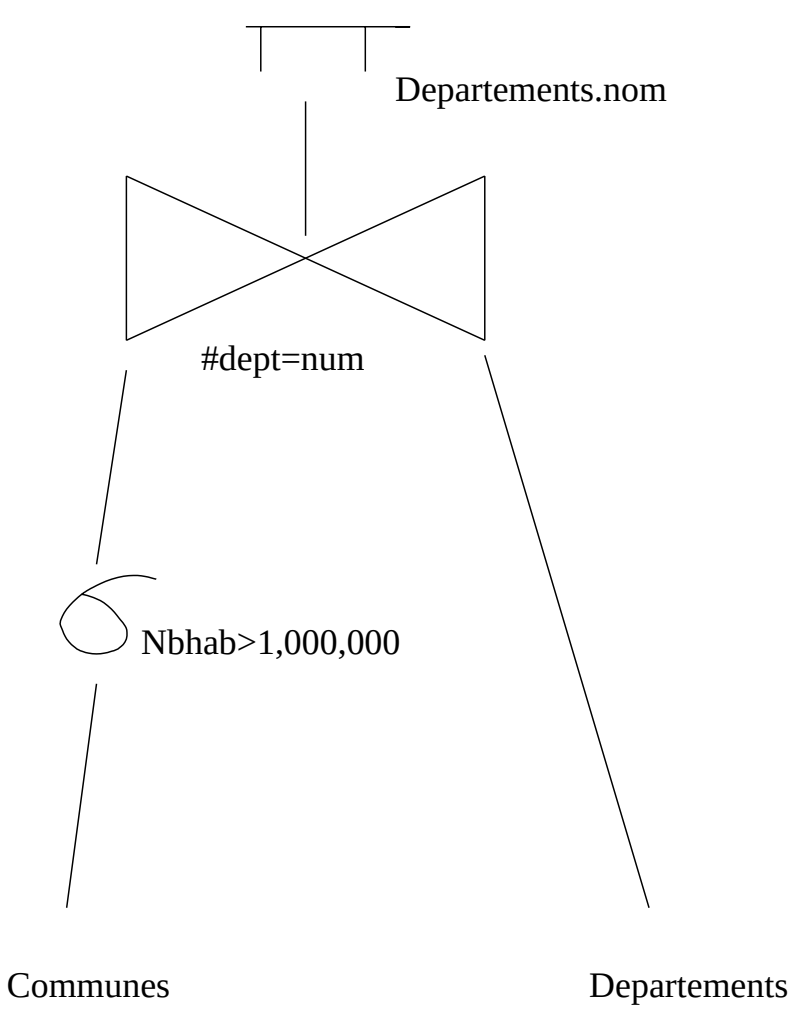

Projection
Fusion
ID_seance
Tri
Fusion
ID_cinema
ID_cinema
ID_seance
Tri
Tri
Tri
'Le Rex'
Horaire=20
Selection
Selection
Sequentiel
Sequentiel
Sequentiel
Cinema
Salle
Seance

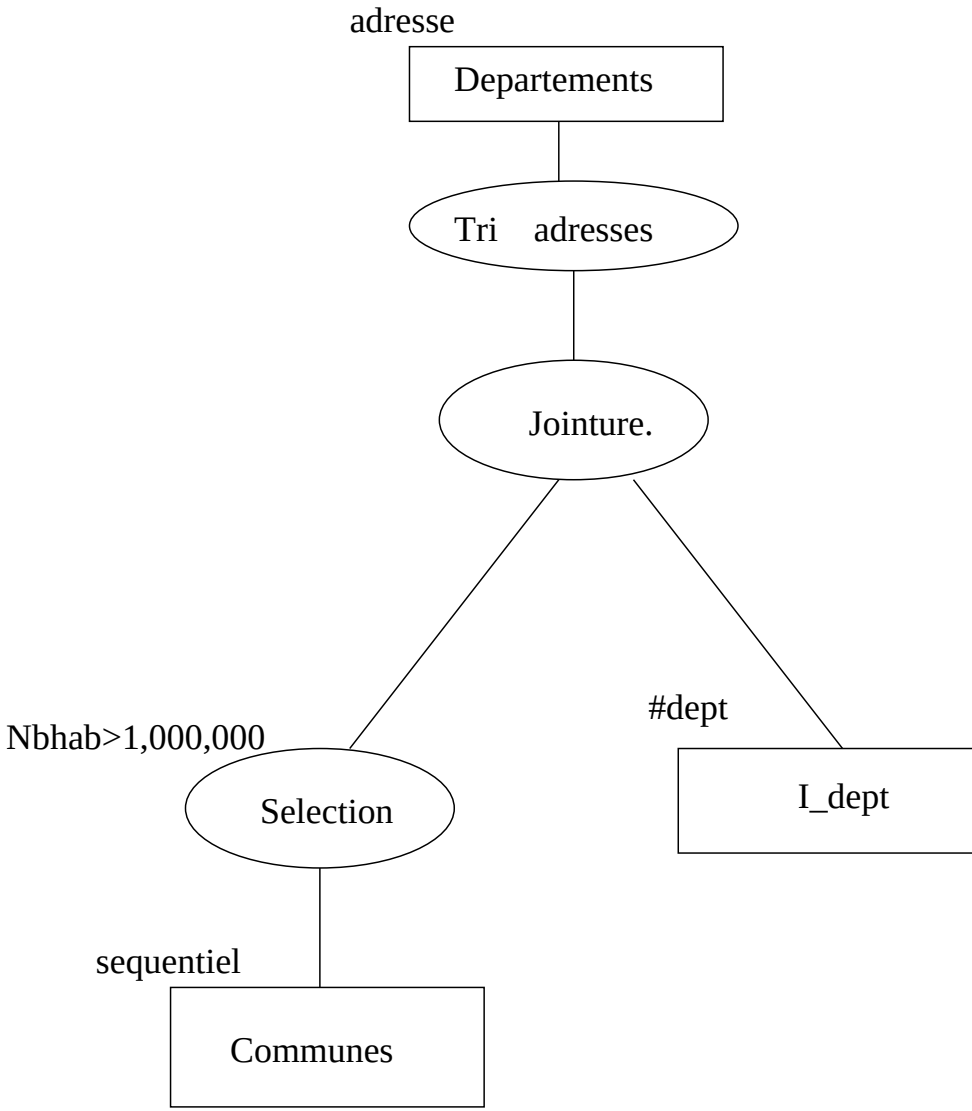
Un autre PEP

table indexée pour chaque jointure


Projection
Fusion
ID_seance
Tri
Fusion
ID_cinema
ID_cinema
ID_seance
Tri
Tri
Tri
'Le Rex'
Horaire=20
Selection
Selection
Sequentiel
Sequentiel
Sequentiel
Cinema
Salle
Seance

Un autre exemple


Departements.nom
#dept=num
Nbhab>1,000,000
Communes
Departements


adresse
Departements
Tri adresses
Jointure.
#dept
Nbhab>1,000,000
Selection
I_dept
sequentiel
Communes

Choix d'un PEP

Le choix d'un PEP consiste à

- choisir pour chaque noeud du PEL un ou une séquence d'opérateurs physiques
- répartir la mémoire disponible entre les différents opérateurs physiques et pipeliner les opérations

Il dépend:

- des chemins d'accès disponibles (index)
- des statistiques rassemblées sur les tables
- de la place disponible en mémoire

Estimation des coûts

Pour choisir entre deux opérateurs physiques, le système utilise une estimation simple des coûts qui s'appuie sur

- les statistiques disponibles
- l'estimation du coût de chaque chemin d'accès

Estimation du coût de la sélection

Prenons comme exemple l'estimation du coût de plusieurs stratégies de sélection sur A de la relation R, dans le cas où

- le critère de sélection utilise l'égalité
- il y a un index dense et non unique sur A

Le système connait:

- N le nombre de nuplets de R
- B, le nombre de blocs de R
- la sélectivité S de A

- Balayage séquentiel, estimation du coût: B
- Stratégie avec index, estimation du coût: $I + F + P$ où I est le coût de traversée d'index, $F = (N \times S)/(N/B) = B \times S$ est le nombre de feuilles en séquence à lire, et $P = N \times S$ est le nombre de nuplets qui satisfont le critère. En résumé le coût est $I + (B + N) \times S$.