

ENSTA

Mastère Spécialisé en Architecture des Systèmes d'Information

Cours C1-3

Systèmes de Gestion de Bases de Données (SGBD) relationnels

Maude Manouvrier

Partie II : les SGBD vus du côté Administrateur de Bases de Données

- Architecture générale d'un SGBD
- Organisation des données
- Évaluation et optimisation de requêtes
- Gestion de la concurrence / transactions
- Reprise sur pannes

BIBLIOGRAPHIE

Ouvrages de référence utilisés pour le cours :

R. Ramakrishnan et J. Gehrke, *Database Management Systems*, Second Edition; McGraw-Hill, 2000, disponible à la BU 055.7 RAM

H. Garcia Molina, J.D. Ullman et J. Widom, *Database System Implementation*, Prentice Hall, 2000, disponible à la BU 005.7 GAR

H. Garcia Molina, J.D. Ullman et J. Widom, *Database Systems - The Complete Book*, Prentice Hall, 2002

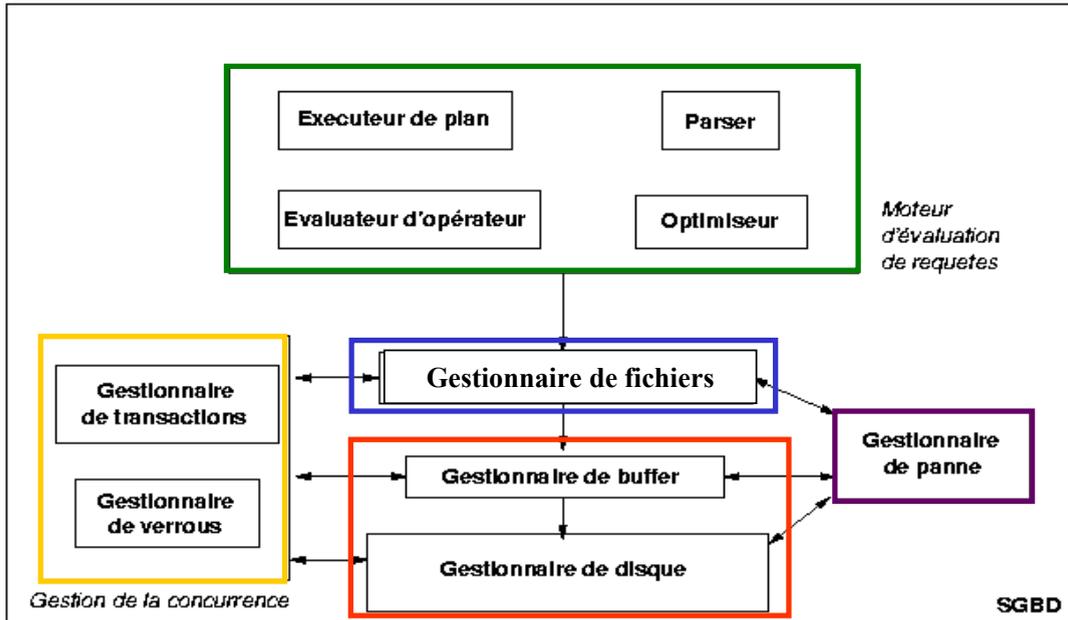
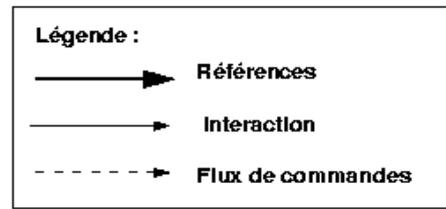
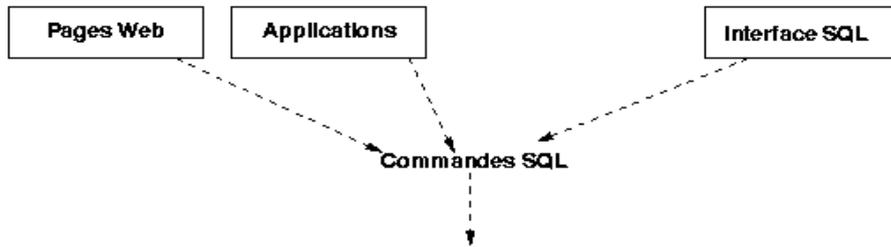
T. Connolly, C. Begg et A. Strachan, *Database Systems A Pratical Approach to Desigh, Implementation and Management*, 1998, disponible à la BU 055.7 CON

A. Silberschatz, H.F. Korth et S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2002, version de 1996 disponible à la BU 005.7 DAT

C.J. Date, *An Introduction aux bases de données*, 6ème édition, Thomson publishing, 1998, disponible à la BU 005.7 DAT

R.A. El Masri et S.B. Navathe, *Fundamentals of Database Systems*, Prentice Hall, disponible à la BU 005.7 ELM

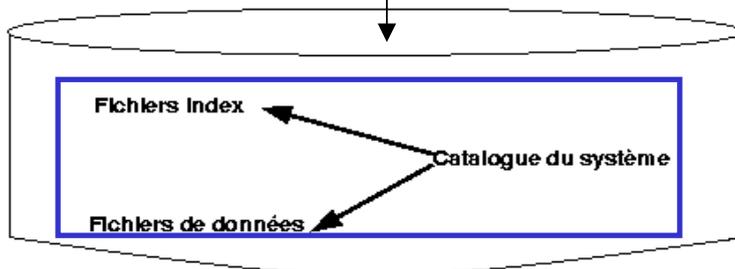
G. Gardarin, *Bases de Données - objet/relationnel*, Eyrolles, 1999, disponible à la BU 005.74 GAR + *Le client - serveur*, Eyrolles, 1996004.21 GAR



Chapitre 3

Chapitre 4

Chapitre 1 Chapitre 5



Chapitre 2

Chap. I - Architecture d'un SGBD

- Vision des données par le SGBD : un **ensemble d'enregistrements mémoire**
- Vision des données par le **gestionnaire de fichiers** : un **ensemble de pages mémoire**
- Vision des données par le **gestionnaire de disque** : un **ensemble de pages disque**
- Rôle du **gestionnaire de buffer** : passage des pages du disque vers la mémoire (et inversement)

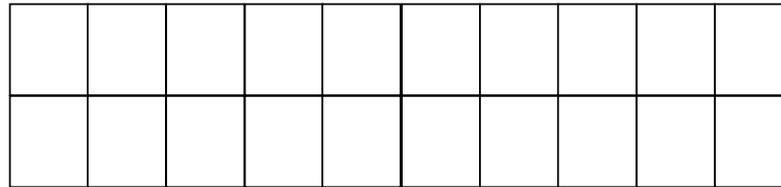
Gestionnaire de buffer

Rôle : placer, au moment voulu, une page du disque vers la mémoire et inversement

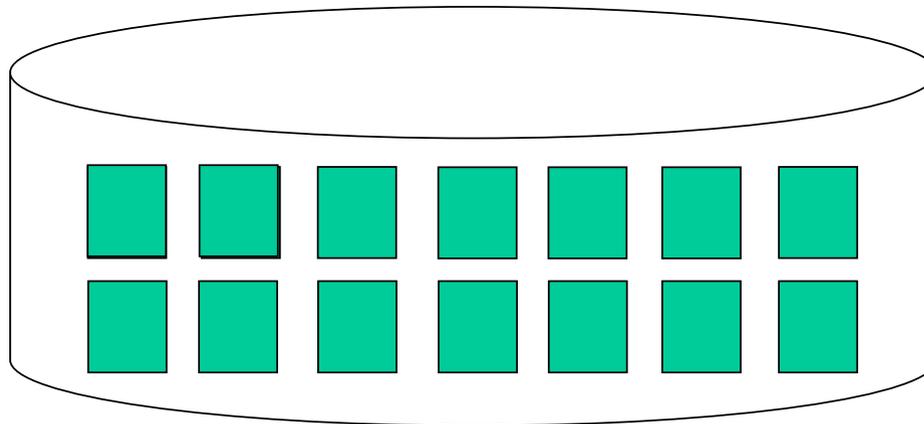
- Politique de remplacement (ex. LRU)
- Gestion des pages mises à jour
- Partition de la mémoire
- Vérification des droits sur les pages

Gestionnaire de buffer

Mémoire



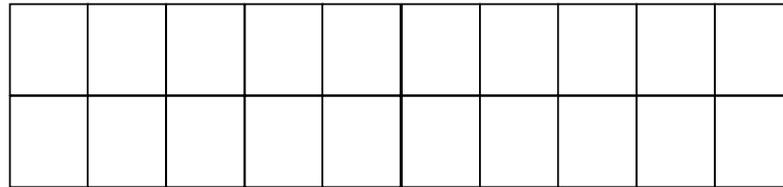
Disque



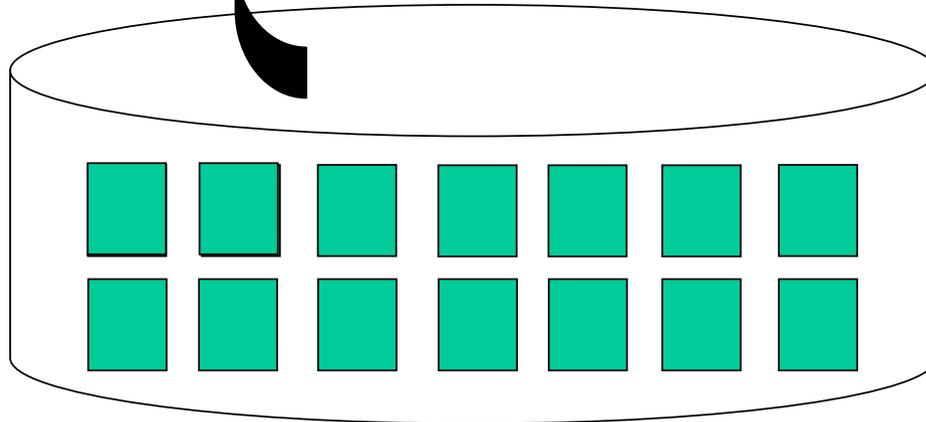
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

Mémoire



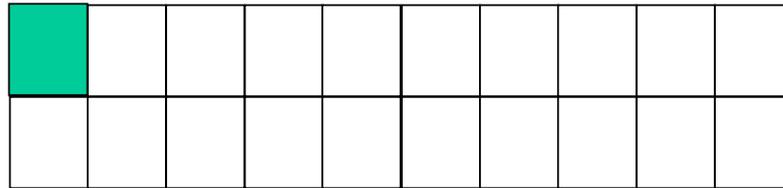
Disque



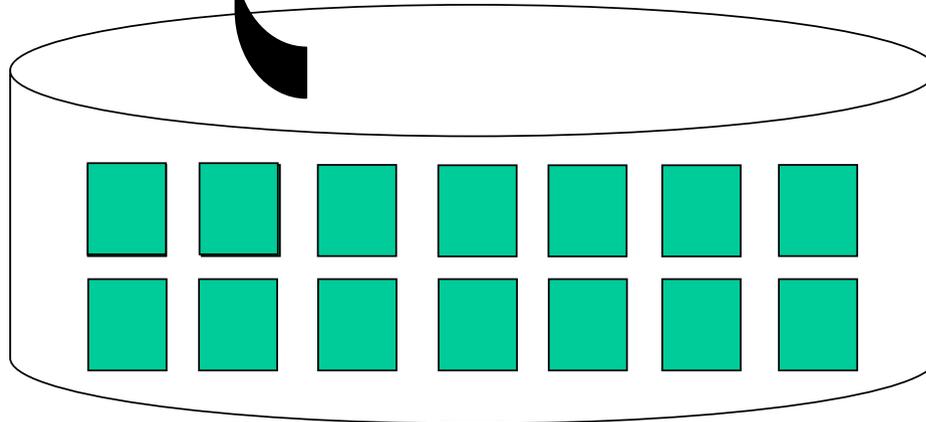
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

Mémoire



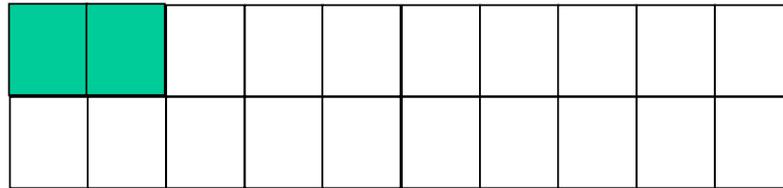
Disque



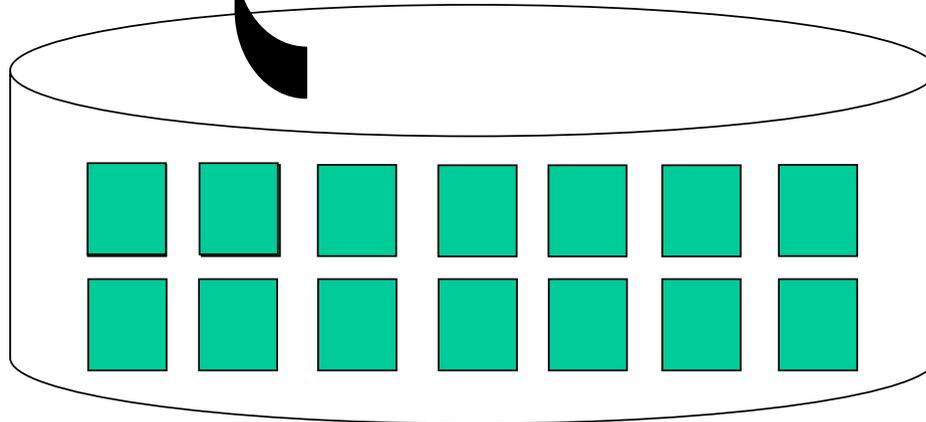
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

Mémoire



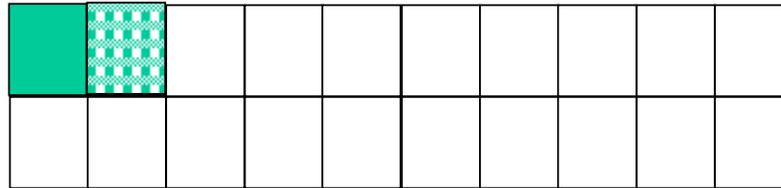
Disque



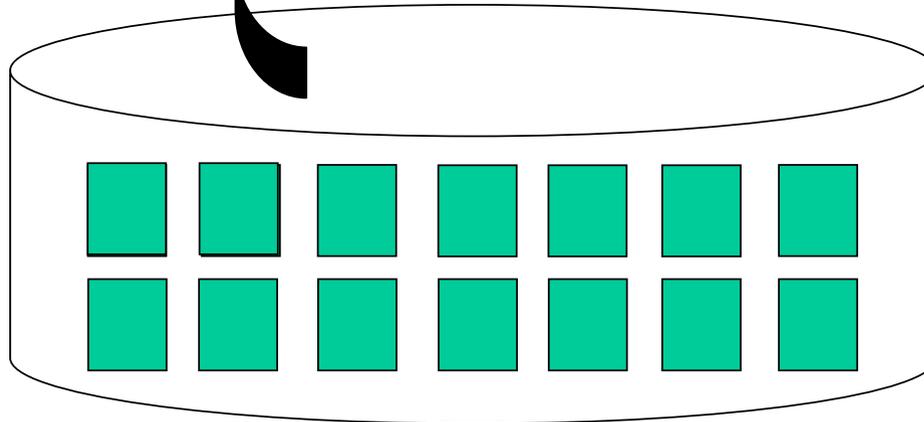
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

Mémoire

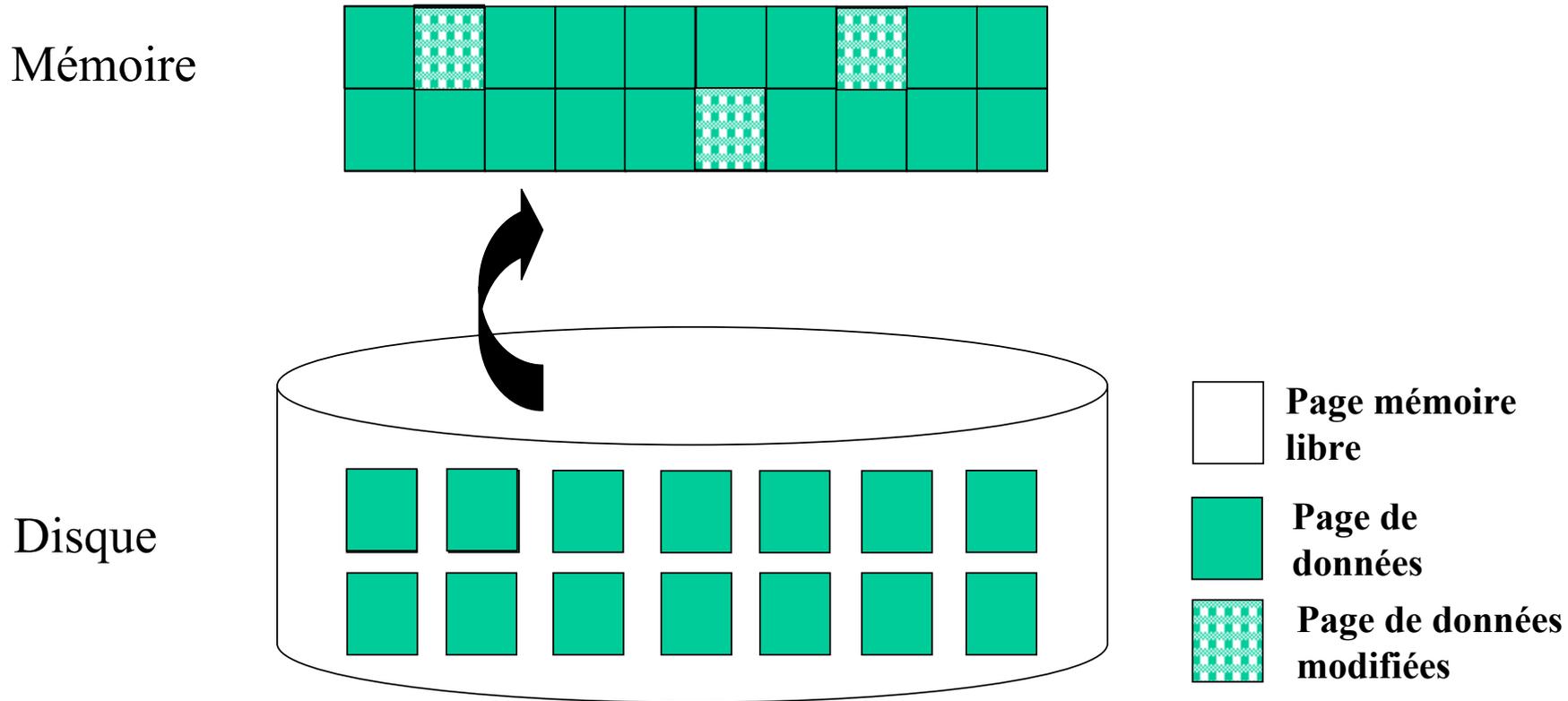


Disque



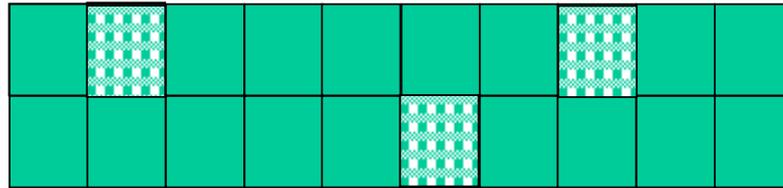
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer



Gestionnaire de buffer

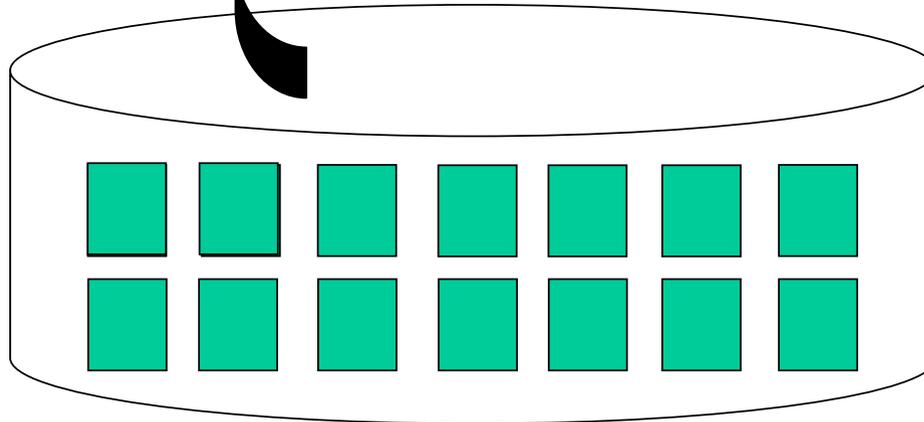
Mémoire



Date de montée
Date de dernier accès
...



Disque

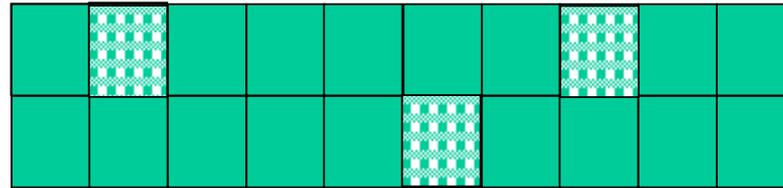


-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

Page non accédée depuis longtemps (LRU)

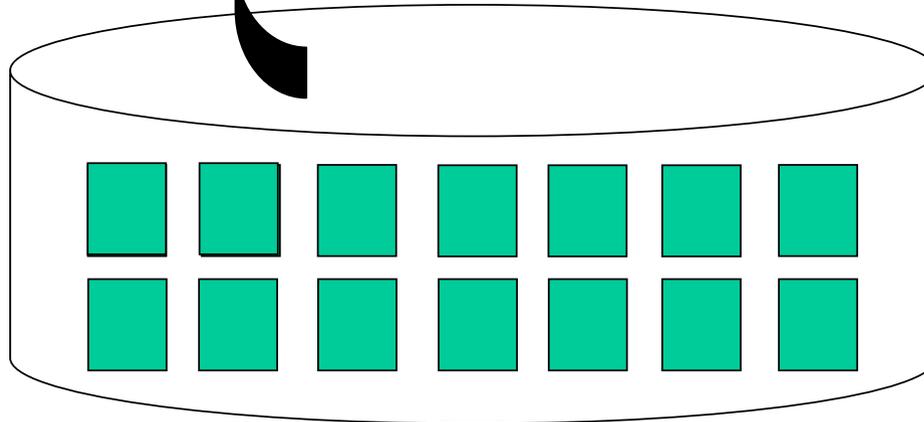
Mémoire



Date de montée
Date de dernier accès
...



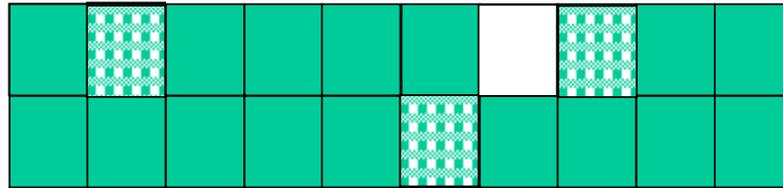
Disque



-  Page mémoire libre
-  Page de données
-  Page de données modifiées

Gestionnaire de buffer

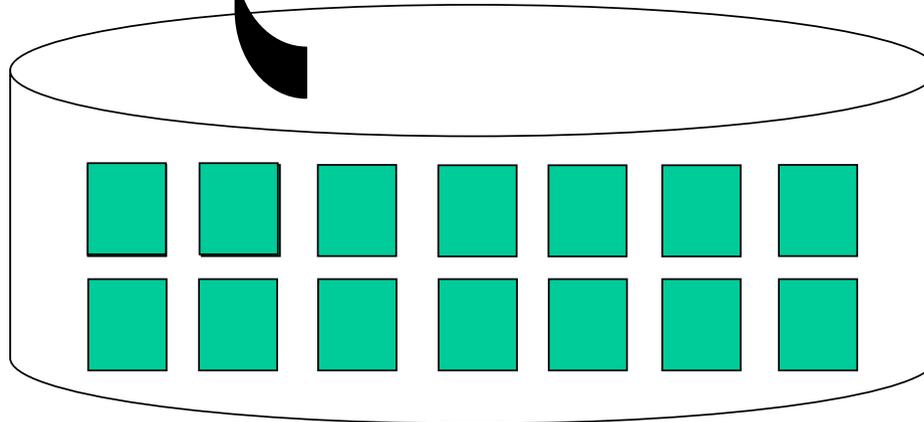
Mémoire



Date de montée
Date de dernier accès
...

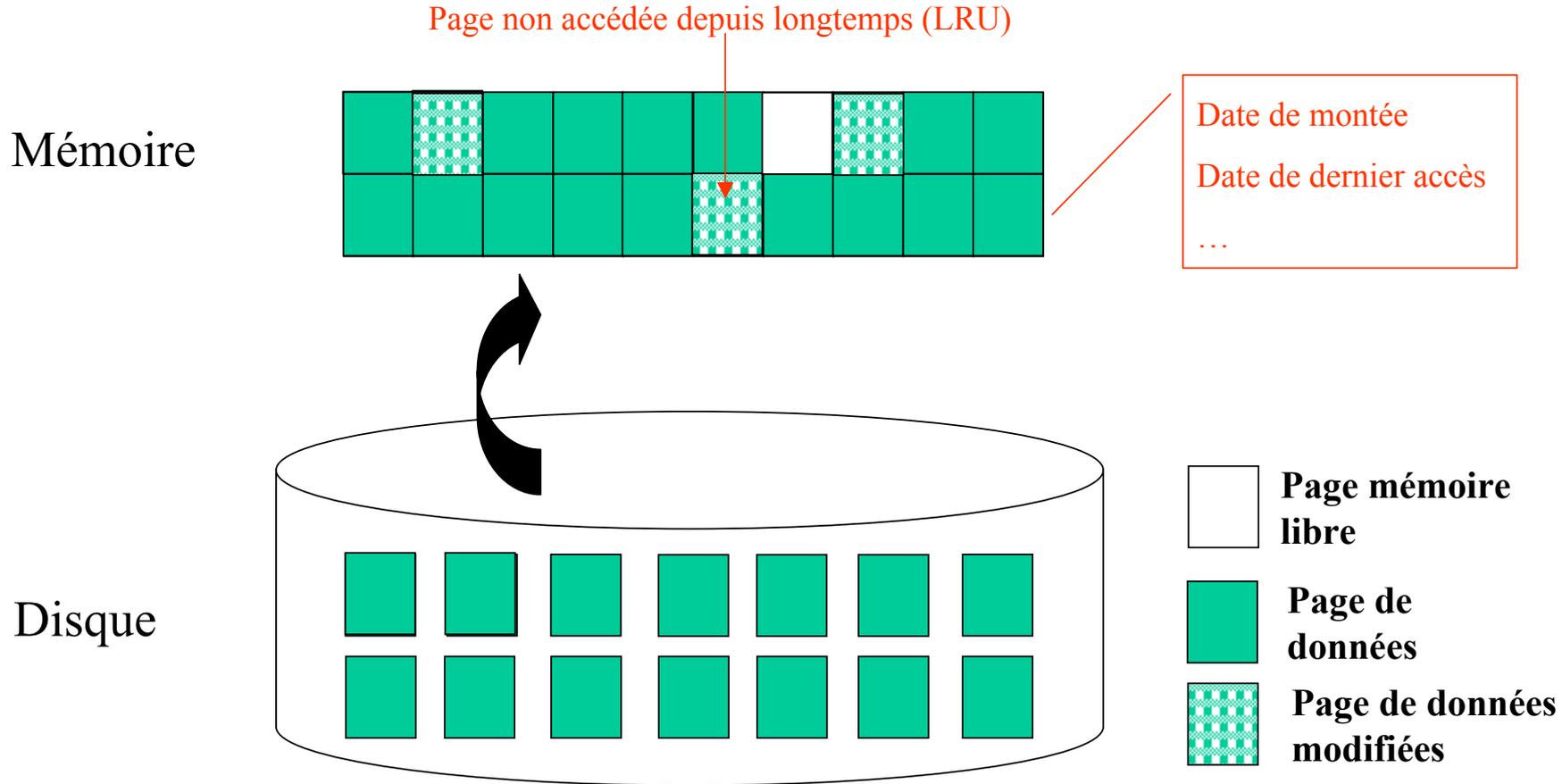


Disque

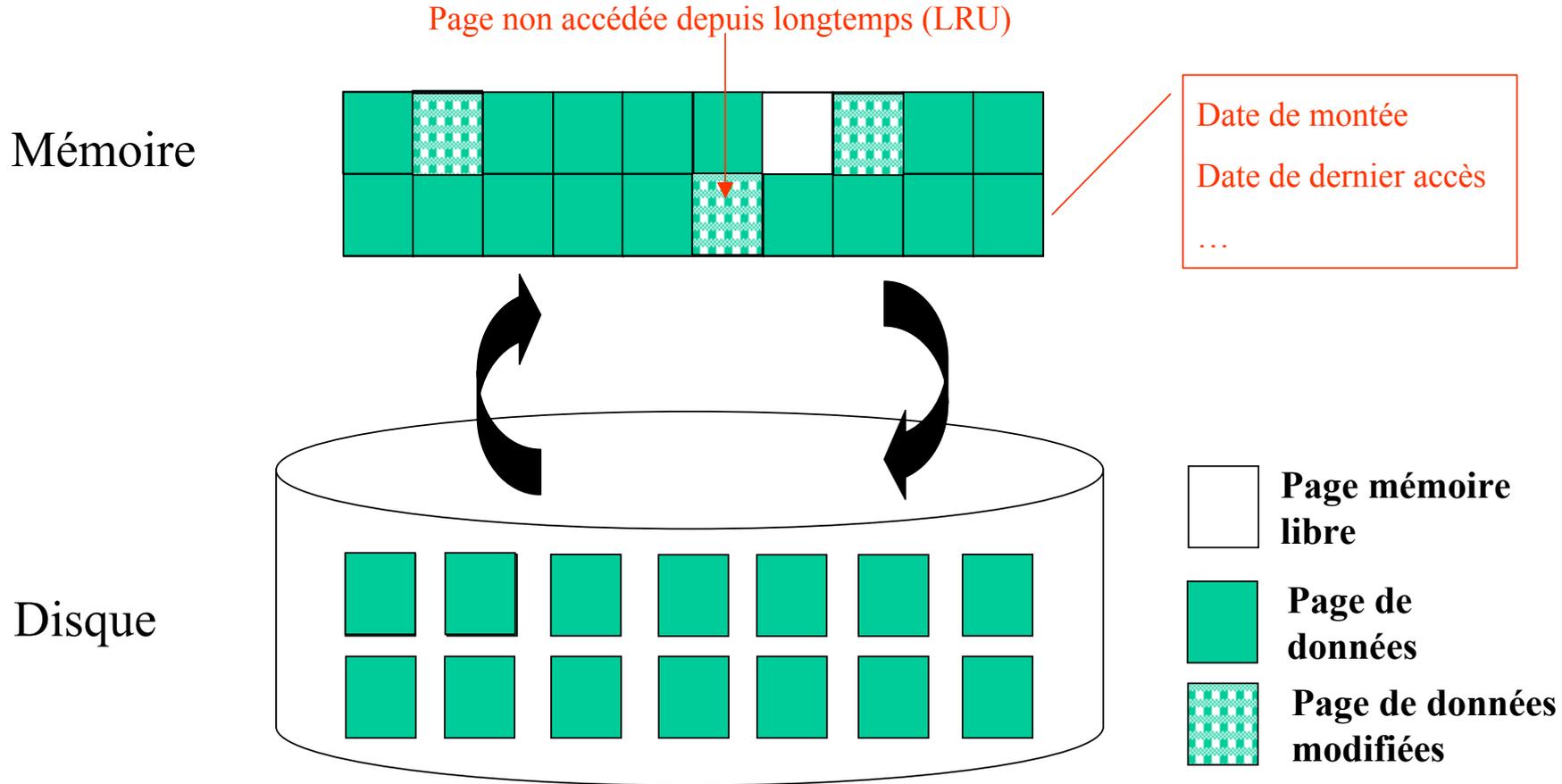


-  **Page mémoire libre**
-  **Page de données**
-  **Page de données modifiées**

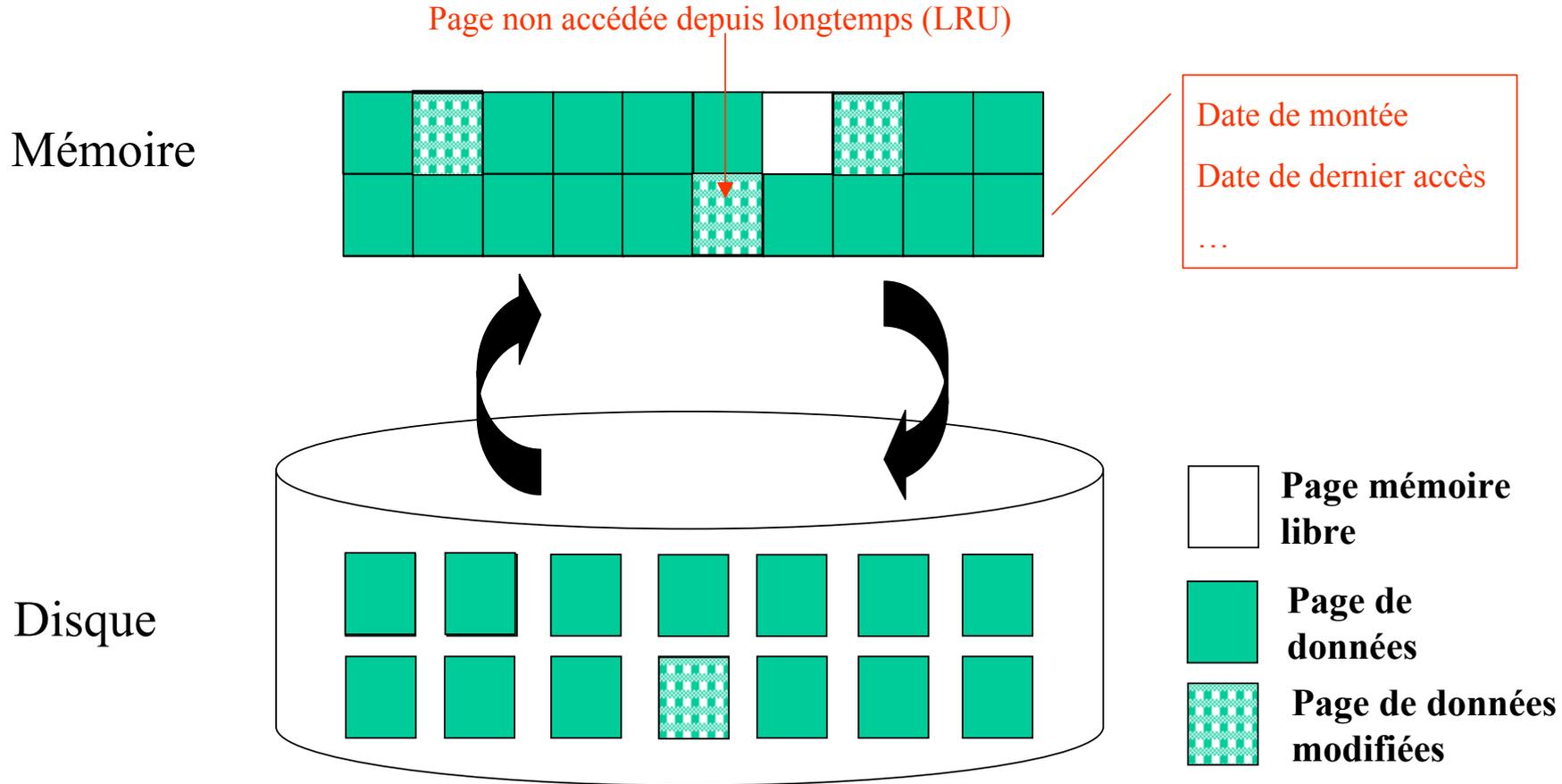
Gestionnaire de buffer



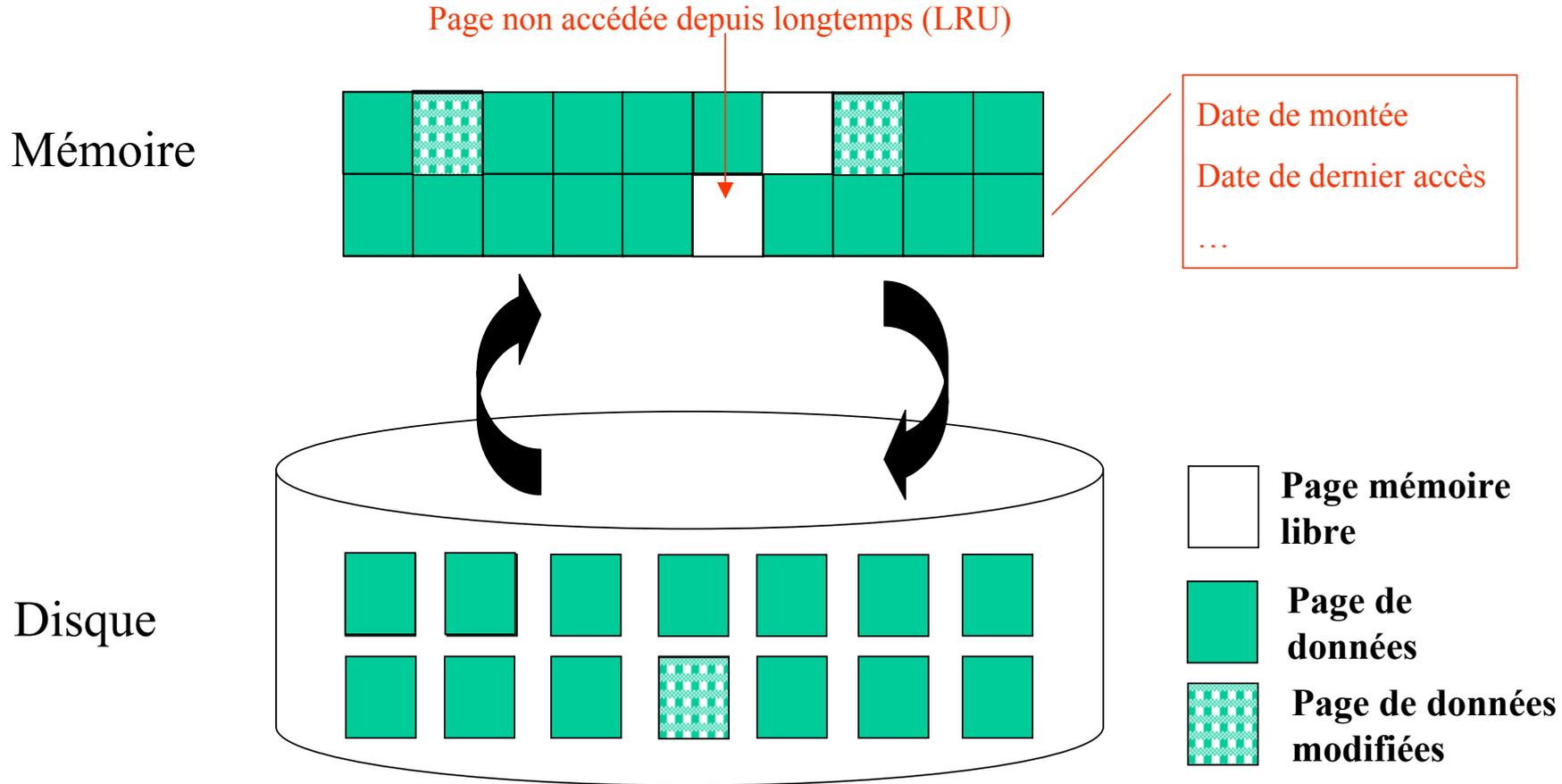
Gestionnaire de buffer



Gestionnaire de buffer



Gestionnaire de buffer



Systeme de fichiers

Intégration ou non des fonctionnalités du SGF du système d'exploitation :

① A chaque relation correspond un fichier

⇒ liaison forte du SGBD et du SGF

② Stockage de toute la base de données dans un seul fichier

⇒ le SGF donne accès aux différentes pages

⇒ le SGBD contrôle tout

Les pages doivent être connues du SGBD

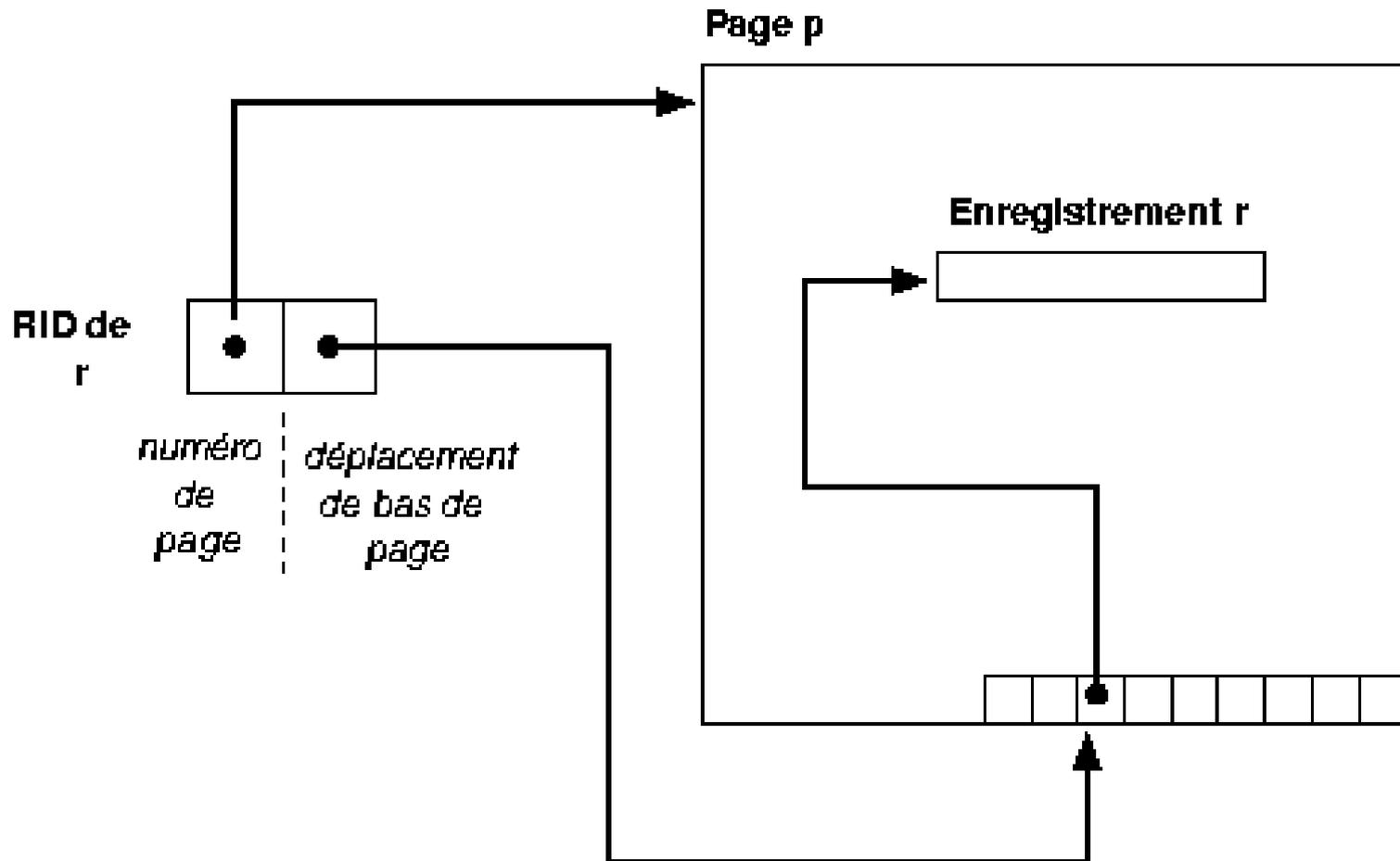
Chap. II - Organisation des données

- Stockage des données
 - Conservation
 - Accès
- Structuration des données
- Moyens de manipulation des données

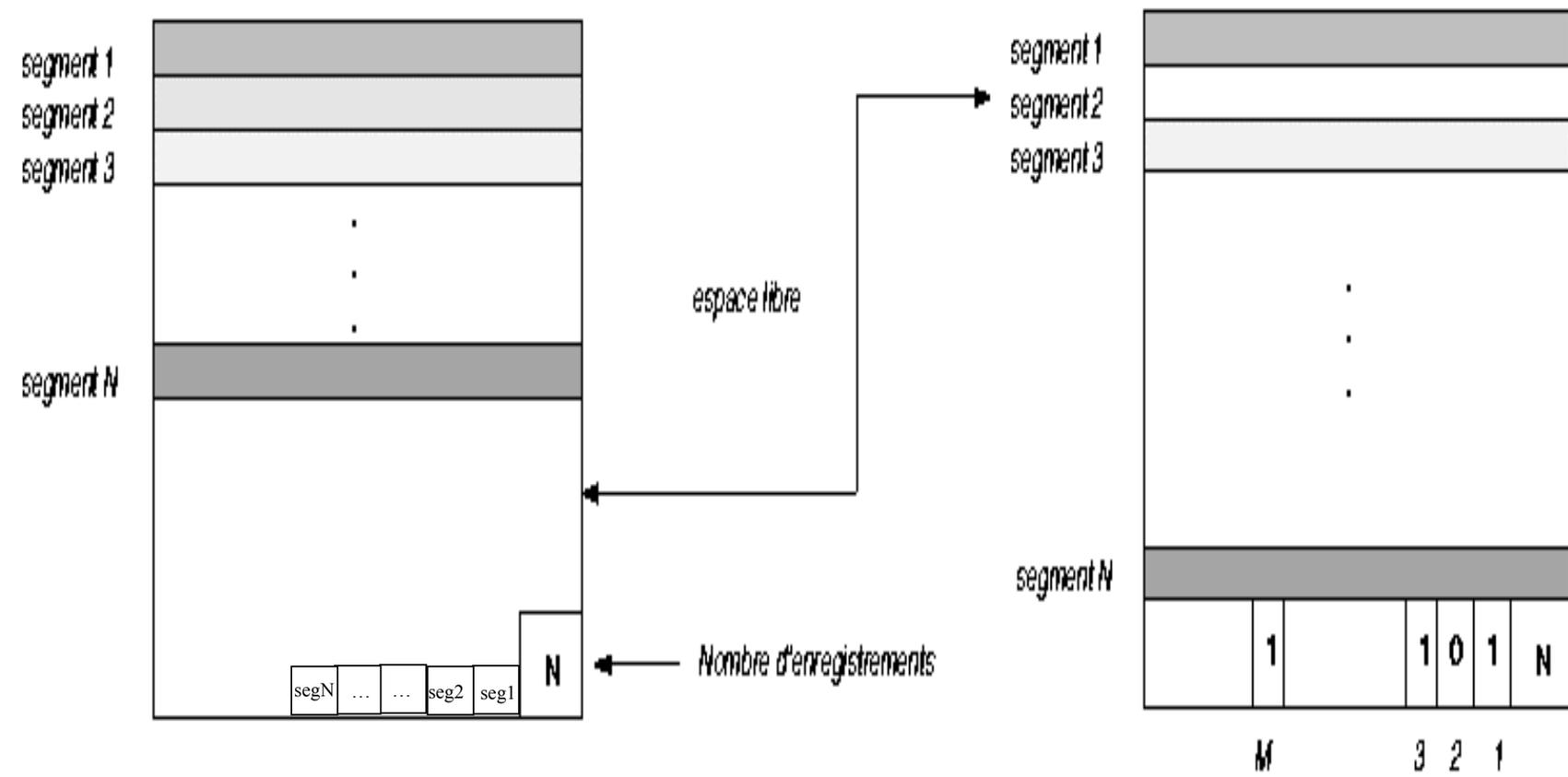
Gestion des fichiers

- Relation : collection de **pages** ou **blocs** disque ou mémoire
- **champ** : séquences d'octets de taille fixe ou variable représentant la valeur d'un attribut de nuplet sur le disque ou en mémoire
- **enregistrement** : collection de taille fixe ou variable de champs

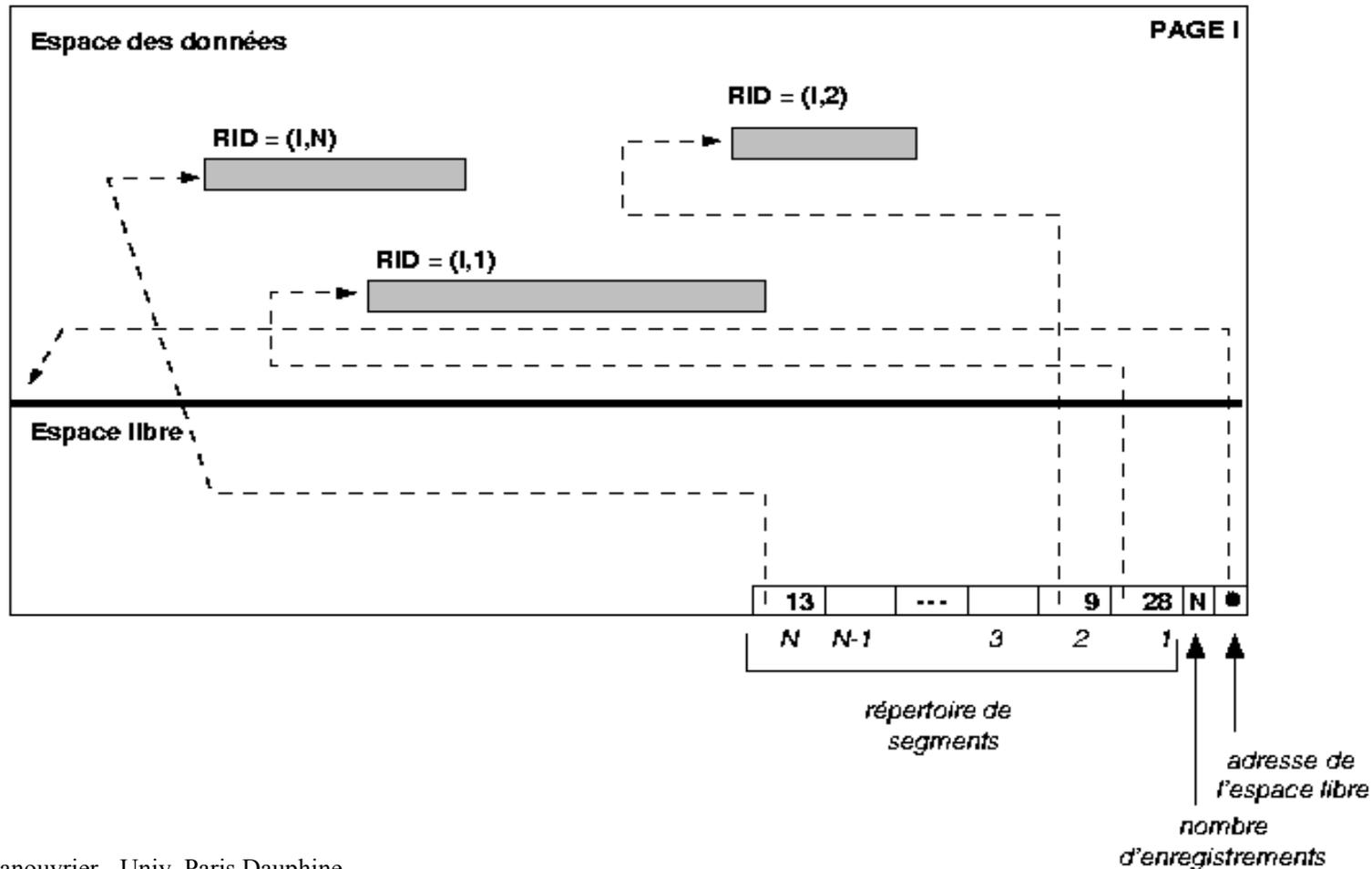
Identification des enregistrements



Placement des enregistrements de taille fixe



Placement des enregistrements de taille variable



Organisation des fichiers

Modèle de coût [RG00] :

- B pages de disque
- R enregistrements par page
- Temps moyen de lecture d'une page : D
- Temps moyen d'accès à un enregistrement : C
- Temps de calcul d'une valeur de fonction de hachage : H

Organisation des fichiers

Trois organisations de fichier :

- **aléatoire** (*Heap File*)
- **ordonné** (*Sorted File*)
- **hachage** (*Hash File*)

Plusieurs opérations :

- Lecture complète du fichier
- Recherche par égalité
- Recherche par intervalle
- Insertion
- Suppression

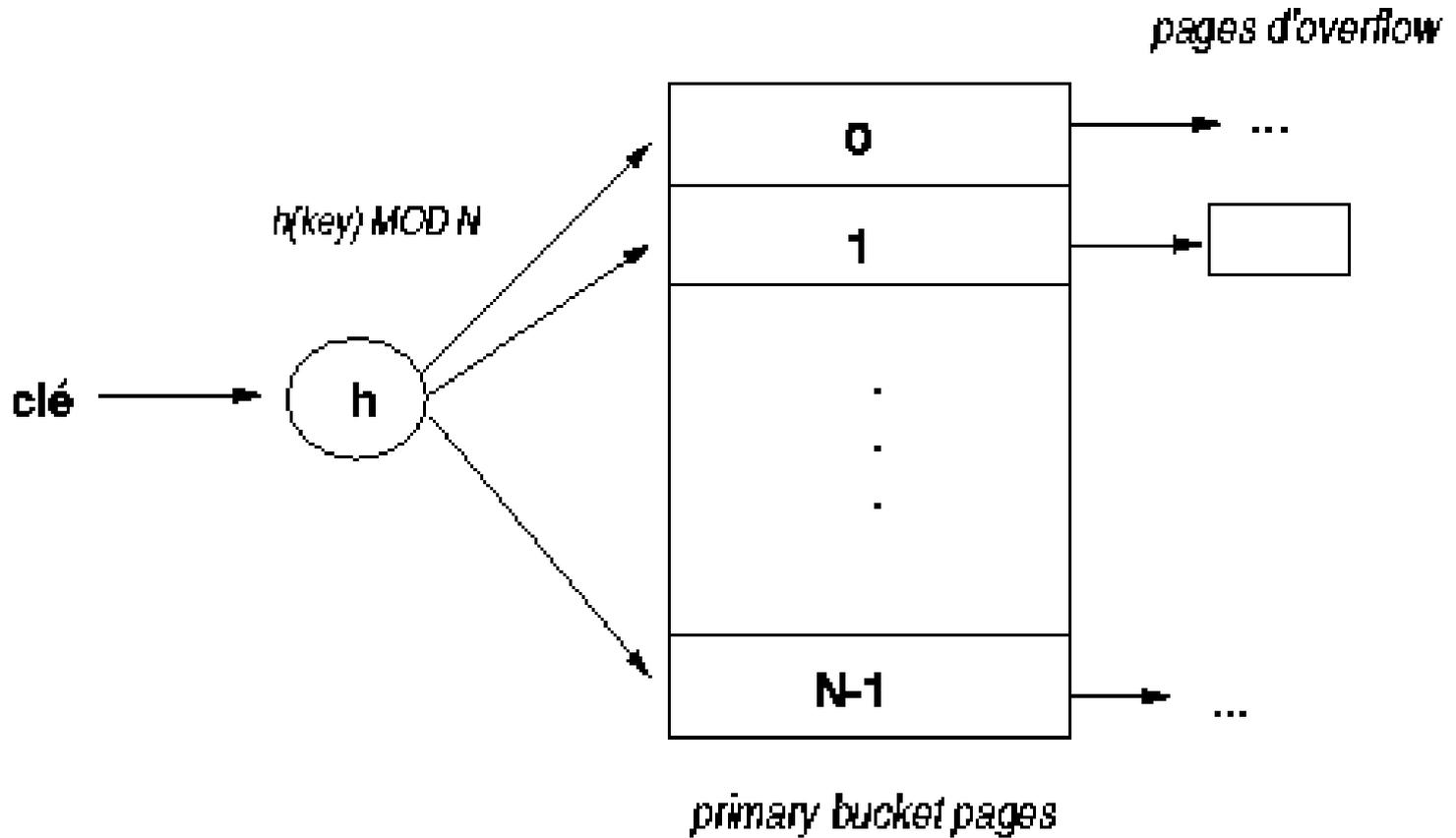
Fichier de hachage

- **Fonction de hachage :**

calcule la page où doit être stocké l'enregistrement en fonction de la valeur d'un ou plusieurs de ses champs

- Regroupement de pages d'un fichier de hachage en **buckets** composés de **segments**
- Gestion de **pages d'overflow**
- Placement des enregistrements par ordre d'arrivée dans le bucket
- Pas de garantie d'adresse unique
- Gestion des **collisions**

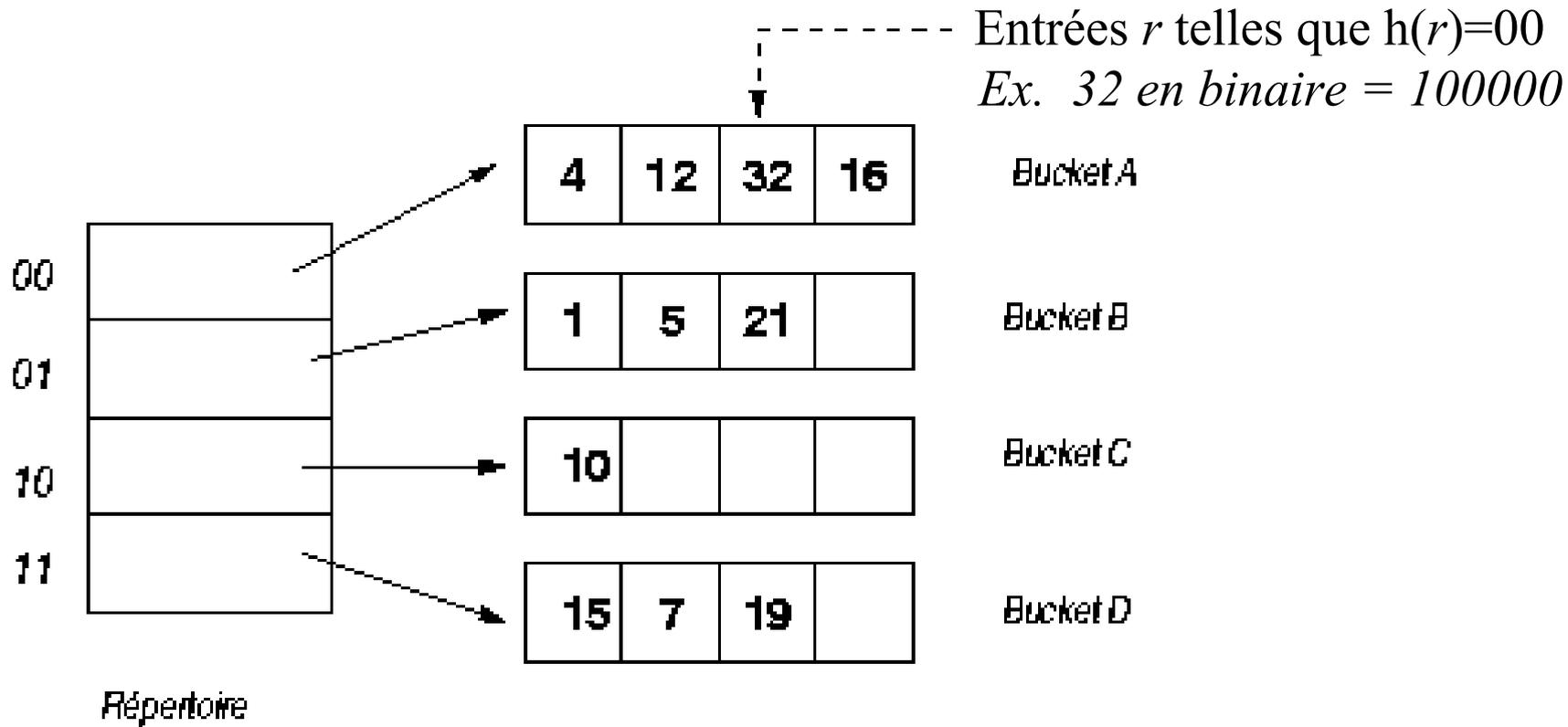
Hachage statique



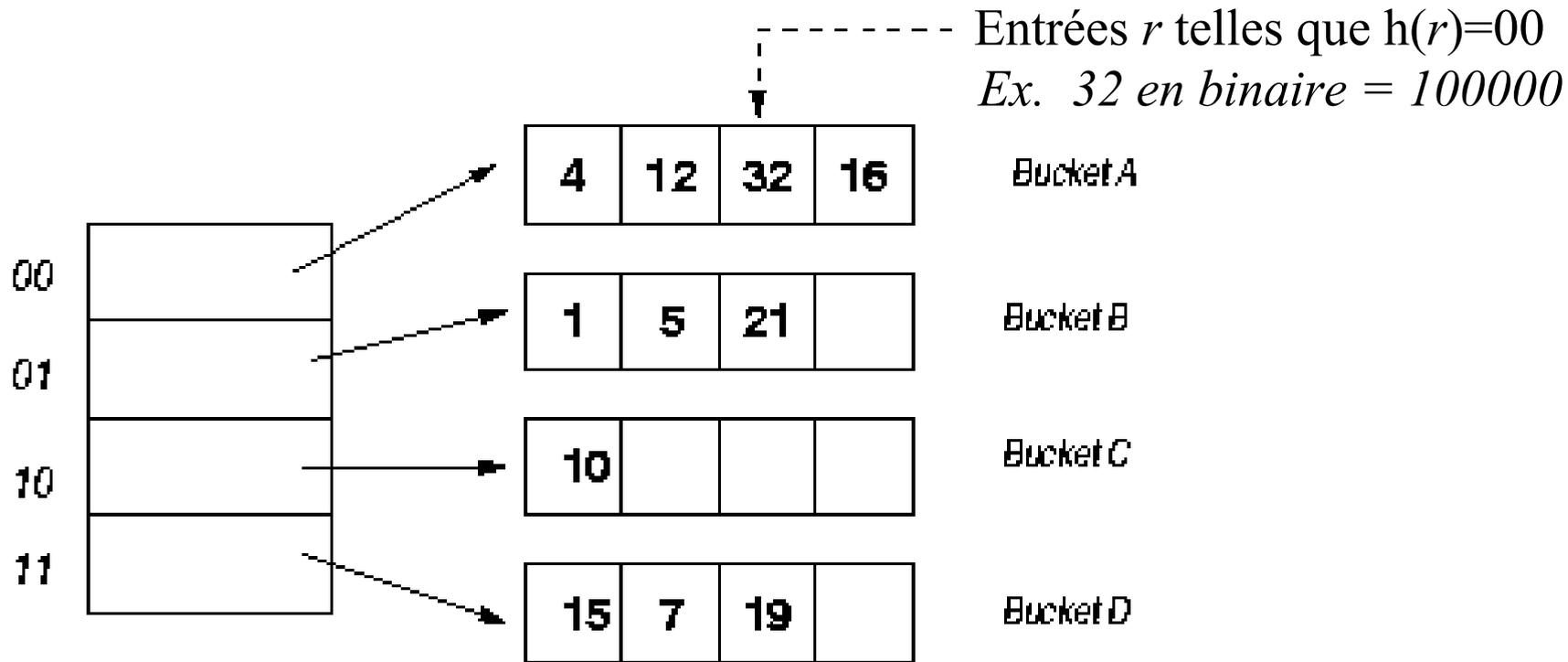
Gestion des collisions

- *Open addressing* : Recherche linéaire du premier segment vide
- *Unchained overflow* : Maintien d'une zone d'overflow
- *Chained overflow* : Maintien, pour chaque bucket, d'un pointeur vers une zone d'overflow
- *Multiple hashing* : Utilisation d'une deuxième fonction de hachage pour placer les enregistrements dans la zone d'overflow

Hachage extensible (1/2)



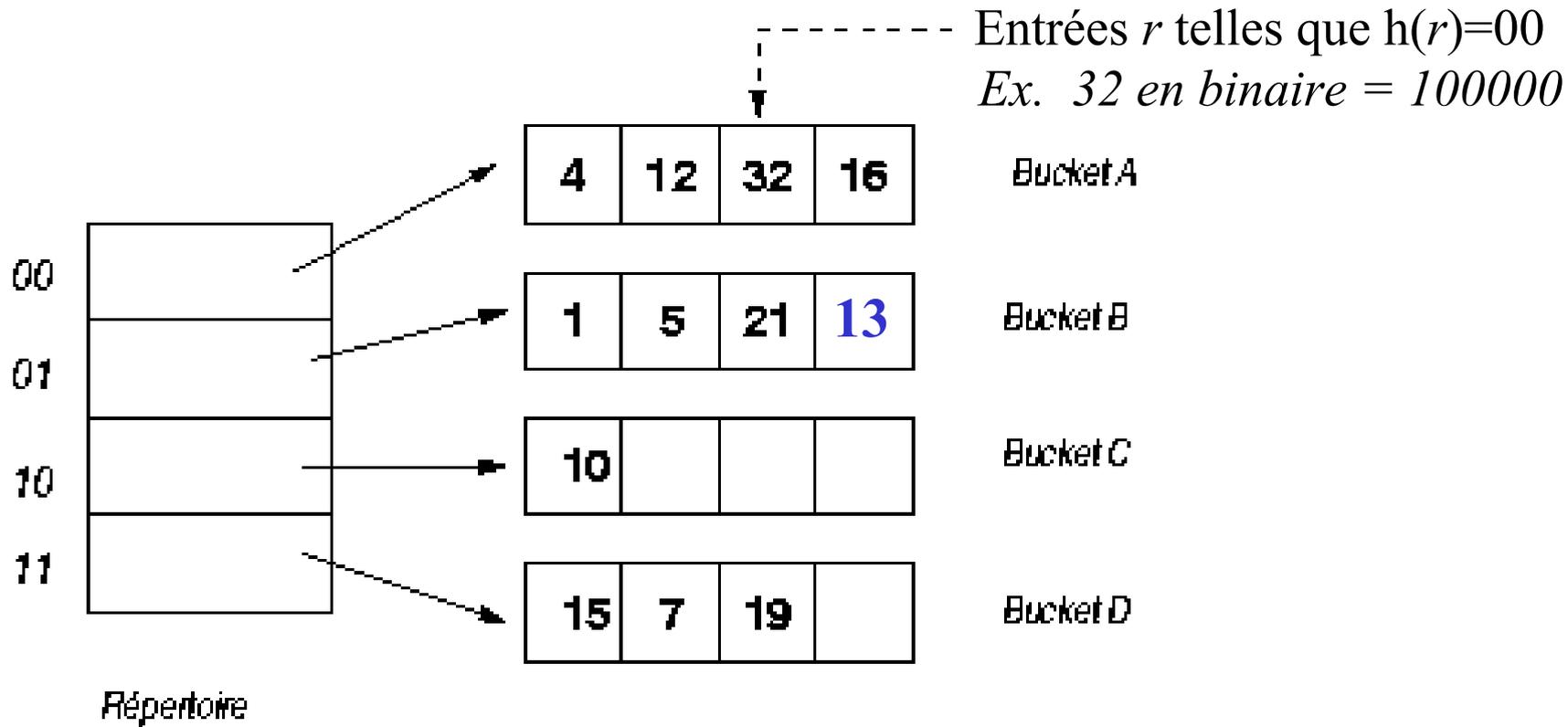
Hachage extensible (1/2)



Répertoire

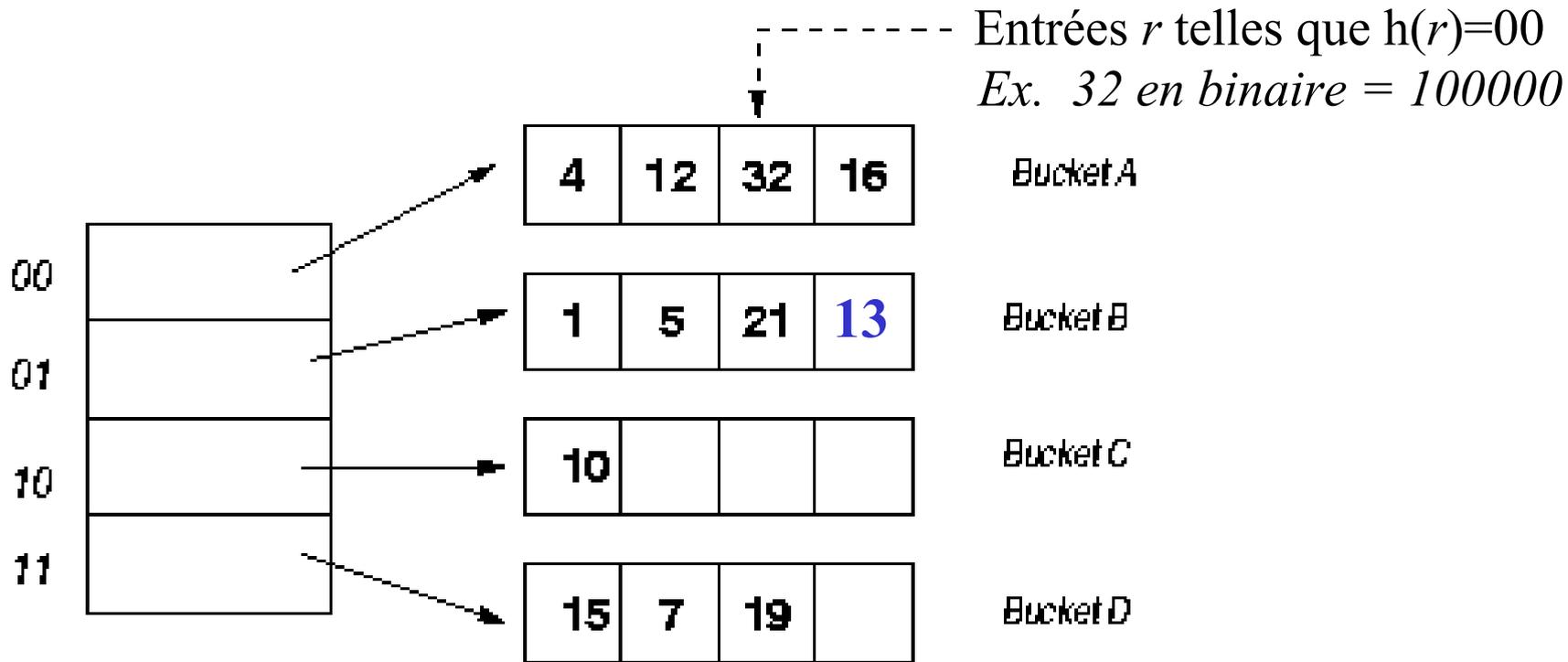
Insertions : 13 en binaire = 1101

Hachage extensible (1/2)



Insertions : 13 en binaire = 1101

Hachage extensible (1/2)



Répertoire

Insertions : 13 en binaire = 1101

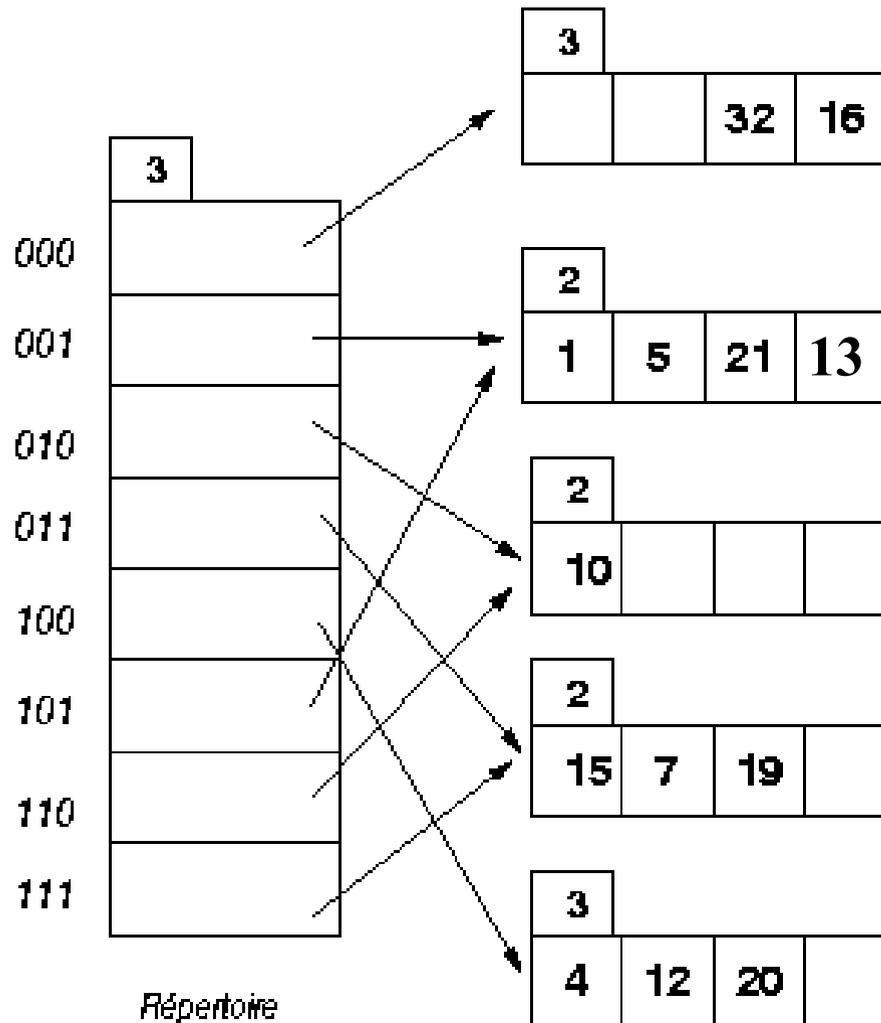
20 en binaire = 10100 mais bucket A plein

Hachage extensible (2/2)

**La taille du
répertoire
est doublée
et le bucket
A est divisé
en 2**

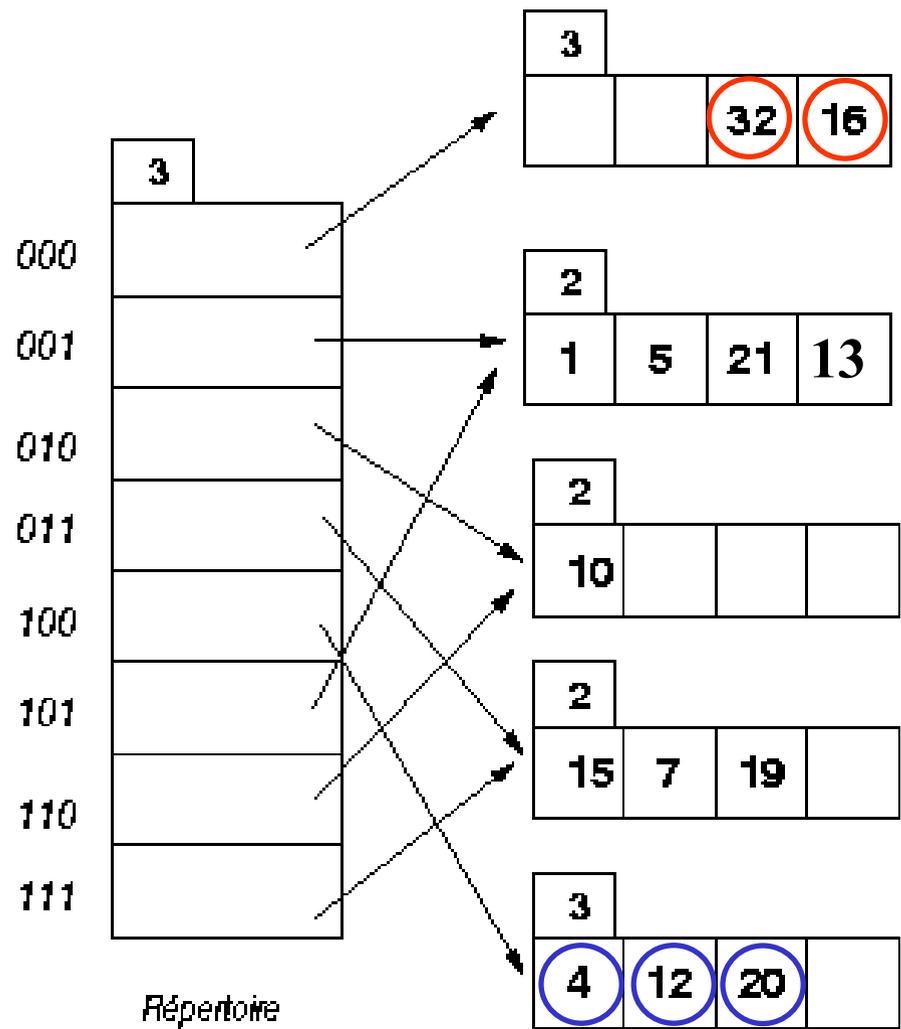
Hachage extensible (2/2)

La taille du répertoire est doublée et le bucket A est divisé en 2



Hachage extensible (2/2)

La taille du répertoire est doublée et le bucket A est divisé en 2

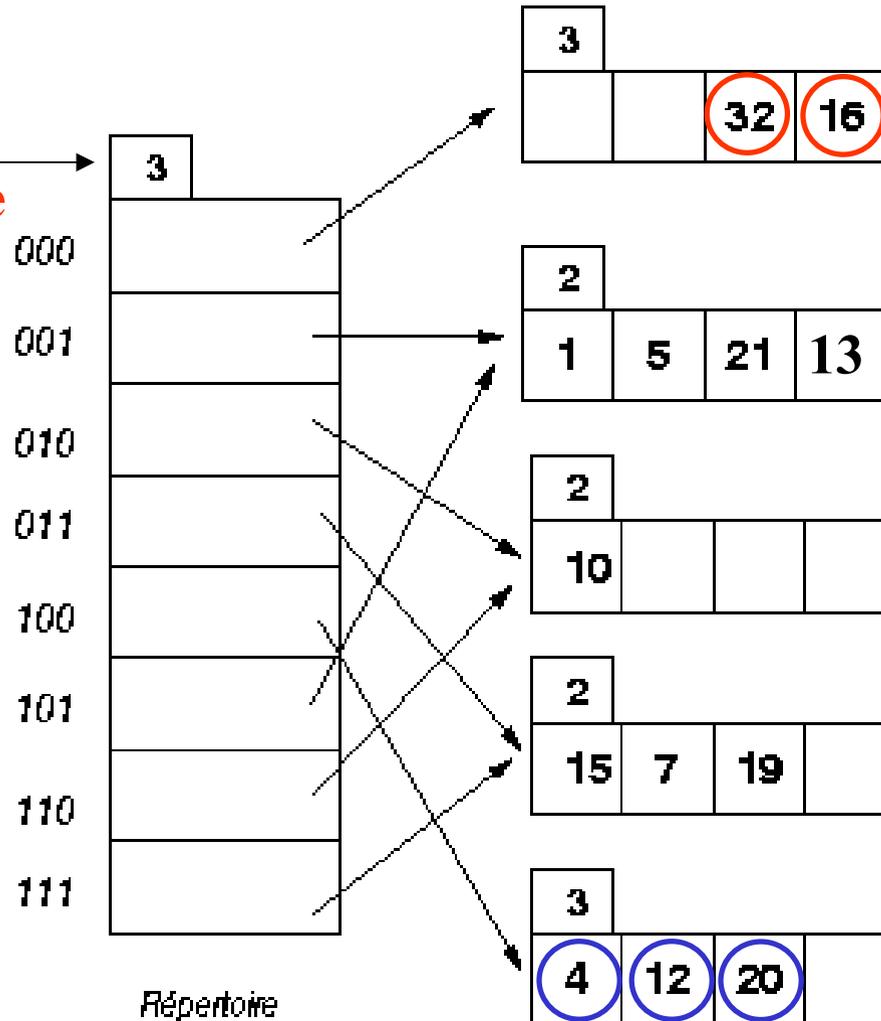


- $h(4) = 100$
- $h(12) = 100$
- $h(16) = 000$
- $h(32) = 000$
- $h(20) = 100$

Hachage extensible (2/2)

Taille globale

La taille du répertoire est doublée et le bucket A est divisé en 2



$$h(4)=100$$

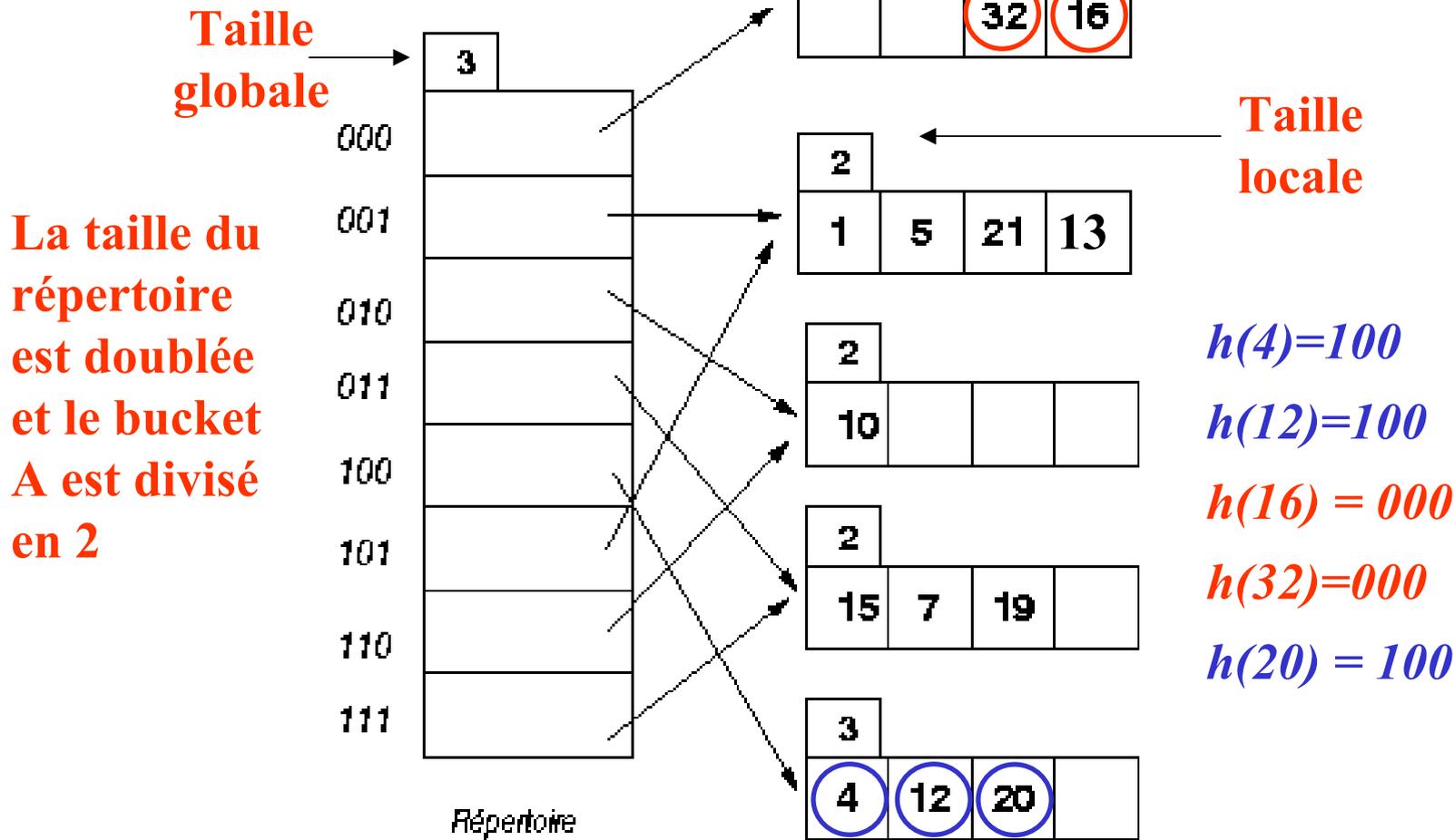
$$h(12)=100$$

$$h(16) = 000$$

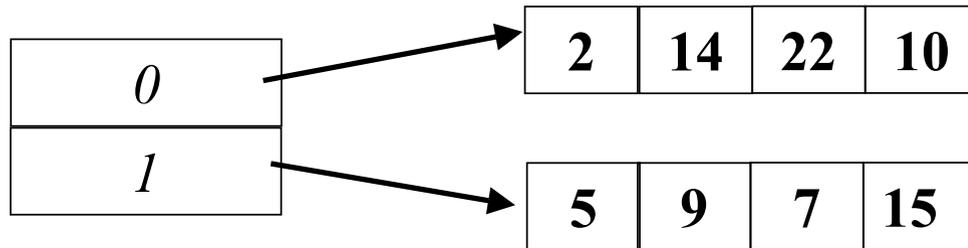
$$h(32)=000$$

$$h(20) = 100$$

Hachage extensible (2/2)



Autre exemple de hachage extensible (1/4)



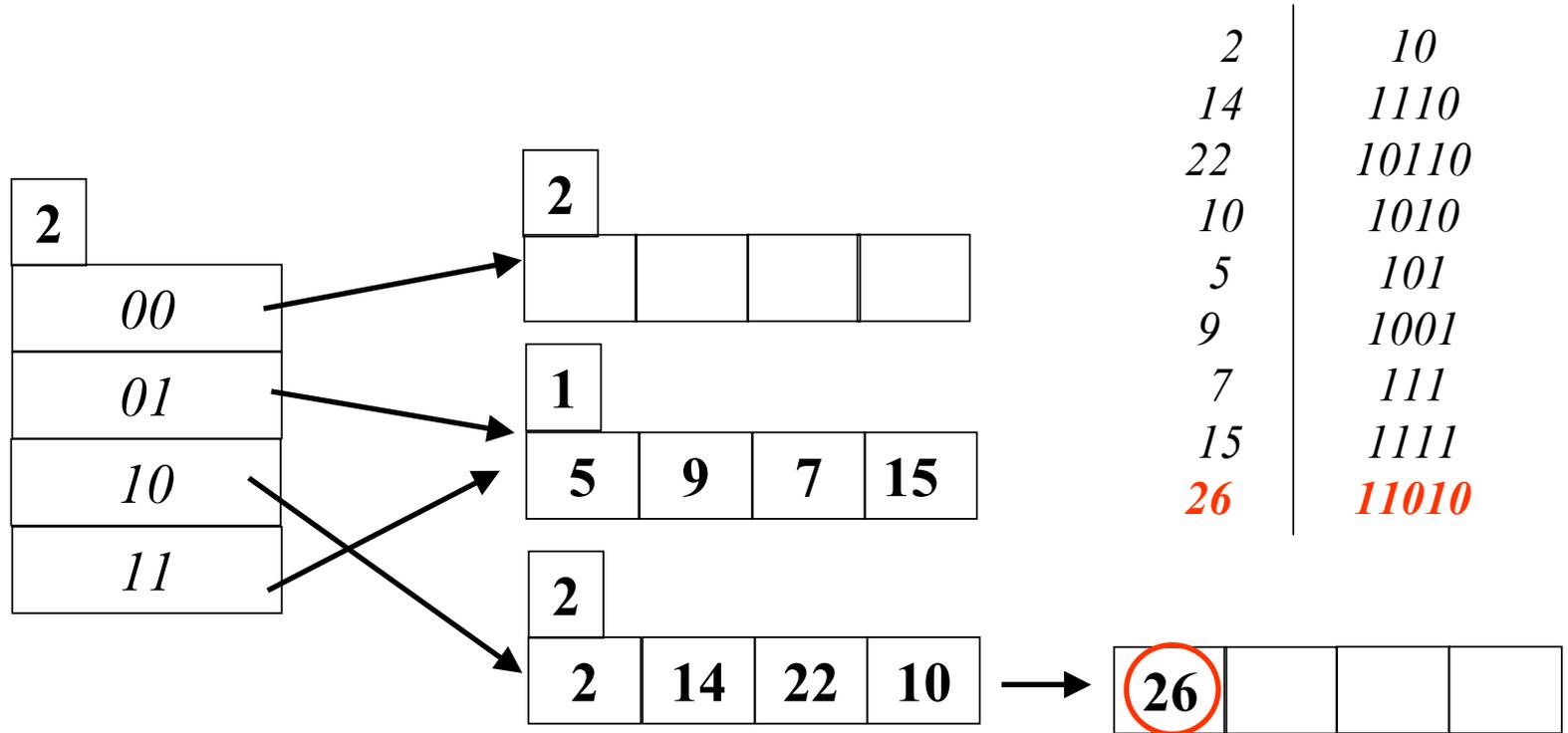
*Répertoire de
buckets*

buckets

2		10
14		1110
22		10110
10		1010
5		101
9		1001
7		111
15		1111

On ajoute l'entrée 26 (11010)

Autre exemple de hachage extensible (2/4)

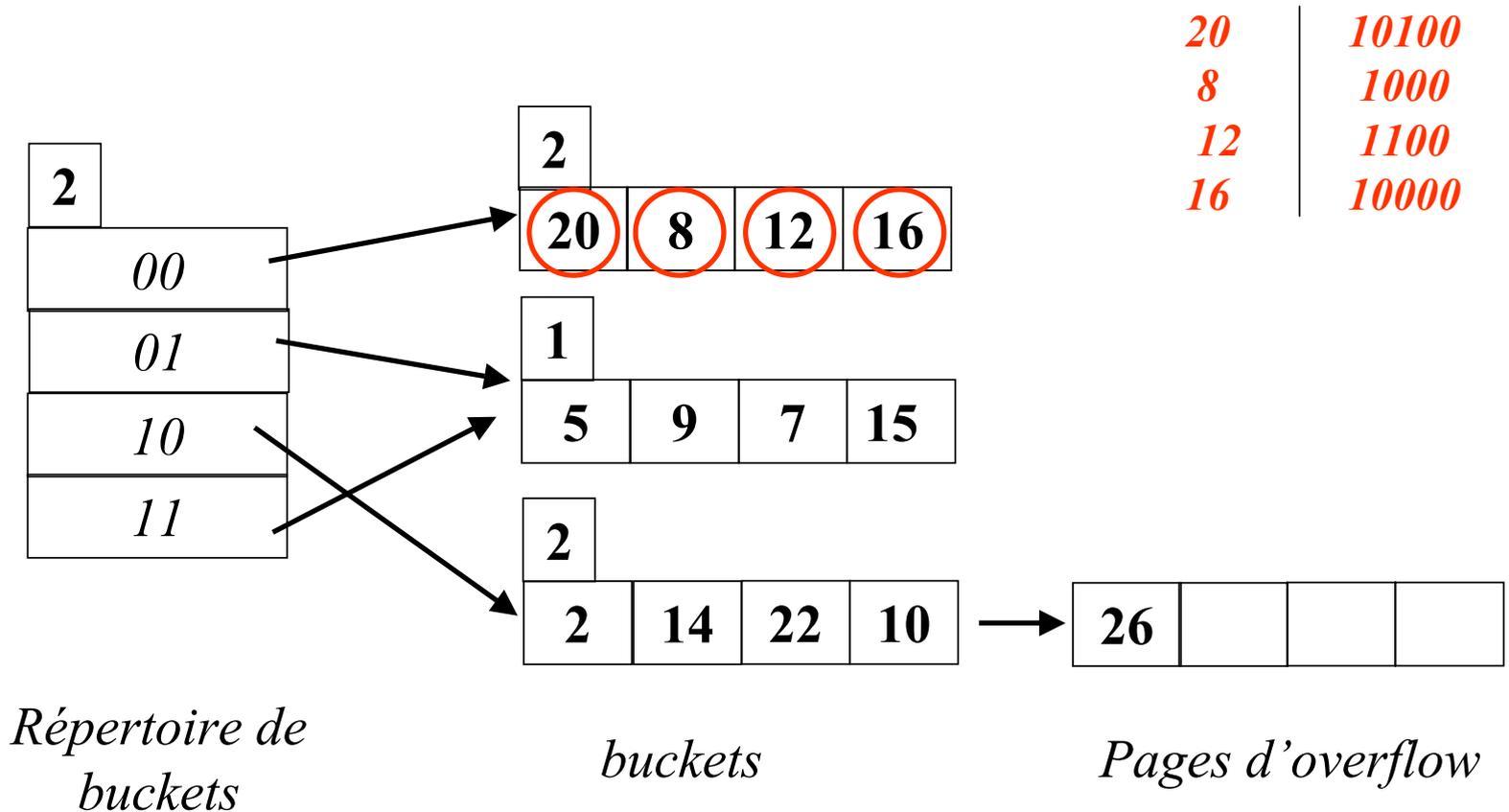


Répertoire de buckets

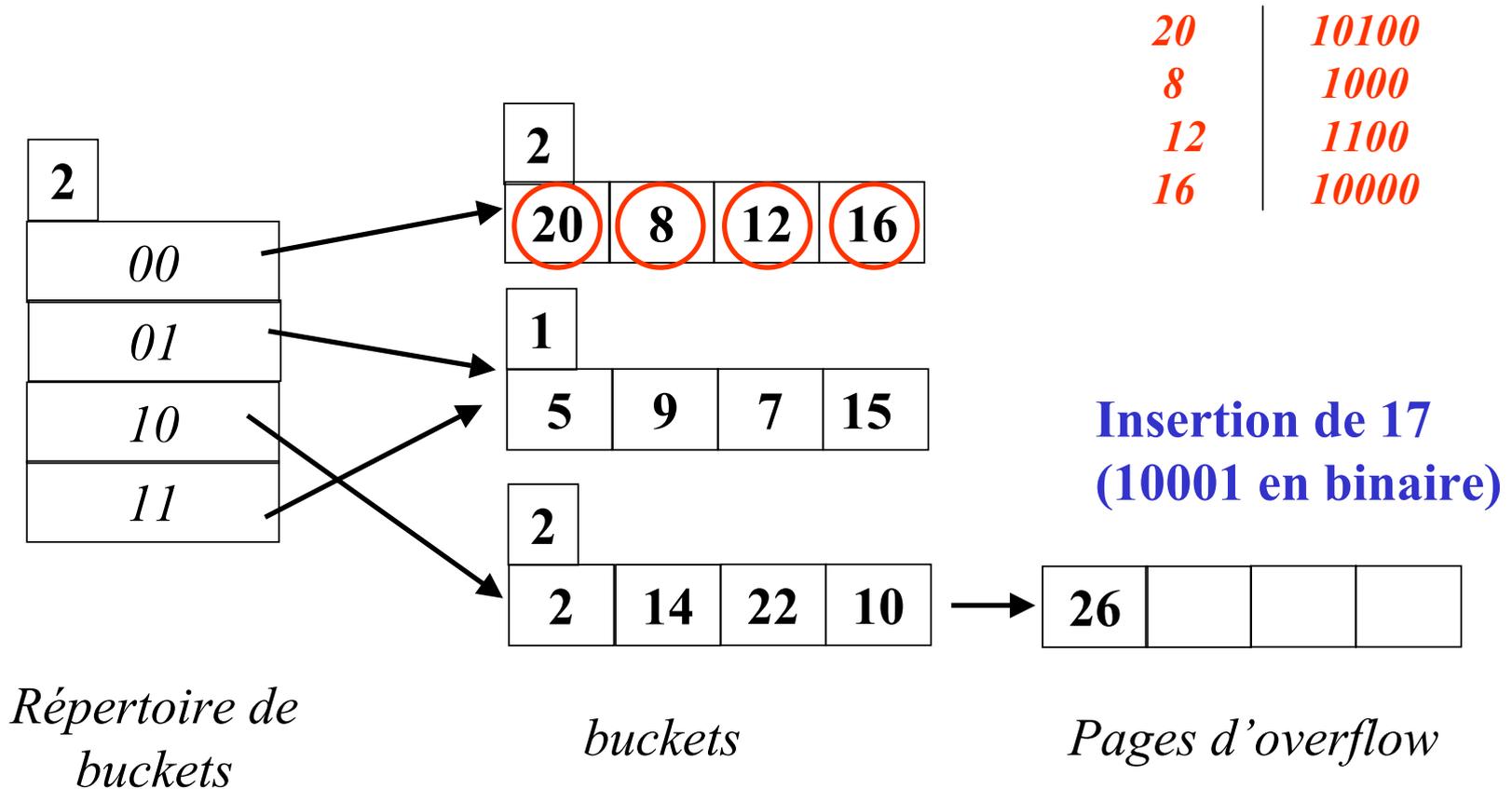
buckets

Pages d'overflow

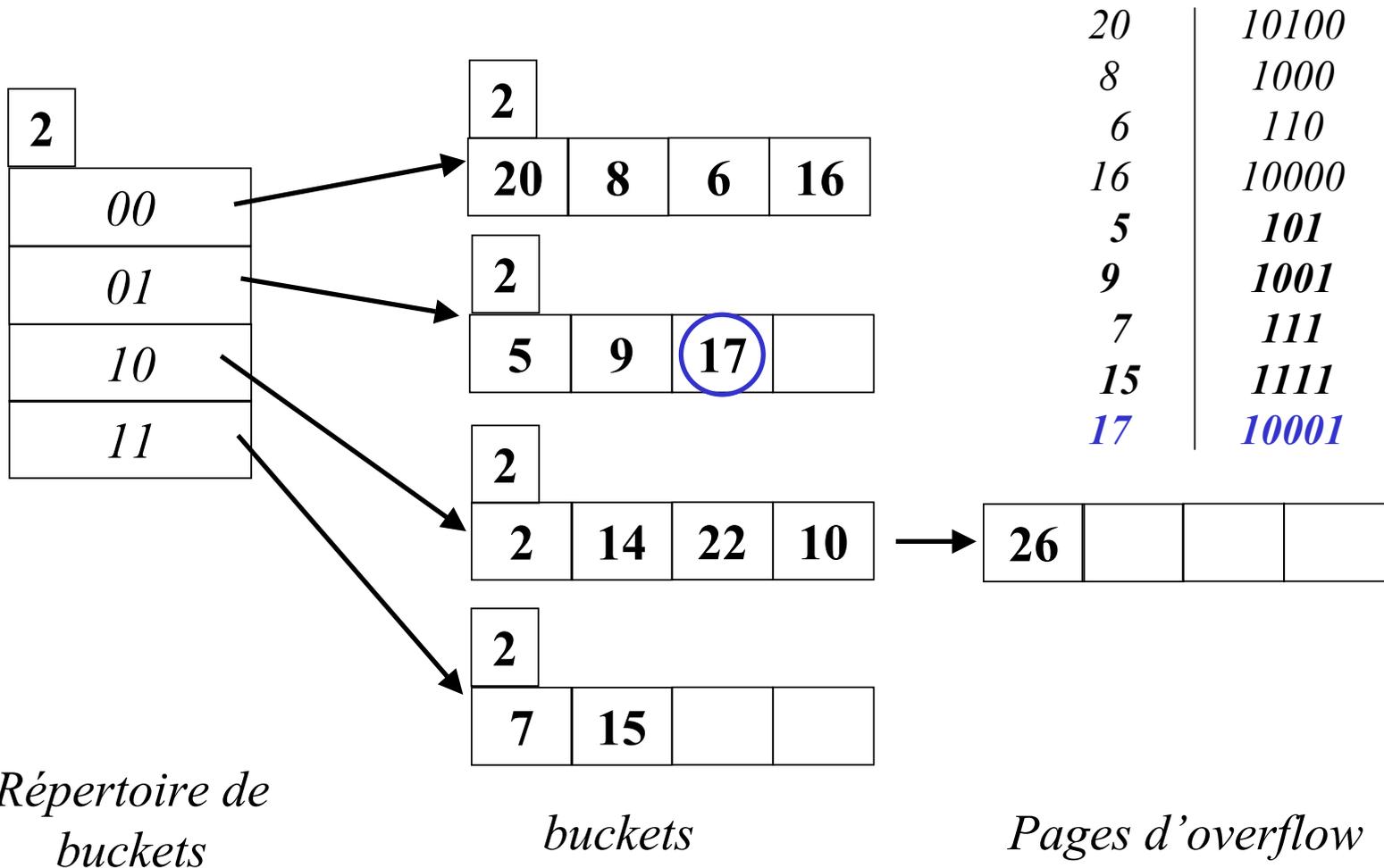
Autre exemple de hachage extensible (3/4)



Autre exemple de hachage extensible (3/4)



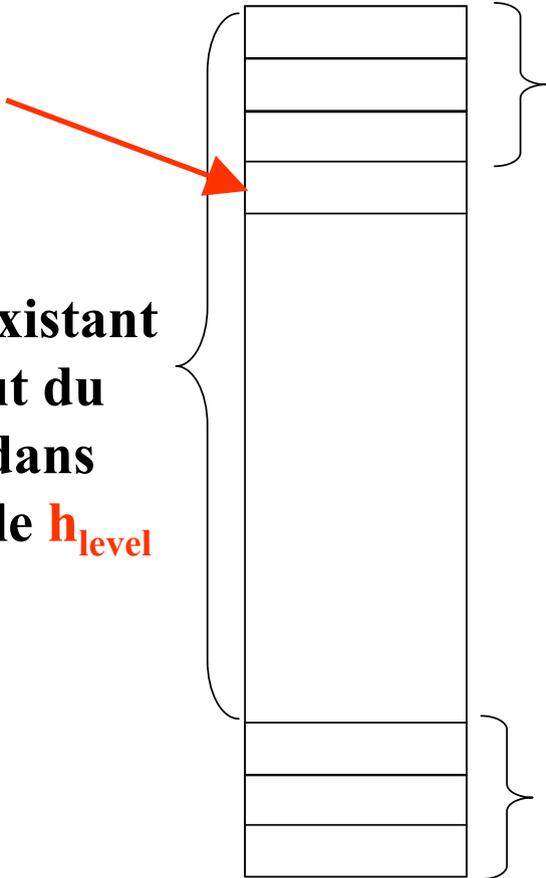
Autre exemple de hachage extensible (4/4)



Hachage linéaire (1/6)

Prochain
bucket à
diviser

Buckets existant
au début du
round dans
l'intervalle h_{level}



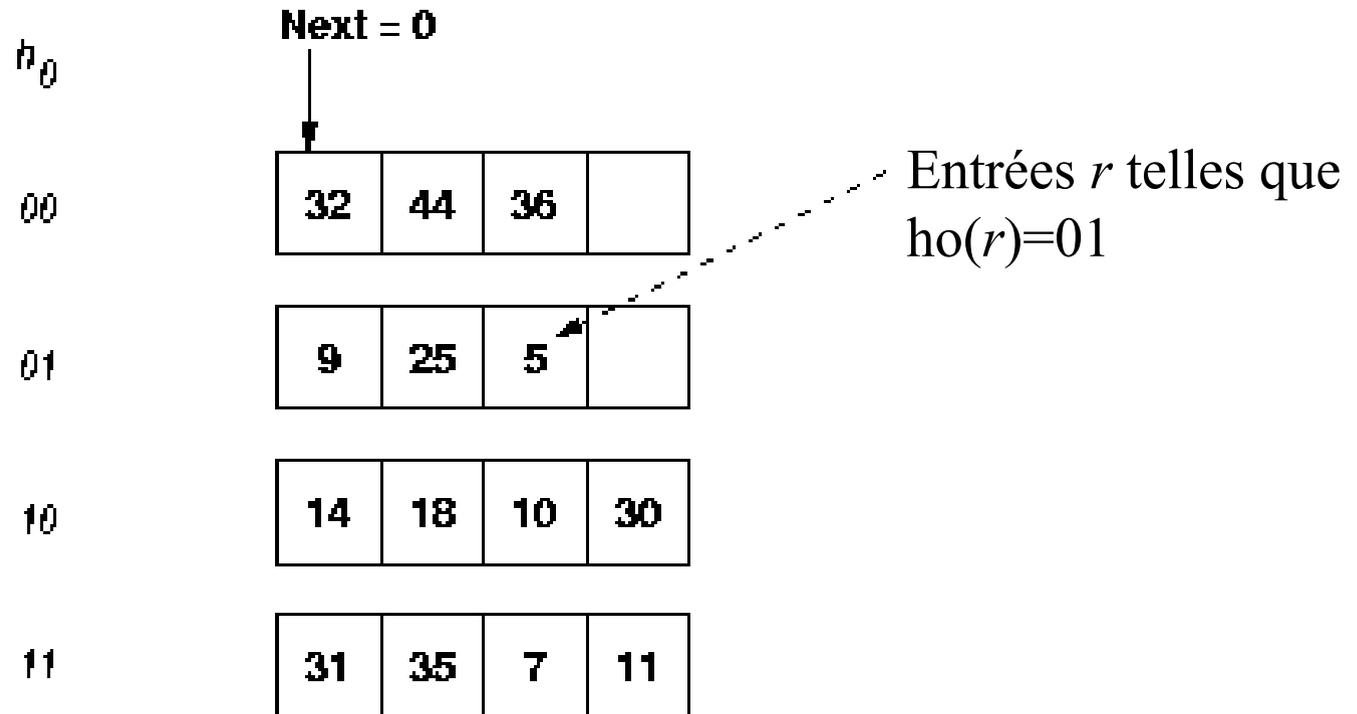
Buckets dédoublés pendant le *round*

Si $h_{\text{level}}(\text{clé})$ est dans cet intervalle, il faut utiliser $h_{\text{level}+1}(\text{clé})$ pour savoir si l'entrée est dans les buckets dédoublés

Buckets dédoublés pendant le *round*

Hachage linéaire (2/6)

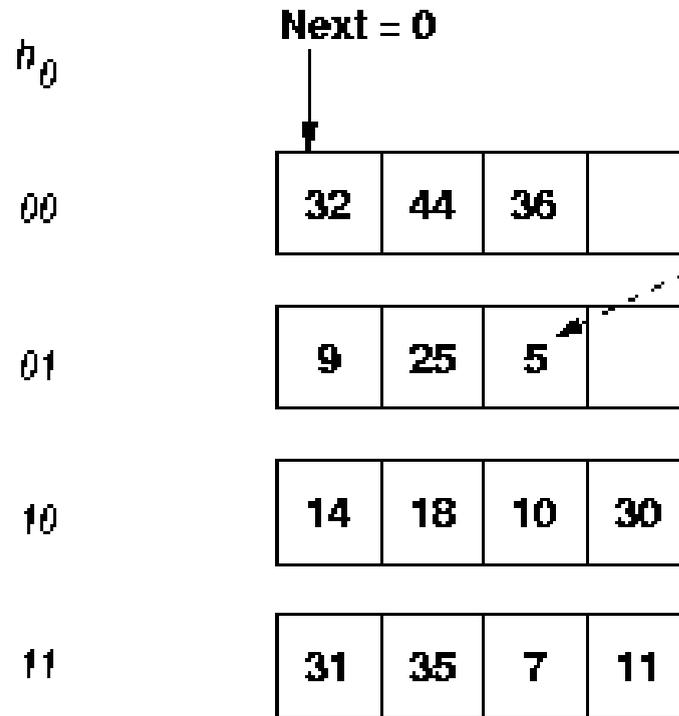
level = 0, N=4



Four info

Hachage linéaire (2/6)

level = 0, N=4



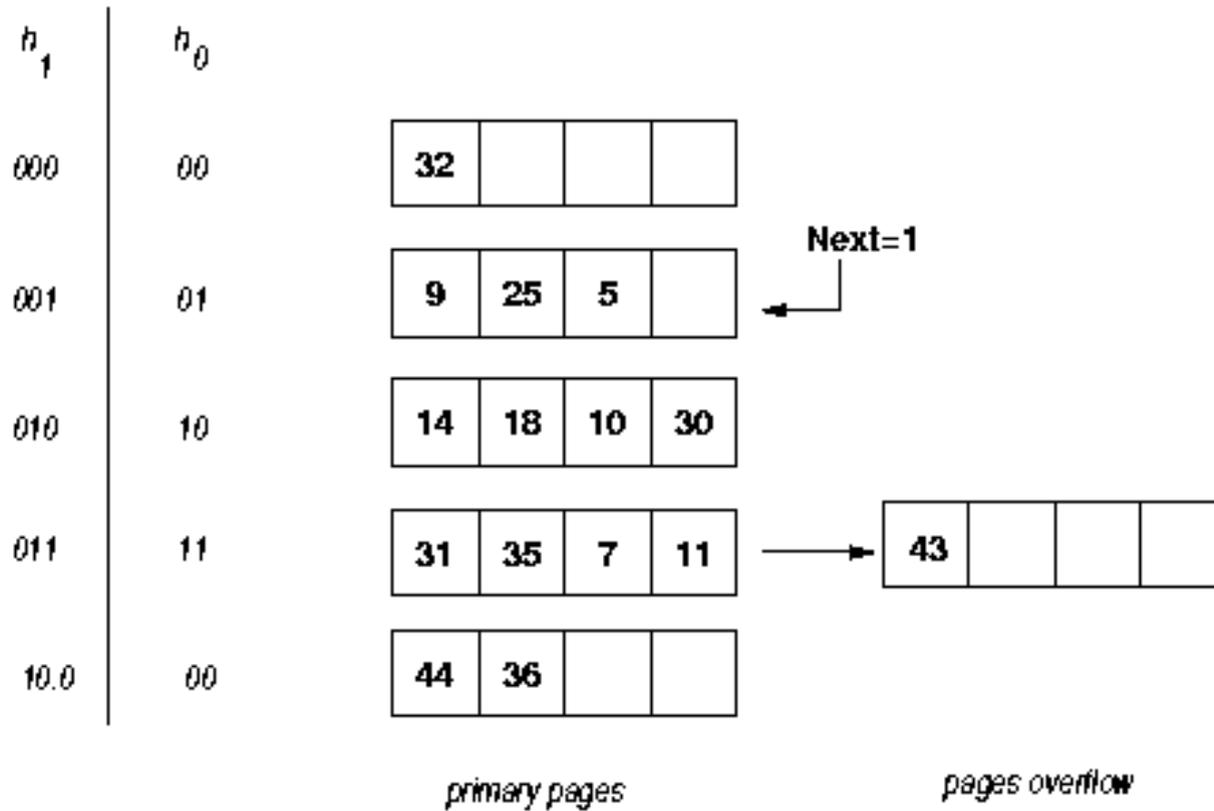
Entrées r telles que $h_0(r)=01$

Insertion de l'entrée 43 :
43 en binaire = 101011
 $h_0(43)=11$
Or le bucket est plein
⇒ Division du bucket Next

Four info

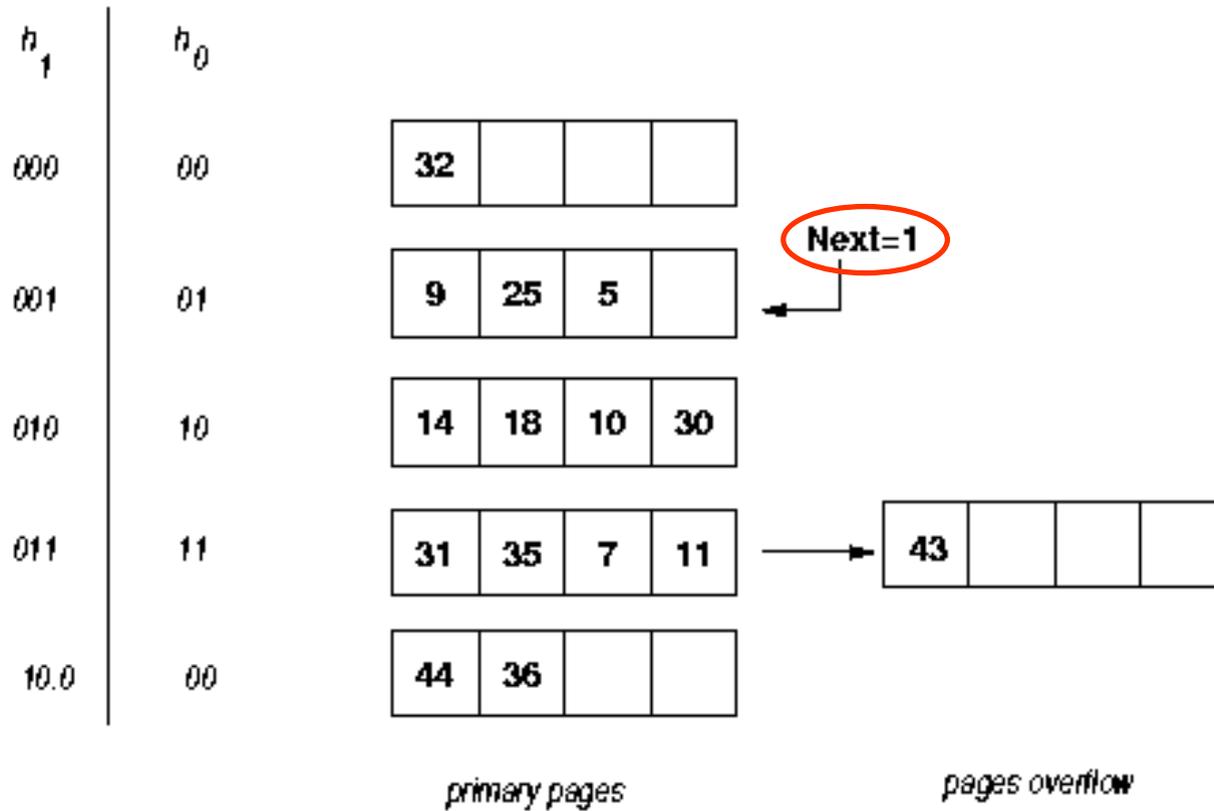
Hachage linéaire (3/6)

level = 0, N=4



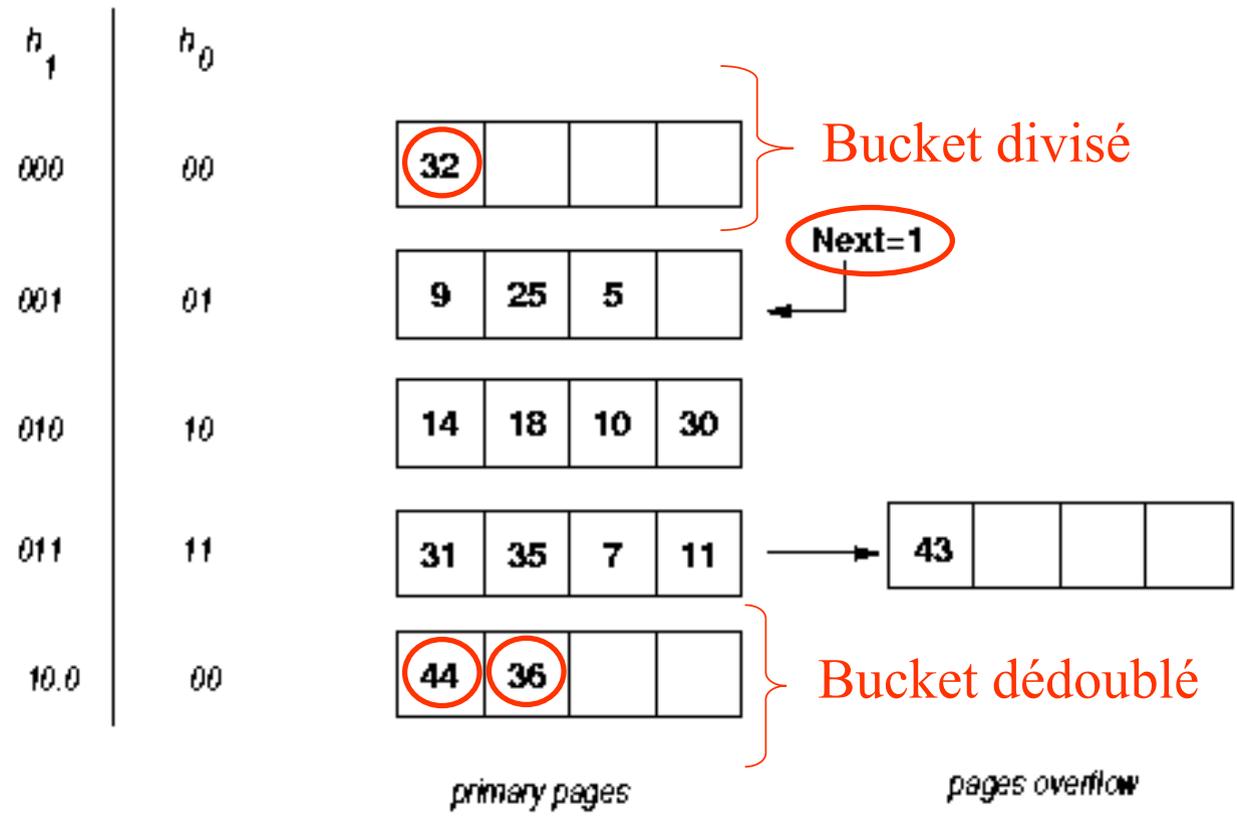
Hachage linéaire (3/6)

level = 0, N=4



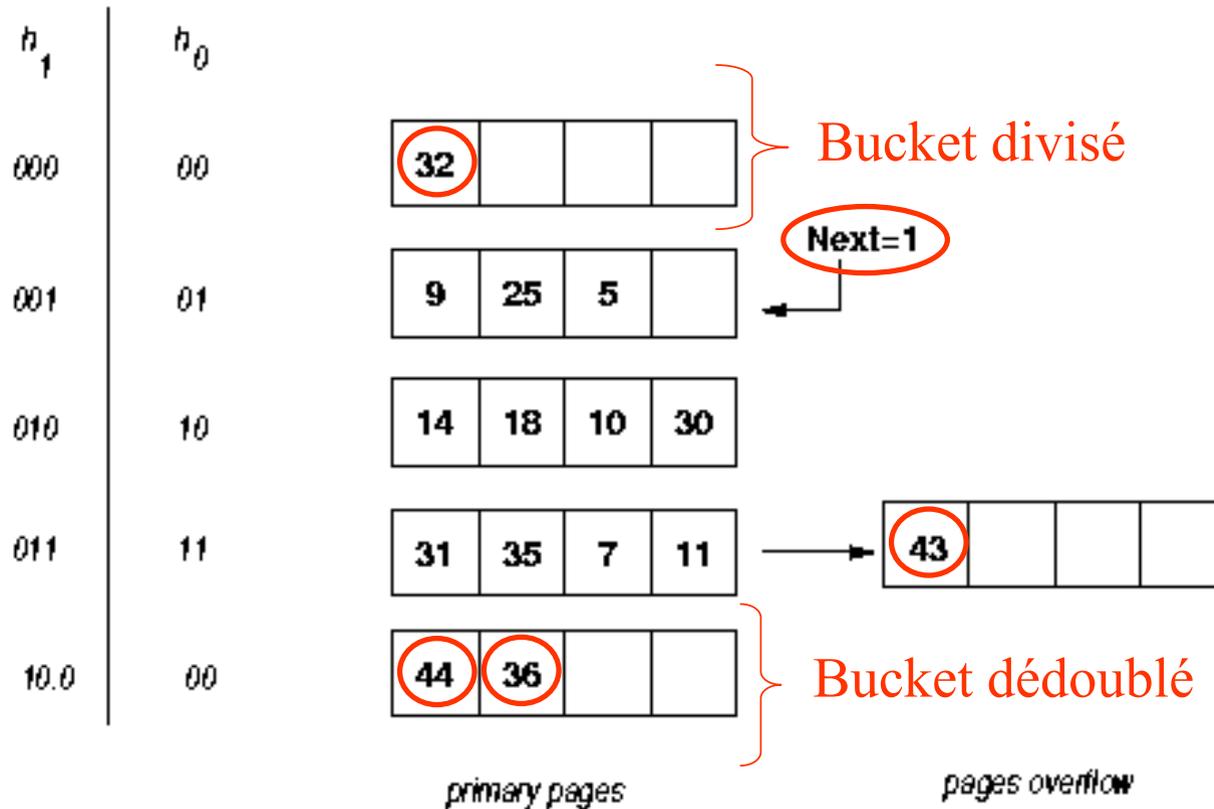
Hachage linéaire (3/6)

level = 0, N=4



Hachage linéaire (3/6)

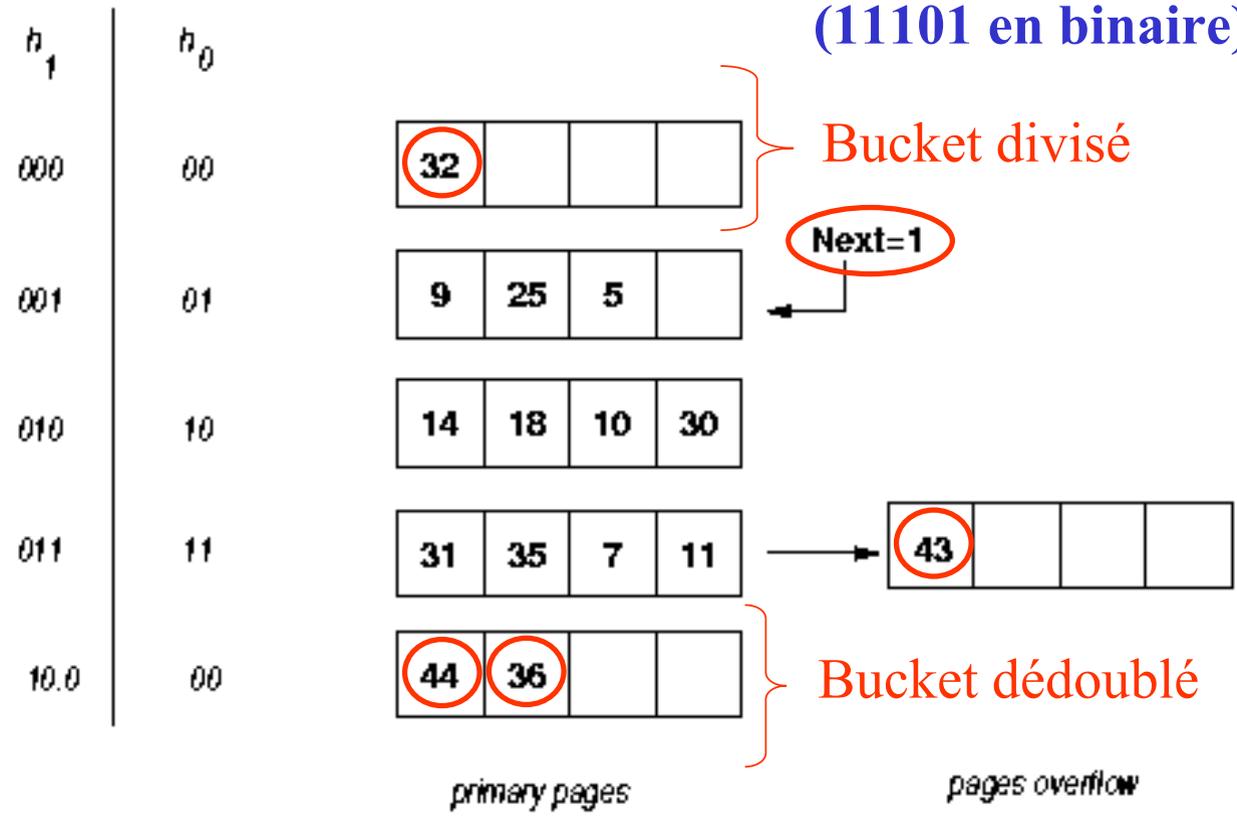
level = 0, N=4



Hachage linéaire (3/6)

Insertion des entrées 37
(100101 en binaire) et 29
(11101 en binaire)

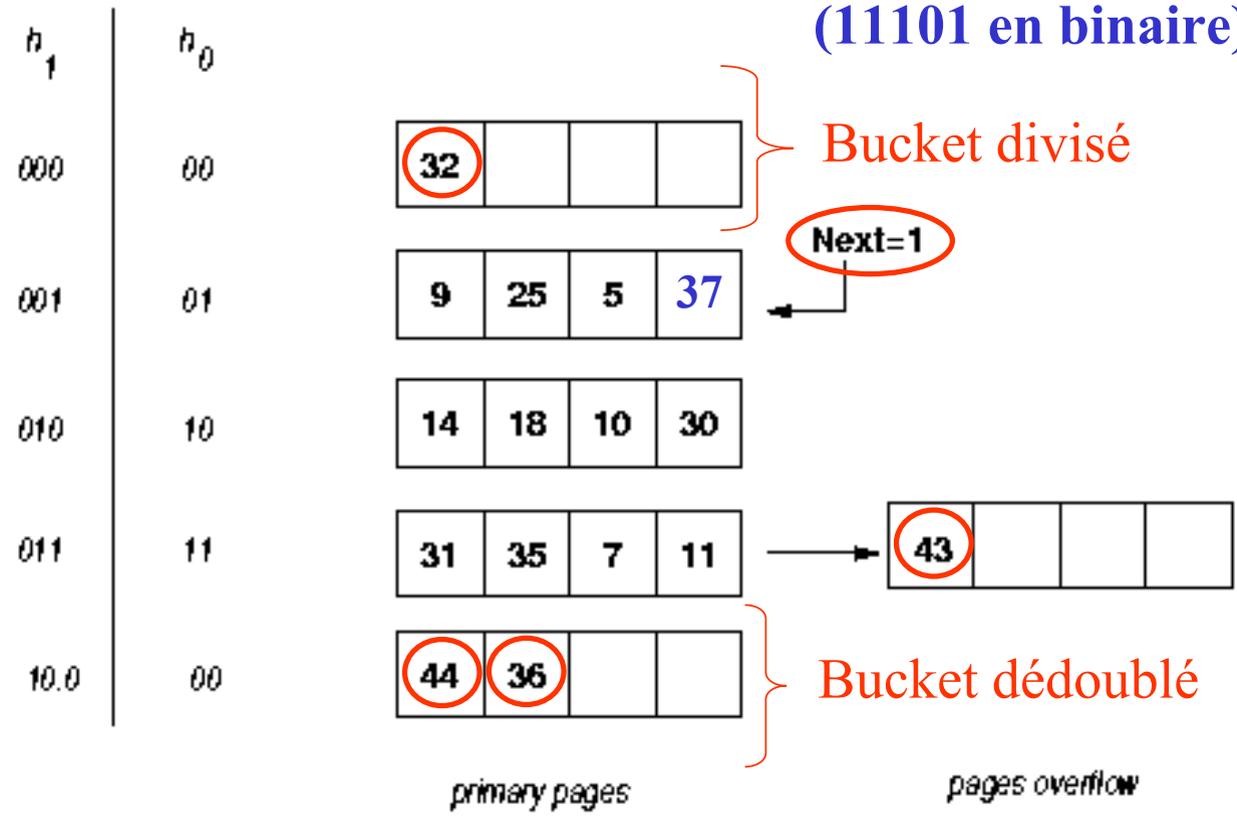
level = 0, N=4



Hachage linéaire (3/6)

Insertion des entrées 37
(100101 en binaire) et 29
(11101 en binaire)

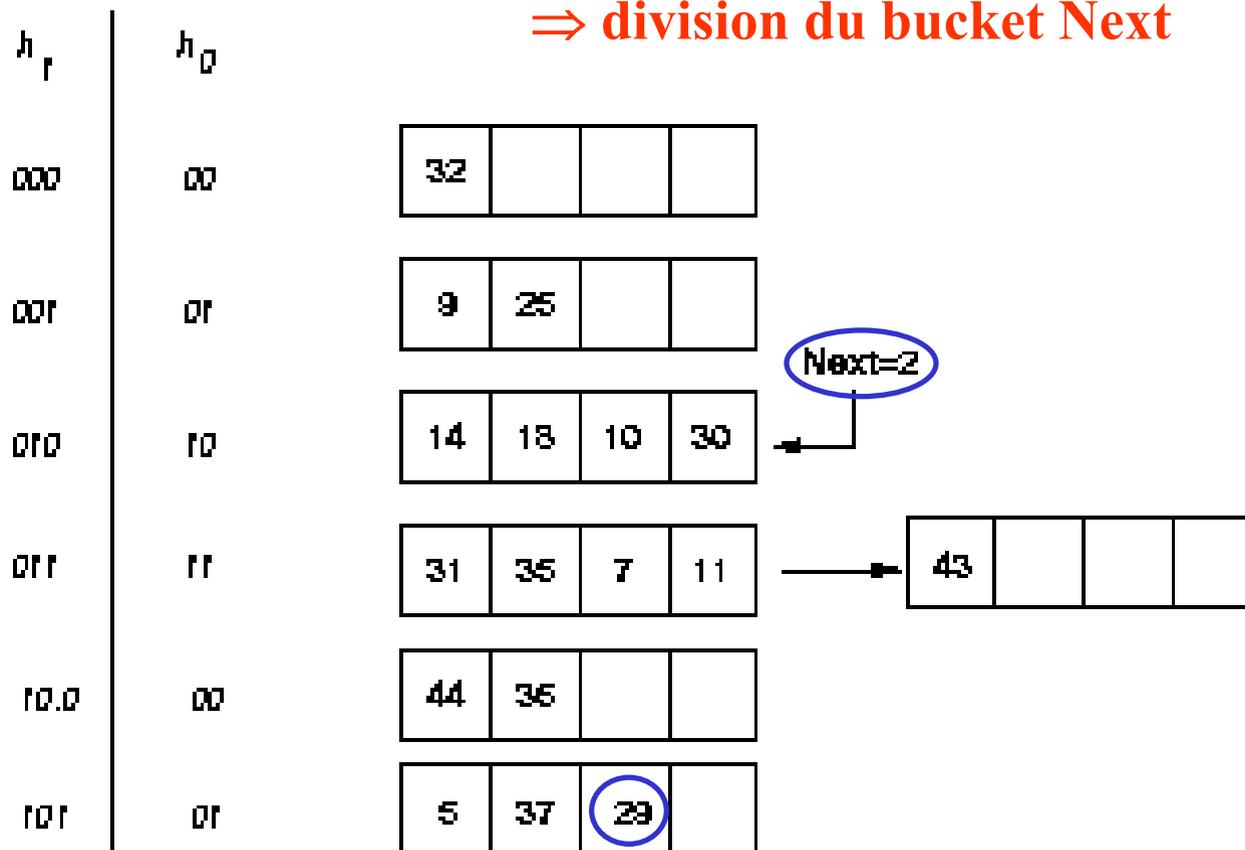
level = 0, N=4



Hachage linéaire (4/6)

level = 0. N=4

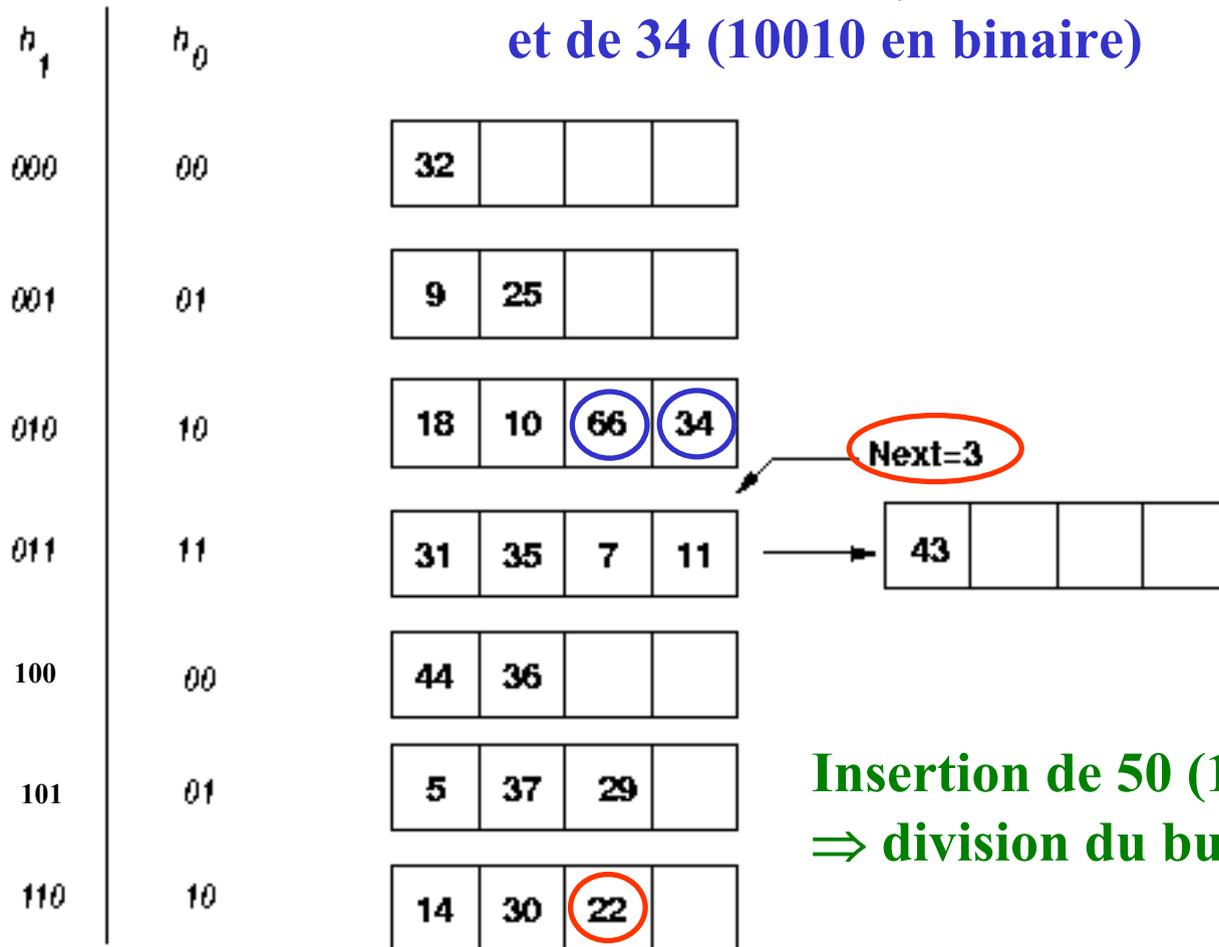
Insertion de 22 (10110 en binaire)
 ⇒ division du bucket Next



Hachage linéaire (5/6)

level = 0, N=4

Insertion de 66 (100010 en binaire)
et de 34 (10010 en binaire)



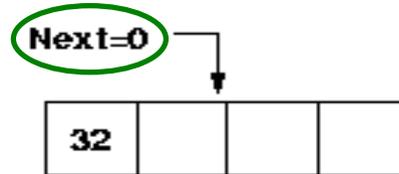
Insertion de 50 (110010)
⇒ division du bucket Next

Hachage linéaire (6/6)

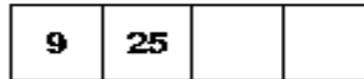
level = 1, N=8

h_1

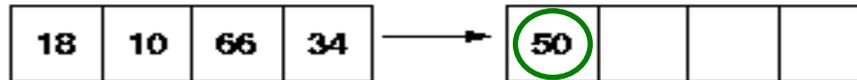
000



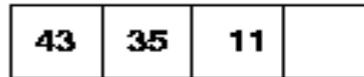
001



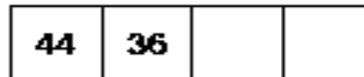
010



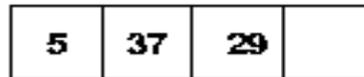
011



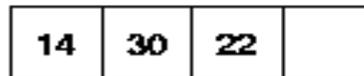
100



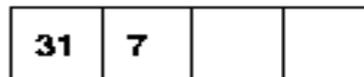
101



110



111



Après la division du bucket $Next = N-1$
 $\Rightarrow N = N*2$ et $Next = 0$

Index (1/4)

- **3 alternatives**

- Les **entrées de clé de recherche** k sont les enregistrements mêmes

- Les entrées sont des couples (k, rid)

- Les entrées sont des couples $(k, liste_rid)$

- **Index primaire**

- Clé de recherche = clé primaire de la relation**

- **Index secondaire**

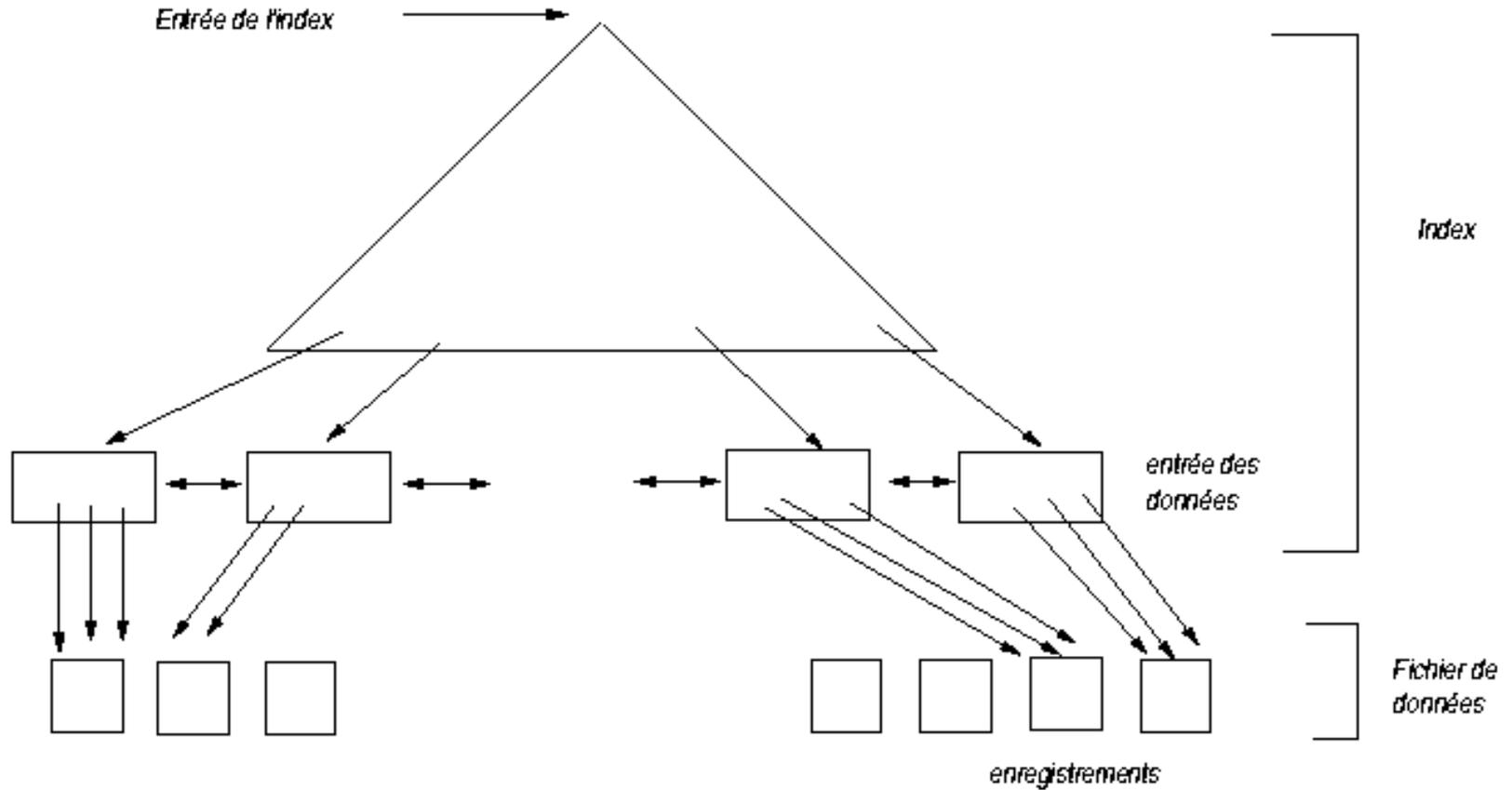
- *(clé de recherche, valeur(s) de clé primaire)*

- *(clé de recherche, pointeur(s) vers les pages du fichier)*

- ⇒ **l'index primaire doit être lu après l'index secondaire**

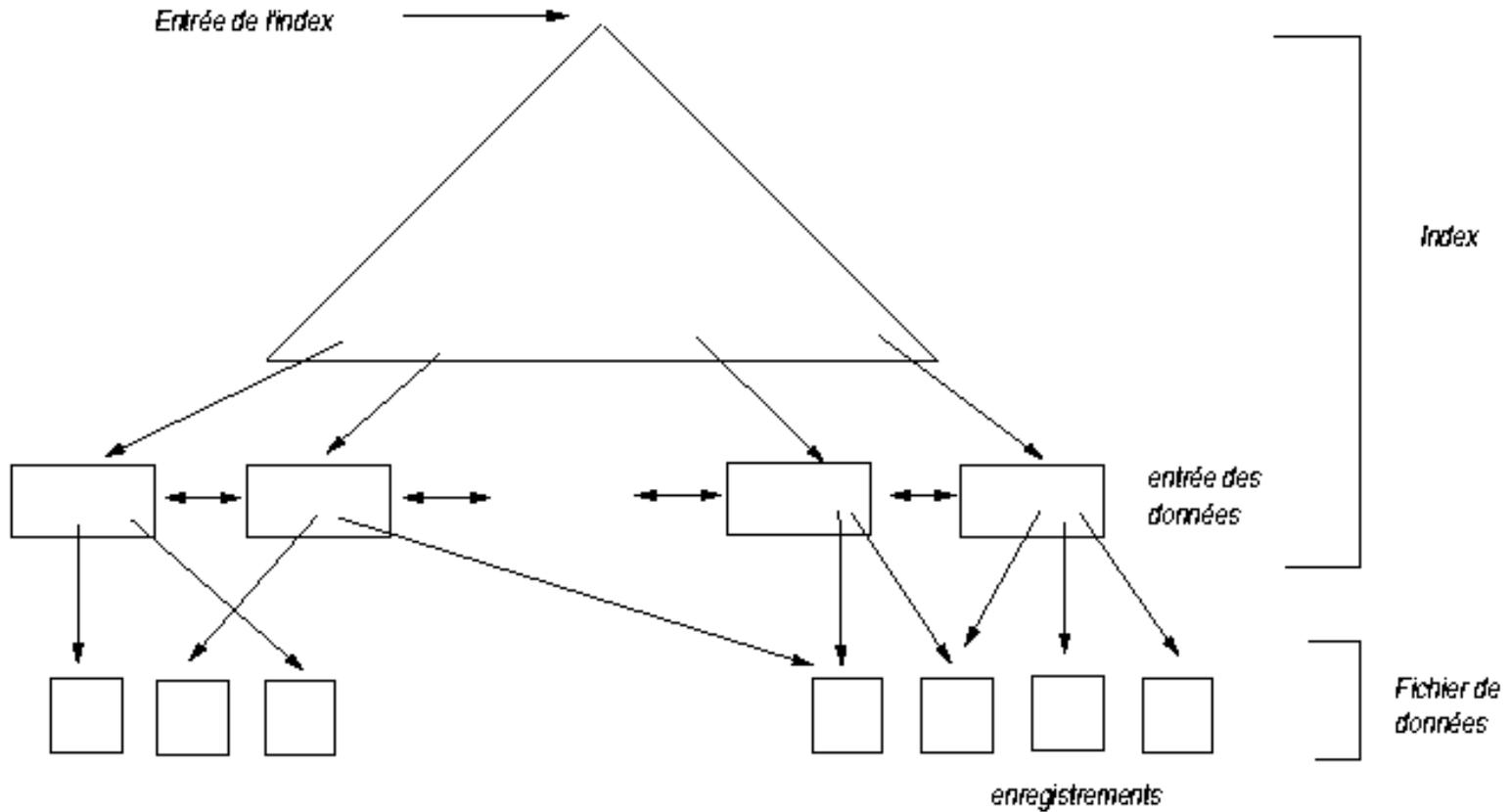
Index (2/4)

- **Clustered index**



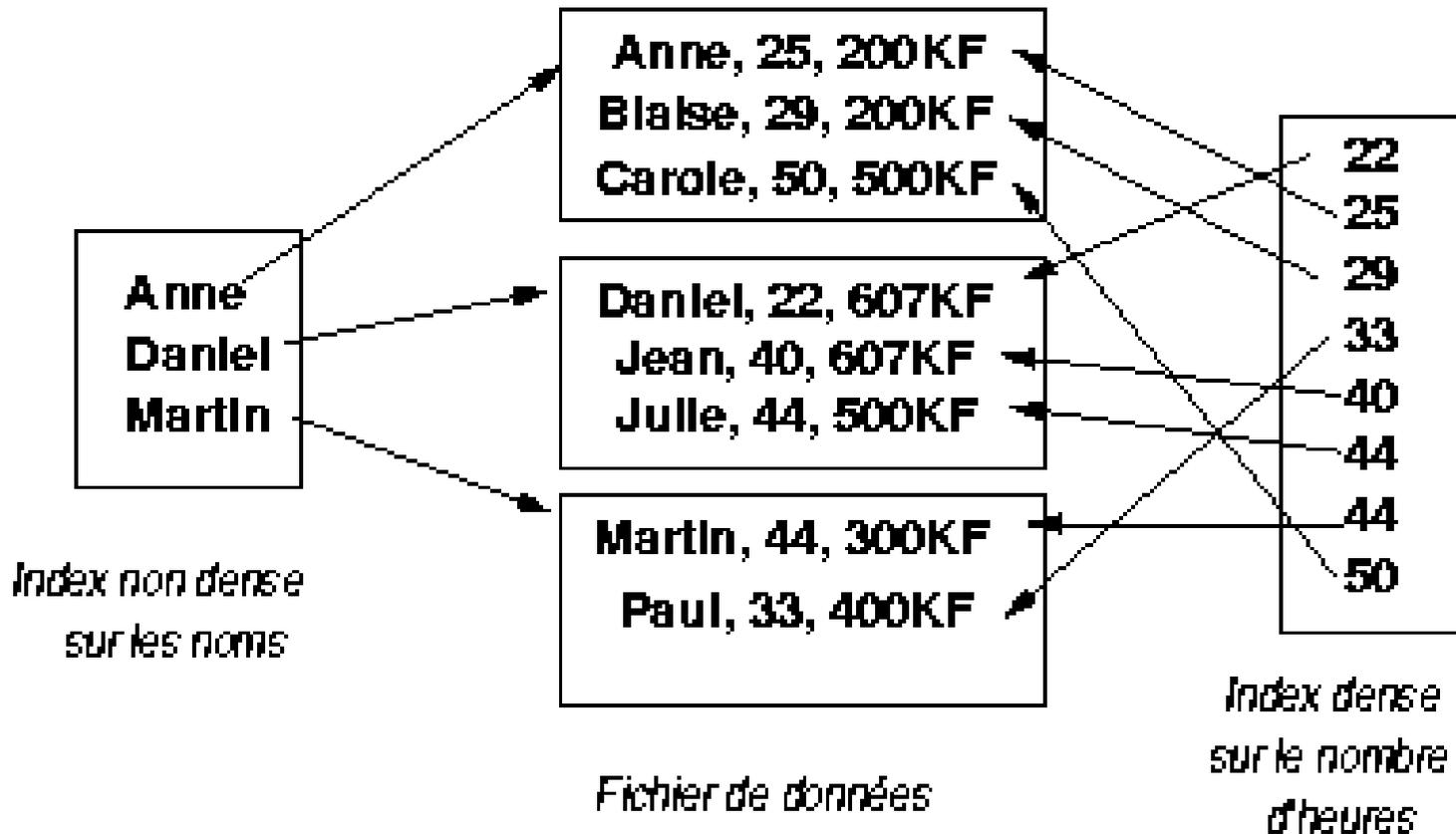
Index (3/4)

- Unclustered index



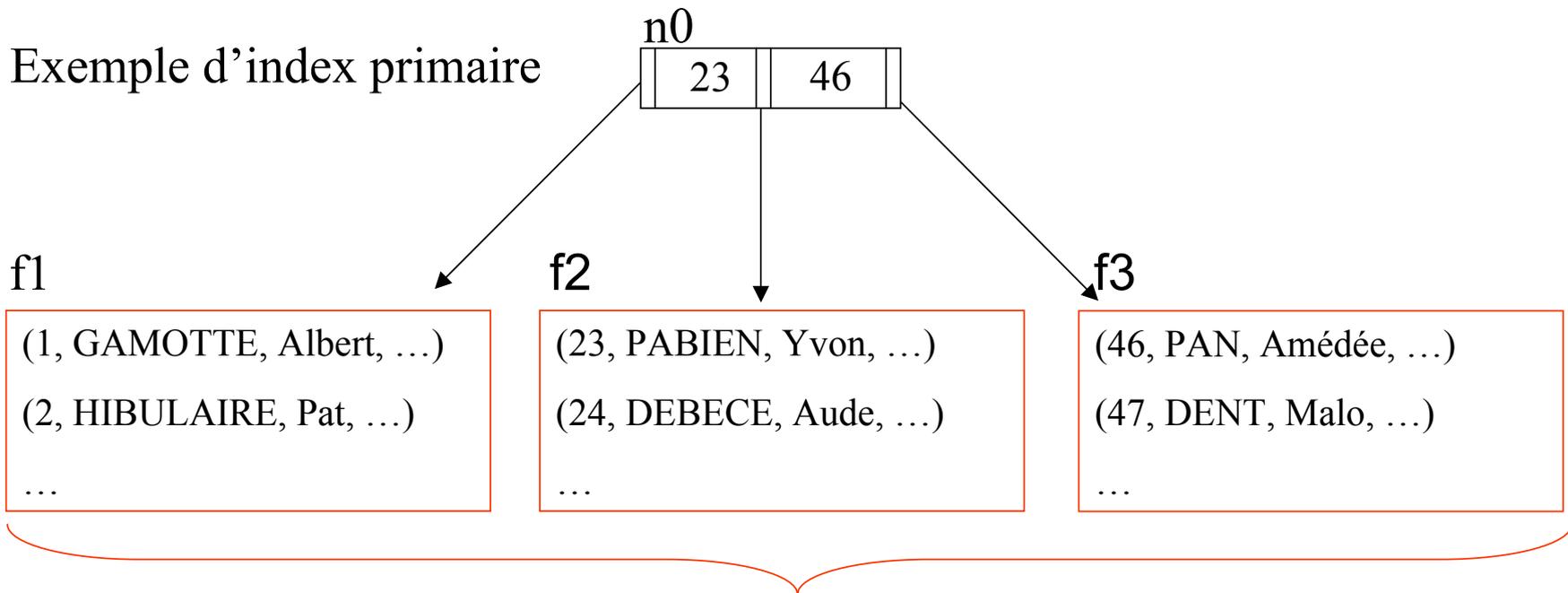
Index (4/4)

- **Index dense / non dense (*sparse*)**





Index basé sur les structures arborescentes : Arbre B+ (1/3)

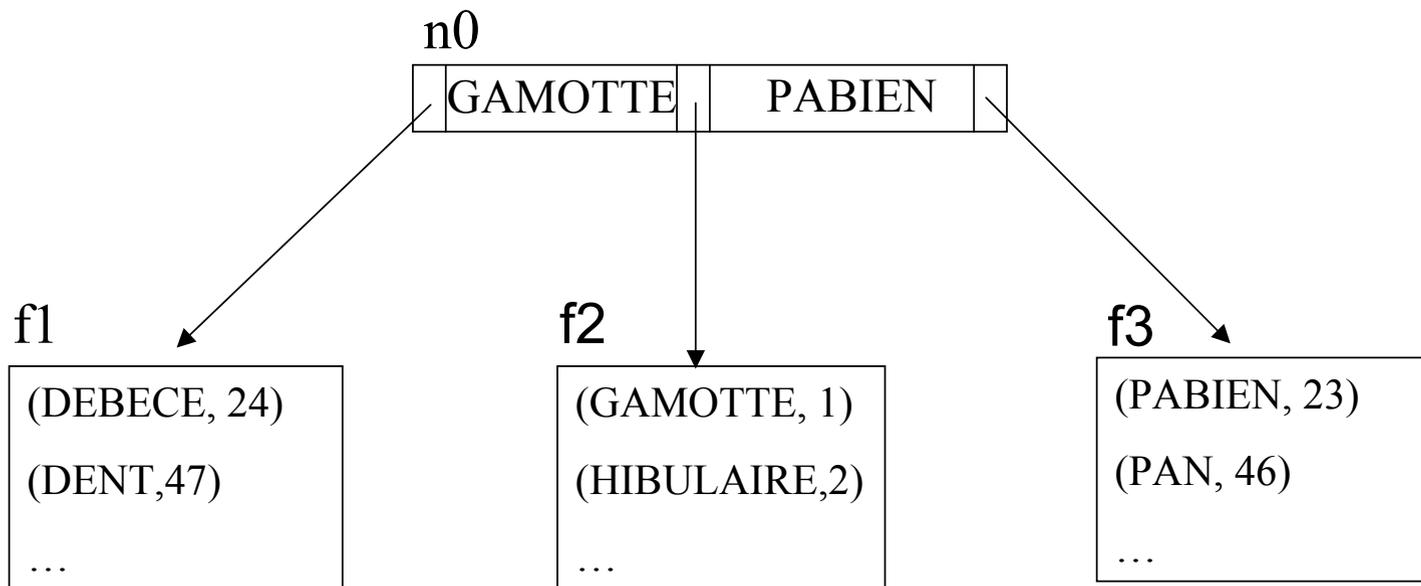


Pages de la relation et feuilles de l'index



Arbre B+ (2/3)

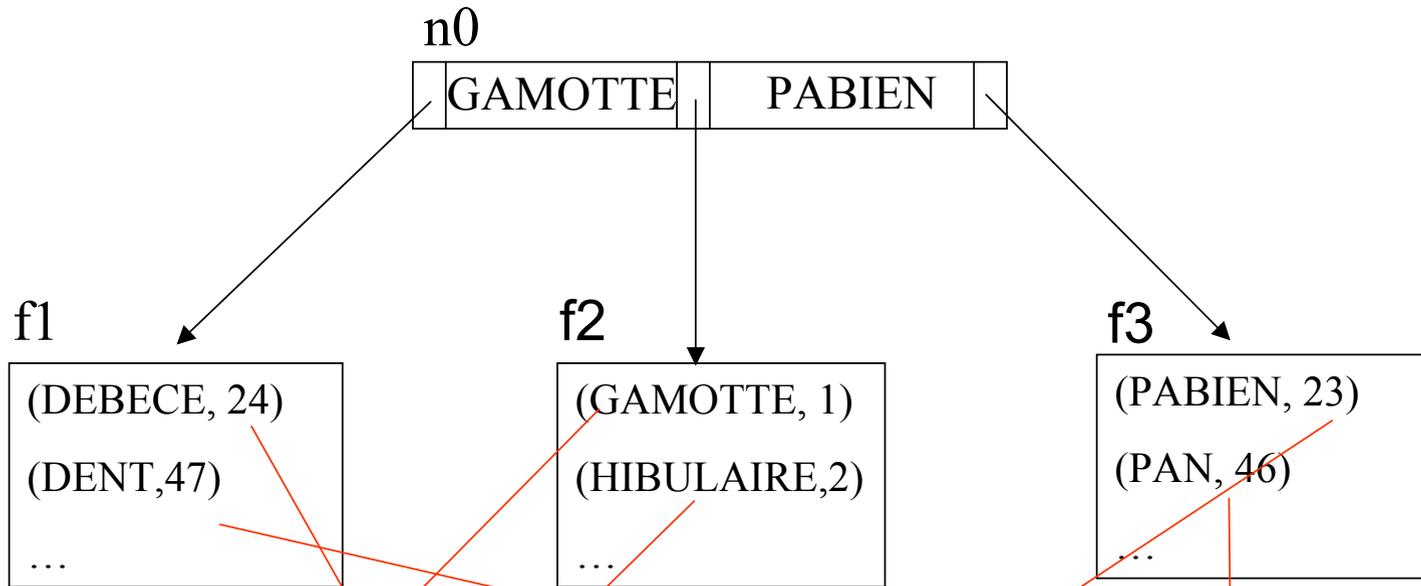
Exemple d'index secondaire





Arbre B+ (2/3)

Exemple d'index secondaire



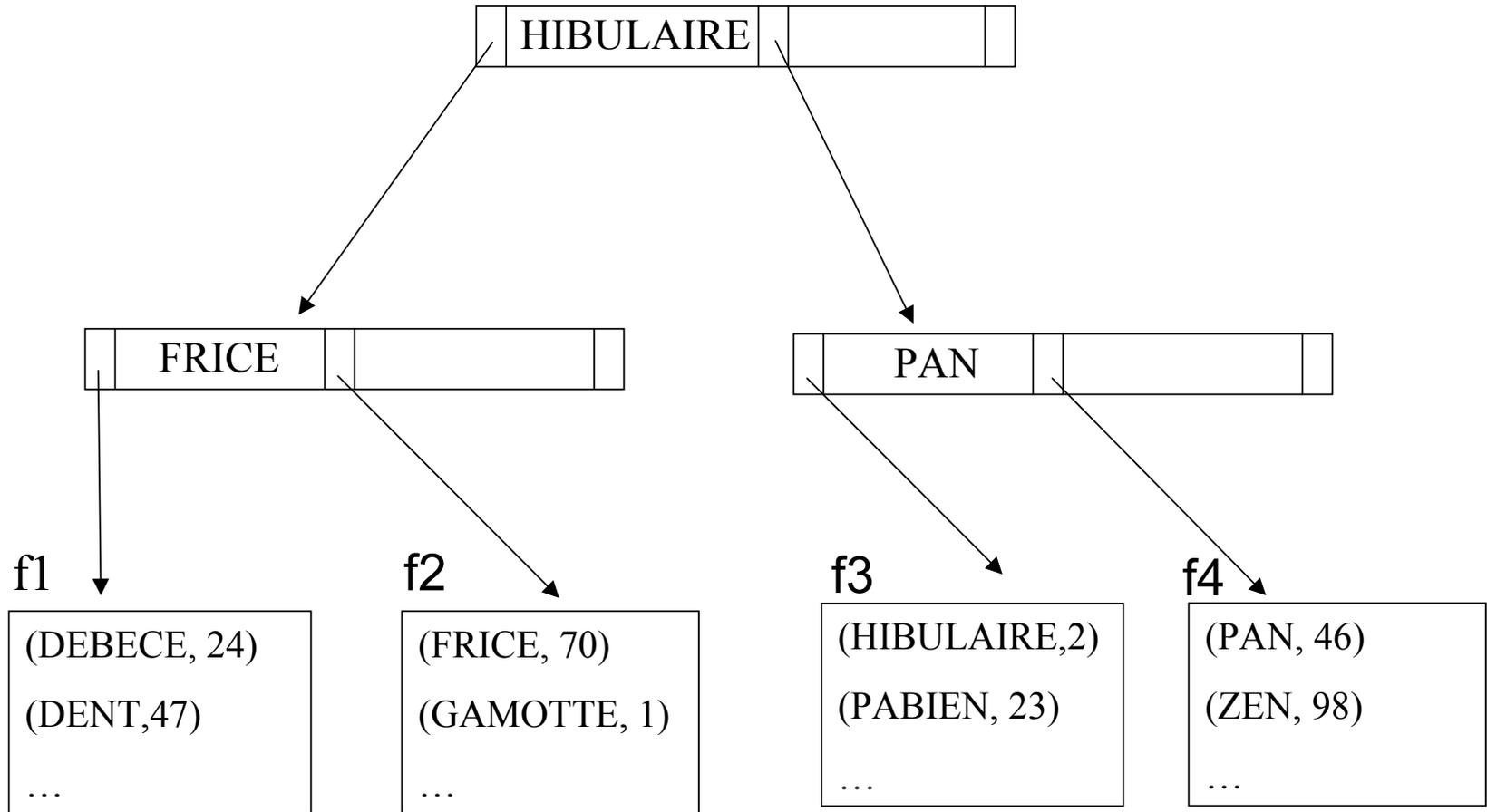
Pages de la relation

(1, GAMOTTE, Albert, ...)
(2, HIBULAIRE, Pat, ...)
...

(23, PABIEN, Yvon, ...)
(24, DEBECE, Aude, ...)
...

(46, PAN, Amédée, ...)
(47, DENT, Malo, ...)
...

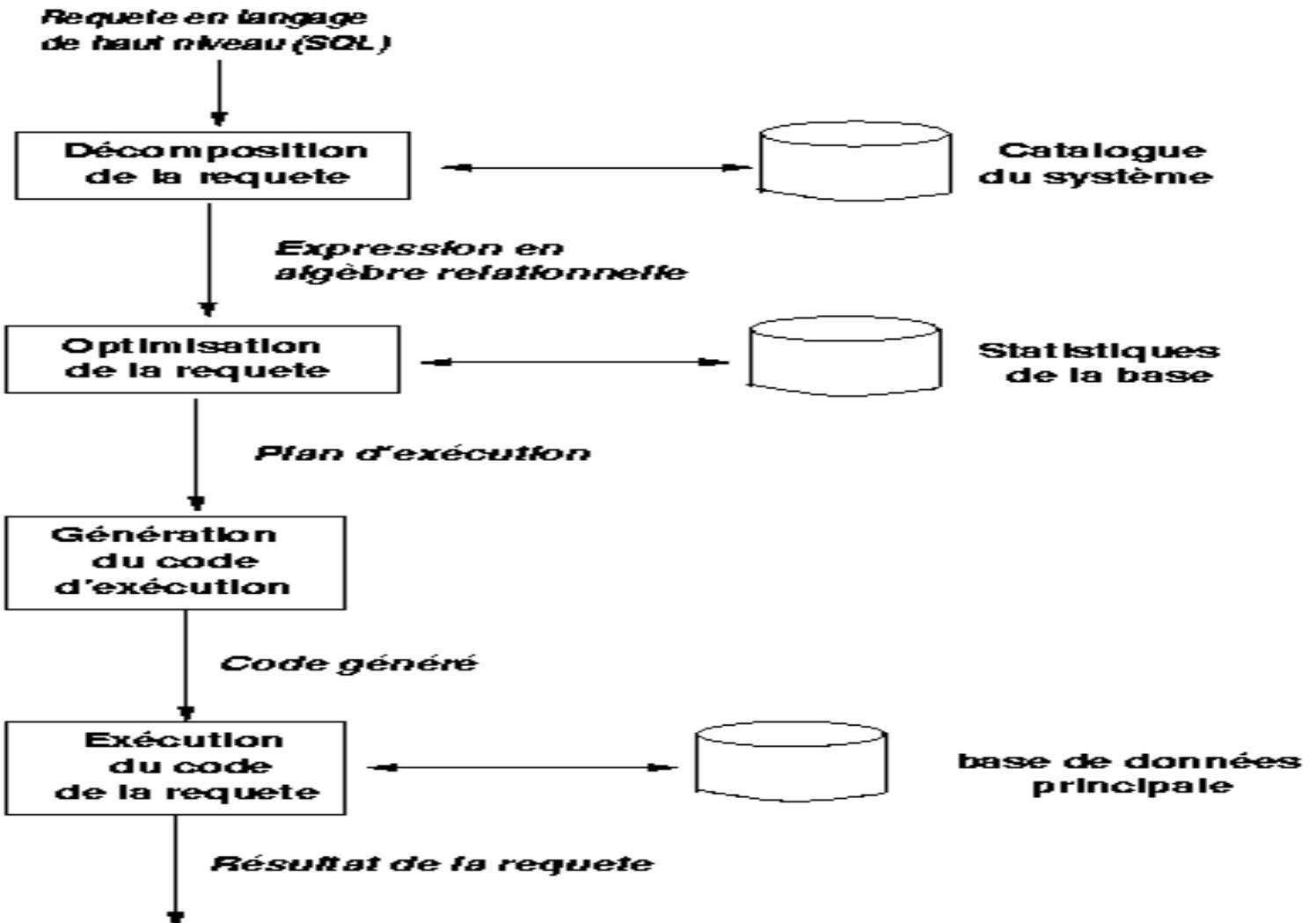
Arbre B+ (3/3)



Chap. III - Optimisation de requêtes

- **Exécution de requête** : séries d'opérations permettant d'extraire des données de la base
- **Optimisation de requête** : activité permettant de choisir la meilleure stratégie d'exécution d'une requête

Phases d'exécution d'une requête



Exemple

"Quels sont les noms de commerciaux basés dans les filiales de Londres ? »

```
SELECT e.Nom
FROM Employe e, Filiale f
WHERE e.#Filiale=f.#Filiale
      AND e.Position = 'Commercial'
      AND f.Ville='Londres'
```

*Employe contient 1000 nuplets, Filiale en contient 50
Il y a 50 commerciaux et 5 filiales à Londres*

- Trois requêtes possibles en algèbre relationnelle
- Calcul du coût de chaque requête en terme E/S

Phase 1 : Décomposition

Transformation de la requête SQL en une requête en algèbre relationnelle

- Vérification syntaxique et sémantique de la requête
- Utilisation du **catalogue** du système
- Représentation de la requête par un **arbre d'opérateurs algébriques**

Catalogue du système (1/2)

- Appelé également **dictionnaire de données**
- Contient la description des données de la base
 - ◆ **Pour chaque relation :**
 - **nom de la relation, identificateur du fichier et structure du fichier**
 - **nom et domaine de chaque attribut**
 - **nom des index**
 - **contraintes d'intégrité**
 - ◆ **Pour chaque index :**
 - **nom et structure de l'index**
 - **attribut appartenant à la clé de recherche**
 - ◆ **Pour chaque vue :**
 - **nom de la vue**
 - **définition de la vue**

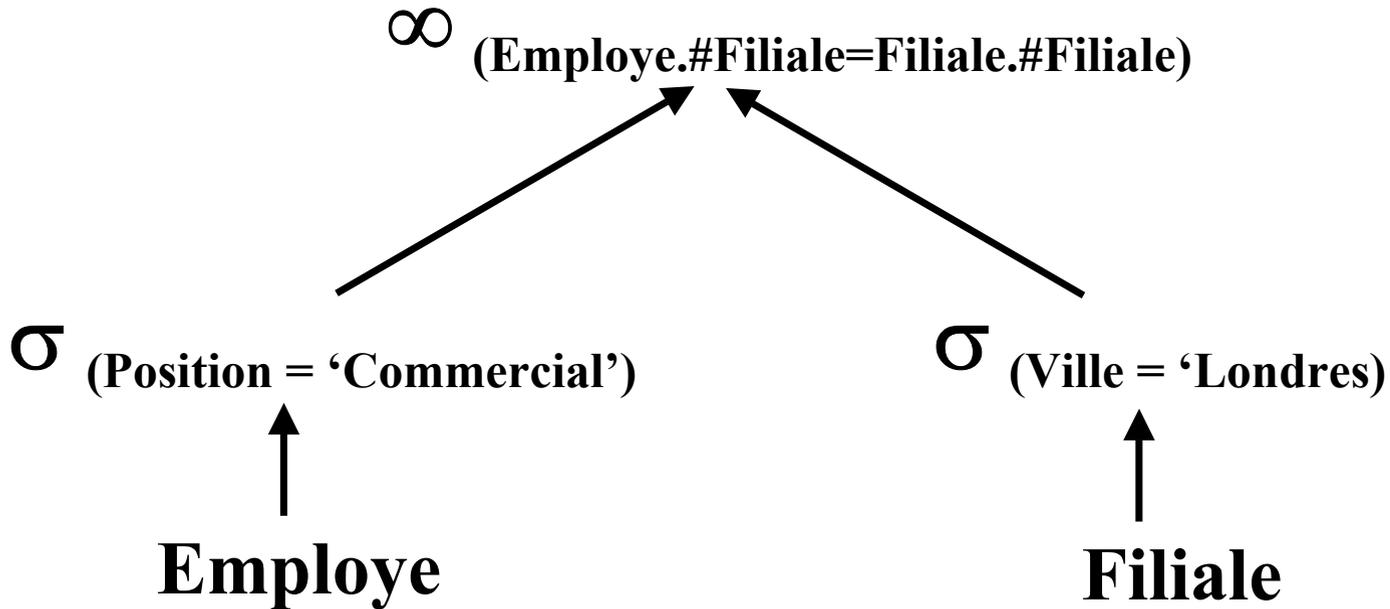
Catalogue du système (2/2)

- Contient également des données statistiques
 - ◆ **Cardinalité de chaque relation**
 - ◆ **Nombre de pages de chaque relation**
 - ◆ **Nombre de valeurs distinctes de clé de recherche pour chaque index**
 - ◆ **Hauteur des index de structures arborescente**
 - ◆ **Valeur minimum et valeur maximum de chaque clé de recherche dans chaque index**
- Exemple sous Oracle8 : USER_ALL_TABLES, USER_CONSTRAINTS etc.

Arbre algébrique (1/2)

- **Représentation des relations impliquées dans la requête par les nœuds feuille de l'arbre**
- **Représentation des résultats intermédiaires par des nœuds non feuille**
- **Représentation du résultat de la requête par la racine de l'arbre**
- **Ordre des séquences d'opérations : des feuilles vers la racine**

Arbre algébrique (2/2)



Phase 2 : Optimisation

Equivalences d'expressions (1/3)

1) Cascade de sélections : $\sigma_{p \wedge q \wedge r}(\mathbf{R}) =$

2) Commutativité des sélections : $\sigma_p(\sigma_q(\mathbf{R})) =$

3) Séquence de projections : $\Pi_L(\Pi_M(\dots \Pi_N(\mathbf{R}))) =$

4) Commutativité des sélections et des projections :

$$\Pi_{A_1 \dots A_n} \sigma_p(\mathbf{R}) =$$

5) Commutativité des jointures : $\mathbf{R} \bowtie_p \mathbf{S} =$

6) Commutativité des jointures et des sélections

$$\sigma_p(\mathbf{R} \bowtie_p \mathbf{S}) = \quad \text{et} \quad \sigma_p(\mathbf{R} * \mathbf{S}) =$$

Equivalence d'expressions (2/3)

7) Commutativité des jointures et des projections

$$\Pi_{L_1 \cup L_2}(\mathbf{R} \bowtie_p \mathbf{S}) =$$

8) Commutativité des unions et des intersections

$$(\mathbf{R} \cup \mathbf{S}) = \quad \text{et } (\mathbf{R} \cap \mathbf{S}) =$$

9) Commutativité des unions, intersections, différences et des sélections

$$\sigma_p(\mathbf{R} \cup \mathbf{S}) =$$

$$\sigma_p(\mathbf{R} \cap \mathbf{S}) =$$

$$\sigma_p(\mathbf{R} - \mathbf{S}) =$$

Equivalence d'expressions (3/3)

10) Commutativité des projections et des unions

$$\Pi_L(\mathbf{R} \cup \mathbf{S}) =$$

11) Associativité des jointures

$$(\mathbf{R} \bowtie \mathbf{S}) \bowtie \mathbf{T} =$$

12) Associativité des unions et des intersections

$$(\mathbf{R} \cup \mathbf{S}) \cup \mathbf{T} =$$

$$(\mathbf{R} \cap \mathbf{S}) \cap \mathbf{T} =$$

Transformation d'un arbre algébrique

- ① **Division des conjonctions de sélections**
- ② **Ré-ordonnement des sélections en utilisant les règles 2 et 4**
- ③ **Application des sélections les plus sélectives en premier**
- ④ **Transformation des produits cartésiens en jointure**
- ⑤ **Ré-ordonnement des équi-jointures en utilisant la règle 11**
- ⑥ **Déplacement des projections et création de nouvelles projections en utilisant les règles 4 et**

Π Nom
▲

```
SELECT Nom
FROM Employe, Equipe, Projet
WHERE Nom_Projet = 'Sirius'
AND #Projet = #Projet_Equipe
AND #Equipe = #Appartenance
AND DaeNais=1973
```

Π Nom
▲

```
SELECT Nom
FROM Employe, Equipe, Projet
WHERE Nom_Projet = 'Sirius'
AND #Projet = #Projet_Equipe
AND #Equipe = #Appartenance
AND DaeNais=1973
```

Π_{Nom}
▲

```
SELECT Nom
FROM Employe, Equipe, Projet
WHERE Nom_Projet = 'Sirius'
AND #Projet = #Projet_Equipe
AND #Equipe = #Appartenance
AND DaeNais=1973
```

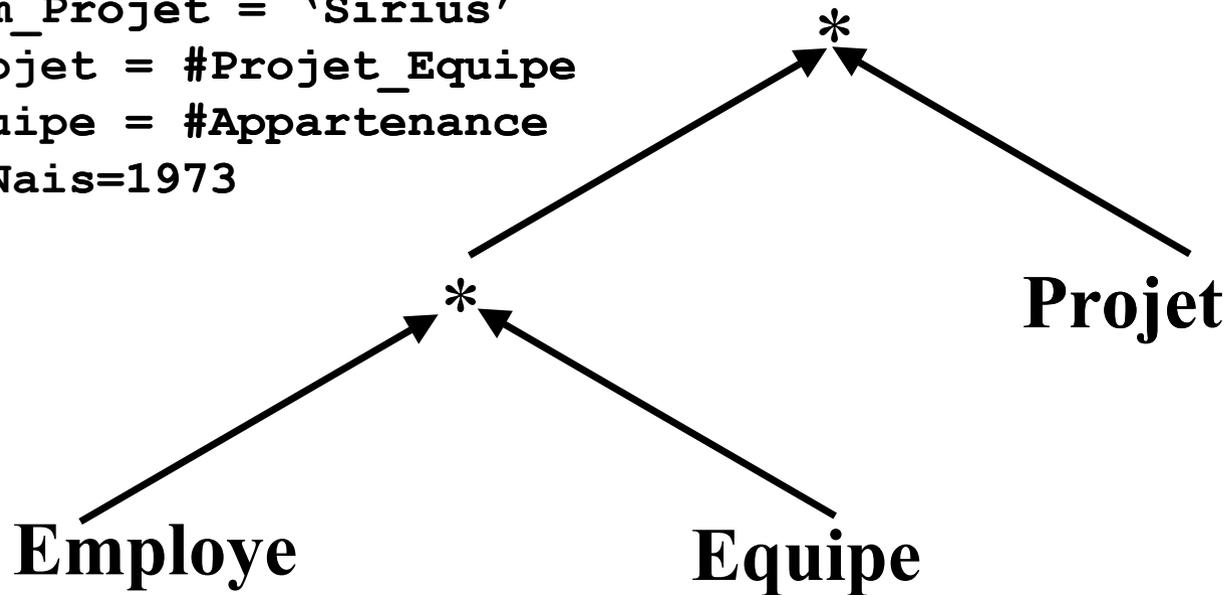
Projet

Employe

Equipe

Π_{Nom}
▲

```
SELECT Nom  
FROM Employe, Equipe, Projet  
WHERE Nom_Projet = 'Sirius'  
AND #Projet = #Projet_Equipe  
AND #Equipe = #Appartenance  
AND DaeNais=1973
```



Π_{Nom}
▲

SELECT Nom

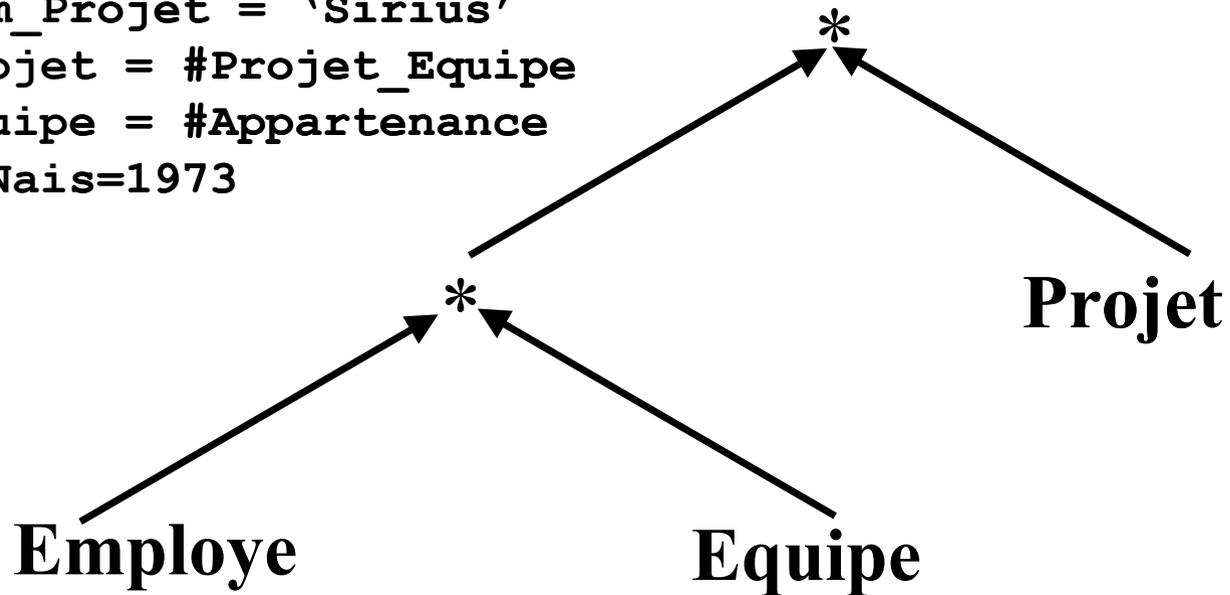
FROM Employe, Equipe, Projet

WHERE Nom_Projet = 'Sirius'

AND #Projet = #Projet_Equipe

AND #Equipe = #Appartenance

AND DaeNais=1973



Π_{Nom}



$\sigma_{(\text{Nom_Projet}=\text{'Siruis'}) \wedge (\#\text{Projet}=\#\text{Projet_Equipe}) \wedge (\#\text{Equipe}=\#\text{Appartenance}) \wedge (\text{DateNais}=1973)}$

SELECT Nom

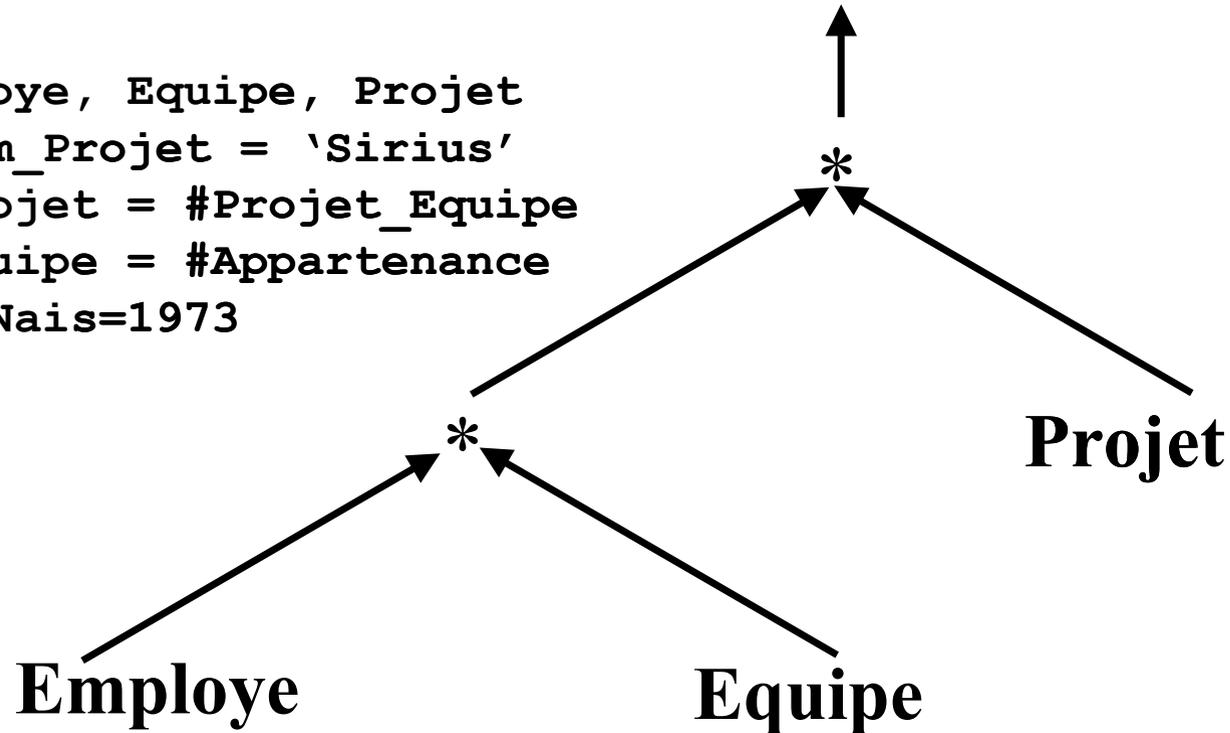
FROM Employe, Equipe, Projet

WHERE Nom_Projet = 'Sirius'

AND #Projet = #Projet_Equipe

AND #Equipe = #Appartenance

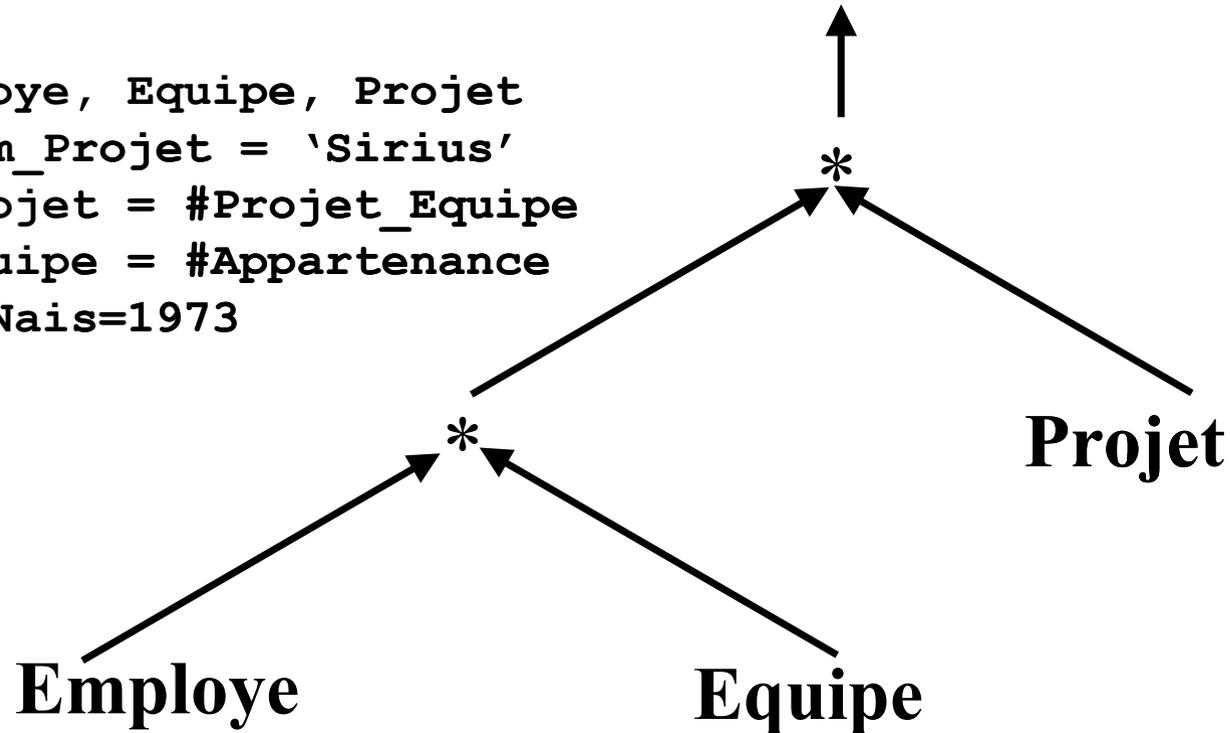
AND DaeNais=1973



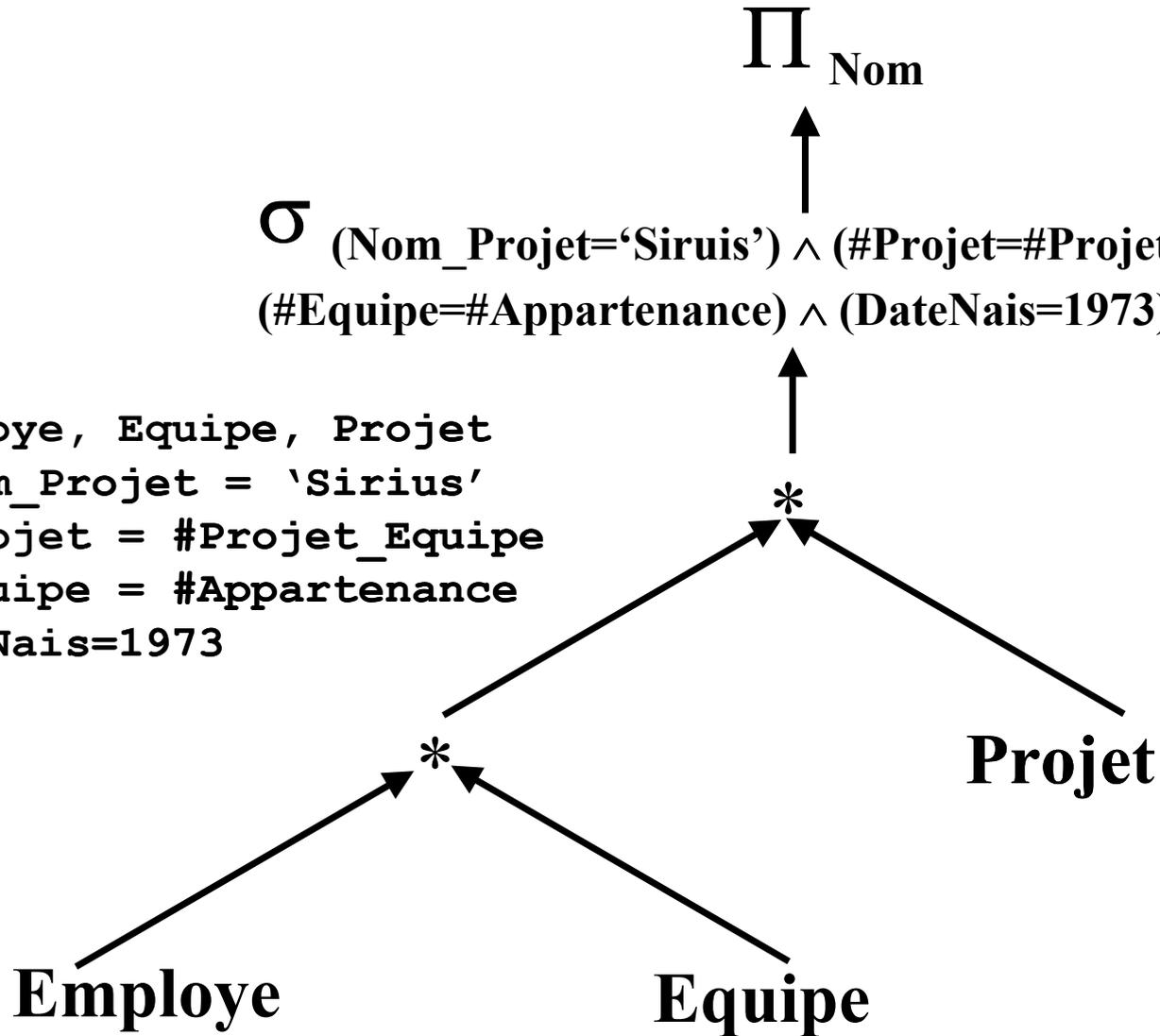
Π_{Nom}
▲

$\sigma_{(\text{Nom_Projet}=\text{'Siruis'}) \wedge (\#\text{Projet}=\#\text{Projet_Equipe}) \wedge (\#\text{Equipe}=\#\text{Appartenance}) \wedge (\text{DateNais}=1973)}$

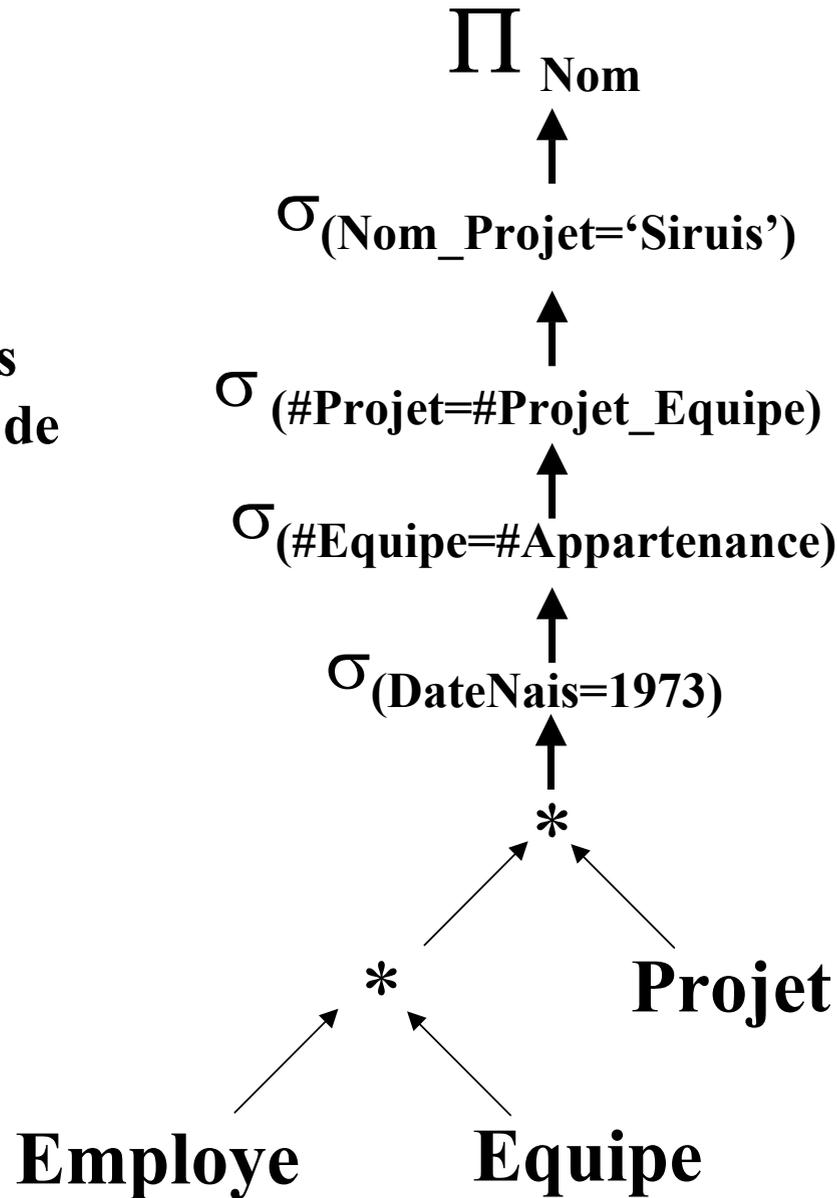
```
SELECT Nom  
FROM Employe, Equipe, Projet  
WHERE Nom_Projet = 'Sirius'  
AND #Projet = #Projet_Equipe  
AND #Equipe = #Appartenance  
AND DaeNais=1973
```



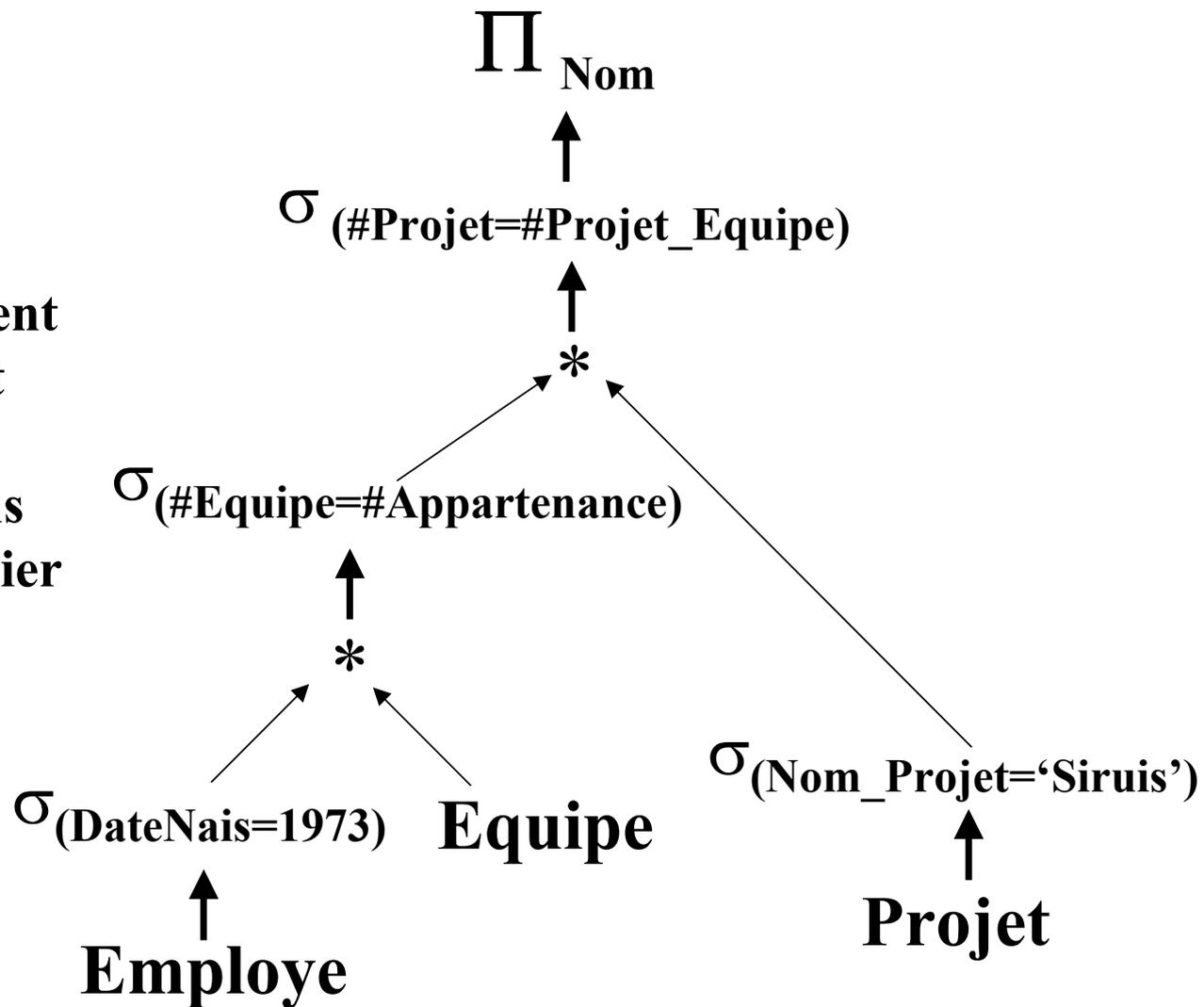
```
SELECT Nom  
FROM Employe, Equipe, Projet  
WHERE Nom_Projet = 'Sirius'  
AND #Projet = #Projet_Equipe  
AND #Equipe = #Appartenance  
AND DaeNais=1973
```



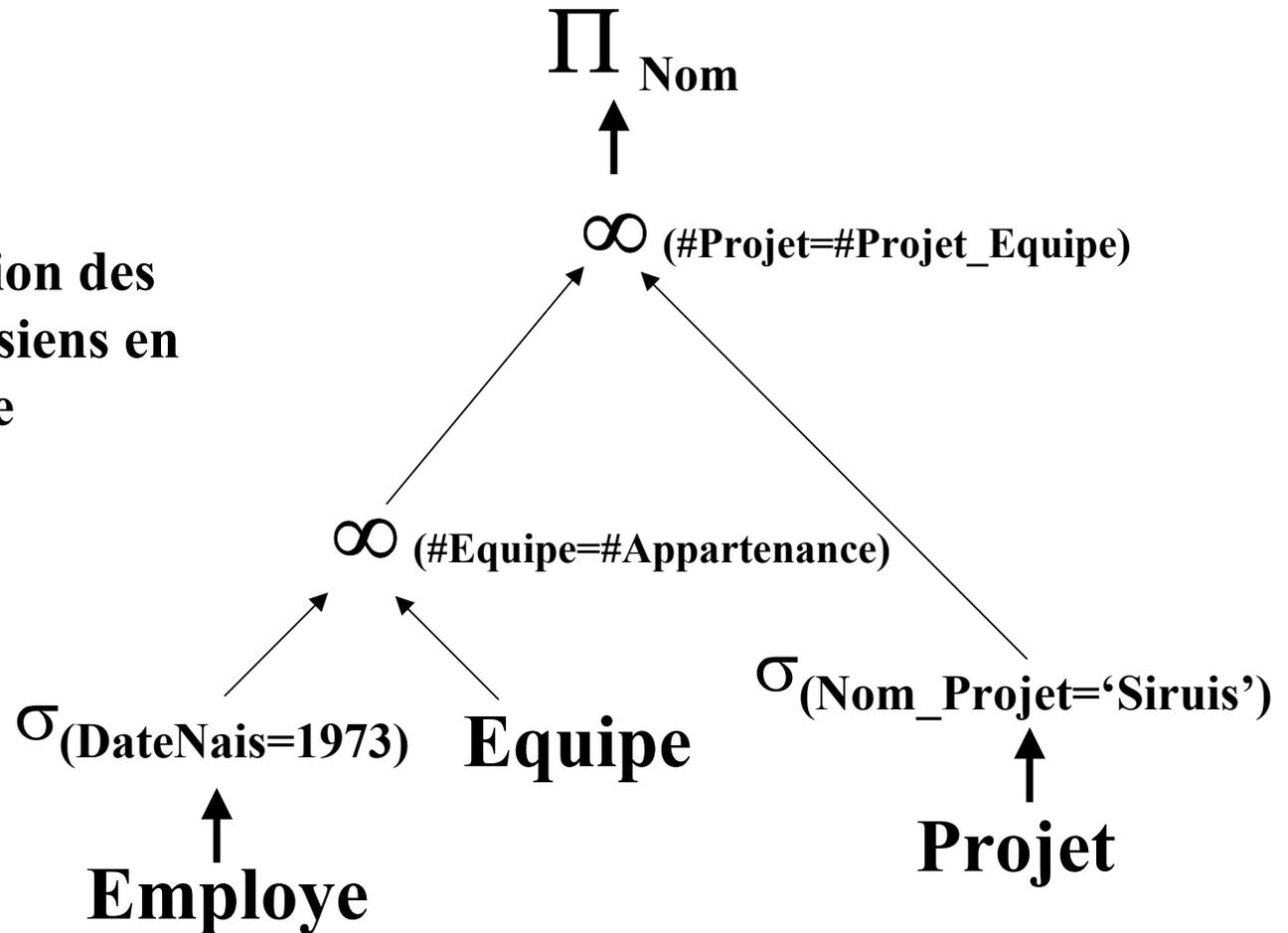
**Division des
conjonctions de
sélections**



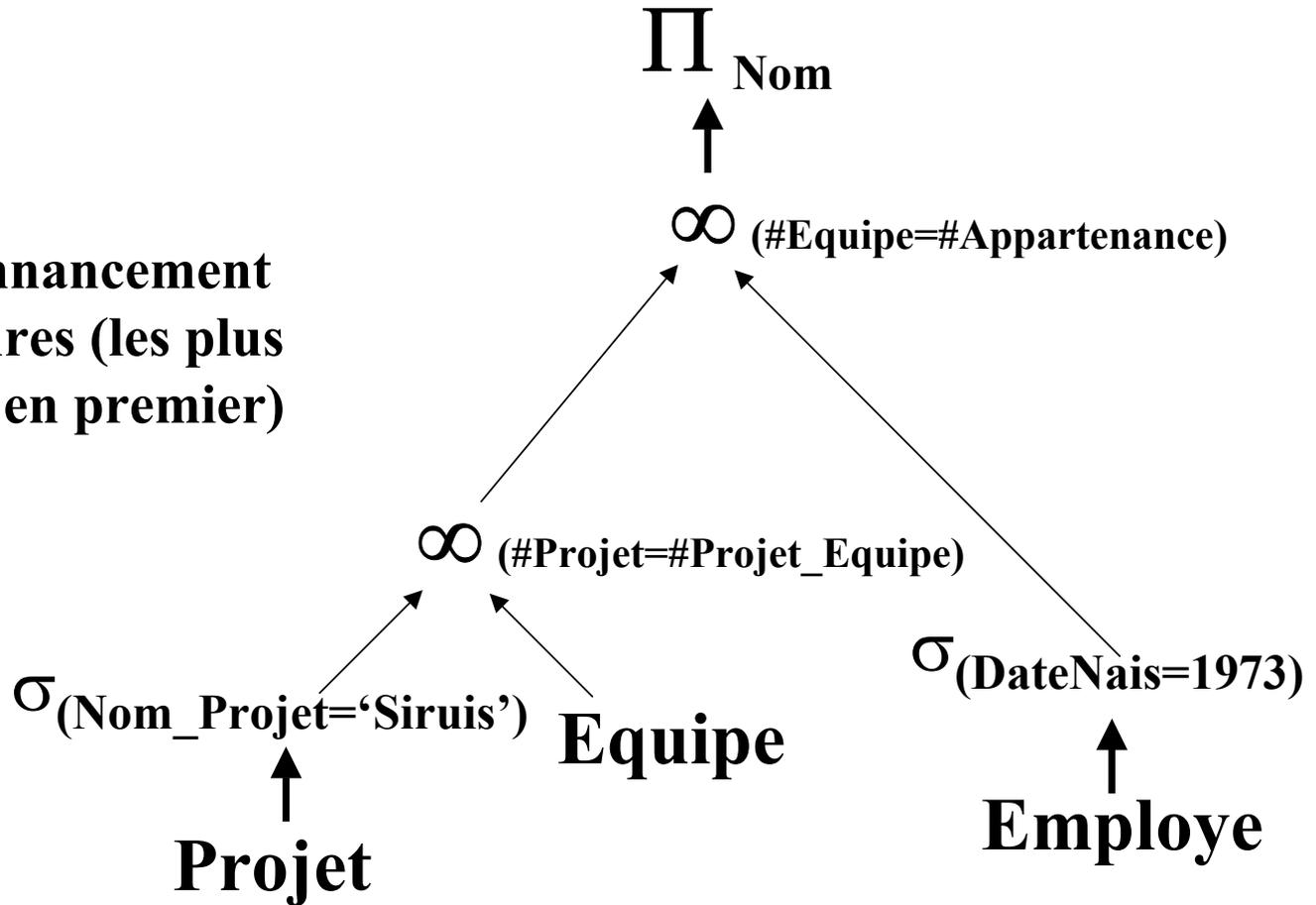
Ré-ordonnancement
des sélections et
application des
sélections les plus
sélectives en premier



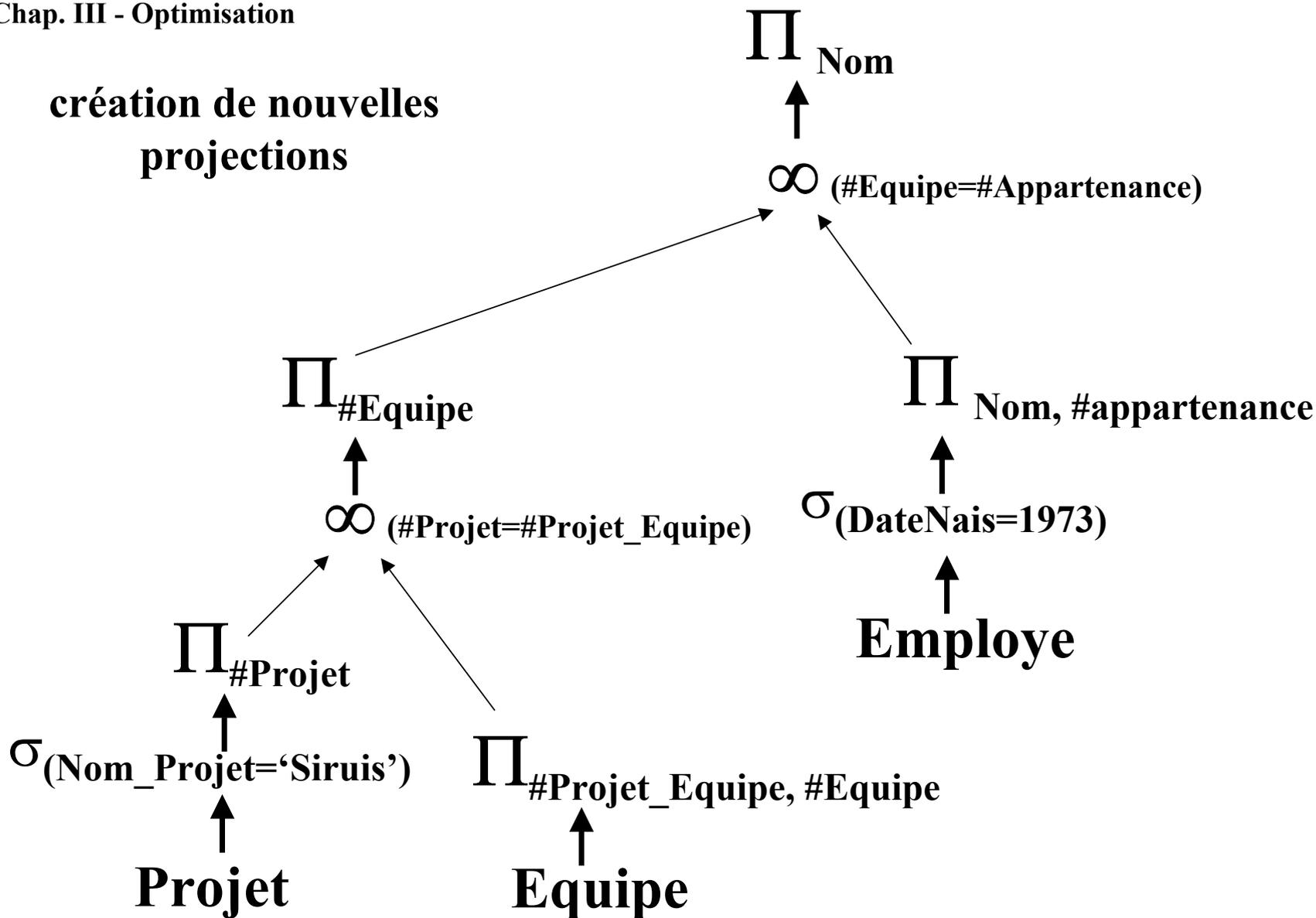
**Transformation des
produits cartésiens en
jointure**



**Ré-ordonnancement
des jointures (les plus
sélectives en premier)**



création de nouvelles projections



Statistiques (1/2)

- **Pour chaque relation R [CBS98] :**

- ◆ $NTuples(R)$: nombre de nuplets

- ◆ $Bfactor(R)$: nombre de nuplets par bloc

- ◆ $NBlocks(R)$: nombre de blocs pour la relation

$$NBlocks(R) =$$

- **Pour chaque attribut A de R :**

- ◆ $NDistinct_A(R)$: nombre de valeurs distinctes de A

- ◆ $Min_A(R)$ et $Max_A(R)$: valeurs min et max de A

- ◆ $SC_A(R)$: nombre moyen de nuplets satisfaisant un prédicat sur A

Statistiques (2/2)

◆ $SC_A(R)$ dépend du prédicat

– Egalité :

– $A > c$

– $A < c$

– $A \in \{c_1, \dots, c_n\}$

– $A \wedge B$

– $A \vee B$

● Pour index I de R sur un attribut A :

◆ $NLevels_A(I)$: nombre de niveaux pour I

◆ $NLBlocks_A(I)$: nombre de blocs utilisés pour I

Coût de $\sigma_p(\mathbf{R})$

Stratégies :

- Recherche linéaire (fichier non ordonné sans index)
- Recherche binaire (fichier ordonné sans index)
- Condition d'égalité sur la valeur d'une fonction de hachage
- Condition d'égalité sur la clé primaire
- Condition d'inégalité sur la clé primaire
- Condition d'égalité sur la clé d'un index secondaire clustered
- Condition d'égalité sur la clé d'un index secondaire unclustered
- Condition d'inégalité sur un index secondaire en arbre B+

Coût de $\sigma_p(R)$

Exemple :

Fonction de hachage *Index clustered*

Employe (NSS, Departement, Salaire, Position ...)

NTuples(e)=3000

BFactor(e)=30

NDistinct_D(e)=500

NDistinct_P(e)=10

NDistinct_S(e)=500 Min_S(e)=1000 Max_S(e)=5000

NLevels_D(I)=2

NLevels_S(I)=2 NBlocks_S(I)=50

Index unclustered

$$\sigma_{NSS} = 273... (e)$$

$$\sigma_{Position = 'Commercial'} (e) \quad \sigma_{Salaire > 2000} (e)$$

$$\sigma_{Departement = 'RH'} (e) \quad \sigma_{(Position = 'Cadres') \wedge (Departement = 'R \& D')} (e)$$

Coût de $R \bowtie_F S$

Stratégies d'implantations de la jointure :

- **Boucles imbriquées (*Block nested loop join*)**
- **Boucles imbriquées en utilisant un index (*Indexed nested loop join*)**
- **Tri-fusion (*Sort-merge join*)**
- **Hachage (*Hash join*)**

Pour chaque algorithme, possibilité de le faire en **multi-passes**

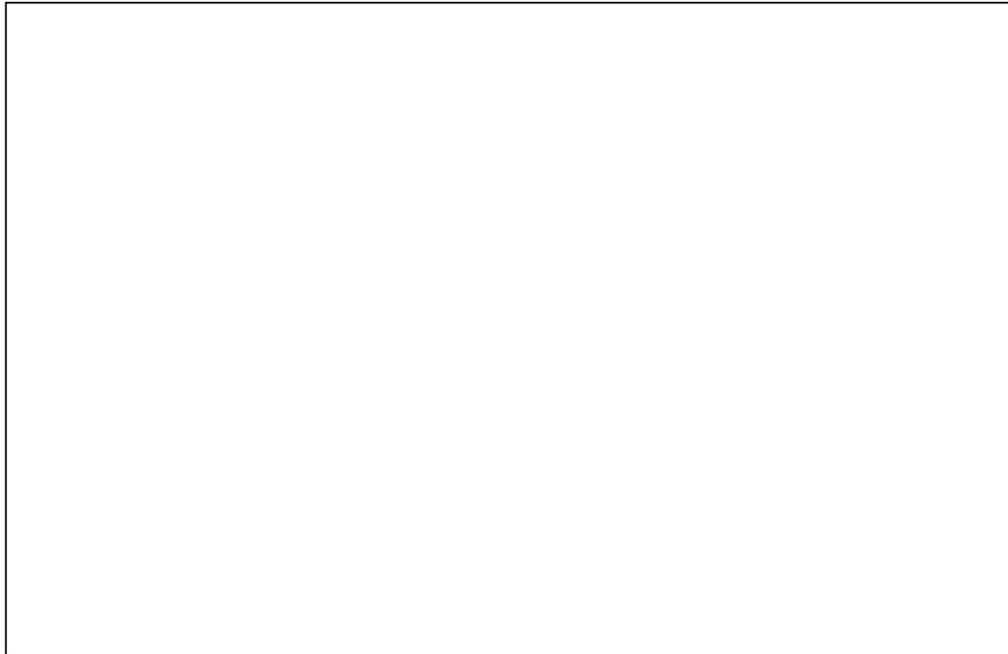
Cardinalité de la $R \bowtie_F S$

- $NTuples(R \bowtie_F S) \leq NTuples(R) * NTuples(S)$
- Lorsque le prédicat est $R.A=S.B$
 - ◆ Si A est clé de R : $NTuples(R \bowtie_F S) \leq NTuples(S)$
 - ◆ Si B est clé de S : $NTuples(R \bowtie_F S) \leq NTuples(R)$
 - ◆ Si ni A ni B ne sont clés :

$$NTuples(R \bowtie_F S) \sim \text{Max}(SC_A(R) * NTuples(S), SC_B(S) * NTuples(R))$$

Boucles imbriquées

Zone mémoire du buffer



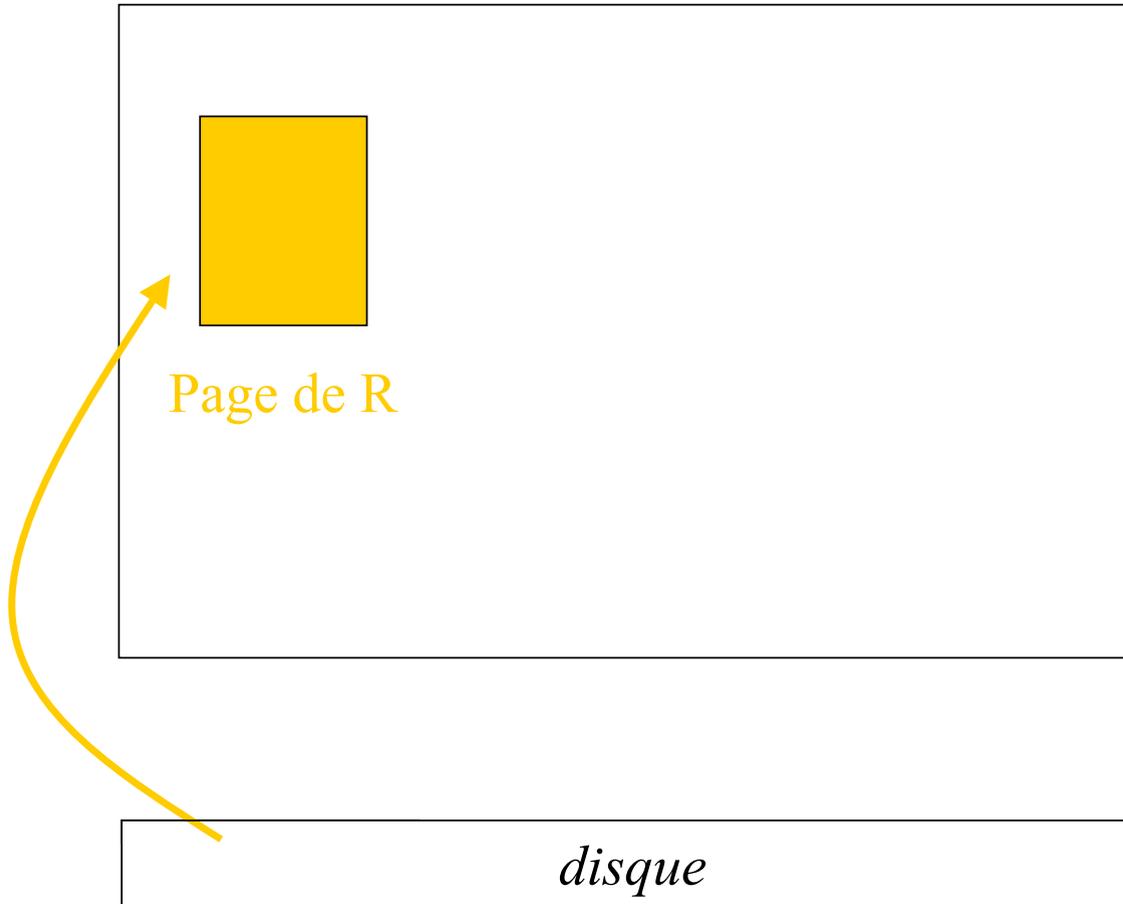
Les blocs de S sont montés en mémoire autant de fois qu'il y a de blocs pour R

S'il y a NBuffer places en mémoire, on monte les pages de R en mémoire par paquets de (NBuffer - 2) pages

S'il existe un index sur l'attribut de jointure d'une des deux relations, on l'utilise pour trouver les nuplets participant à la jointure

Boucles imbriquées

Zone mémoire du buffer



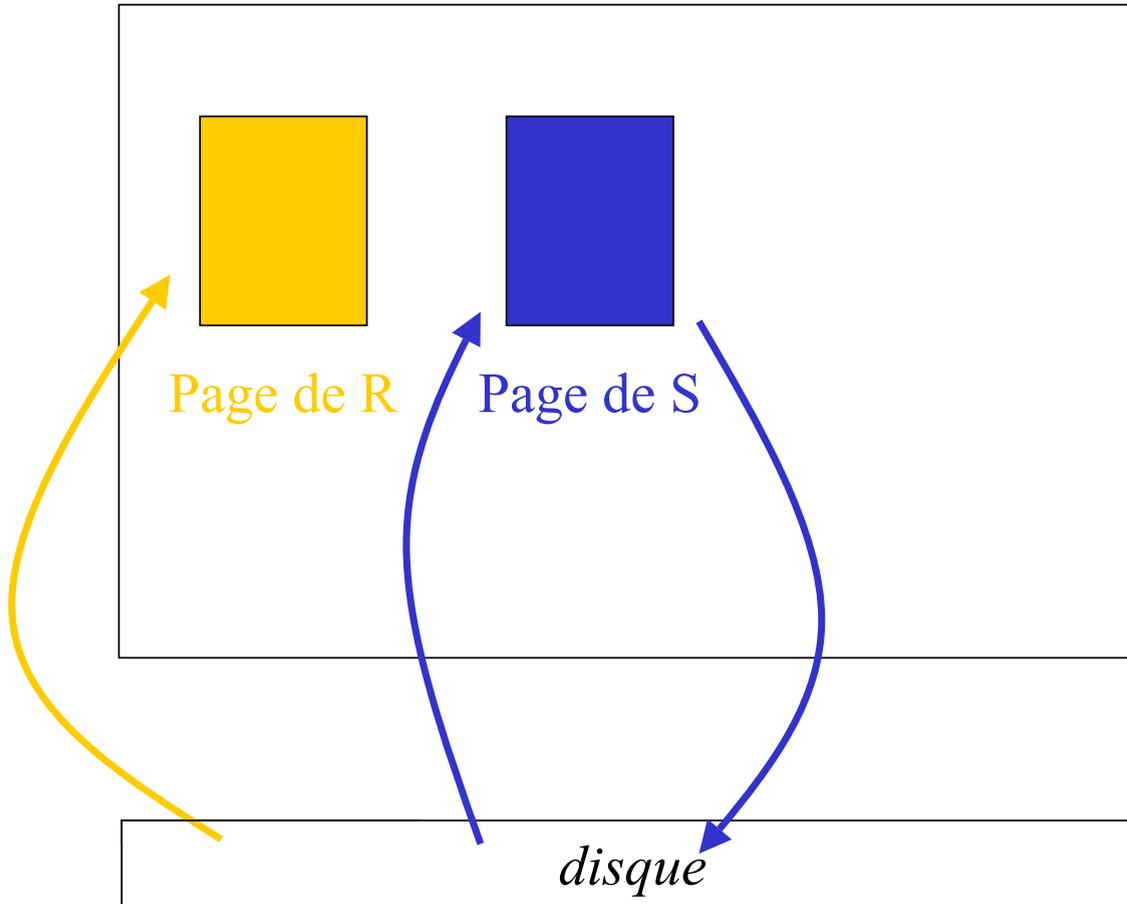
Les blocs de S sont montés en mémoire autant de fois qu'il y a de blocs pour R

S'il y a N_{Buffer} places en mémoire, on monte les pages de R en mémoire par paquets de $(N_{Buffer} - 2)$ pages

S'il existe un index sur l'attribut de jointure d'une des deux relations, on l'utilise pour trouver les nuplets participant à la jointure

Boucles imbriquées

Zone mémoire du buffer



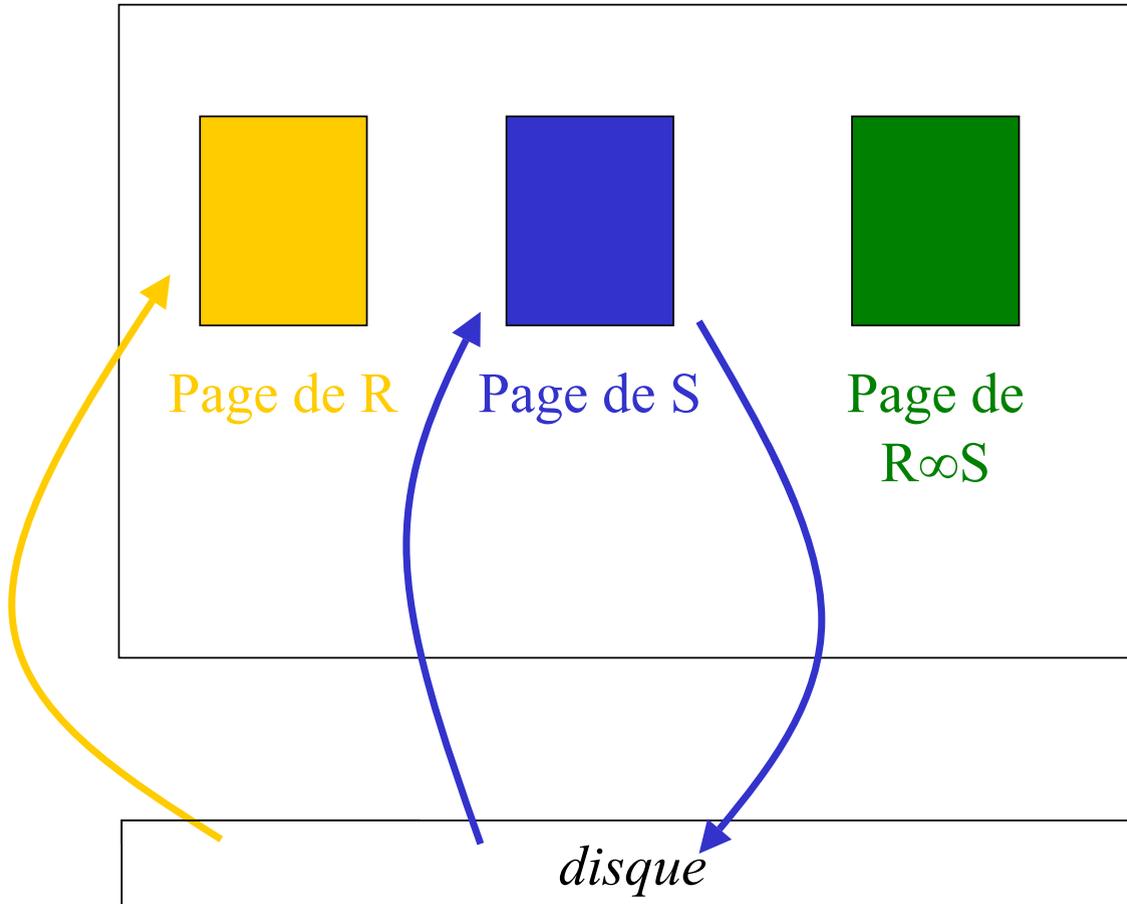
Les blocs de S sont montés en mémoire autant de fois qu'il y a de blocs pour R

S'il y a NBuffer places en mémoire, on monte les pages de R en mémoire par paquets de (NBuffer - 2) pages

S'il existe un index sur l'attribut de jointure d'une des deux relations, on l'utilise pour trouver les nuplets participant à la jointure

Boucles imbriquées

Zone mémoire du buffer



Les blocs de S sont montés en mémoire autant de fois qu'il y a de blocs pour R

S'il y a NBuffer places en mémoire, on monte les pages de R en mémoire par paquets de (NBuffer - 2) pages

S'il existe un index sur l'attribut de jointure d'une des deux relations, on l'utilise pour trouver les nuplets participant à la jointure

Tri-Fusion

① Tri des relations sur l'attribut de jointure

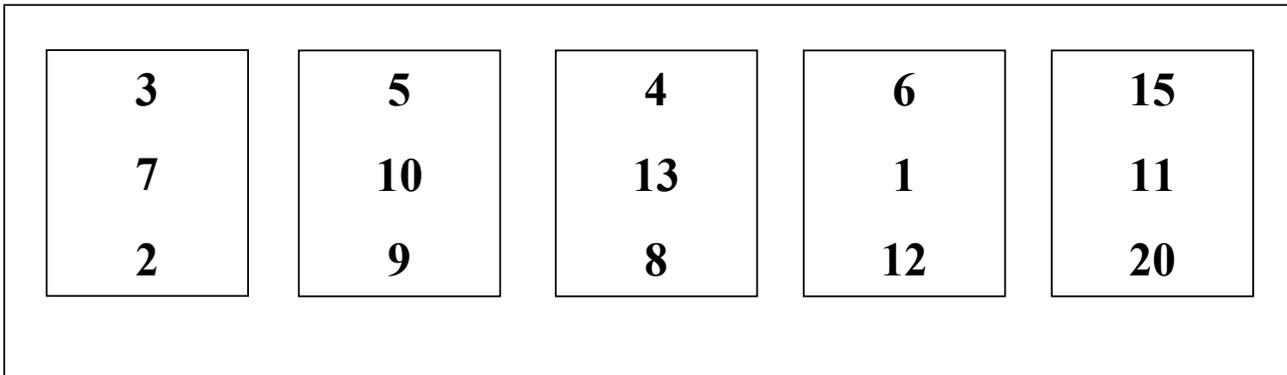
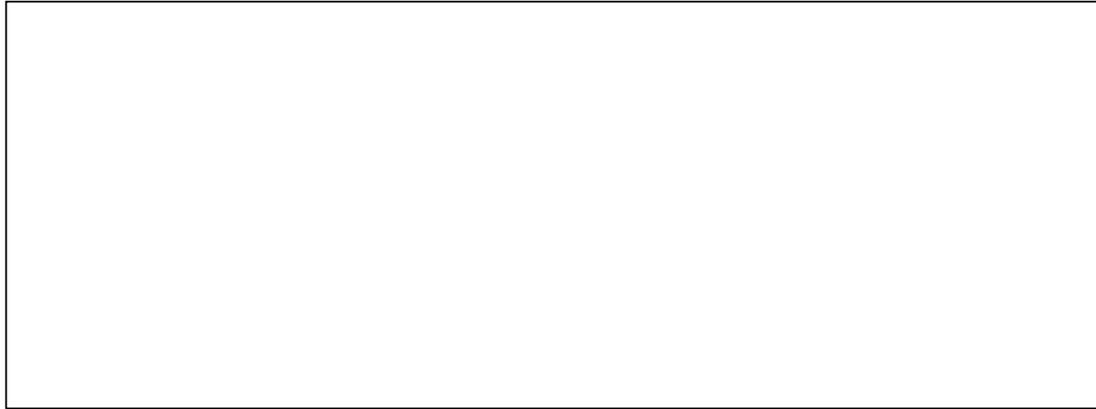
Ex. Tri externe ou tri par séries monotones

② Equi-jointure des deux relations triées

En parcourant simultanément les deux relations pages par pages

Tri externe : 1ère étape

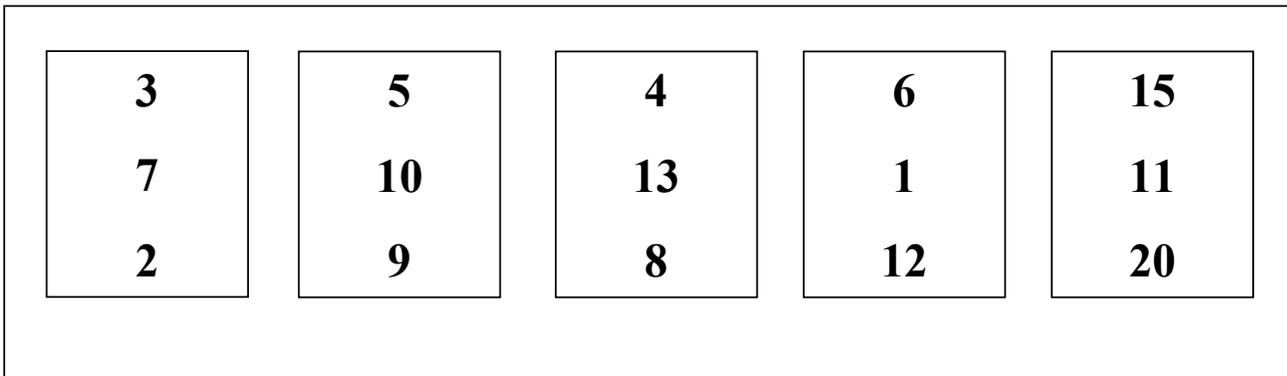
*La mémoire
contient 3
emplacements*



disque

Tri externe : 1ère étape

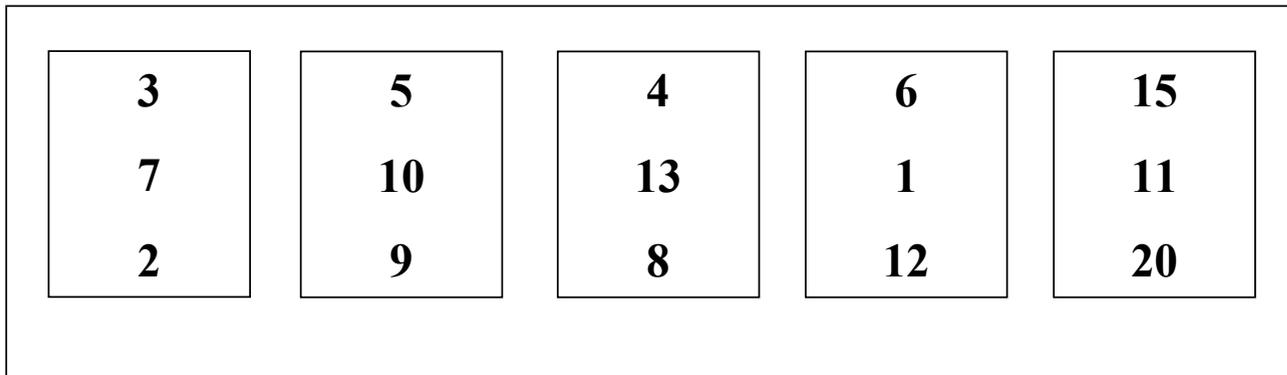
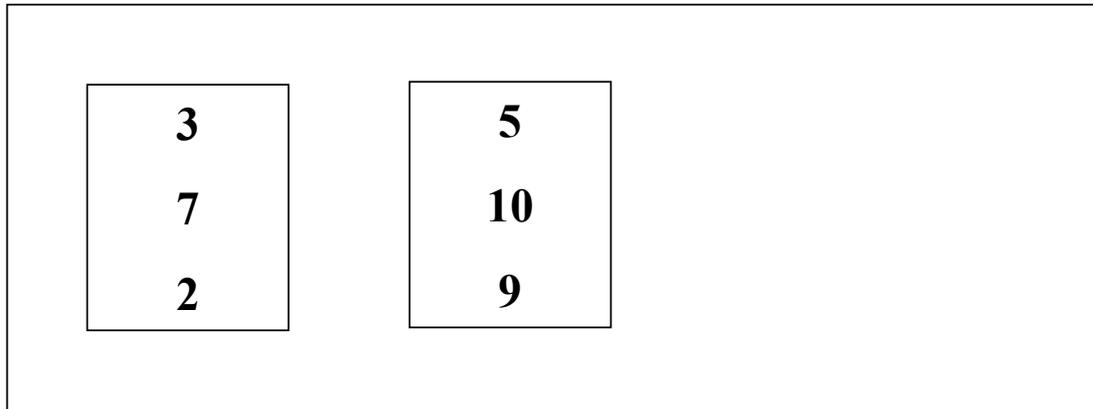
*La mémoire
contient 3
emplacements*



disque

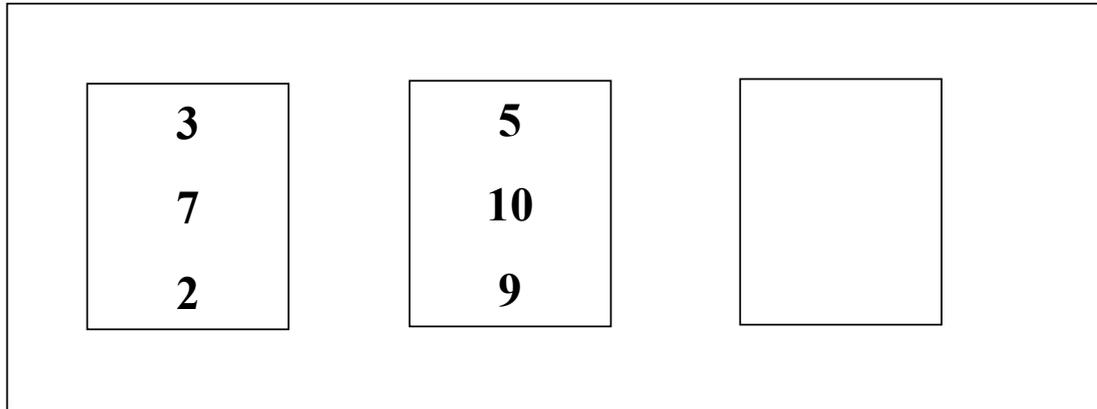
Tri externe : 1ère étape

*La mémoire
contient 3
emplacements*

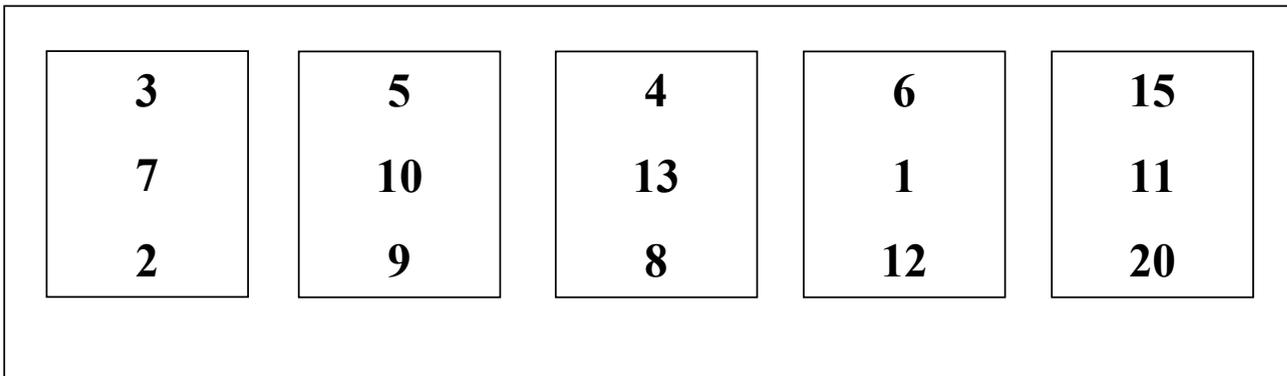


disque

Tri externe : 1ère étape



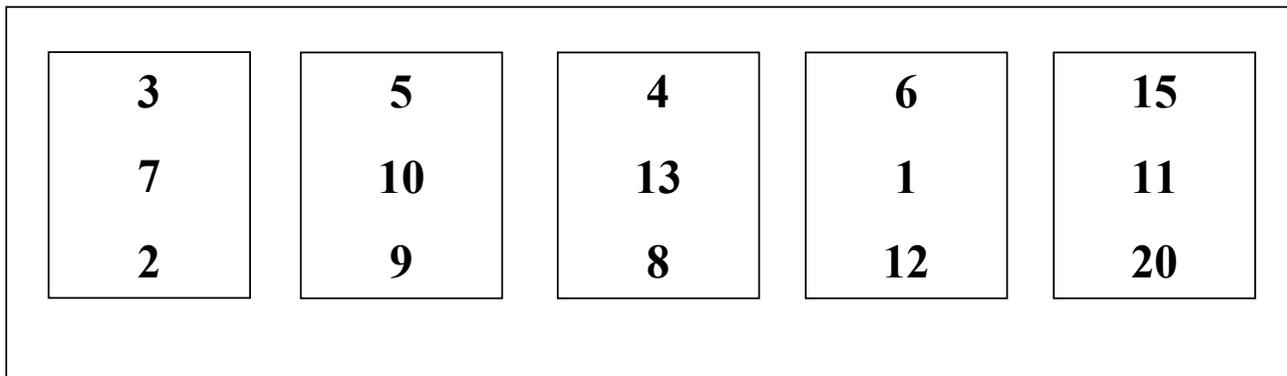
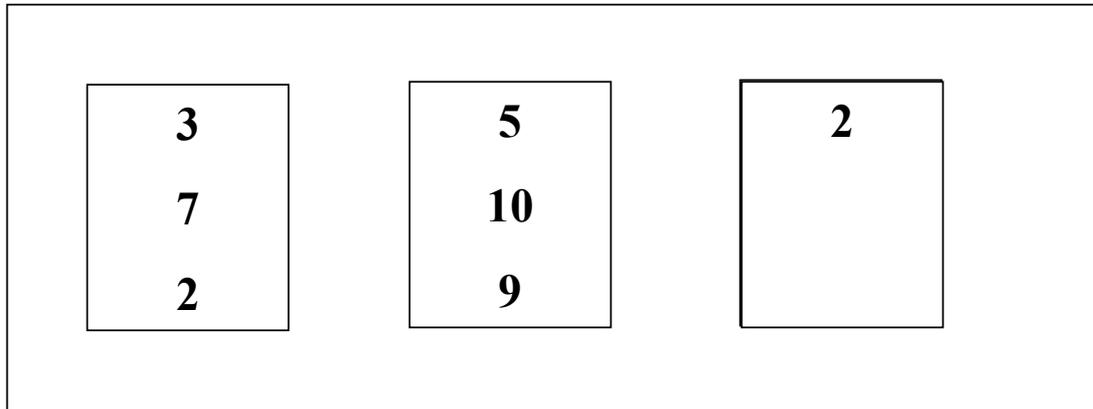
*La mémoire
contient 3
emplacements*



disque

Tri externe : 1ère étape

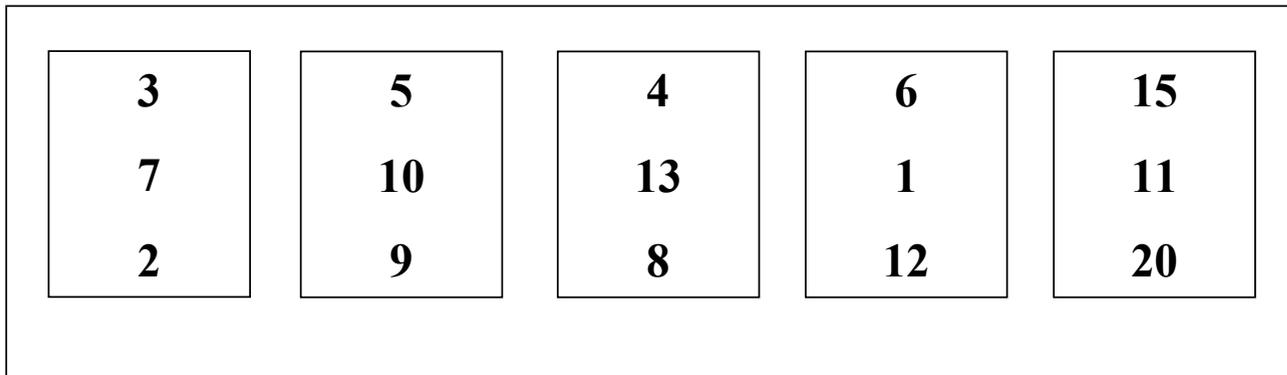
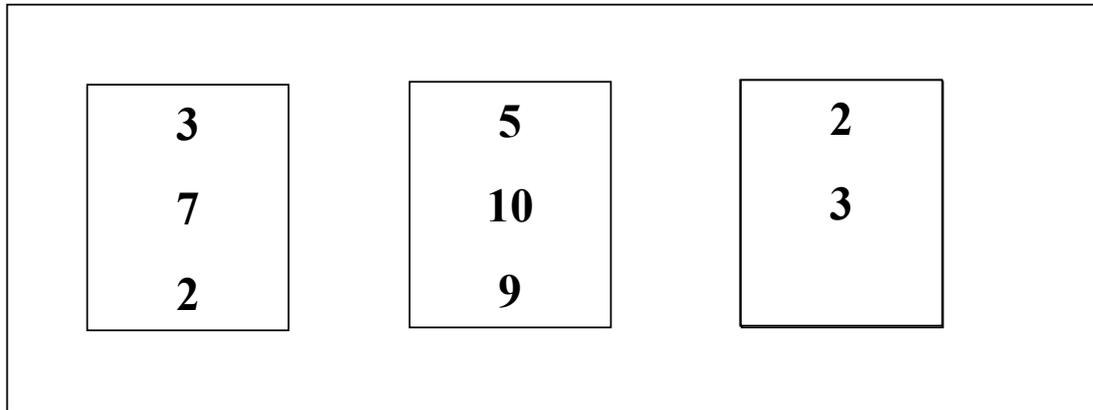
*La mémoire
contient 3
emplacements*



disque

Tri externe : 1ère étape

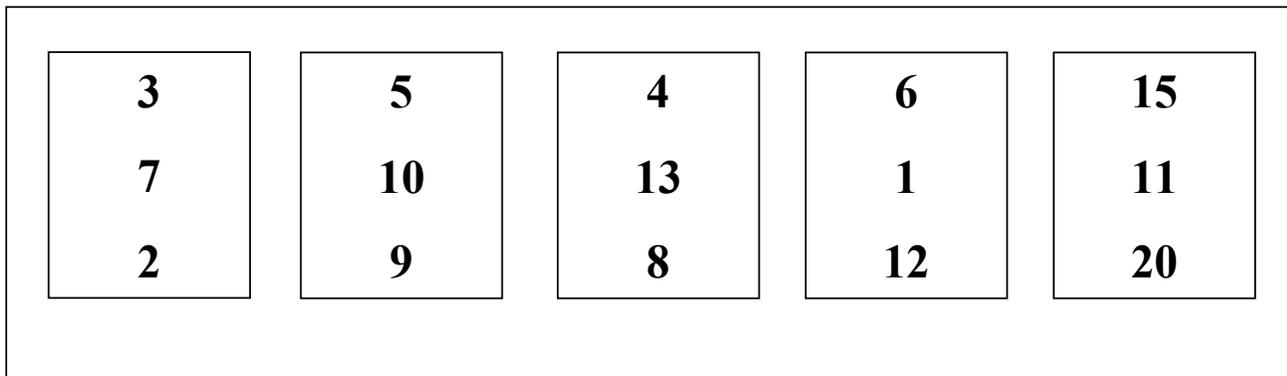
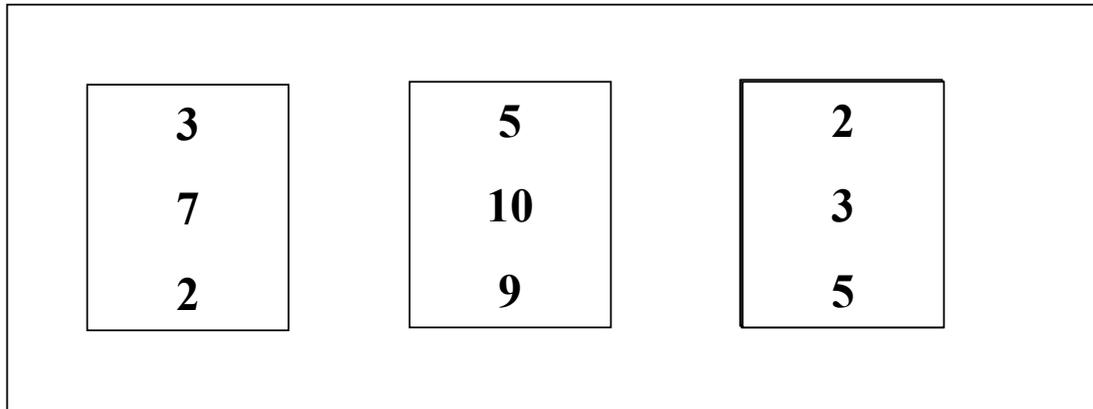
*La mémoire
contient 3
emplacements*



disque

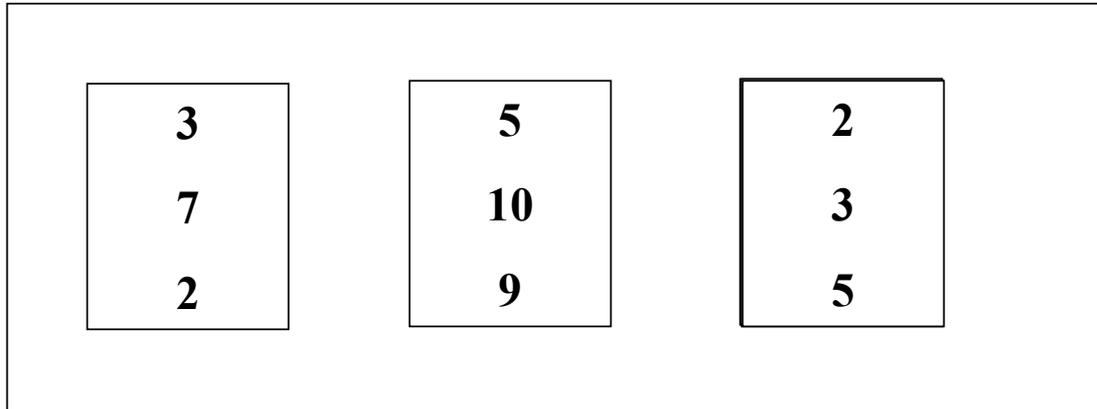
Tri externe : 1ère étape

*La mémoire
contient 3
emplacements*

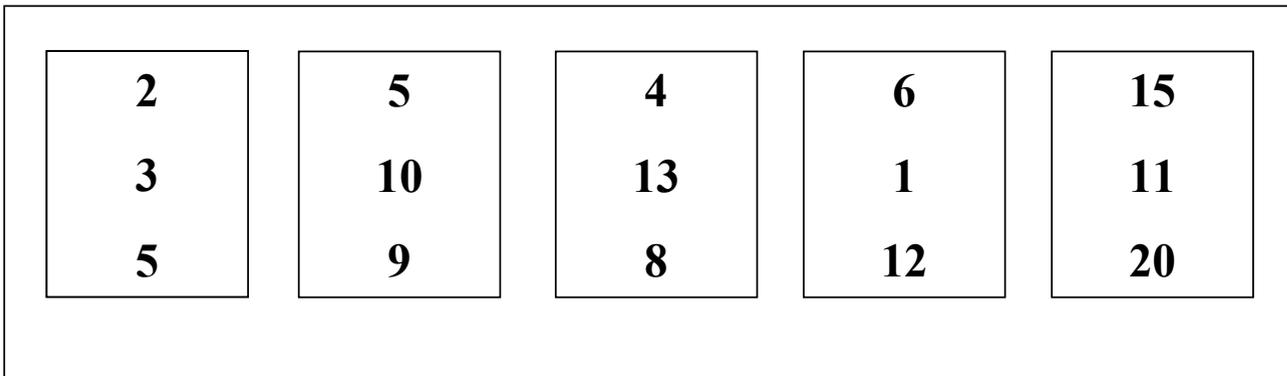


disque

Tri externe : 1ère étape

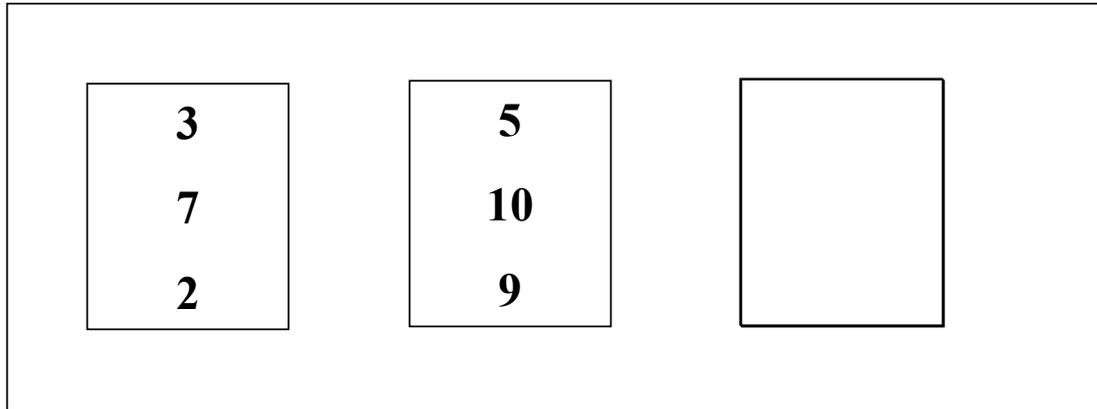


*La mémoire
contient 3
emplacements*

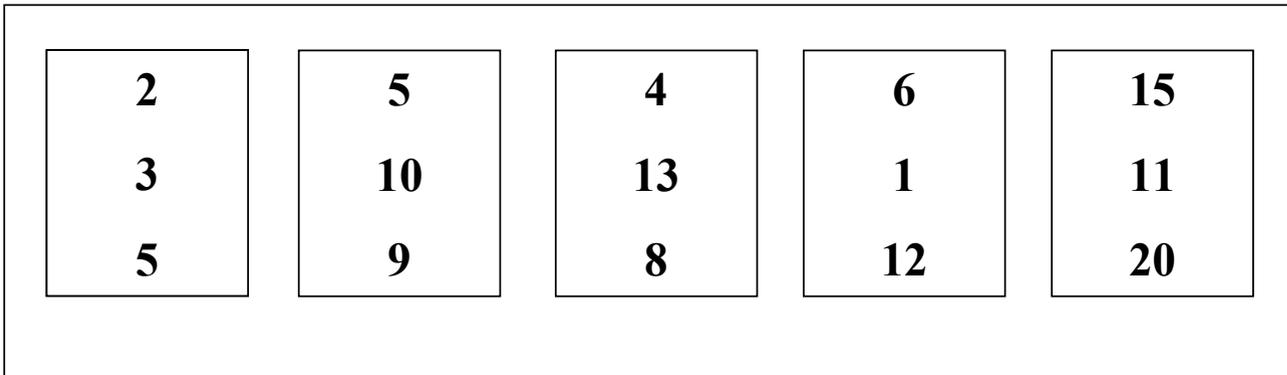


disque

Tri externe : 1ère étape

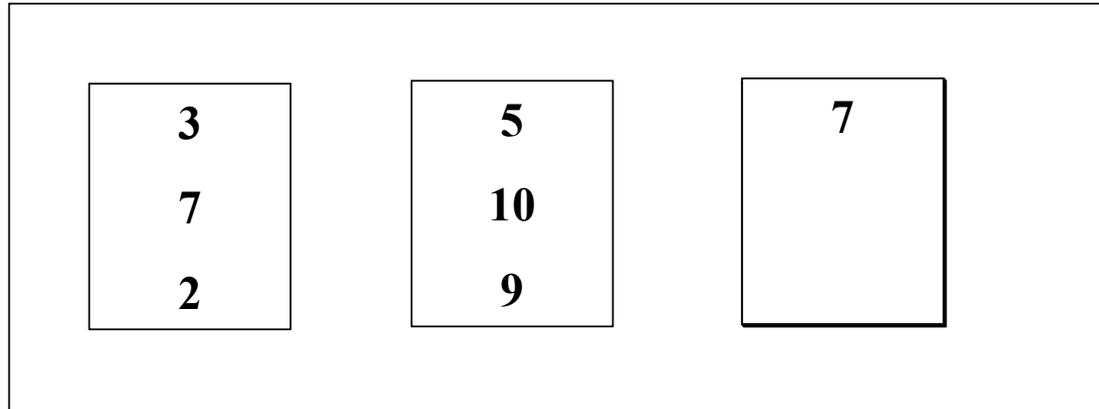


*La mémoire
contient 3
emplacements*

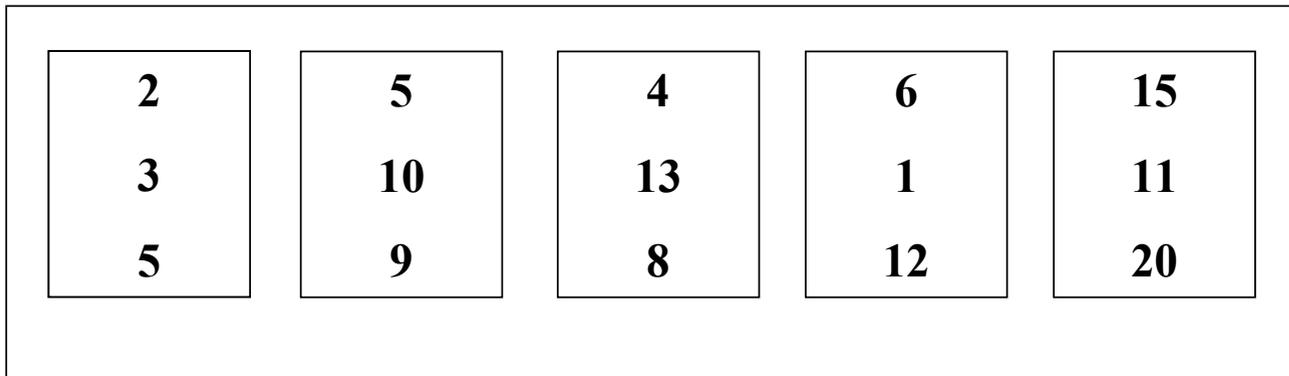


disque

Tri externe : 1ère étape



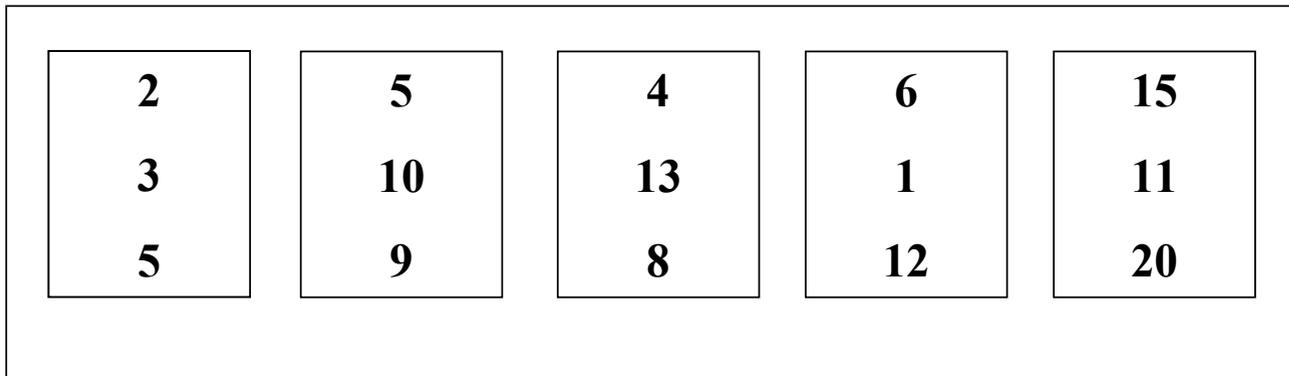
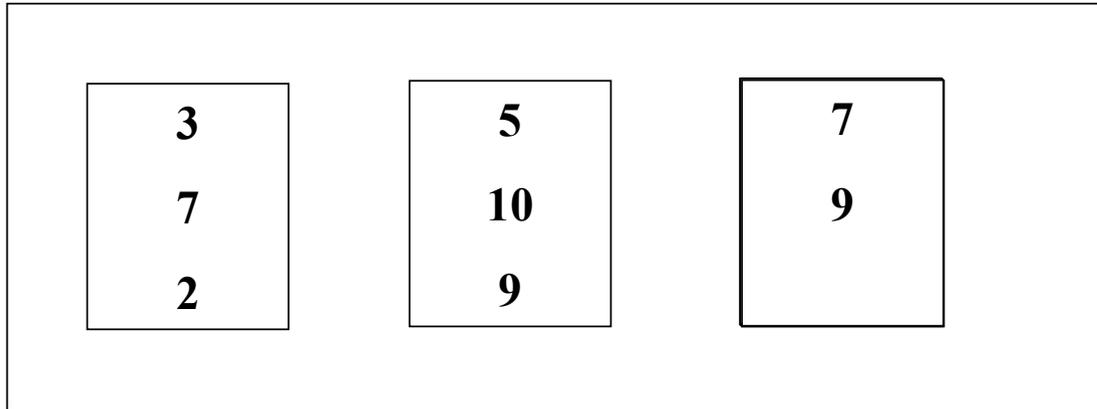
*La mémoire
contient 3
emplacements*



disque

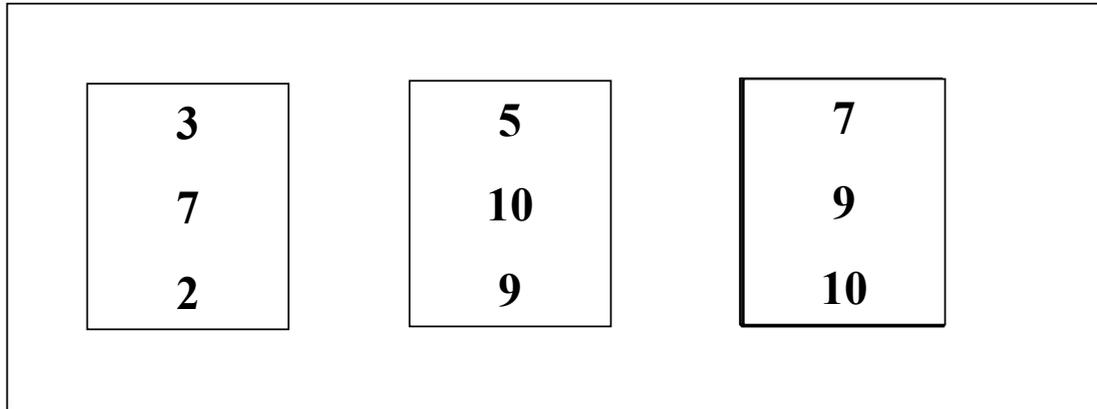
Tri externe : 1ère étape

*La mémoire
contient 3
emplacements*

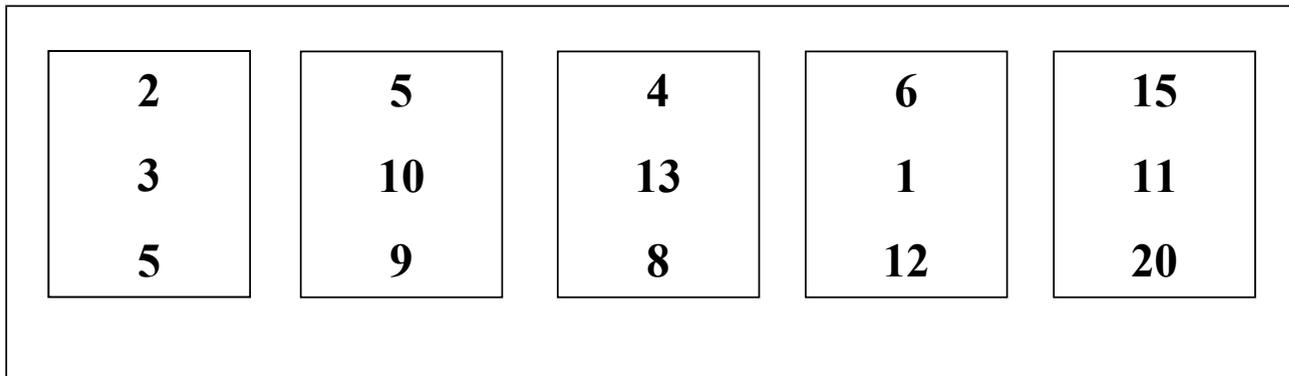


disque

Tri externe : 1ère étape

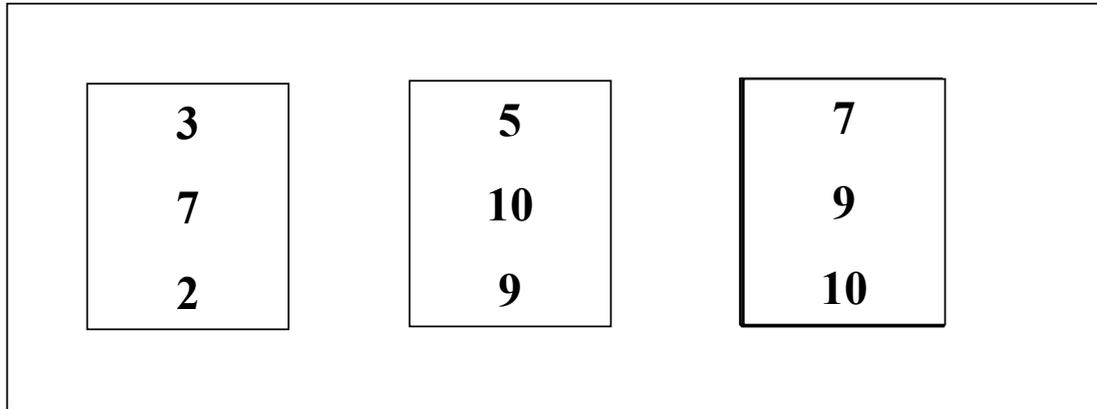


*La mémoire
contient 3
emplacements*

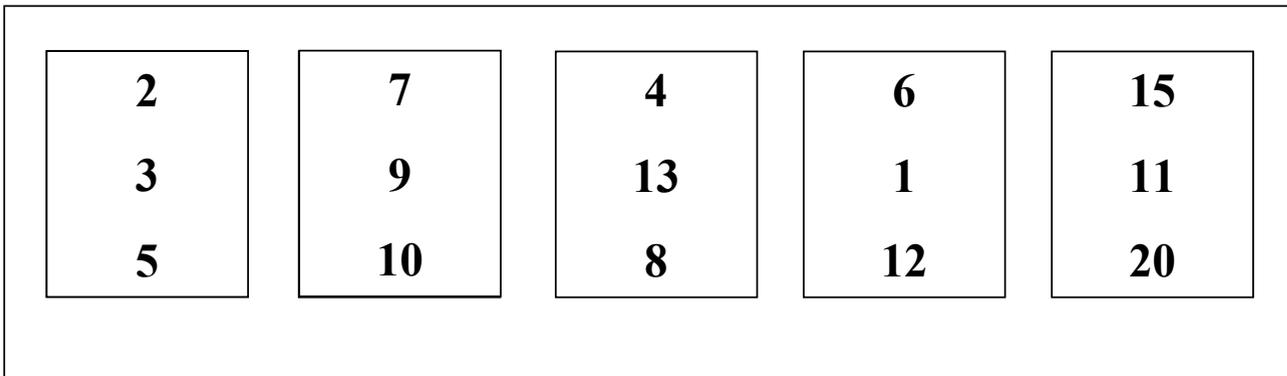


disque

Tri externe : 1ère étape

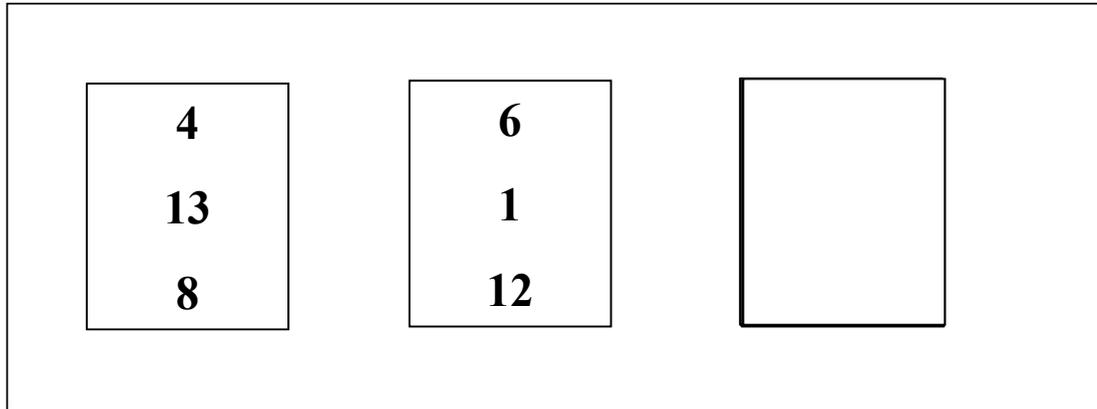


*La mémoire
contient 3
emplacements*

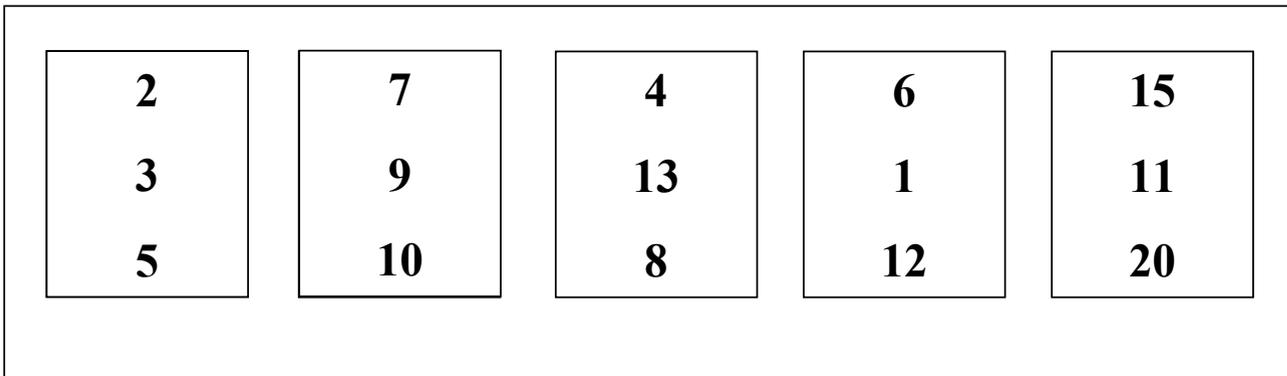


disque

Tri externe : 1ère étape

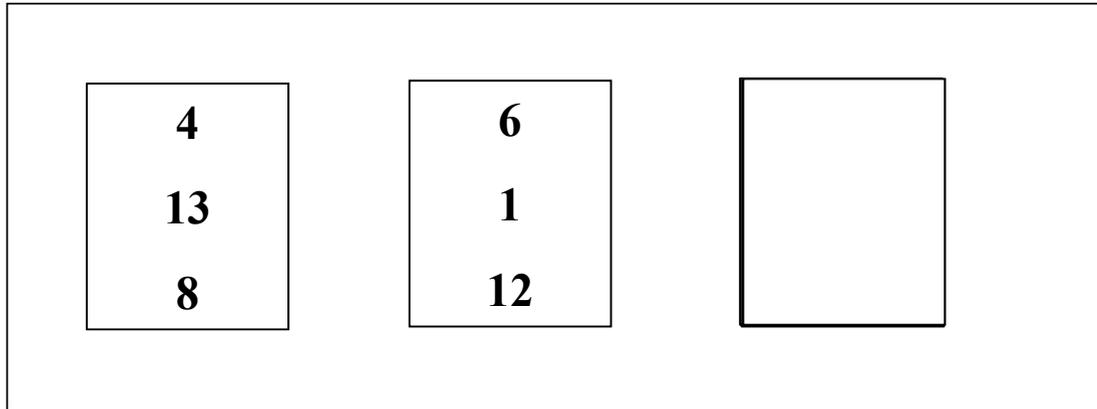


*La mémoire
contient 3
emplacements*

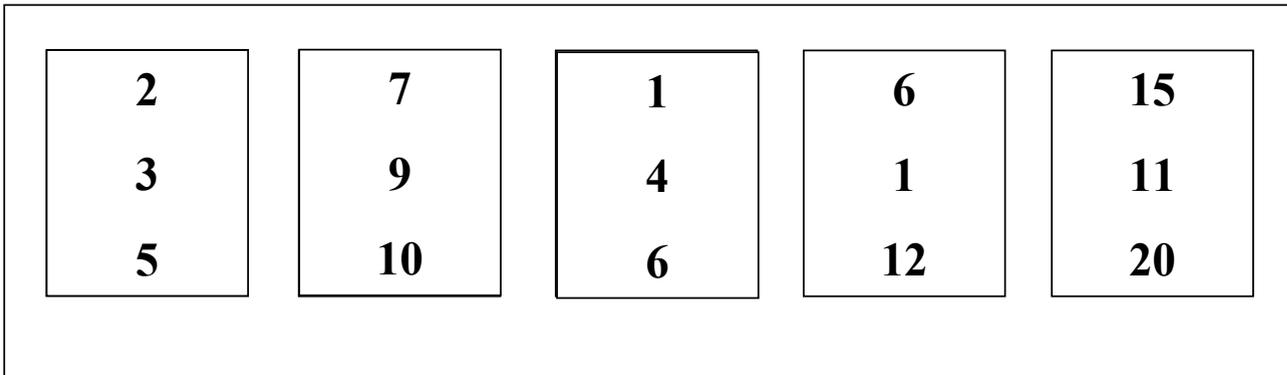


disque

Tri externe : 1ère étape

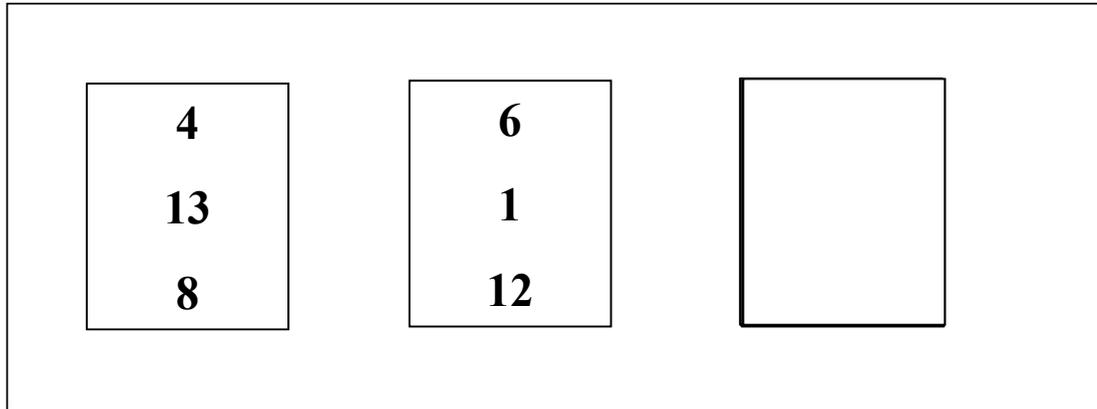


*La mémoire
contient 3
emplacements*

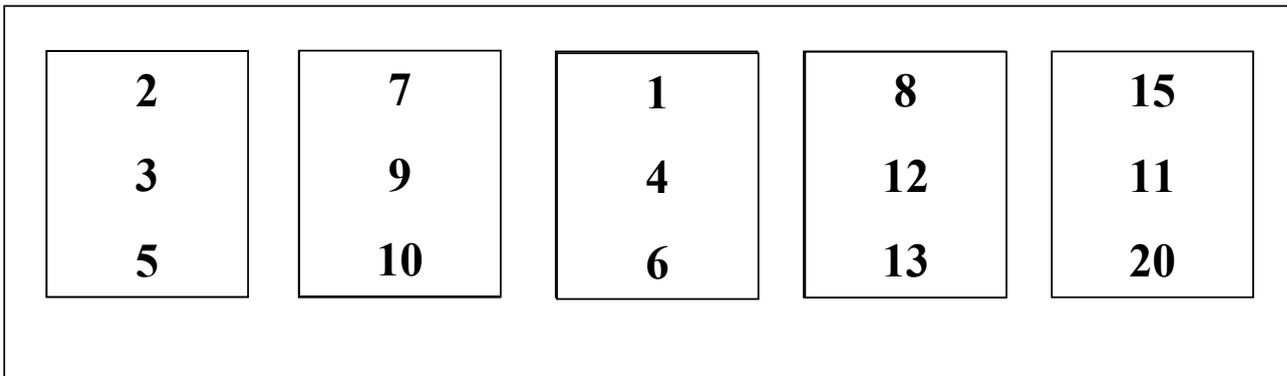


disque

Tri externe : 1ère étape



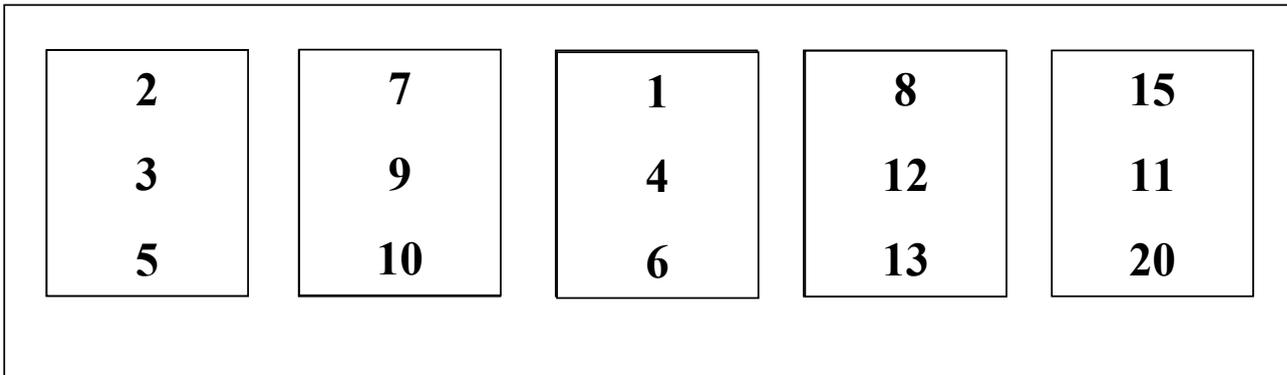
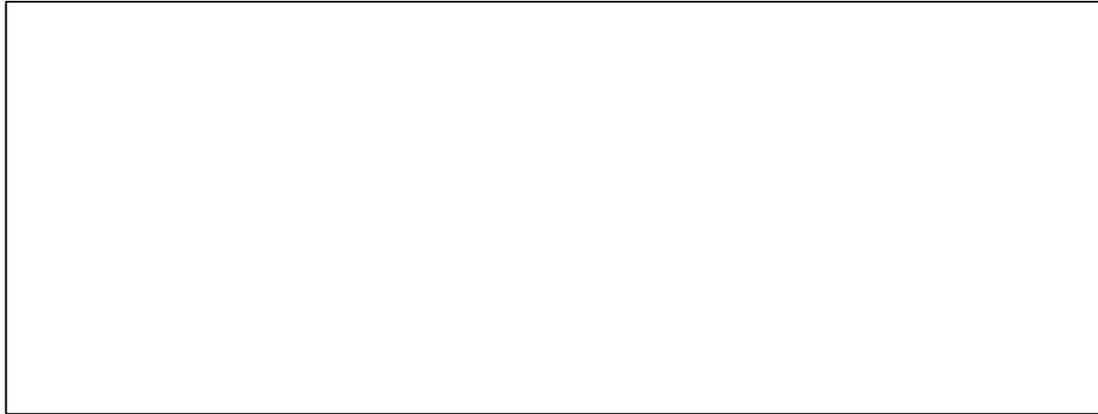
*La mémoire
contient 3
emplacements*



disque

Tri externe : 1ère étape

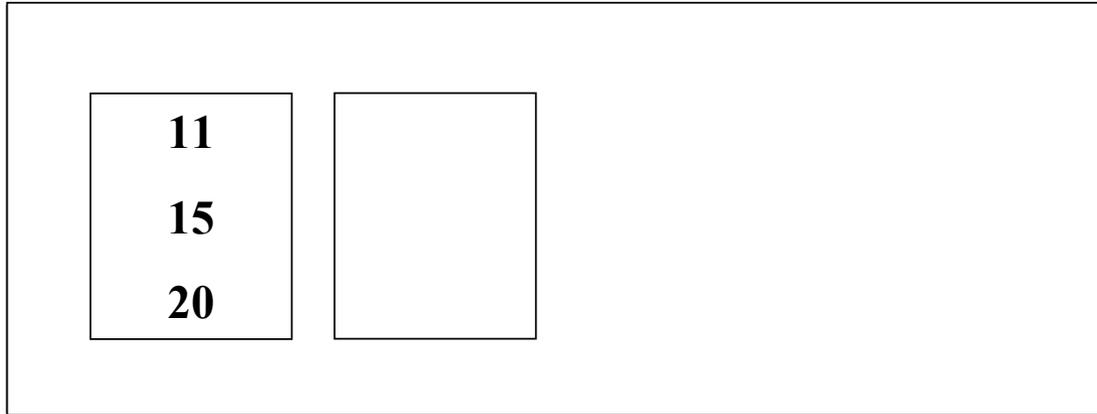
*La mémoire
contient 3
emplacements*



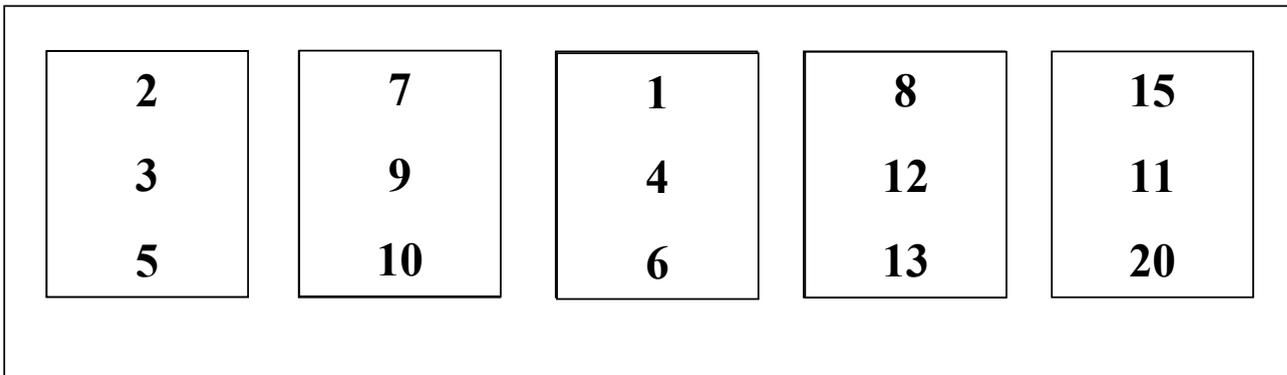
disque

Tri externe : 1ère étape

*La mémoire
contient 3
emplacements*

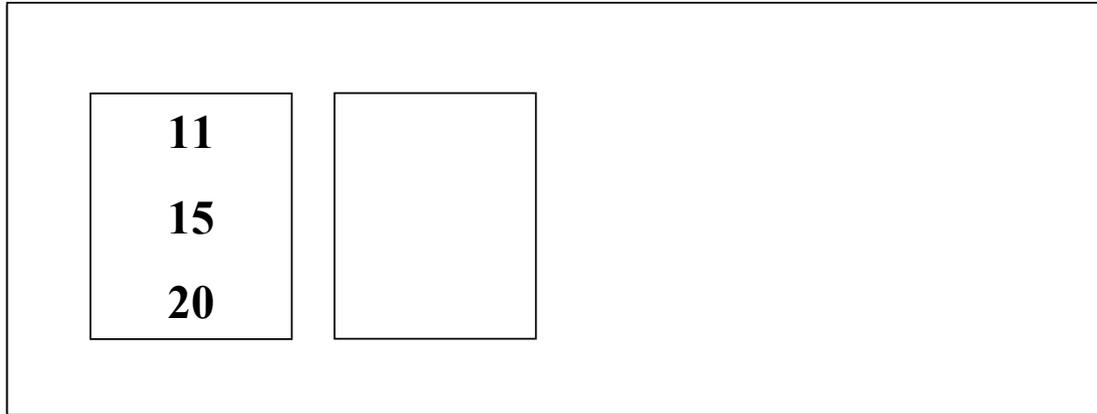


disque

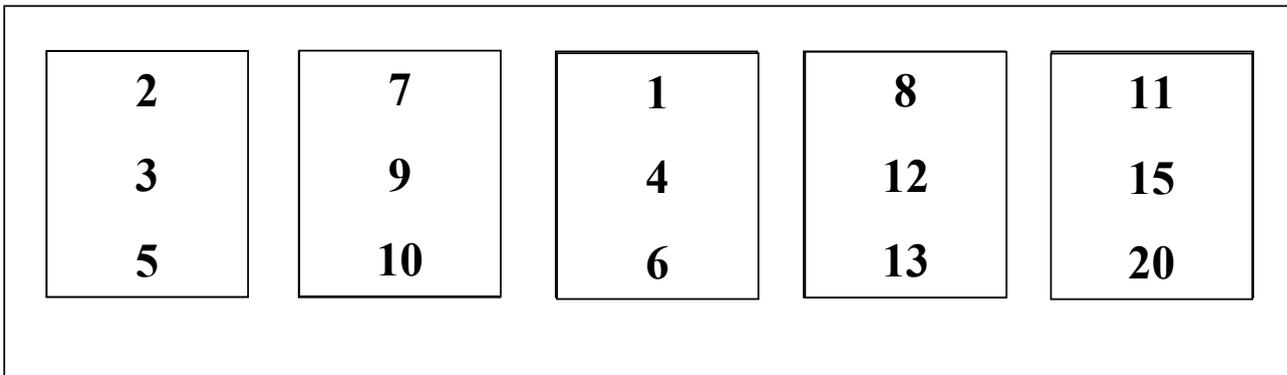


Tri externe : 1ère étape

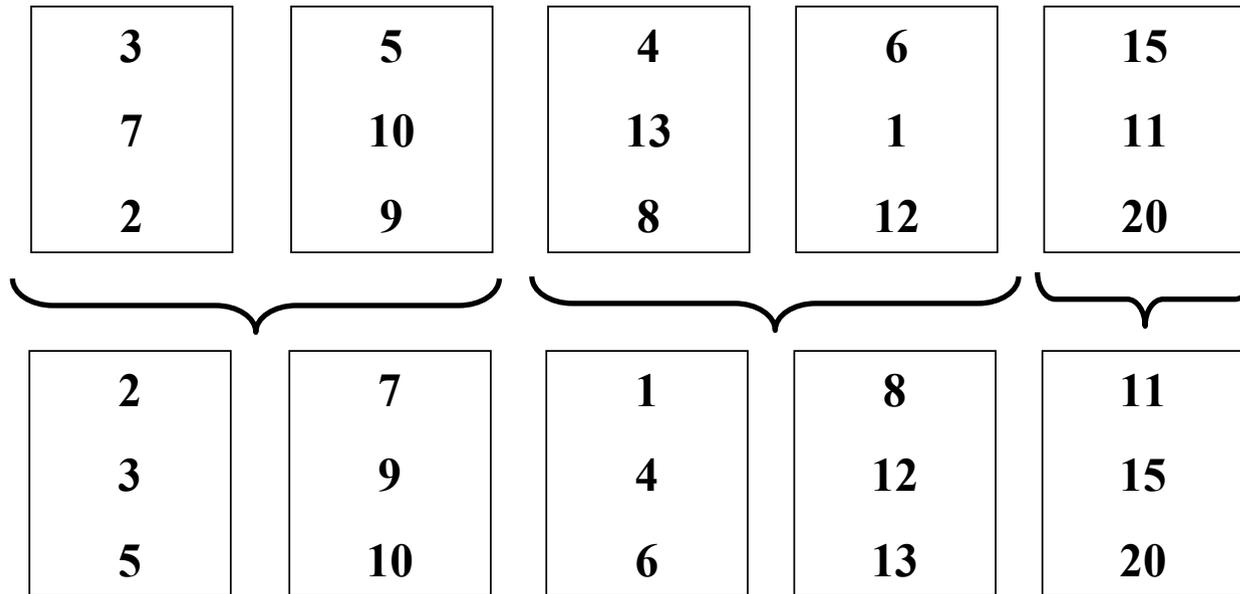
*La mémoire
contient 3
emplacements*



disque



Tri externe

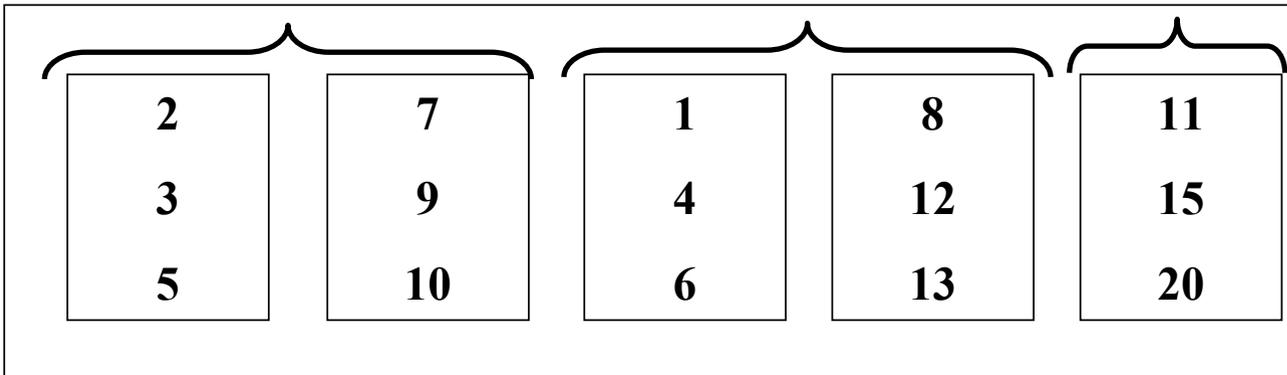
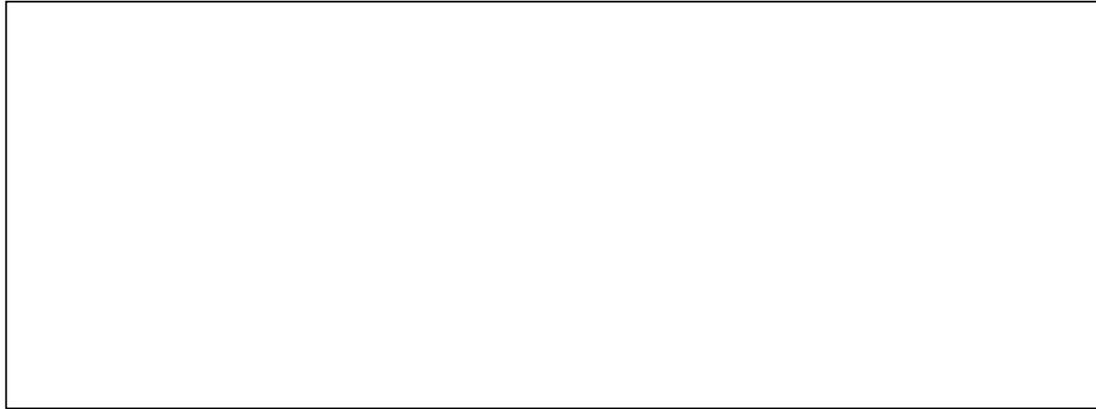


Ensemble de départ

*Après la 1ère étape de tri : Les pages sont triées par paquets de 2
 ⇒ 3 paquets triés de 2 pages soit 2*5 E/S*

Tri externe : 2ème étape

*La mémoire
contient 3
emplacements*

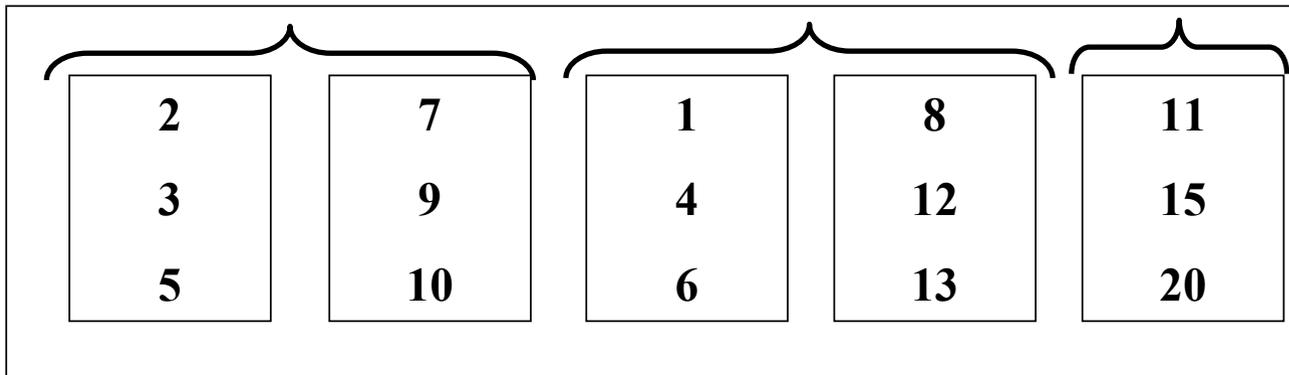


disque

Tri externe : 2ème étape

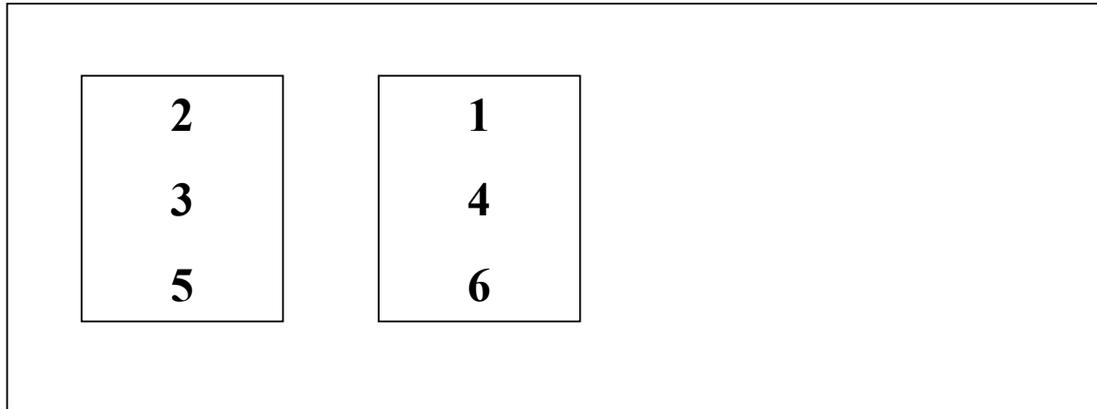


*La mémoire
contient 3
emplacements*

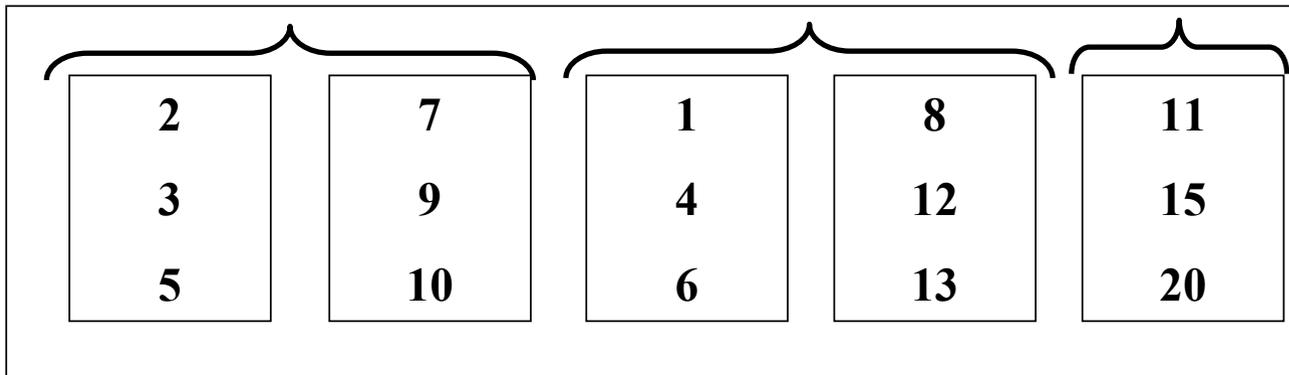


disque

Tri externe : 2ème étape



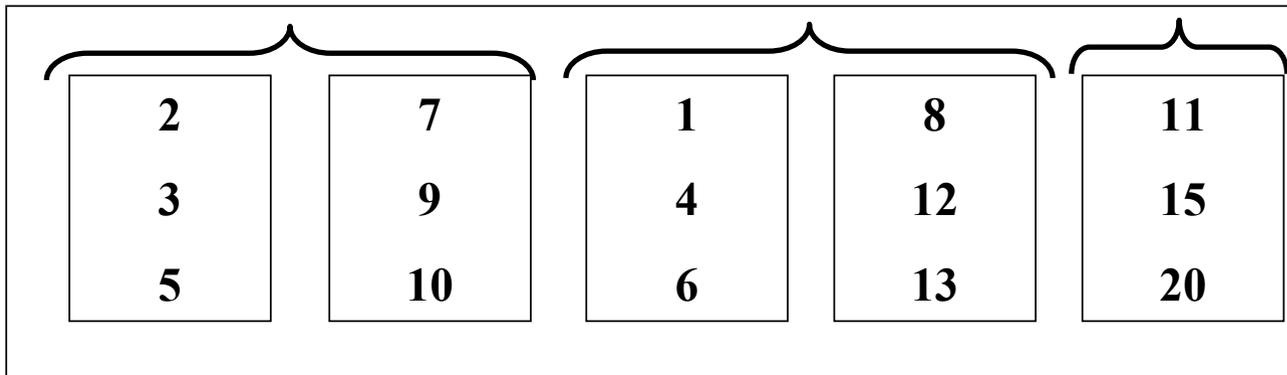
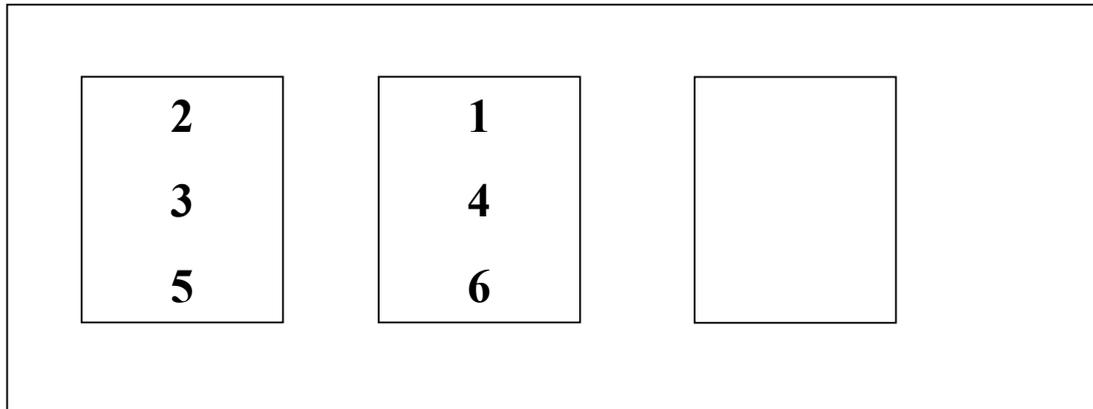
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

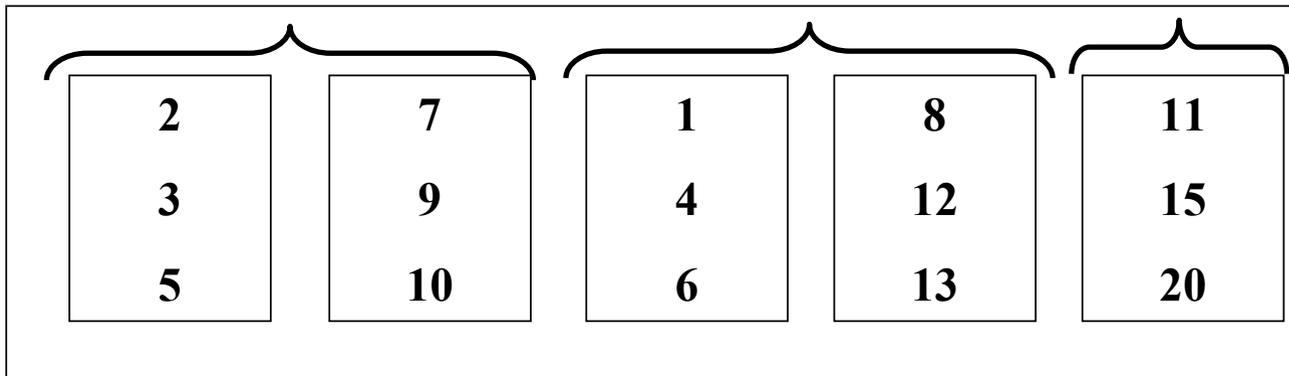
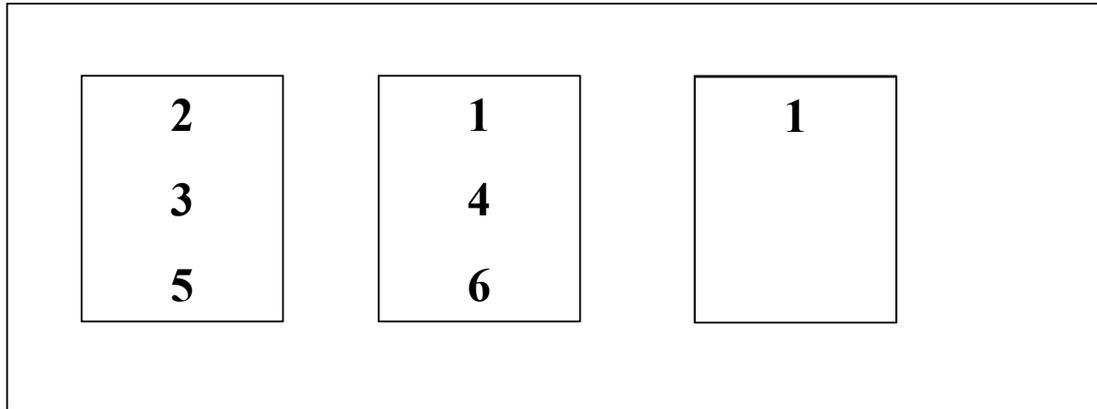
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

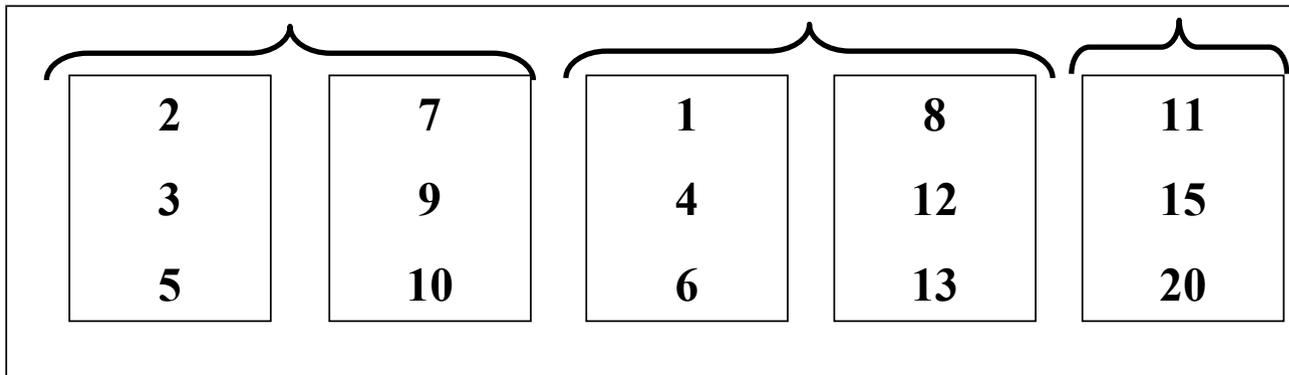
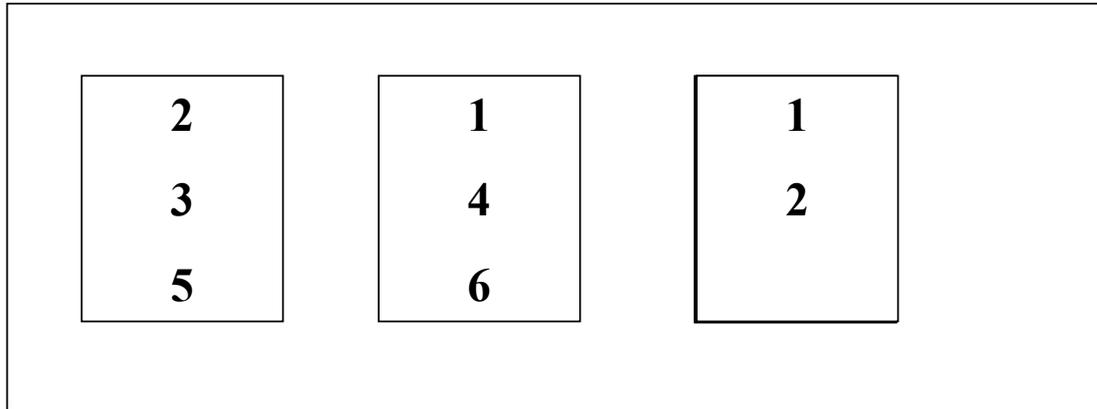
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

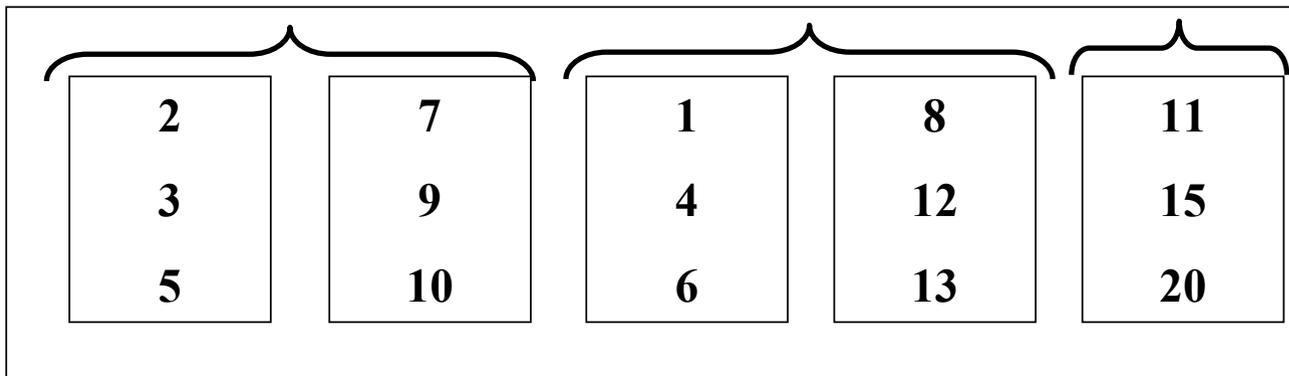
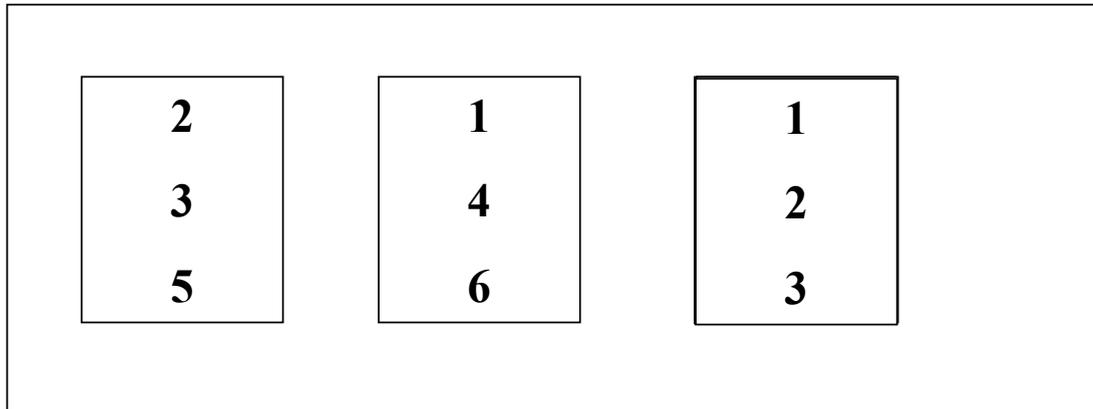
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

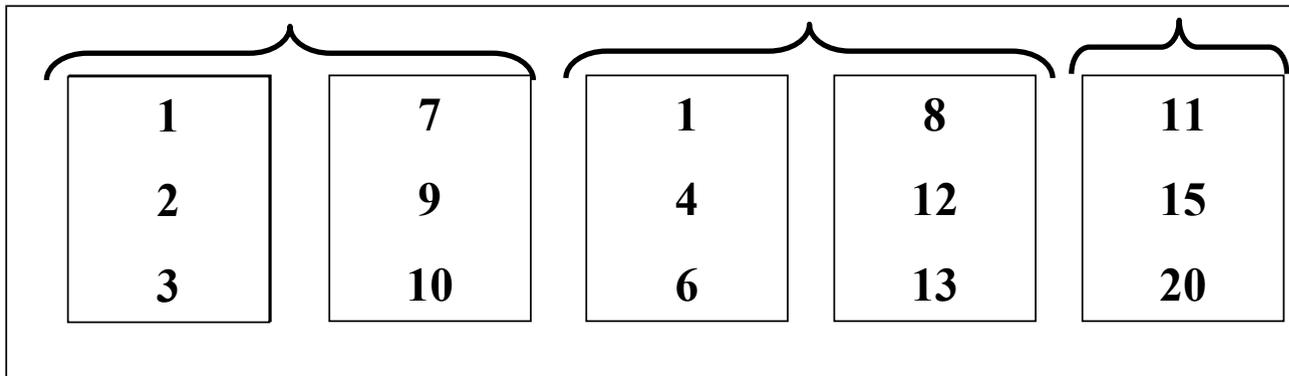
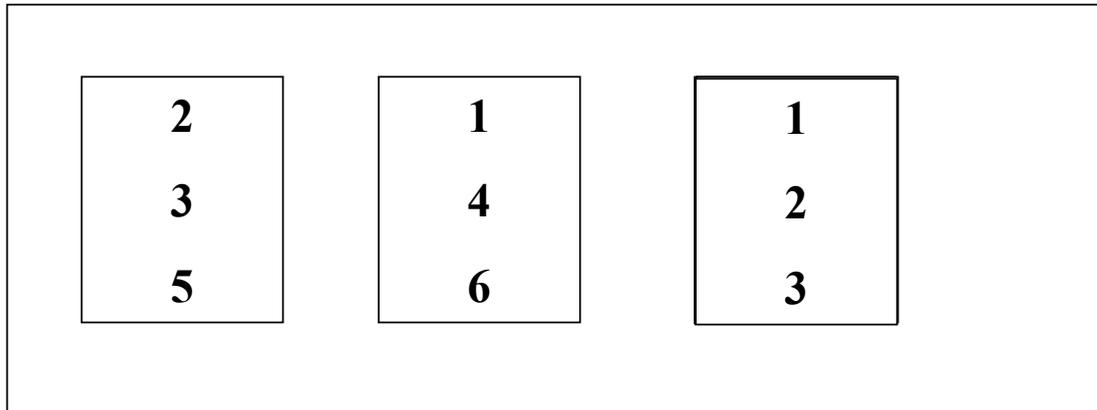
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

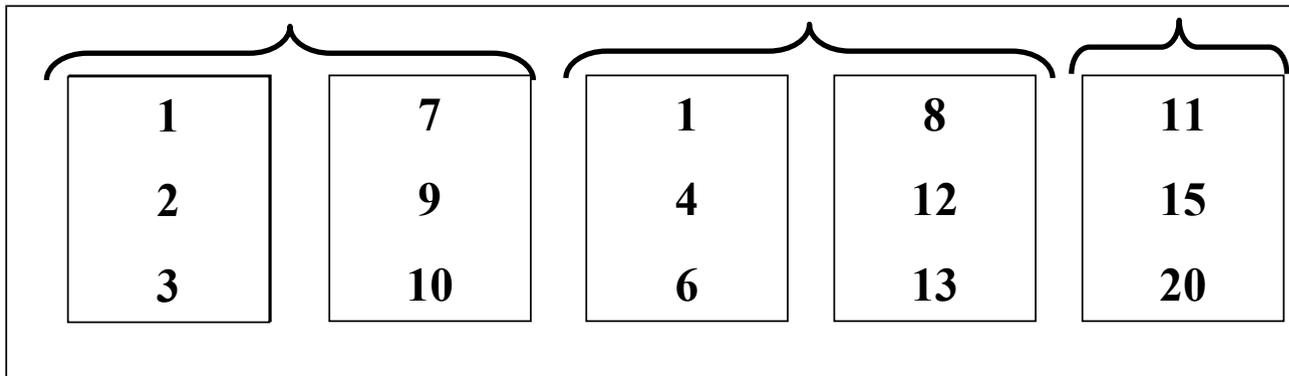
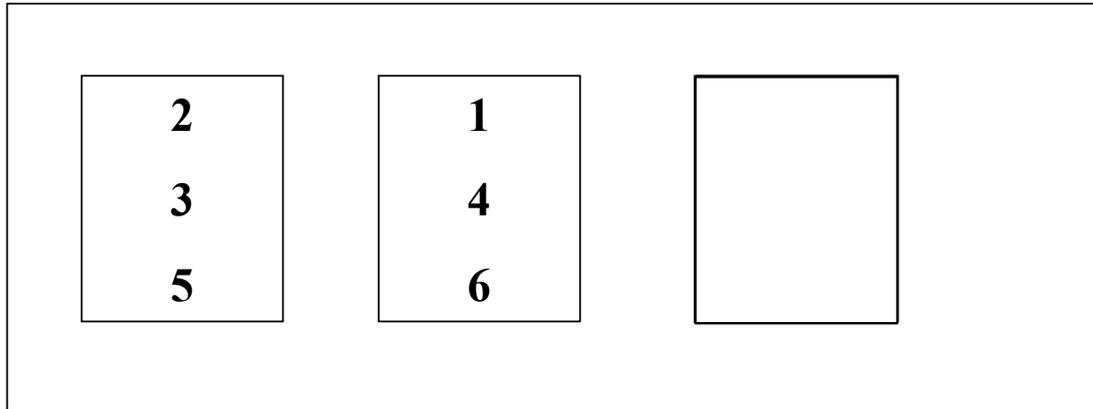
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

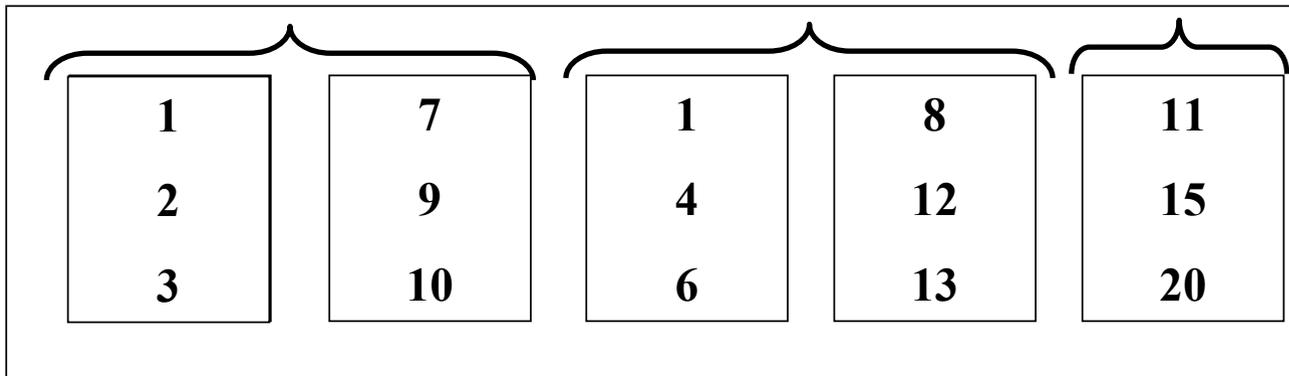
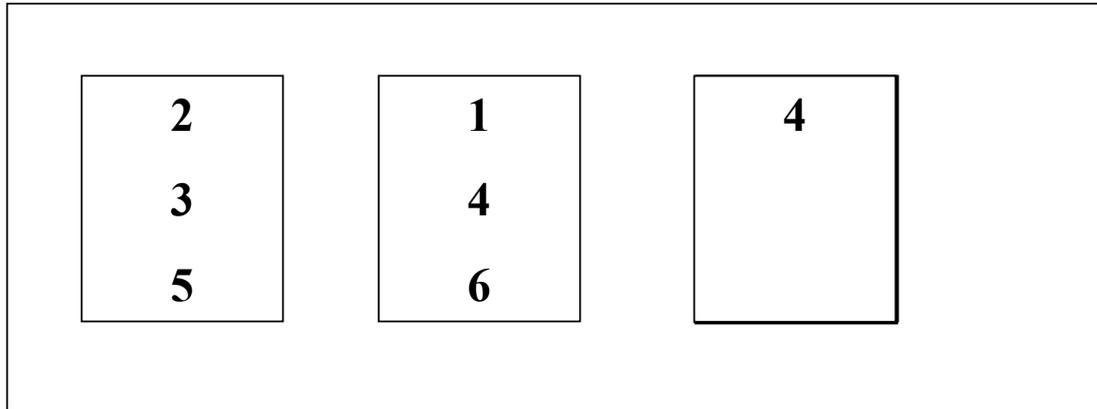
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

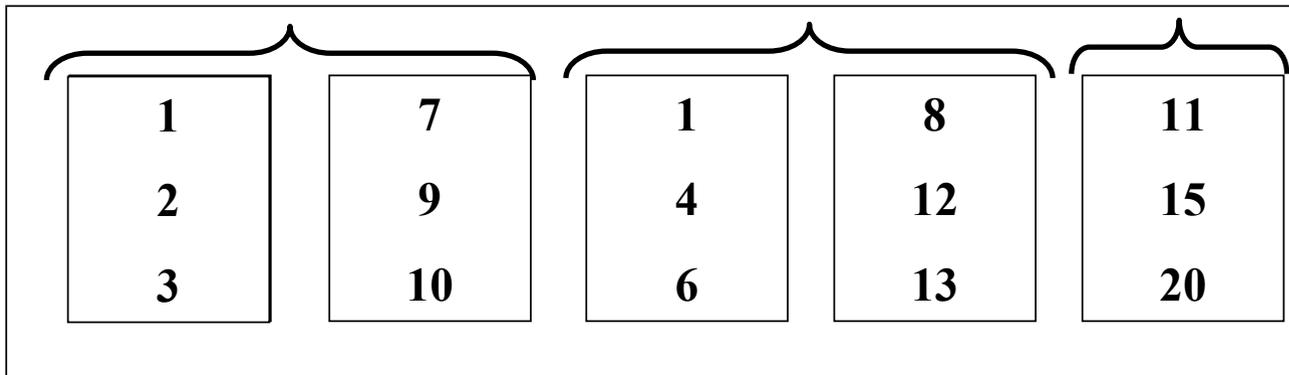
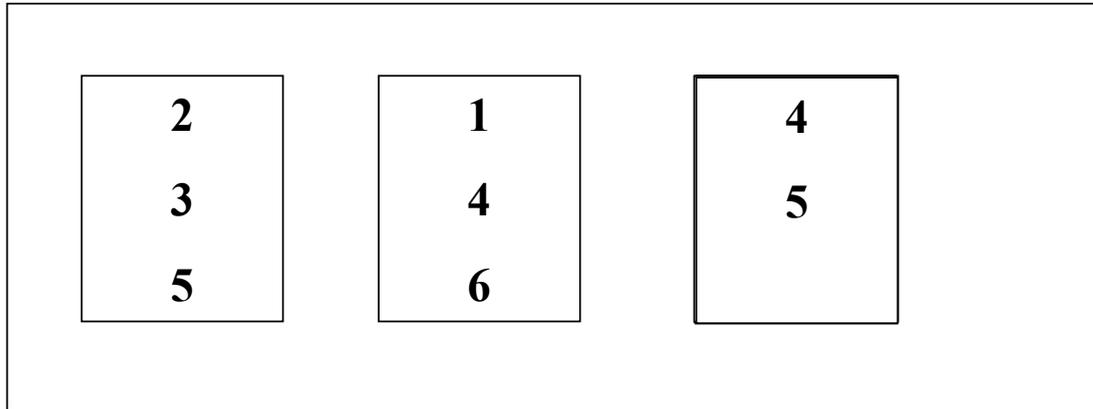
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

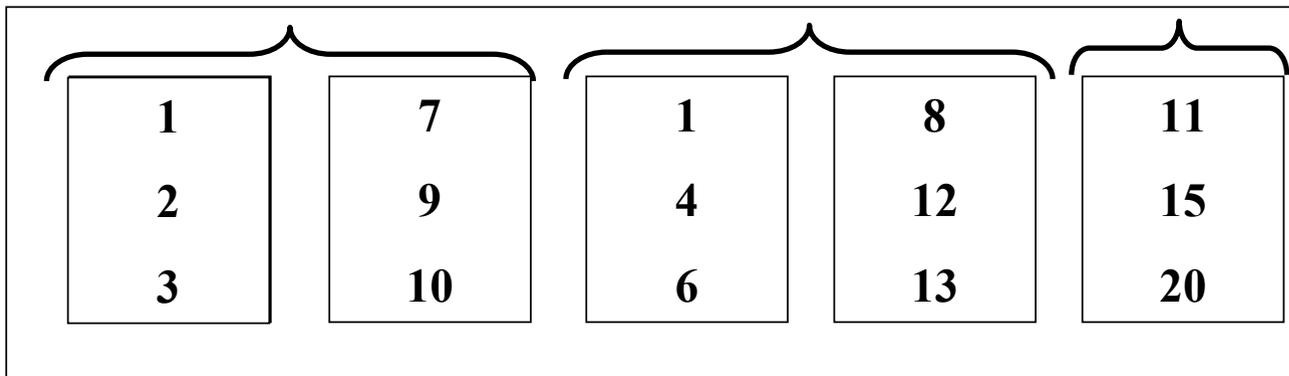
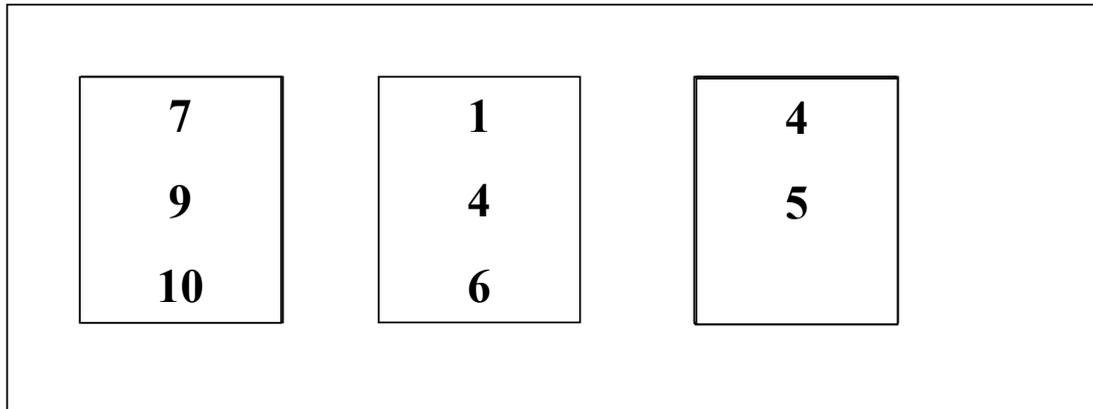
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

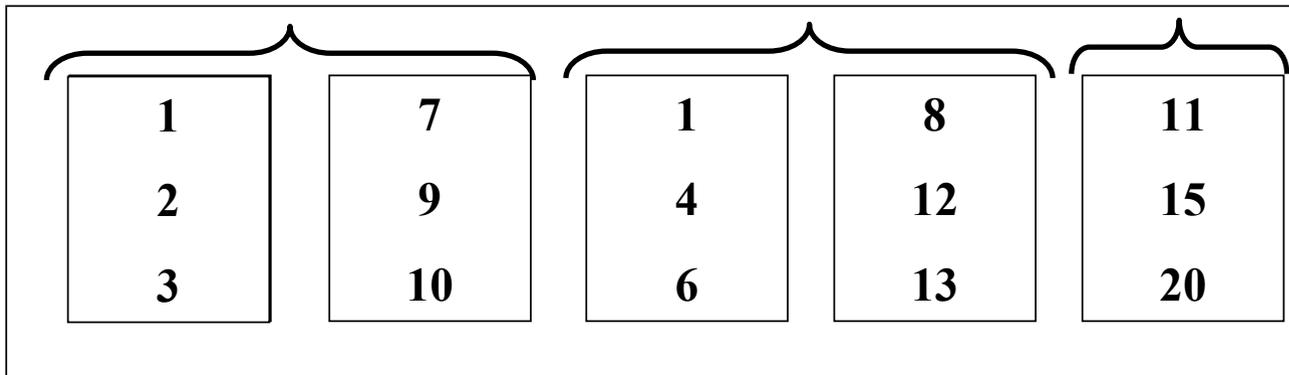
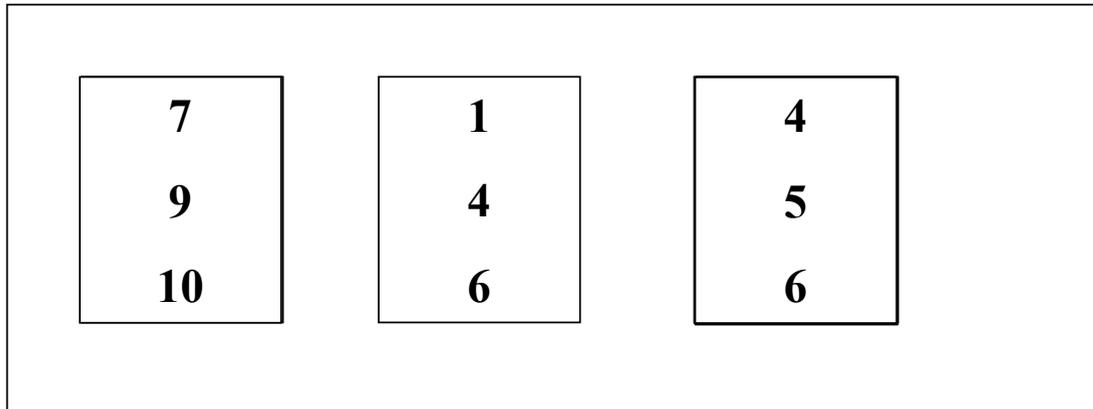
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

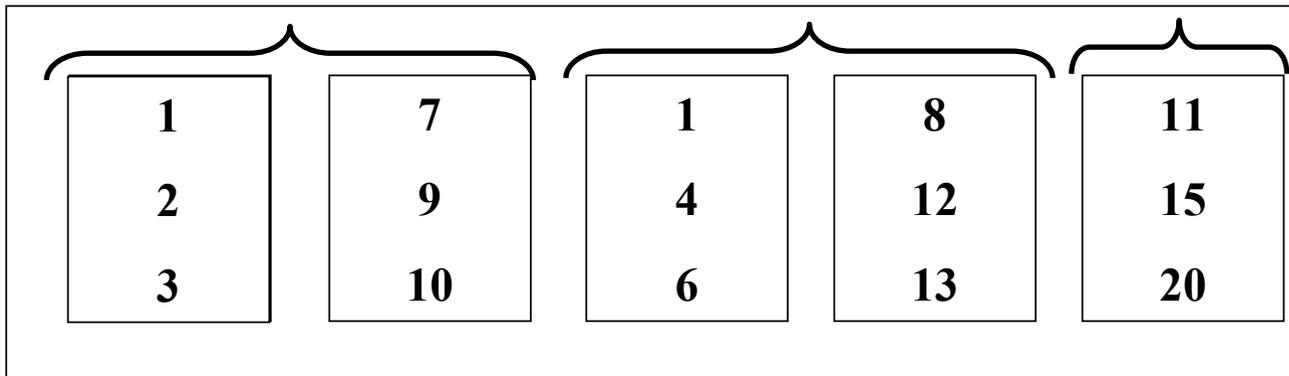
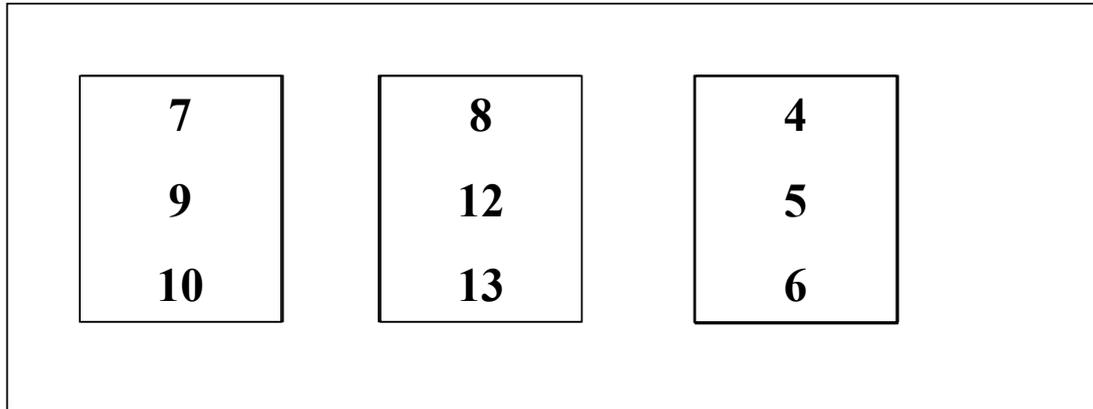
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

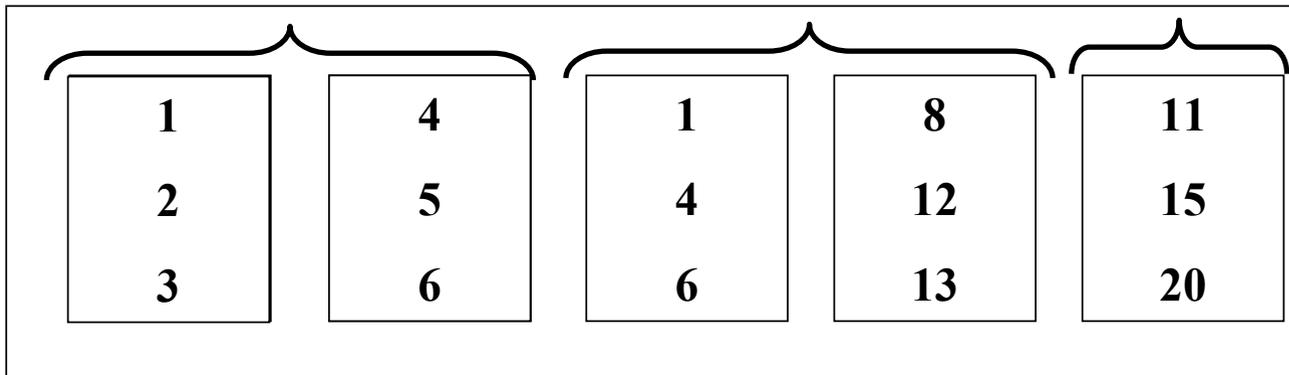
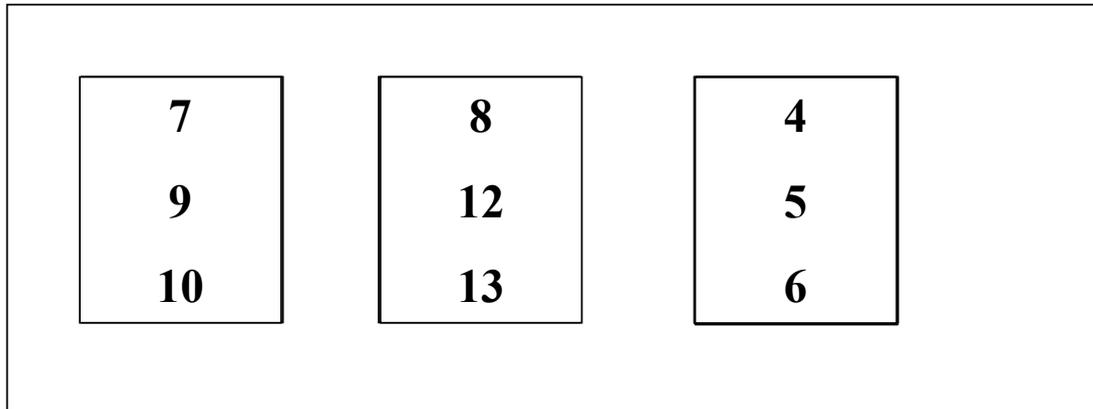
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

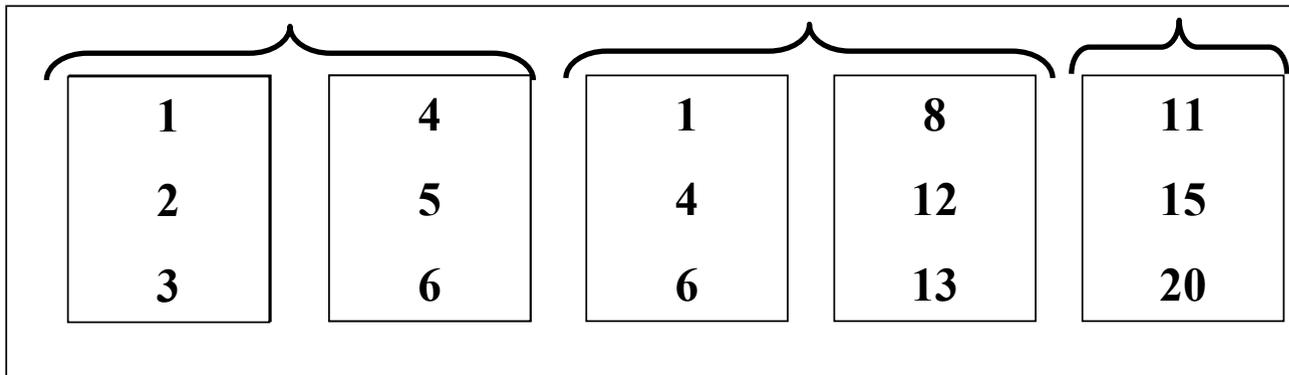
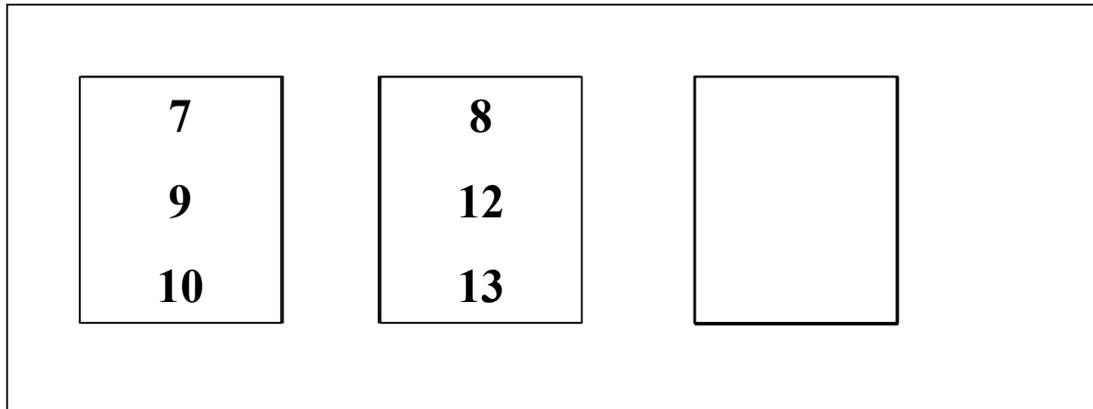
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

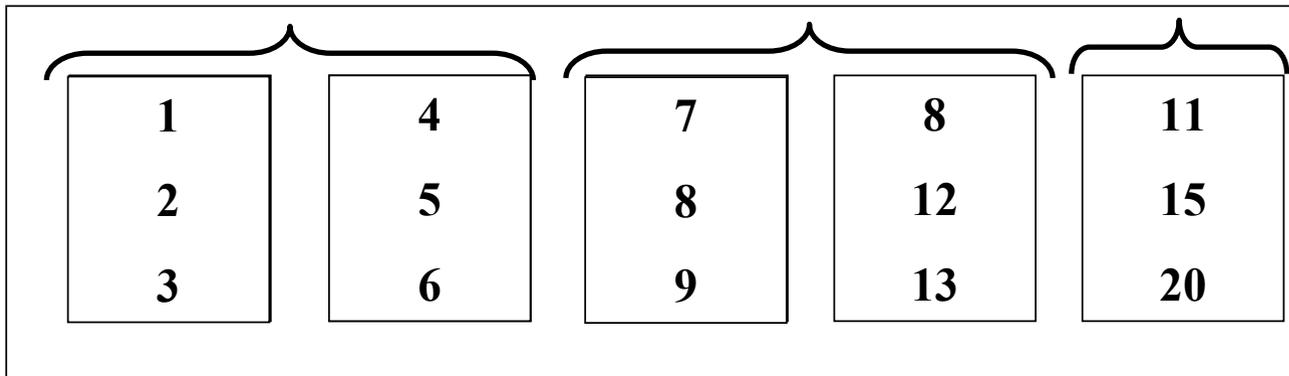
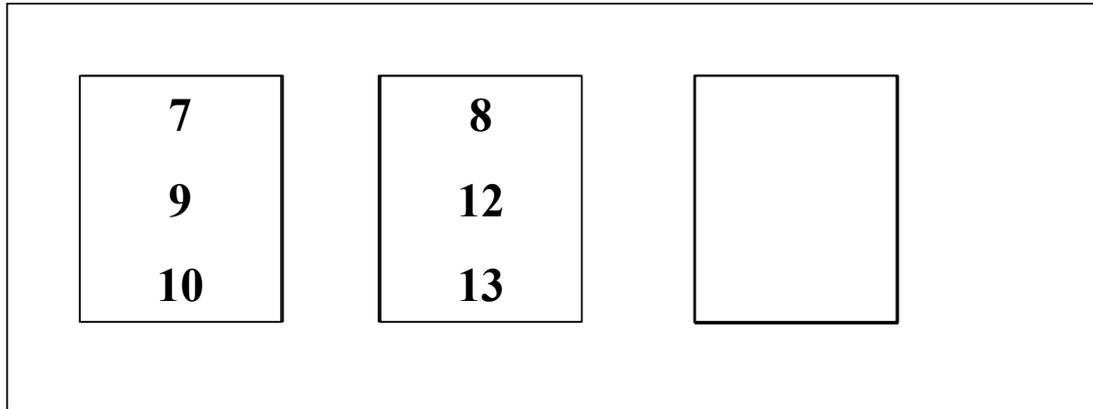
*La mémoire
contient 3
emplacements*



disque

Tri externe : 2ème étape

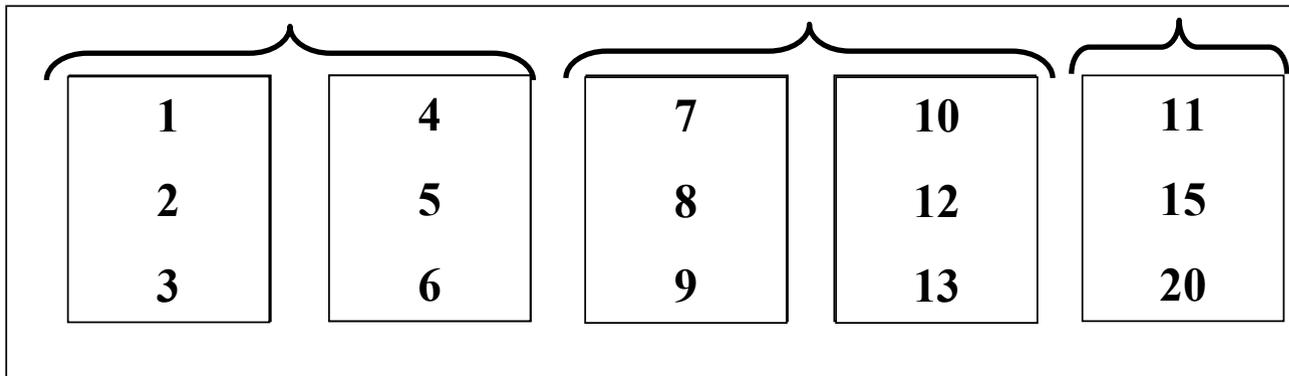
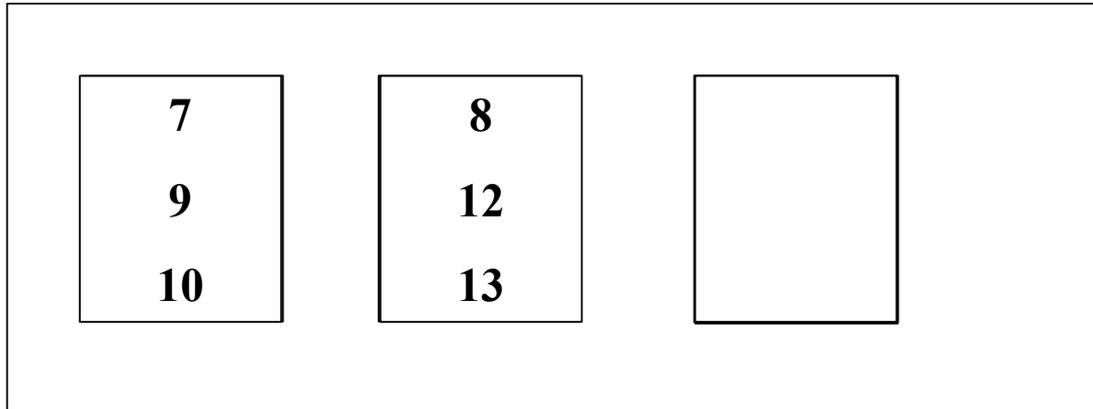
*La mémoire
contient 3
emplacements*



disque

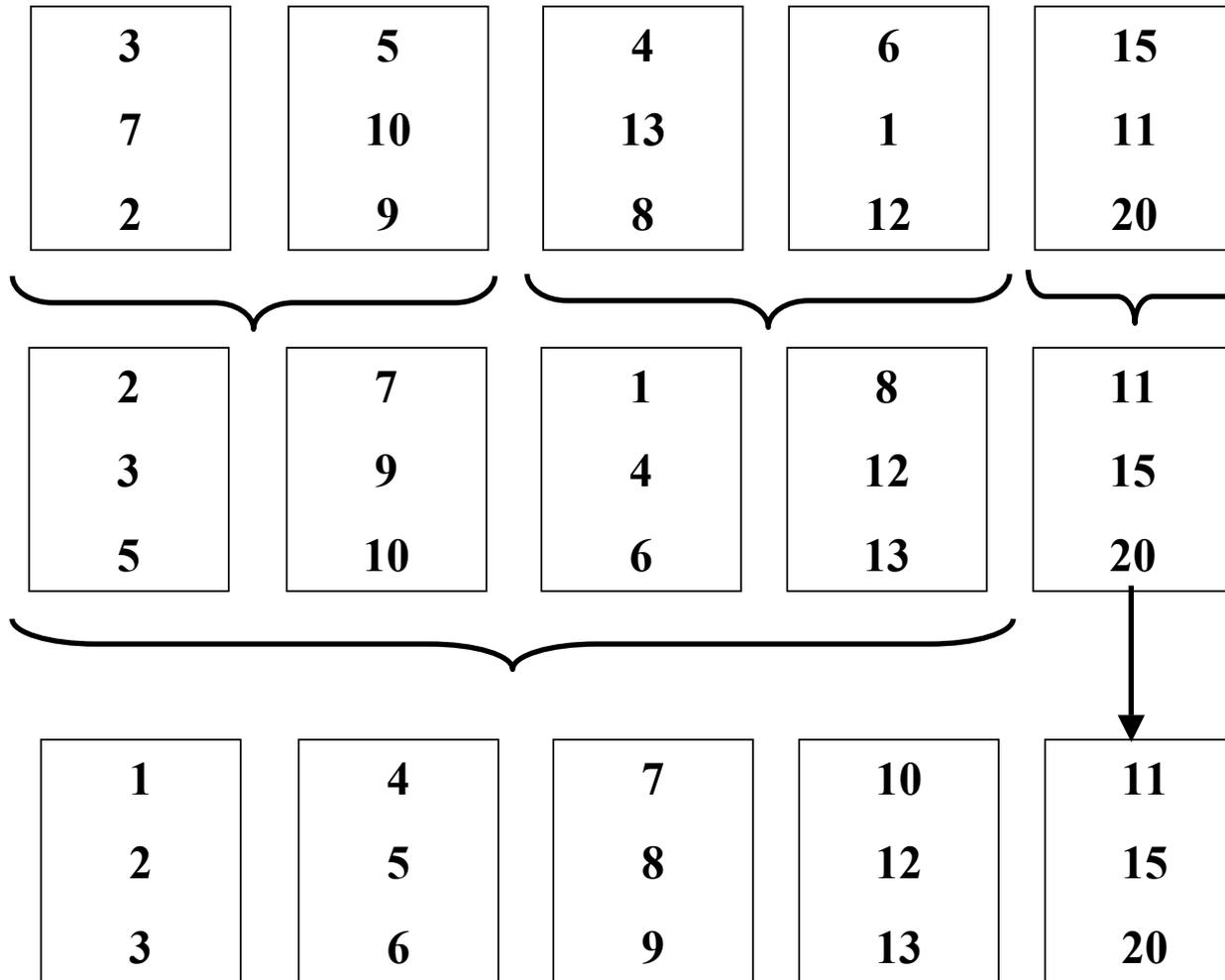
Tri externe : 2ème étape

*La mémoire
contient 3
emplacements*



disque

Tri externe



Ensemble de départ

*Après la 1ère étape de tri : Les pages sont triées par paquets de 2 ⇒ 3 paquets triés de 2 pages soit 2*5 E/S*

*Après la 2ème étape de tri : Les paquets sont triés 2 par 2 ⇒ 1 paquets trié de 4 pages et 1 paquet de 1 pages soit 2*5 E/S*

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF



SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour

Fusion

RID	Nom	Heure	Salaire	SID	Département	Jour
22	Daniel	7	145KF	28	R&D	02/02/2001
28	Jeanne	9	175KF	28	R&D	05/02/2001
31	Paul	12	200KF	31	Compta	03/02/2001
36	Pierre	6	120KF	31	Direction	05/02/2001
				31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001

Fusion

RID	Nom	Heure	Salaire
 22	Daniel	7	145KF
 28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
 28	R&D	02/02/2001
 28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001

Fusion

RID	Nom	Heure	Salaire
 22	Daniel	7	145KF
 28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
 28	R&D	02/02/2001
 28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001

Fusion

RID	Nom	Heure	Salaire
22	Daniel	7	145KF
28	Jeanne	9	175KF
31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
28	R&D	02/02/2001
28	R&D	05/02/2001
31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001

Fusion

RID	Nom	Heure	Salaire
 22	Daniel	7	145KF
 28	Jeanne	9	175KF
 31	Paul	12	200KF
36	Pierre	6	120KF

SID	Département	Jour
 28	R&D	02/02/2001
 28	R&D	05/02/2001
 31	Compta	03/02/2001
31	Direction	05/02/2001
31	Compta	06/02/2001

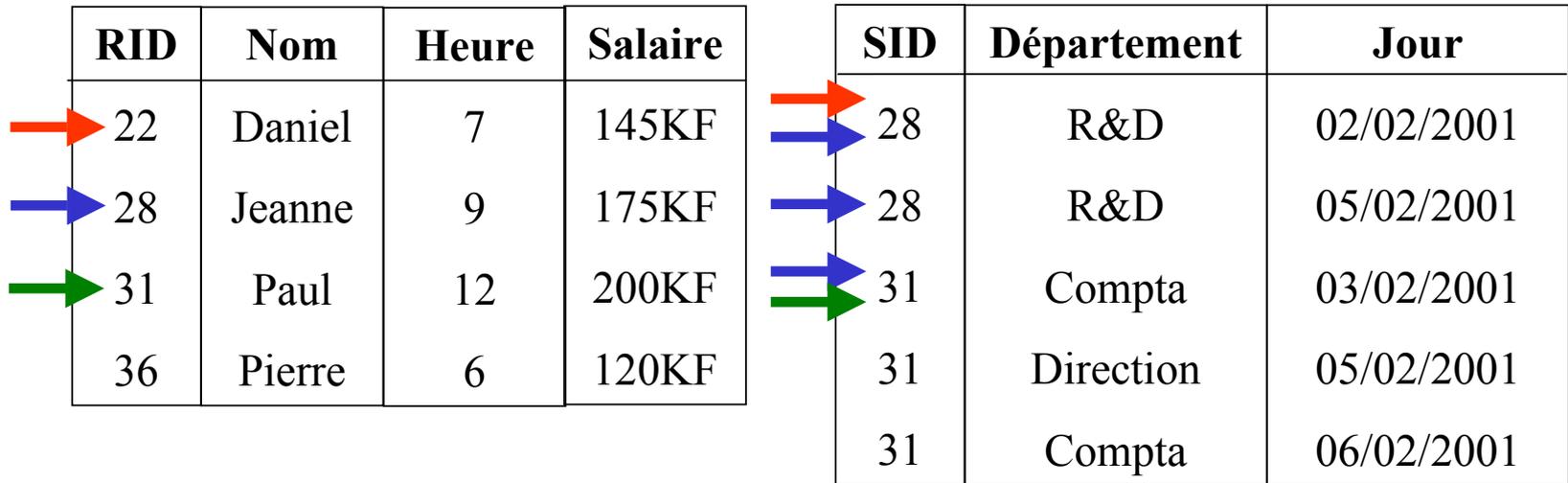
RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001

Fusion

RID	Nom	Heure	Salaire	SID	Département	Jour
22	Daniel	7	145KF	28	R&D	02/02/2001
28	Jeanne	9	175KF	28	R&D	05/02/2001
31	Paul	12	200KF	31	Compta	03/02/2001
36	Pierre	6	120KF	31	Direction	05/02/2001
				31	Compta	06/02/2001

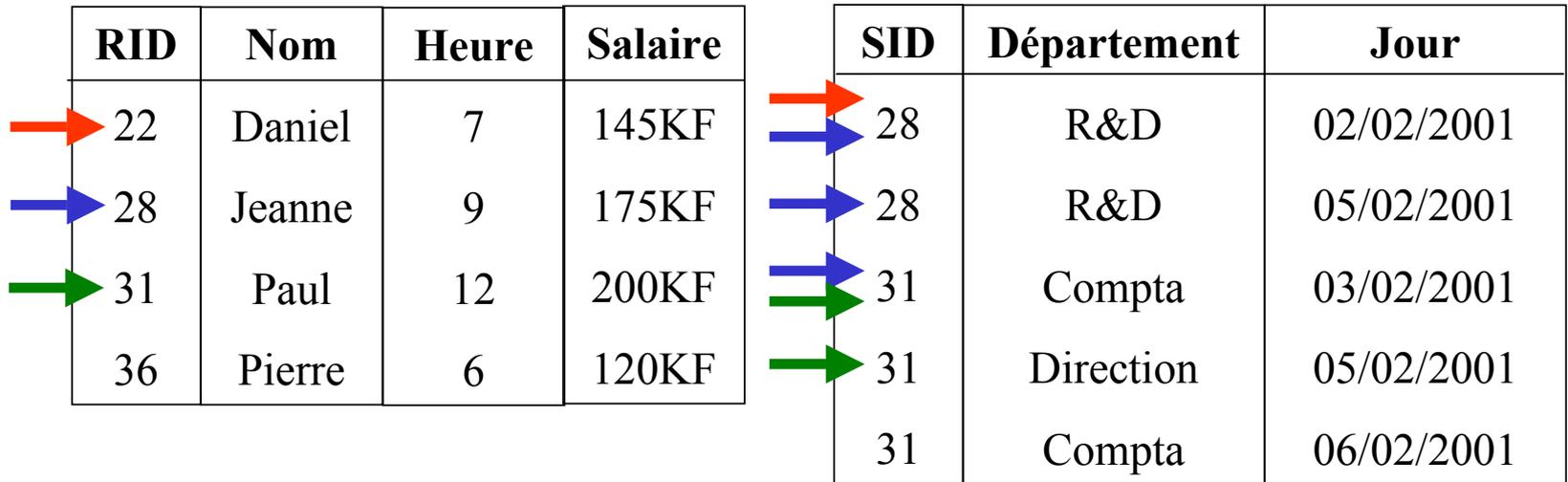
RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001

Fusion



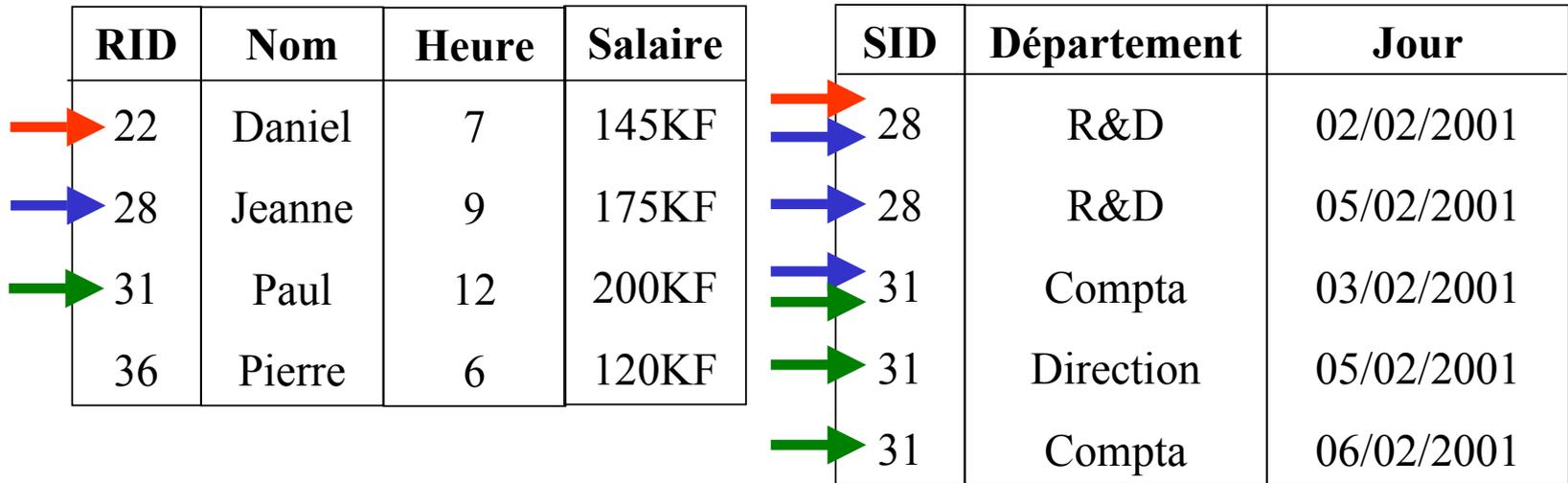
RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001
31	Paul	12	200KF	Compta	03/02/2001
...

Fusion



RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001
31	Paul	12	200KF	Compta	03/02/2001
...

Fusion



RID	Nom	Heure	Salaire	Département	Jour
28	Jeanne	9	175KF	R&D	02/02/2001
28	Jeanne	9	175KF	R&D	05/02/2001
31	Paul	12	200KF	Compta	03/02/2001
...

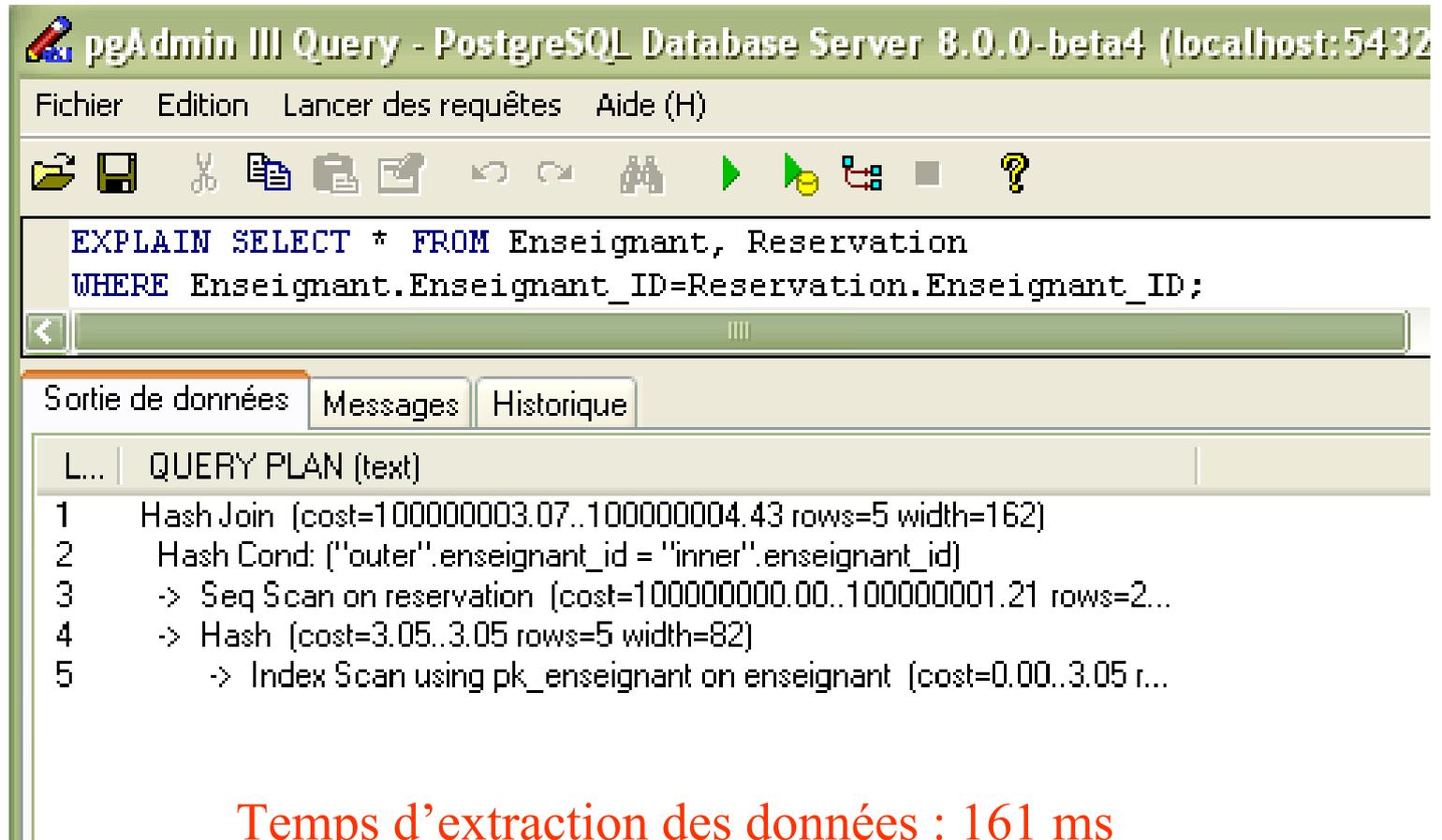
Exemples en utilisant PostgreSQL

Commandes

- **VACUUM** : Mise à jour des statistiques
- **VACUUM ANALYSE VERBOSE** : met à jour analyse et affiche le résultat de l'analyse des statistiques
- **EXPLAIN** : affiche le plan d'exécution d'une requête
- **SET ENABLE_SEQSCAN TO OFF** : interdit l'utilisation du parcours séquentiel (pour forcer l'utilisation des index)
- **CREATE INDEX "Nom_Index" ON Relation USING btree (nom)** : pour créer un index en précisant son type

Exemples en utilisant PostgreSQL

Attention à l'écriture des requêtes!!



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432)". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains various icons for file operations and execution. The query editor contains the following SQL query:

```
EXPLAIN SELECT * FROM Enseignant, Reservation
WHERE Enseignant.Enseignant_ID=Reservation.Enseignant_ID;
```

Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the following query plan:

```
L... | QUERY PLAN (text)
1   Hash Join (cost=100000003.07..100000004.43 rows=5 width=162)
2   Hash Cond: ("outer".enseignant_id = "inner".enseignant_id)
3   -> Seq Scan on reservation (cost=100000000.00..100000001.21 rows=2...)
4   -> Hash (cost=3.05..3.05 rows=5 width=82)
5   -> Index Scan using pk_enseignant on enseignant (cost=0.00..3.05 r...
```

Temps d'extraction des données : 161 ms

Exemples en utilisant PostgreSQL

pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432)

Fichier Edition Lancer des requêtes Aide (H)

```
EXPLAIN SELECT * FROM Enseignant
WHERE Enseignant_ID IN (SELECT Enseignant_ID FROM Reservation);
```

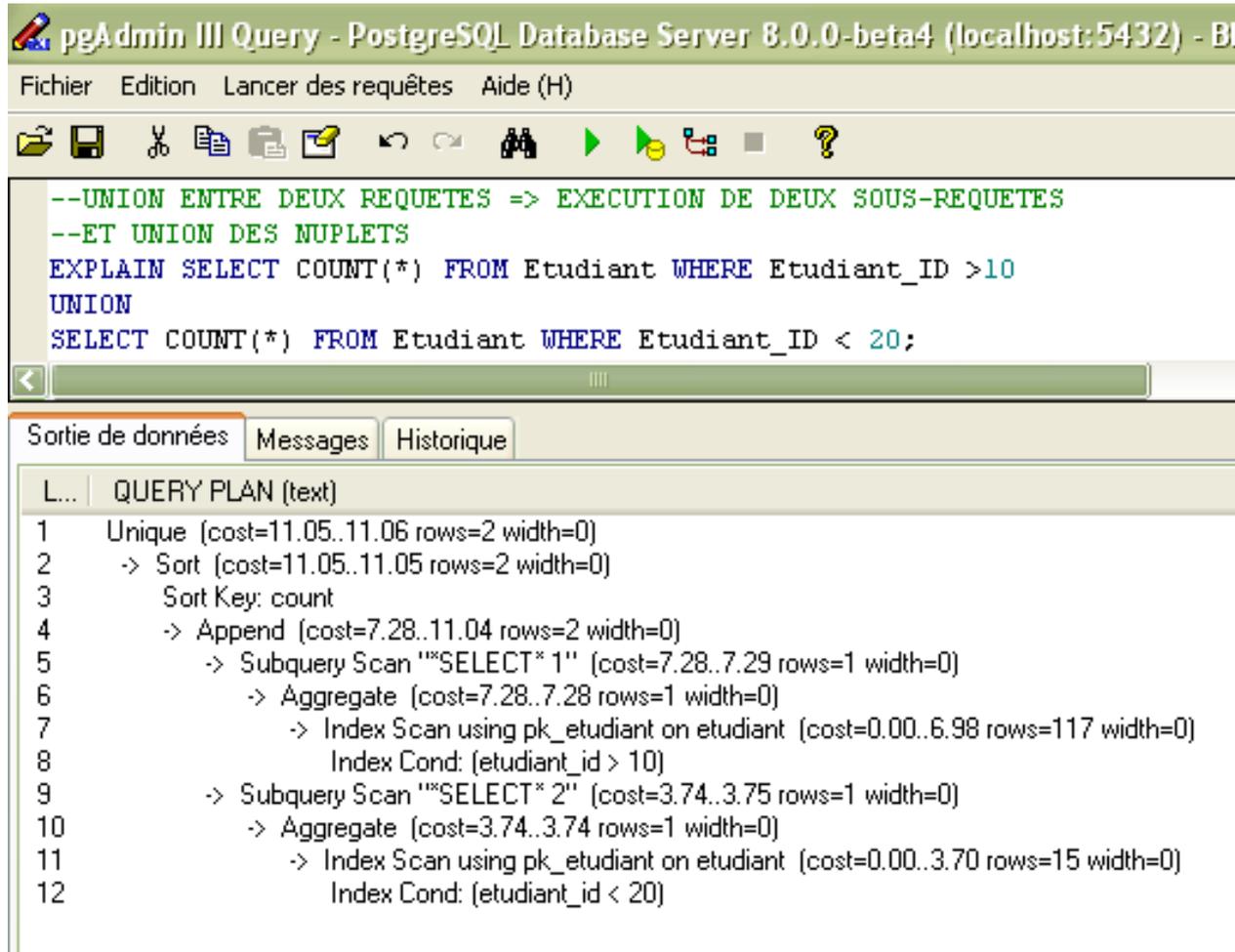
Sortie de données Messages Historique

L...	QUERY PLAN (text)
1	Hash Join (cost=100000004.33..100000004.49 rows=5 width=82)
2	Hash Cond: ("outer".enseignant_id = "inner".enseignant_id)
3	-> HashAggregate (cost=100000001.26..100000001.26 rows=21 width=4)
4	-> Seq Scan on reservation (cost=100000000.00..100000001.21 row...)
5	-> Hash (cost=3.05..3.05 rows=5 width=82)
6	-> Index Scan using pk_enseignant on enseignant (cost=0.00..3.05 r...

Ces opérations ont été ajoutées par rapport à l'exécution de la requête précédente

Temps d'extraction des données : 250 ms (pour un requête donnant le même résultat que précédemment)

Exemples en utilisant PostgreSQL



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - B...". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations and execution. The SQL editor contains the following text:

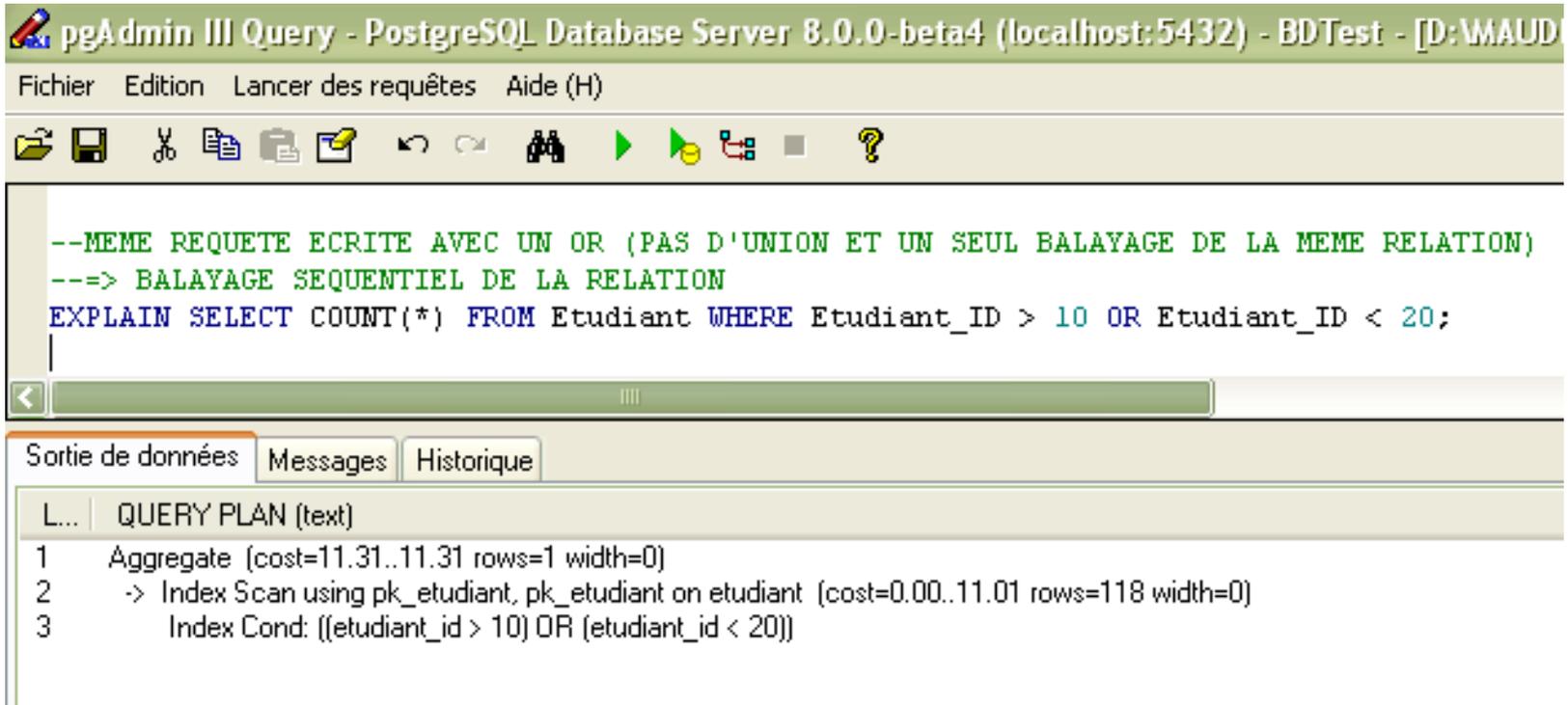
```
--UNION ENTRE DEUX REQUETES => EXECUTION DE DEUX SOUS-REQUETES
--ET UNION DES NUPLETS
EXPLAIN SELECT COUNT(*) FROM Etudiant WHERE Etudiant_ID >10
UNION
SELECT COUNT(*) FROM Etudiant WHERE Etudiant_ID < 20;
```

Below the editor, the "Sortie de données" tab is active, displaying the "QUERY PLAN (text)" output:

```
L... | QUERY PLAN (text)
1   Unique (cost=11.05..11.06 rows=2 width=0)
2   -> Sort (cost=11.05..11.05 rows=2 width=0)
3     Sort Key: count
4     -> Append (cost=7.28..11.04 rows=2 width=0)
5       -> Subquery Scan ""SELECT* 1"" (cost=7.28..7.29 rows=1 width=0)
6         -> Aggregate (cost=7.28..7.28 rows=1 width=0)
7           -> Index Scan using pk_etudiant on etudiant (cost=0.00..6.98 rows=117 width=0)
8             Index Cond: (etudiant_id > 10)
9       -> Subquery Scan ""SELECT* 2"" (cost=3.74..3.75 rows=1 width=0)
10        -> Aggregate (cost=3.74..3.74 rows=1 width=0)
11          -> Index Scan using pk_etudiant on etudiant (cost=0.00..3.70 rows=15 width=0)
12            Index Cond: (etudiant_id < 20)
```

Temps d'extraction des données : 280ms

Exemples en utilisant PostgreSQL



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTest - [D:\MAUD...". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations and execution. The main text area contains the following SQL query:

```
--MEME REQUETE ECRITE AVEC UN OR (PAS D'UNION ET UN SEUL BALAYAGE DE LA MEME RELATION)  
--=> BALAYAGE SEQUENTIEL DE LA RELATION  
EXPLAIN SELECT COUNT(*) FROM Etudiant WHERE Etudiant_ID > 10 OR Etudiant_ID < 20;
```

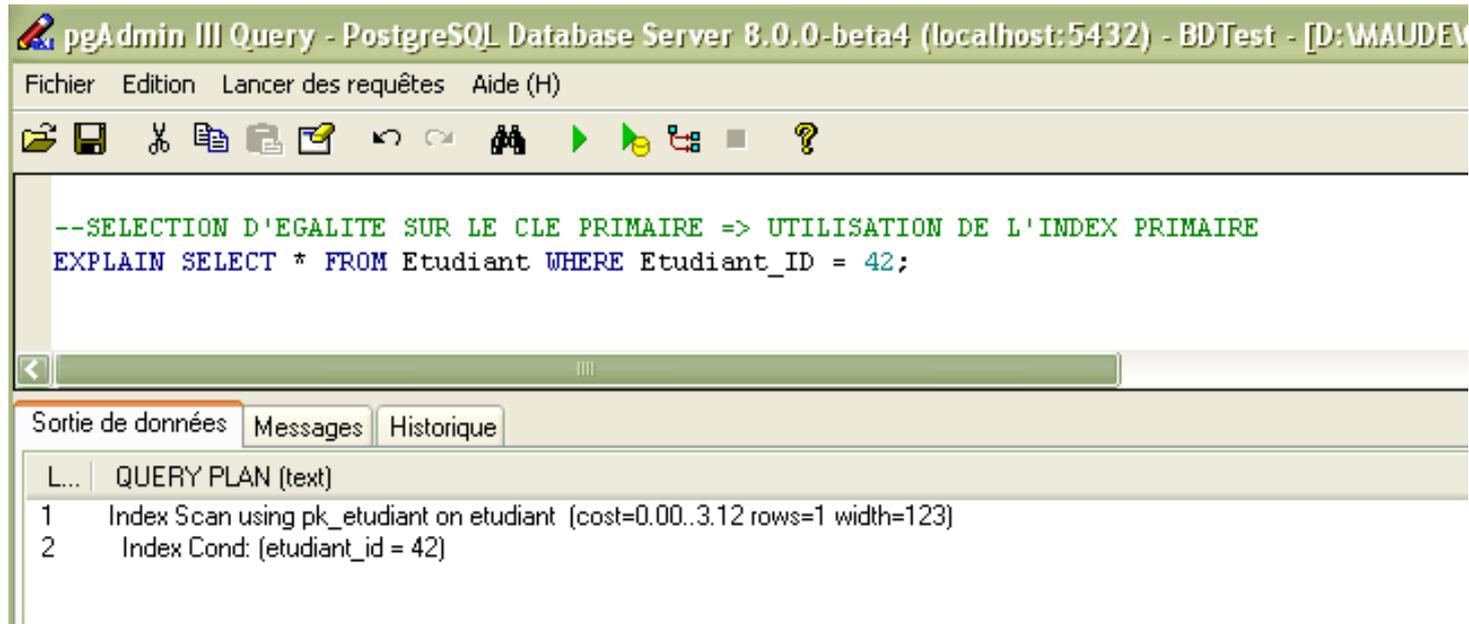
Below the query, the "Sortie de données" tab is active, showing the "QUERY PLAN (text)" for the executed query:

```
1  Aggregate (cost=11.31..11.31 rows=1 width=0)  
2  ->  Index Scan using pk_etudiant, pk_etudiant on etudiant (cost=0.00..11.01 rows=118 width=0)  
3      Index Cond: ((etudiant_id > 10) OR (etudiant_id < 20))
```

Temps d'extraction des données : 231ms (pour une requête donnant le même résultat)

Exemples en utilisant PostgreSQL

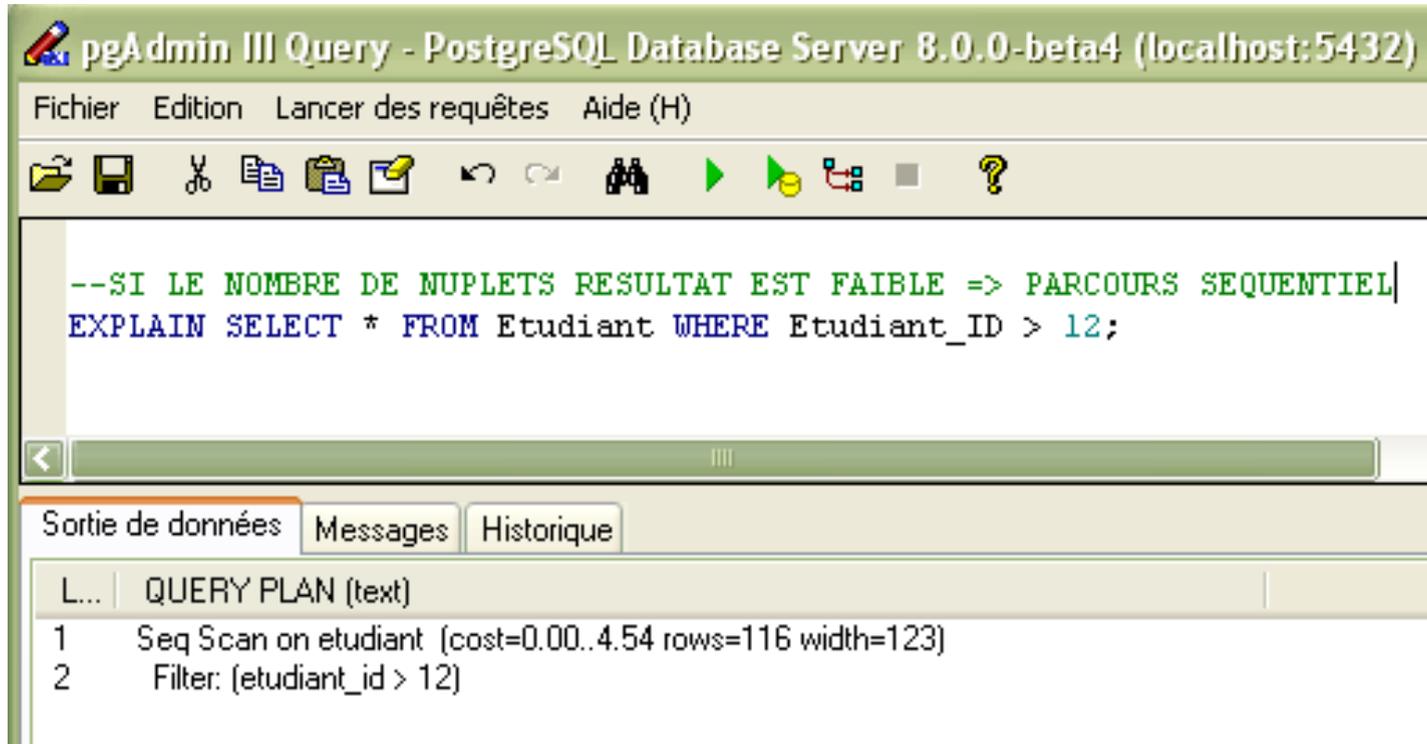
Utilisation des index : quand un balayage séquentiel est plus coûteux



Temps d'extraction des données : 231ms

Exemples en utilisant PostgreSQL

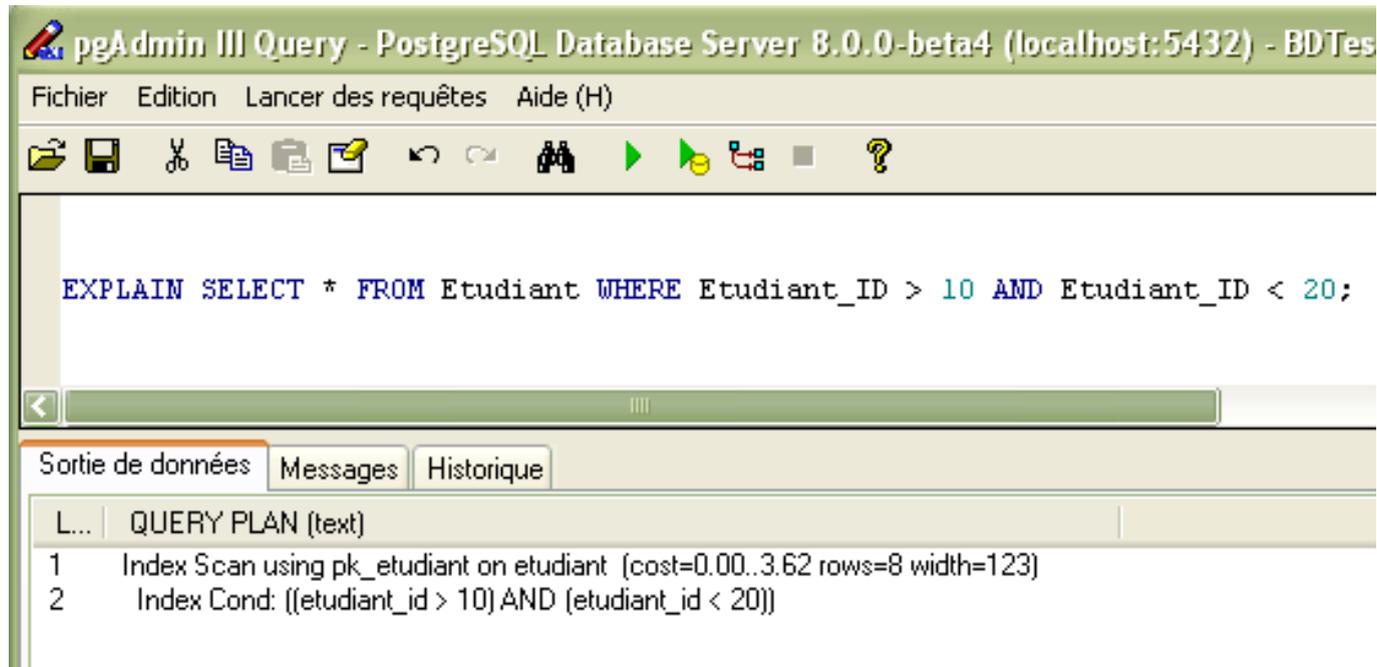
Si l'index n'est pas utile pour la requête, il n'est pas utilisé



Temps d'extraction des données : 220ms

Exemples en utilisant PostgreSQL

Si l'index est utile pour la requête, il est utilisé



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTes". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, undo, redo, and execution. The query editor contains the following SQL query:

```
EXPLAIN SELECT * FROM Etudiant WHERE Etudiant_ID > 10 AND Etudiant_ID < 20;
```

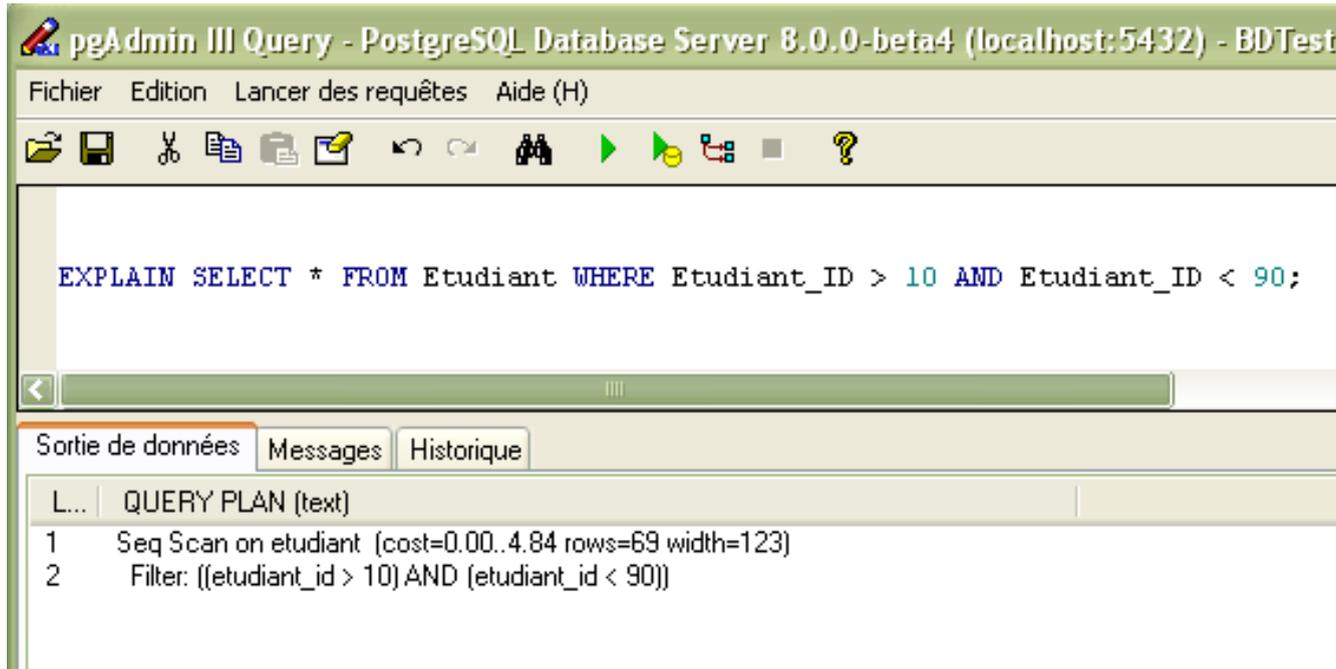
Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the query plan:

```
L... | QUERY PLAN (text)
1   | Index Scan using pk_etudiant on etudiant (cost=0.00..3.62 rows=8 width=123)
2   |   Index Cond: ((etudiant_id > 10) AND (etudiant_id < 20))
```

Temps d'extraction des données : 230ms

Exemples en utilisant PostgreSQL

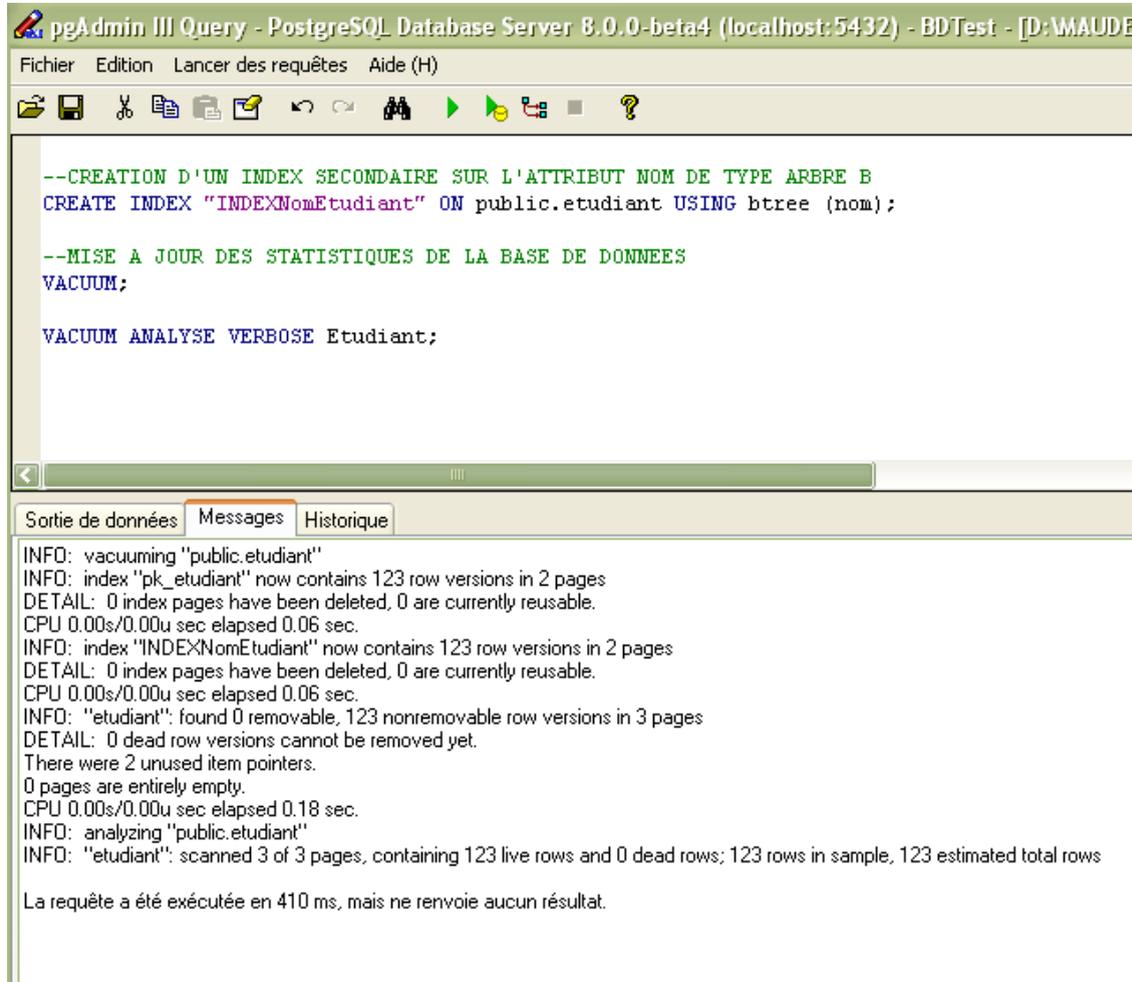
L'utilisation des index dépend du nombre de nuplets potentiellement résultats



Temps d'extraction des données : 231ms

Exemples en utilisant PostgreSQL

Création d'un index



The screenshot shows the pgAdmin III Query tool interface. The title bar indicates the connection to a PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTest - [D:WAUDE]. The menu bar includes 'Fichier', 'Edition', 'Lancer des requêtes', and 'Aide (H)'. The toolbar contains various icons for file operations and execution. The main query area contains the following SQL commands:

```
--CREATION D'UN INDEX SECONDAIRE SUR L'ATTRIBUT NOM DE TYPE ARBRE B
CREATE INDEX "INDEXNomEtudiant" ON public.etudiant USING btree (nom);

--MISE A JOUR DES STATISTIQUES DE LA BASE DE DONNEES
VACUUM;

VACUUM ANALYSE VERBOSE Etudiant;
```

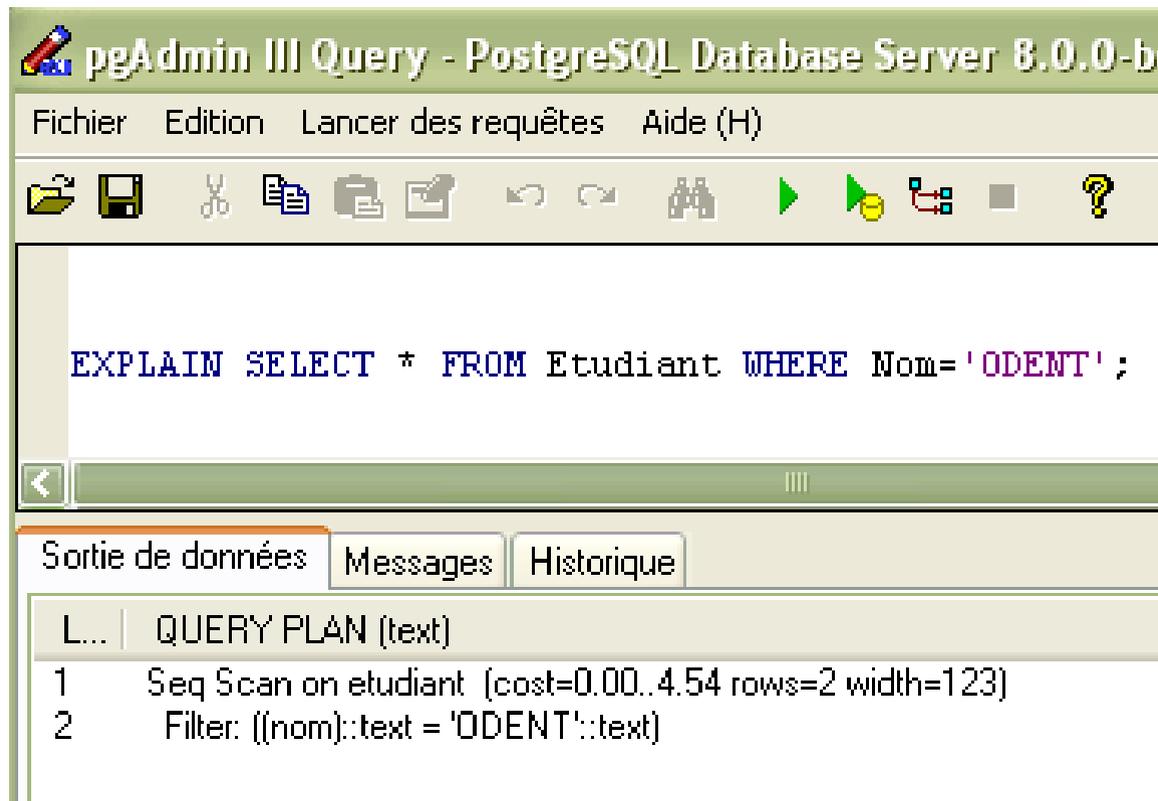
The output pane shows the following messages:

```
Sortie de données Messages Historique
INFO: vacuuming "public.etudiant"
INFO: index "pk_etudiant" now contains 123 row versions in 2 pages
DETAIL: 0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.06 sec.
INFO: index "INDEXNomEtudiant" now contains 123 row versions in 2 pages
DETAIL: 0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.06 sec.
INFO: "etudiant": found 0 removable, 123 nonremovable row versions in 3 pages
DETAIL: 0 dead row versions cannot be removed yet.
There were 2 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.18 sec.
INFO: analyzing "public.etudiant"
INFO: "etudiant": scanned 3 of 3 pages, containing 123 live rows and 0 dead rows; 123 rows in sample, 123 estimated total rows

La requête a été exécutée en 410 ms, mais ne renvoie aucun résultat.
```

Exemples en utilisant PostgreSQL

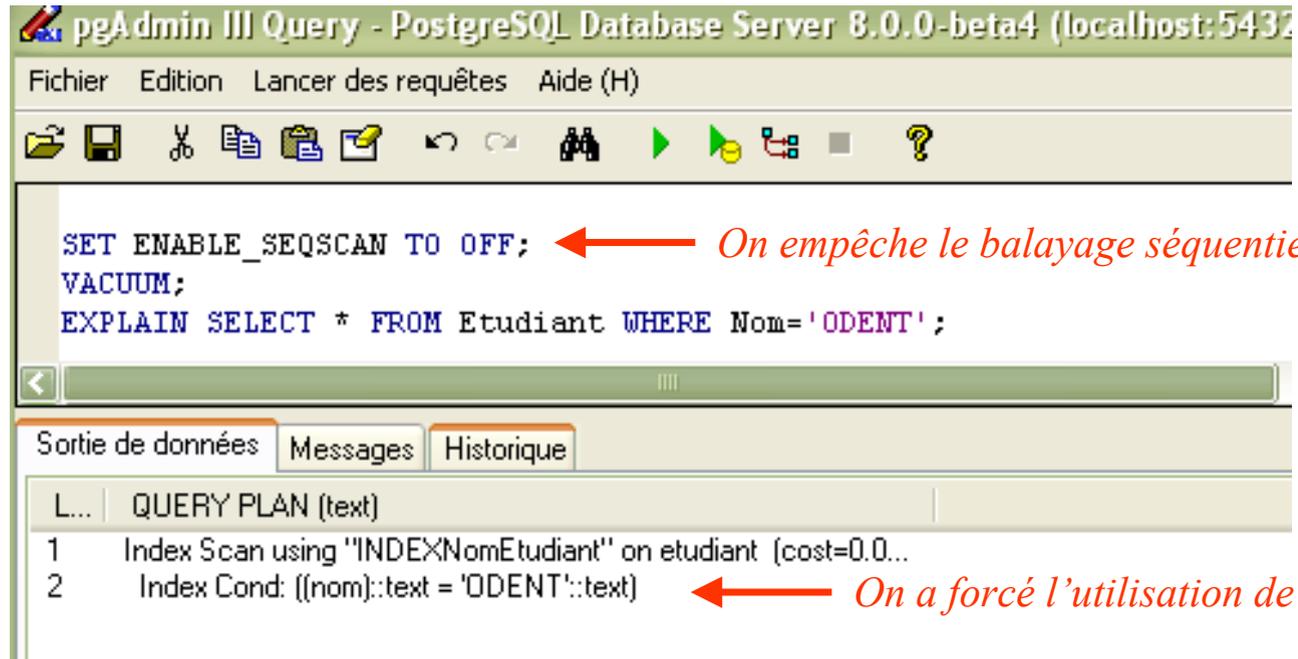
Le SGBD peut choisir de ne pas utiliser les index



Temps d'extraction des données : 221ms

Exemples en utilisant PostgreSQL

On peut forcer l'utilisation des index



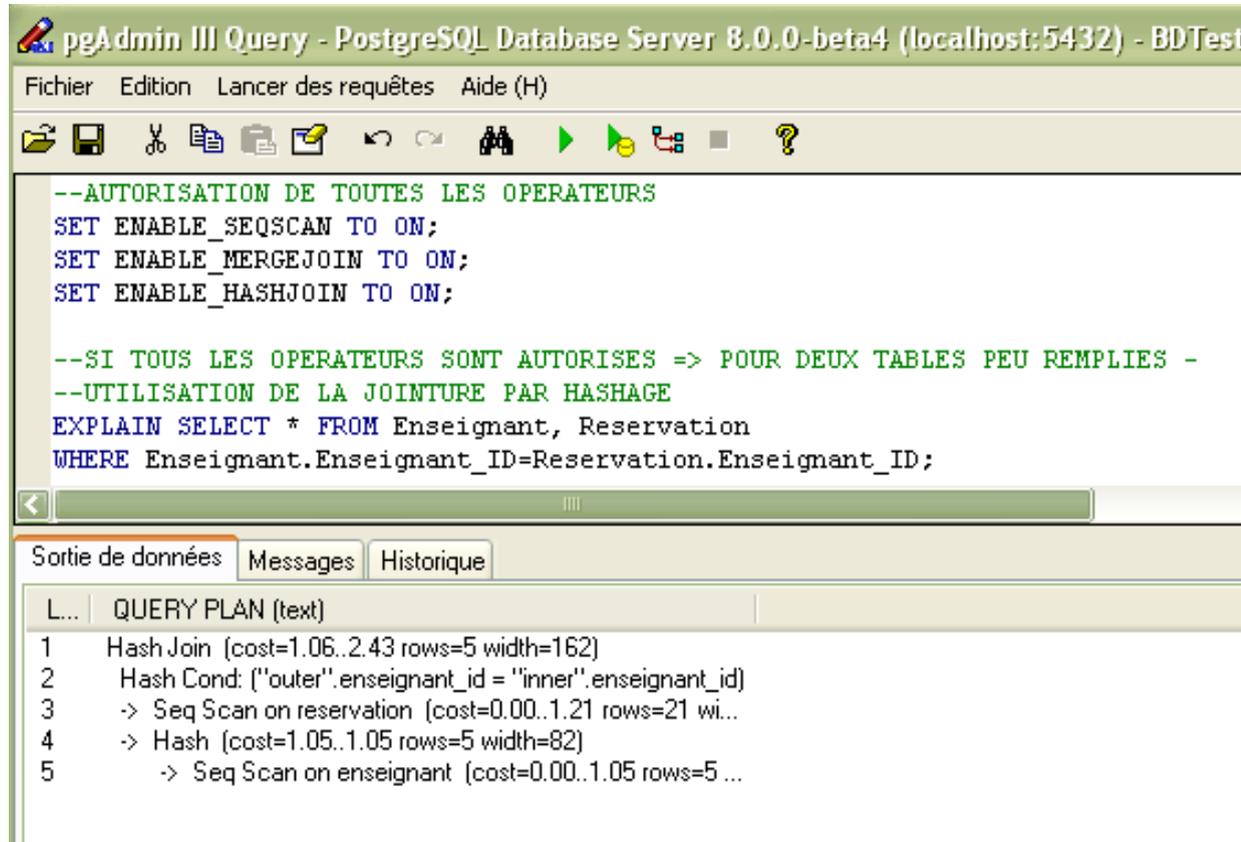
← *On empêche le balayage séquentiel*

← *On a forcé l'utilisation de l'index*

Temps d'extraction des données : 220ms

Exemples en utilisant PostgreSQL

Algorithmes de jointure



The screenshot shows the pgAdmin III Query tool interface. The title bar indicates the connection to a PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTest. The menu bar includes Fichier, Edition, Lancer des requêtes, and Aide (H). The toolbar contains icons for file operations and execution. The main text area contains the following SQL code:

```
--AUTORISATION DE TOUTES LES OPERATEURS
SET ENABLE_SEQSCAN TO ON;
SET ENABLE_MERGEJOIN TO ON;
SET ENABLE_HASHJOIN TO ON;

--SI TOUS LES OPERATEURS SONT AUTORISES => POUR DEUX TABLES PEU REMPLIES -
--UTILISATION DE LA JOINTURE PAR HASHAGE
EXPLAIN SELECT * FROM Enseignant, Reservation
WHERE Enseignant.Enseignant_ID=Reservation.Enseignant_ID;
```

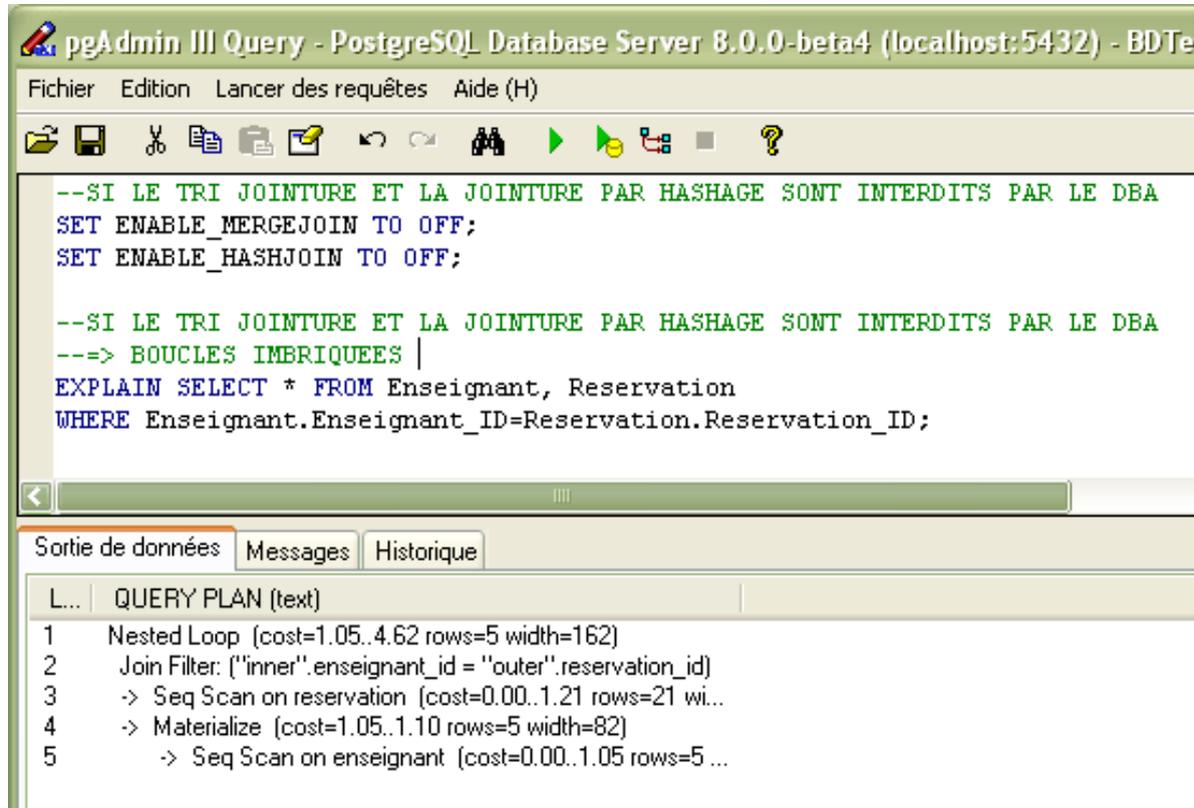
Below the text area, there are tabs for 'Sortie de données', 'Messages', and 'Historique'. The 'Sortie de données' tab is active, displaying the following query plan:

L...	QUERY PLAN (text)
1	Hash Join (cost=1.06..2.43 rows=5 width=162)
2	Hash Cond: ("outer".enseignant_id = "inner".enseignant_id)
3	-> Seq Scan on reservation (cost=0.00..1.21 rows=21 wi...
4	-> Hash (cost=1.05..1.05 rows=5 width=82)
5	-> Seq Scan on enseignant (cost=0.00..1.05 rows=5 ...)

Temps d'extraction des données : 201ms

Exemples en utilisant PostgreSQL

Algorithmes de jointure



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTe". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, execution, and help. The main text area contains the following SQL code:

```
--SI LE TRI JOINTURE ET LA JOINTURE PAR HASHAGE SONT INTERDITS PAR LE DBA
SET ENABLE_MERGEJOIN TO OFF;
SET ENABLE_HASHJOIN TO OFF;

--SI LE TRI JOINTURE ET LA JOINTURE PAR HASHAGE SONT INTERDITS PAR LE DBA
--=> BOUCLES IMBRIQUEES |
EXPLAIN SELECT * FROM Enseignant, Reservation
WHERE Enseignant.Enseignant_ID=Reservation.Reservation_ID;
```

Below the code, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the following query plan:

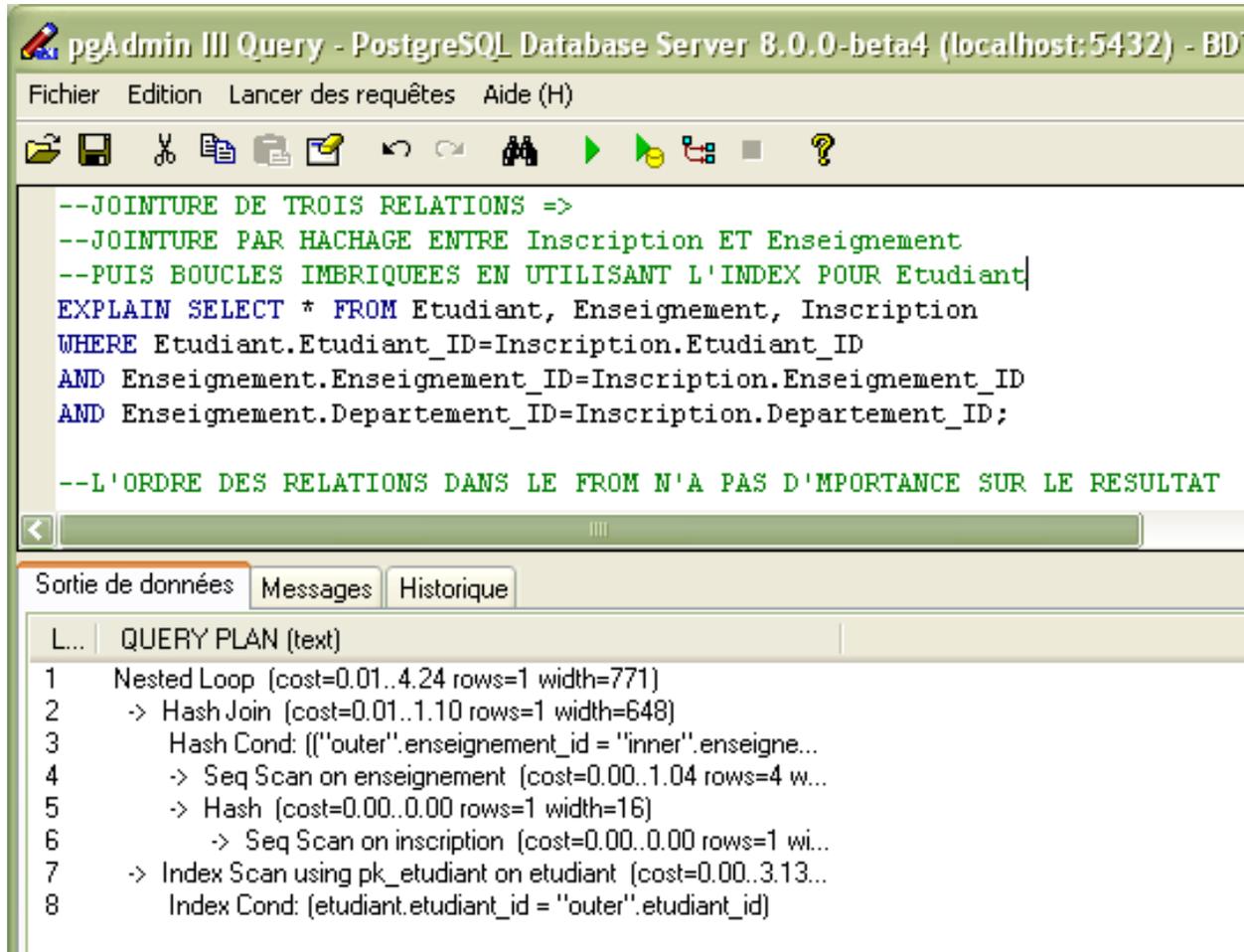
L...	QUERY PLAN (text)
1	Nested Loop (cost=1.05..4.62 rows=5 width=162)
2	Join Filter: ("inner".enseignant_id = "outer".reservation_id)
3	-> Seq Scan on reservation (cost=0.00..1.21 rows=21 wi...
4	-> Materialize (cost=1.05..1.10 rows=5 width=82)
5	-> Seq Scan on enseignant (cost=0.00..1.05 rows=5 ...)

Temps d'extraction des données : 190ms

Exemples en utilisant PostgreSQL

Algorithmes de jointure

Temps
d'extraction
des données :
200ms



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BD". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, editing, and execution. The main text area contains the following SQL query:

```
--JOINTURE DE TROIS RELATIONS =>
--JOINTURE PAR HACHAGE ENTRE Inscription ET Enseignement
--PUIS BOUCLES IMBRIQUEES EN UTILISANT L'INDEX POUR Etudiant
EXPLAIN SELECT * FROM Etudiant, Enseignement, Inscription
WHERE Etudiant.Etudiant_ID=Inscription.Etudiant_ID
AND Enseignement.Enseignement_ID=Inscription.Enseignement_ID
AND Enseignement.Departement_ID=Inscription.Departement_ID;

--L'ORDRE DES RELATIONS DANS LE FROM N'A PAS D'IMPORTANCE SUR LE RESULTAT
```

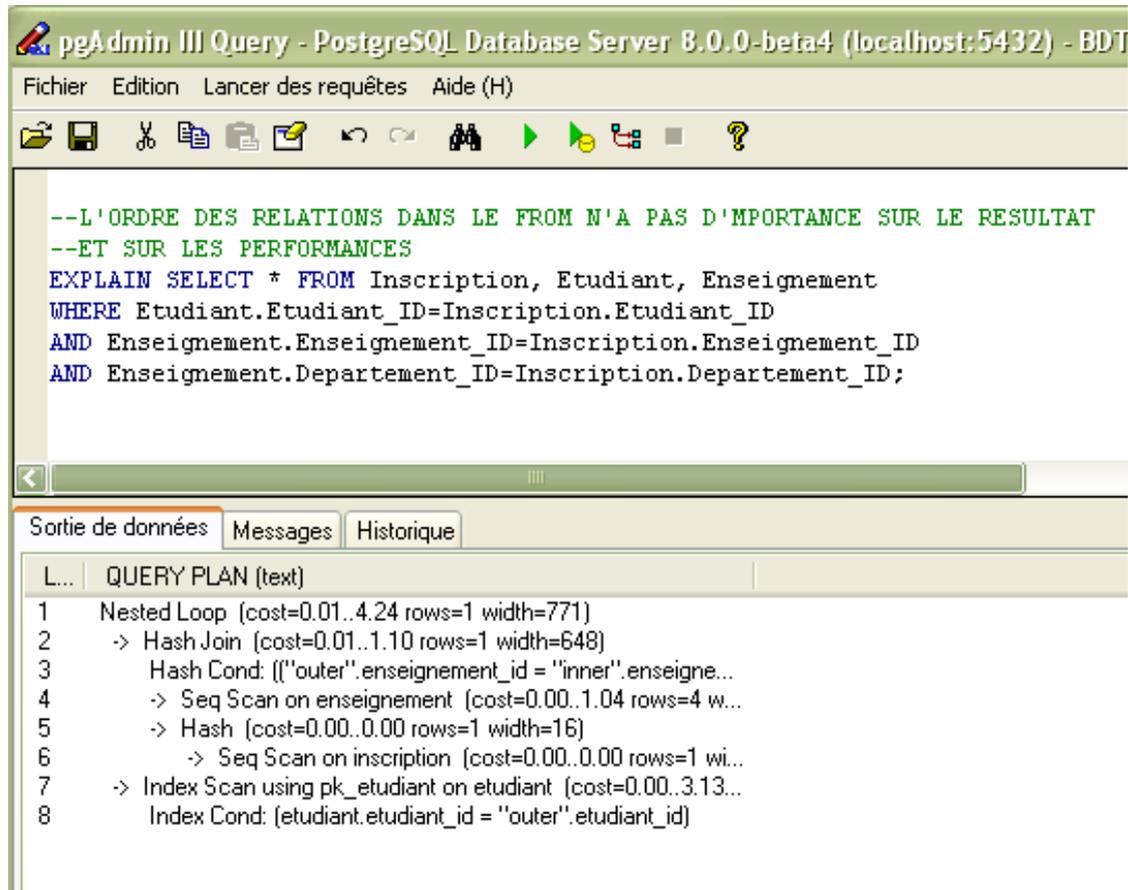
Below the query, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the following query plan:

```
L... QUERY PLAN (text)
1  Nested Loop (cost=0.01..4.24 rows=1 width=771)
2    -> Hash Join (cost=0.01..1.10 rows=1 width=648)
3      Hash Cond: (["outer".enseignement_id = "inner".enseigne...
4        -> Seq Scan on enseignement (cost=0.00..1.04 rows=4 w...
5          -> Hash (cost=0.00..0.00 rows=1 width=16)
6            -> Seq Scan on inscription (cost=0.00..0.00 rows=1 wi...
7        -> Index Scan using pk_etudiant on etudiant (cost=0.00..3.13...
8          Index Cond: (etudiant.etudiant_id = "outer".etudiant_id)
```

Exemples en utilisant PostgreSQL

Algorithmes de jointure

Temps
d'extraction
des données :
200ms



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDT". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations and execution. The main text area contains the following SQL query:

```
--L'ORDRE DES RELATIONS DANS LE FROM N'A PAS D'IMPORTANCE SUR LE RESULTAT  
--ET SUR LES PERFORMANCES  
EXPLAIN SELECT * FROM Inscription, Etudiant, Enseignement  
WHERE Etudiant.Etudiant_ID=Inscription.Etudiant_ID  
AND Enseignement.Enseignement_ID=Inscription.Enseignement_ID  
AND Enseignement.Departement_ID=Inscription.Departement_ID;
```

Below the query, the "Sortie de données" tab is active, displaying the "QUERY PLAN (text)" for the executed query. The plan is as follows:

```
1  Nested Loop (cost=0.01..4.24 rows=1 width=771)  
2  -> Hash Join (cost=0.01..1.10 rows=1 width=648)  
3      Hash Cond: (["outer".enseignement_id = "inner".enseigne...  
4          -> Seq Scan on enseignement (cost=0.00..1.04 rows=4 w...  
5          -> Hash (cost=0.00..0.00 rows=1 width=16)  
6              -> Seq Scan on inscription (cost=0.00..0.00 rows=1 wi...  
7  -> Index Scan using pk_etudiant on etudiant (cost=0.00..3.13...  
8      Index Cond: (etudiant.etudiant_id = "outer".etudiant_id)
```

Coût de $\pi_A R$

- **Élimination des attributs n'apparaissant pas dans la projection**
- **Élimination des doublons**
 - ◆ **Par tri**
 - ◆ **Par hachage**

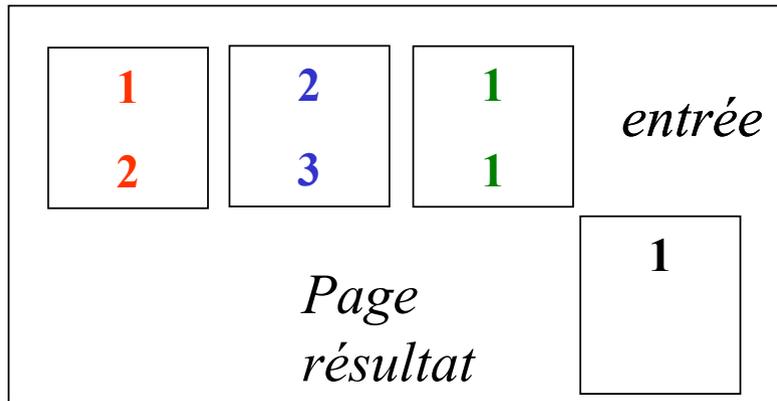
Élimination des doublons par tri

2, 5, 2, 1, 2, 2, 4, 5, 4, 3, 4, 2, 1, 5, 2, 1, 3

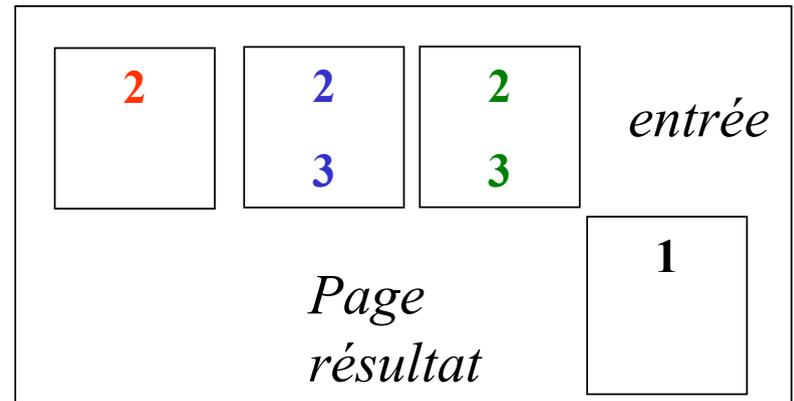
17 nuplets 2 nuplets / bloc
4 pages en mémoire

1, 2, 2, 2, 2, 5, 2, 3, 4, 4, 4, 5, 1, 1, 2, 3, 5

Étape 1 : Tri des blocs par paquets de 3 en mémoire



Étape 2 : la valeur 1 est la plus petite valeur, on l'écrit dans le résultat et on supprime les doublons



Étape 3 : la valeur 1 est la plus petite valeur, on l'écrit dans le résultat et on supprime les doublons ...

Chap. IV - Gestion de la concurrence

Transaction : action ou série d'actions d'un utilisateur ou d'une application, qui accède(nt) ou modifie(nt) les données de la base

[BEGIN TRANSACTION]

...

COMMIT / ROLLBACK

- Lecture \Rightarrow Placement des pages en mémoire + Copies éventuelles de valeurs dans les variables de programme
- Ecriture \Rightarrow Mise à jour des données en mémoire + Ecriture des pages sur le disque APRES validation

Propriétés des transactions

- **Atomicité** : Tout ou rien

Une transaction effectue toutes ses actions ou aucune.

En cas d'annulation, les modifications engagées doivent être défaites.

- **Cohérence** : Intégrité des données

Passage d'un état cohérent de la base à un autre état cohérent de la base de données

- **Isolation** : Pas d'interférence entre transactions

Les résultats d'une transaction ne sont visibles par les autres transactions qu'après sa validation

- **Durabilité** : Journalisation des mises à jour

Les modifications effectuées sont garanties même en cas de panne

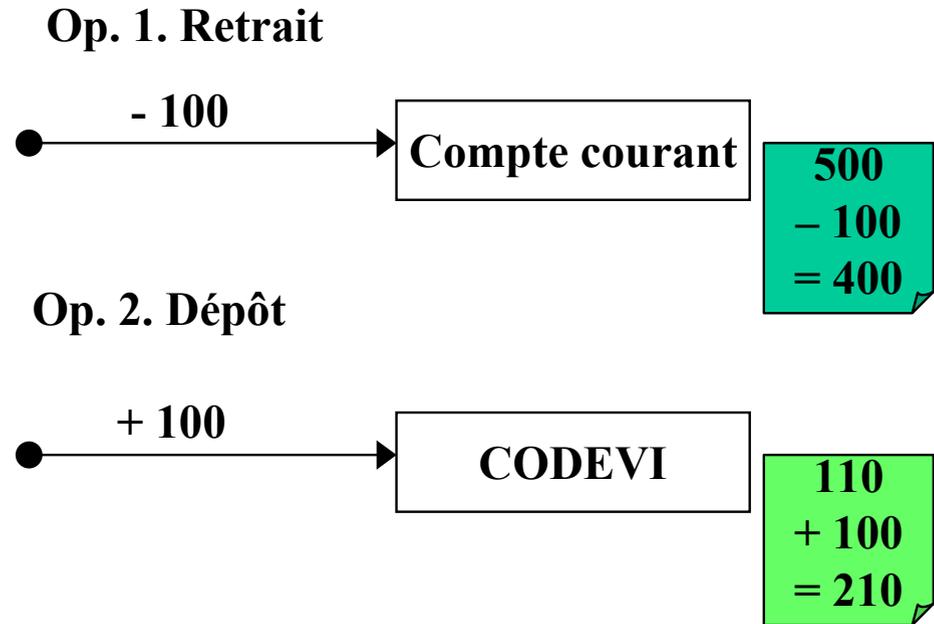
Exemple de transaction

Exemple de transaction

Virement =
2 opérations atomiques

Exemple de transaction

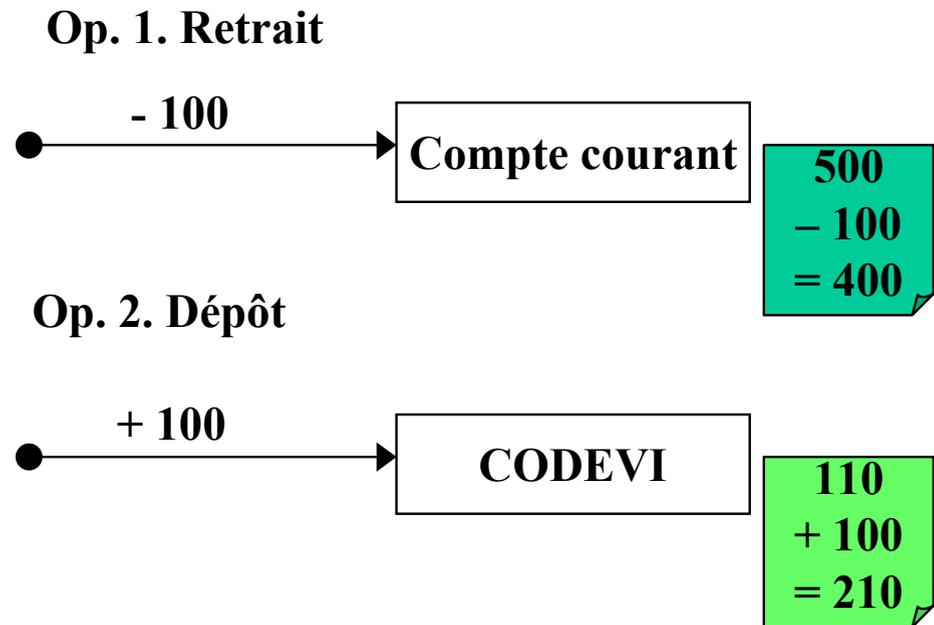
Virement =
2 opérations atomiques



Exemple de transaction

Virement bancaire **sans transaction**

Virement =
2 opérations atomiques

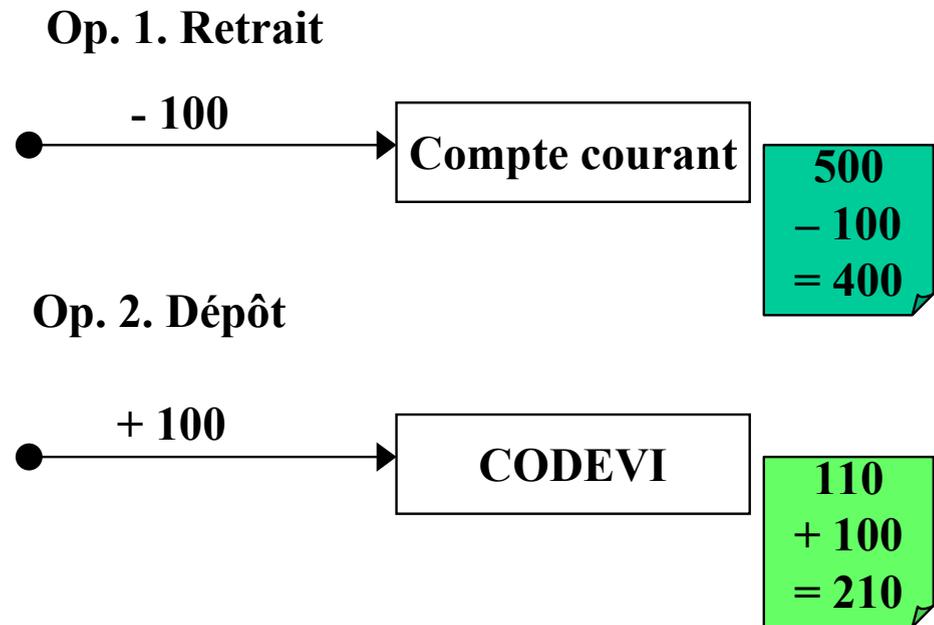


Exemple de transaction

Virement bancaire **sans transaction**

Virement =
2 opérations atomiques

Que se passe-t-il si le
Dépôt échoue ?

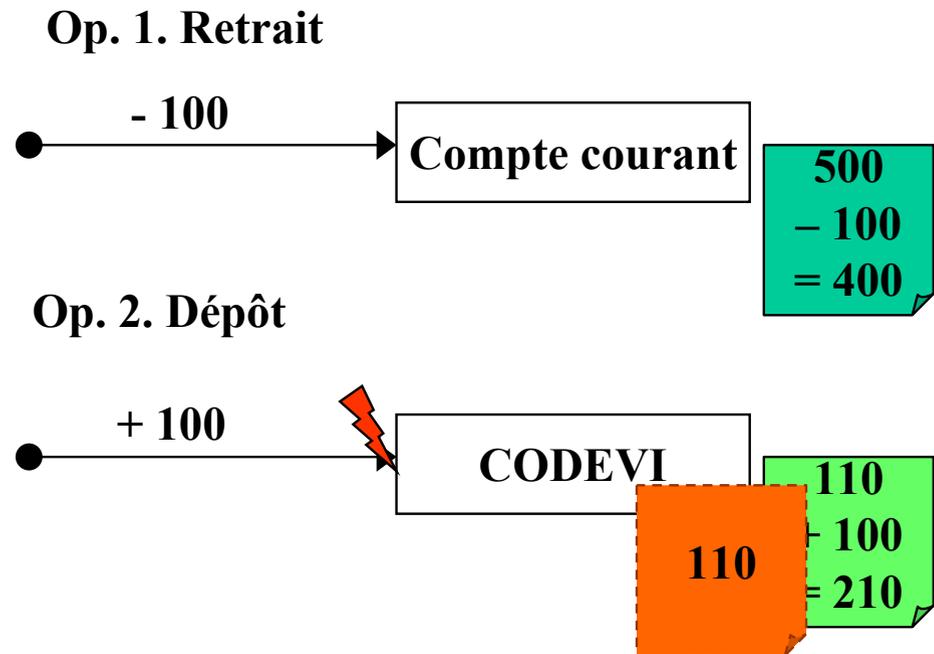


Exemple de transaction

Virement bancaire **sans transaction**

Virement =
2 opérations atomiques

Que se passe-t-il si le
Dépôt échoue ?



Exemple de transaction

Virement bancaire **sans transaction**

Virement =

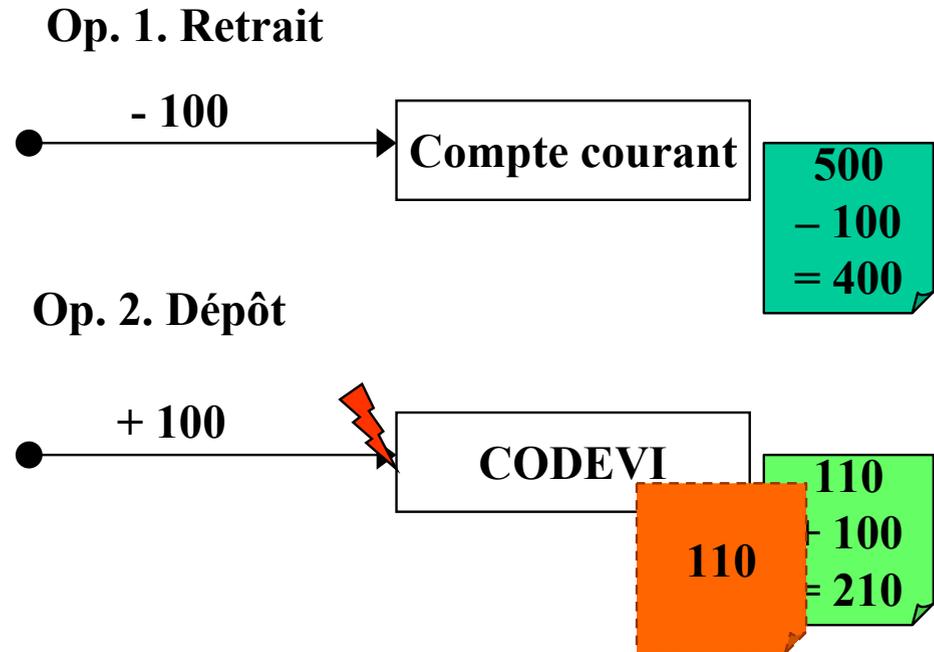
2 opérations atomiques

Que se passe-t-il si le
Dépôt échoue ?

Compte courant = 400

CODEVI = 110

Appelez la banque !!!



Exemple de transaction

Exemple de transaction

Virement bancaire **dans une transaction** (1/2)

Exemple de transaction

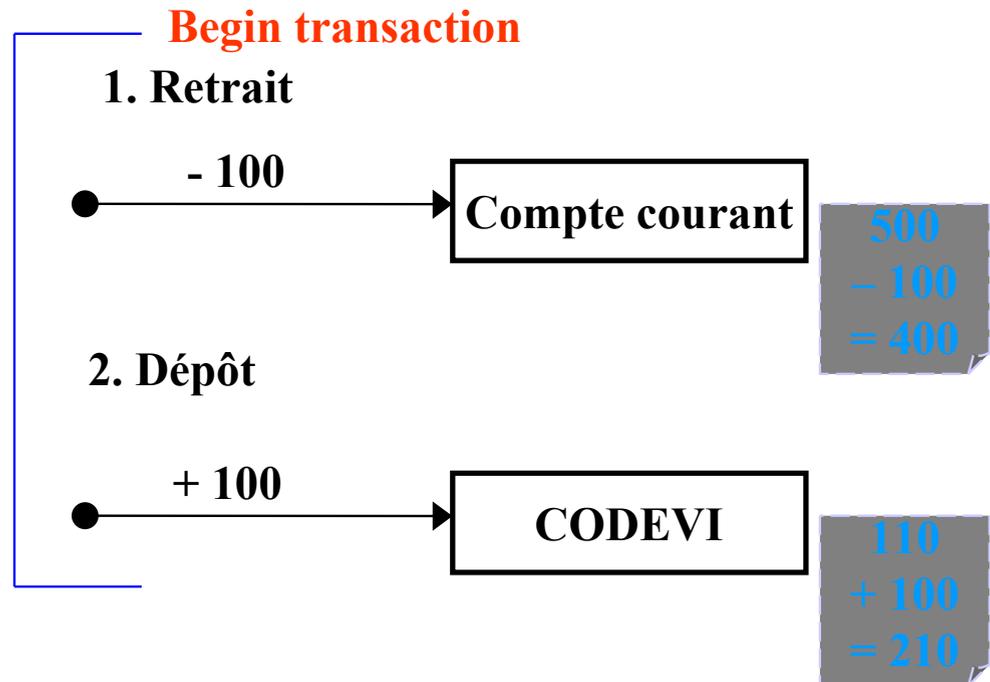
Virement bancaire dans une transaction (1/2)

**Virement = 1 transaction
de 2 opérations
atomiques**

Exemple de transaction

Virement bancaire dans une transaction (1/2)

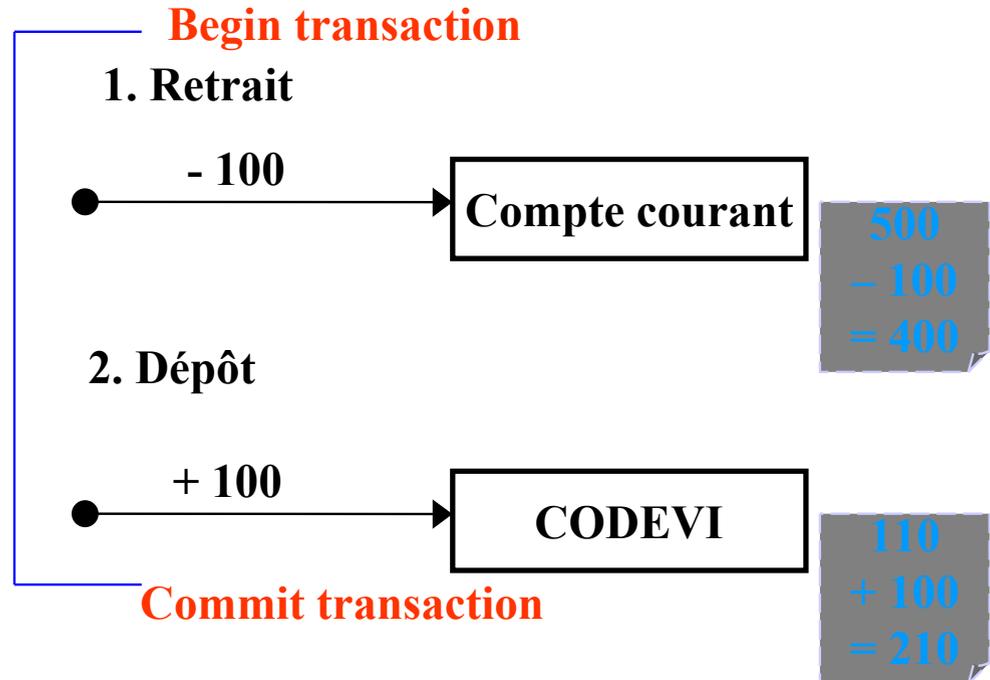
Virement = 1 transaction
de 2 opérations
atomiques



Exemple de transaction

Virement bancaire dans une transaction (1/2)

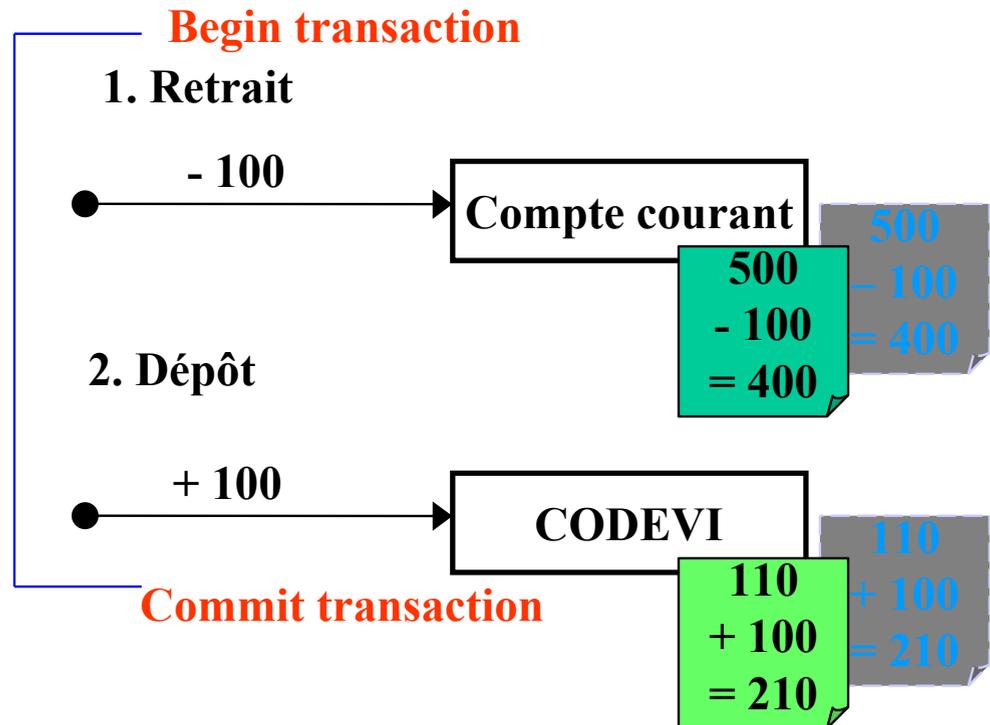
Virement = 1 transaction
de 2 opérations
atomiques



Exemple de transaction

Virement bancaire dans une transaction (1/2)

Virement = 1 transaction
de 2 opérations
atomiques



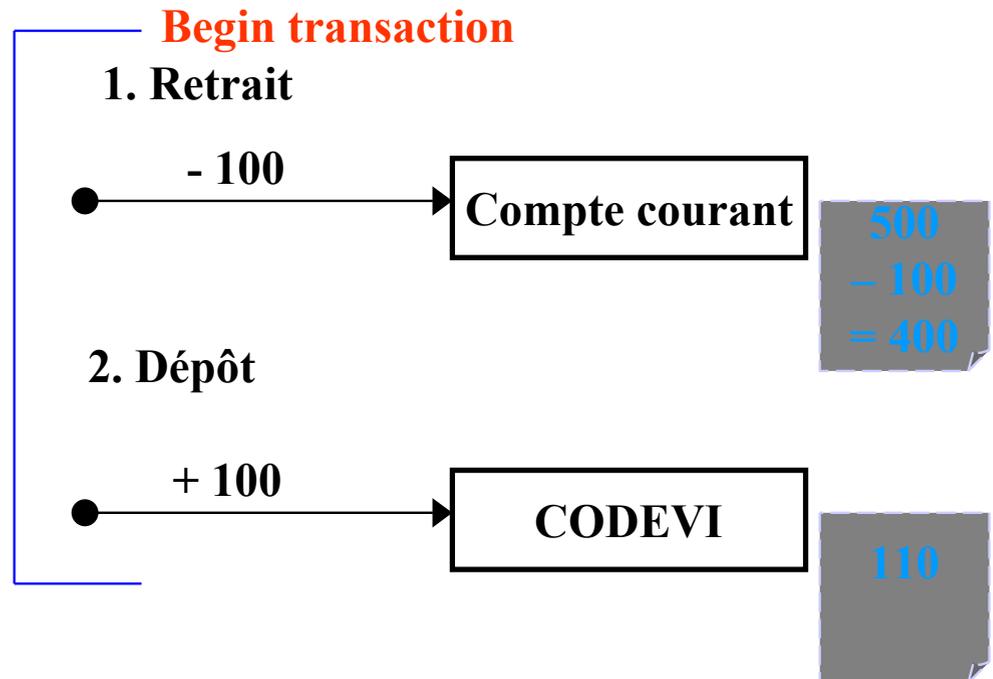
Exemple de transaction

Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

Exemple de transaction

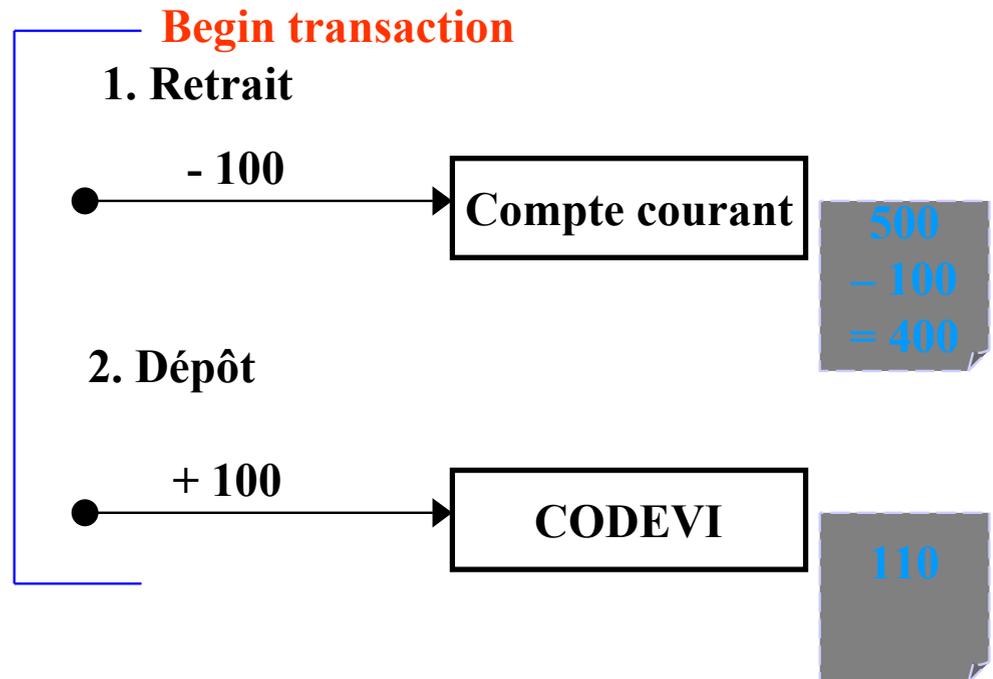
Virement bancaire **dans une transaction** (2/2)



Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

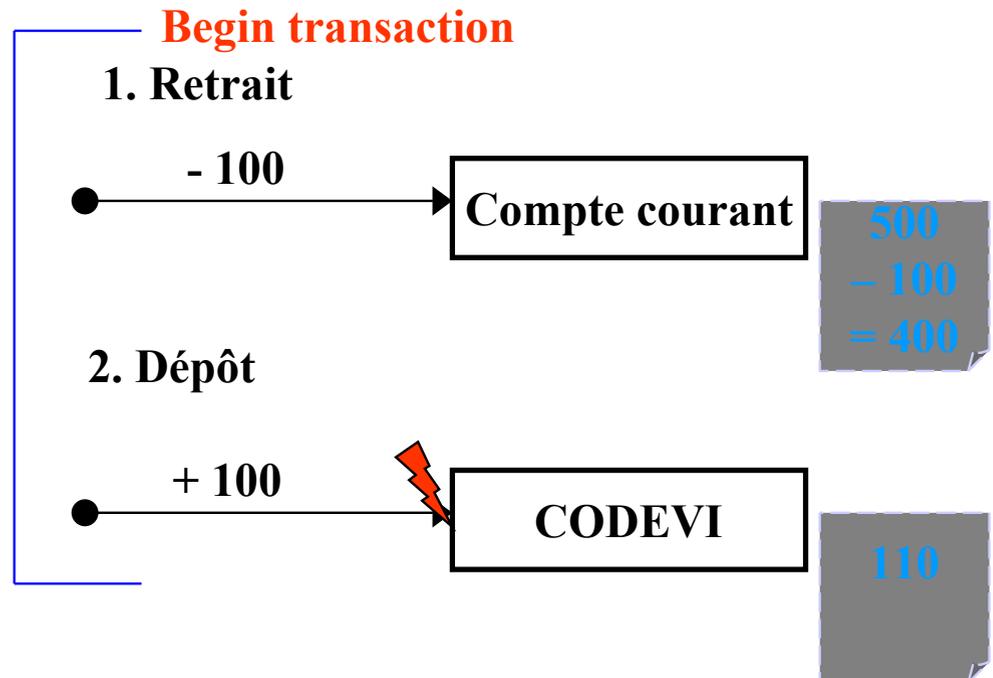
Que se passe-t-il si le
Dépôt échoue ?



Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

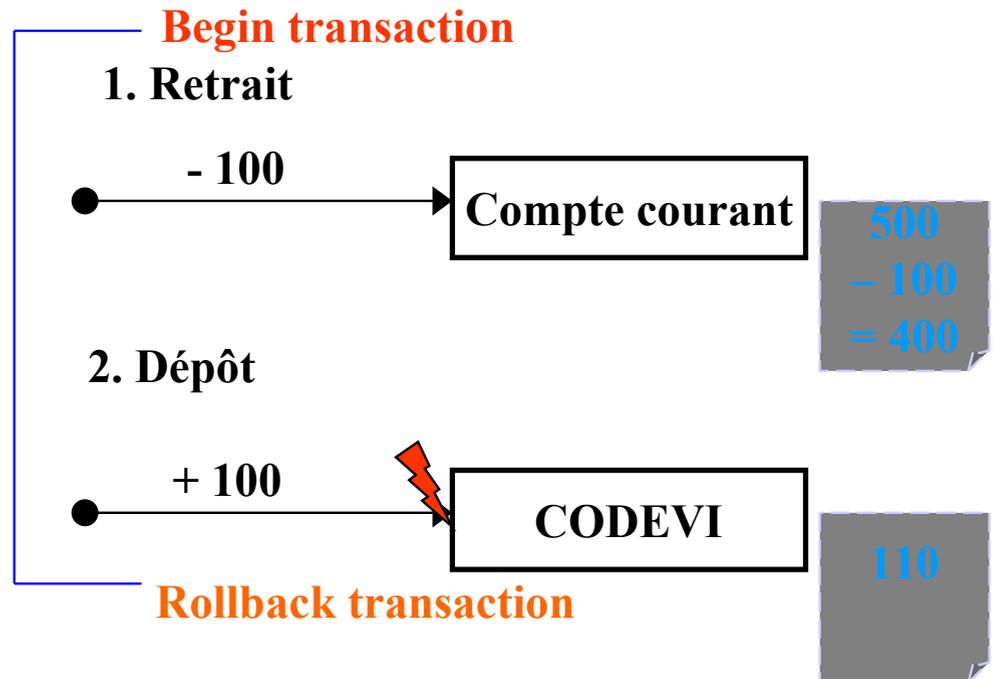
Que se passe-t-il si le
Dépôt échoue ?



Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

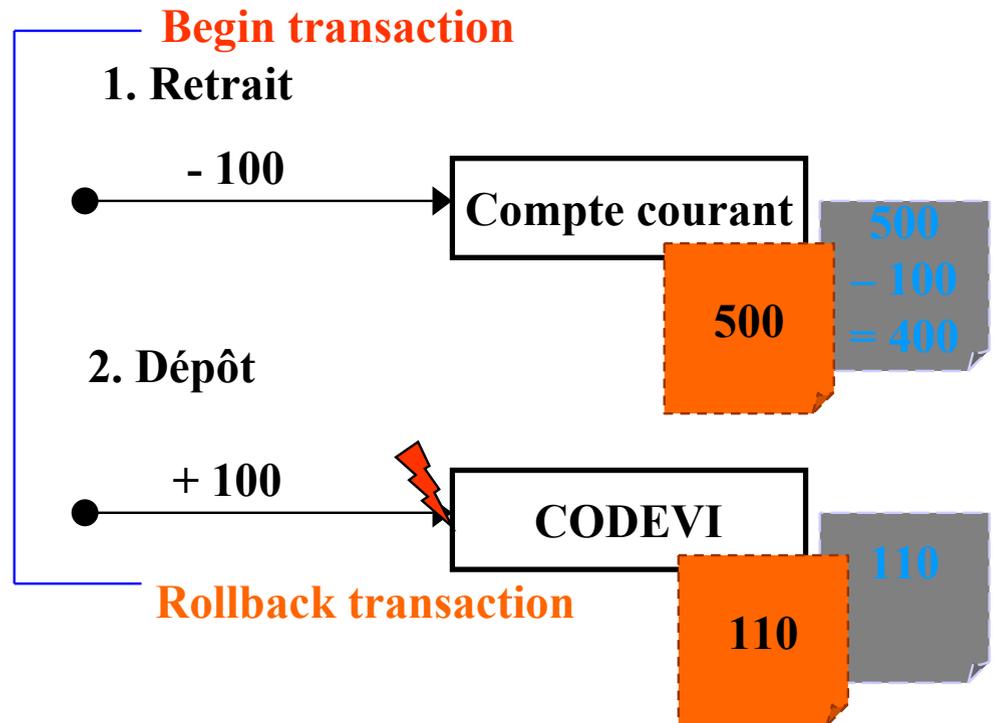
Que se passe-t-il si le
Dépôt échoue ?



Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

Que se passe-t-il si le
Dépôt échoue ?



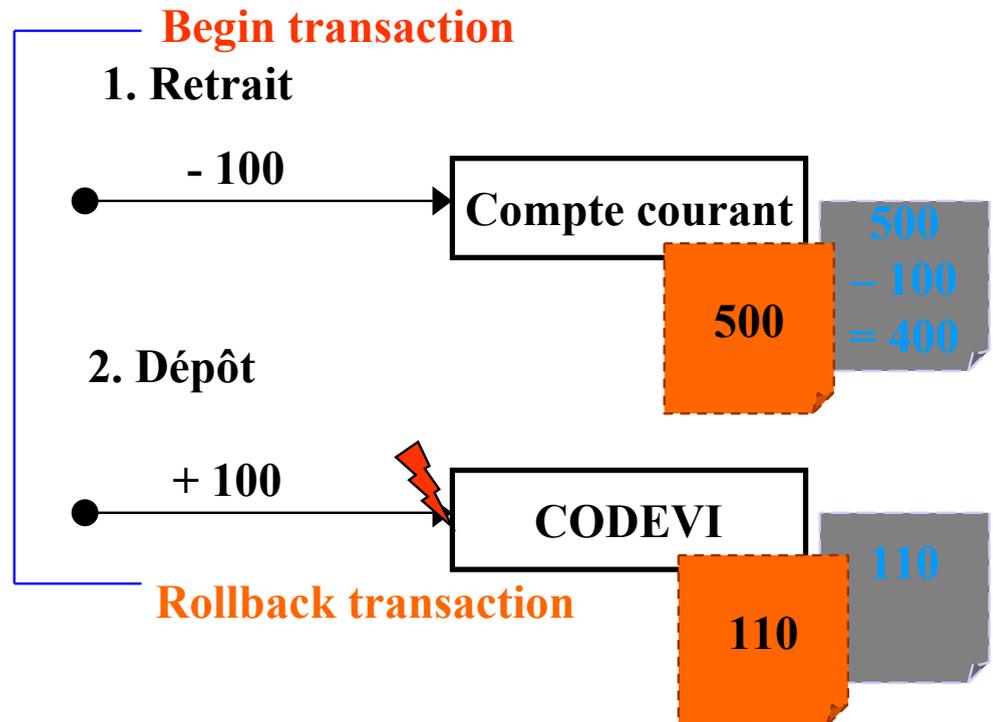
Exemple de transaction

Virement bancaire **dans une transaction** (2/2)

Que se passe-t-il si le
Dépôt échoue ?

Compte courant = 500
CODEVI = 110

Recommencez !



Degrés d'isolation sous SQL2

- **Degré 0** : Une transaction T ne modifie pas de **données salies** par d'autres transactions
- **Degré 1** : Degré 0 + T ne confirme pas ses changements avant la fin de la transaction
- **Degré 2** : Degré 1 + T ne lit pas de données salies par d'autres transactions
- **Degré 3** : Degré 2 + D'autres transactions ne salissent pas les données lues par T avant que T ne soit terminée

Architecture du système de transactions

Missions du système de transactions

Gérer les transactions, maintenir la cohérence, gérer les pannes

- **Gestionnaire de transactions :**

- ◆ Coordination des actions des différentes transactions
- ◆ En communication avec l'ordonnanceur

- **Ordonnanceur (*scheduler*):**

- ◆ Maintien de la cohérence
- ◆ Gestion des verrous (**gestionnaire de verrous**)

- **Gestionnaire de pannes (*recovery manager*)**

Remise de la base de données dans un état cohérent après panne

Ordonnement

- **Opération d'une transaction T**

- ◆ $R_T(i)$: lecture de l'item i par T
- ◆ $W_T(i)$: modification de la valeur de l'item i par T
- ◆ $Commit_T$: validation de T
- ◆ $Abort_T$: annulation de T

- **Ordonnement de transactions**

Liste d'actions de plusieurs transactions T_1, \dots, T_n telle que chaque opération de T_i apparaisse dans le même ordre dans T_i et dans l'ordonnement

- **Ordonnement séquentiel**

Pas d'entrelacement des actions des différentes transactions

Concurrency

- **Transactions concurrentes**

Deux transactions accédant en même temps aux mêmes items

- **Ordonnancement sérialisable**

- ◆ Résultat équivalent au résultat d'un ordonnancement séquentiel
- ◆ Les items voient passer toutes les transactions dans le même ordre

- **Anomalies dues à l'entrelacement des transactions**

- ◆ Pas de conflit si accès simultanés à un même item en lecture par deux transactions différentes
- ◆ Pas de conflit si accès simultanés à deux items différents en lecture ou écriture par deux transactions

Odonnancement sérialisable

T_1 : Solde A - x; Solde B + x;

T_2 : Solde A - y; Solde B + y;

T_3 : Solde A - z; Solde B + z;

Exécution séquentielle : $T_1 T_2 T_3$

Une exécution sérialisable équivalente à $T_1 T_2 T_3$:

Solde A - x;

Solde A - y;

Solde B + x;

Solde A - z;

Solde B + y;

Solde B + z;

L'item A voit passer les transaction
dans l'ordre $T_1 T_2 T_3$

L'item B voit passer les transaction
dans l'ordre $T_1 T_2 T_3$

Conflicts

- **Ecriture - Lecture :**

- ◆ **Lecture impropre ou parasite (*dirty read*)**
- ◆ **Placement momentanée de la base dans un état incohérent**
- ◆ **Annulation en cascade de transactions**

- **Lecture - Ecriture :**

- ◆ **Lecture non reproductible (*unrepeatable read*)**
- ◆ **Incohérence**

- **Ecriture- Ecriture :**

Perte de mises à jour (*blind write*)

Degrés d'isolation et conflits

ANSI SQL92 définit 3 types d'anomalies d'isolation

- **Lectures sales ou impropres**

Une transaction T1 lit des modifications non validées d'items effectuées par T2.

En cas de annulation de T2, T1 a lu des valeurs invalides

- **Lecture non reproductibles**

T1 lit un item, T2 modifie ce même item, T1 relit ce item et obtient une valeur différente

- **Lectures fantômes**

T1 lit un ensemble de nuplets, T2 ajoute/supprime des nuplets, T1 relit l'ensemble de nuplets et obtient un ensemble différent comme résultat

Degrés d'isolation et conflits

- **Degré 0**

Résolution des pertes de mises à jour

- **Degré 1**

Pas d'annulation en cascade + Degré 0

- **Degré 2**

Pas de lecture impropre + Degré 1

- **Degré 3**

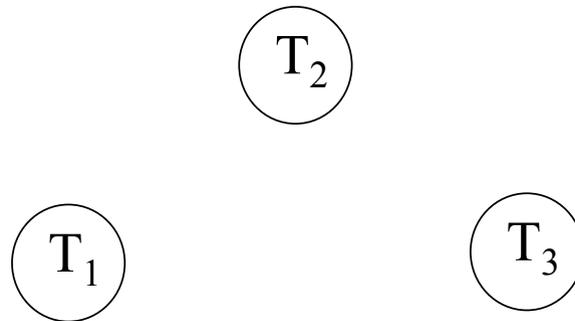
Isolation totale - Mise en attente des transactions en conflit

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A), R_1(B), W_2(A), R_3(A), W_1(B), W_3(A), R_2(B), W_2(B)$



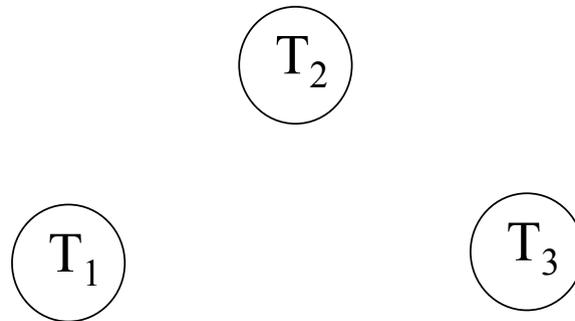
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A), R_1(B), W_2(A), \underline{R_3(A)}, W_1(B), W_3(A), R_2(B), W_2(B)$



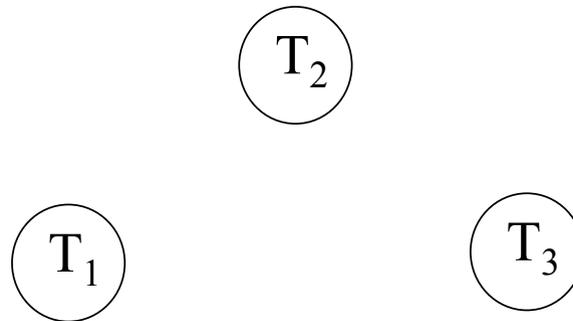
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A)$, $R_1(B)$, $W_2(A)$, $R_3(A)$, $W_1(B)$, $W_3(A)$, $R_2(B)$, $W_2(B)$



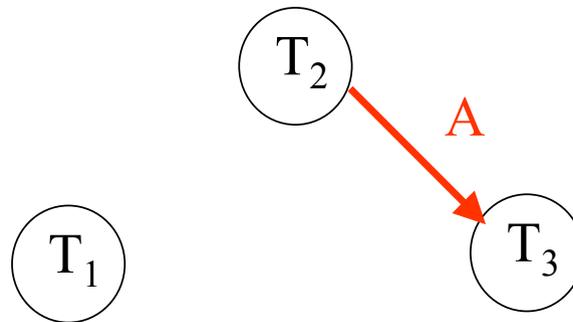
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A)$, $R_1(B)$, $W_2(A)$, $R_3(A)$, $W_1(B)$, $W_3(A)$, $R_2(B)$, $W_2(B)$



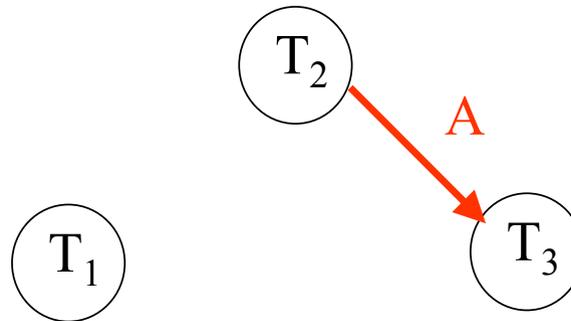
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A)$, $R_1(B)$, $W_2(A)$, $R_3(A)$, $W_1(B)$, $W_3(A)$, $R_2(B)$, $W_2(B)$



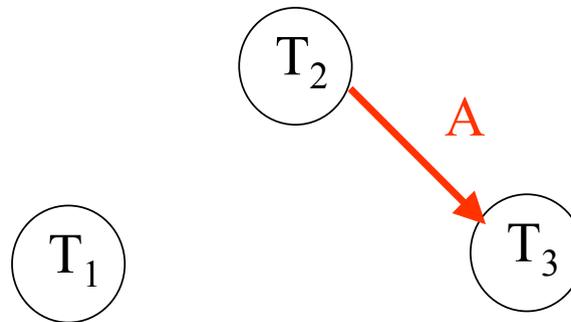
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A)$, $R_1(B)$, $W_2(A)$, $R_3(A)$, $W_1(B)$, $W_3(A)$, $R_2(B)$, $W_2(B)$



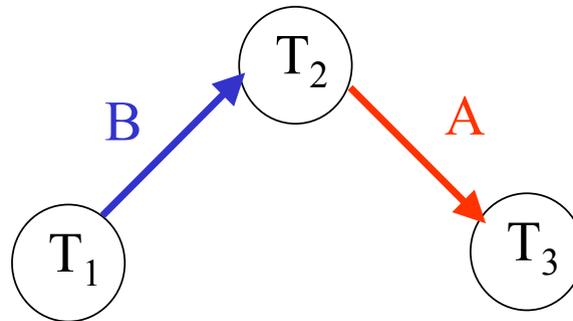
Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Graphe de précédence

Pour trouver les conflits potentiels

- Chaque nœud représente une transaction
- Un arc de T_i vers T_j signifie qu'une action de T_i précède et entre en conflit avec une ou plusieurs actions de T_j

$R_2(A)$, $R_1(B)$, $W_2(A)$, $R_3(A)$, $W_1(B)$, $W_3(A)$, $R_2(B)$, $W_2(B)$



Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable

Verrouillage des données

- **Gestion des verrous**

- ◆ $V_T(i)$: Verrouillage de l'item i par la transaction T

- Verrou partagé $VP_T(i)$ ou exclusif $VX_T(i)$

- ◆ $D_T(i)$: Déverrouillage de l'item i par T

- **Table des verrous : pour chaque item verrouillé**

- ◆ Mode de verrouillage

- ◆ Indicateur de transactions en attente

- ◆ Liste des transactions détenant un verrou ou en attente d'un verrou

- Nom de la transaction

- Mode de verrouillage obtenu ou souhaité

- Indicateur d'attente

- Lien vers les autres items verrouillés par la transaction

Architecture du gestionnaire de verrous

- Réception de la requête par le 1^{er} module
- Transmission des transactions au 2^{ème} module après insertion des verrous
- Détection des demandes de verrous par le 2^{ème} module
 - ◆ Vérification dans la table des verrous
 - ◆ Si demande acceptée, exécution de l'action sur la BD
 - ◆ Si demande rejetée, mise en attente de la transaction
- Après validation ou annulation d'une transaction, libération des verrous par le 1^{er} module informé par le 2^{ème}
- Après libération d'un verrou, transmission du verrou à une transaction en attente par le 2^{ème} module

Inter-blocage

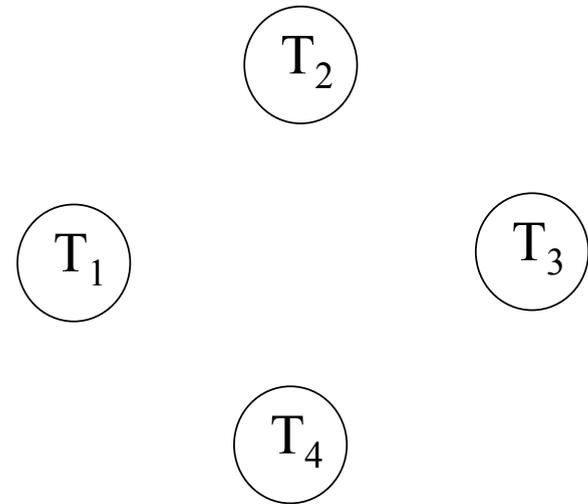
- **Inter-blocage (*Deadlock*)** : Attente mutuelle de deux transactions
- **Détection par un graphe d'attente**
 - ◆ Chaque nœud représente une transaction en cours d'exécution
 - ◆ Un arc de T_i vers T_j signifie que T_i **attend** un verrou détenu par T_j sur un même item
 - ◆ Ajout d'un arc par le gestionnaire de verrous à chaque demande insatisfaite et inversement
- **Prévention par estampillage**
 - ◆ Association d'une estampille à chaque transaction au début de l'exécution
Plus la transaction est ancienne, plus la priorité est grande
 - ◆ **Wait-Die** : Si T_i a une priorité plus forte que T_j alors T_i attend, sinon T_i est annulée
 - ◆ **Wound-Wait** : Si T_i a une priorité plus forte que T_j alors T_j est annulée, sinon T_i attend

Graphe d'attente



$VP_1(A), R_1(A), VX_2(B), W_2(B), VP_1(B), VP_3(C), R_3(C), VX_2(C), VX_4(A), VX_3(A)$

Table des verrous



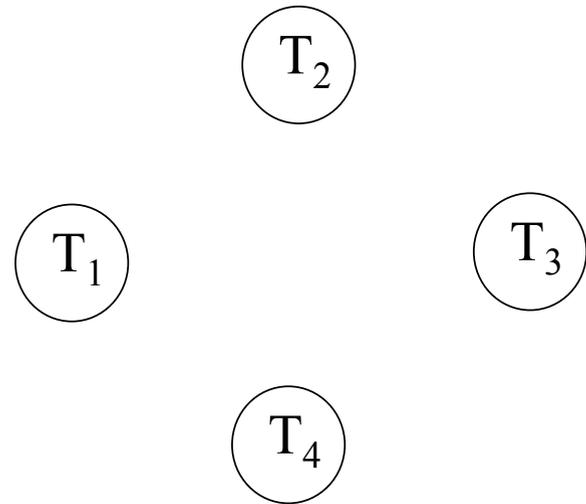
Graphe d'attente



$VP_1(A), R_1(A), VX_2(B), W_2(B), VP_1(B), VP_3(C), R_3(C), VX_2(C), VX_4(A), VX_3(A)$

Table des verrous

A	VP (T_1)	
---	--------------	--



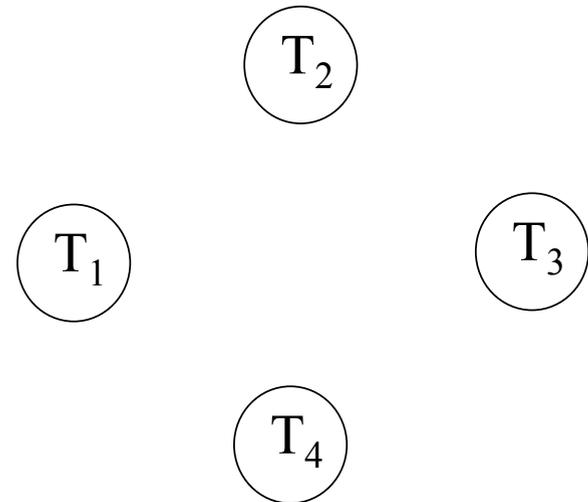
Graphe d'attente



$VP_1(A), R_1(A), VX_2(B), W_2(B), VP_1(B), VP_3(C), R_3(C), VX_2(C), VX_4(A), VX_3(A)$

Table des verrous

A	VP (T_1)	
B	VX (T_2)	



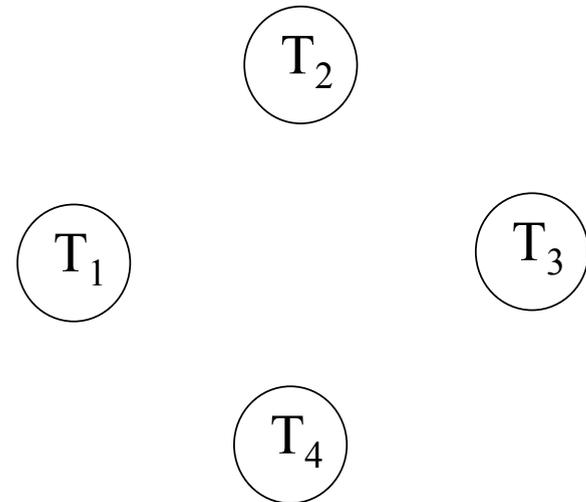
Graphe d'attente



$VP_1(A), R_1(A), VX_2(B), W_2(B), \underline{VP_1(B)}, VP_3(C), R_3(C), VX_2(C), VX_4(A), VX_3(A)$

Table des verrous

A	VP (T_1)	
B	VX (T_2)	



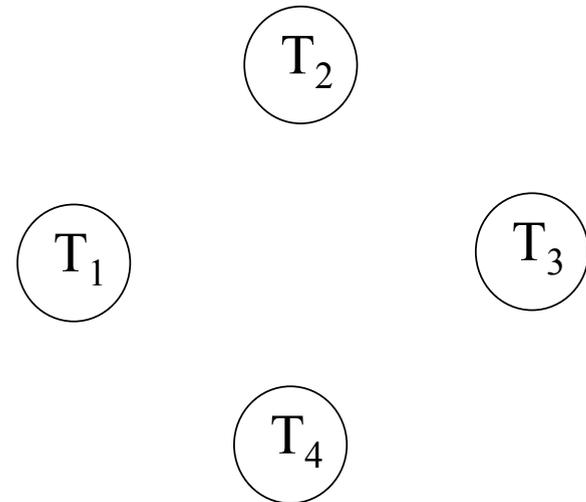
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	



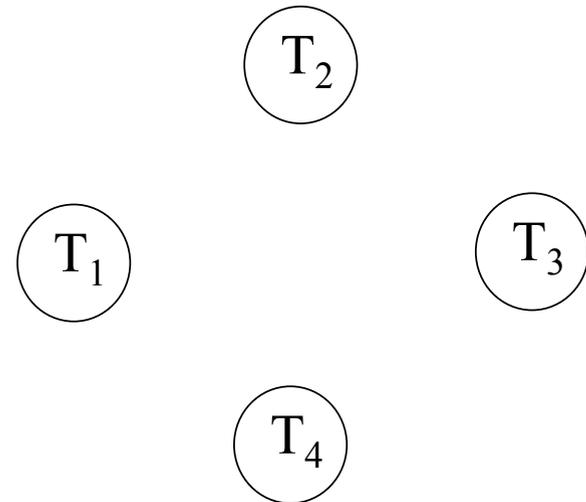
Graphe d'attente



$VP_1(A)$, $R_1(A)$, $VX_2(B)$, $W_2(B)$, $VP_1(B)$, $VP_3(C)$, $R_3(C)$, $VX_2(C)$, $VX_4(A)$, $VX_3(A)$

Table des verrous

A	VP (T_1)	
B	VX (T_2)	T_1 (VP)



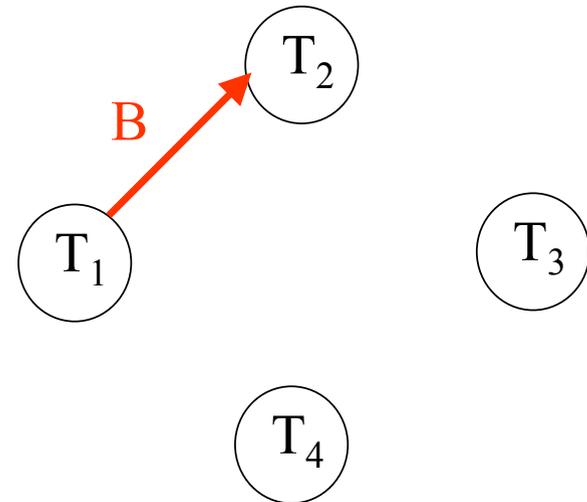
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)



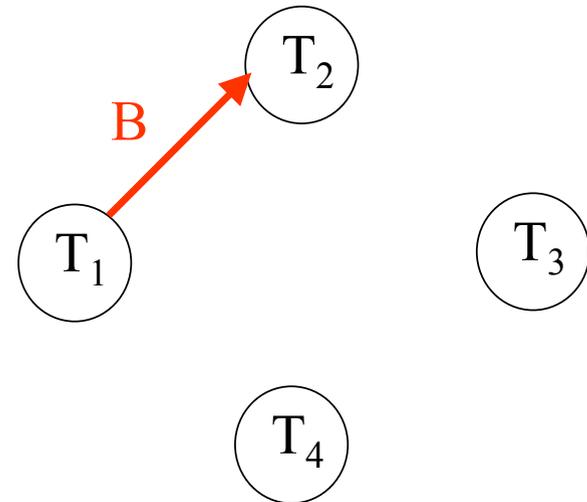
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	



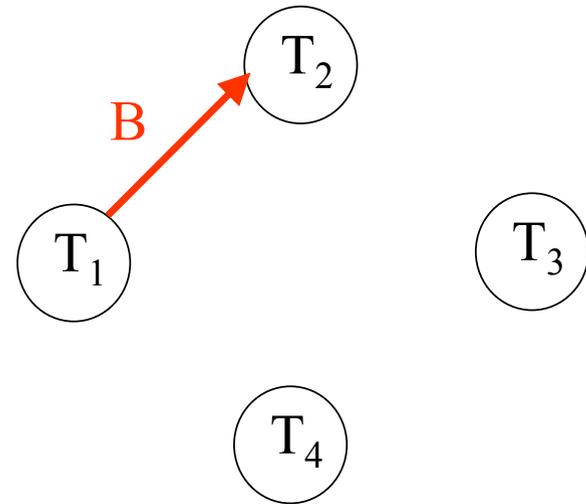
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	



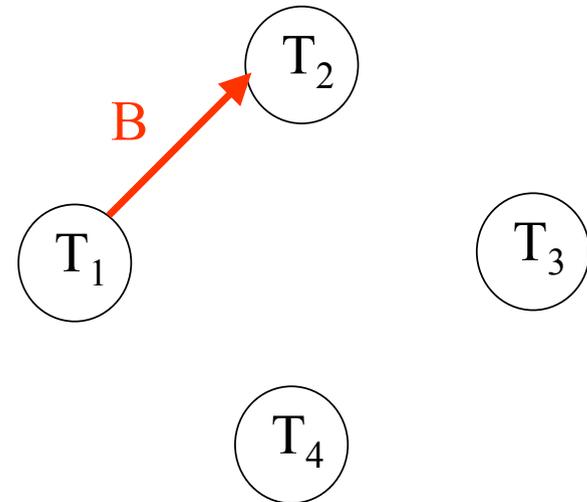
Graphe d'attente



$VP_1(A)$, $R_1(A)$, $VX_2(B)$, $W_2(B)$, $VP_1(B)$, $VP_3(C)$, $R_3(C)$, $VX_2(C)$, $VX_4(A)$, $VX_3(A)$

Table des verrous

A	VP (T_1)	
B	VX (T_2)	T_1 (VP)
C	VP (T_3)	



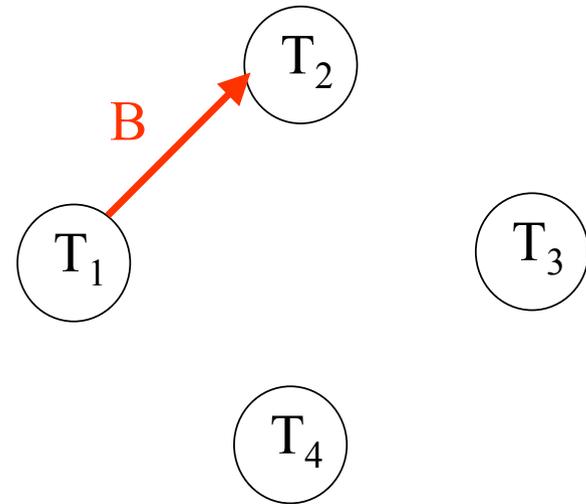
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



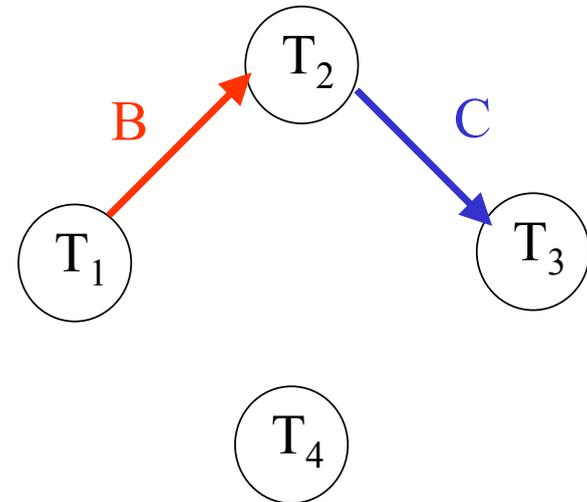
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



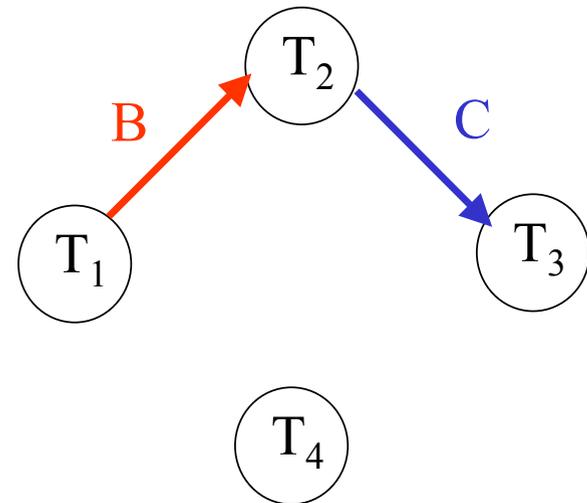
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



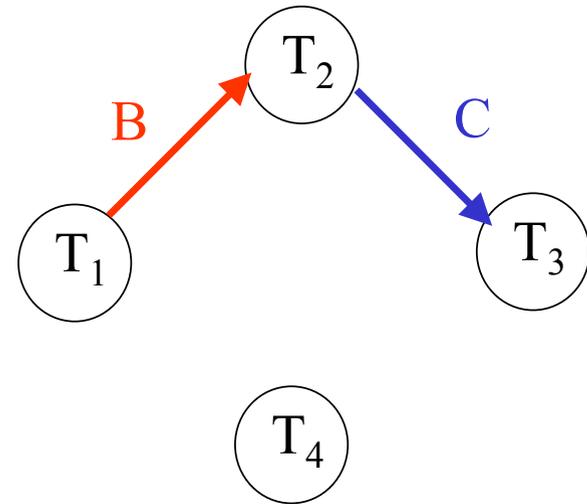
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



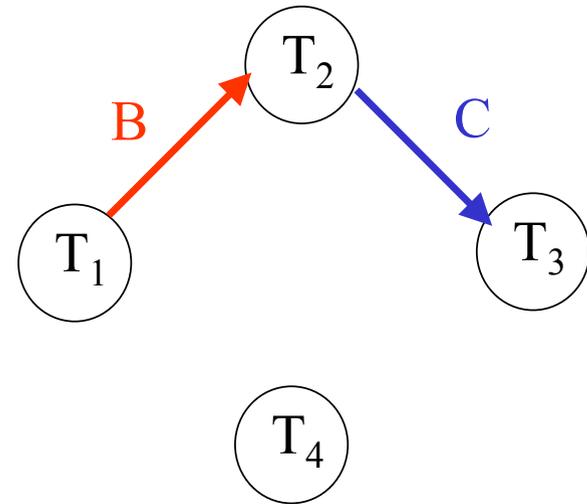
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



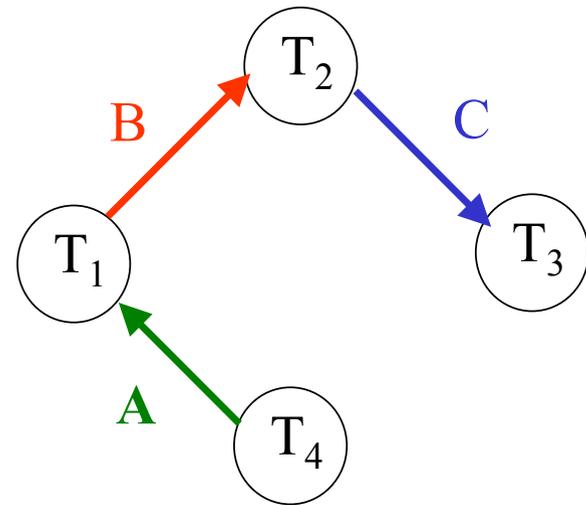
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



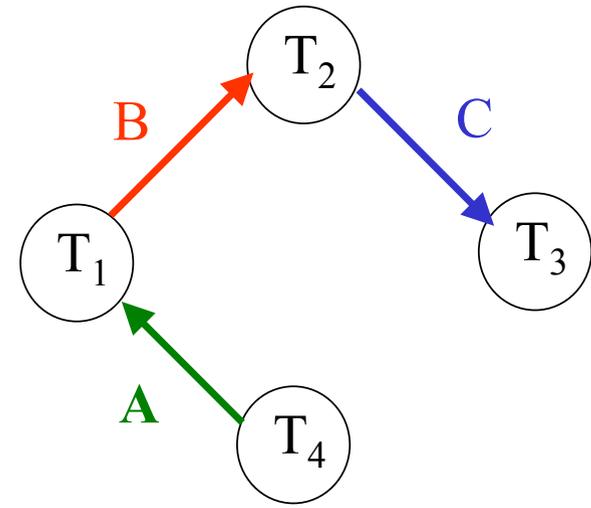
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



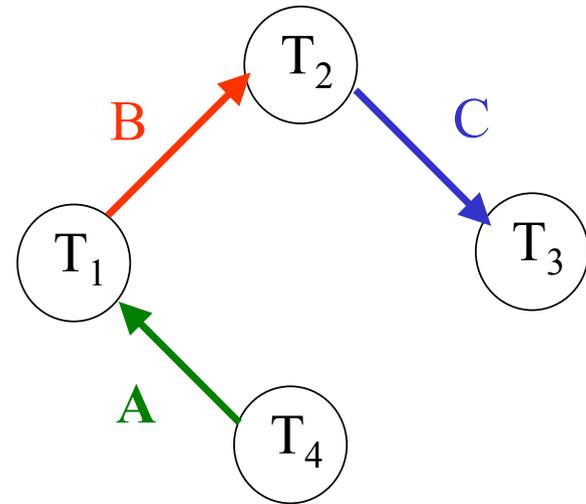
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)
B	VX (T ₂)	T ₁ (VP)
C	VP (T ₃)	T ₂ (VX)



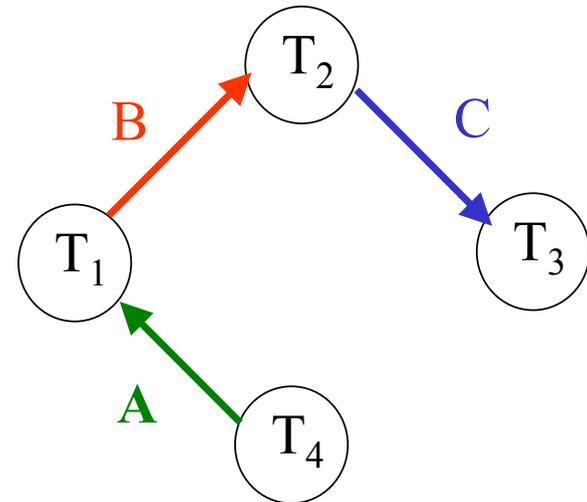
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)	T ₃ (VX)
B	VX (T ₂)	T ₁ (VP)	
C	VP (T ₃)	T ₂ (VX)	



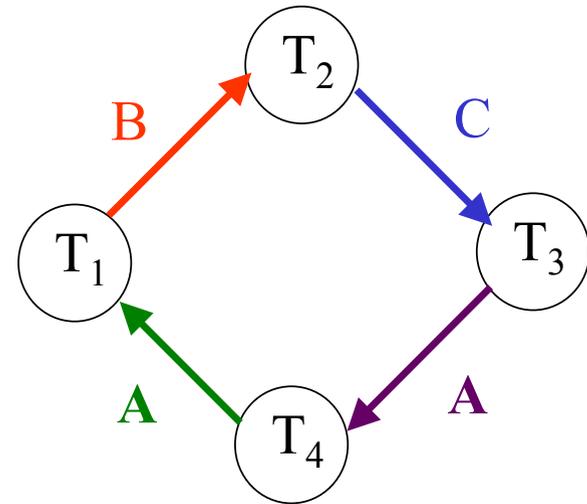
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)	T ₃ (VX)
B	VX (T ₂)	T ₁ (VP)	
C	VP (T ₃)	T ₂ (VX)	



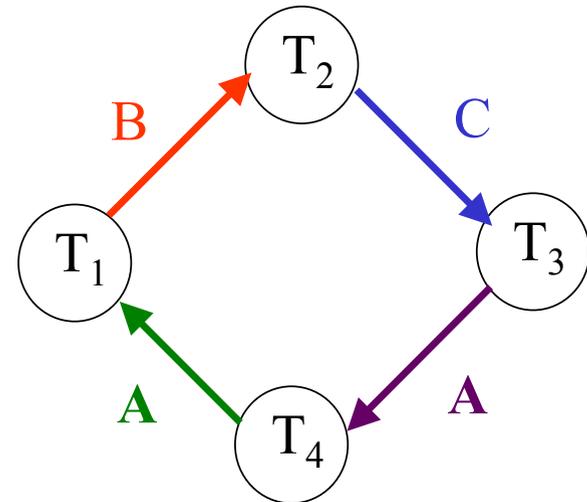
Graphe d'attente



VP₁(A), R₁(A), VX₂(B), W₂(B), VP₁(B), VP₃(C), R₃(C), VX₂(C), VX₄(A), VX₃(A)

Table des verrous

A	VP (T ₁)	T ₄ (VX)	T ₃ (VX)
B	VX (T ₂)	T ₁ (VP)	
C	VP (T ₃)	T ₂ (VX)	



Cycle dans le graphe d'attente ⇒
 détection d'un inter-blocage ⇒
 Annulation de T₃

Protocole de verrouillage en deux phases

(Two-Phases Locking)

- **Principe**

- ◆ Phase 1 : Verrouillage des items / Phase ascendante
- ◆ Phase 2 : Déverrouillage des items / Phase descendante

- **Théorème**

Un ordonnancement obtenu par le protocole V2P est sérialisable

- **Inconvénients**

Risque d'annulation de transactions en cascade et d'inter-blocages

- **Protocole V2P strict**

Les verrous sont conservés par une transaction jusqu'à la fin

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précedence :

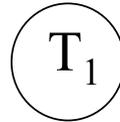
Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x), R_2(x), W_1(x), W_2(x)$

Graphe de précédence :



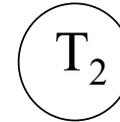
Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x), R_2(x), W_1(x), W_2(x)$

Graphe de précédence :



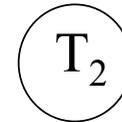
Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :

Protocole de verrouillage en deux phases

Exemple

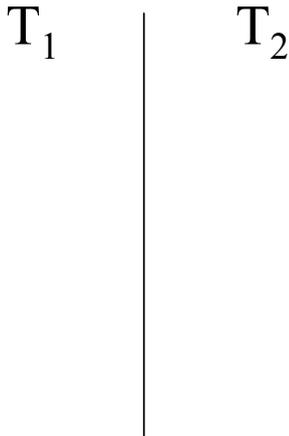
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

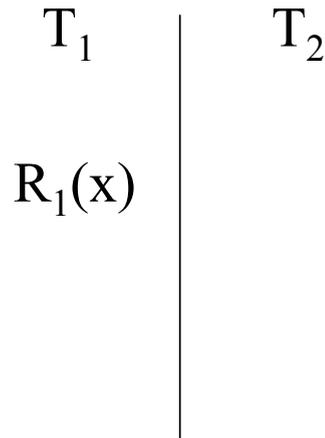
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

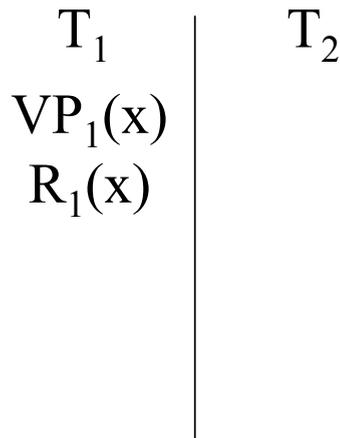
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

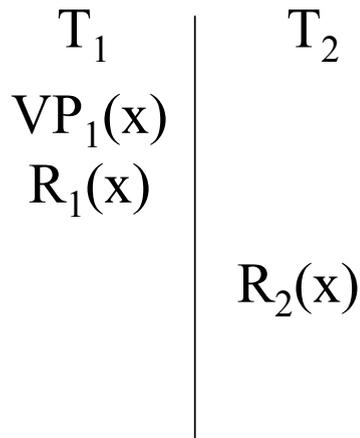
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

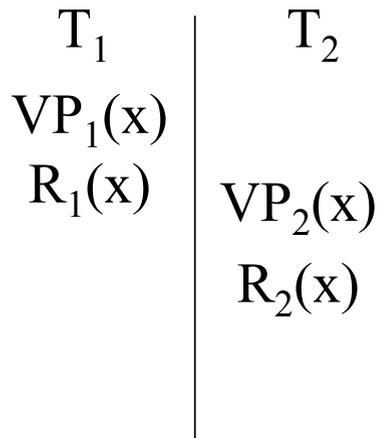
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

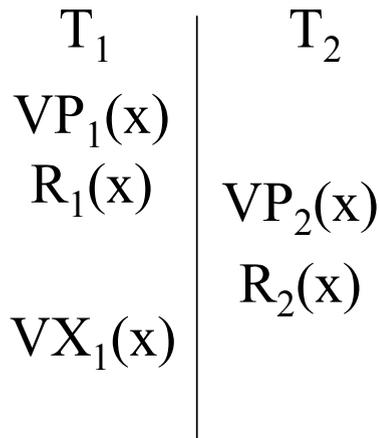
Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :

T_1		T_2
$VP_1(x)$		
$R_1(x)$		$VP_2(x)$
		$R_2(x)$
$VX_1(x)$		
$W_1(x)$		

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence \Rightarrow ordonnancement non sérialisable

Protocole V2P :

T_1	T_2
$VP_1(x)$	
$R_1(x)$	$VP_2(x)$
$VX_1(x)$	$R_2(x)$
$W_1(x)$	

Pour durcir le verrou de T_1 , il faudrait que T_2 n'ait pas de verrou sur x

\Rightarrow V2P non applicable

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x), R_2(y), R_1(y), R_2(x), C_1, C_2$

Graphe de précedence :

Protocole V2P strict :

Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x), R_2(y), R_1(y), R_2(x), C_1, C_2$

Graphe de précédence :



Protocole V2P strict :

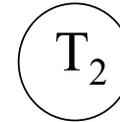
Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x), R_2(y), R_1(y), R_2(x), C_1, C_2$

Graphe de précédence :



Protocole V2P strict :

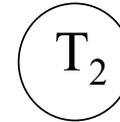
Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Protocole V2P strict :

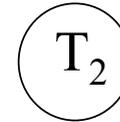
Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $R_1(y)$, $\underline{R_2(x)}$, C_1 , C_2

Graphe de précédence :



Protocole V2P strict :

Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Protocole V2P strict :

Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

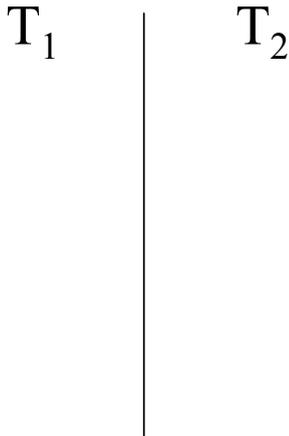
Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

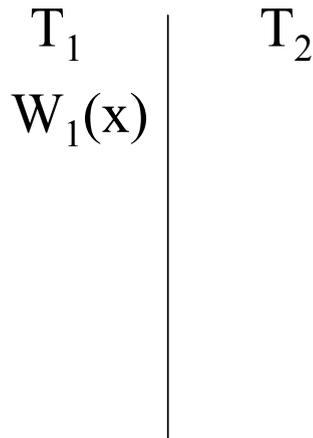
Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

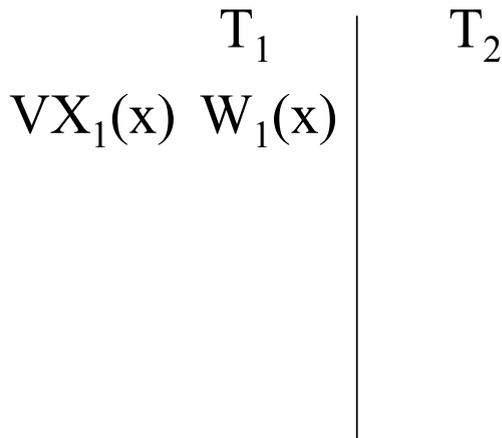
Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

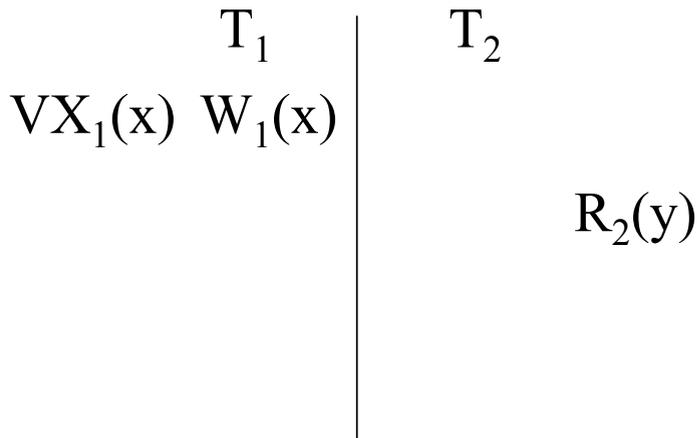
Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

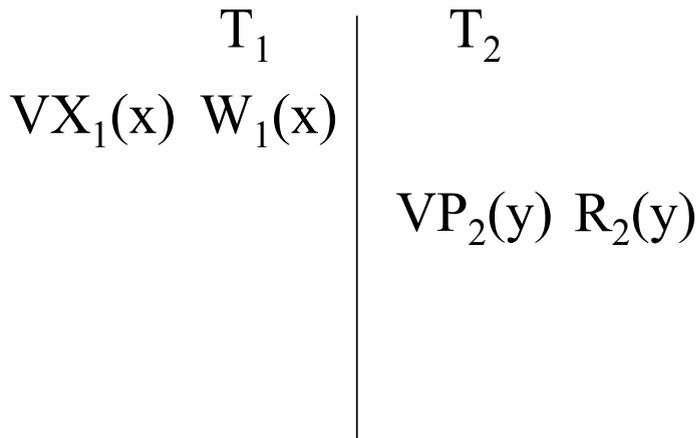
Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T_1	T_2
$VX_1(x) \ W_1(x)$	
	$VP_2(y) \ R_2(y)$
$R_1(y)$	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T_1	T_2
$VX_1(x) \ W_1(x)$	
	$VP_2(y) \ R_2(y)$
$VP_1(y) \ R_1(y)$	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP ₂ (x) R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T_1	T_2
$VX_1(x)$	$W_1(x)$	
		$VP_2(y)$ $R_2(y)$
$VP_1(y)$	$R_1(y)$	
		$VP_2(x)$ $R_2(x)$

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T_1	T_2
$VX_1(x) \ W_1(x)$	
	$VP_2(y) \ R_2(y)$
$VP_1(y) \ R_1(y)$	
	$VP_2(x) \ R_2(x)$

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_1(y)$, $R_2(x)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
	W ₁ (x)	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		R ₂ (y)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
VP ₁ (y) R ₁ (y)	
	VP₂(x) R₂(x)

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
R ₁ (y)	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
VP ₁ (y) R ₁ (y)	
	VP₂(x) R₂(x)

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
VP ₁ (y) R ₁ (y)	

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
VP ₁ (y) R ₁ (y)	
	VP₂(x) R₂(x)

T ₁	T ₂
VX ₁ (x) W ₁ (x)	
	VP ₂ (y) R ₂ (y)
VP ₁ (y) R ₁ (y)	
	R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP ₂ (x) R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
	D ₁ (x)	
		VP ₂ (x) R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : W₁(x), R₂(y), R₁(y), R₂(x), C₁, C₂

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
		VP₂(x) R₂(x)

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
VP ₁ (y)	R ₁ (y)	
	D ₁ (x)	
		VP ₂ (x) R ₂ (x)

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précedence :

Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x), R_2(y), R_2(x), R_1(y), C_1, C_2$

Graphe de précédence :



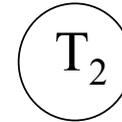
Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



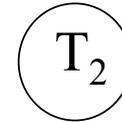
Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



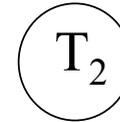
Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :

Protocole de verrouillage en deux phases

Exemple

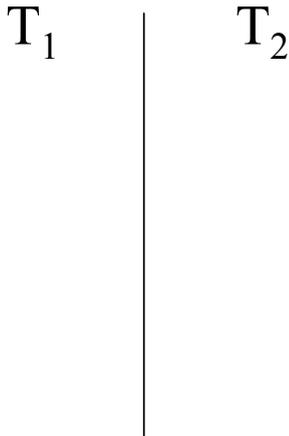
Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

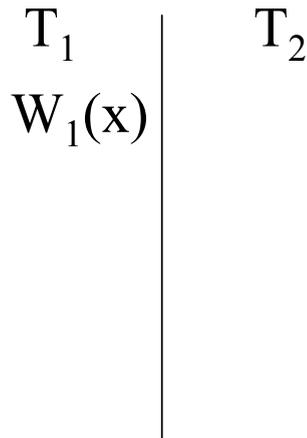
Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

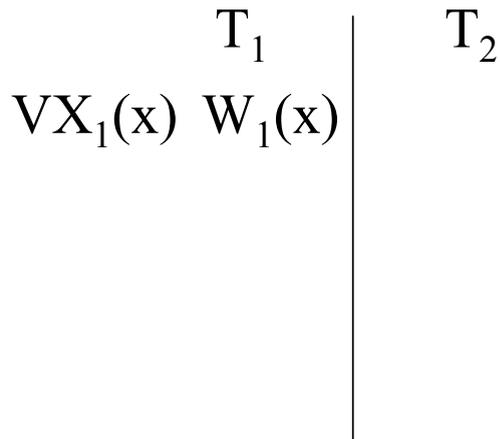
Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

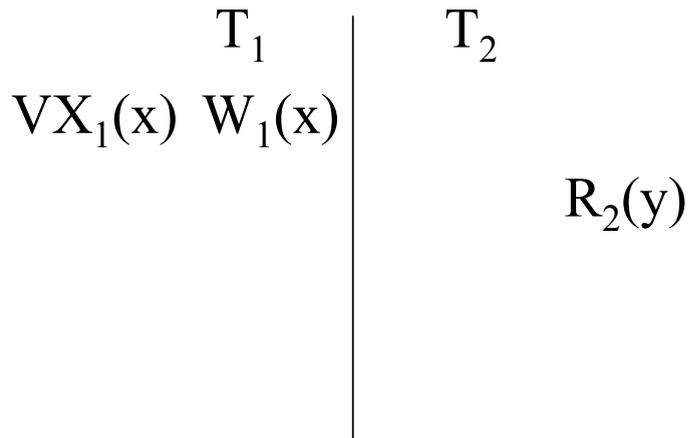
Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

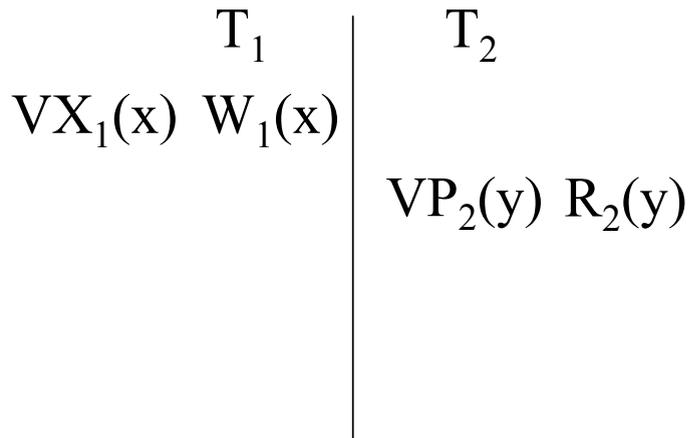
Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

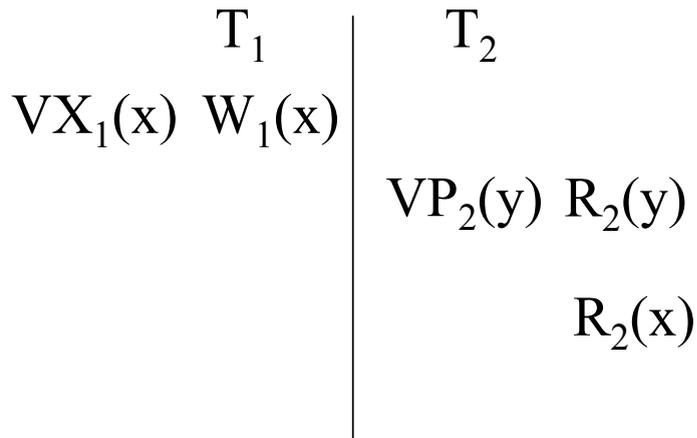
Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

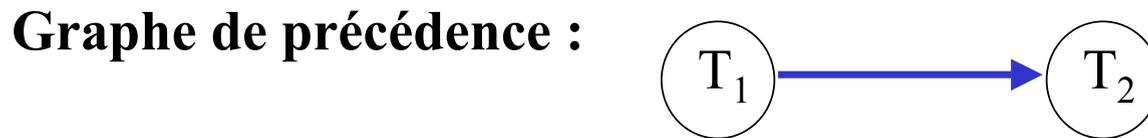
Protocole V2P strict ou non strict :

T_1		T_2
$VX_1(x) \ W_1(x)$		
		$VP_2(y) \ R_2(y)$
		$VP_2(x) \ R_2(x)$

Protocole de verrouillage en deux phases

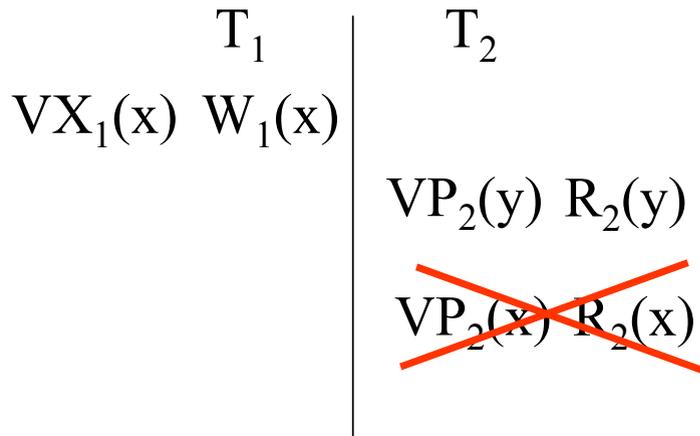
Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Protocole de verrouillage en deux phases

Exemple

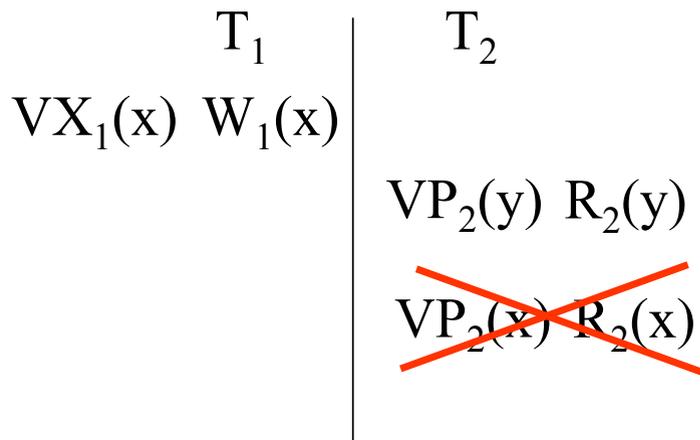
Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Opération impossible car pour que $VP_2(x)$ il faudrait $D_1(x)$ or T_1 est en phase ascendante du protocole V2P

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :

T_1	T_2
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
	$VP_2(x)$ $R_2(x)$
$R_1(y)$	

Opération impossible car pour que $VP_2(x)$ il faudrait $D_1(x)$ or T_1 est en phase ascendante du protocole V2P

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :

T_1	T_2
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
	$VP_2(x)$ $R_2(x)$
$VP_1(y)$ $R_1(y)$	

Opération impossible car pour que $VP_2(x)$ il faudrait $D_1(x)$ or T_1 est en phase ascendante du protocole V2P

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $\underline{W_1(x)}$, $R_2(y)$, $\underline{R_2(x)}$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :

T_1	T_2
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
	$VP_2(x)$ $R_2(x)$
$VP_1(y)$ $R_1(y)$	

Opération impossible car pour que $VP_2(x)$ il faudrait $D_1(x)$ or T_1 est en phase ascendante du protocole V2P \Rightarrow application V2P impossible

Protocole de verrouillage en deux phases

Exemple

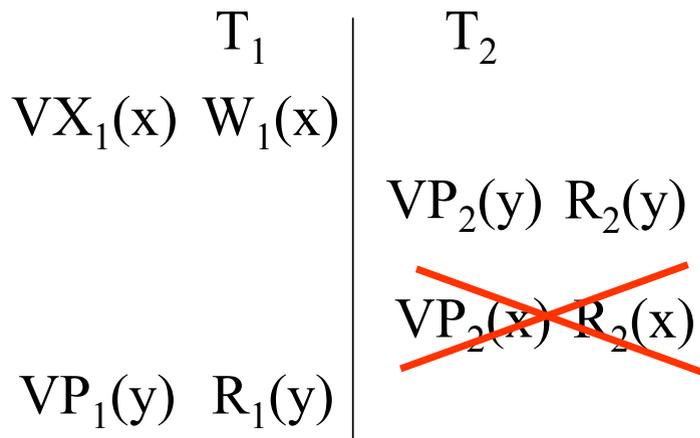
Soit l'ordonnancement : $W_1(x), R_2(y), R_2(x), R_1(y), C_1, C_2$

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :



Opération impossible car pour que $VP_2(x)$ il faudrait $D_1(x)$ or T_1 est en phase ascendante du protocole V2P \Rightarrow application V2P impossible

V2P \Rightarrow sérialisable

Protocole de verrouillage en deux phases

Exemple

Soit l'ordonnancement : $W_1(x)$, $R_2(y)$, $R_2(x)$, $R_1(y)$, C_1 , C_2

Graphe de précédence :



Pas de cycle dans le graphe de précédence \Rightarrow ordonnancement sérialisable

Protocole V2P strict ou non strict :

	T ₁	T ₂
VX ₁ (x)	W ₁ (x)	
		VP ₂ (y) R ₂ (y)
		VP₂(x) R₂(x)
VP ₁ (y)	R ₁ (y)	

Opération impossible car pour que VP₂(x) il faudrait D₁(x) or T₁ est en phase ascendante du protocole V2P \Rightarrow application V2P impossible

V2P \Rightarrow sérialisable

V2P ~~\Rightarrow sérialisable~~

Ordonnement par estampillage

- Association d'une estampille $TS(T)$ à chaque transaction T
- Association de deux estampilles à chaque item $RTS(A)$ et $WTS(A)$
- Si T veut lire l'item x 
 - ◆ Si $TS(T) < WTS(x)$
alors T est annulée et relancée avec une nouvelle estampille
 - ◆ Sinon, $RTS(x) = \text{Max}[TS(T), RTS(x)]$
- Si T veut écrire sur l'item x 
 - ◆ Si $TS(T) < RTS(x)$
alors T est annulée et relancée avec une nouvelle estampille
 - ◆ Si $TS(T) < WTS(x)$, alors l'action de T est ignorée (*règle de Thomas*) 
 - ◆ Sinon, $WTS(x) = TS(T)$

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	RTS ^x	WTS	RTS ^y	WTS
100	125	0	0	0	0

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 100	T_j 125	RTS ^x 0	WTS ^x 0	RTS ^y 0	WTS ^y 0
$W_i(x)$					

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
100	125	RTS	WTS	RTS	WTS
		0	0	0	0
$W_i(x)$			100		

$T_i +$ ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
100	125	0	0	0	0
$W_i(x)$	$R_j(y)$		100		

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
100	125	0	0	0	0
$W_i(x)$			100		
	$R_j(y)$			125	

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
100	125	RTS	WTS	RTS	WTS
		0	0	0	0
$W_i(x)$			100		
	$R_j(y)$			125	
$R_i(y)$					

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 100	T_j 125	x		y		$T_i +$ ancienne que T_j
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$						← $TS(T_i) > WTS(y)$ et la dernière transaction la plus récente ayant lu y est toujours T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y		
RTS	RTS	WTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_i) > WTS(y)$ et la dernière transaction la plus récente ayant lu y est toujours T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 100	T_j 125	x		y		$T_i +$ ancienne que T_j
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_i) > WTS(y)$ et la dernière transaction la plus récente ayant lu y est toujours T_j
	$R_j(x)$					

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 100	T_j 125	x		y		$T_i +$ ancienne que T_j
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_j) > WTS(y)$ et la dernière transaction la plus récente ayant lu y est toujours T_j
	$R_j(x)$					← $TS(T_j) > WTS(x) \Rightarrow$ la dernière transaction ayant mis à jour x est plus ancienne que $T_j \Rightarrow$ écriture autorisée

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y		
RTS	WTS	RTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_i) > WTS(y)$ et la dernière transaction la plus récente ayant lu y est toujours T_j
	$R_j(x)$	125				← $TS(T_j) > WTS(x) \Rightarrow$ la dernière transaction ayant mis à jour x est plus ancienne que $T_j \Rightarrow$ écriture autorisée

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	RTS ^x	WTS	RTS ^y	WTS
99	80	0	0	0	0

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 99	T_j 80	RTS ^x 0	WTS ^x 0	RTS ^y 0	WTS ^y 0	$T_i + \text{récente que } T_j$
$W_i(x)$						

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
		0	0	0	0
$W_i(x)$			99		

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 99	T_j 80	RTS ^x 0	WTS 0	RTS ^y 0	WTS 0
$W_i(x)$	$R_j(y)$		99		

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$	$R_j(y)$		99	80	

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 99	T_j 80	x		y		$T_i + \text{récente que } T_j$
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$	$R_j(y)$		99			
$R_i(y)$					80	

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 99	T_j 80	x		y	
		RTS 0	WTS 0	RTS 0	WTS 0
$W_i(x)$			99		
	$R_j(y)$			80	
$R_i(y)$				99	

T_i + récente que T_j

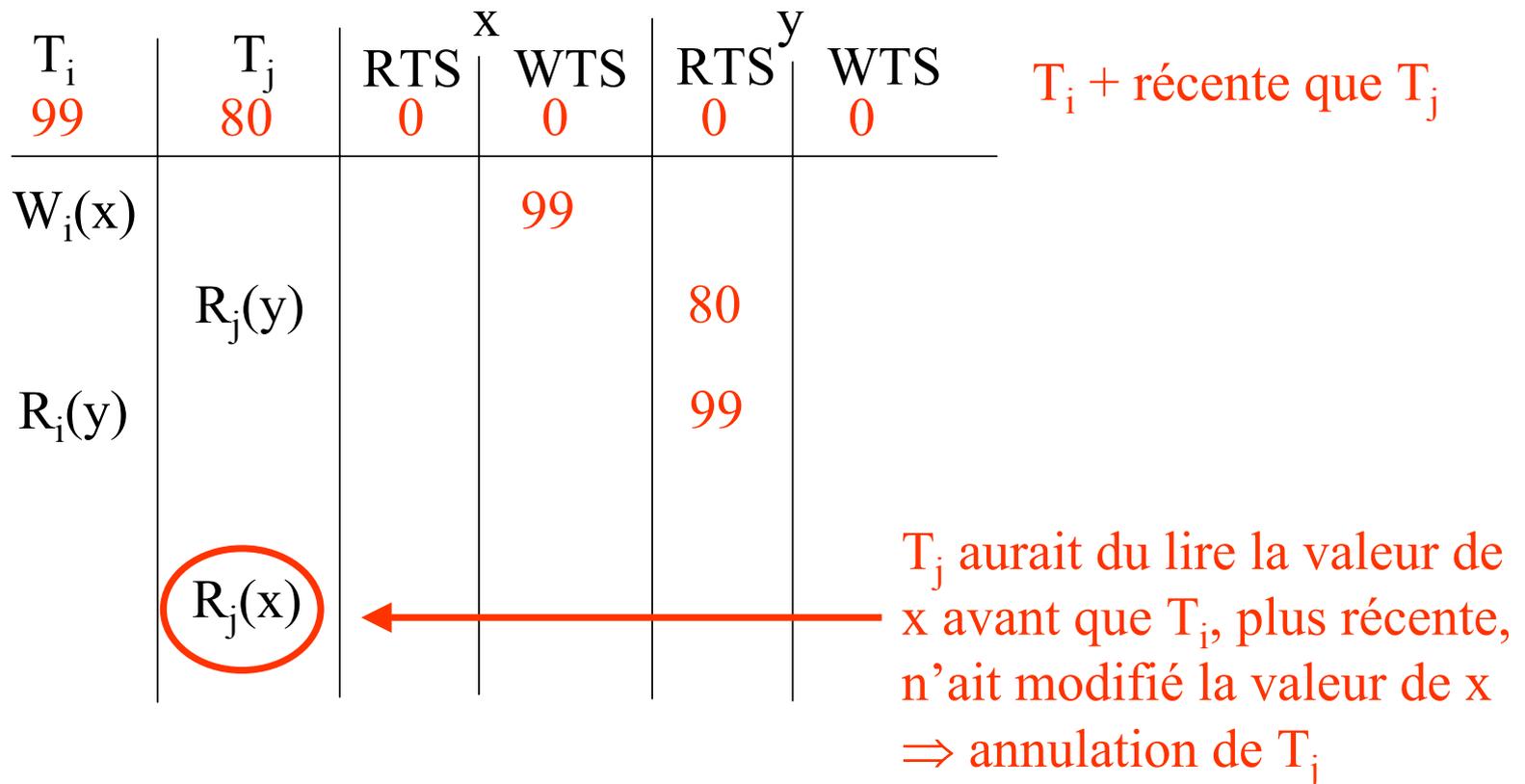
Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), R_j(x)$...

T_i 99	T_j 80	x		y		$T_i + \text{récente que } T_j$
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$			99			
	$R_j(y)$			80		
$R_i(y)$				99		
	$R_j(x)$					

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x)$, $R_j(y)$, $R_i(y)$, $R_j(x)$...



Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	RTS ^x	WTS	RTS ^y	WTS
100	125	0	0	0	0

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	RTS^x	WTS^x	RTS^y	WTS^y
100	125	0	0	0	0
$R_i(x)$					

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
		RTS	WTS	RTS	WTS
100	125	0	0	0	0
$R_i(x)$		100			

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
100	125	RTS	WTS	RTS	WTS
$R_i(x)$	$R_j(y)$	0	0	0	0
		100			

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
100	125	RTS	WTS	RTS	WTS
$R_i(x)$	$R_j(y)$	0	0	0	0
		100			125

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
100	125	RTS	WTS	RTS	WTS
		0	0	0	0
$R_i(x)$		100			
	$R_j(y)$			125	
$R_i(y)$					

T_i + ancienne que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	RTS	WTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$R_i(x)$		100				
	$R_j(y)$			125		
$R_i(y)$						← $TS(T_i) > WTS(y) \Rightarrow$ la dernière transaction la plus récente ayant lu y est toujours T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	RTS	WTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$R_i(x)$		100				
	$R_j(y)$			125		
$R_i(y)$				125		$TS(T_i) > WTS(y) \Rightarrow$ la dernière transaction la plus récente ayant lu y est toujours T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	RTS	WTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$R_i(x)$		100				
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_i) > WTS(y) \Rightarrow$ la dernière transaction la plus récente ayant lu y est toujours T_j
	$W_j(x)$					

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	WTS	RTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$R_i(x)$		100				
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_j) > WTS(y) \Rightarrow$ la dernière transaction la plus récente ayant lu y est toujours T_j
	$W_j(x)$					← $TS(T_j) > RTS(x) \Rightarrow$ la dernière transaction ayant lu x est plus ancienne que $T_j \Rightarrow$ écriture autorisée

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	WTS	RTS	WTS	RTS	WTS	
100	125	0	0	0	0	T_i + ancienne que T_j
$R_i(x)$		100				
	$R_j(y)$			125		
$R_i(y)$				125		← $TS(T_j) > WTS(y) \Rightarrow$ la dernière transaction la plus récente ayant lu y est toujours T_j
	$W_j(x)$		125			← $TS(T_j) > RTS(x) \Rightarrow$ la dernière transaction ayant lu x est plus ancienne que $T_j \Rightarrow$ écriture autorisée

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	RTS ^x	WTS	RTS ^y	WTS
99	80	0	0	0	0

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i 99	T_j 80	RTS^x 0	WTS 0	RTS^y 0	WTS 0
$R_i(x)$					

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
		0	0	0	0
$R_i(x)$		99			

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
$R_i(x)$	$R_j(y)$	0	0	0	0
		99			

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
$R_i(x)$	$R_j(y)$	0	0	0	0
		99		80	

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
		0	0	0	0
$R_i(x)$		99			
	$R_j(y)$			80	
$R_i(y)$					

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
99	80	RTS	WTS	RTS	WTS
		0	0	0	0
$R_i(x)$		99			
	$R_j(y)$			80	
$R_i(y)$				99	

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i 99	T_j 80	x		y		$T_i + \text{récente que } T_j$
		RTS 0	WTS 0	RTS 0	WTS 0	
$R_i(x)$		99				
	$R_j(y)$			80		
$R_i(y)$				99		
	$W_j(x)$					

Ordonnement par estampillage

Soit l'ordonnement : ... $R_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y		
RTS	WTS	RTS	WTS	RTS	WTS	
99	80	0	0	0	0	T_i + récente que T_j
$R_i(x)$		99				
	$R_j(y)$			80		
$R_i(y)$				99		
	$W_j(x)$					T_j aurait dû modifier la valeur de x avant que T_i , plus récente, ne la lise \Rightarrow annulation de T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i 99	T_j 80	RTS ^x 0	WTS ^x 0	RTS ^y 0	WTS ^y 0	$T_i + \text{récente que } T_j$
$W_i(x)$						

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$			99		

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i 99	T_j 80	RTS ^x 0	WTS 0	RTS ^y 0	WTS 0
$W_i(x)$	$R_j(y)$		99		

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$	$R_j(y)$		99	80	

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$			99		
	$R_j(y)$			80	
$R_i(y)$					

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$			99		
	$R_j(y)$			80	
$R_i(y)$				99	

T_i + récente que T_j

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x)$...

T_i 99	T_j 80	x		y		$T_i + \text{récente que } T_j$
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$			99			
	$R_j(y)$			80		
$R_i(y)$				99		
	$W_j(x)$					

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x)$, $R_j(y)$, $R_i(y)$, $W_j(x)$...

T_i	T_j	x		y	
RTS	WTS	RTS	WTS	RTS	WTS
99	80	0	0	0	0
$W_i(x)$			99		
	$R_j(y)$			80	
$R_i(y)$				99	
	$W_j(x)$				

T_i + récente que T_j

T_j aurait dû modifier la valeur de x avant que T_i , plus récente, ne la modifie
 \Rightarrow si l'ordre des estampilles avait été suivi, la MàJ de T_j aurait été remplacée par celle de T_i (règles de Thomas)

Ordonnement par estampillage

Soit l'ordonnement : ... $W_i(x), R_j(y), R_i(y), W_j(x) \dots$

	T_i 99	T_j 80	x		y	
			RTS 0	WTS 0	RTS 0	WTS 0
$W_i(x)$				99		
$R_j(y)$					80	
$R_i(y)$					99	
		$W_j(x)$				

T_i + récente que T_j

T_j aurait dû modifier la valeur de x avant que T_i , plus récente, ne la modifie
 \Rightarrow si l'ordre des estampilles avait été suivi, la MàJ de T_j aurait été remplacée par celle de T_i (règles de Thomas)

\Rightarrow L'opération $W_j(x)$ est ignorée, et T_j n'est pas annulée

Ordonnancement par estampillage

T_i	T_j	T_k	T_n	A
150	200	175	252	$RTS=0$ $WTS=0$
$R_i(A)$	$R_j(A)$ $W_j(A)$			$RTS=150$ $WTS=0$
$W_i(A)$				$RTS=150$ $WTS=150$
				$RTS=200$ $WTS=150$
				$RTS=200$ $WTS=200$
		$R_k(A)$		
		Abort		
			$R_n(A)$	$RTS=252$ $WTS=200$

T_k n'a pas le droit de lire un item modifié par une transaction plus récente

Estampillage multiversion

T_i	T_j	T_k	T_n	A_0	A_{150}	A_{200}
<i>150</i>	<i>200</i>	<i>175</i>	<i>252</i>			
$R_i(A)$ $W_i(A)$	$R_j(A)$ $W_j(A)$	$R_k(A)$ Abort	$R_n(A)$	Lue	Crée Lue Lue	Crée Lue

T_k lit une ancienne version de l'item

Verrouillage hiérarchique

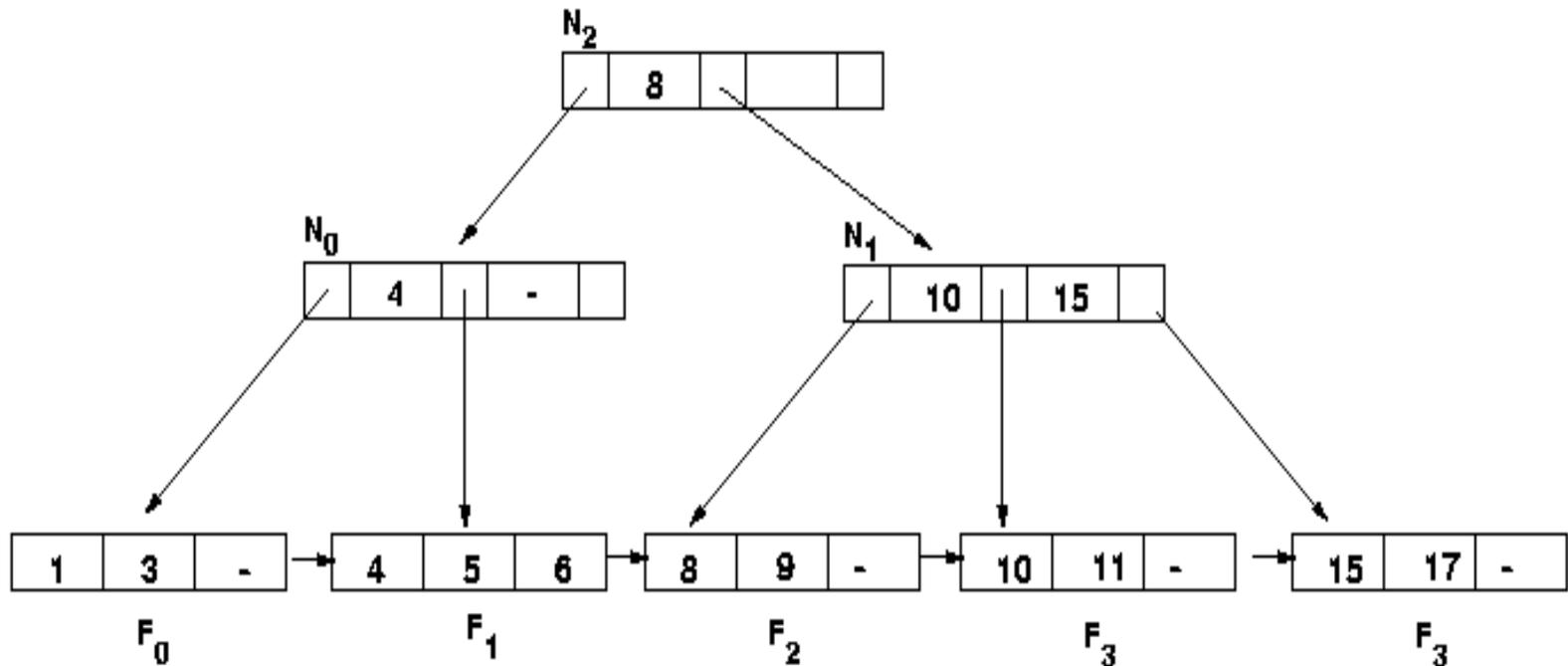
- **Hiérarchie : relation, page, nuplet**
- **Intention d'obtenir un verrou partagé (IP) ou exclusif (IX)**
- **Protocole garantissant la sérialisation et évitant les inter-blocages**

	IP	IX	VP	VX
IP	OUI	OUI	OUI	OUI
IX	OUI	OUI	NON	NON
VP	OUI	NON	OUI	NON
VX	NON	NON	NON	NON

Matrice de compatibilité des verrous et des intentions de verrous

Concurrency dans un arbre B+

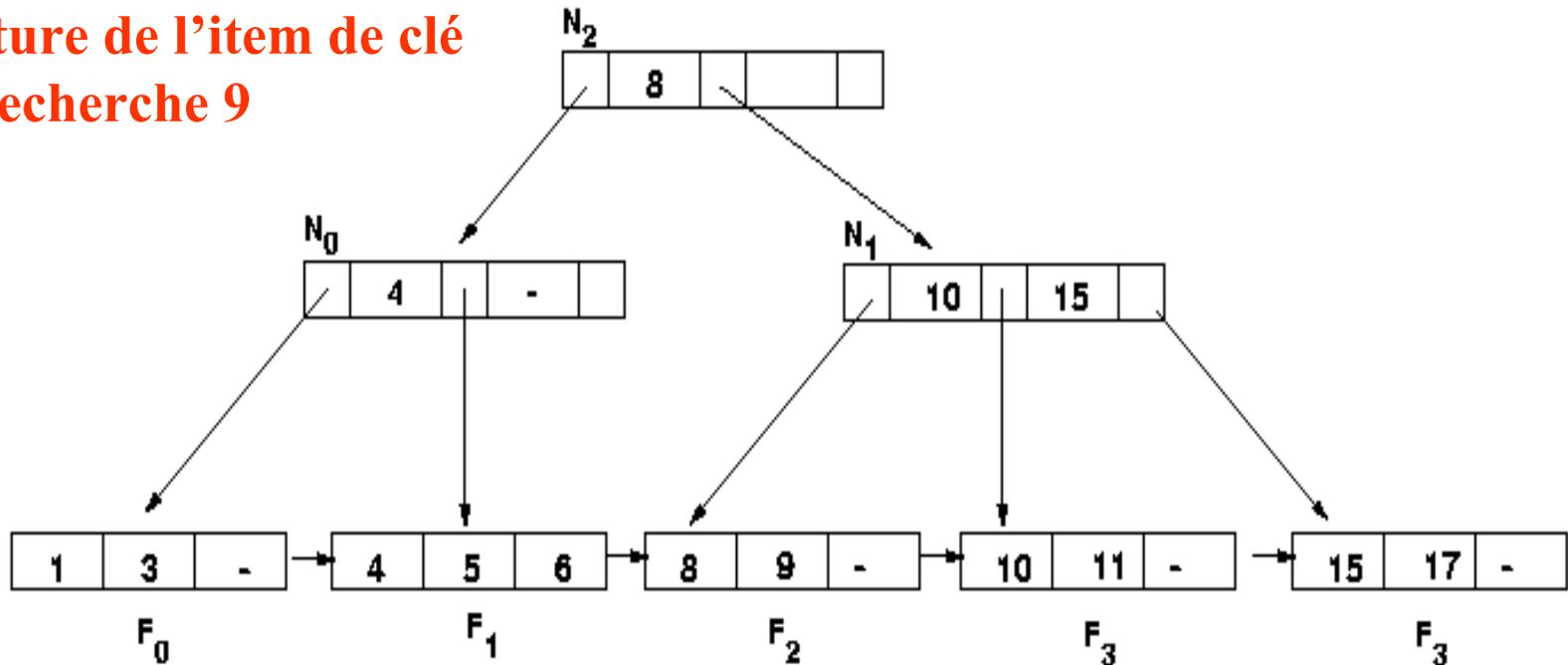
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

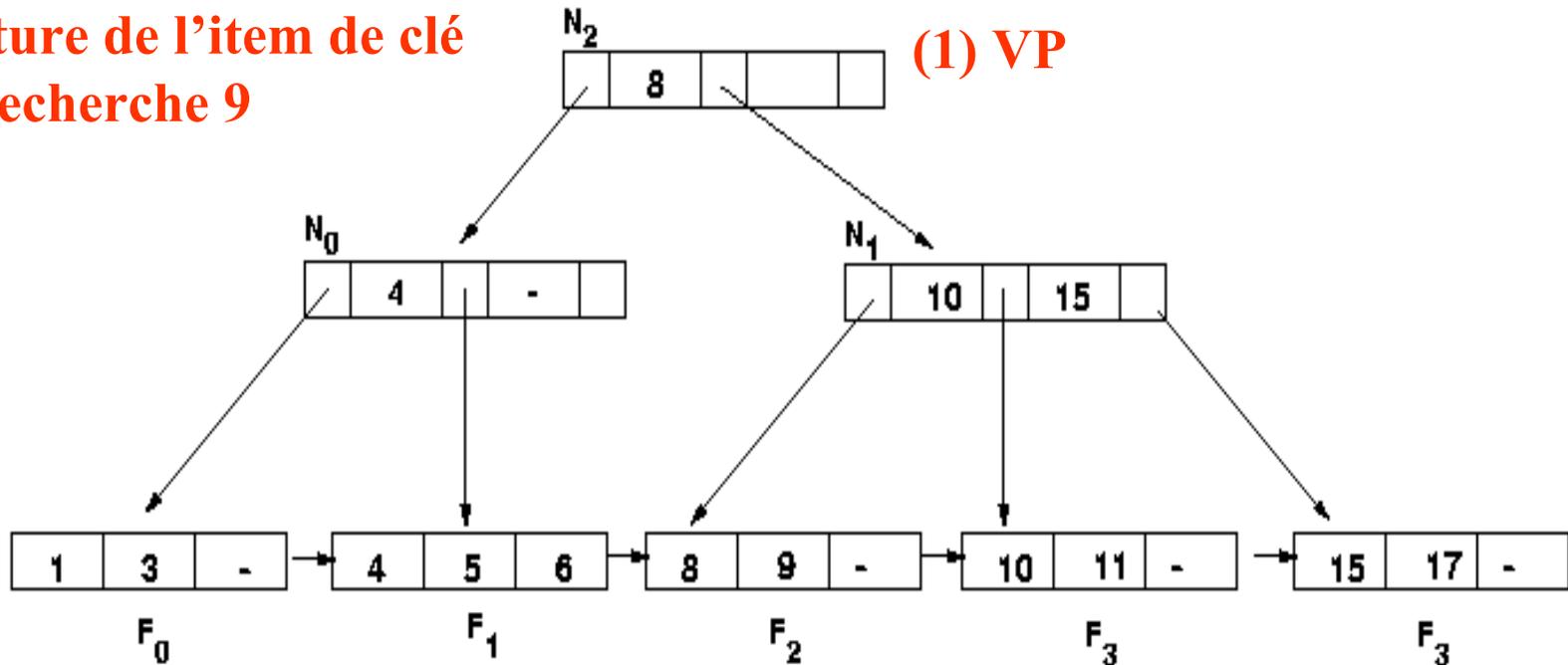
Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

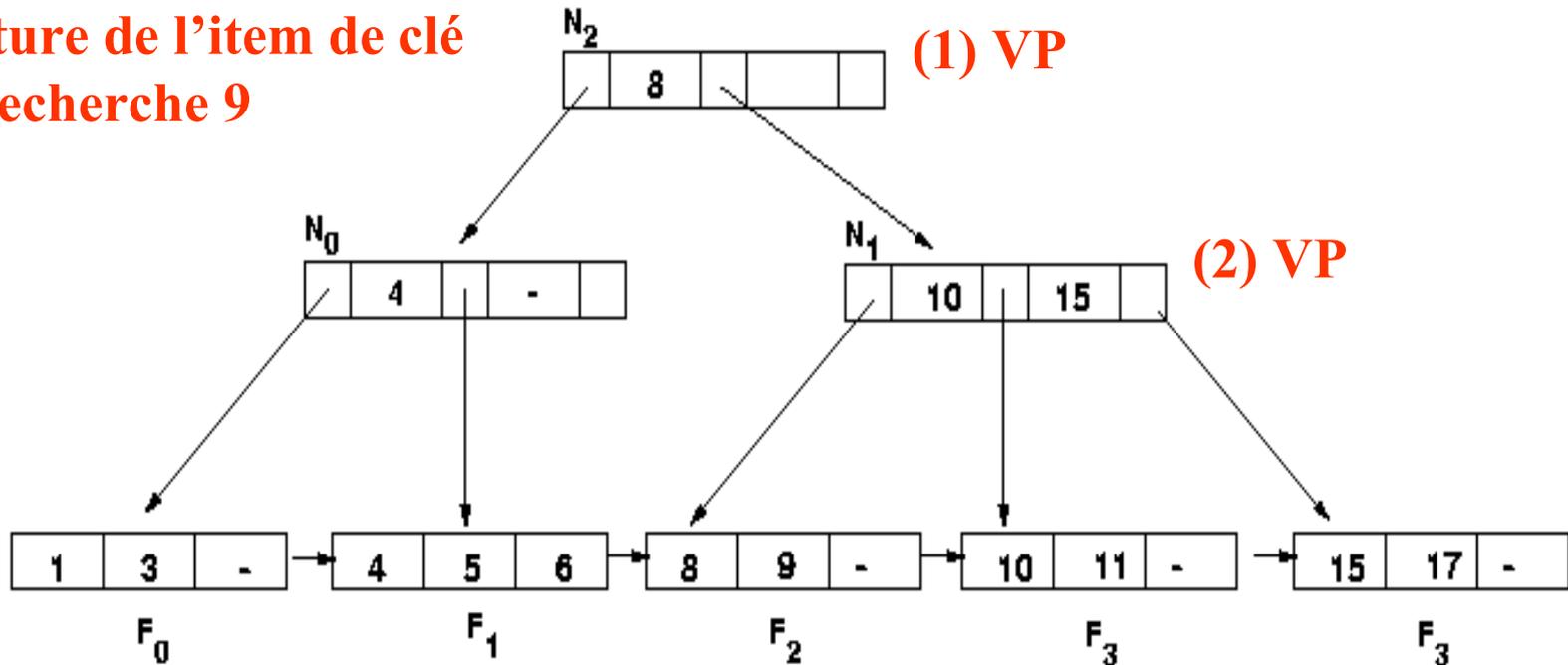
Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

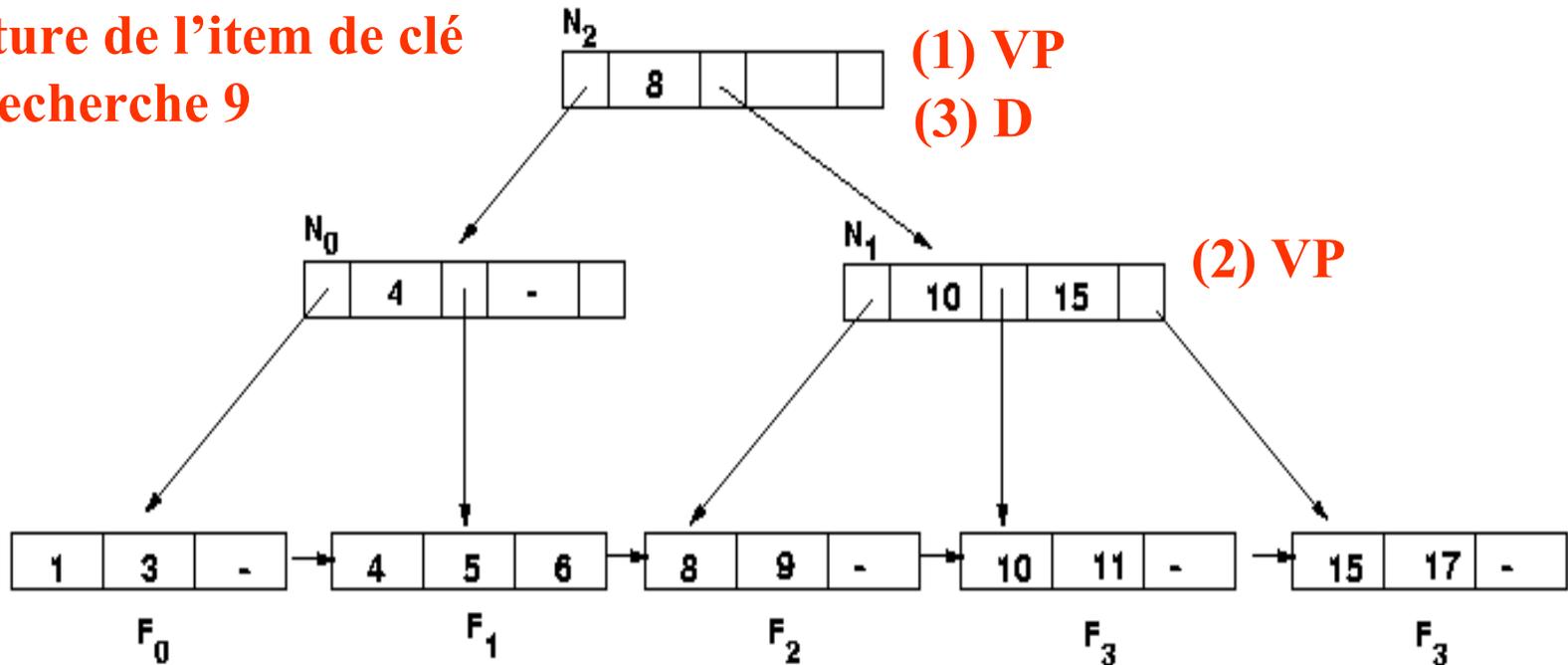
Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

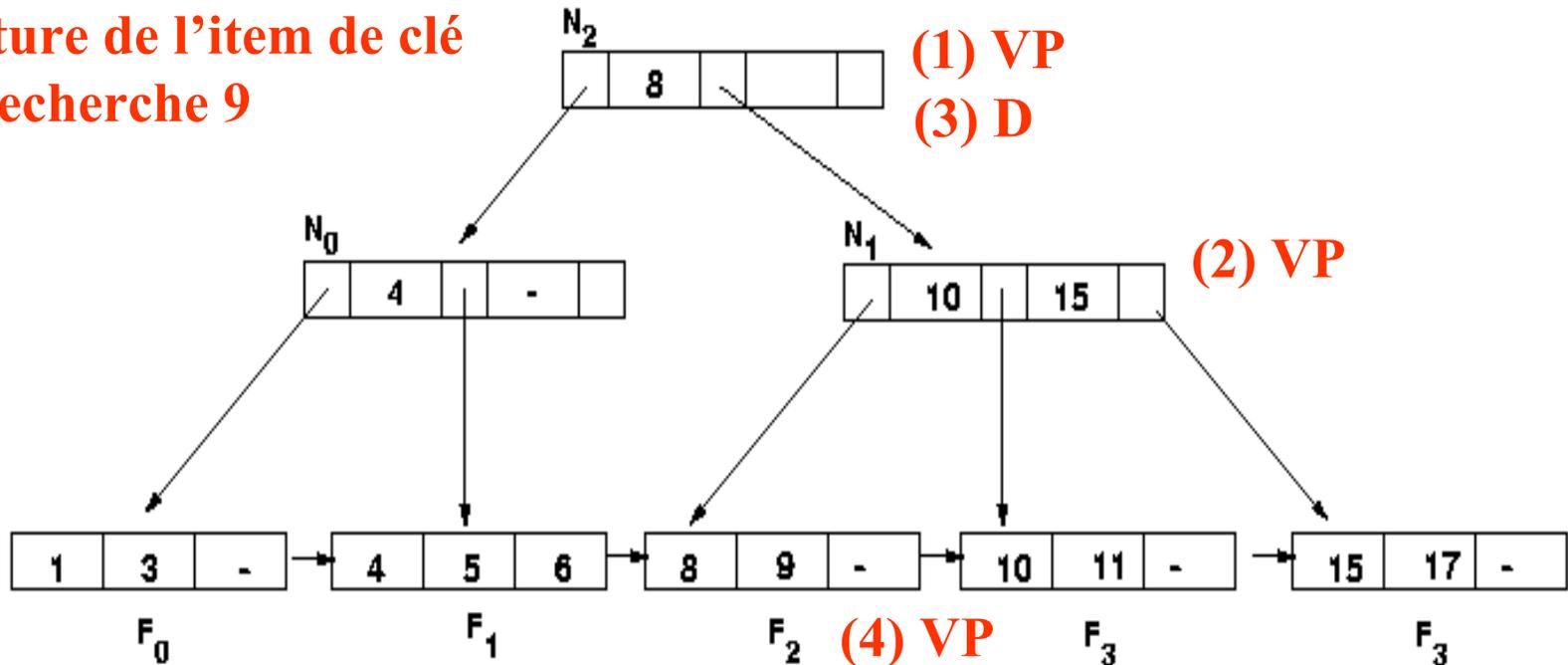
Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

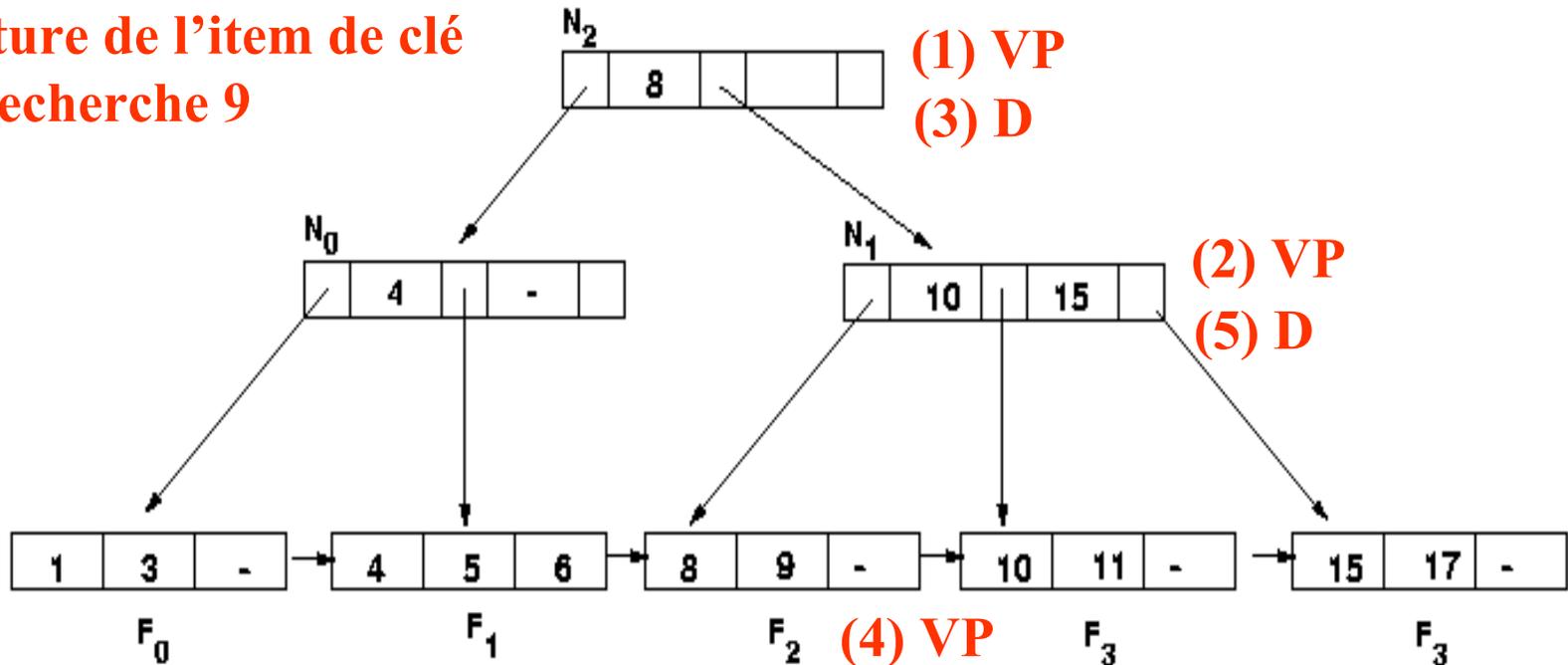
Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

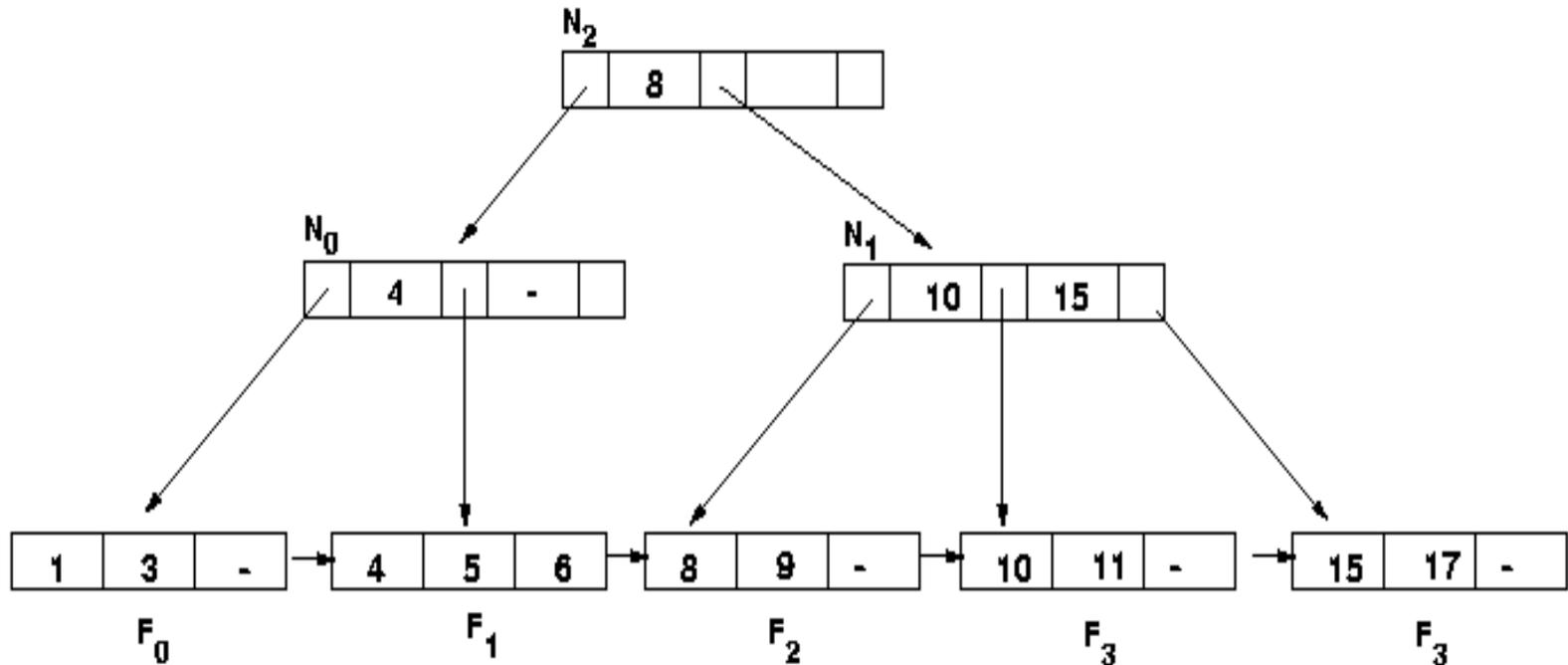
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

Lecture de l'item de clé de recherche 9



Concurrency dans un arbre B+

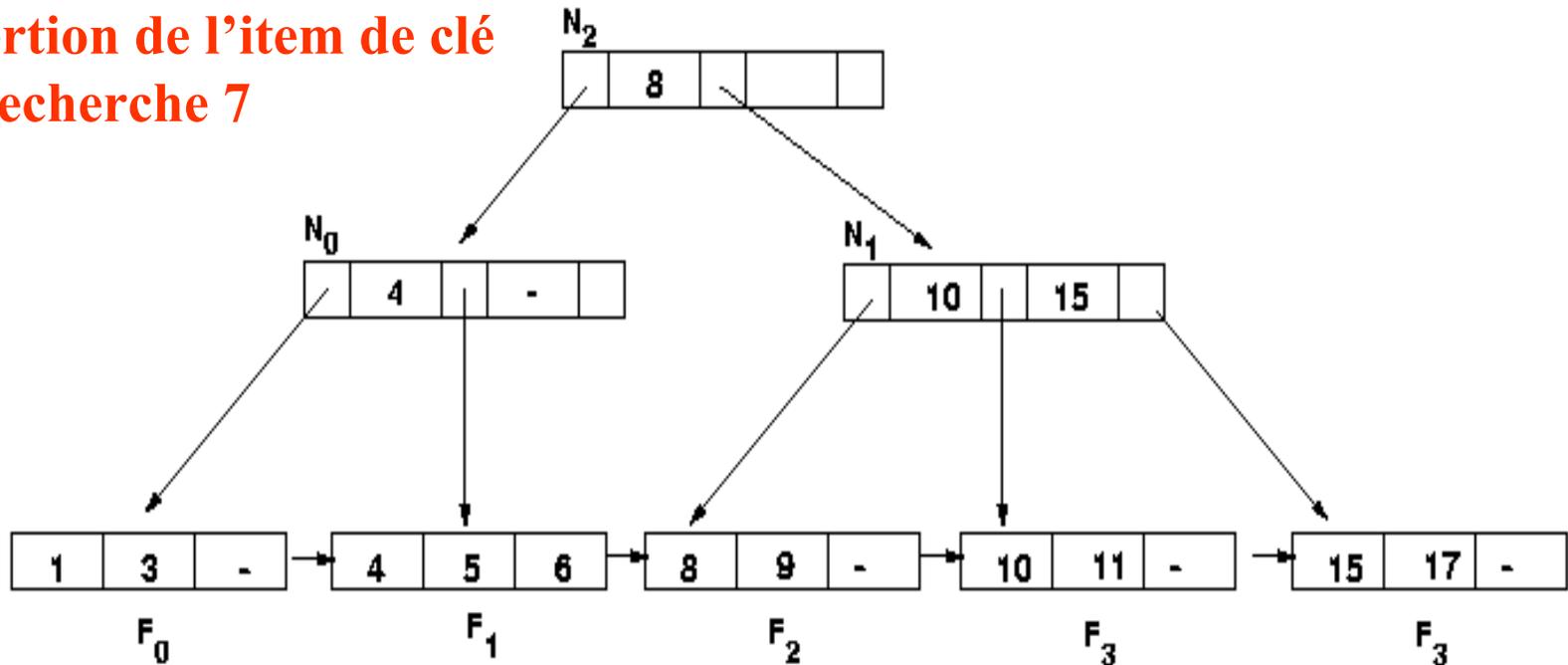
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

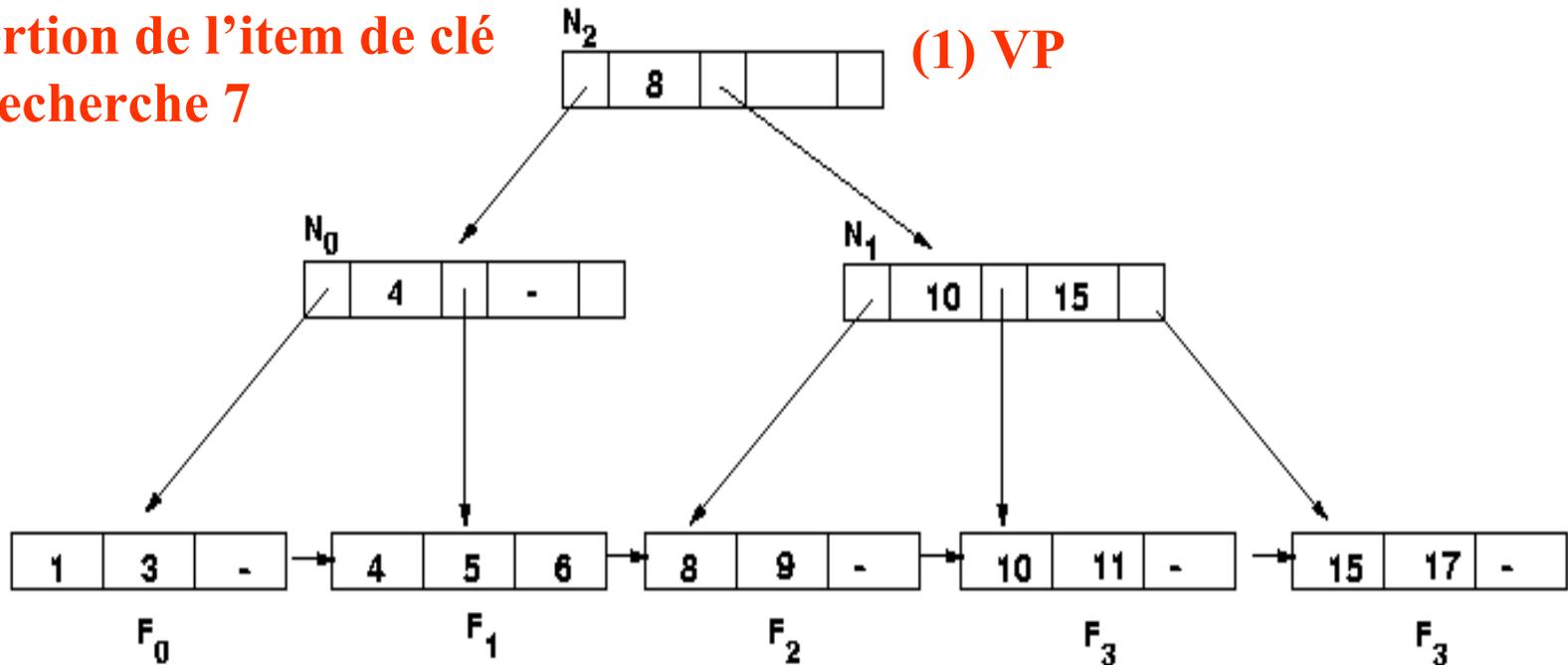
Insertion de l'item de clé de recherche 7



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

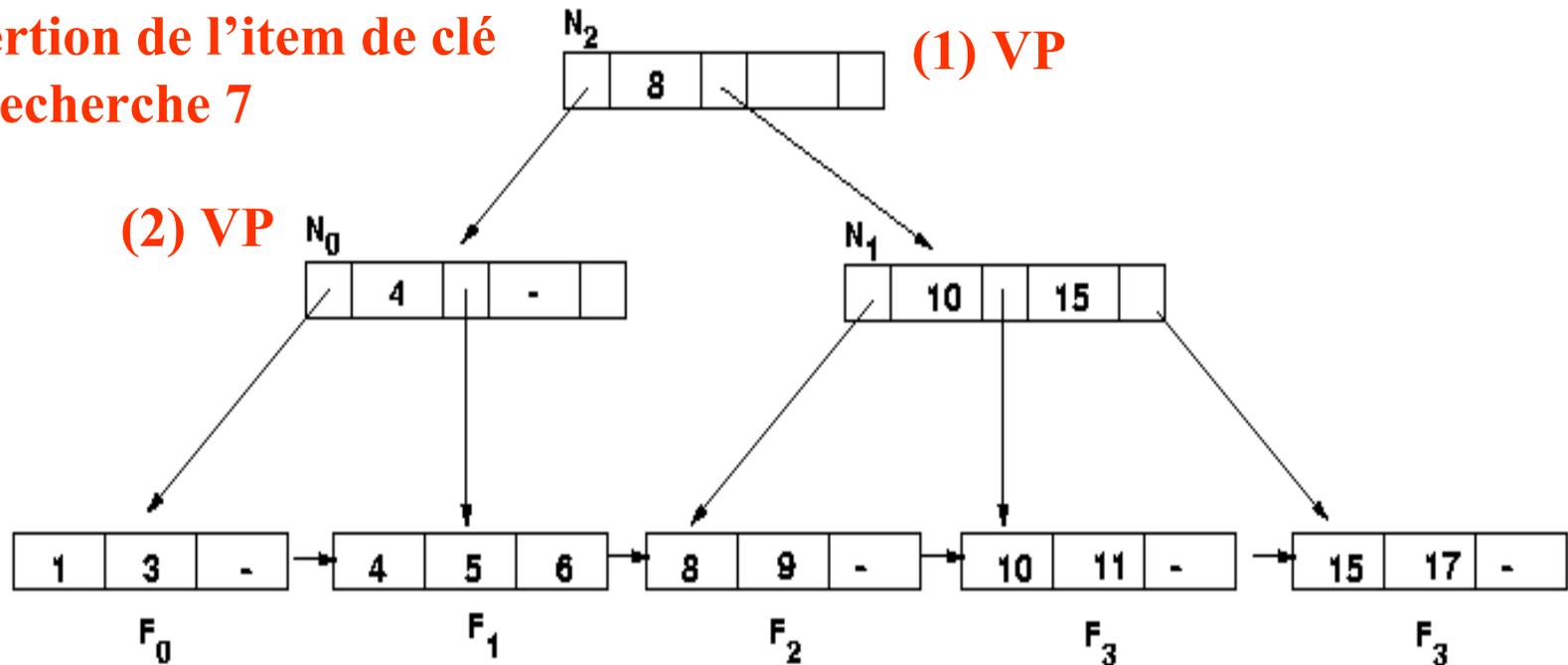
Insertion de l'item de clé de recherche 7



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

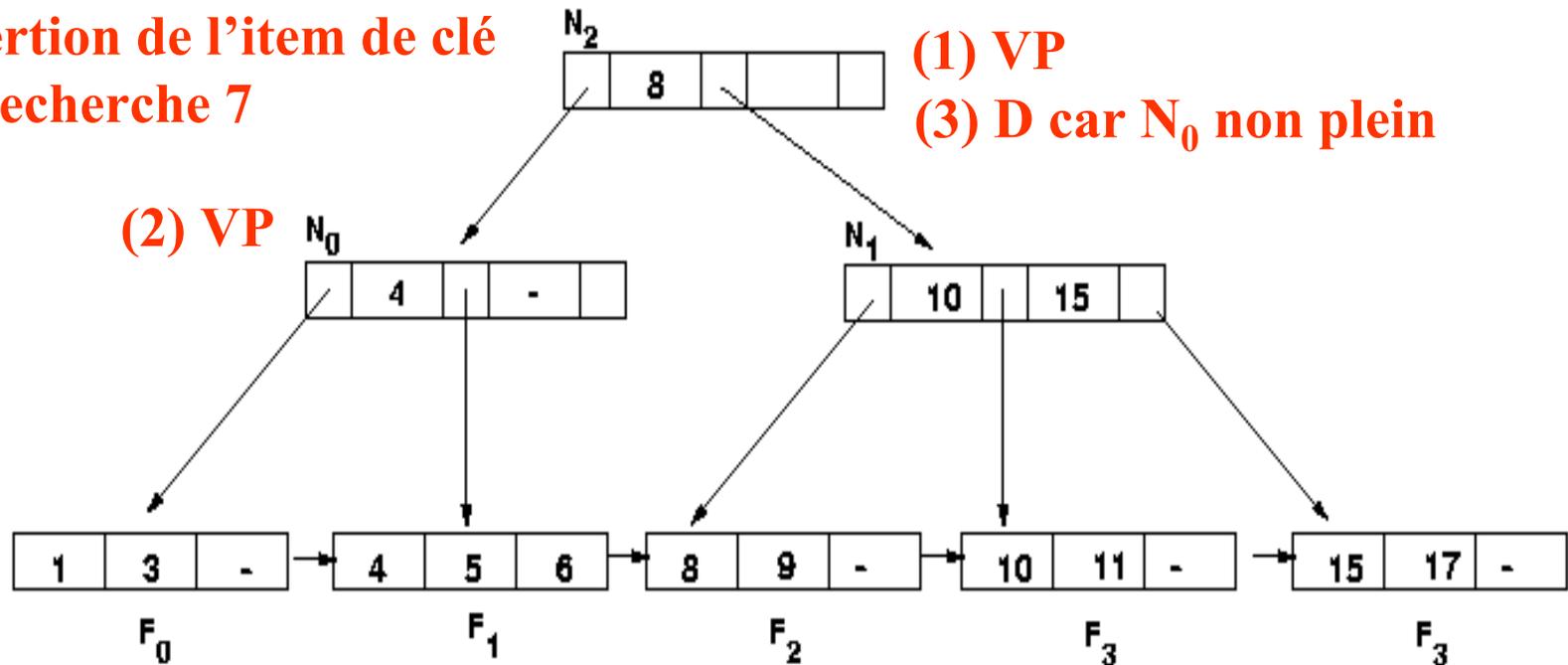
Insertion de l'item de clé de recherche 7



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

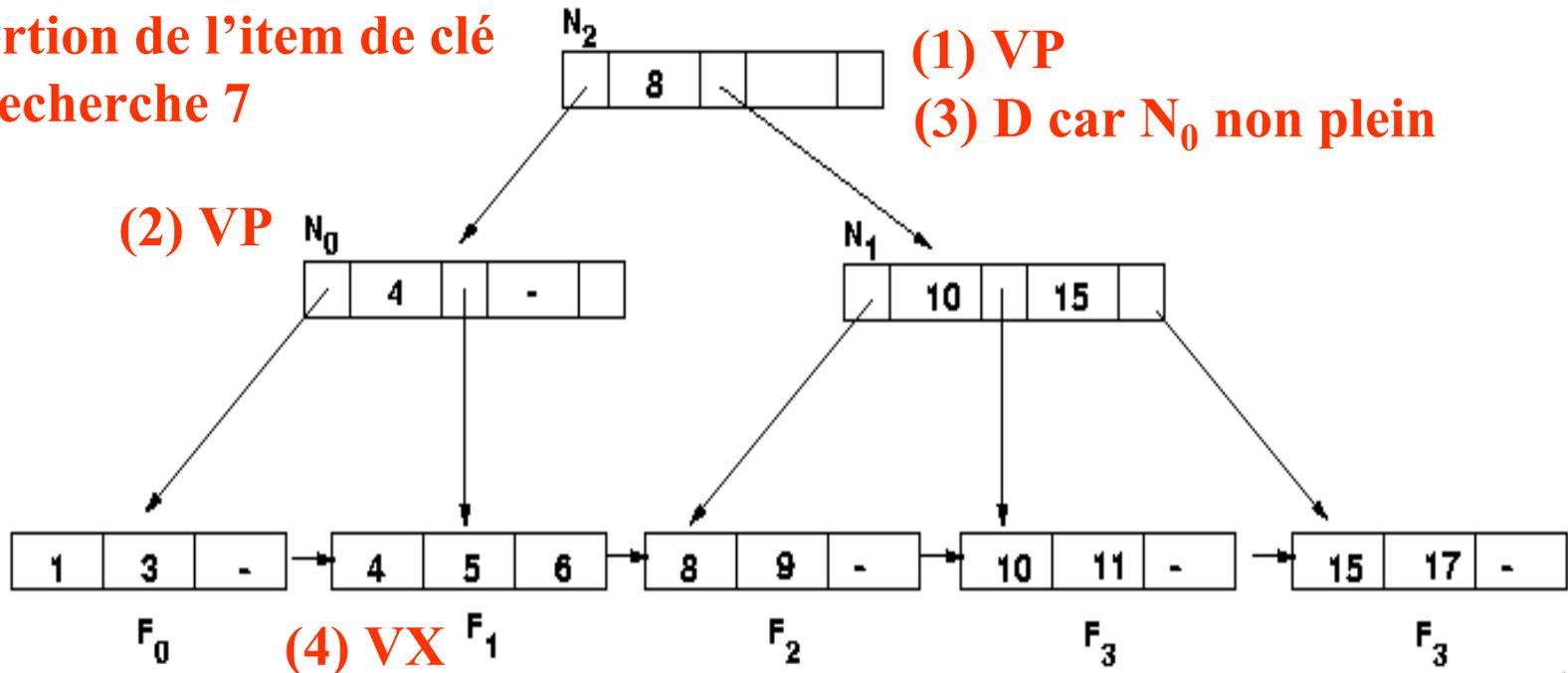
Insertion de l'item de clé de recherche 7



Concurrency dans un arbre B+

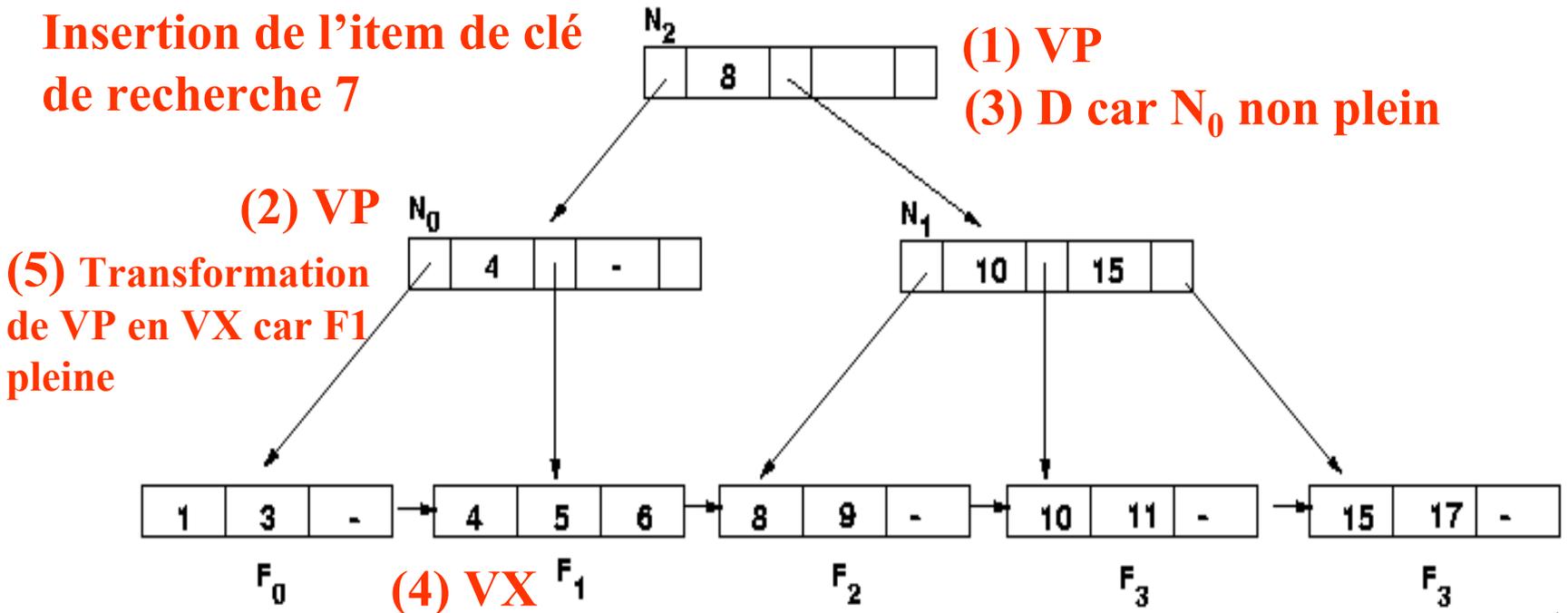
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

Insertion de l'item de clé de recherche 7



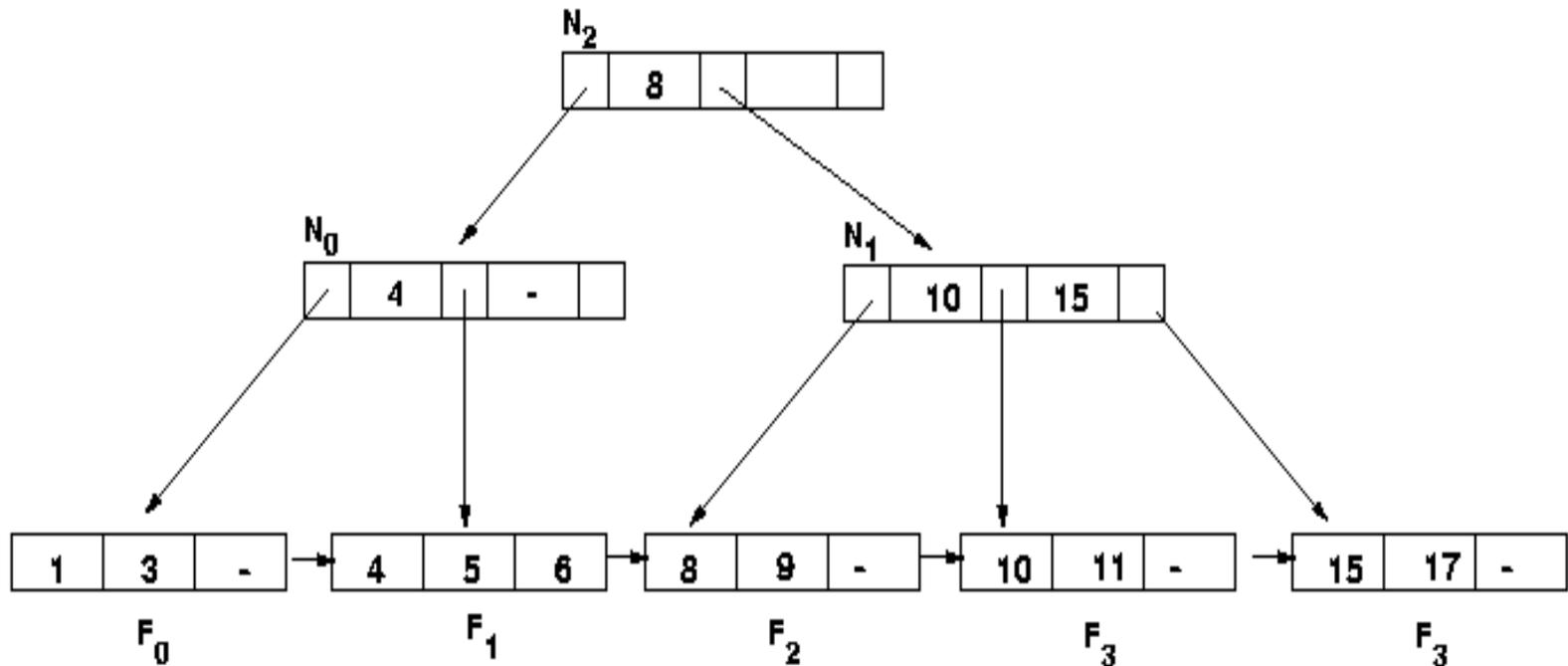
Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



Concurrency dans un arbre B+

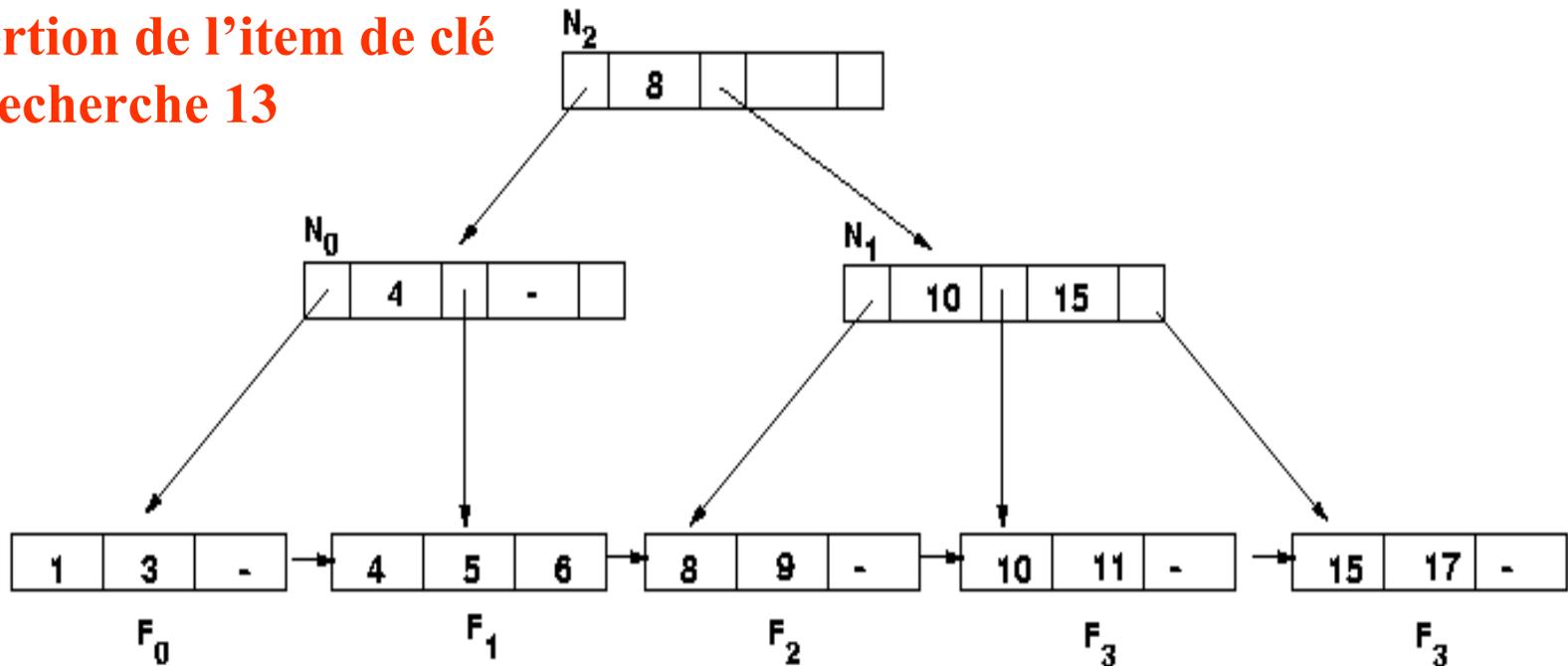
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

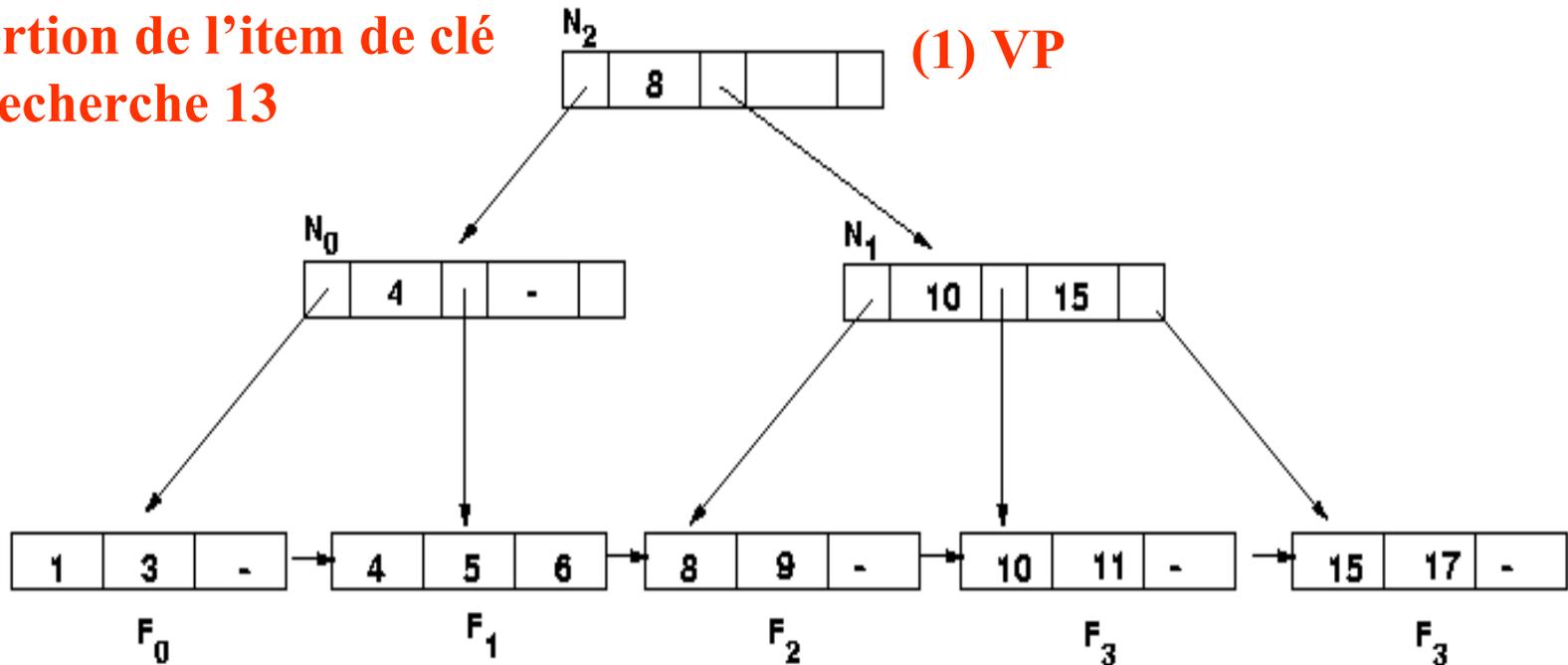
Insertion de l'item de clé de recherche 13



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

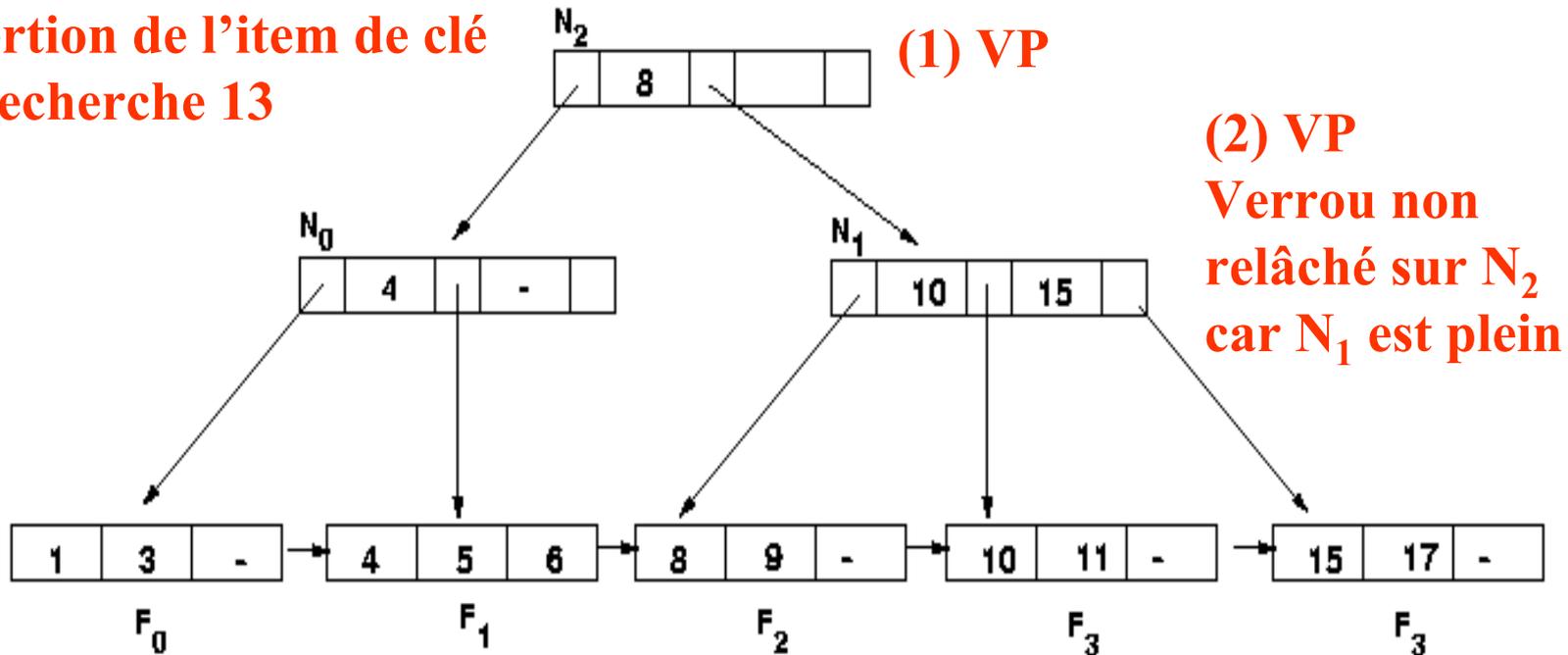
Insertion de l'item de clé de recherche 13



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

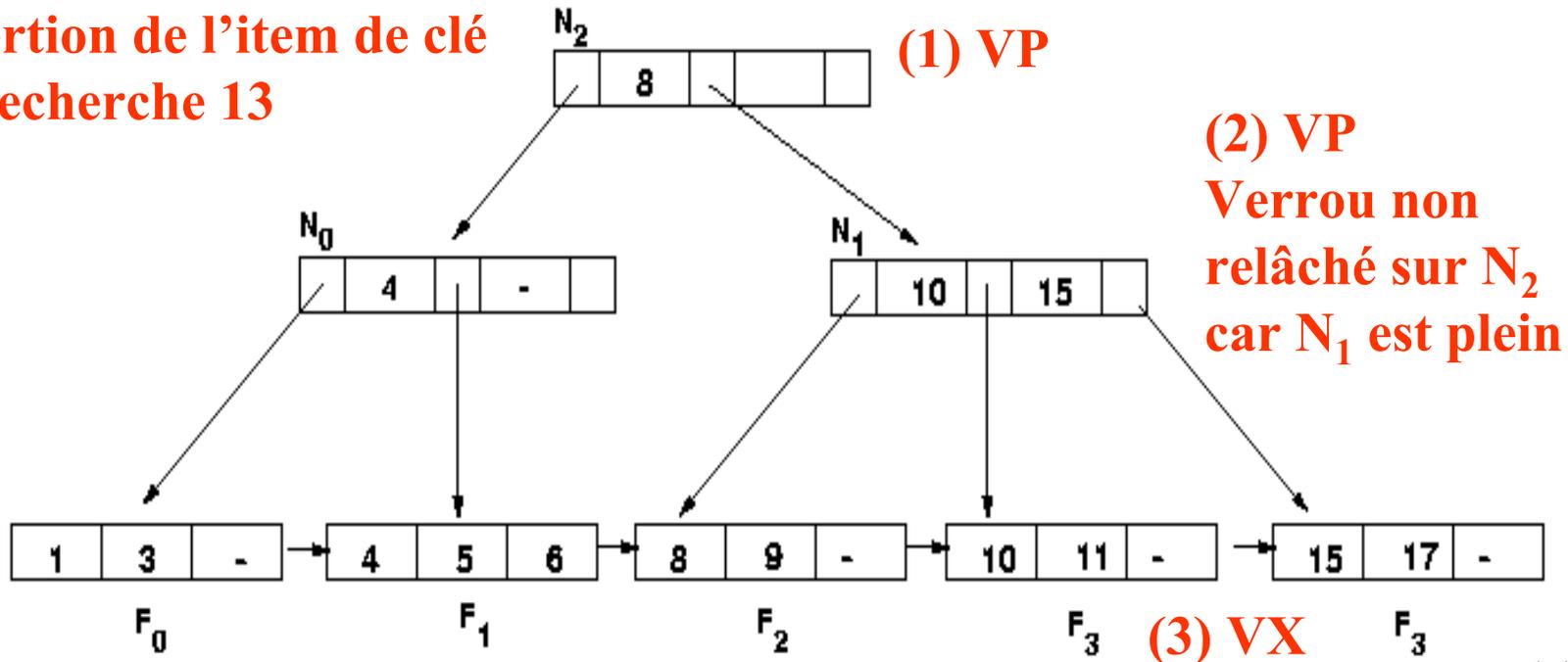
Insertion de l'item de clé de recherche 13



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

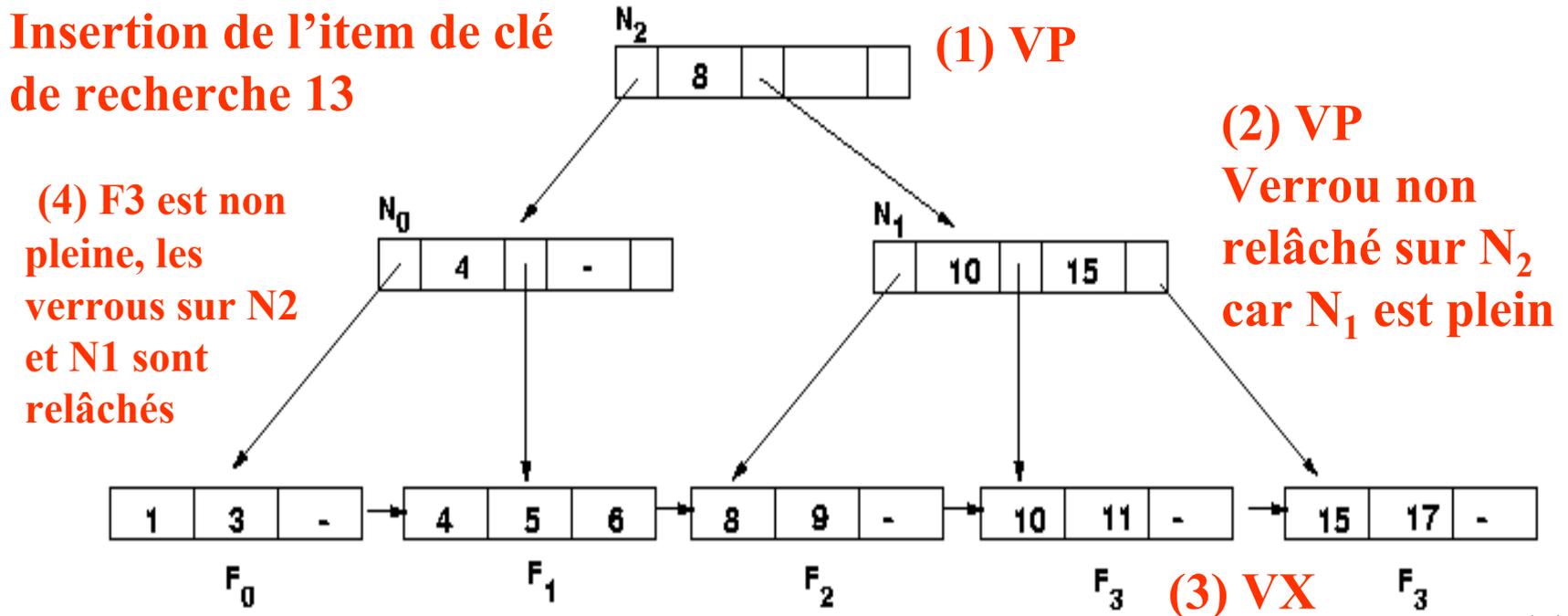
Insertion de l'item de clé de recherche 13



Concurrency dans un arbre B+

- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

Insertion de l'item de clé de recherche 13



Transactions et SQL2

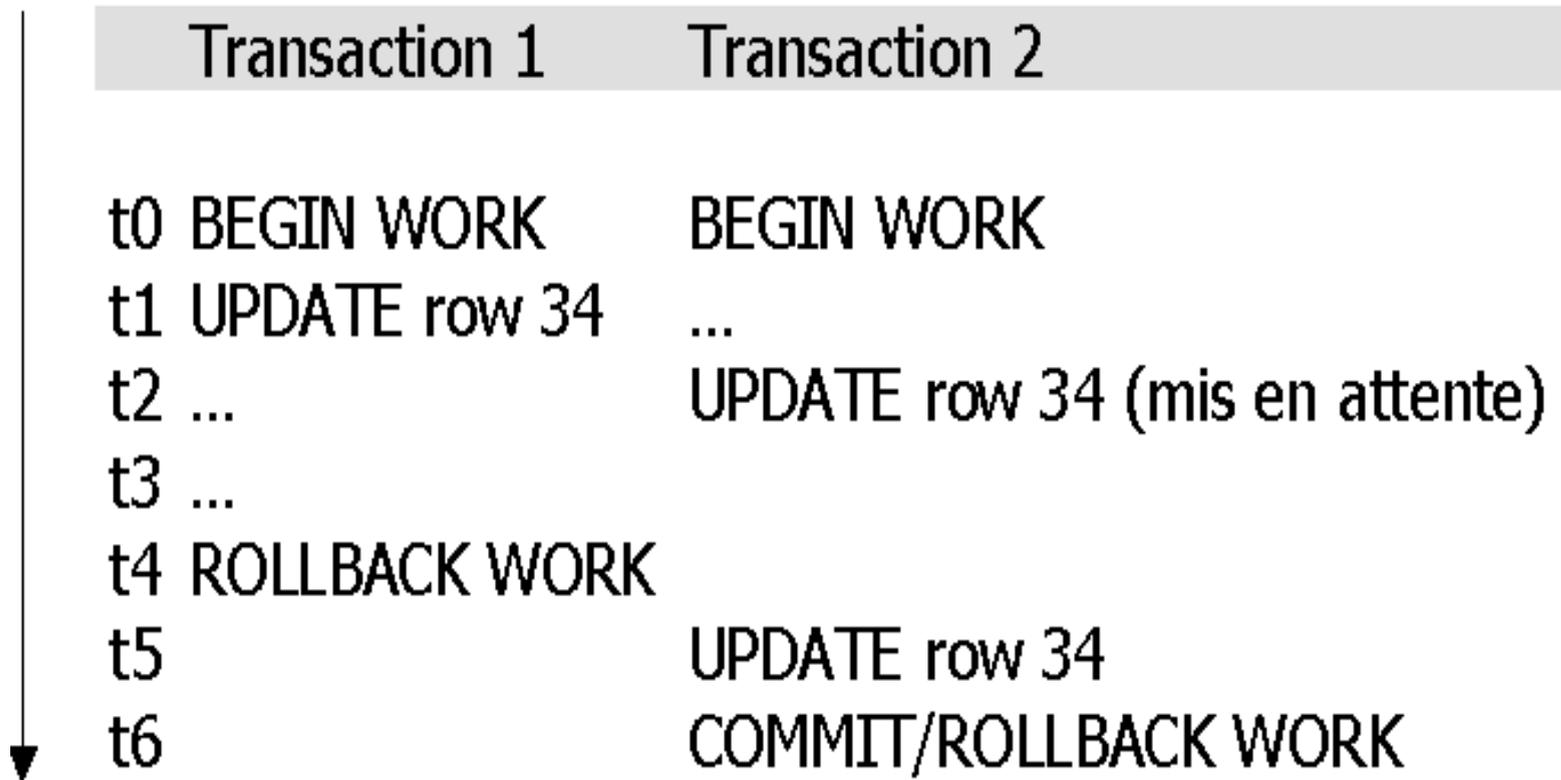
- Une transaction commence dès la 1ère requête ou tout de suite après un *COMMIT* ou un *ROLLBACK*
- Propriétés *READ ONLY* ou *READ WRITE*
- Degrés d'isolation

Degré	Lecture impropre	Lecture non reproductible	Références fantômes
READ UNCOMMITTED	OUI	OUI	OUI
READ COMMITTED	NON	OUI	OUI
REPEATABLE READ	NON	NON	OUI
SERIALIZABLE	NON	NON	NON

- *SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY*

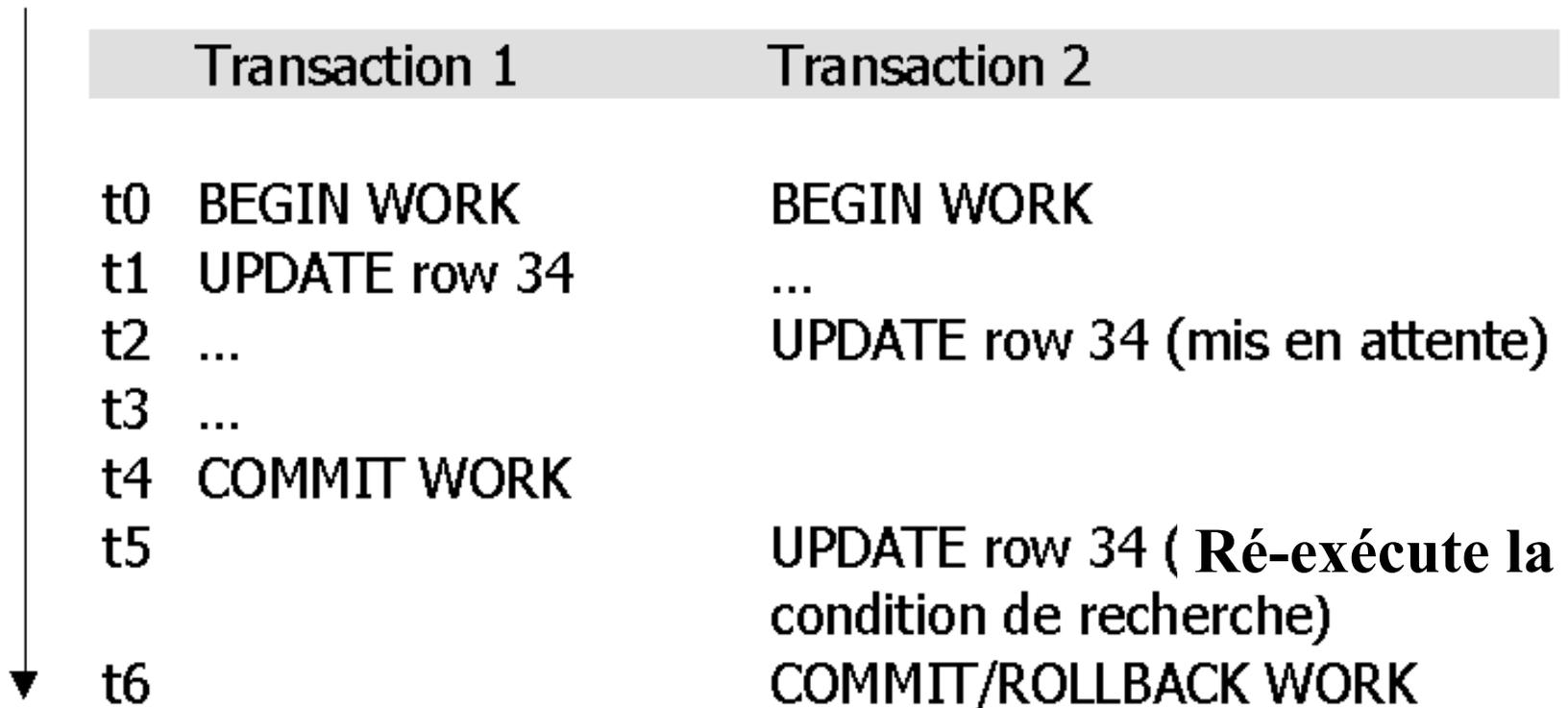
Exemple de PostgreSQL

Pour le niveau d'isolation par défaut : **READ COMMITTED**



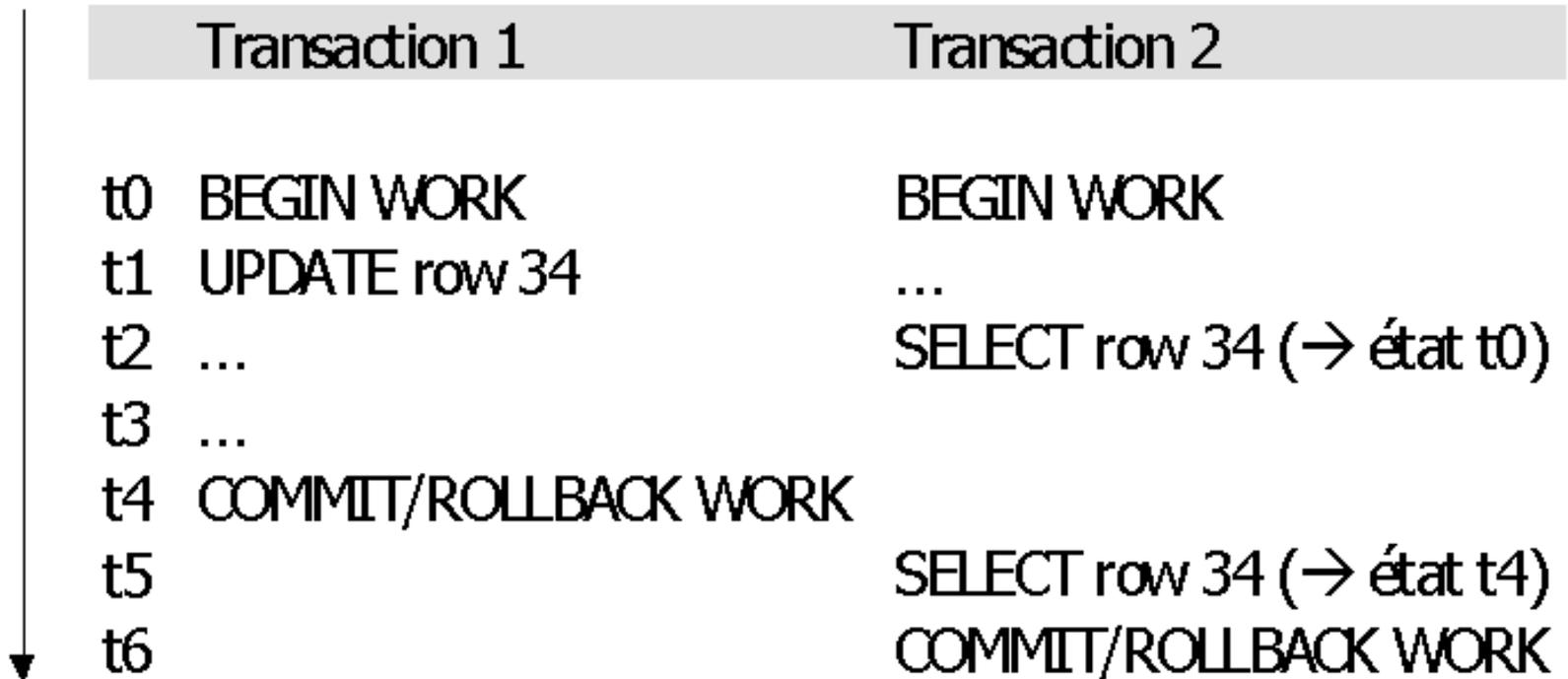
Exemple de PostgreSQL

Pour le niveau d'isolation par défaut : **READ COMMITTED**



Exemple de PostgreSQL

Pour le niveau d'isolation : **SERIALIZABLE**



Exemple de PostgreSQL

Pour le niveau d'isolation : **SERIALIZABLE**

	Transaction normale	Transaction sérializable
t0	BEGIN WORK	BEGIN WORK
t1	UPDATE row 34	...
t2	...	UPDATE row 34 (mis en attente)
t3	...	
t4	COMMIT WORK	
t5		Auto-ROLLBACK WORK



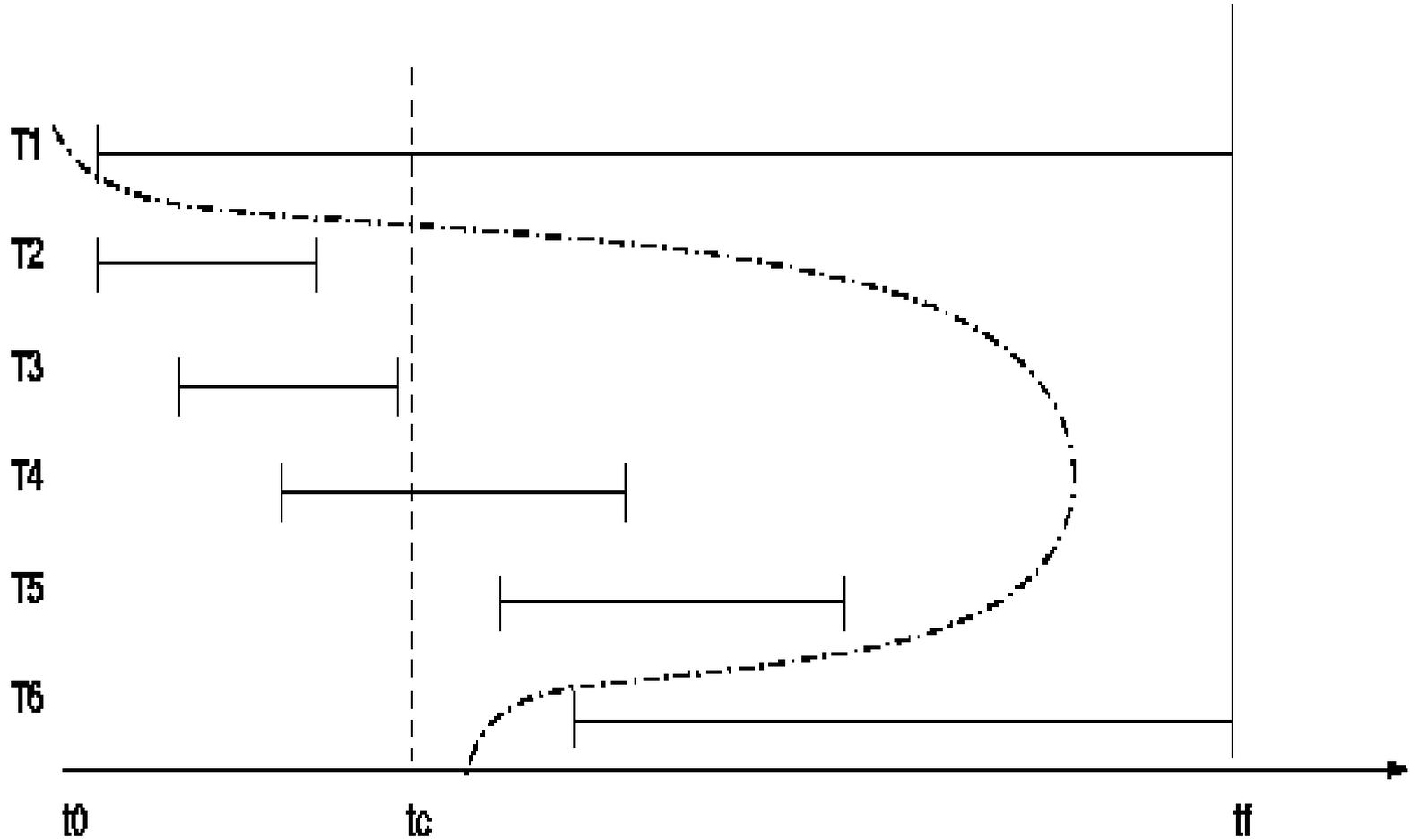
Chap. V - Reprise après panne

- **Types de panne dans les SGBD**
- **Journaux des mises à jour**
- **Validation des transactions**
- **Procédures de reprise**
- **Algorithme ARIES**

Pannes

- Fonctions du **gestionnaire de pannes**
 - ◆ **Atomicité**
 - ◆ **Durabilité**
- Différents types de panne [Gar99]
 - ◆ Panne d'action
 - ◆ Panne de transaction
 - ◆ Panne du système
 - ◆ Panne de la mémoire secondaire

Exemple



Journaux

- **Journal** ou *log*

Historique des modifications effectuées sur la base

- **Journal des images avant (*rollback segment*)**

- ◆ Valeurs des pages avant modifications
- ◆ Pour défaire (*undo*) les mises à jour d'une transaction

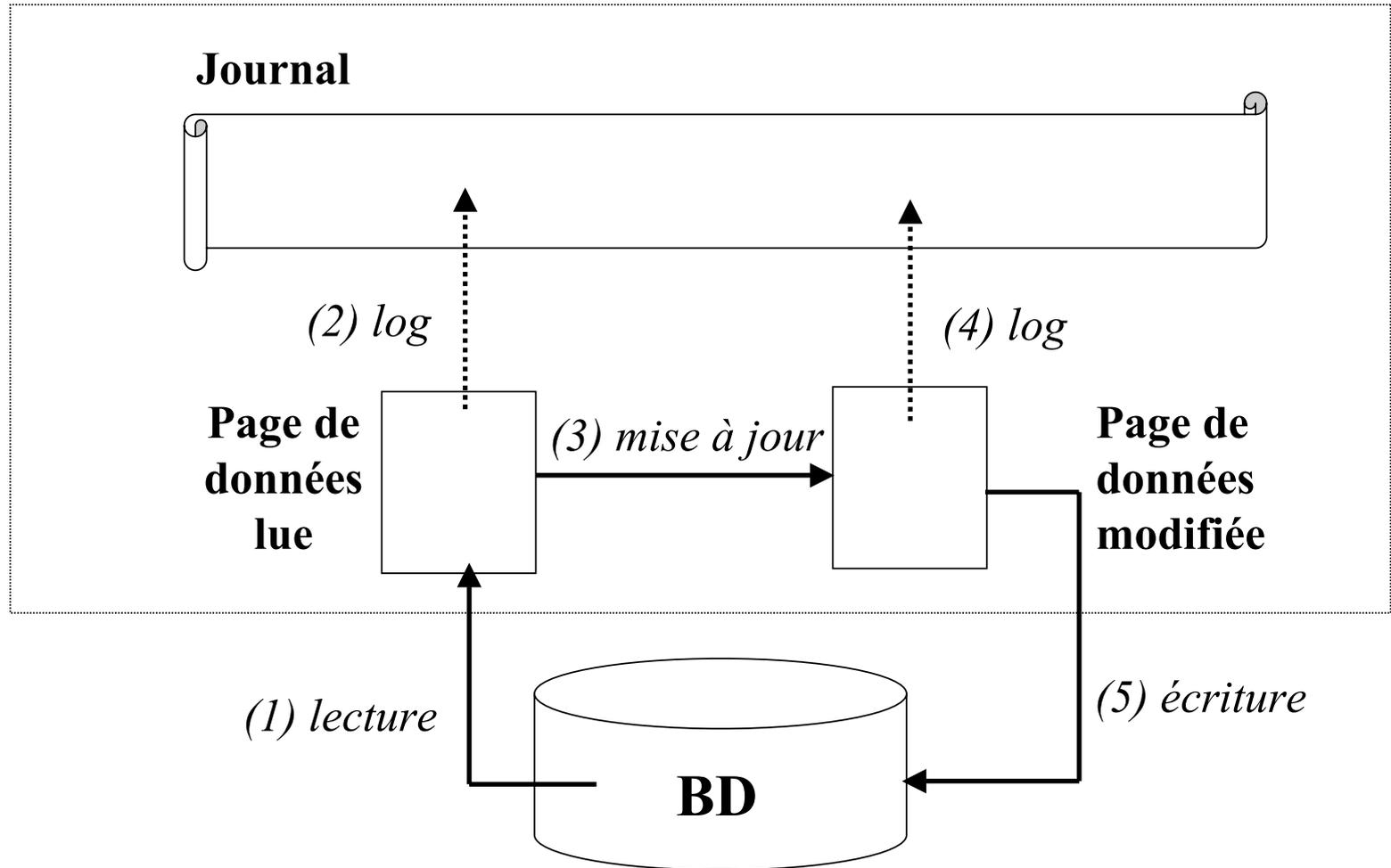
- **Journal des images après (*redo log*)**

- ◆ Valeurs des pages après modifications
- ◆ Pour refaire (*redo*) les mises à jour d'une transaction

- **Points de reprise**

Processus de journalisation

Mémoire



Gestion du journal

- Ecriture des pages du journal dans un *buffer* en mémoire
- Sauvegarde du journal lorsque le *buffer* est plein
- Sauvegarde du journal lorsqu'il y a validation d'une transaction ou d'un groupe de transactions
- **Ecriture du journal sur le disque avant l'écriture des pages de données modifiées**
- **Structures des enregistrements**
 - ◆ Numéro de transaction
 - ◆ Type d'enregistrement (*start, update, commit, abort ...*)
 - ◆ Adresse de la page modifiée
 - ◆ Image avant
 - ◆ Image après

Modification immédiate

- **Etapes**

- ♦ Insertion d'un enregistrement de début de transaction dans le journal
- ♦ A chaque opération d'écriture, insertion d'un enregistrement de modification dans le journal
- ♦ Une fois les enregistrements de modifications inscrits dans le journal, modification des pages de données du *buffer*
- ♦ Report des mises à jour sur le disque quand le *buffer* est plein ou quand la transaction valide
- ♦ Insertion d'un enregistrement de validation dans le journal

- Opérations *undo* et *redo*

- **Lecture du journal en sens inverse pour annuler une transaction**

Modification différée

- **Etapas**

- ◆ Insertion d'un enregistrement de début de transaction dans le journal
- ◆ A chaque opération d'écriture, insertion d'un enregistrement de modification dans le journal
- ◆ Insertion d'un enregistrement de validation dans le journal
- ◆ Après la validation de la transaction, mise à jour des pages du *buffer* en fonction du contenu du journal

- Pas d'opérations *undo*

- Opération *redo*

Procédures de reprise

- **Objectif**

Reconstruire, à partir du journal et éventuellement de sauvegarde, **un état proche de l'état cohérent de la base avant la panne**, en perdant le minimum de travail

- **Reprise à chaud**

Perte de la mémoire mais pas de la mémoire secondaire

- ♦ *No Undo, Redo*

- ♦ *Undo, Redo*

- ♦ *Undo, No Redo*

- **Reprise à froid**

Perte de tout ou partie de la mémoire secondaire

Algorithme ARIES (1/2)

*Algorithm for Recovery and Isolation Exploiting Semantics
(IBM DB2)*

- **Structure des enregistrements**
 - ♦ *LSN (Log Sequence Number)*
 - ♦ *Type*
 - ♦ *PrevLSN*
 - ♦ *PageID*
 - ♦ *UndoNxtLSN*
 - ♦ *Data*
- **Table des transactions : transactions actives ou validées**
- **Tables des pages sales**

Algorithme ARIES (2/2)

- **Journalisation avant écriture**
(*Write Ahead Logging - WAL*)
- **Validation après écriture** (*Write Before Commit*)
- **Validation à deux phases**
- **Enregistrement de compensation**
(*Compensation Log Record*)
- **Algorithme à trois étapes**
 - ① **Analyse**
 - ② **Reconstruction avant**
 - ③ **Reconstruction après**

Oracle

- **Index en arbre B+**
- **Gestion des pannes**
 - ♦ **Journal Avant et Journal Après**
 - ♦ **Ecriture des journaux sur le disque à chaque validation de transaction**
 - ♦ **Possibilité de différer l'écriture des journaux et des pages mémoire pour les groupes de transactions courtes**
- **Utilisation des boucles imbriquées et du tri-fusion**
- **Verrouillage nuplet**
- **Dans le cas réparti**
 - ♦ **Validation à deux phases**
 - ♦ **Réplication asynchrone et synchrone**

DB2

- **Index en arbre B+**
- **Gestion des pannes**
 - ♦ **Journal Avant**
 - ♦ **Journal Après**
 - ♦ **Blocage des validations**
- **Utilisation des boucles imbriquées et du tri-fusion**
- **Verrouillage nuplet-table escaladant**
- **Dans le cas réparti**
 - ♦ **Validation à deux phases**
 - ♦ **Réplication asynchrone**

Sybase

- **Index en arbre B***
- **Gestion des pannes**
 - ♦ **Journalisation des intentions d'écriture**
 - ♦ **Blocage des écritures**
- **Utilisation des boucles imbriquées avec ou sans index**
- **Verrouillage page-table escaladant**
- **Dans le cas réparti**
 - ♦ **Validation à deux phases**
 - ♦ **Réplication asynchrone**

CA-OpenIngres

- **Index en arbre B, ISAM et table de hachage**
- **Gestion des pannes**
 - ♦ **Journal avant et journal après**
 - ♦ **Blocage des écritures**
- **Utilisation des boucles imbriquées, tri fusion et hachage**
- **Verrouillage en deux phases table ou page**
- **Dans le cas réparti**
 - ♦ **Validation à deux phases**
 - ♦ **Réplication asynchrone et synchrone**