

ENSTA

Mastère Spécialisé en Architecture des Systèmes d'Information

Cours C1-3

---

# **Systèmes de Gestion de Bases de Données (SGBD) relationnels**

**Maude Manouvrier**

---

Partie I : les SGBD vus du côté Utilisateur/Programmeur

- Modélisation Entité/Association et UML
- Modèle relationnel et passage au modèle relationnel
- Algèbre relationnelle
- SQL / Embedded SQL / ODBC

# BIBLIOGRAPHIE

## Ouvrages de référence utilisés pour le cours :

**T. Connolly, C. Begg et A. Strachan, *Database Systems A Pratical Approach to Desigh, Implementation and Management*, 1998, ISBN: 0-201-34287-1,**

**G. Gardarin, *Bases de Données - objet/relationnel*, Eyrolles, 1999, ISBN: 2-212-09060-9, disponible à la BU 005.74 GAR**

**R. Ramakrishnan et J. Gehrke, *Database Management Systems*, Second Edition; McGraw-Hill, 2000, ISBN: 0-07-232206-3, disponible à la BU 055.7 RAM**

**A. Silberschatz, H.F. Korth et S. Sudarshan, *Database System Concepts*, McGraw-Hill, 1996, ISBN: 0-07-114810-8, disponible à la BU 005.7 DAT**

**J.D. Ullman et J. Widom, *A first Course in Database Systems*, Prentice Hall, 1997, ISBN: 0-13-887647-9, disponible à la BU 005.7 ULL**

# BIBLIOGRAPHIE

**Autres ouvrages de référence, disponibles à la BU :**

**C.J. Date, *An Introduction to Database Systems*, Addison Wesley**

**C.J. Date, *A Guide to SQL Standard*, Addison Wesley**

**R.A. El Masri et S.B. Navathe, *Fundamentals of Database Systems*, Prentice Hall**

**Ouvrages pédagogiques contenant des exercices corrigés :**

**Philip J. Pratt, *Initiation à SQL - Cours et Exercices corrigés*, Eyrolles, 2001 –  
BU : 005.72 SQL**

**Christian Soutou, *De UML à SQL - Conception de bases de données*, Eyrolles,  
2002 – BU : 005.72 SOU**

**F. Brouard, C. Soutou , *SQL (Synthèse de cours et exercices corrigés)*. Pearson  
Education 2005 – BU : 005.72 SQL**

**Christian Soutou, *SQL Pour Oracle (avec exercices corrigés)*, Eyrolles, 2005 –  
BU 005.72 SOU**

**Nicolas Larousse, *Création de bases de données*, Coll. Synthex, Pearson  
Education, 2006**

# Chap. I - Introduction

- **Base de données :**
  - collection d'informations ou de données qui existent sur une **longue période de temps** [UW97] et qui décrivent les activités d'une ou plusieurs organisations [RG00]
  - ensemble de données **modélisant les objets d'une partie du monde réel** et servant de support à une application informatique [Gar99]
- **SGBD** : Systèmes de Gestion de Bases de Données (*DataBase Management Systems - DBMS*)  
**ensemble de logiciels systèmes** permettant aux utilisateurs d'insérer, de modifier, et de rechercher efficacement des données spécifiques dans **une grande masse d'informations** (pouvant atteindre plusieurs milliards d'octets) **partagée par de multiples utilisateurs** [Gar99]

# SGBD

Principaux composants :

- **Système de gestion de fichiers**
- **Gestionnaire de requêtes**
- **Gestionnaire de transactions**

Principales fonctionnalités :

- **Contrôle de la redondance d'information**
- **Partage des données**
- **Gestion des autorisations d'accès**
- **Vérifications des contraintes d'intégrité**
- **Sécurité et reprise sur panne**

# Abstraction des données

- **Niveau interne ou physique :**
  - plus bas niveau
  - indique **comment** (avec quelles structures de données) sont stockées physiquement les données
- **Niveau logique ou conceptuel :**
  - décrit par un **schéma conceptuel**
  - indique quelles sont les données stockées et quelles sont leurs relations **indépendamment de l'implantation physique**
- **Niveau externe ou vue :**
  - propre à chaque utilisateur
  - décrit par un ou plusieurs **schémas externes**

# Instances et schéma

- **Instances de base de données :**
  - données de la base à un instant donné
  - manipulées par un **langage de manipulation de données (DML - *Data Manipulation Language*)**
- **Schéma de base de données :**
  - description de la structure des données
  - ensemble de définitions exprimées en **langage de description de données (DDL - *Data Definition Language*)**

# Petit historique

- **1960** : systèmes de gestion de fichiers
- **1970** : début des SGBD réseaux et hiérarchiques proches des systèmes de gestion de fichiers  $\Rightarrow$  pas d'interrogation sans savoir où est l'information recherchée ("navigation") et sans écrire de programmes
- **1970** : papier fondateur de CODD sur la théorie des relations  
**fondement de la théorie des bases de données relationnelles**  
INGRES à Berkeley - langage QUEL  
System R IBM à San Jose - langages SEQUEL et QBE
- **1980** : **Apparition des SGBD relationnels sur le marché** (Oracle, Ingres, Informix, Sybase, DB2 ...)
- **1990** : **début des SGBD orientés objet** (Gemstone, O<sub>2</sub>, Orion, Objectstore, Versant, Matisse...).
- **Aujourd'hui** : **relationnel-objet, semi-structuré, multimédia ...**



# Chap II - Modélisation

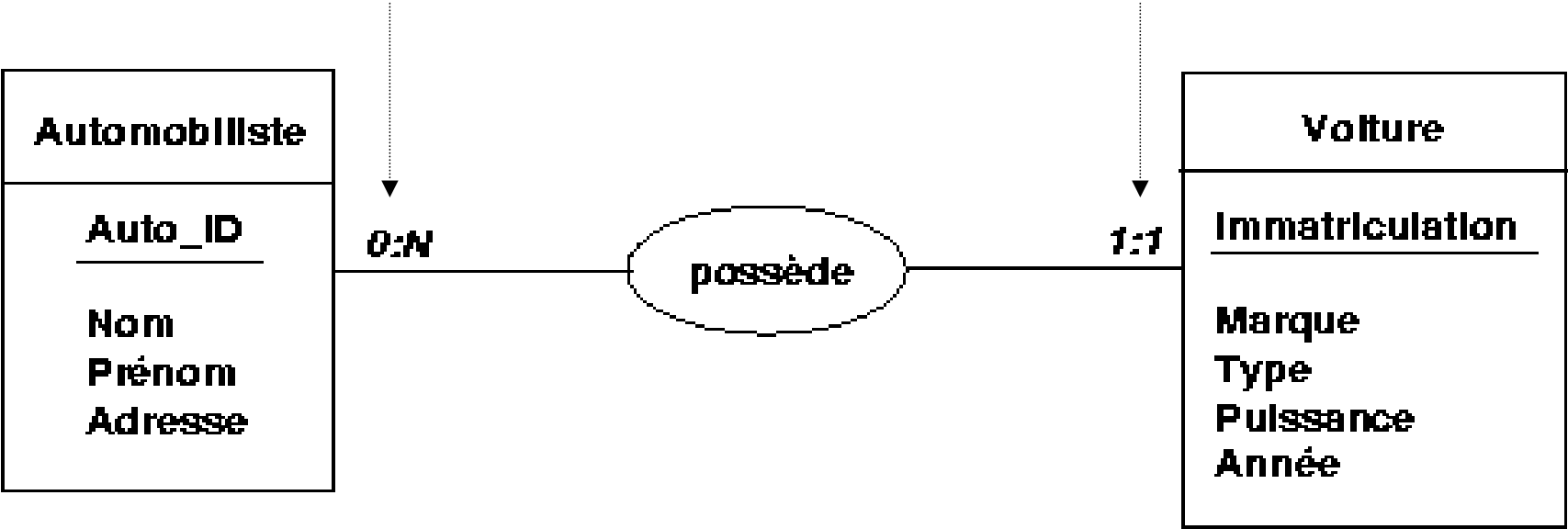
## Méthodologie à suivre pour modéliser un problème

- Déterminer les **entités/classes** et **attributs** :
  - entité/instance de classe = objet décrit par de l'information
  - objet caractérisé uniquement par un identifiant = attribut
  - attribut multi-valué ou avec une association 1:N = entité ou instance
  - attacher les attributs aux ensemble d'entités/classes qu'ils décrivent le plus directement
  - éviter au maximum les identificateurs composites
- Identifier les **généralisations-spécialisations/héritage**
- Définir les **associations**
  - éliminer les associations redondantes
  - éviter les associations n-aires
  - calculer les **cardinalités** de chaque association

# Modélisation Entité/Association (Format Merise)

*Un automobiliste possède  
entre zéro et N voitures*

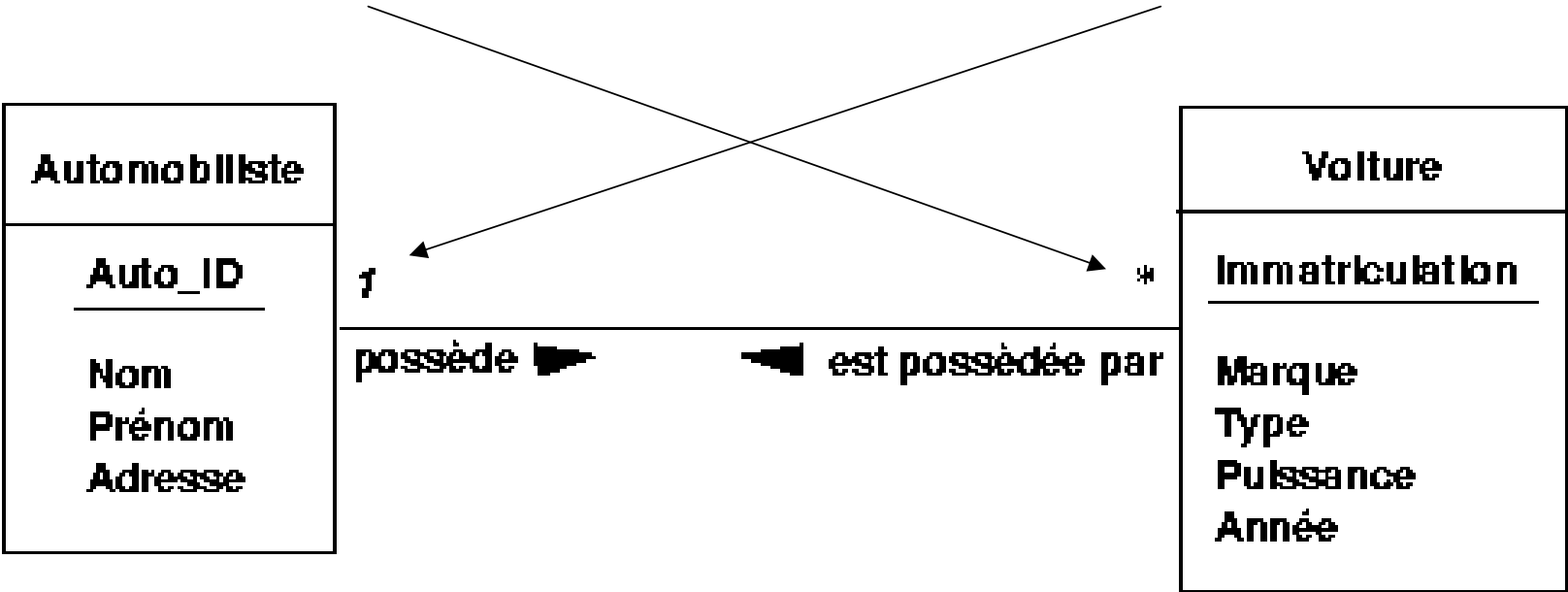
*Une voiture a un et un  
seul propriétaire*



# Modélisation UML

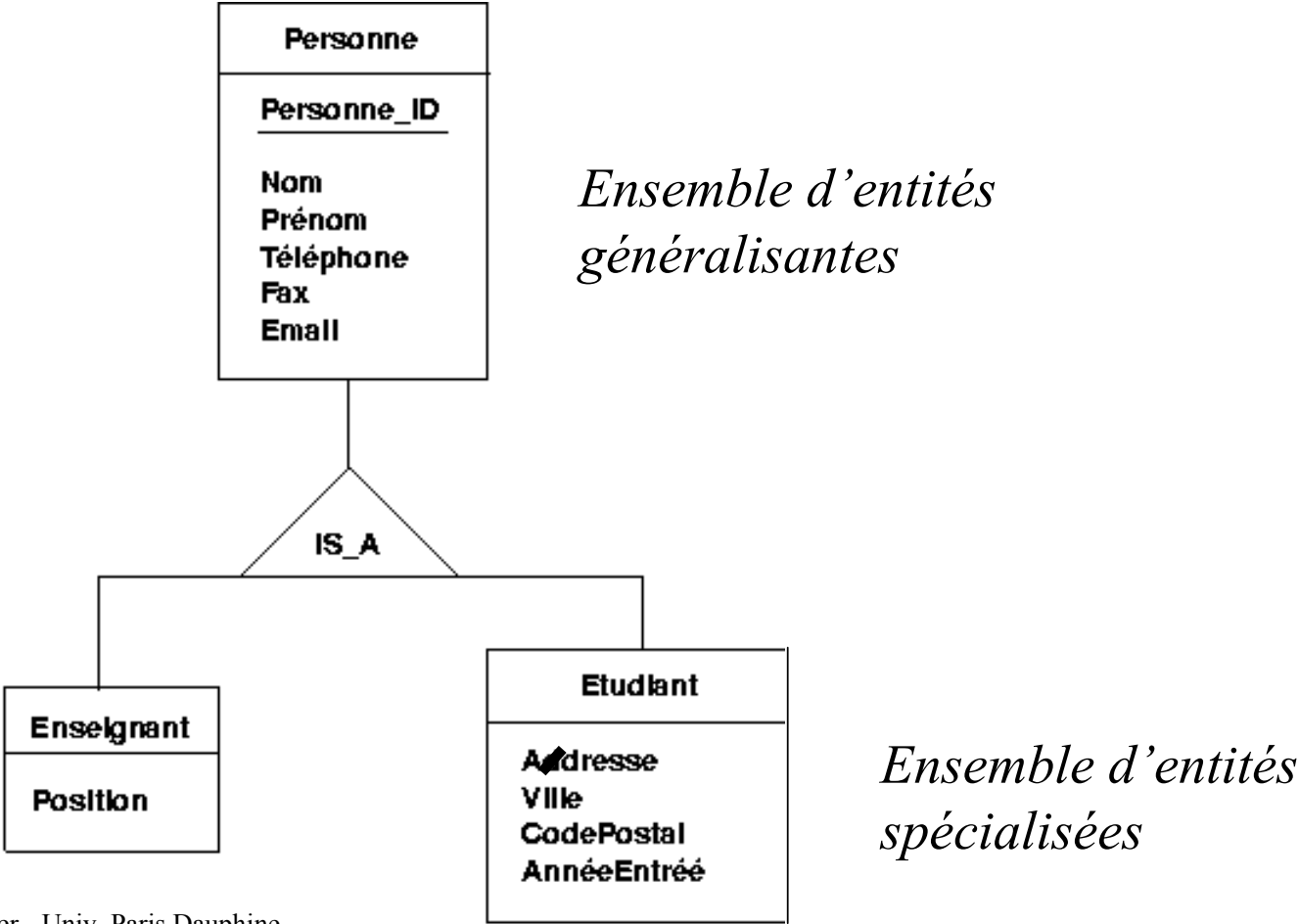
*Un automobiliste possède  
entre zéro et N voitures*

*Une voiture a un et un  
seul propriétaire*

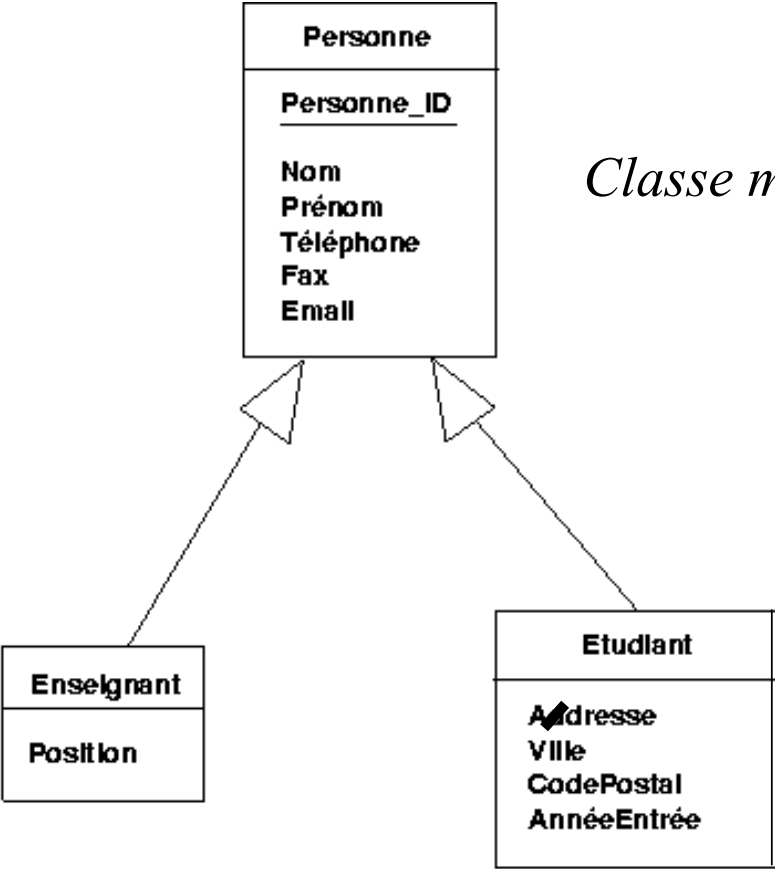


*Attention : petite liberté prise avec UML, les attributs soulignés ici ne correspondent pas à des attributs dérivés mais aux identificateurs (pour ne pas les oublier lors du passage au relationnel!!)*

# Généralisation/Spécialisation (E/A - Merise)



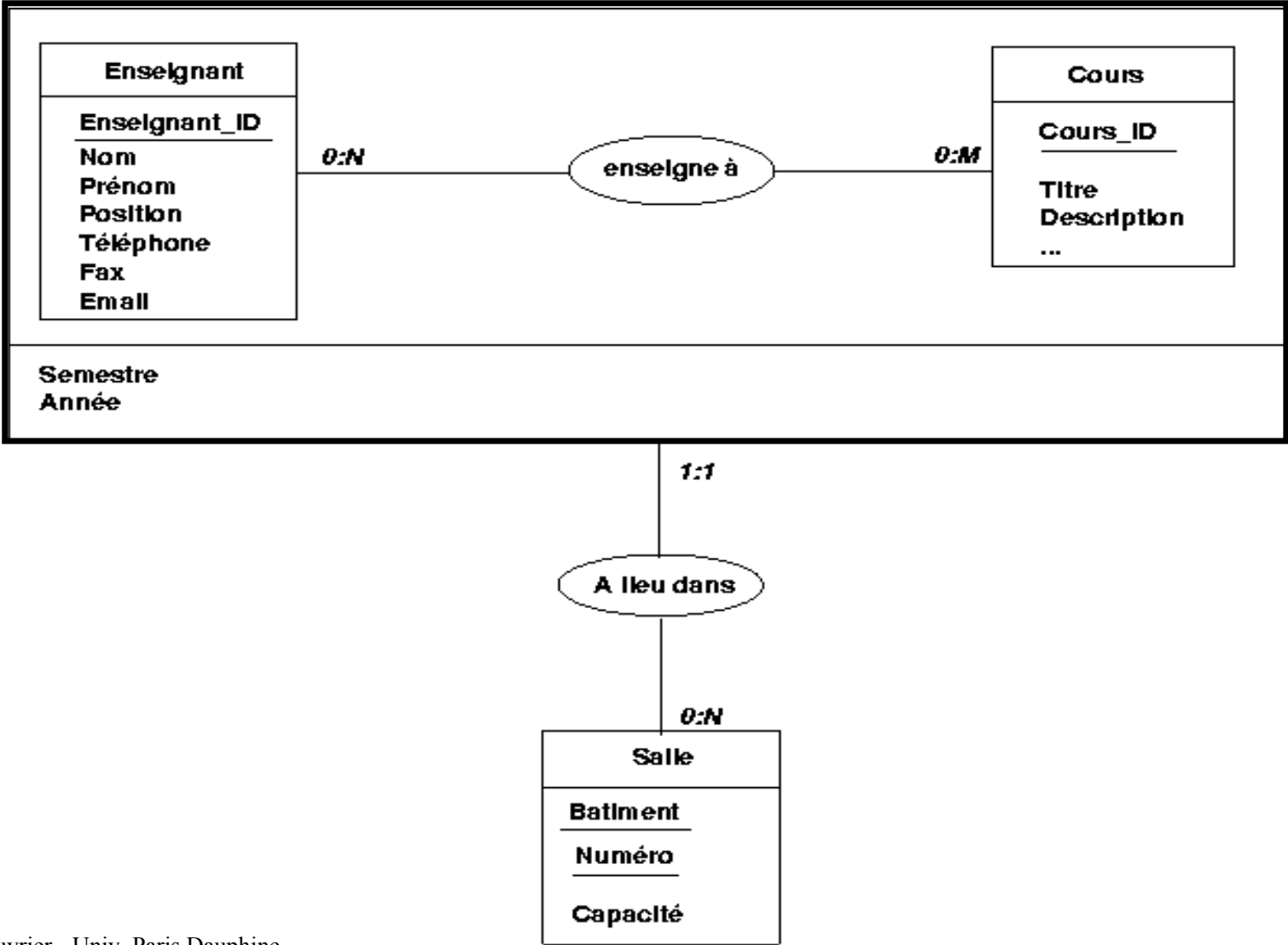
# Héritage (UML)



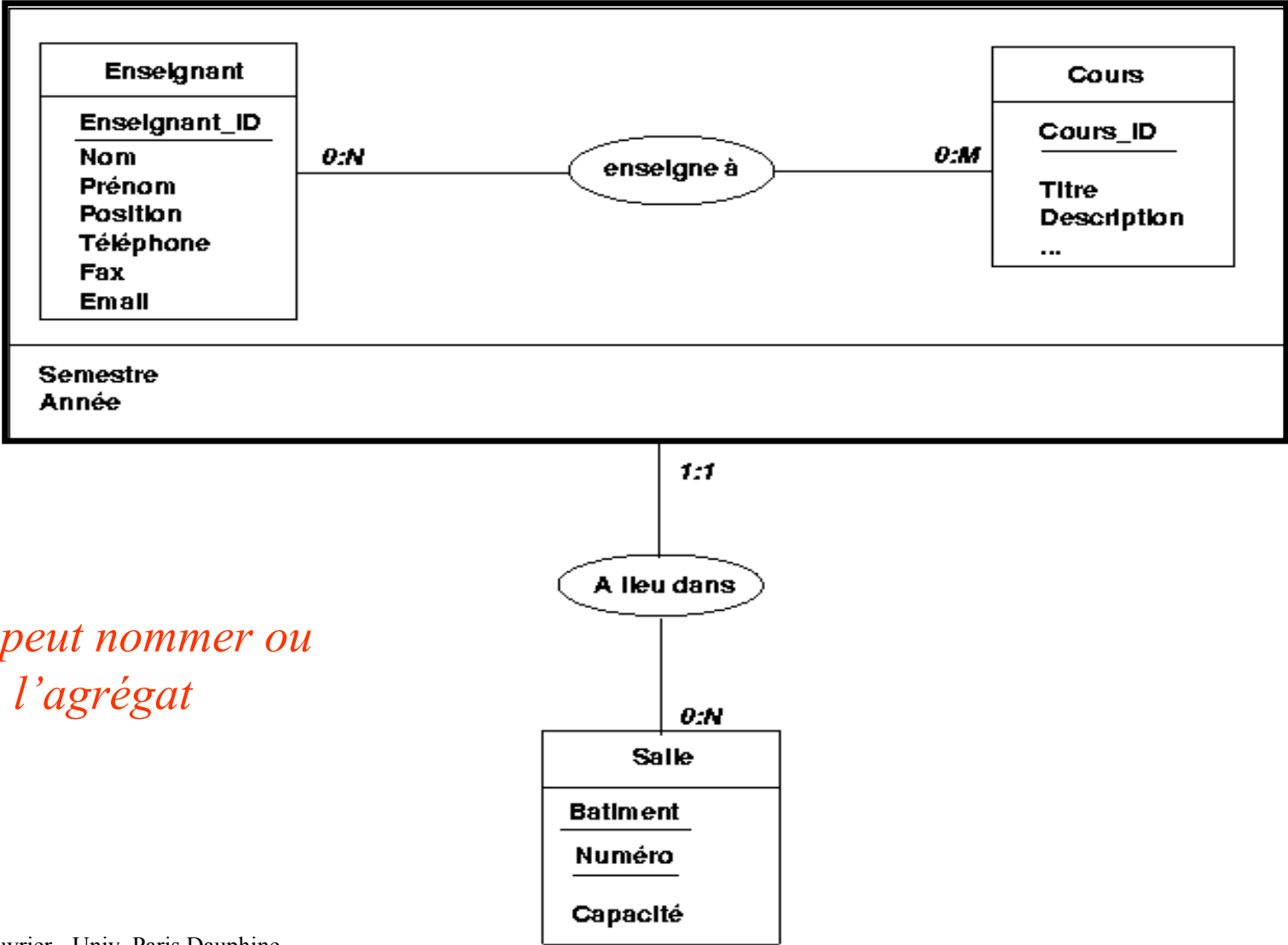
*Classe mère / Sur-classe*

*Classes dérivées ou filles / sous-classes*

# Agrégat (E/A - Merise)



# Agrégat (E/A - Merise)



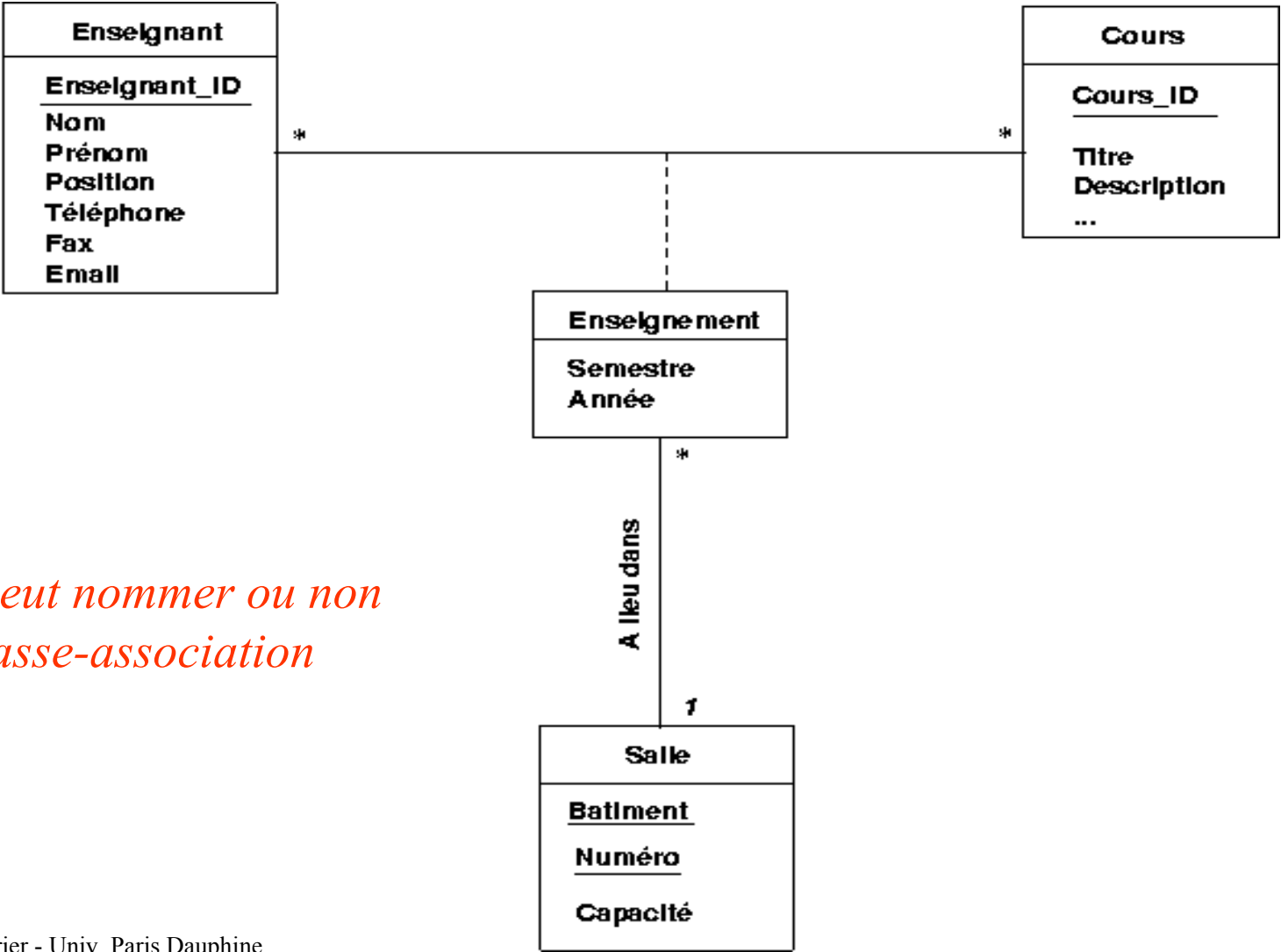
*On peut nommer ou non l'agrégat*

# Classe-Association (UML)



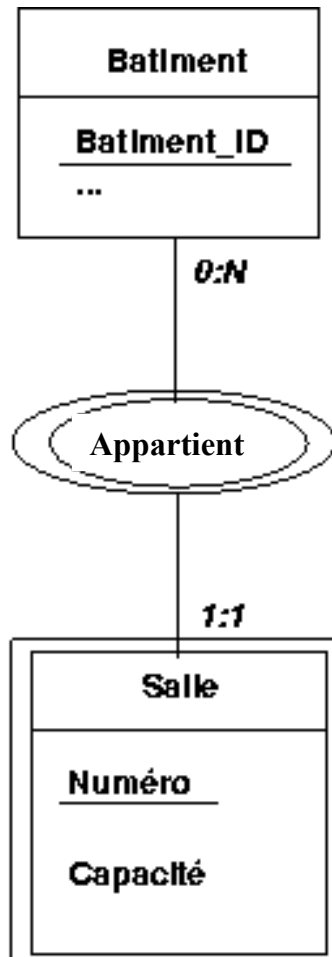


# Classe-Association (UML)



*On peut nommer ou non  
la classe-association*

# Entité Faible (E/A - Merise)

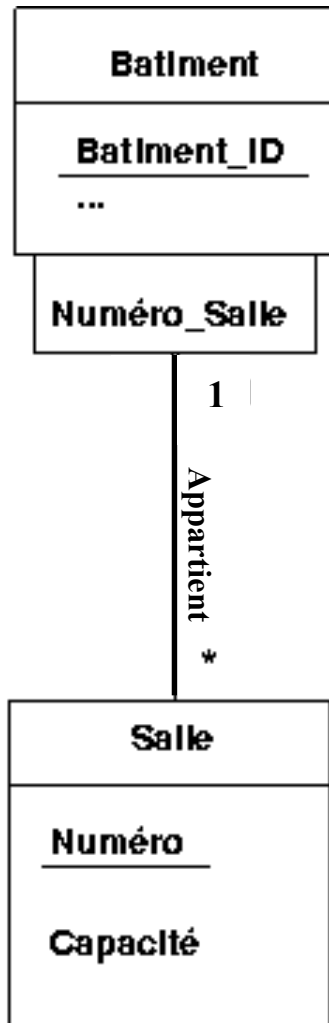


*Chaque salle a un  
numéro unique dans un  
bâtiment donné*

*Ex. Salle 1 du bâtiment A  
et Salle 1 du bâtiment C*

*Pour distinguer une salle  
d'une autre, il faut  
connaître le bâtiment  
auquel elle est rattachée*

# Association qualifiée (UML)

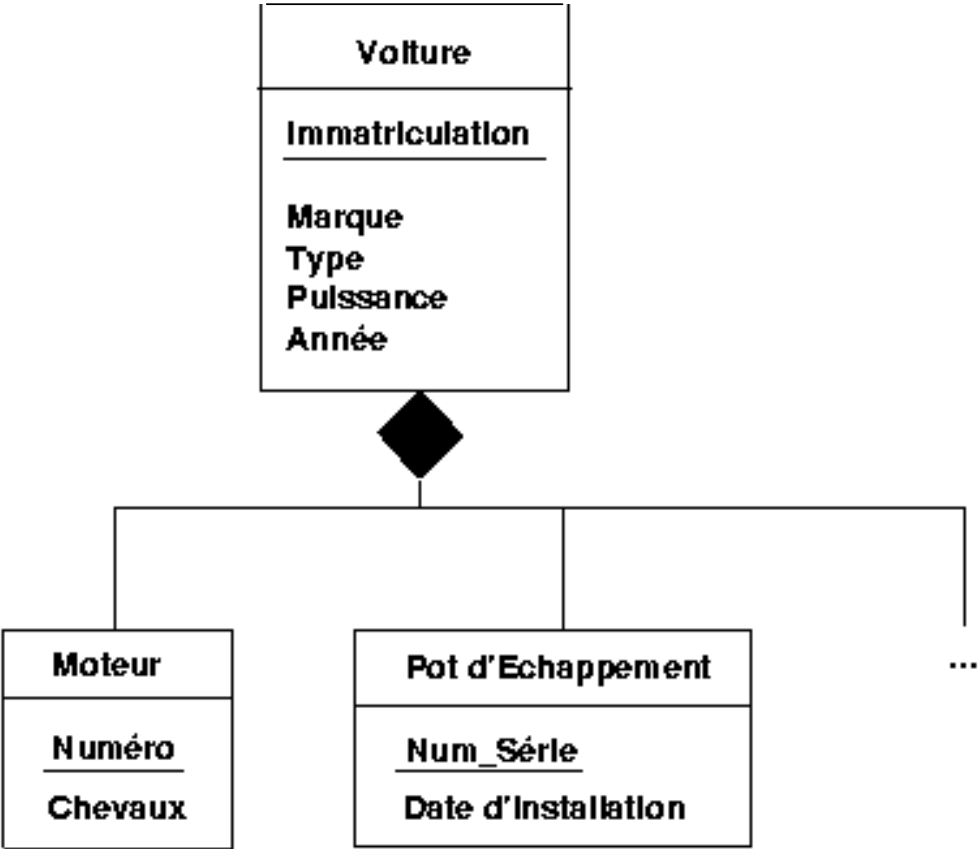


*Chaque salle a un  
numéro unique dans un  
bâtiment donné*

*Ex. Salle 1 du bâtiment A  
et Salle 1 du bâtiment C*

*Pour distinguer une salle  
d'une autre, il faut  
connaître le bâtiment  
auquel elle est rattachée*

# Composition (UML)



# Contraintes

## Contraintes d'intégrité :

toutes règles implicites ou explicites que doivent suivre les données [Gar99]

- **Contraintes d'entité**: toute entité doit posséder un identificateur
- **Contraintes de domaine** : les valeurs de certains attributs doivent être prises dans un ensemble donné
- **Contraintes d'unicité** : une valeur d'attribut ne peut pas être affectée deux fois à deux entités différentes
- **Contraintes générales** : règle permettant de conserver la cohérence de la base de manière générale

# Exemples de contraintes

- **Contraintes de domaine :**

"La fonction d'un enseignant à l'Université prend sa valeur dans l'ensemble {vacataire, moniteur, ATER, MCF, Prof., PRAG, PAST}."

- **Contraintes d'unicité :**

"Un département, identifié par son numéro, a un nom unique (il n'y a pas deux départements de même nom)."

- **Contraintes générales :**

"Un même examen ne peut pas avoir lieu dans deux salles différentes à la même date et à la même heure. "

# Dépendances fonctionnelles

Un attribut (ou un groupe d'attributs)  $Y$  **dépend fonctionnellement** d'un attribut (ou groupe d'attributs)  $X$  si :

étant donné une valeur de  $X$ , il lui correspond une valeur unique de  $Y$  ( $\forall$  l'instant considéré)

$X \rightarrow Y$  :  $Y$  **dépend fonctionnellement de  $X$**   
ou  $X$  **détermine  $Y$**

Déclaration des dépendances **au niveau du schéma conceptuel**

# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>



# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

Marque, Type, Puissance, Année → Immatriculation

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

## Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

## Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

Type → Marque

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

## Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

Type → Marque *Ex. Le type "Twingo" sera toujours associé, dans la base de données, à la marque "Renault".*

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u>Immatriculation</u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

Type → Marque *Ex. Le type "Twingo" sera toujours associé, dans la base de données, à la marque "Renault".*

<b>Enseignant</b>
<u>Enseignant_ID</u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

EnseignantID → Nom, Prénom, Position ...

Nom, Prénom, Position, ... → Enseignant\_ID

# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u><b>Immatriculation</b></u>
<b>Marque</b>
<b>Type</b>
<b>Puissance</b>
<b>Année</b>

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

Type → Marque *Ex. Le type "Twingo" sera toujours associé, dans la base de données, à la marque "Renault".*

<b>Enseignant</b>
<u><b>Enseignant_ID</b></u>
<b>Nom</b>
<b>Prénom</b>
<b>Position</b>
<b>Téléphone</b>
<b>Fax</b>
<b>Email</b>

EnseignantID → Nom, Prénom, Position ...

Nom, Prénom, Position, ... → Enseignant\_ID

*Si un numéro de téléphone est associé à un seul enseignant :*

Telephone → Enseignant\_ID

# Chap III - Modèle relationnel

- **Domaine** : ensemble de valeurs caractérisé par un nom
- **Relation** : sous-ensemble du produit cartésien d'une liste de domaines caractérisé par **un nom unique**
  - représentée sous forme de **table à deux dimensions**
  - colonne = un domaine du produit cartésien
  - **un même domaine peut apparaître plusieurs fois**
  - ensemble de **nuplets sans doublon**
- **Attribut** : une colonne dans une relation
  - caractérisé par un nom et dont **les valeurs appartiennent à un domaine**
  - **les valeurs sont atomiques**
- **Nuplet** : une ligne d'une relation
  - correspondant à un enregistrement, c-à-d **une entité/instance de classe**
  - **les nuplets d'une relation sont tous différents**



# Exemple de relation

Nom d'attribut



NSS	Nom	Prénom	Fonction
273...	<i>Manouvrier</i>	<i>Maude</i>	<i>MCF</i>
...			
...			

*La relation Enseignant*

**Nuplets** ou *tuples*

# Instances et schéma

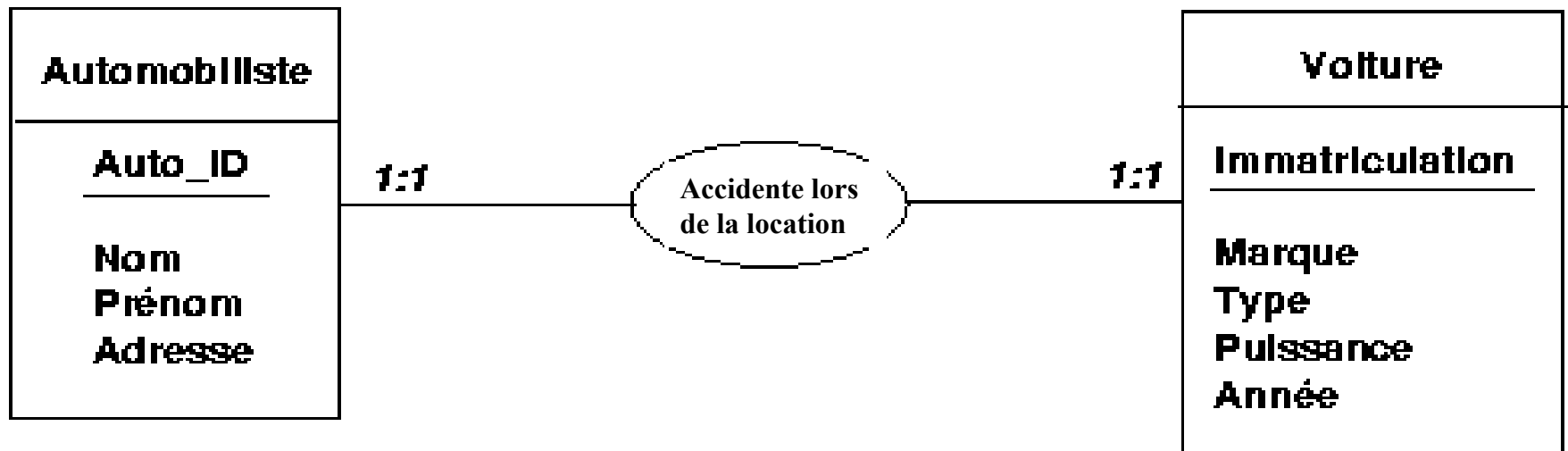
- **Instances de base de données :**  
les nuplets (les valeurs) contenus dans la base à un instant donné
- **Schéma de base de données :**
  - ensemble de **schémas de relation**
  - modélisation logique de la base de données à l'aide du modèle relationnel
- **Schéma de relation :**  
liste d'attributs et leurs domaines

# Passage au relationnel

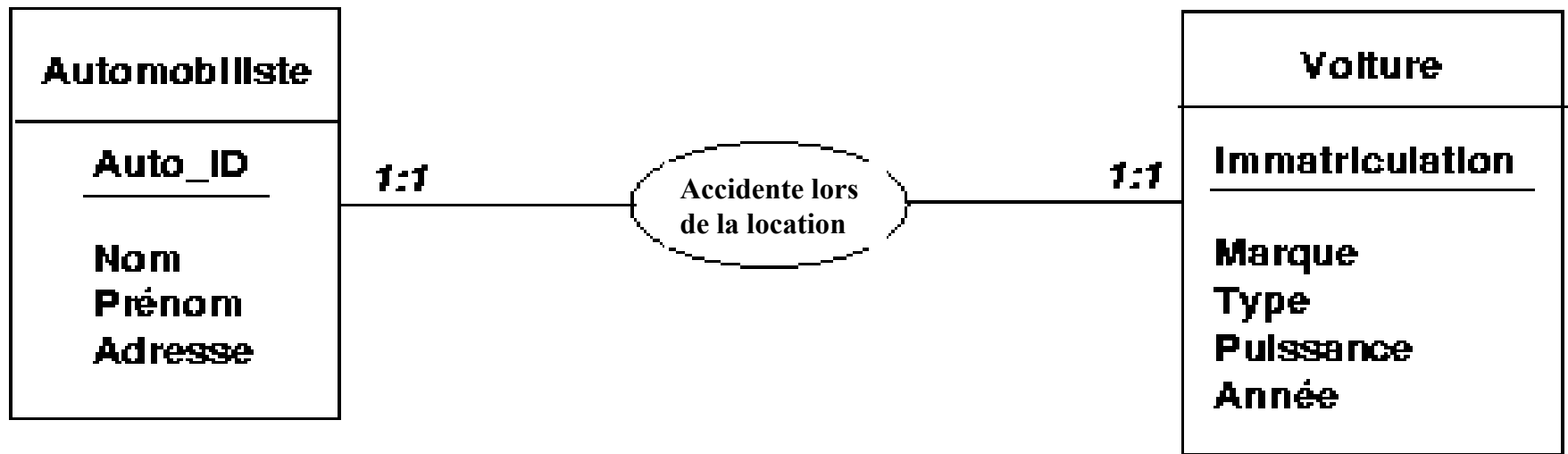
## Transformation des ensembles d'entités :

- **chaque ensemble d'entités/classes  $E \Rightarrow$** 
  - une relation  $R$  dont le schéma est celui de l'ensemble d'entités/classe
  - l'identificateur de  $E$  devient la clé de  $R$
- **chaque ensemble d'entités faibles/association qualifiée  $E \Rightarrow$** 
  - une relation  $R$  qui comprend tous les attributs de  $E$  +  
l'identificateur de l'ensemble d'entités fortes/classe associé(e)
- **généralisation-spécialisation/héritage  $\Rightarrow$** 
  - l'ensemble d'entités généralisante/classe mère  $E \Rightarrow$  une relation  $R$
  - chaque ensemble d'entités  $E_i$  spécialisé/classe fille  
 $\Rightarrow$  une relation  $R_i$  dans laquelle l'identifiant est de même domaine que l'identifiant de  $E$

# Transformation des ensembles d'associations E/A



# Transformation des ensembles d'associations E/A

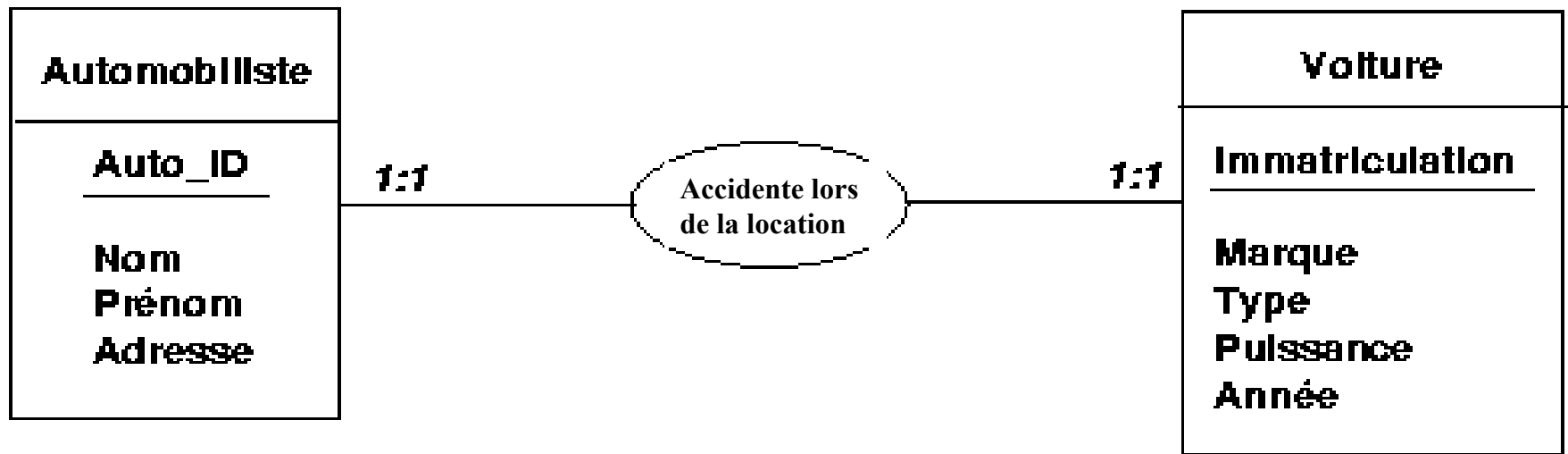


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse)

*Voiture* (Immatriculation, Marque, Type, Puissance, Année )

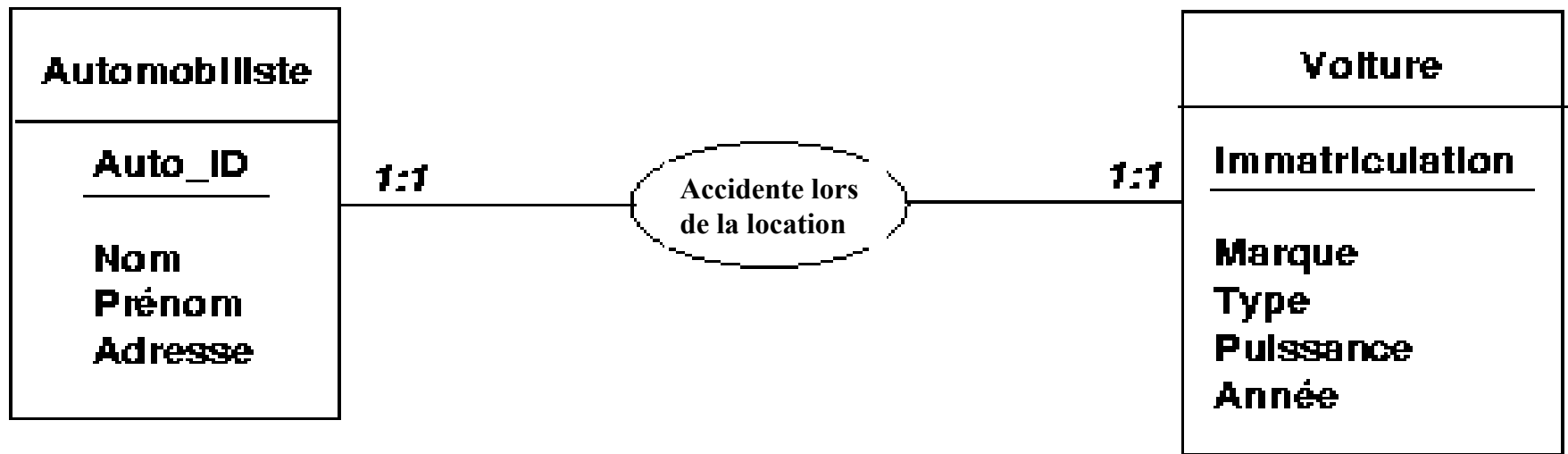
Comment faire le lien ?

# Transformation des ensembles d'associations E/A



*Accident* ( Auto\_ID, Nom, Prénom, Adresse, Immatriculation, Marque, Type, Puissance, Année )

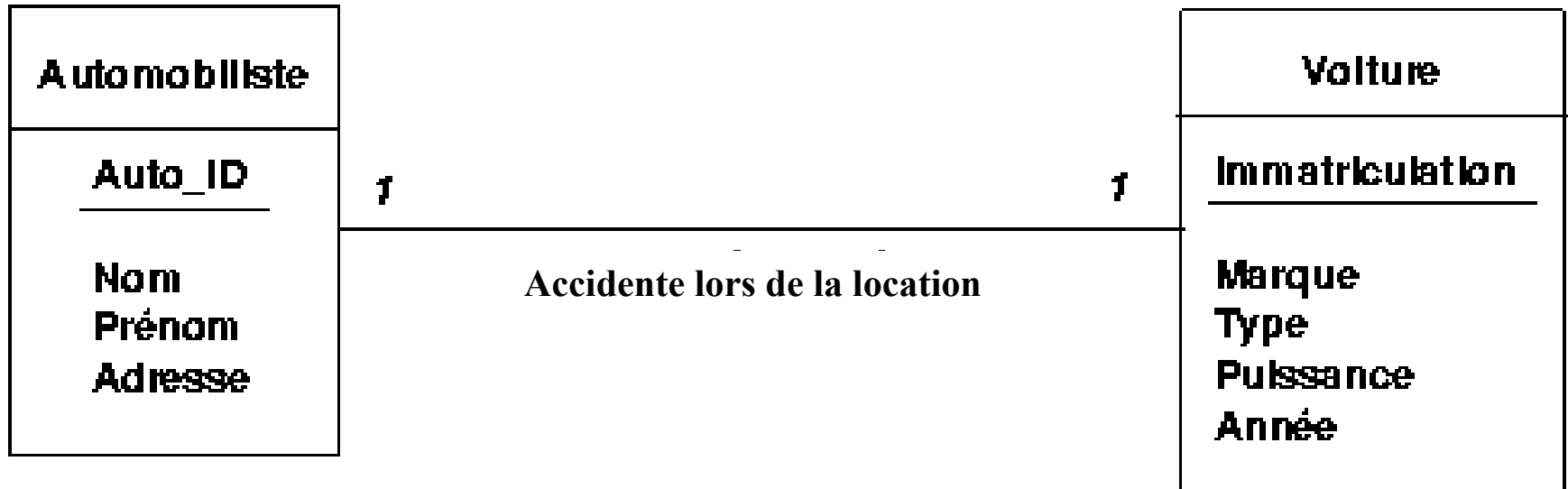
# Transformation des ensembles d'associations E/A



*Accident* ( Auto\_ID, Nom, Prénom, Adresse, Immatriculation, Marque, Type, Puissance, Année )

*On peut choisir l'un ou l'autre comme clé primaire*

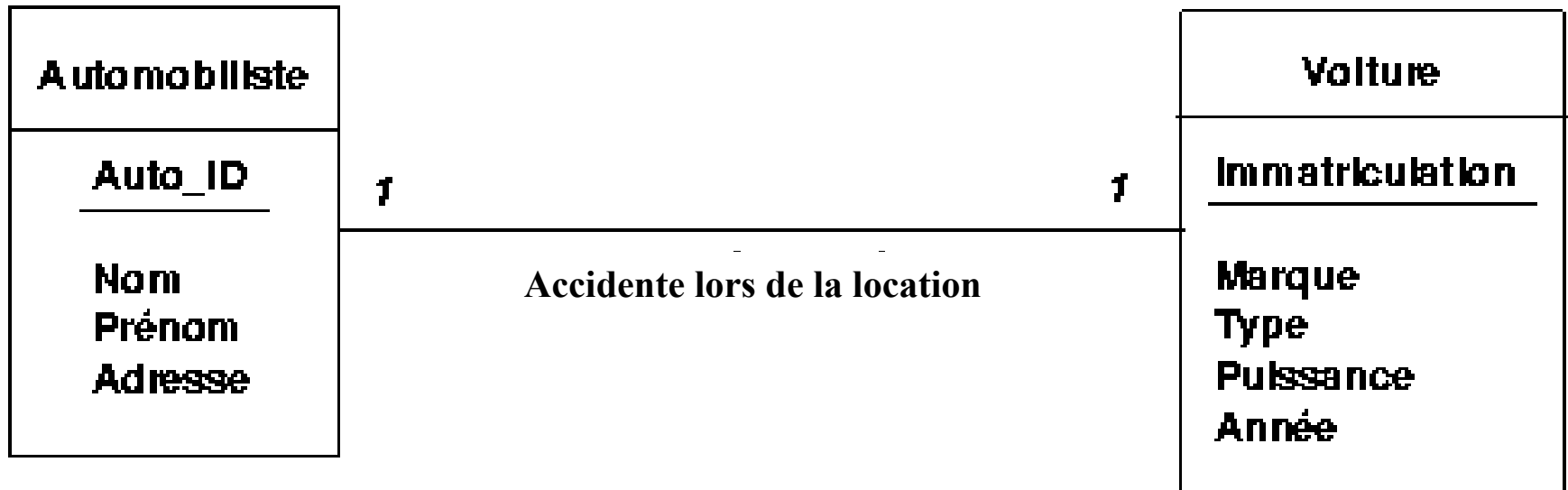
# Transformation des ensembles d'associations UML



*Accidente* ( Auto\_ID, Nom, Prénom, Adresse, Immatriculation, Marque, Type, Puissance, Année )



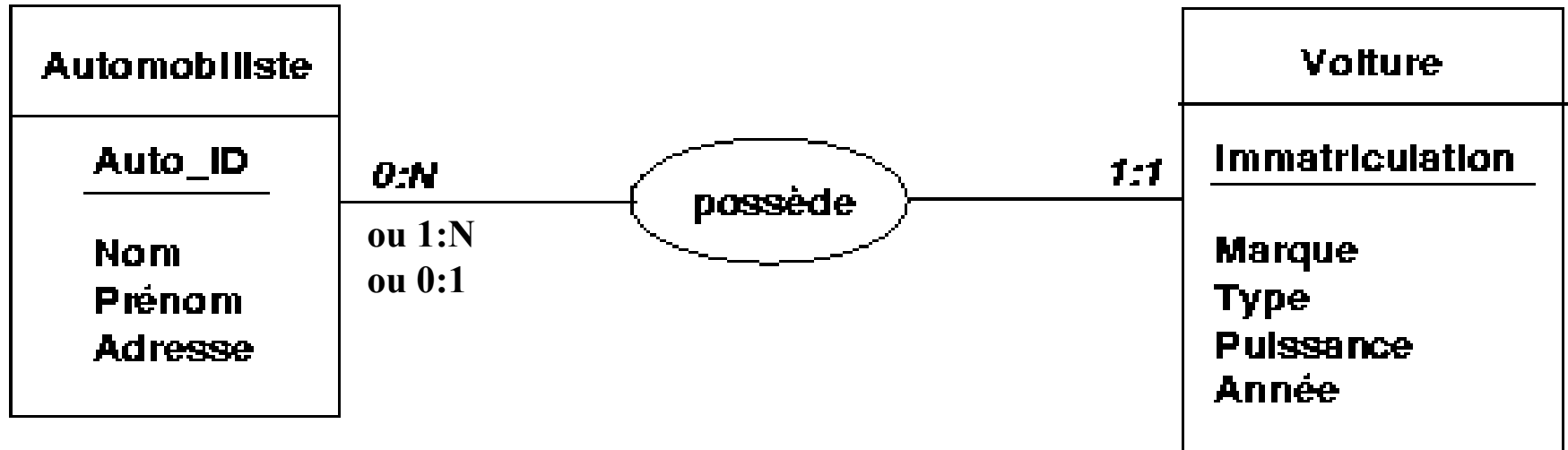
# Transformation des ensembles d'associations UML



*Accidente* ( Auto\_ID, Nom, Prénom, Adresse, Immatriculation, Marque, Type, Puissance, Année )

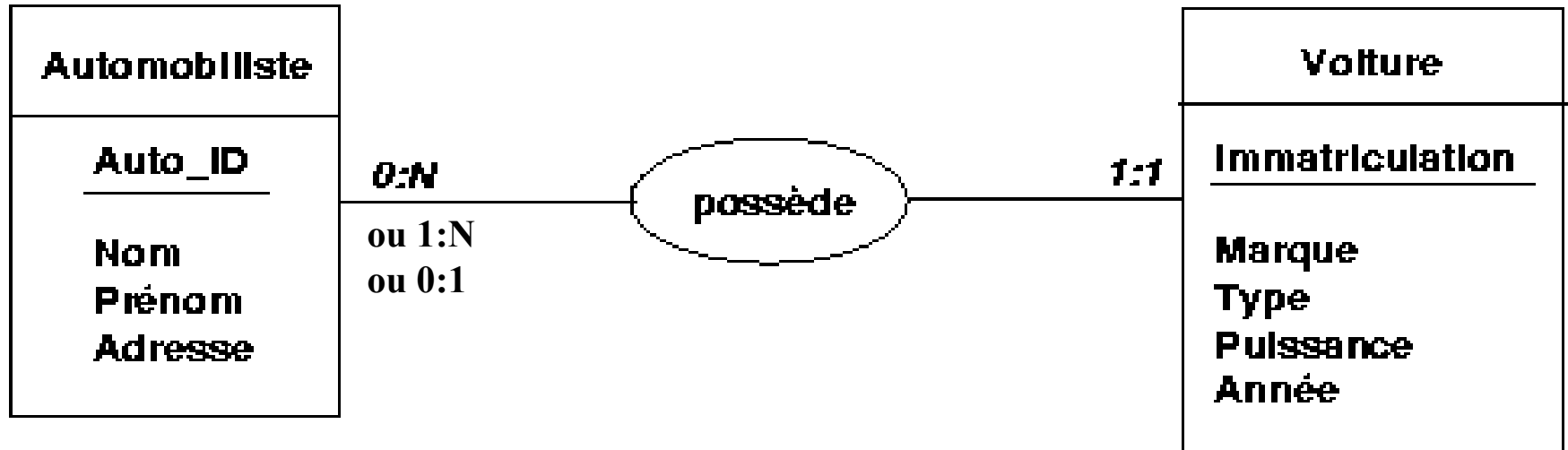
*On peut choisir l'un ou l'autre comme clé primaire*

# Transformation des ensembles d'associations E/A



,

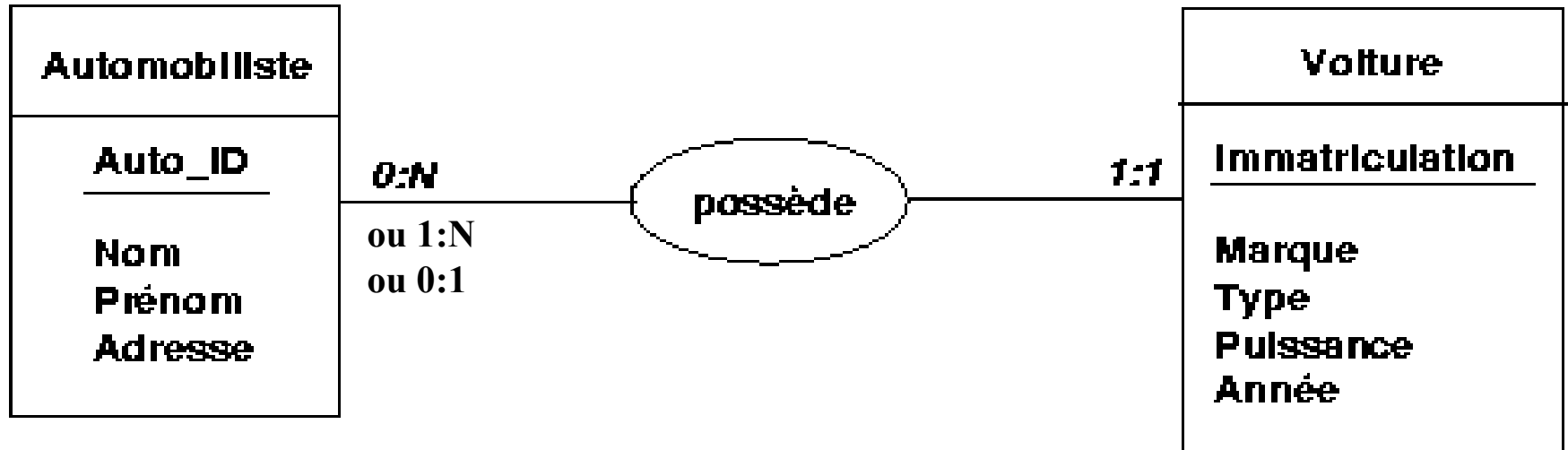
# Transformation des ensembles d'associations E/A



*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

,

# Transformation des ensembles d'associations E/A

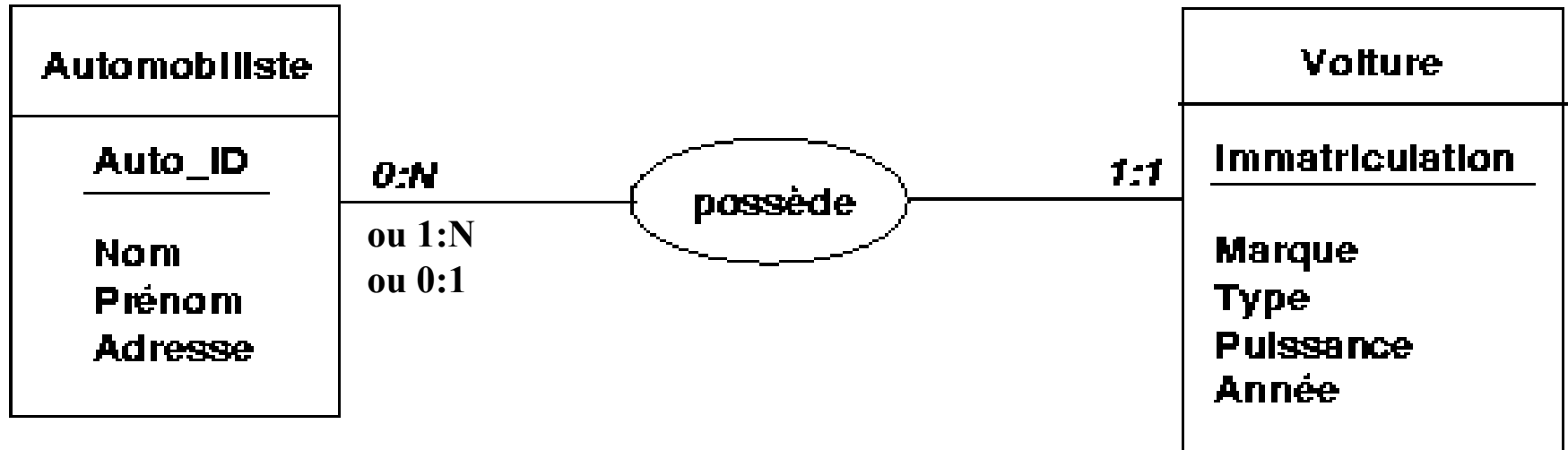


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

,

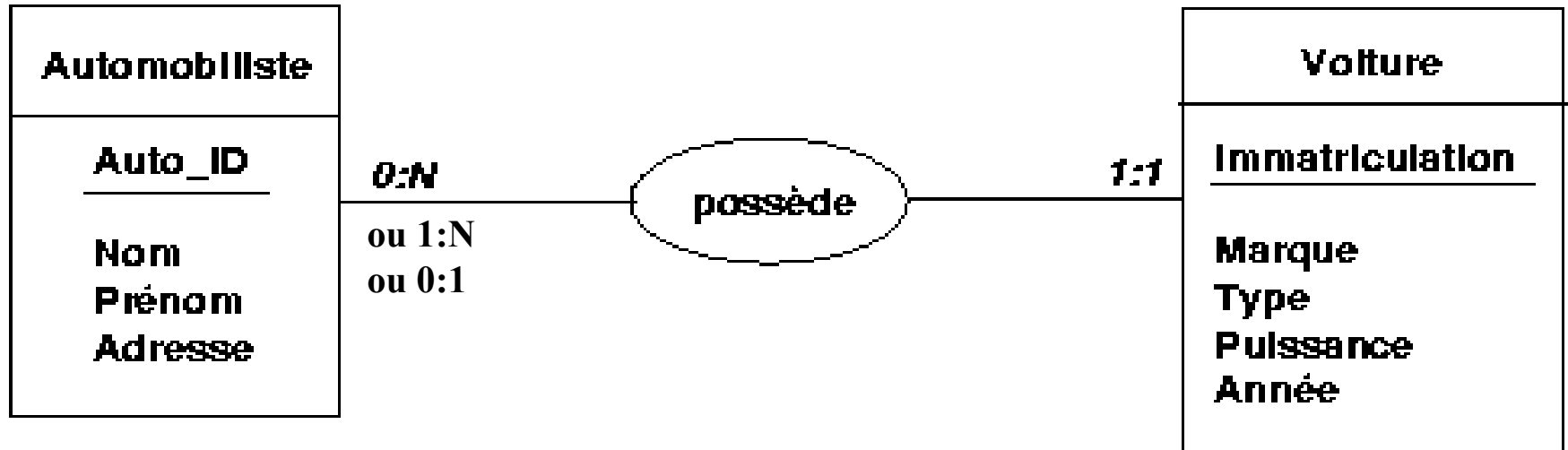
# Transformation des ensembles d'associations E/A



*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année,  
#Auto\_ID )

# Transformation des ensembles d'associations E/A

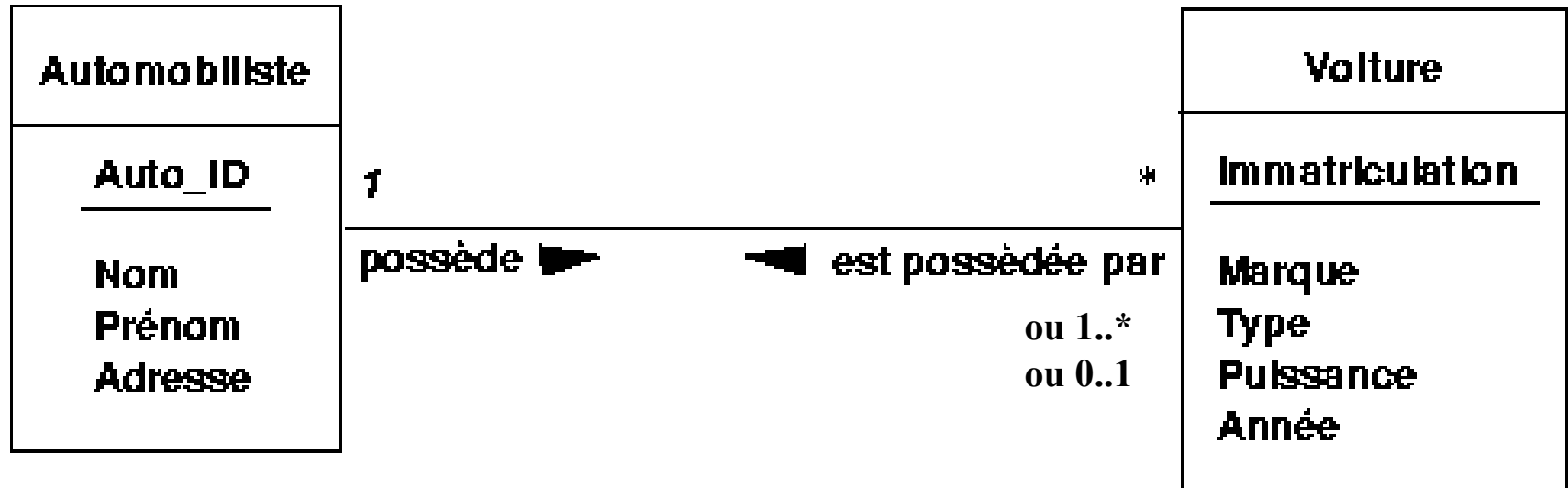


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année,  
#Auto\_ID )

**NB** : #Auto\_ID fait référence à Auto\_ID de Automobiliste

# Transformation des ensembles d'associations UML

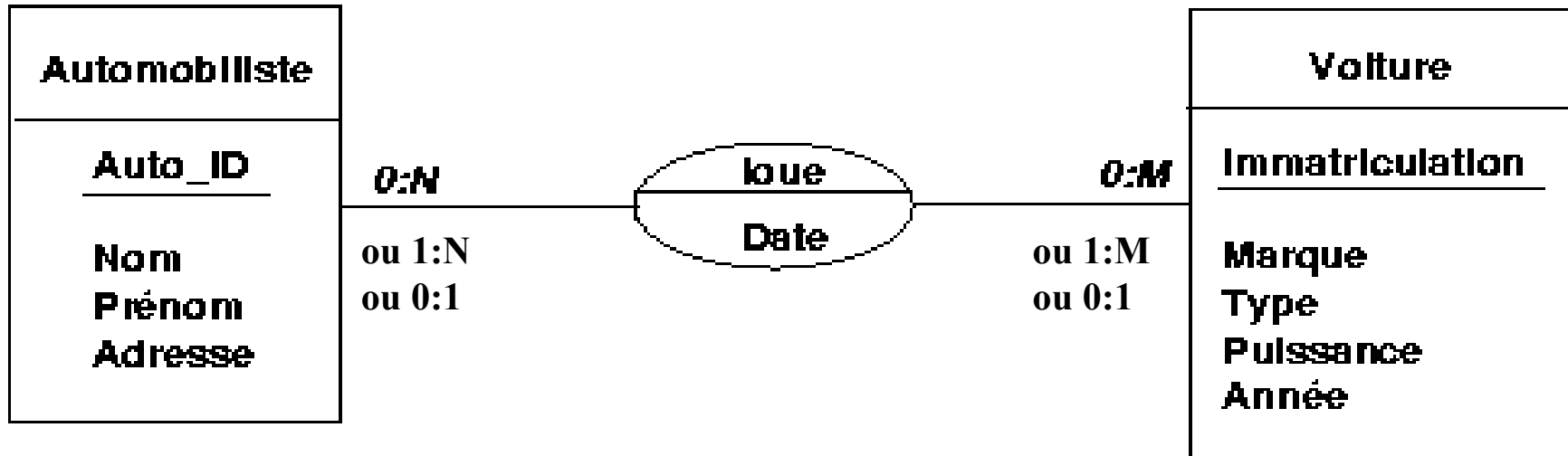


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année, **#Auto\_ID** )

**NB** : #Auto\_ID fait référence à Auto\_ID de Automobiliste

# Transformation des ensembles d'associations E/A

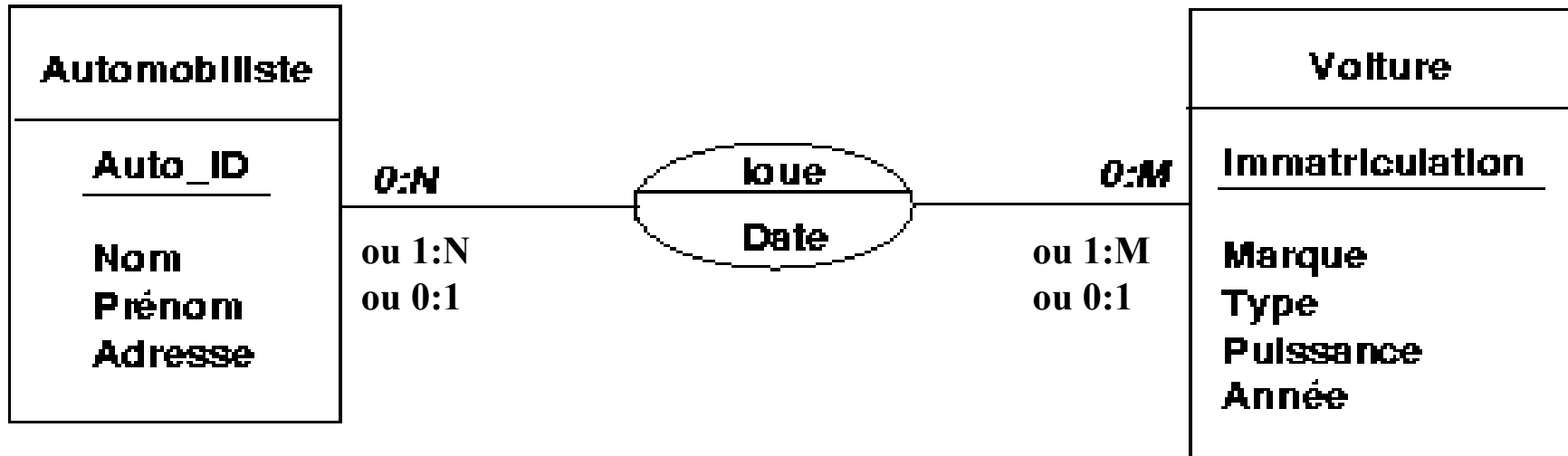


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )



# Transformation des ensembles d'associations E/A



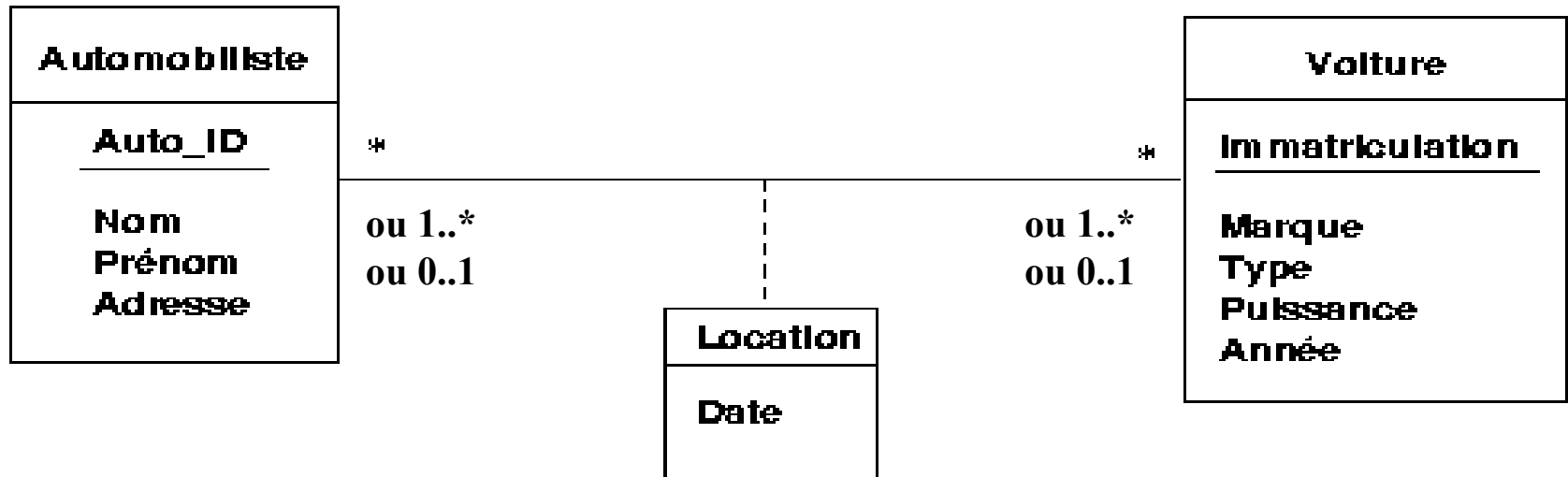
*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Location* ( #Auto\_ID, #Immatriculation, Date ) ou

*Location* ( Loc\_ID, #Auto\_ID, #Immatriculation, Date )

# Transformation des ensembles d'associations UML



*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

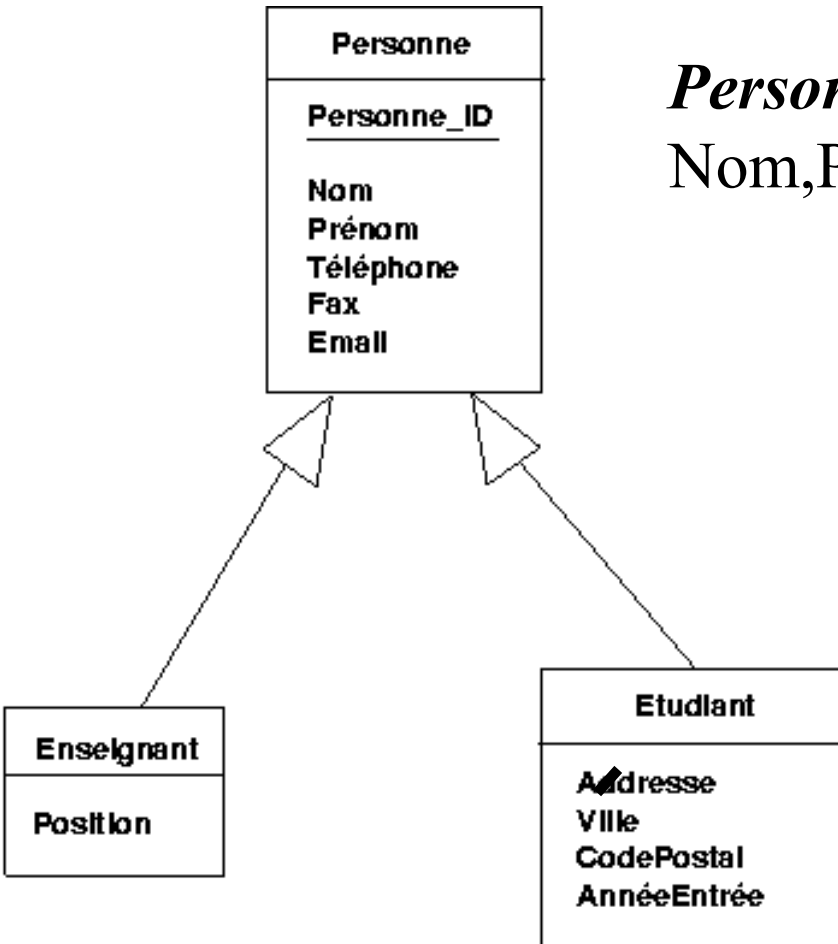
*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Location* ( #Auto\_ID, #Immatriculation, Date ) ou

*Location* ( Loc\_ID, #Auto\_ID, #Immatriculation, Date )

# Transformation des concepts

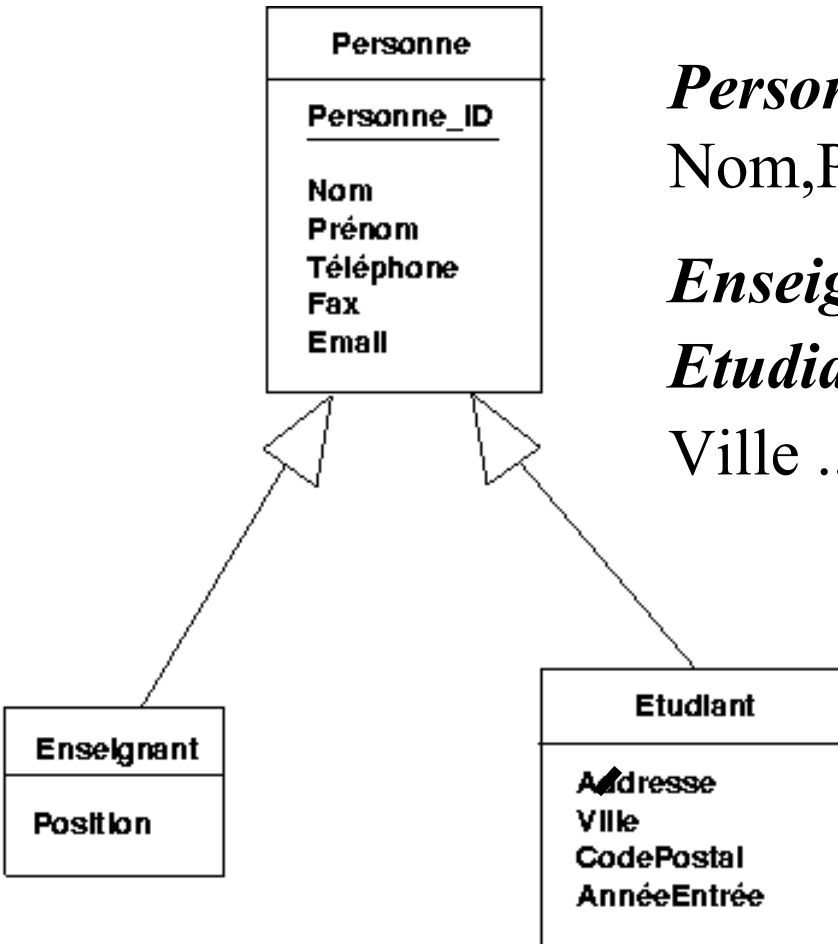
## Généralisation-Spécialisation / Héritage



*Personne* ( Personne\_ID,  
Nom, Prénom, Téléphone ... )

# Transformation des concepts

## Généralisation-Spécialisation / Héritage



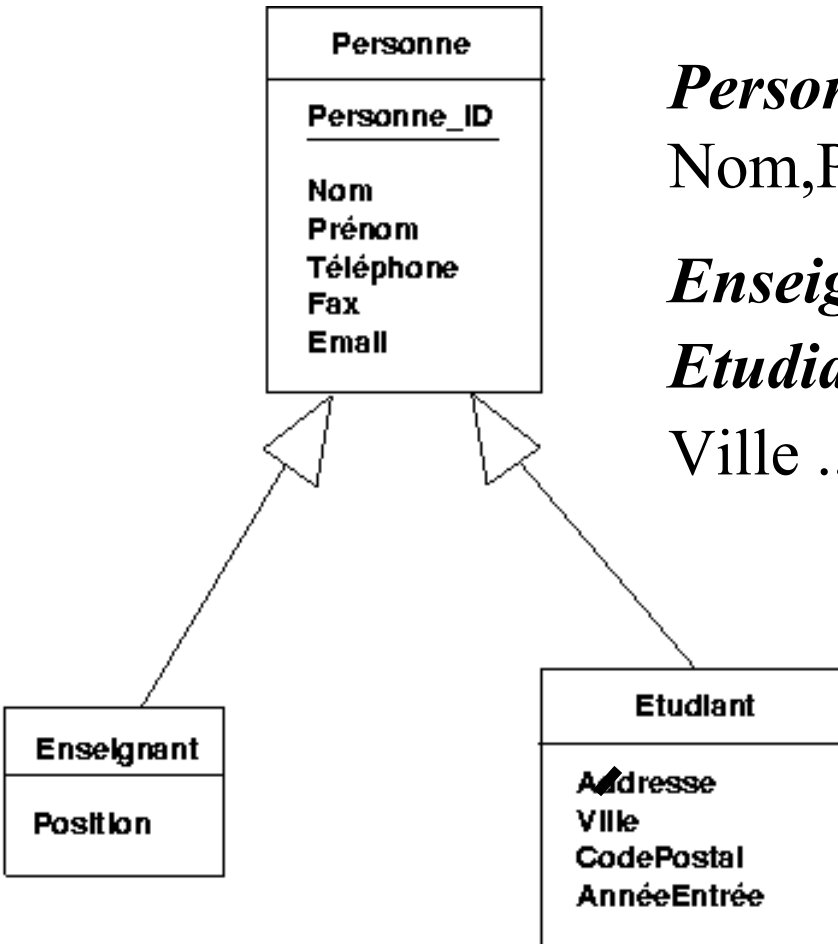
*Personne* ( Personne\_ID,  
Nom, Prénom, Téléphone ... )

*Enseignant* ( #Personne\_ID, Position )

*Etudiant* ( #Personne\_ID, Adresse,  
Ville ... )

# Transformation des concepts

## Généralisation-Spécialisation / Héritage



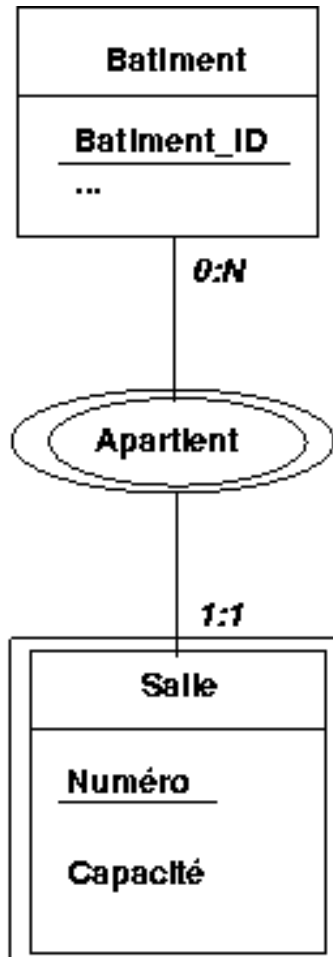
*Personne* ( Personne\_ID,  
Nom, Prénom, Téléphone ... )

*Enseignant* ( #Personne\_ID, Position )

*Etudiant* ( #Personne\_ID, Adresse,  
Ville ... )

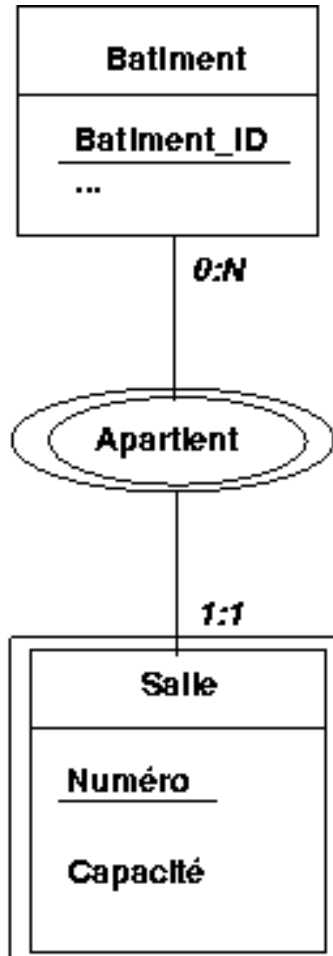
**NB :** #*Personne\_ID* dans  
*Enseignant* et *Etudiant* font  
référence à *Personne\_ID* dans  
*Personne*

# Transformation des entités faibles E/A



*Bâtiment* ( Bâtiment\_ID, ... )

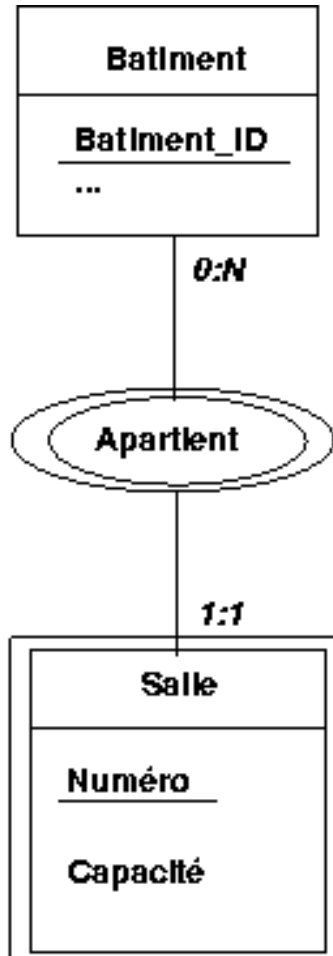
# Transformation des entités faibles E/A



*Bâtiment* ( Bâtiment\_ID, ... )

*Salle* ( Numéro, #Bâtiment\_ID, Capacité )

# Transformation des entités faibles E/A



***Bâtiment** ( Bâtiment\_ID, ... )*

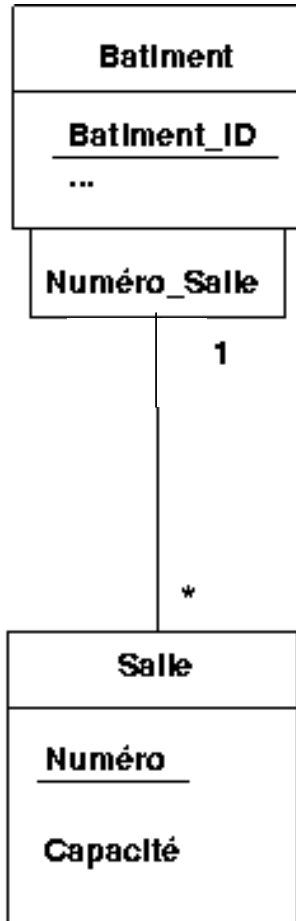
***Salle** ( Numéro, #Bâtiment\_ID, Capacité)*

**NB :** *Une salle est identifiée par le couple (Numéro, #Bâtiment\_ID)*

*#Bâtiment\_ID fait référence à Bâtiment\_ID de Bâtiment*



# Transformation des associations qualifiées UML



*Bâtiment* ( Bâtiment\_ID, ... )

*Salle* ( Numéro, #Bâtiment\_ID, Capacité)

**NB :** *Une salle est identifiée par le couple (Numéro, #Bâtiment\_ID) ;*

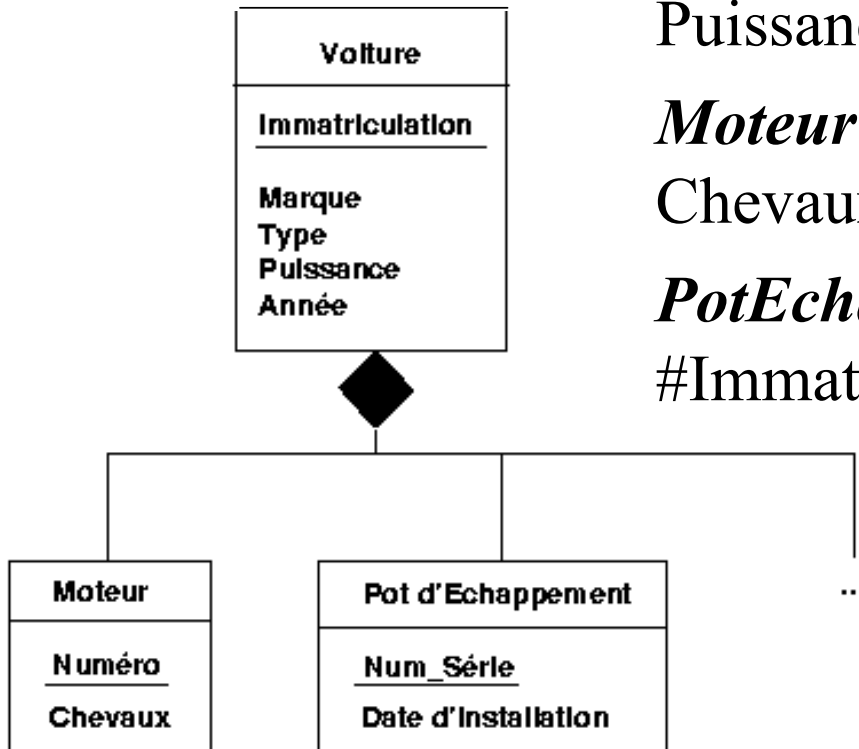
*#Bâtiment\_ID fait référence à Bâtiment\_ID de Bâtiment*

# Transformation de la composition UML

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Moteur* ( Numéro, #Immatriculation, Chevaux )

*PotEchappement* ( Num\_Série, #Immatriculation, DateInstallation )



# **Dépendances fonctionnelles**

**Ne pas oublier de définir les DF :**

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID → Nom, Prénom, Adresse

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )



# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

Immatriculation  $\rightarrow$  Auto\_ID

+ les Dépendances fonctionnelles de Voiture

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

Immatriculation  $\rightarrow$  Auto\_ID     Auto\_ID  ~~$\rightarrow$~~  Immatriculation

+ les Dépendances fonctionnelles de Voiture

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

Immatriculation  $\rightarrow$  Auto\_ID      Auto\_ID  ~~$\rightarrow$~~  Immatriculation

+ les Dépendances fonctionnelles de Voiture

***Location*** ( Auto\_ID, Immatriculation, Date )

# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

***Accidente*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID  $\rightarrow$  Nom, Prénom, Adresse

Immatriculation  $\rightarrow$  Marque, Type, Puissance, Année

Type  $\rightarrow$  Marque

Auto\_ID  $\rightarrow$  Immatriculation et Immatriculation  $\rightarrow$  Auto\_ID

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

Immatriculation  $\rightarrow$  Auto\_ID      Auto\_ID  ~~$\rightarrow$~~  Immatriculation

+ les Dépendances fonctionnelles de Voiture

***Location*** ( Auto\_ID, Immatriculation, Date )

Pas de dépendance non triviale

# Intégrité structurelle

- **Unicité des clés**

- **ensemble minimal d'attributs** dont la connaissance des valeurs permet d'identifier un nuplet unique de la relation considérée
- $R$  a pour clé  $K$  si :  $\forall t_1, t_2$  nuplets d'une instance de  $R$   
$$t_1.K \neq t_2.K$$

- **Contraintes de référence**

- **contrainte référentielle** : contrainte d'intégrité portant sur une relation  $R$  qui consiste à imposer que la valeur d'un groupe d'attributs apparaissent comme valeur de clé dans une autre relation
- **clé étrangère** : un groupe d'attributs qui doit apparaître comme clé dans une autre relation

# **Clé /Clé minimale /Surclé**

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs



# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

*Clé primaire* : Immatriculation

*Surclé* : (Immatriculation, Marque, Puissance, Type, Année, Auto\_ID)

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

*Clé primaire* : Immatriculation

*Surclé* : (Immatriculation, Marque, Puissance, Type, Année, Auto\_ID)

***Location*** ( Auto\_ID, Immatriculation, Date )

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

*Clé primaire* : Immatriculation

*Surclé* : (Immatriculation, Marque, Puissance, Type, Année, Auto\_ID)

***Location*** ( Auto\_ID, Immatriculation, Date )

*Clé primaire* : (Auto\_ID, Immatriculation, Date)

# Intégrité structurelle

- **Valeur nulle**

- valeur conventionnelle introduite dans une relation pour représenter une information inconnue ou inapplicable
- tout attribut peut prendre une valeur nulle **excepté les attributs de la clé primaire** (contrainte d'entité)

- **Contraintes de domaine**

contrainte d'intégrité qui impose qu'une colonne d'une relation doit comporter des valeurs vérifiant une assertion logique

# Langages d'interrogation

- **Algèbre relationnelle**  
a inspiré le langage SQL
- **Calcul relationnel à variable nuplet**  
a inspiré le langage QUEL du SGBD Ingres
- **Calcul relationnel à variable domaine**  
a inspiré le langage QBE (*Query By Example*) d'IBM
- **SQL (*Structured Query Language*)**

Ces langages sont équivalents : ils permettent de désigner les mêmes ensembles de données

# Chap IV - Algèbre relationnelle

## Opérations unaires :

- **sélection** des nuplets satisfaisant un certain prédicat

**Etudiant**(Etudiant\_ID, Nom, Prénom, Rue, Ville, Code-Postal, Téléphone, Fax, Email, NumAnnées)

$\sigma_{(Ville='Paris')}(Etudiant)$

$\sigma_{(Ville='Paris') \wedge (NumAnnées \geq 2)}(Etudiant)$

- **projection** : élimination de certains attributs d'une relation

$\Pi_{Nom, Prénom}(Etudiant)$

$\Pi_{Nom, Prénom}(\sigma_{(Ville='Paris')}(Etudiant))$

# Exemples de résultats d'opérations unaires

## Relation *Enseignant*

L..	enseignant_id (...)	departement_id ...	nom (varchar)	prenom ...	grade (...)	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
3	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr
4	4	1	LIMAM	Medhi	ATER			
5	5	5	MyTaylor	IsRich	Vacataire			
6	6	1	RIGAUX	Philippe	PROF			
7	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
8	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr

## Résultat de la sélection $\sigma_{(grade='MCF')}(Enseignant)$ :

L..	enseignant_id...	departement_id...	nom (varchar)	prenom...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr

## Résultat de la projection

### $\Pi_{Nom, Prénom}(Enseignant)$ :

L..	nom (varchar)	prenom...
1	MANOUVRIER	Maude
2	NAIJA	Yosr
3	BAHRI	Afef
4	LIMAM	Medhi
5	MyTaylor	IsRich
6	RIGAUX	Philippe
7	CHAKHAR	Salem
8	MURAT	Cécile

## Résultat de la requête

### $\Pi_{Nom, Prénom}(\sigma_{(grade='MCF')}(Enseignant))$ :

Ligne	nom (varchar)	prenom...
1	MANOUVRIER	Maude
2	MURAT	Cécile



# Opérations binaires

- **Union** : rassemblement des nuplets de 2 relations compatibles

*Enseignant*( Enseignant\_ID, Département\_ID, Nom, Prénom, Grade, Téléphone, Fax, Email )

$$\Pi_{\text{Nom, Prénom}}(\text{Etudiant}) \cup \Pi_{\text{Nom, Prénom}}(\text{Enseignant})$$

- **Différence** : des nuplets de 2 relations compatibles

$$\Pi_{\text{Nom, Prénom}}(\text{Enseignant}) - \Pi_{\text{Nom, Prénom}}(\text{Etudiant})$$

- **Produit cartésien** : combinaison des nuplets de 2 relations



*Département*(Département\_ID, Nom\_Département)

Produit cartésien de *Enseignant*  $\times$  *Département* a pour schéma :

(Enseignant\_ID, Enseignant.Département\_ID, Nom, Prénom, Grade, Téléphone, Fax, Email, Département.Département\_ID, Nom\_Département)

# Exemple d'union et de différence

$\Pi_{\text{Nom, Prénom}}(\text{Etudiant}) \cup \Pi_{\text{Nom, Prénom}}(\text{Enseignant}) :$

Ligne	nom (varc...	prenom...
1	BAHRI	Afef
2	CHAKHAR	Salem
3	Debécé	Aude
4	GAMOTTE	Albert
5	HIBULAIRE	Pat
6	LIMAM	Medhi
7	MANOUVRIER	Maude
8	MURAT	Cécile
9	MyTaylor	IsRich
10	NAIJA	Yosr
11	ODENT	Jamal
12	RASLATABLE	Deborah
13	RIG AUX	Philippe

$\Pi_{\text{Nom, Prénom}}(\text{Enseignant}) - \Pi_{\text{Nom, Prénom}}(\text{Etudiant}) :$

Ligne	nom (varchar)	prenom ...
1	MANOUVRIER	Maude
2	MURAT	Cécile
3	MyTaylor	IsRich
4	RIG AUX	Philippe

# Produit cartésien

NSS	Nom	Prénom	Grade	Dept
12345	Manouvrier	Maude	MCF	1
45678	Toto	Titi	Prof	2

*La relation Enseignant*

Dept_ID	Nom_Dept
1	Info
2	Math

*La relation Département*

# Produit cartésien

NSS	Nom	Prénom	Grade	Dept
12345	Manouvrier	Maude	MCF	1
45678	Toto	Titi	Prof	2

*La relation Enseignant*

Dept_ID	Nom_Dept
1	Info
2	Math

*La relation Département*

NSS	Nom	Prénom	Grade	Dept	Dept_ID	Nom_Dept
12345	Manouvrier	Maude	MCF	1	1	Info
45678	Toto	Titi	Prof	2	1	Info
12345	Manouvrier	Maude	MCF	1	2	Math
45678	Toto	Titi	Prof	2	2	Math

*La relation Département  $\times$  Enseignant*

# Autres opérations

- **Renommage :**

$$\Pi_{A',B',...}(\mathbf{r}_{A \rightarrow A', B \rightarrow B', ...})$$

- **Intersection :**

$$\mathbf{r} \cap \mathbf{s} = \mathbf{r} - (\mathbf{r} - \mathbf{s})$$

- **Théta-jointure :**

$$\mathbf{r} \bowtie_{\Theta} \mathbf{s} = \sigma_{\Theta}(\mathbf{r} \times \mathbf{s})$$

- **Jointure naturelle :**  $\mathbf{r}(\mathbf{R})$  et  $\mathbf{s}(\mathbf{S})$  avec  $\mathbf{R} \cap \mathbf{S} = \{A_1, A_2, \dots, A_n\}$

$$\mathbf{r} \bowtie \mathbf{s} = \Pi_{\mathbf{R} \cup \mathbf{S}} (\sigma_{(\mathbf{r}.A_1=\mathbf{s}.A_1) \wedge (\mathbf{r}.A_2=\mathbf{s}.A_2) \wedge \dots \wedge (\mathbf{r}.A_n=\mathbf{s}.A_n)}(\mathbf{r} \times \mathbf{s}))$$

# Exemple de renommage et d'intersection

$\Pi_{\text{Last\_Name, First\_Name}}(\text{Enseignant } \text{Nom} \rightarrow \text{Last\_Name}, \text{Prénom} \rightarrow \text{First\_Name}) :$

Ligne	last_name ...	first_name ...
1	MANOUVRIER	Maude
2	LIMAM	Medhi
3	MyTaylor	IsRich
4	RIGAUX	Philippe
5	MURAT	Cécile
6	CHAKHAR	Salem
7	NAIJA	Yosr
8	BAHRI	Afef

$\Pi_{\text{Nom, Prénom}}(\text{Enseignant}) \cap \Pi_{\text{Nom, Prénom}}(\text{Etudiant}) :$

Ligne	nom (varchar)	prenom...
1	BAHRI	Afef
2	CHAKHAR	Salem
3	LIMAM	Medhi
4	NAIJA	Yosr

# Exemple de produit cartésien

La relation *Enseignant* :

Ligne	enseignant_id ...	departement_id ...	nom (varchar)	prenom ...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	4	1	LIMAM	Medhi	ATER			
3	5	5	MyTaylor	IsRich	Vacataire			
4	6	1	RIGAUX	Philippe	PROF			
5	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr
6	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
7	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
8	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr

La relation *Departement* :

Ligne	departement_id ...	nom_departement...
1	1	INFO
2	2	MATHS
3	3	GESTION
4	4	ECO
5	5	LANGUES
6	7	COMM
7	8	OPTION

*Enseignement*  $\times$  *Departement* :

L...	enseignant_id...	departement_id (...	nom (varchar)	prenom...	grade...	telephone ...	fax ...	email (va...	departement_id...	nom_departement ...
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	1	INFO
2	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	2	MATHS
3	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	3	GESTION
4	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	4	ECO
5	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	5	LANGUES
6	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	7	COMM
7	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	8	OPTION
8	4	1	LIMAM	Medhi	ATER				1	INFO
9	4	1	LIMAM	Medhi	ATER				2	MATHS

# Exemple de jointure

La relation *Enseignant* :

Ligne	enseignant_id ...	departement_id ...	nom (varchar)	prenom ...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	4	1	LIMAM	Medhi	ATER			
3	5	5	MyTaylor	IsRich	Vacataire			
4	6	1	RIGAUX	Philippe	PROF			
5	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr
6	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
7	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
8	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr

La relation *Departement* :

Ligne	departement_id ...	nom_departement...
1	1	INFO
2	2	MATHS
3	3	GESTION
4	4	ECO
5	5	LANGUES
6	7	COMM
7	8	OPTION

*Enseignement*  $\bowtie_{\text{Departement\_ID}}$  *Departement* :

L...	enseignant_id...	departement_id ...	nom (varchar)	prenom...	grade ...	telephone ...	fax ...	email (va...	departement_id...	nom_departement ...
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	1	INFO
2	4	1	LIMAM	Medhi	ATER				1	INFO
3	5	5	MyTaylor	IsRich	Vacata...				5	LANGUES
4	6	1	RIGAUX	Philippe	PROF				1	INFO
5	8	1	MURAT	Cécile	MCF			murat@la...	1	INFO
6	7	1	CHAKHAR	Salem	ATER			chakhar...	1	INFO
7	2	1	NAIJA	Yosr	Moniteur			naija@lm...	1	INFO
8	3	1	BAHRI	Afef	Moniteur			bahri@lm...	1	INFO



# Division

**Requête qui contient le terme « pour tous »**

Soient  $r(R)$  et  $s(S)$  avec  $S \subseteq R$

**la relation  $r \div s$  a pour schéma  $R - S$**

un nuplet  $t$  appartient à  $r \div s$  si :

①  $t \in \Pi_{R-S}(r)$

②  $\forall t_s$  nuplet de  $s$ ,  $\exists t_r$  dans  $r$  qui satisfait :

♦  $t_r(S) = t_s(S)$

♦  $t_r(R-S) = t$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S} [ (\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r) ]$$

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	1	1	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	2	1	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	3	1	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	4	5	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	1	1	1	2004-02-25
2	1	2	1	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	1	3	1	2004-07-22
7	1	4	5	2004-07-22
8	2	4	5	2004-07-22

Ligne	etudiant_id (int4)
1	1

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	①	①	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	②	①	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	③	①	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	④	⑤	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	1	1	1	2004-02-25
2	1	2	1	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	1	3	1	2004-07-22
7	1	4	5	2004-07-22
8	2	4	5	2004-07-22

Ligne	etudiant_id (int4)
1	1

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	①	①	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	②	①	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	③	①	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	④	⑤	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	①	1	1	2004-02-25
2	①	2	1	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	①	3	1	2004-07-22
7	①	4	5	2004-07-22
8	2	4	5	2004-07-22

Ligne	etudiant_id (int4)
1	1

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	①	①	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	②	①	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	③	①	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	④	⑤	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	①	①	①	2004-02-25
2	①	②	①	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	①	③	①	2004-07-22
7	①	④	⑤	2004-07-22
8	2	4	5	2004-07-22

Ligne	etudiant_id (int4)
1	1

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	①	①	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	②	①	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	③	①	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	④	⑤	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	①	①	①	2004-02-25
2	①	②	①	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	①	③	①	2004-07-22
7	①	④	⑤	2004-07-22
8	2	4	5	2004-07-22

$$\Pi_{\text{Etudiant\_ID}, \text{Enseignement\_ID}, \text{Departement\_ID}} (\text{Inscription}) \div \Pi_{\text{Enseignement\_ID}, \text{Departement\_ID}} (\text{Enseignement}) :$$

Ligne	etudiant_id (int4)
1	1

# Contraintes et DF

- **Expressions des contraintes d'intégrité référentielle :**

$$\Pi_{\text{Département\_ID}}(\text{Enseignant}) \subseteq \Pi_{\text{Département\_ID}}(\text{Département})$$

$$\Pi_{\text{Département\_ID}}(\text{Enseignant}) - \Pi_{\text{Département\_ID}}(\text{Département}) = \emptyset$$

- **Expressions des dépendances fonctionnelles :**

$$X \rightarrow Y \Leftrightarrow \forall r \text{ et } \forall t_1, t_2 \in r \text{ on a :}$$

$$\Pi_X(t_1) = \Pi_X(t_2) \Rightarrow \Pi_Y(t_1) = \Pi_Y(t_2)$$

# Chap V - Algèbre relationnelle étendue

- **Projection généralisée :**

ajout d'expressions arithmétiques dans une projection

$$\Pi_{\text{Nom\_Client}, (\text{Crédit} - \text{Débit})}(\text{Compte\_en\_Banque})$$

- **Jointure externe (*outer-join*) :**

- jointure externe à gauche :  $\rfloor^\infty$

- jointure externe à droite :  $^\infty\lceil$

- jointure externe :  $\rfloor^\infty\lceil$

$R \rfloor^\infty S \Rightarrow R \bowtie S$  et conservation des attributs des nuplets de R qui ne joignent avec aucun nuplet de S (les valeurs des attributs de S sont mises à NULL)



*Personnel*

Nom_Employé	Ville
Tom	Marseille
Jerry	Paris
Alex	Limoges
Marthe	Perpignan

*Employé*

Nom_Employé	Filiale	Salaire
Tom	SUD_EST	10000
Jerry	IDF	25000
Sophie	IDF	15000
Marthe	SUD_OUEST	12000

*Personnel*

$\rfloor^\infty$

*Employé*

Nom_Employé	Ville	Filiale	Salaire
Tom	Marseille	SUD_EST	10000
Jerry	Paris	IDF	25000
<b>Alex</b>	<b>Limoges</b>	<b>NULL</b>	<b>NULL</b>
Marthe	Perpignan	SUD_OUEST	12000

*Personnel*

$^\infty\rfloor$

*Employé*

Nom_Employé	Ville	Filiale	Salaire
Tom	Marseille	SUD_EST	10000
Jerry	Paris	IDF	25000
<b>Sophie</b>	<b>NULL</b>	<b>IDF</b>	<b>15000</b>
Marthe	Perpignan	SUD_OUEST	12000

# Fonction d'agrégation

- Somme des places disponibles dans l'Université

**Sum**<sub>Capacité</sub>(Salle)

- Nombre moyen de places disponibles dans les salles de l'Université

**Avg**<sub>Capacité</sub>(Salle)

- Nombre d'étudiants à l'Université

**Count**<sub>Etudiant\_ID</sub>(Etudiant)

- Capacité de la plus petite salle

**Min**<sub>Capacité</sub>(Salle)

- Nombre d'enseignants par départements :

Nom\_Département  $\bowtie$  **Count**<sub>Enseignant\_ID</sub>(Enseignant  $\bowtie$  Département)

# Mise à jour de la base

- **Insertion**

$$\text{Salle} \leftarrow \text{Salle} \cup \{(\ll \text{B} \gg, \ll \text{038} \gg, 15)\}$$

- **Suppression**

$$\text{Salle} \leftarrow \text{Salle} - \sigma_{\text{Salle} \leq 10} (\text{Salle})$$

- **Mise à jour** : utilisation de la projection généralisée

$$r \leftarrow \Pi_{\text{Etudiant\_ID}} [\sigma_{(\text{Nom}=\text{' Dupont '}) \wedge (\text{Prénom}=\text{' Jacques '})} (\text{Etudiant})]$$

$$\text{Etudiant} \leftarrow \sigma_{(\text{Etudiant.Etudiant\_ID} \neq r.\text{Etudiant\_ID})} (\text{Etudiant})$$

$\cup$

*Mise à jour du  
téléphone*

$$\Pi_{\text{Etudiant\_ID}, \text{Nom}, \text{Prénom}, \text{Rue}, \text{Ville}, \text{Code-Postal},$$

$$\text{Téléphone} \leftarrow \ll 45\ 12\ 45\ 86 \gg, \text{Fax}, \text{Email}, \text{NumAnnées}$$

$$[\sigma_{(\text{Etudiant.Etudiant\_ID} = r.\text{Etudiant\_ID})} (\text{Etudiant}) ]$$

# Vue

Table virtuelle dont le schéma et les instances sont dérivés de la base réelle par une requête et qui est utilisée pour :

- Cacher certaines informations à un groupe d'utilisateurs
- Faciliter l'accès à certaines données

**create view** nom\_vue **as** < requête >

Exemple :

**create view** Info\_Non\_Confidentielle\_Etudiant

**as**  $\Pi_{\text{Etudiant\_ID, Nom, Prénom, Email}}$  (Etudiant)

# Chap VI - SQL

## *Structured Query Language*

- **SQL2** : standard adopté en 1992
- **SQL3** : extension de SQL2 avec "gestion" d'objets

## **SQL :**

- **Langage de Manipulation de Données (DML)** : interroger et modifier les données de la base
- **Langage de Définition de Données (DDL)** : définir le schéma de la base de données
- **Langage de contrôle d'accès aux données**

# Bibliographie

- *SQL2 - Application à Oracle, Access et RDB*  
Pierre DELMAL, 2ème Edition, De Boeck Université,  
1998 (BU: 005.74 SQL)
- *SQL Pour Oracle (avec exercices corrigés)*  
Christian Soutou, Eyrolles, 2005
- *Initiation à SQL (cours et exercices corrigés)*  
Philip J. Pratt, Eyrolles, 2001
- *Oracle PL/SQL - Précis & concis*  
Steven Feuerstein, Bill Pribyl et Chip Dawes, O'Reilly,  
2000

# DML

```
SELECT [DISTINCT] *  
FROM table_1 [synonyme_1], table_2 [synonyme_1], ...  
[WHERE prédicat_1  
    AND [ou OR] prédicat_2 ...]
```

```
SELECT [DISTINCT] exp_1 [AS nom_1], exp_2 ...  
FROM table_1 [synonyme_1], table_2 [synonyme_1], ...  
[WHERE prédicat_1  
    AND [ou OR] prédicat_2 ...]
```

# DML

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Ville = ' Paris ' ;
```

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Ville = ' Paris '  
AND Nom  
LIKE '_AR%' ;
```

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Fax IS NULL;
```

```
SELECT Intitulé,  
(NbSeances*3) AS NbHeures  
FROM Cours  
WHERE (NbSeances*3)  
BETWEEN 24 AND 27 ;
```

```
SELECT Nom, Prénom  
FROM Enseignant  
WHERE Département_ID IN  
(' INFO ', ' MATH ', ' ECO ')
```



# DML

## Prédicats du WHERE de la forme :

exp1 = exp2

exp1 != exp2

exp1 > exp2

exp1 < exp2

exp1 <= exp2

exp1 >= exp2

exp1 BETWEEN exp2 AND exp3

exp1 LIKE exp2

exp1 IN (exp2, exp3, ...)

exp1 NOT IN (exp2, exp3, ...)

exp1 IS NULL

exp1 IS NOT NULL

exp *op* ANY (SELECT ...)

exp *op* ALL (SELECT ...)

avec *op* tel que =, !=, <, > ...

exp IN (SELECT ...)

exp NOT IN (SELECT ...)

```
SELECT Intitulé,  
FROM Cours  
WHERE NbSeances <=  
  ( SELECT AVG(NbSeances)  
    FROM Cours);
```

# DML

## Clause EXISTS :

- Retourne VRAI si au moins un nuplet est renvoyé par la requête
- FAUX si aucun nuplet n'est retourné.
- La valeur NULL n'a aucun effet sur le booléen résultat

```
SELECT Nom, Prénom  
FROM Enseignant E  
WHERE NOT EXISTS  
  ( SELECT *  
    FROM Reservation_Salle S  
      WHERE S.Enseignant_ID = E.Enseignant_ID  
  );
```

# DML

## Fonctions de groupe :

COUNT, MIN, MAX, AVG, SUM, ORDER BY, GROUP BY

**SELECT COUNT(\*)**

**FROM Etudiant ;**

**SELECT AVG(Capacité), SUM(Capacité)**

**FROM Salle ;**

**SELECT Département\_ID, Nom, Prénom**

**FROM Enseignant**

**ORDER BY Département\_ID DESC, Nom, Prénom ;**

**SELECT Département\_ID, COUNT(\*)**

**FROM Réservation\_Salle**

**GROUP BY Département\_ID HAVING COUNT(\*) >=4 ;**

# DML

## Jointure :

```
SELECT Nom, Prénom, Nom_Département  
FROM Enseignant E, Département D  
WHERE E.Département_ID = D.Département_ID ;
```

## Jointure externe : sous Oracle

```
SELECT Nom, Prénom, Nom_Département  
FROM Enseignant E, Département D  
WHERE E.Département_ID = D.Département_ID (+);
```

**S'il existe des enseignants attaché à aucun département, la valeur de Département\_ID sera NULL.**

**En SQL2 : [RIGHT | LEFT | FULL] OUTER JOIN**

# DML

## Opérateurs ensemblistes : sous Oracle

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
INTERSECT
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
UNION
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '  
ORDER BY Nom,Prénom
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
MINUS
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '
```

**MINUS = EXCEPT en standard SQL2**

# DML

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels livres ont été empruntés par tous les étudiants? »

$$\{t.\text{Titre} / \text{Livre}(t) \wedge [ \forall u \text{ Etudiant}(u) \\ (\exists v \text{ Emprunt}(v) \wedge \\ (v.\text{Etudiant\_ID}=u.\text{Etudiant\_ID}) \wedge (v.\text{ISBN}=t.\text{ISBN}) \\ ) \\ ] \\ \}$$

# DML

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels livres ont été empruntés par tous les étudiants? »

$$\{t.\text{Titre} / \text{Livre}(t) \wedge [ \forall u \text{ Etudiant}(u) \\ (\exists v \text{ Emprunt}(v) \wedge \\ (v.\text{Etudiant\_ID}=u.\text{Etudiant\_ID}) \wedge (v.\text{ISBN}=t.\text{ISBN}) \\ ) \\ ] \\ \}$$

*Il n'y a pas de mot-clé  
"quel que soit " en SQL2*

# DML

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels livres ont été empruntés par tous les étudiants? »

$$\{t.\text{Titre} / \text{Livre}(t) \wedge \neg [\exists u \text{ Etudiant}(u) \wedge \neg (\exists v \text{ Emprunt}(v) \wedge (v.\text{Etudiant\_ID}=u.\text{Etudiant\_ID}) \wedge (v.\text{ISBN}=t.\text{ISBN}))]$$

$$\}$$

*Il n'y a pas de mot-clé  
"quel que soit " en SQL2*



# DML

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels livres ont été empruntés par tous les étudiants? »

$$\{t.\text{Titre} / \text{Livre}(t) \wedge \neg [ \exists u \text{ Etudiant}(u) \wedge \neg (\exists v \text{ Emprunt}(v) \wedge (v.\text{Etudiant\_ID}=u.\text{Etudiant\_ID}) \wedge (v.\text{ISBN}=t.\text{ISBN}) ) ] \}$$

*Il n'y a pas de mot-clé  
"quel que soit " en SQL2*

```
SELECT t.Titre FROM Livre t WHERE NOT EXISTS
  ( SELECT * FROM Etudiant u WHERE NOT EXISTS
    ( SELECT * FROM Emprunt v
      WHERE u.EtudiantID=v.EtudiantID AND
        v.ISBN=t.ISBN
    )
  );
```

# DML

- **Insertion**

**INSERT INTO table(col1, col2, ... coln)**

**VALUES (val1, val2, ... valn)**

**INSERT INTO table(col1, col2, ... coln)**      *Sous Oracle*

**SELECT**

- **Suppression**

**DELETE FROM table**

**WHERE prédicat**

- **Mise à jour**

**UPDATE table**

**SET col1 = exp1, col2 = exp2 WHERE prédicat**

- **Transactions : COMMIT, ROLLBACK [TO], SAVE POINT**

# DDL

```
CREATE TABLE table (col1 type 1 [NOT NULL] ,  
                    col2 type2 [NOT NULL] ...  
                    )
```

**Contraintes :**

**CONSTRAINT** *nom\_contrainte*

**PRIMARY KEY** (liste attributs clé primaire)

| **NOT NULL** *immédiatement après la déclaration de l'attribut*

| **CHECK** (condition) *après la déclaration de l'attribut*

| **UNIQUE** *après la déclaration de l'attribut*

| **FOREIGN KEY** (clé étrangère)

**REFERENCES** nom\_table (liste-colonne)

```
CREATE TABLE table  
AS SELECT ...
```

# DDL

**CREATE TABLE Enseignant**

**(**

**Enseignant\_ID**                      **integer,**

**Departement\_ID**                  **integer NOT NULL,**

**Nom**                                  **varchar(25) NOT NULL,**

**Prenom**                            **varchar(25) NOT NULL,**

**Grade**                              **varchar(25)**

**CONSTRAINT CK\_Enseignant\_Grade**

**CHECK (Grade IN ('Vacataire', 'Moniteur', 'ATER', 'MCF', 'PROF')),**

**Telephone**                        **varchar(10) DEFAULT NULL,**

**Fax**                                  **varchar(10) DEFAULT NULL,**

**Email**                               **varchar(100) DEFAULT NULL,**

**CONSTRAINT PK\_Enseignant PRIMARY KEY (Enseignant\_ID),**

**CONSTRAINT "FK\_Enseignant\_Departement\_ID"**

**FOREIGN KEY (Departement\_ID)**

**REFERENCES Departement (Departement\_ID)**

**ON UPDATE RESTRICT ON DELETE RESTRICT**

**);**

*Contrainte  
de domaine*

*Définition de la  
clé primaire*

*Définition d'une clé  
étrangère*

```

CREATE TABLE Reservation
(
  Reservation_ID    integer,
  Batiment          varchar(1) NOT NULL,
  Numero_Salle      varchar(10) NOT NULL,
  Enseignement_ID   integer NOT NULL,
  Departement_ID    integer NOT NULL,
  Enseignant_ID     integer NOT NULL,
  Date_Resa         date NOT NULL DEFAULT CURRENT_DATE,
  Heure_Debut       time NOT NULL DEFAULT CURRENT_TIME,
  Heure_Fin         time NOT NULL DEFAULT '23:00:00',
  Nombre_Heures     integer NOT NULL,
  CONSTRAINT PK_Reservation PRIMARY KEY (Reservation_ID),
  CONSTRAINT "FK_Reservation_Salle" FOREIGN KEY (Batiment,Numero_Salle) REFERENCES
    Salle (Batiment,Numero_Salle) ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT "FK_Reservation_Enseignement" FOREIGN KEY (Enseignement_ID,Departement_ID)
    REFERENCES Enseignement (Enseignement_ID,Departement_ID) ON UPDATE RESTRICT ON
    DELETE RESTRICT,
  CONSTRAINT "FK_Reservation_Enseignant" FOREIGN KEY (Enseignant_ID) REFERENCES
    Enseignant (Enseignant_ID) ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT CK_Reservation_Nombre_Heures CHECK (Nombre_Heures >=1),
  CONSTRAINT CK_Reservation_HeureDebFin
    CHECK (Heure_Debut < Heure_Fin)
);

```

# DDL

**CREATE ASSERTION <nom contrainte>**

**[ {BEFORE COMMIT |**

**AFTER {INSERT | DELETE | UPDATE[OF (Attributs)]} ON**

**<Relation>} ...]**

**CHECK <Condition>**

**[FOR [EACH ROW OF] <Relation> ]**

**CREATE ASSERTION CA\_Place\_Université**

**BEFORE COMMIT**

**CHECK( (SELECT SUM(Capacité) FROM Salle)**

**>= (SELECT COUNT(\*) FROM Etudiant)**

**)**

# DDL

**CREATE [OR REPLACE] TRIGGER nom {BEFORE | AFTER}  
événement\_déclencheur ON nom\_table  
[FOR EACH ROW]  
[WHEN (condition) ]  
bloc PL/SQL *sous Oracle*  
| inst\_de\_suppr | inst\_de\_modif | instr\_d\_ajout | ERROR *en SQL2*  
*événement\_déclencheur* = INSERT, UPDATE, DELETE**

- Déclencheur de *niveau instruction* : pas de clause **FOR EACH ROW**
- Déclencheur de *niveau ligne* : variables liens *:new* et *:old*
  - INSERT : valeurs à insérer dans *:new.nom\_colonne*
  - UPDATE : valeur originale dans *:old.nom\_colonne*, nouvelle valeur dans *:new.nom\_colonne*
  - DELETE : valeur en cours de suppression *:old.nom\_colonne*

# DDL

```
CREATE OR REPLACE TRIGGER Enseignant_Actif
BEFORE DELETE ON Enseignant
FOR EACH ROW
  declare
    counter number;
  begin
    SELECT count(*) INTO counter
    FROM Enseignements
    WHERE Enseignant_ID = :old.Enseignant_ID;
    if counter > 0 then
      raise_application_error (-20800, 'Enseignant actif ne
      pouvant être supprimé');
    end if;
  end;
```



# DDL

```
CREATE OR REPLACE TRIGGER UPD_salaire_personnel
BEFORE UPDATE salaire ON Personnel
FOR EACH ROW
WHEN (:old.salaire > :new.salaire)
declare
    salaire_diminution EXCEPTION;
begin
    raise salaire_diminution ;
    when salaire_diminution then
        raise_application_error(-20001, 'Le salaire ne peut pas
diminuer ')
end;
```

# DDL

## Sous PostgreSQL :

```
CREATE OR REPLACE FUNCTION GetSalleCapaciteSuperieurA(int)  
RETURNS SETOF Salle  
AS '  
    SELECT * FROM Salle WHERE Capacite > $1;  
;  
LANGUAGE SQL;  
  
SELECT * FROM GetSalleCapaciteSuperieurA(300) ;
```

# DDL

```
CREATE OR REPLACE FUNCTION FunctionTriggerReservation()  
  RETURNS trigger AS  
' DECLARE  
  resa Reservation.Reservation_ID%TYPE;  
BEGIN  
  SELECT INTO resa Reservation_ID  
    FROM Reservation  
    WHERE ...  
  IF FOUND THEN RAISE EXCEPTION "Réservation impossible, salle  
    occupée à la date et aux horaires demandés";  
  ELSE RETURN NEW;  
  END IF;  
END;  
LANGUAGE 'plpgsql';
```

# DDL

## Sous PostgreSQL :

```
CREATE TRIGGER InsertionReservation  
BEFORE INSERT ON Reservation  
FOR EACH ROW  
EXECUTE PROCEDURE  
FunctionTriggerReservation();
```

# DDL

**ALTER TABLE table**

**ADD (col1 type1, col2 type2 ...)**

**| MODIFY (col1 type1, col2 type2 ...)**

**| DROP PRIMARY KEY**

**| DROP CONSTRAINT nom\_contrainte**

**DROP TABLE table**

**CREATE VIEW vue (col1, col2)**

**AS SELECT ...**

**DROP VIEW vue**

**CREATE [UNIQUE] INDEX nom\_index ON table (col1,col ...)**

# Embedded SQL

Utilisation de commandes SQL à l'intérieur d'un langage hôte :

- Commandes SQL remplacée par des appel de fonctions du langage hôte par le précompilateur.
- Commandes SQL reconnues par **EXEC SQL**

```
/* déclaration de variables hôtes */  
EXEC SQL BEGIN DECLARE SECTION  
char d_name[20];  
char d_id;  
EXEC SQL END DECLARE SECTION  
...  
EXEC SQL INSERT INTO Department  
VALUES (:d_id, :d_name);
```

# Embedded SQL

Gestion des erreurs :

```
EXEC INCLUDE SQLCA;
```

```
...
```

```
EXEC SQL WHENEVER SQLERROR GOTO erreur
```

```
...
```

```
erreur :
```

```
    printf(`erreur : les transactions en  
cours vont être annulées'\n');
```

```
    EXEC SQL ROLLBACK WORK RELEASE;
```

```
    exit(1);
```

# Embedded SQL

Gestion de curseur :

```
/* Déclaration d'un curseur pour manipuler la
   table Department */
EXEC SQL DECLARE c1 CURSOR FOR
    SELECT * FROM Department ;

/* Ouverture du curseur */
EXEC SQL OPEN c1;

/* Lecture de la première ligne de la table */
EXEC SQL FETCH c1 INTO :d_id, :d_name ;
printf(`Nom du département %s, identifiant :
    %s\n`, d_name, d_id);

/* fermeture du curseur */
EXEC SQL CLOSE c1;
```



# Middleware d'accès aux bases de données

## *Open DataBase Connectivity (ODBC)*

- *Middleware* propriétaire (Windows)
- Architecture logicielle définissant une interface standard d'accès aux SGBD
- A chaque SGBD correspond un pilote (*driver*)

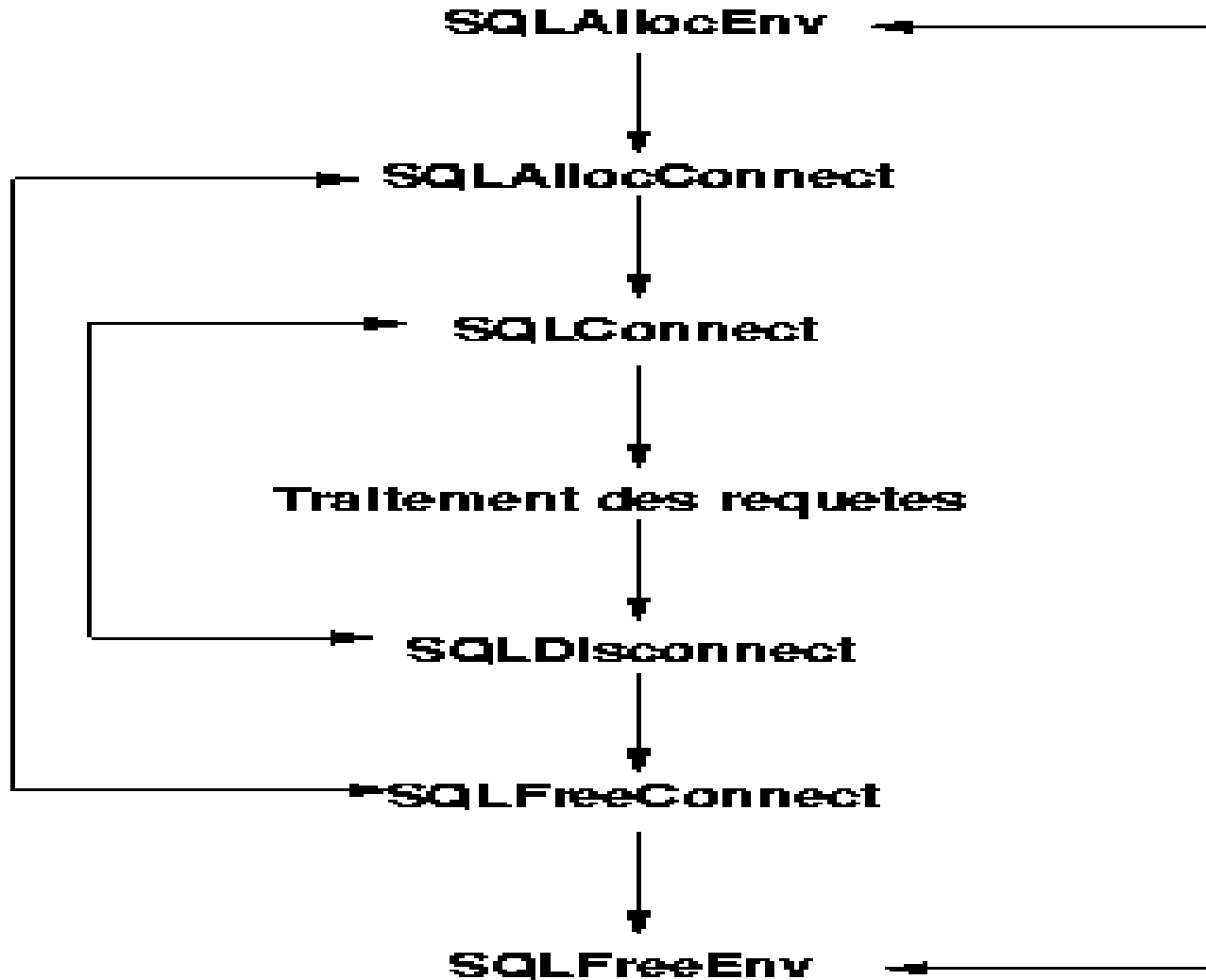
⇒ un même programme peut interroger différentes bases de données dans différents SGBD

## *Java DataBase Connectivity (JDBC)*

# ODBC

- Appel des fonctions de l'API ODBC :  
⇒ lien entre l'application et le **gestionnaire de pilotes**
- **Gestionnaire de pilotes** = DLL qui charge les pilotes associés à chaque **source de données** (BD + SGBD)
- **Pilote** = DLL qui contient les appels ODBC et traduit les requêtes en requêtes propres au SGBD

# ODBC



# ODBC

## Exemple de programme en C sous Visual C++ :

```
#include <stdio.h>
#include <conio.h>
#include <afxdb.h> // MFC ODBC database classes

char *cBASE ;    /* Nom de la source de données */
char *cLOGIN ;   /* Login utilisateur */
char *cPASSWD ;  /* Mot de passe utilisateur */

void main()
{
    HENV d_env;      /* Descripteur d'environnement */
    HDBC d_connex;   /* Descripteur de connexion */
    HSTMT curseur;   /* Curseur */
    RETCODE retcode; /* Code de retour de fonction */
```

# ODBC

```
UCHAR ucLastName[20],ucCity[20];
SDWORD ceLastName,ceCity;
char *cREQUETESQL; /* Variable recevant une requête SQL */

/* Saisie du nom de la source de données */
cBASE=(char*)malloc(20);
printf("Nom de la base de donnees :"); scanf("%s",cBASE);
/* Saisie du login */
cLOGIN=(char*)malloc(20); printf("Login :");
scanf("%s",cLOGIN);
/* Saisie du password */
cPASSWD=(char*)malloc(20); printf("Mot de passe : ");

/* Pour ne pas afficher le mot de passe à l'écran */
int iPosCaractere=0; fflush(stdin);
do { if((cPASSWD[iPosCaractere]=_getch())!='\r')
    printf("*");
    } while(cPASSWD[iPosCaractere++]!='\r' && iPosCaractere <20);
cPASSWD[--iPosCaractere]='\0';
```

# ODBC

```
/* Création d'un environnement ODBC */  
retcode = SQLAllocEnv(&d_env);  
/* Si la création d'un environnement ODBC est correcte */  
if (retcode == SQL_SUCCESS)  
{  
    /* Création d'une connexion ODBC */  
    retcode = SQLAllocConnect(d_env, &d_connex);  
  
    /* Si la connexion ODBC s'est bien passée */  
    if (retcode == SQL_SUCCESS)  
    {  
        /* Initialisation du temps de connexion à 5 secondes. */  
        SQLSetConnectOption(d_connex, SQL_LOGIN_TIMEOUT, 5);  
  
        /* Connexion à une source de données */  
        retcode = SQLConnect(d_connex, (unsigned char*)cBASE, SQL_NTS, (unsigned  
            char*)cLOGIN, SQL_NTS, (unsigned char*)cPASSWD, SQL_NTS);
```

↑  
*Longueur de la chaîne ou on indique que la chaîne se termine par le code NULL*

# ODBC

```
/* Si la connexion à la source de données s'est bien passée */
if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
    printf("Connection a la base (source de données).\n");

    /* Pause dans l'affichage */
    printf("Taper une touche pour continuer \n"); getchar();
    /* Allocation mémoire du curseur et association du curseur à la
       source de données identifiée par d_connex. */
    retcode = SQLAllocStmt(d_connex, &curseur);

    /* Si l'allocation mémoire du curseur est correcte */
    if (retcode == SQL_SUCCESS)
    { /* Création de la requête SQL */
        cREQUETESQL = "SELECT Nom, Ville FROM Etudiant";

        /* Execution directe de la requête sur la base */
        retcode = SQLExecDirect(curseur, (unsigned char*)cREQUETESQL,
        SQL_NTS);
        printf("EXECUTION DE LA REQUETE, CODE ERREUR %d, CODE DE SUCCES
        %d \n",retcode,SQL_SUCCESS);
    }
}
```

# ODBC

```
/* Tant le parcours du curseur est valide */
while (retcode == SQL_SUCCESS)
{
    /* Parcourt de l'enregistrement résultat de la requête */
    retcode = SQLFetch(curseur);

    /* Si le parcours est incorrect */
    if (retcode == SQL_ERROR || retcode == SQL_SUCCESS_WITH_INFO)
    {
        printf("Erreur %d\n",SQL_ERROR);
    }

    /* Si le parcours des enregistrements est correct */
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
    {
        /* récupération des données des colonnes 1 et 2 de la table
        résultat */
        SQLGetData(curseur, 1, SQL_C_CHAR, ucLastName, 30, &ceLastName);
        SQLGetData(curseur, 2, SQL_C_CHAR, ucCity, 30, &ceCity);
        /* Affichage du résultat */
        printf("Etudiant : %s %s\n",ucLastName,ucCity);
    }
}
```



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

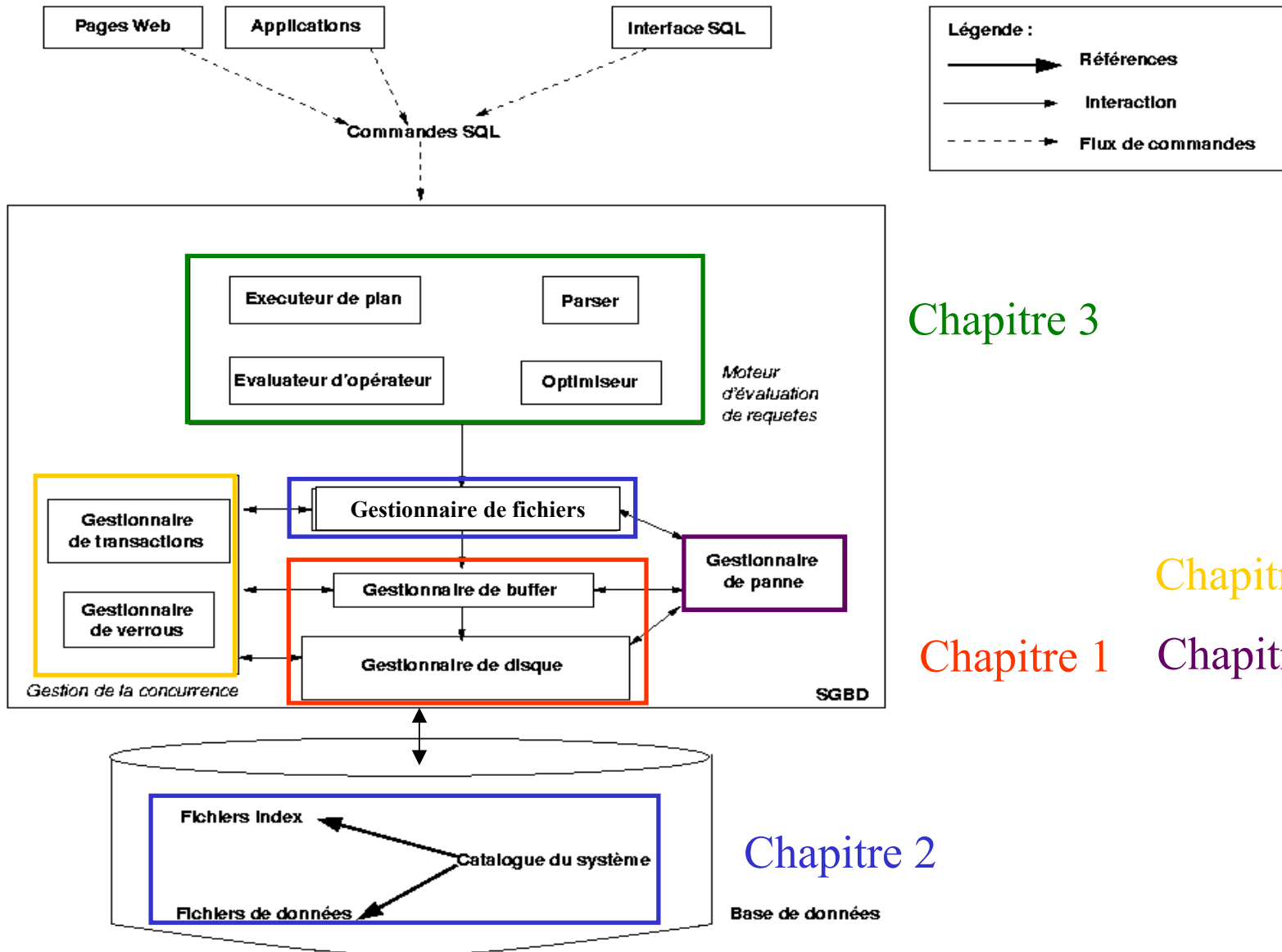
T. Connolly, C. Begg et A. Strachan, *Database Systems A Pratical Approach to Desigh, Implementation and Management*, 1998, disponible à la BU 055.7 CON

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)

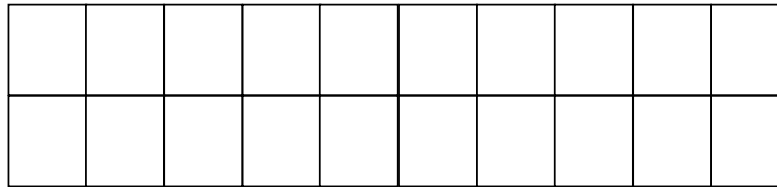
# Gestionnaire de buffer

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

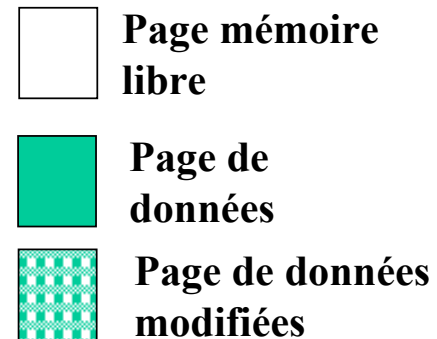
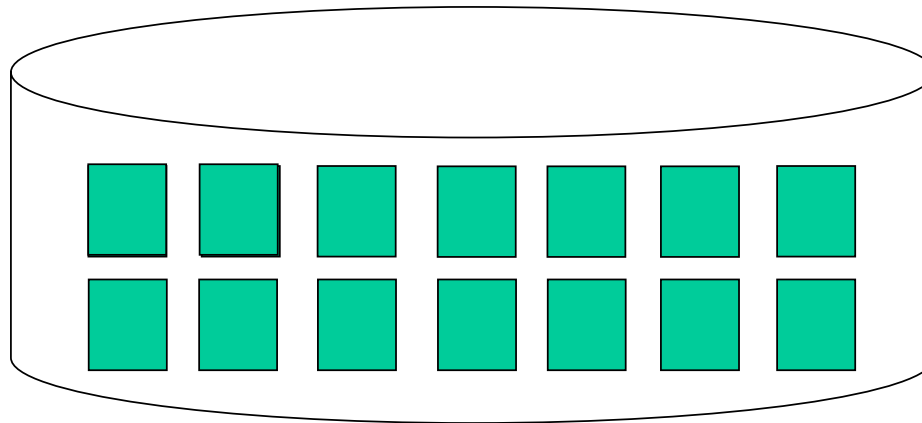
- Politique de remplacement (ex. LRU)
- Gestion des pages mises à jour
- Partition de la mémoire
- Vérification des droits sur les pages

# Gestionnaire de buffer

Mémoire

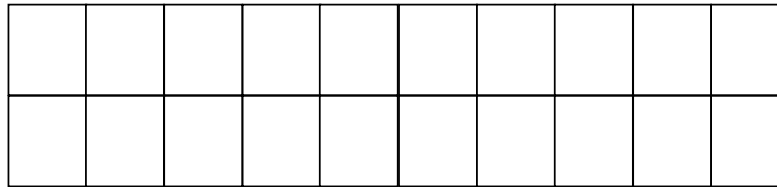


Disque

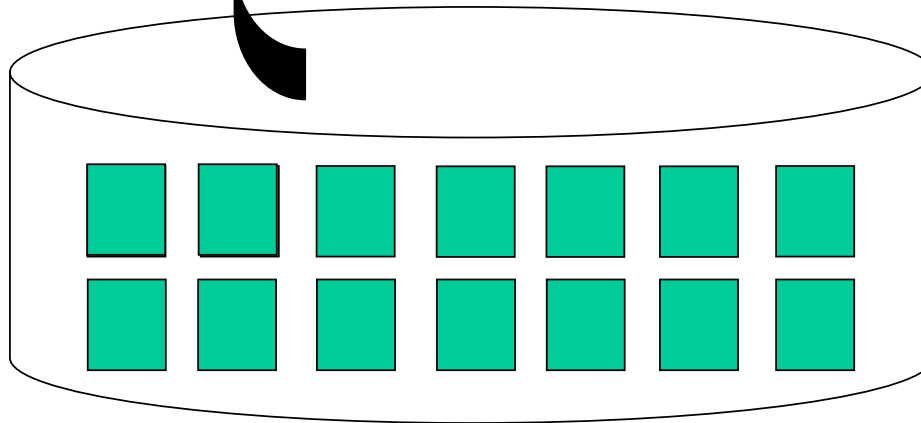


# Gestionnaire de buffer

Mémoire



Disque



**Page mémoire  
libre**

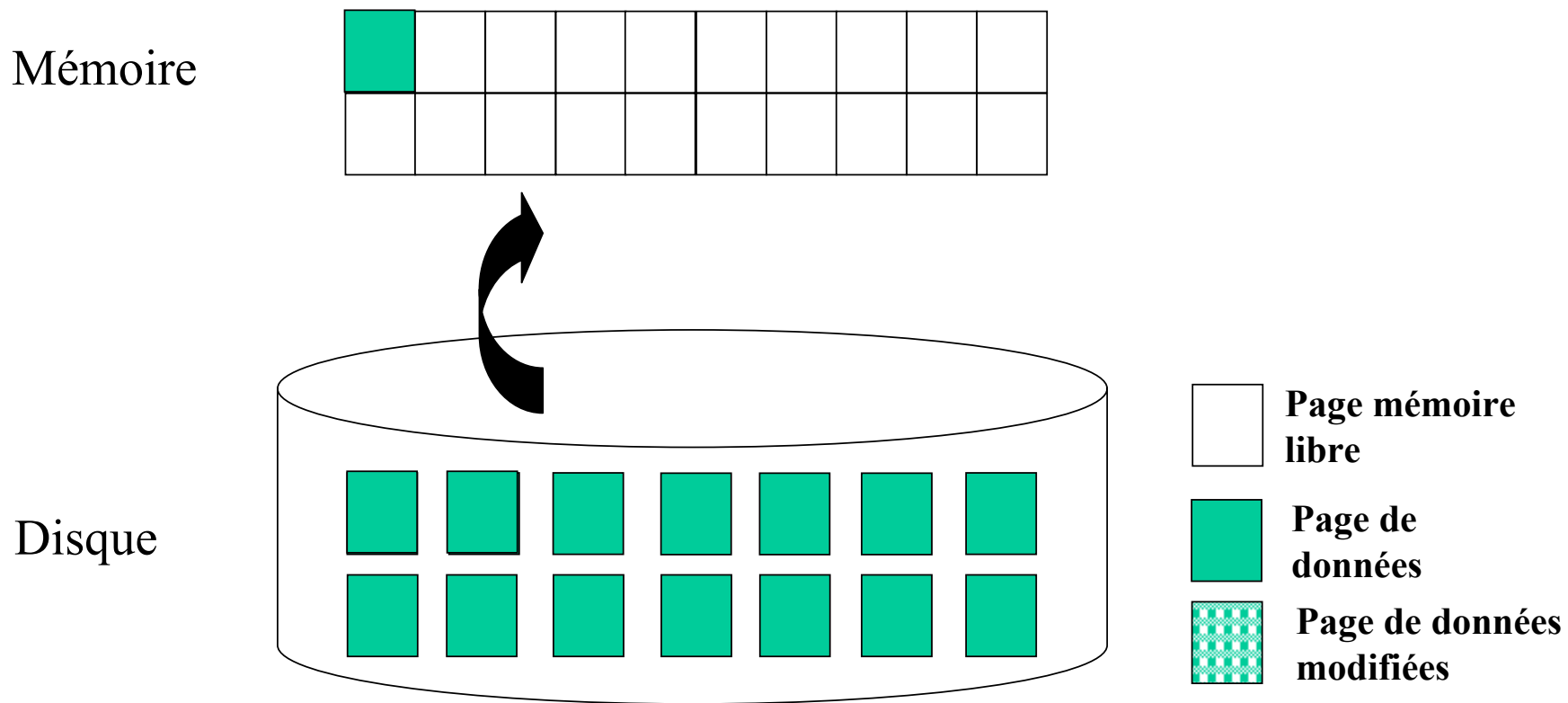


**Page de  
données**



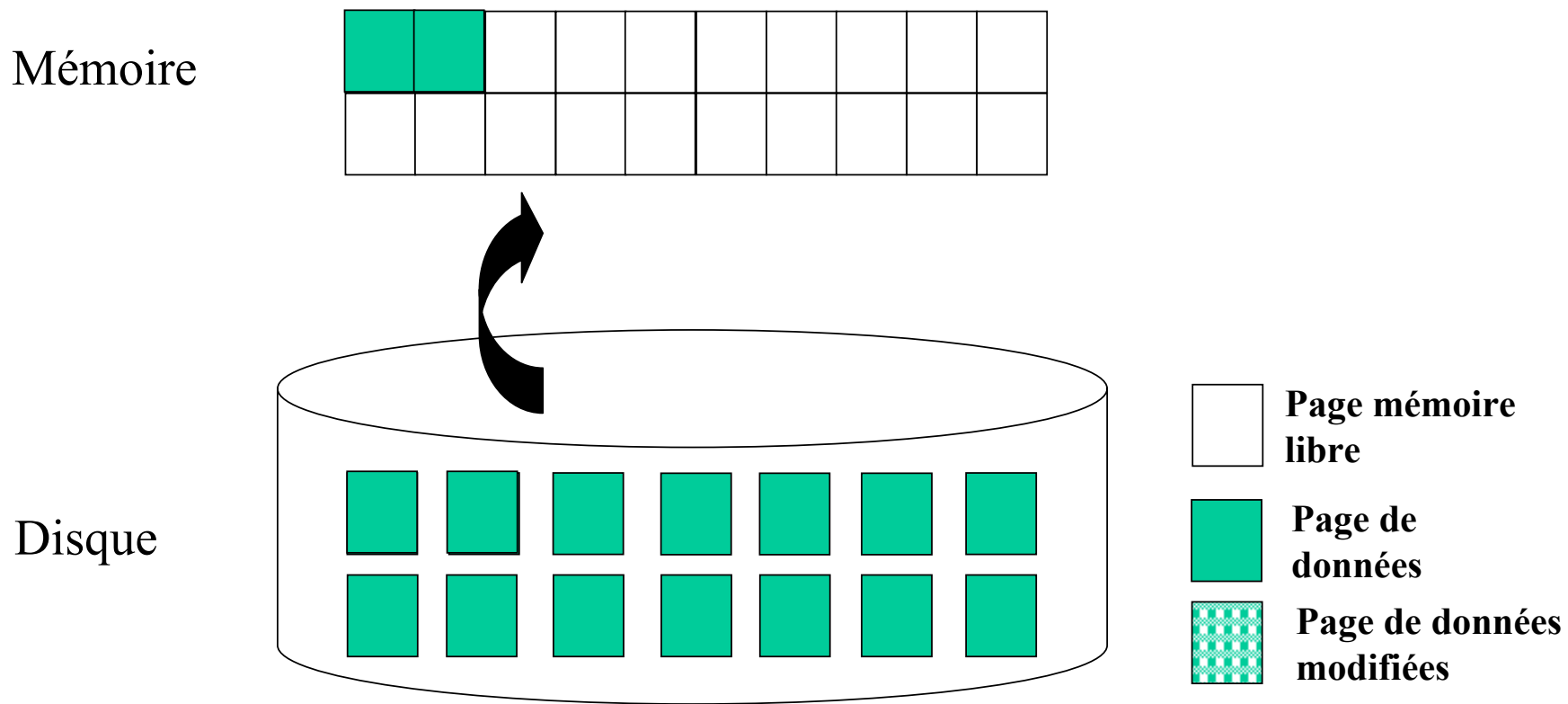
**Page de données  
modifiées**

# Gestionnaire de buffer

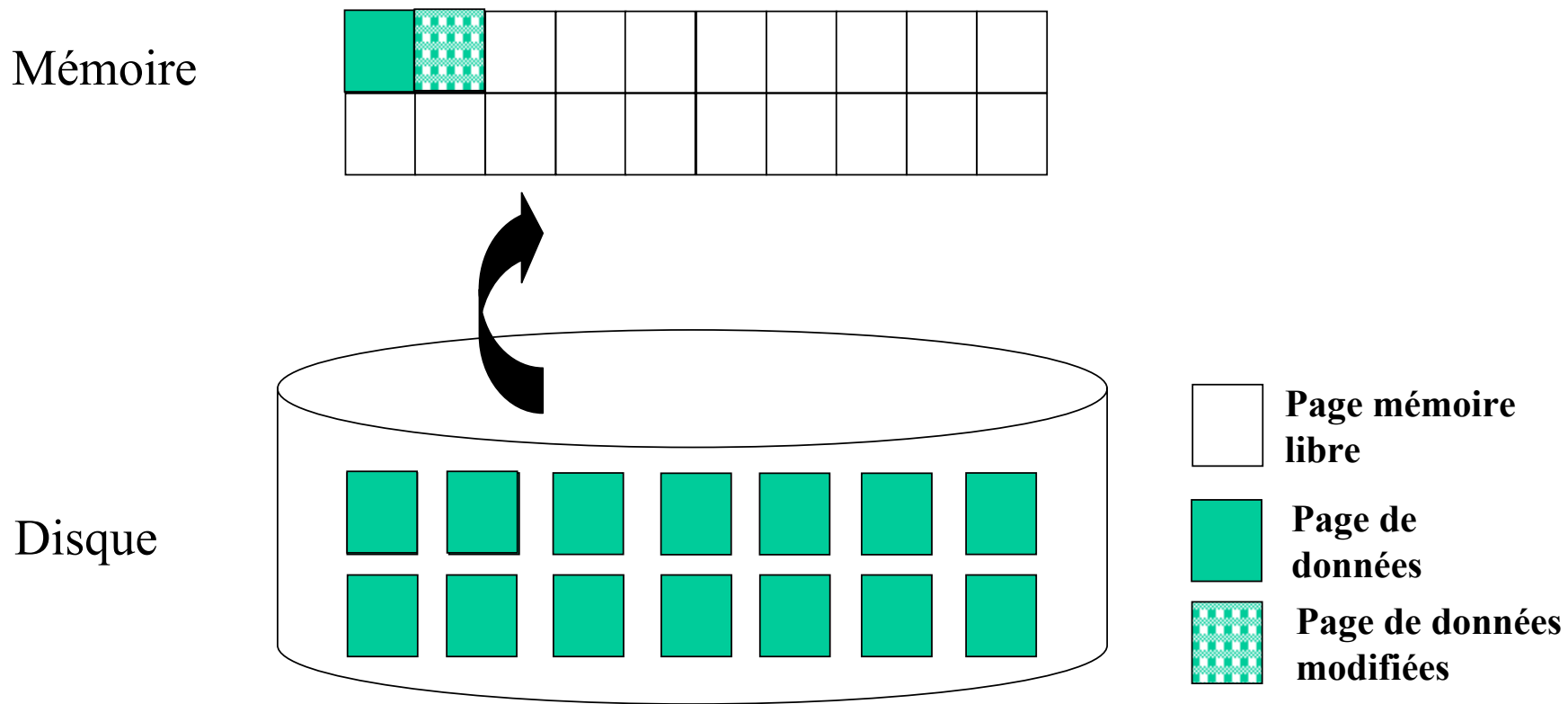




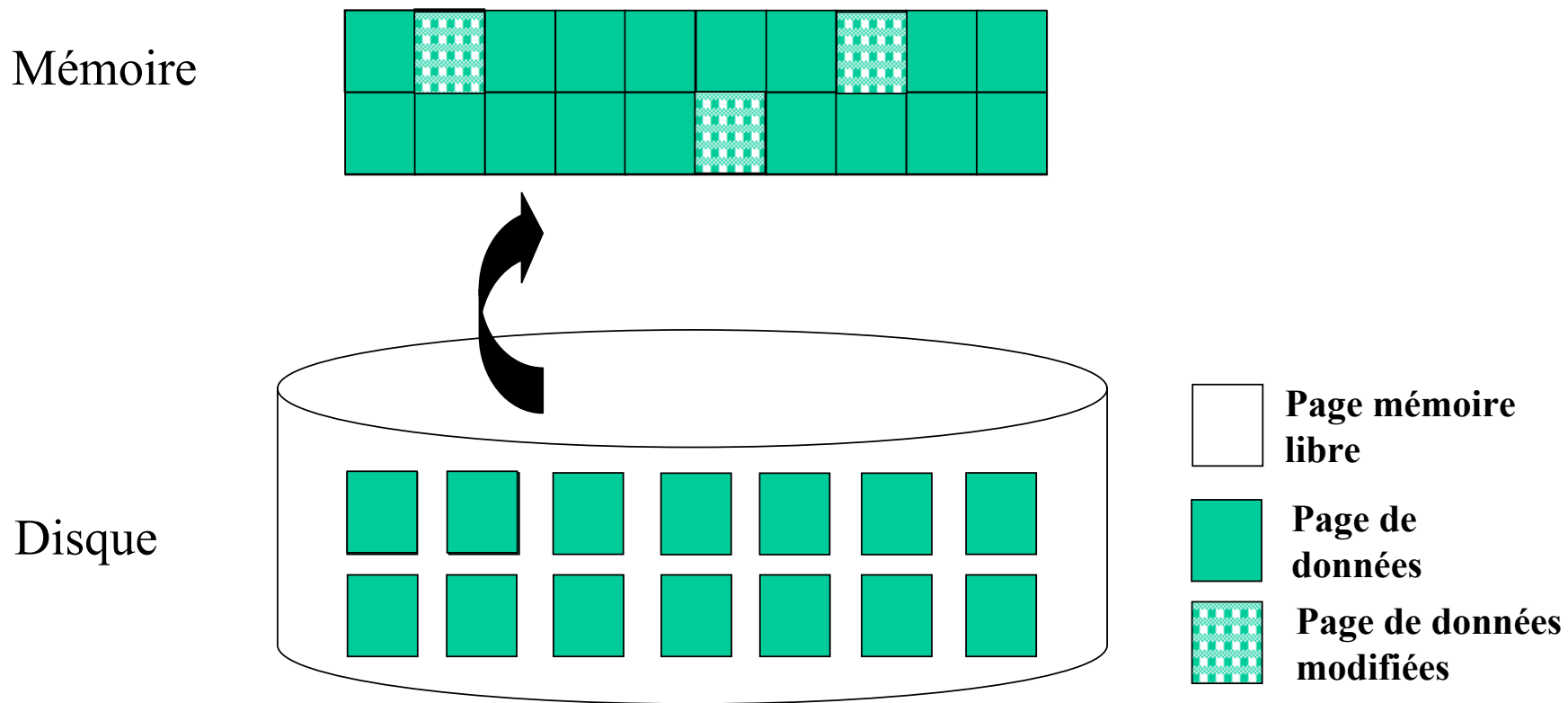
# Gestionnaire de buffer



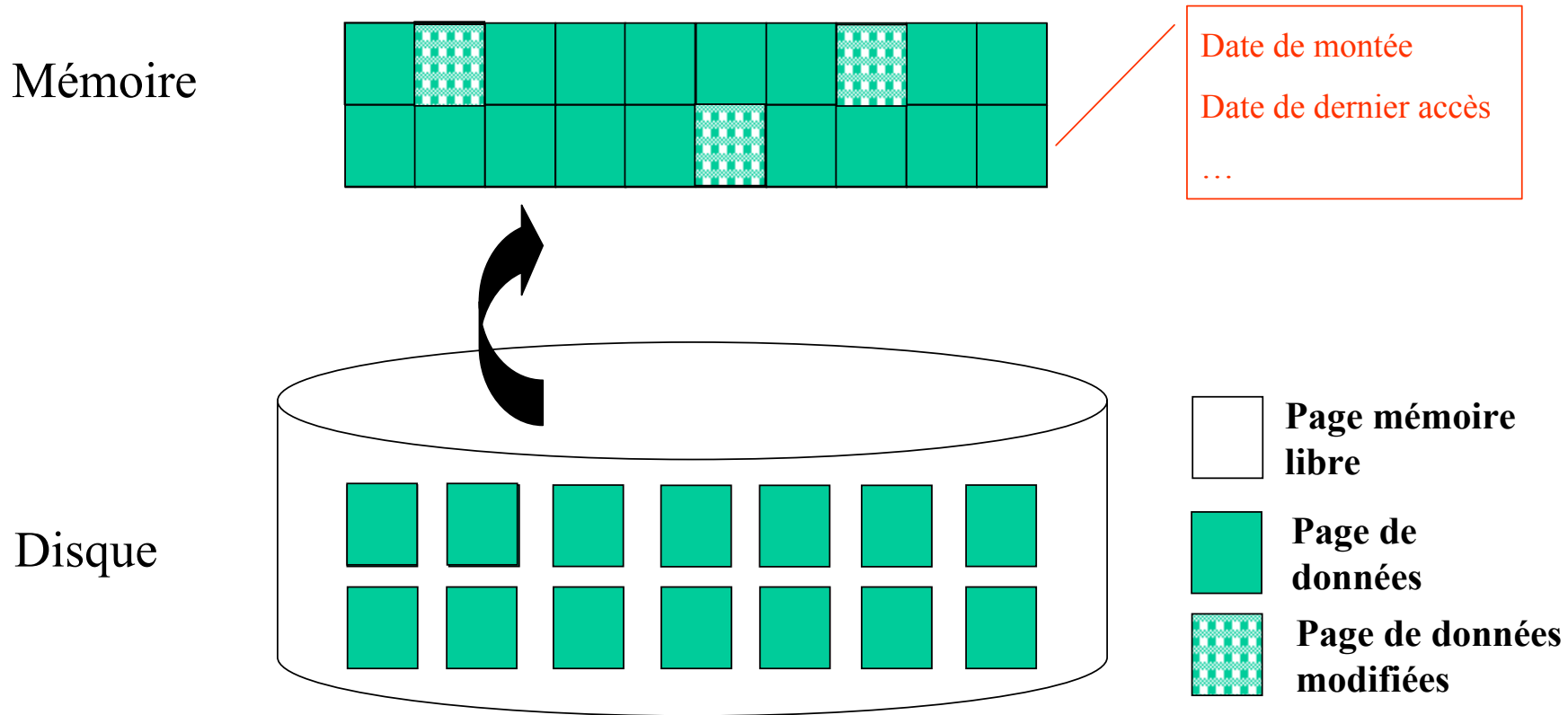
# Gestionnaire de buffer



# Gestionnaire de buffer



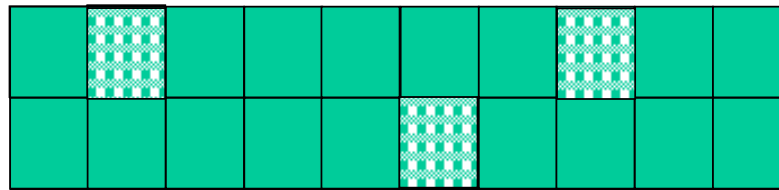
# Gestionnaire de buffer



# Gestionnaire de buffer

Page non accédée depuis longtemps (LRU)

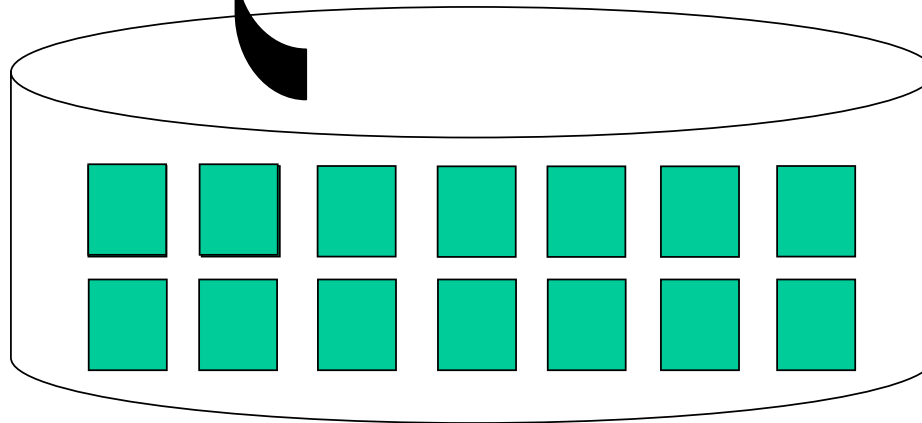
Mémoire






Date de montée  
Date de dernier accès  
...



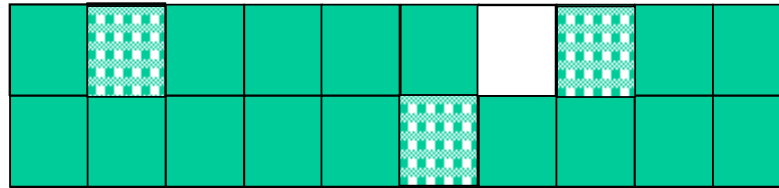
Disque



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

# Gestionnaire de buffer

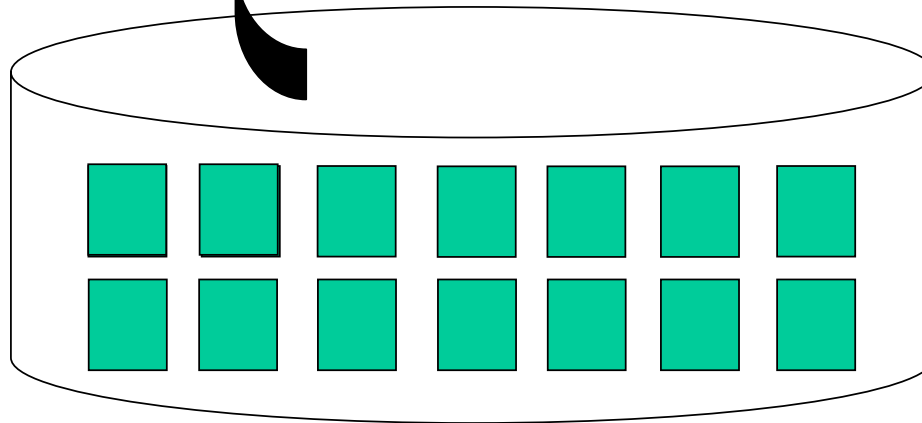
Mémoire






Date de montée  
Date de dernier accès  
...

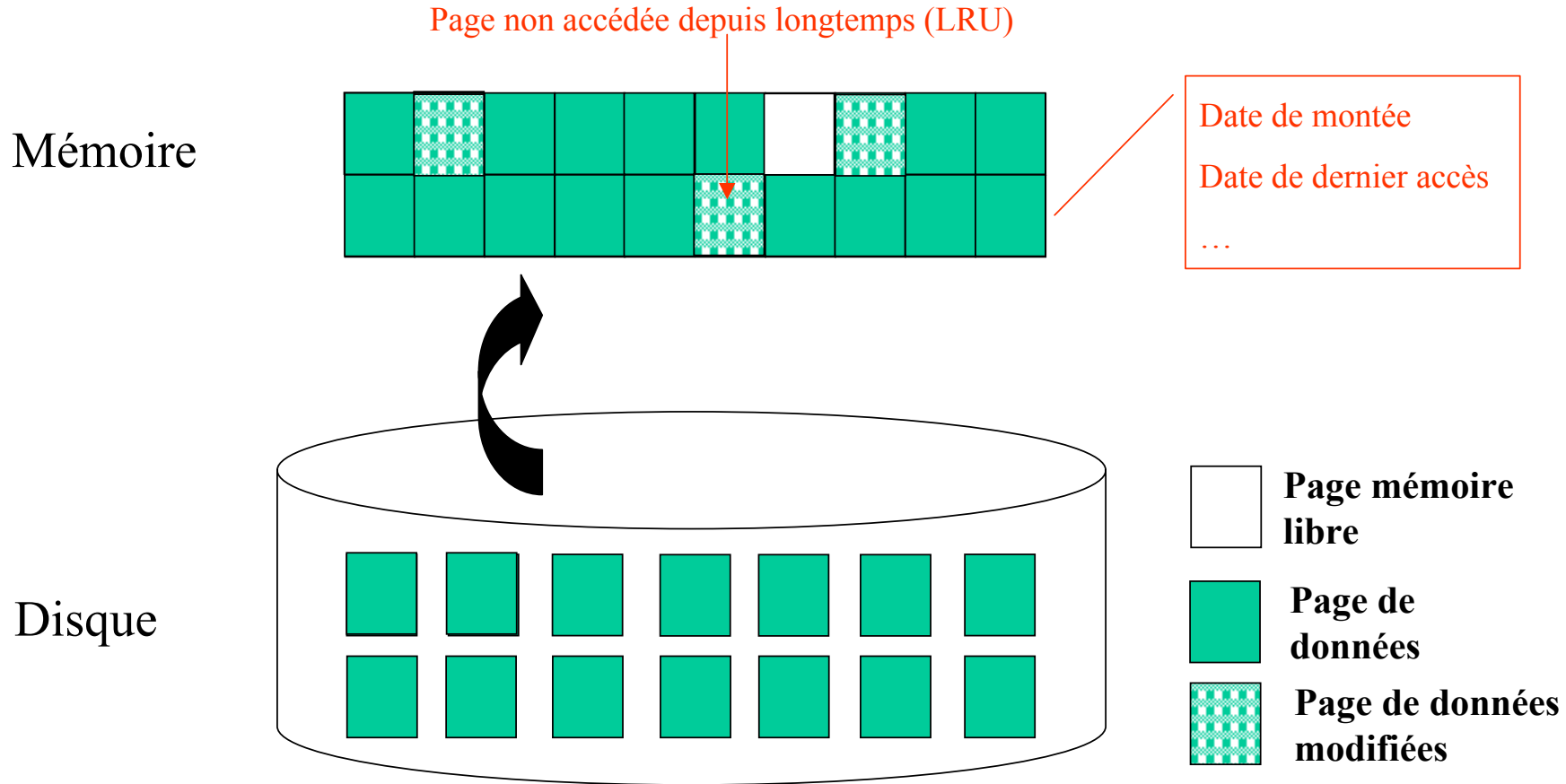


Disque

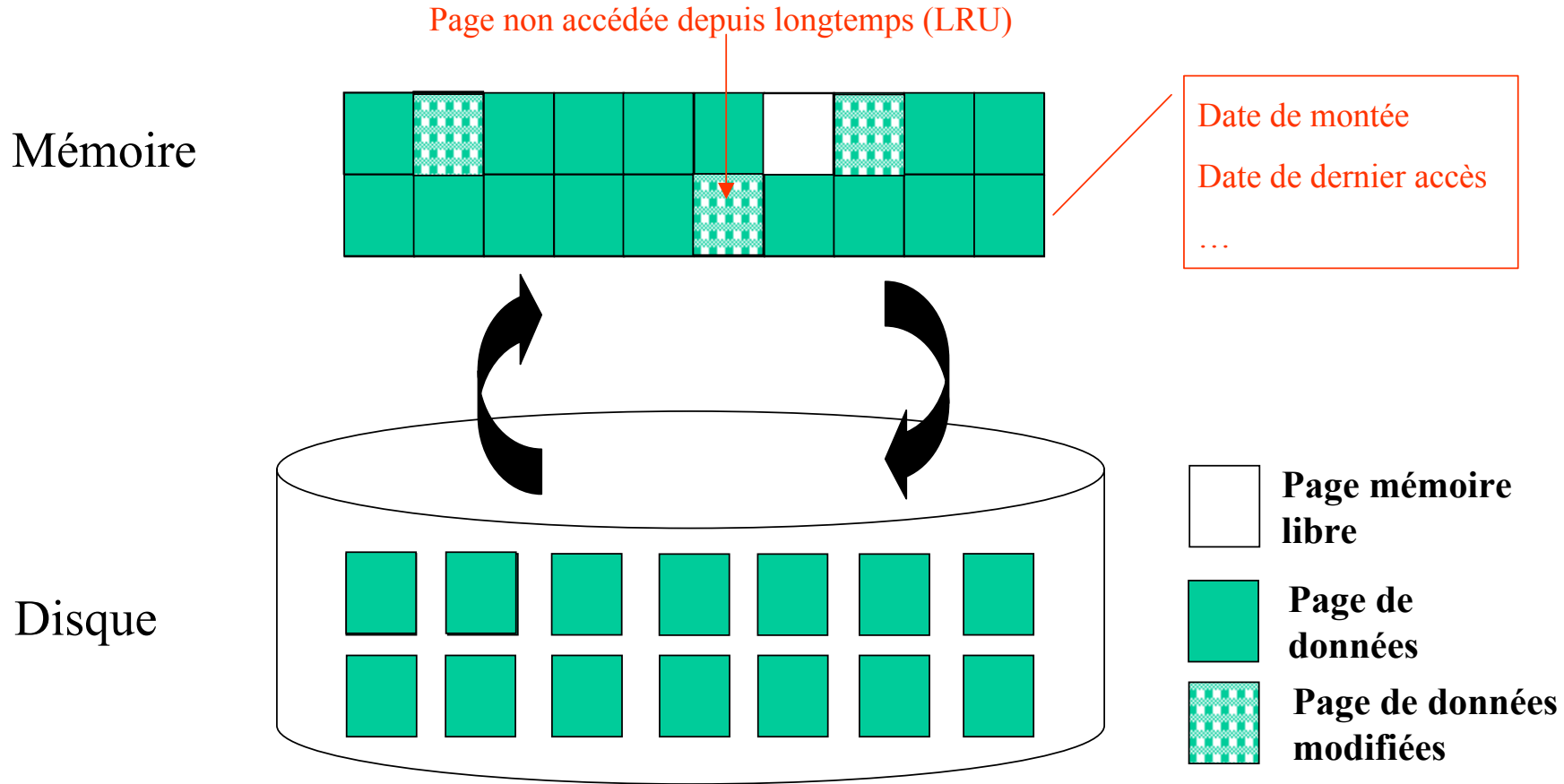


-  **Page mémoire libre**
-  **Page de données**
-  **Page de données modifiées**

# Gestionnaire de buffer

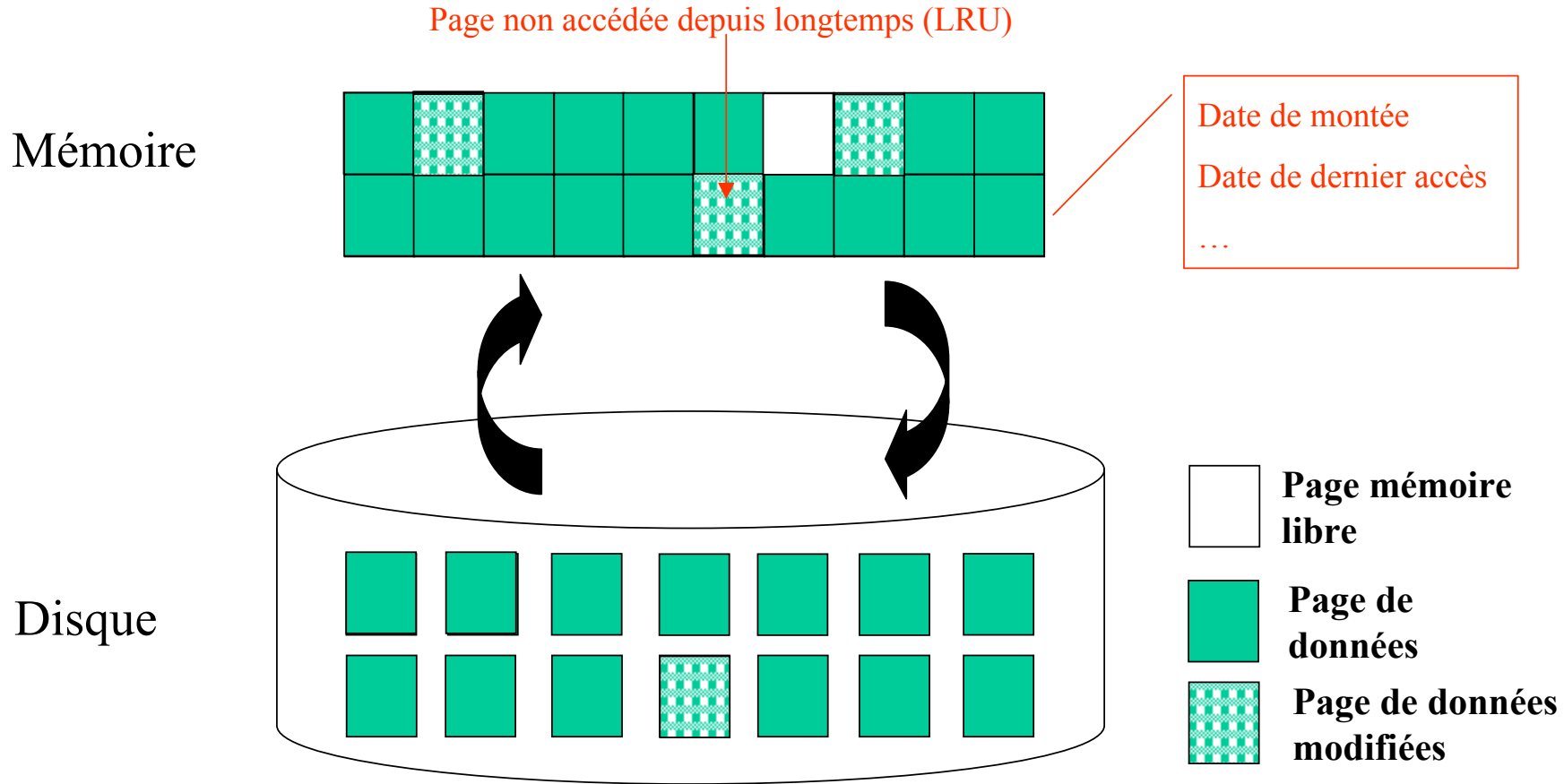


# Gestionnaire de buffer

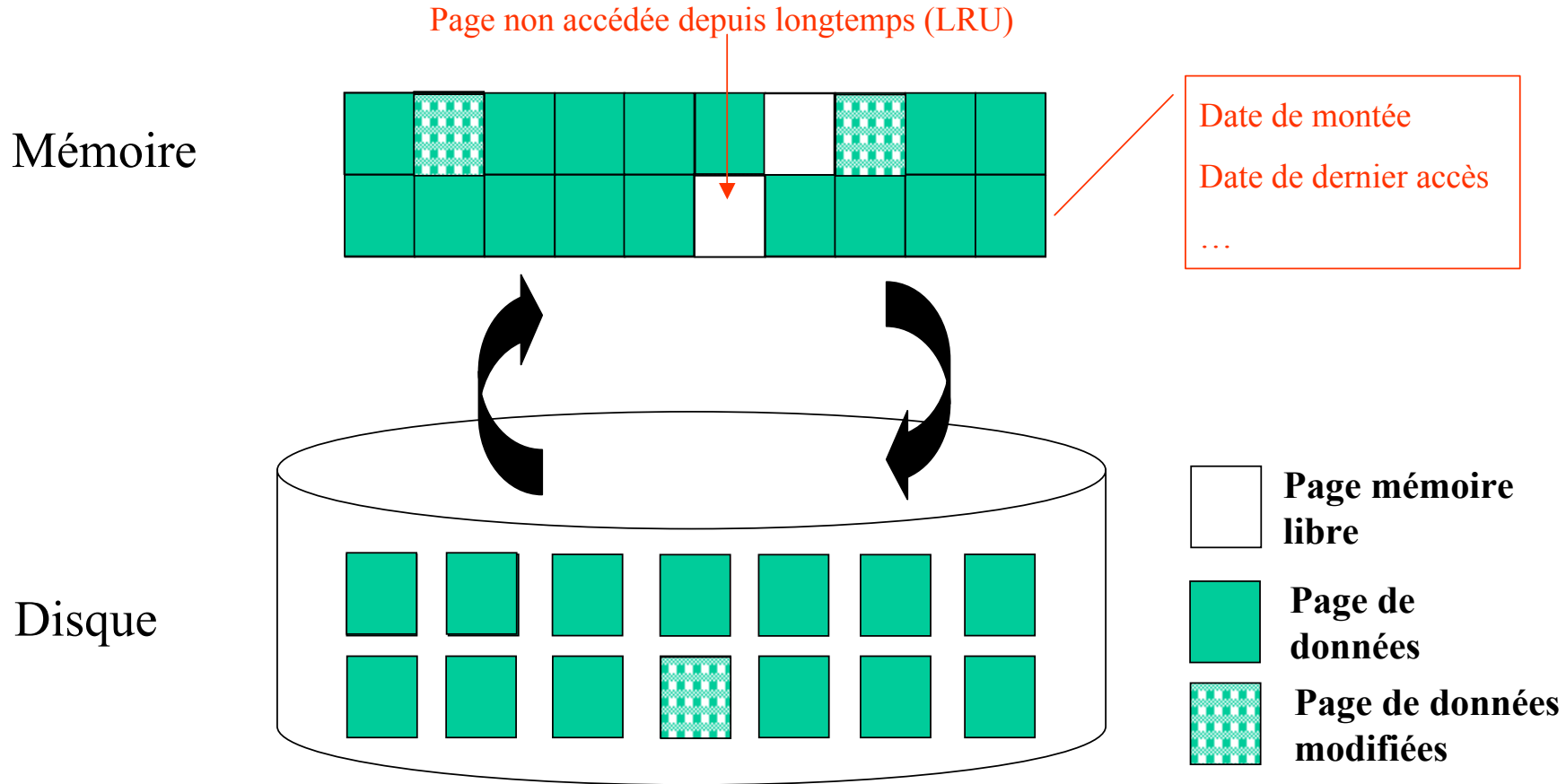




# Gestionnaire de buffer



# 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**

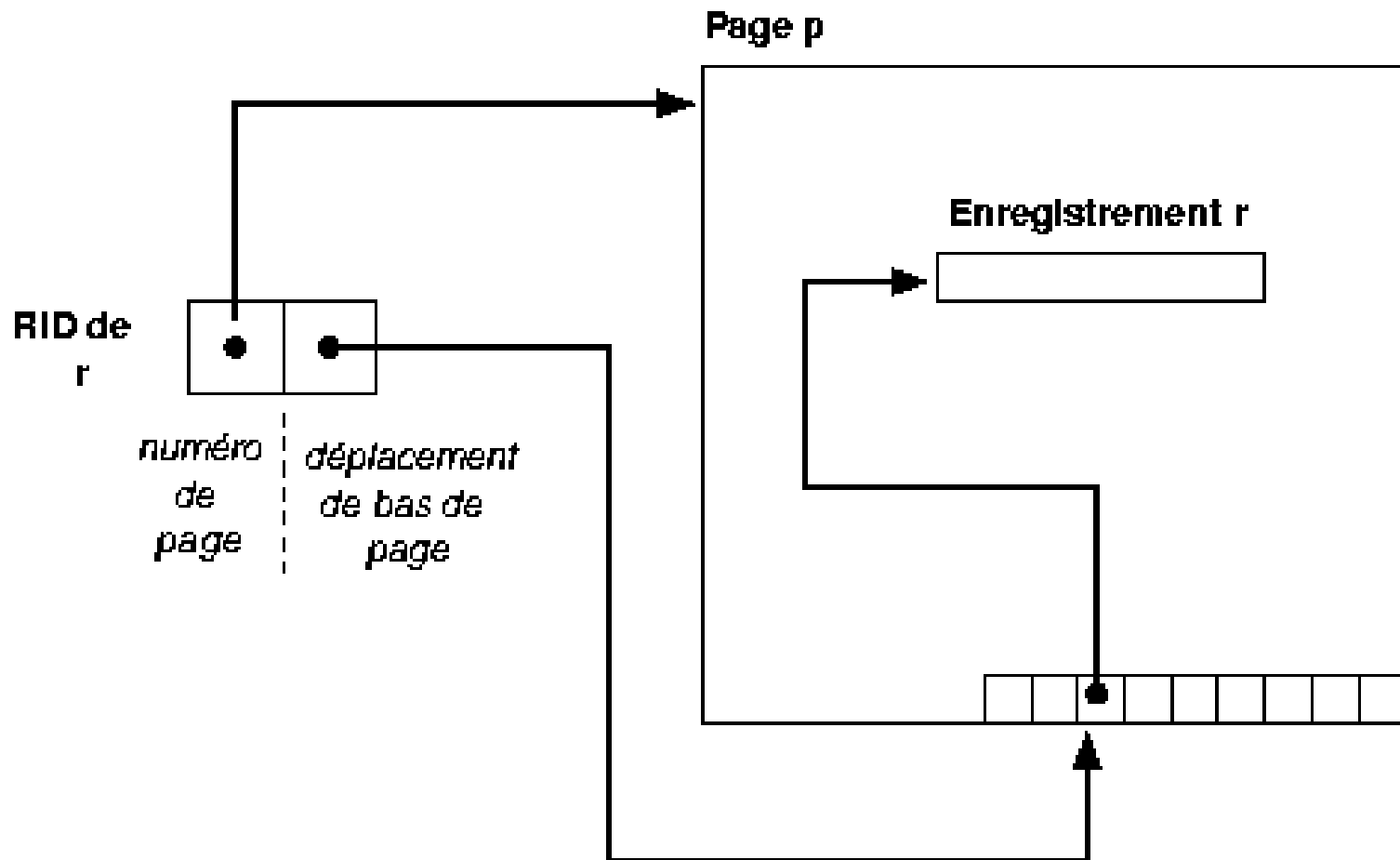
# Chap. II - Organisation des données

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

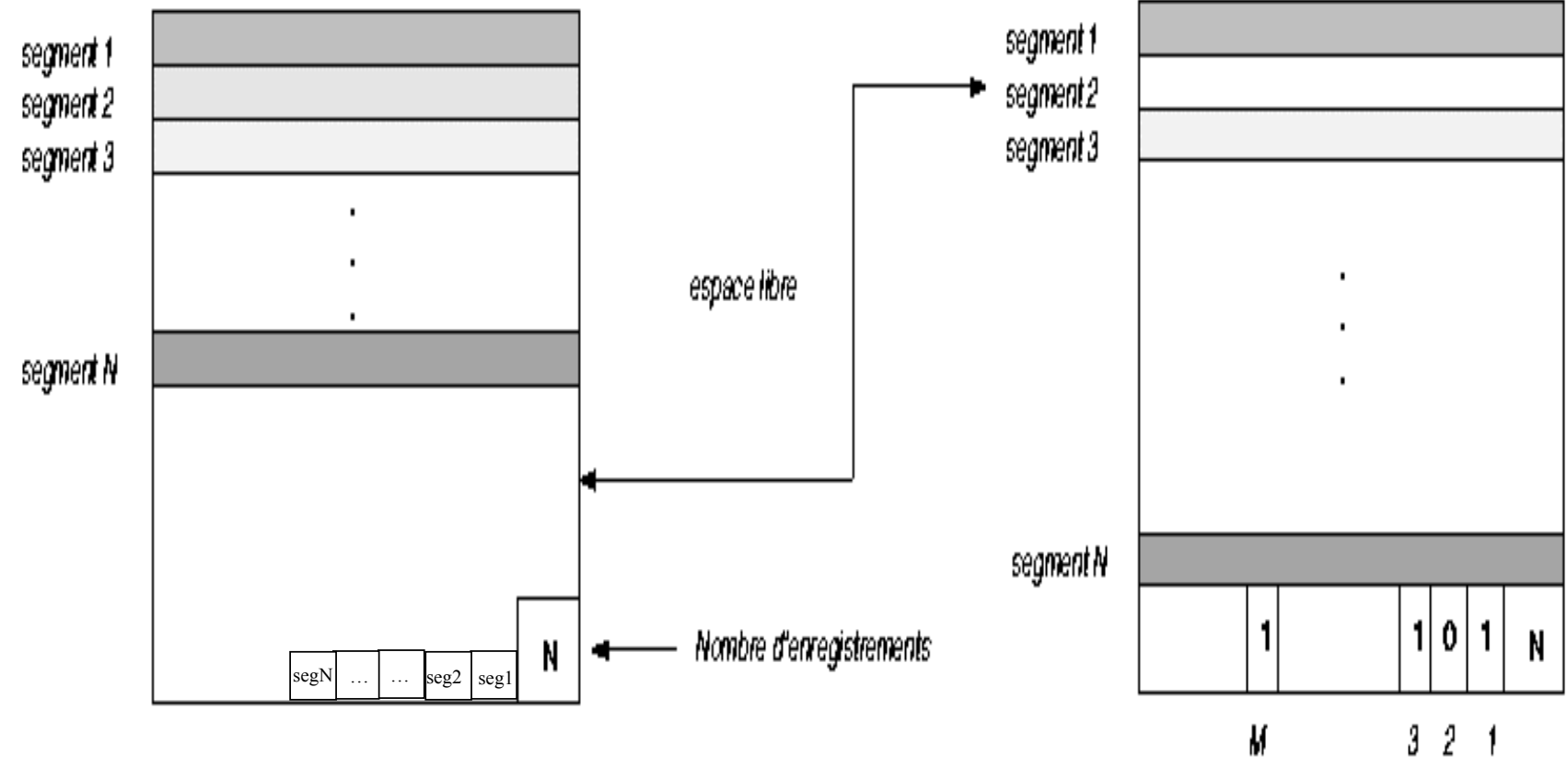
# Gestion des fichiers

- 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

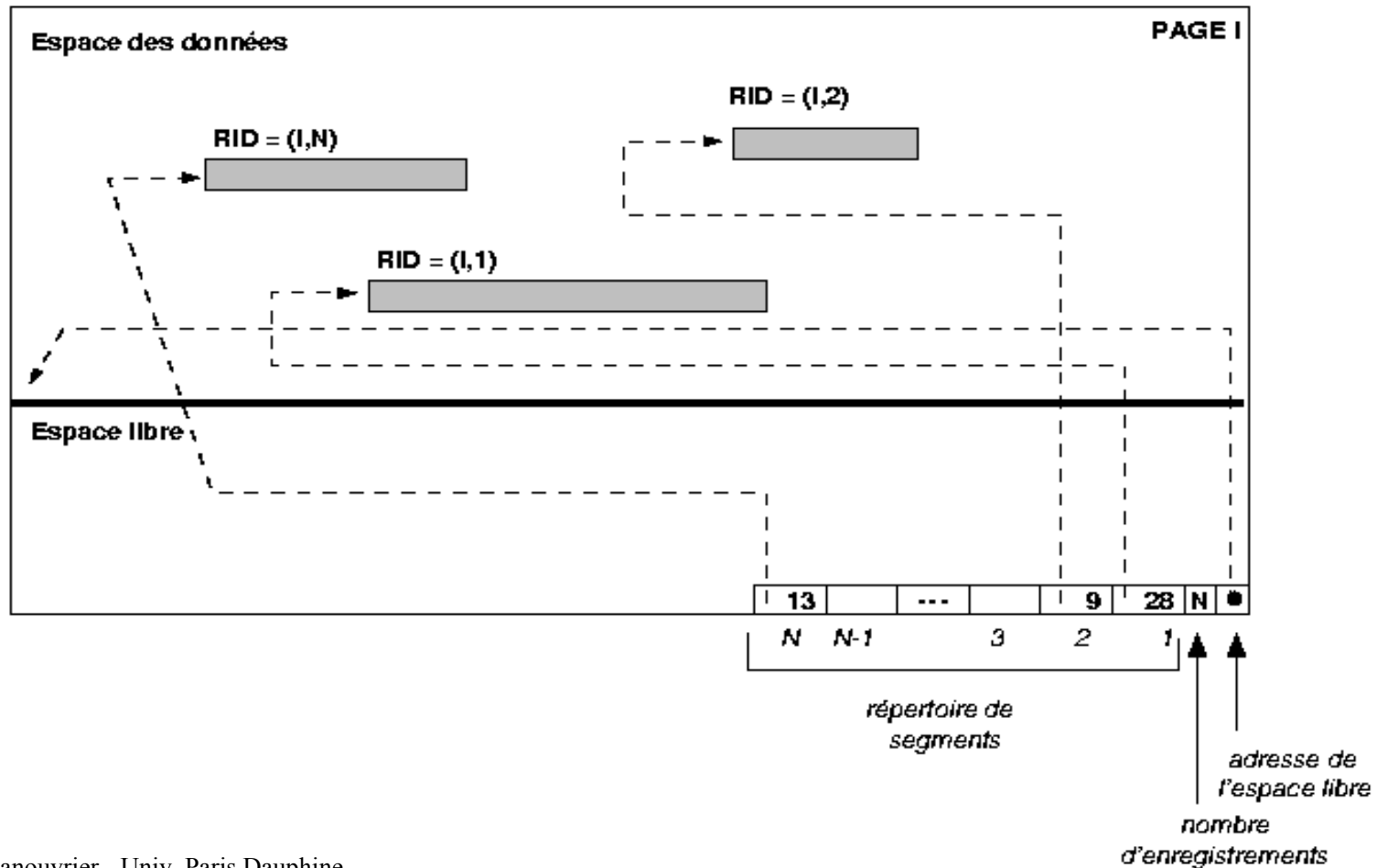
# Identification des enregistrements



# 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

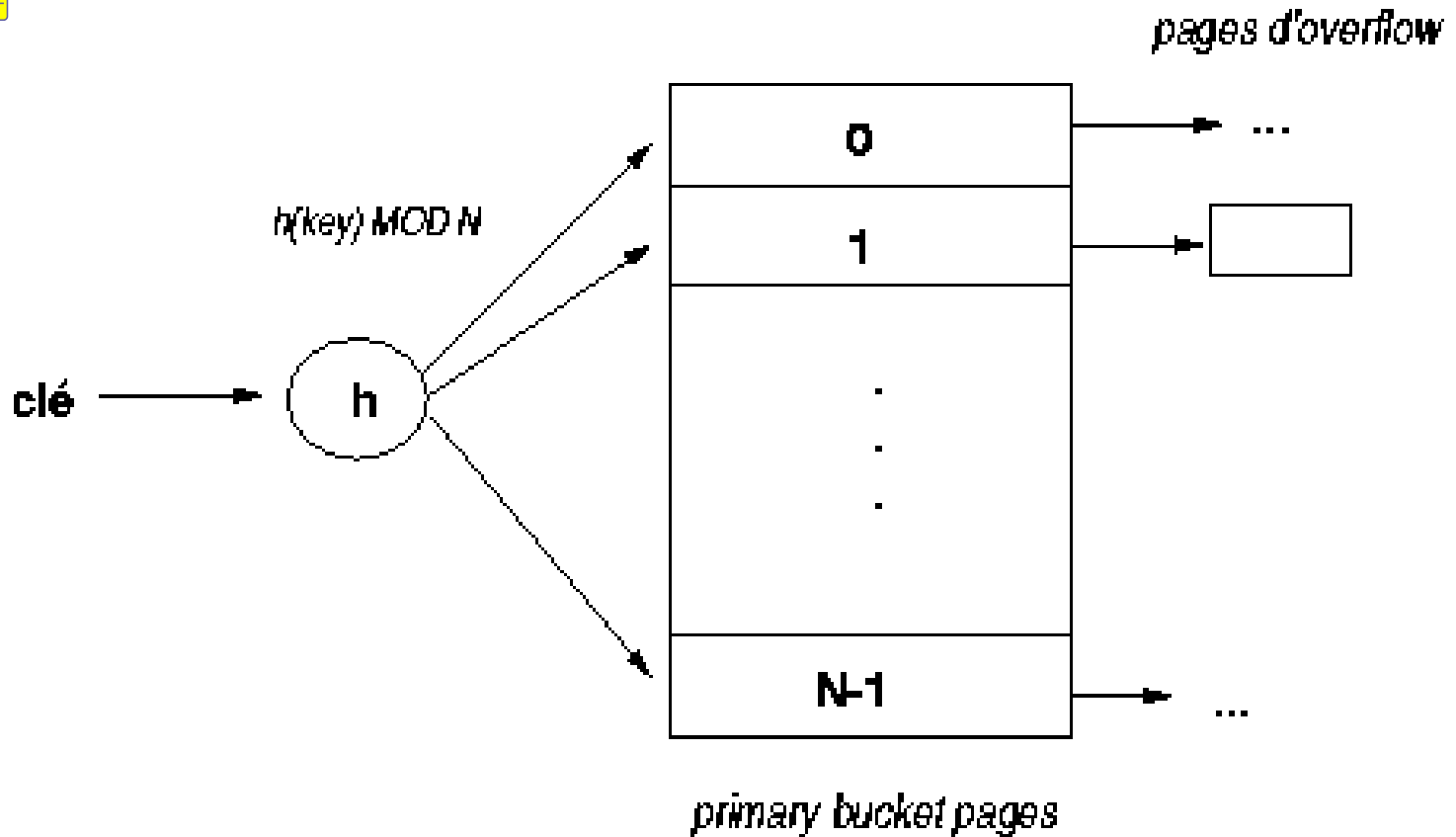
# Fichier de hachage

- **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**

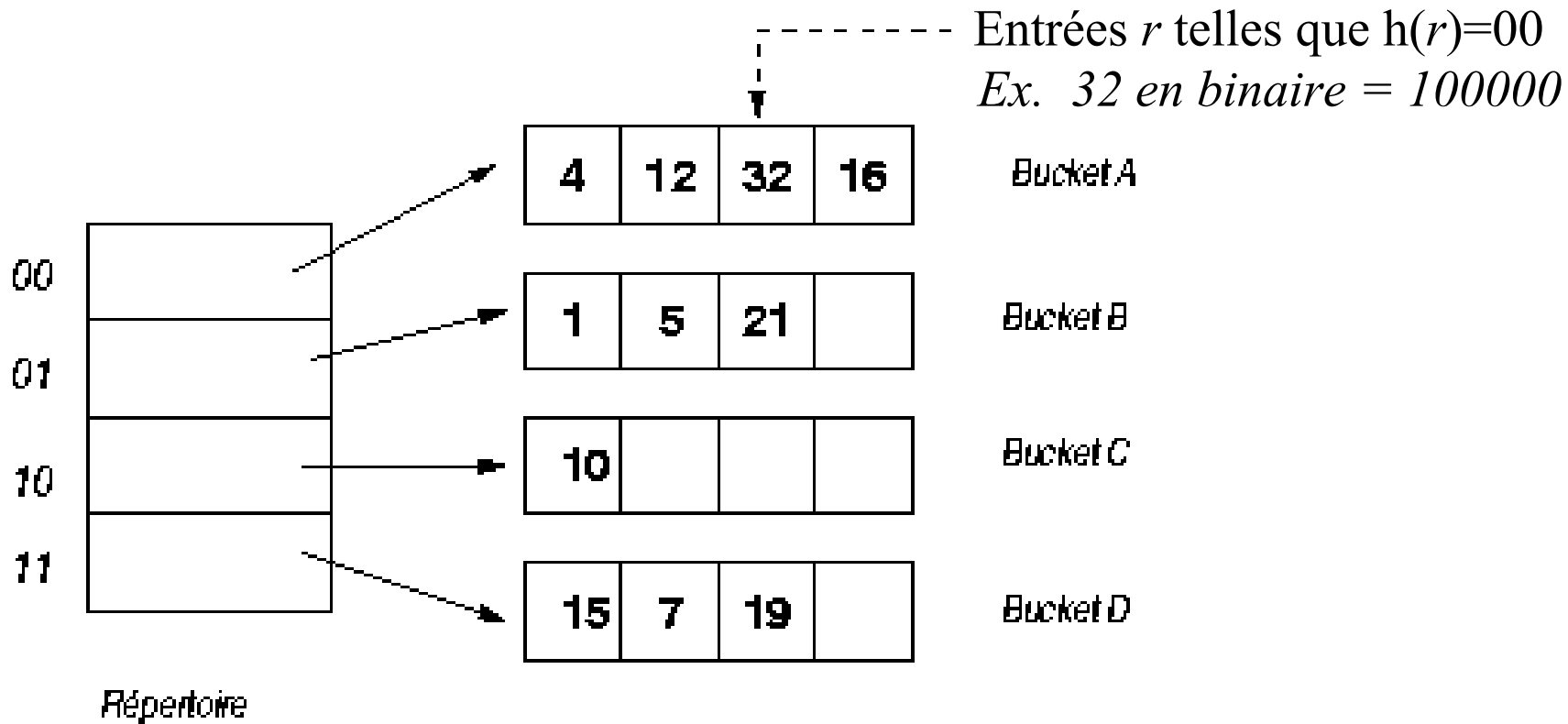
# Hachage statique



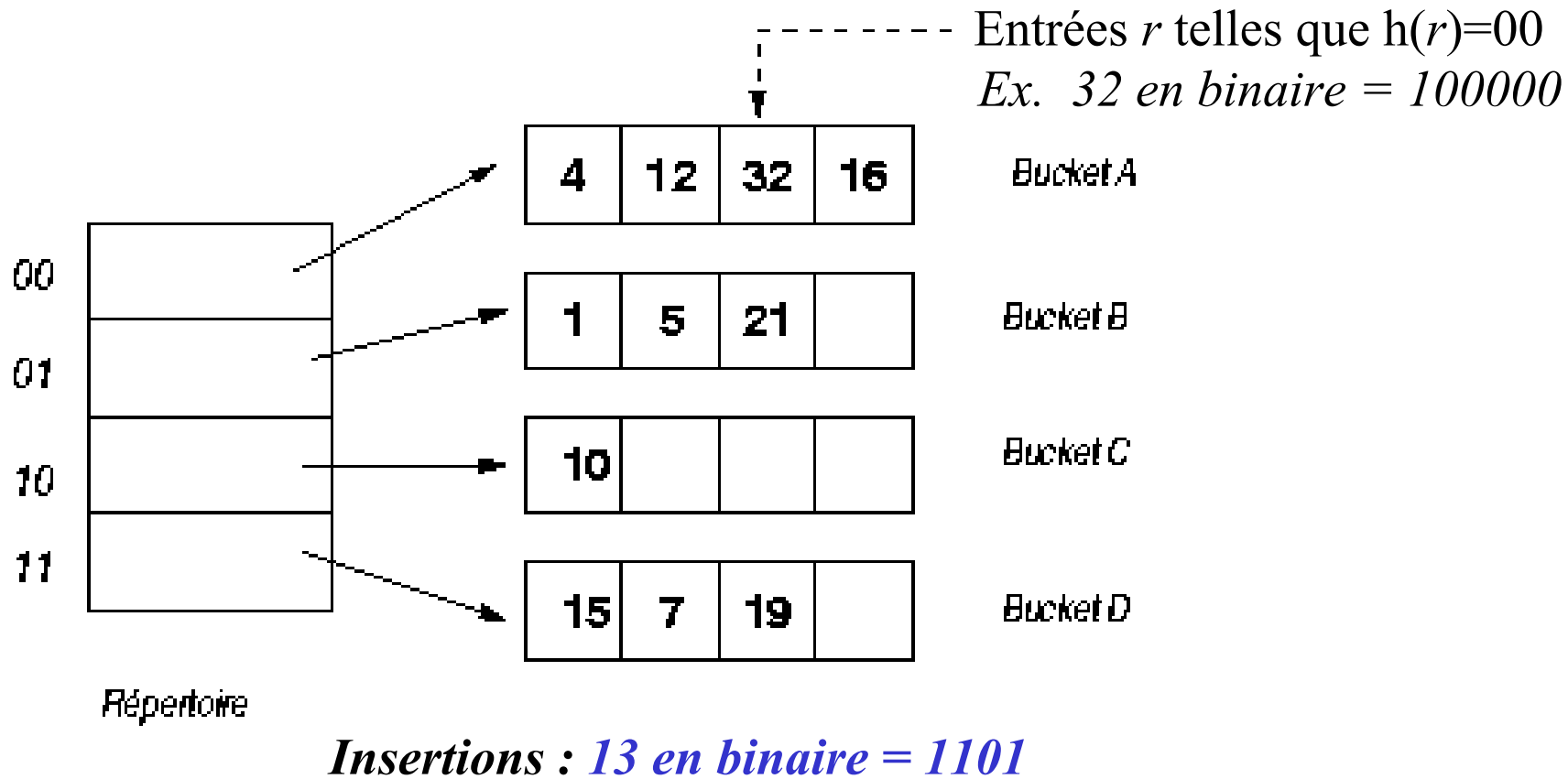
# Gestion des collisions

- *Open adressing* : 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

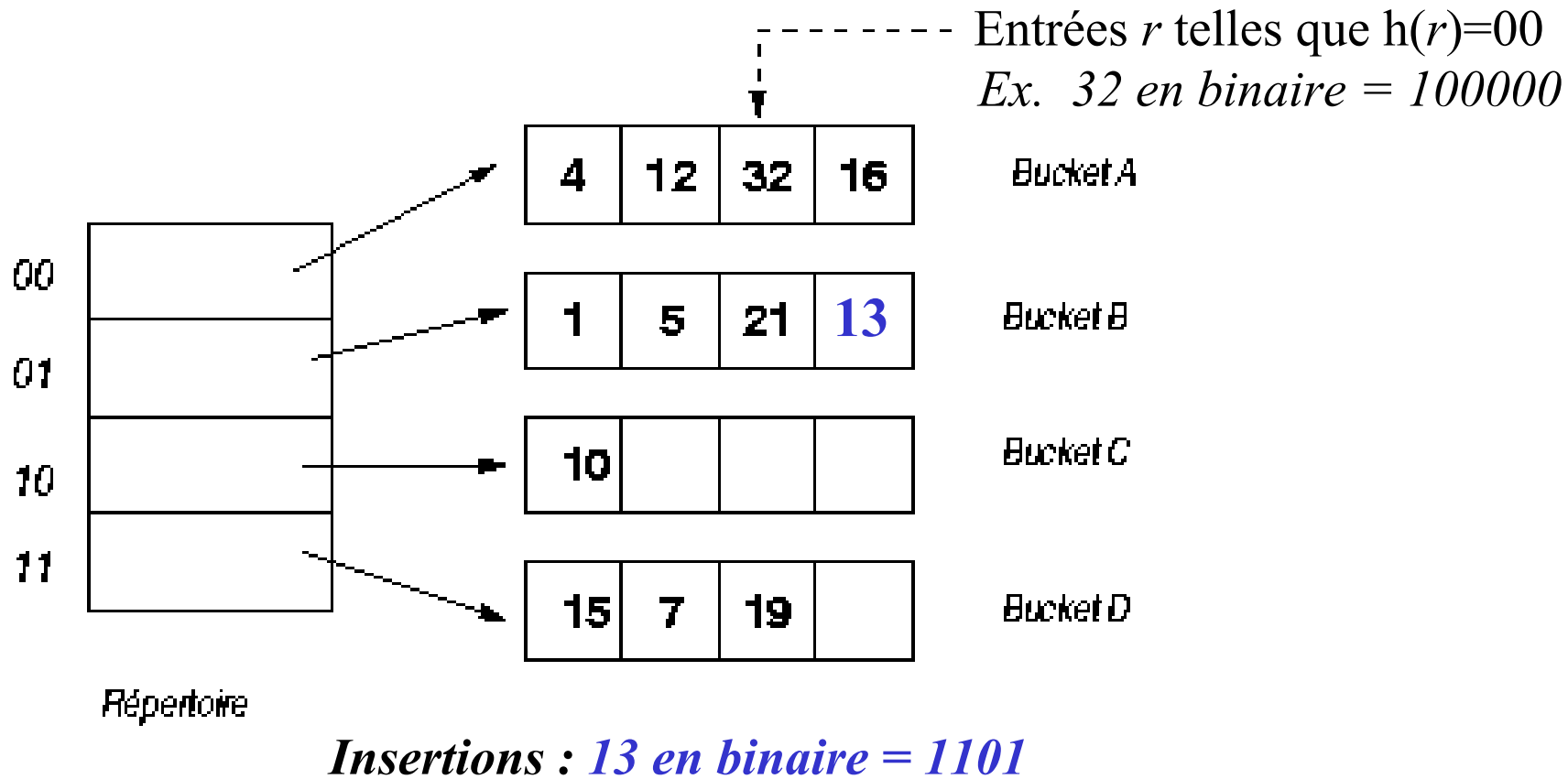
# Hachage extensible (1/2)



# Hachage extensible (1/2)

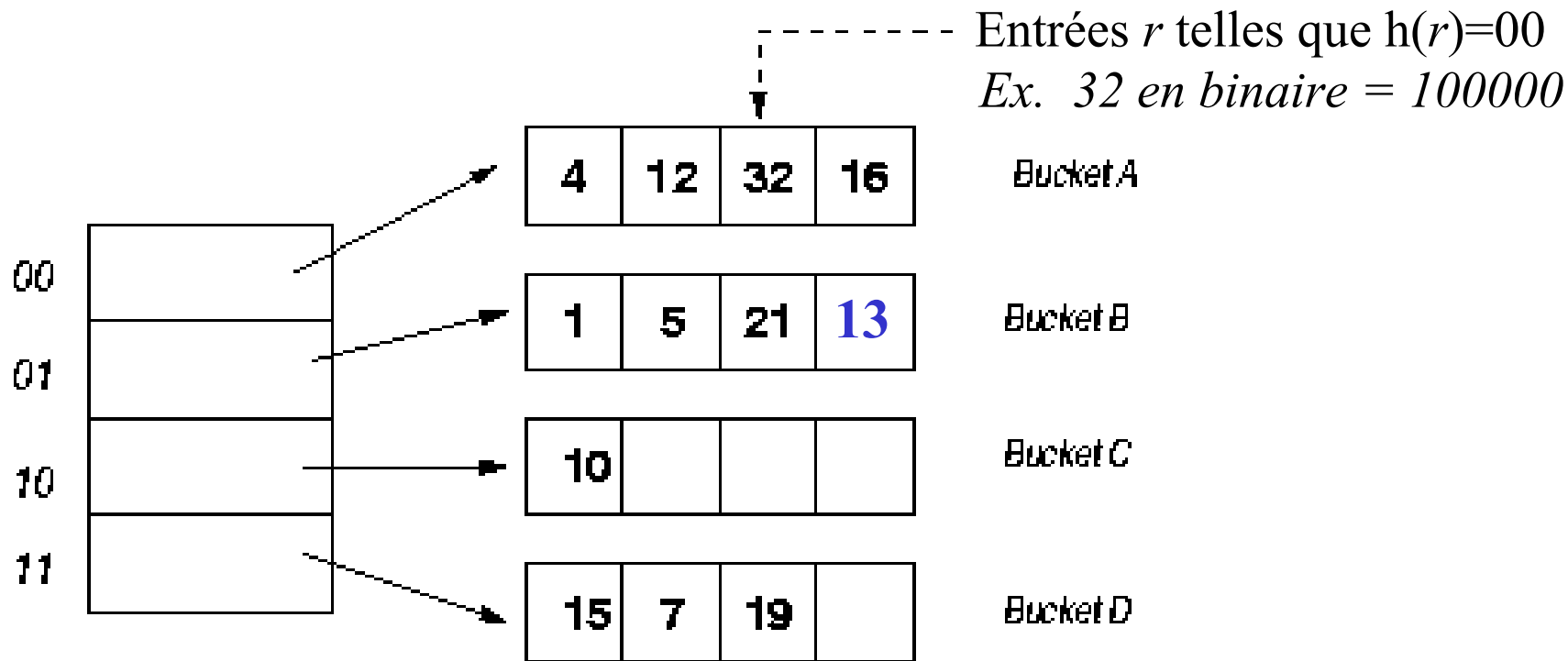


# Hachage extensible (1/2)





# Hachage extensible (1/2)



*Insertions : 13 en binaire = 1101*

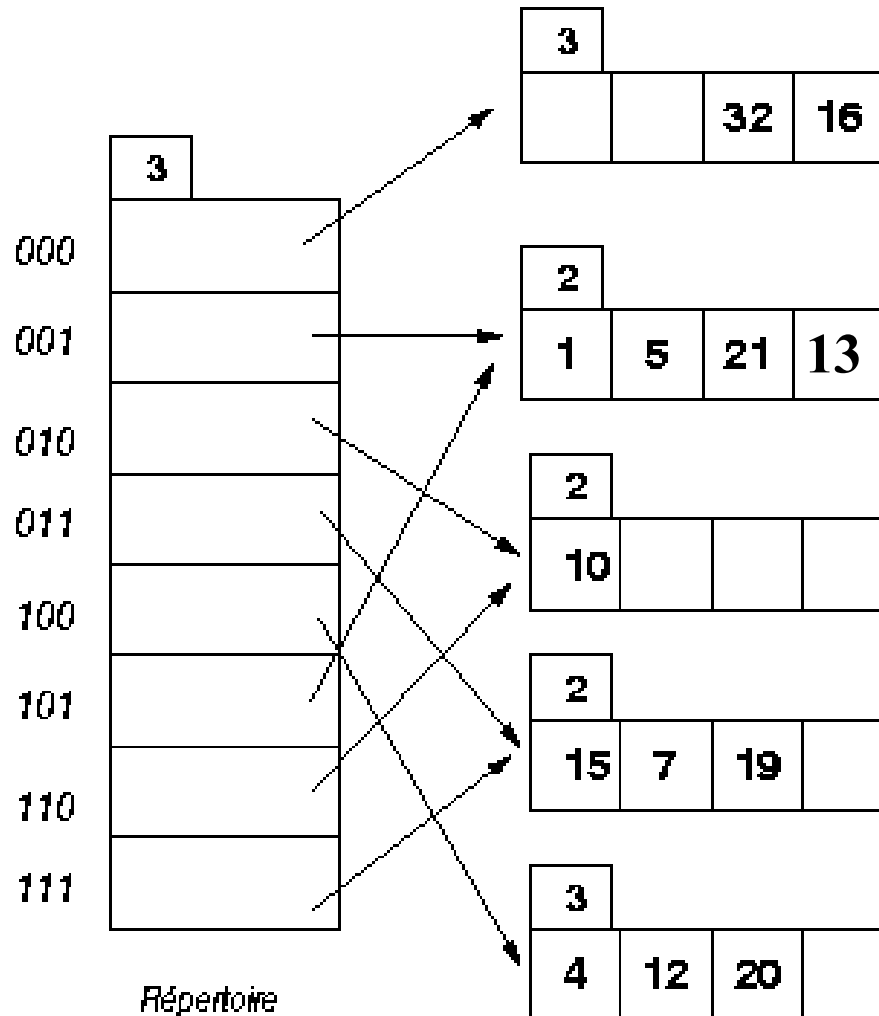
*20 en binaire = 10100 mais bucket A plein*

# Hachage extensible (2/2)

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

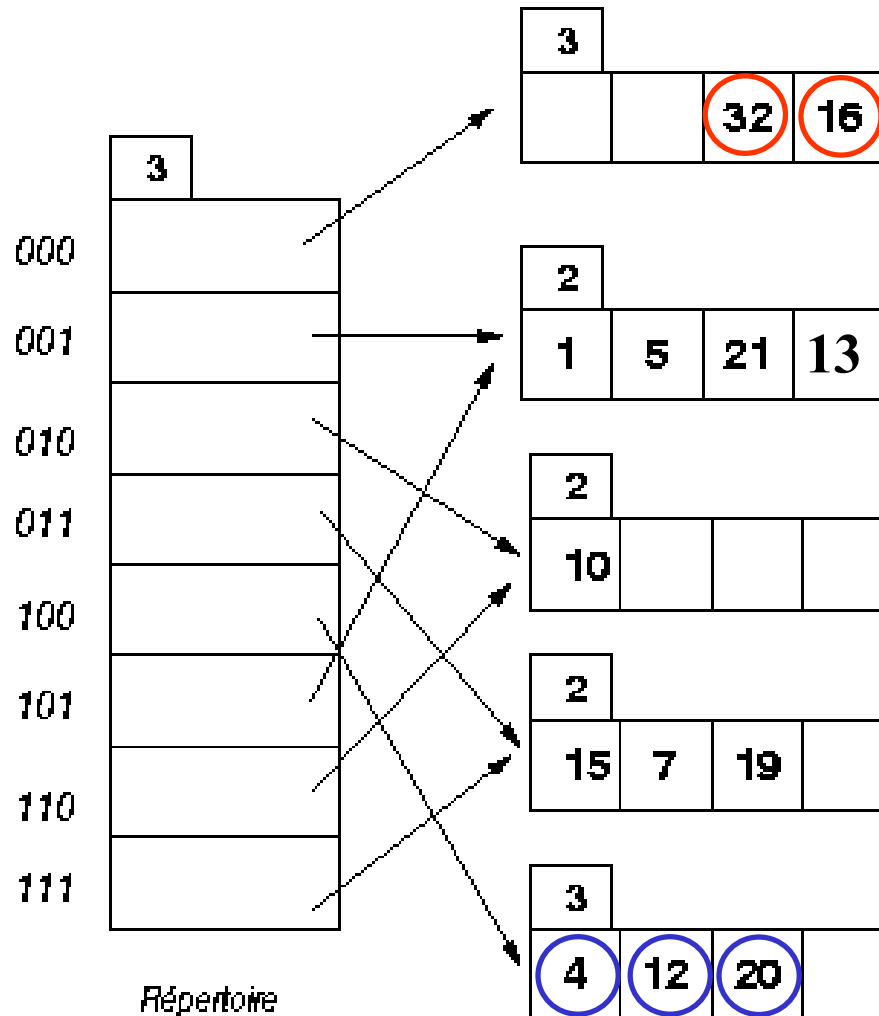
# Hachage extensible (2/2)

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



$h(4)=100$

$h(12)=100$

$h(16) = 000$

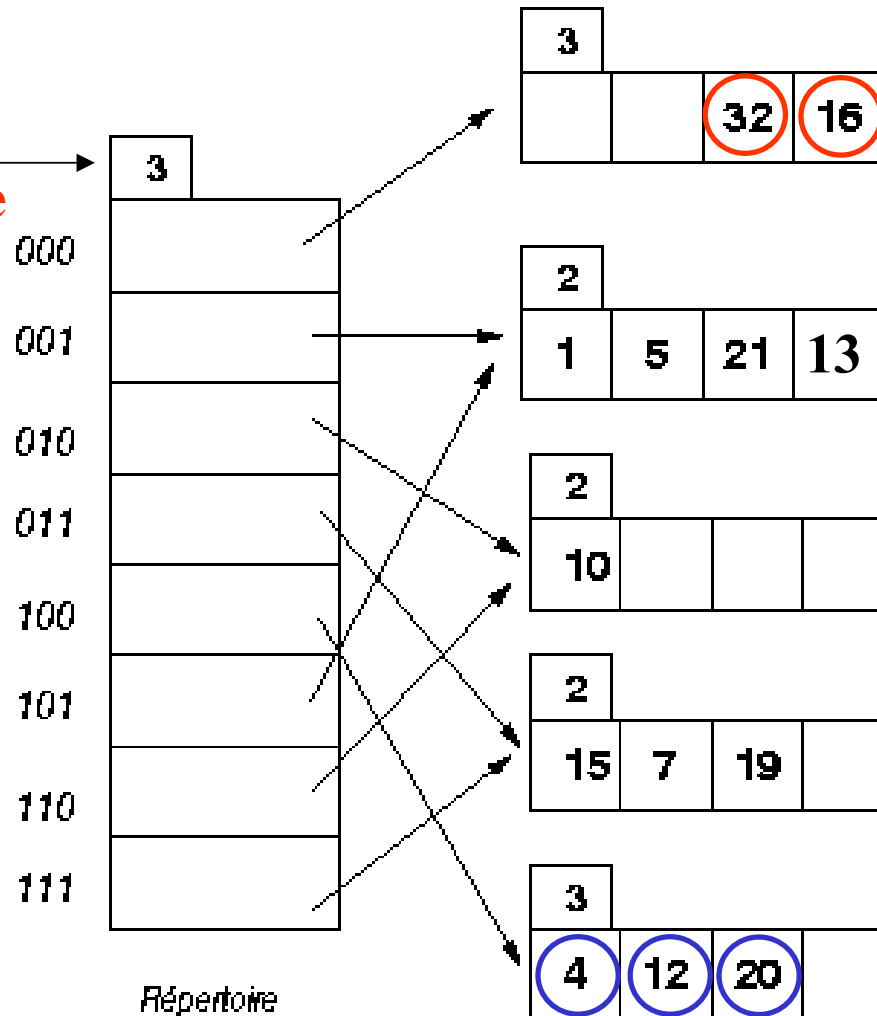
$h(32)=000$

$h(20) = 100$

# Hachage extensible (2/2)

Taille globale

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



$$h(4)=100$$

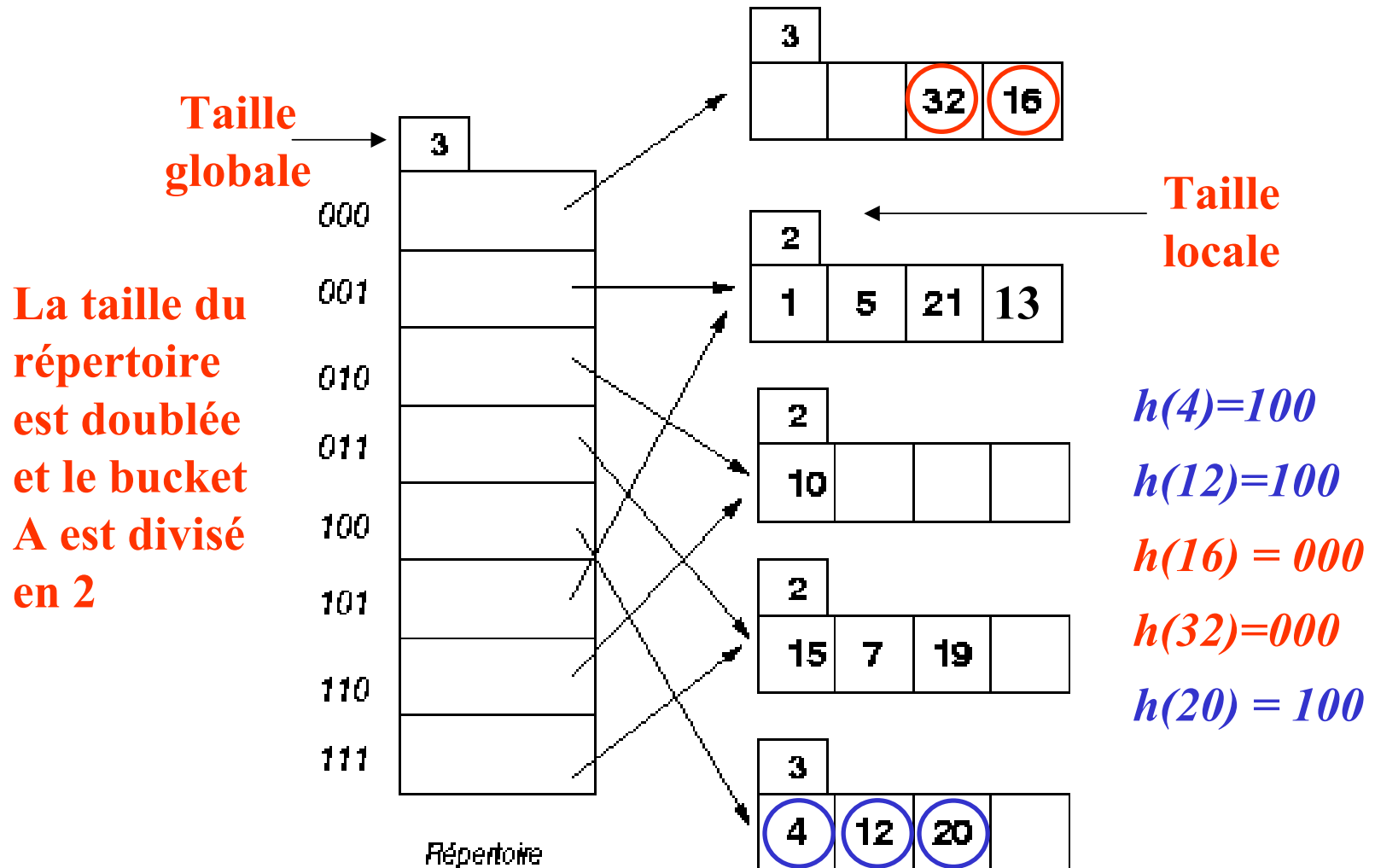
$$h(12)=100$$

$$h(16) = 000$$

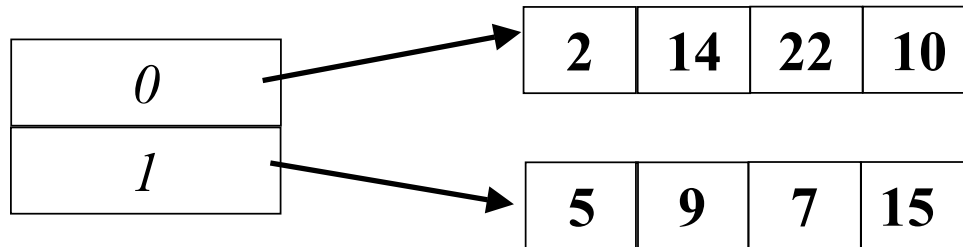
$$h(32)=000$$

$$h(20) = 100$$

# Hachage extensible (2/2)



# Autre exemple de hachage extensible (1/4)



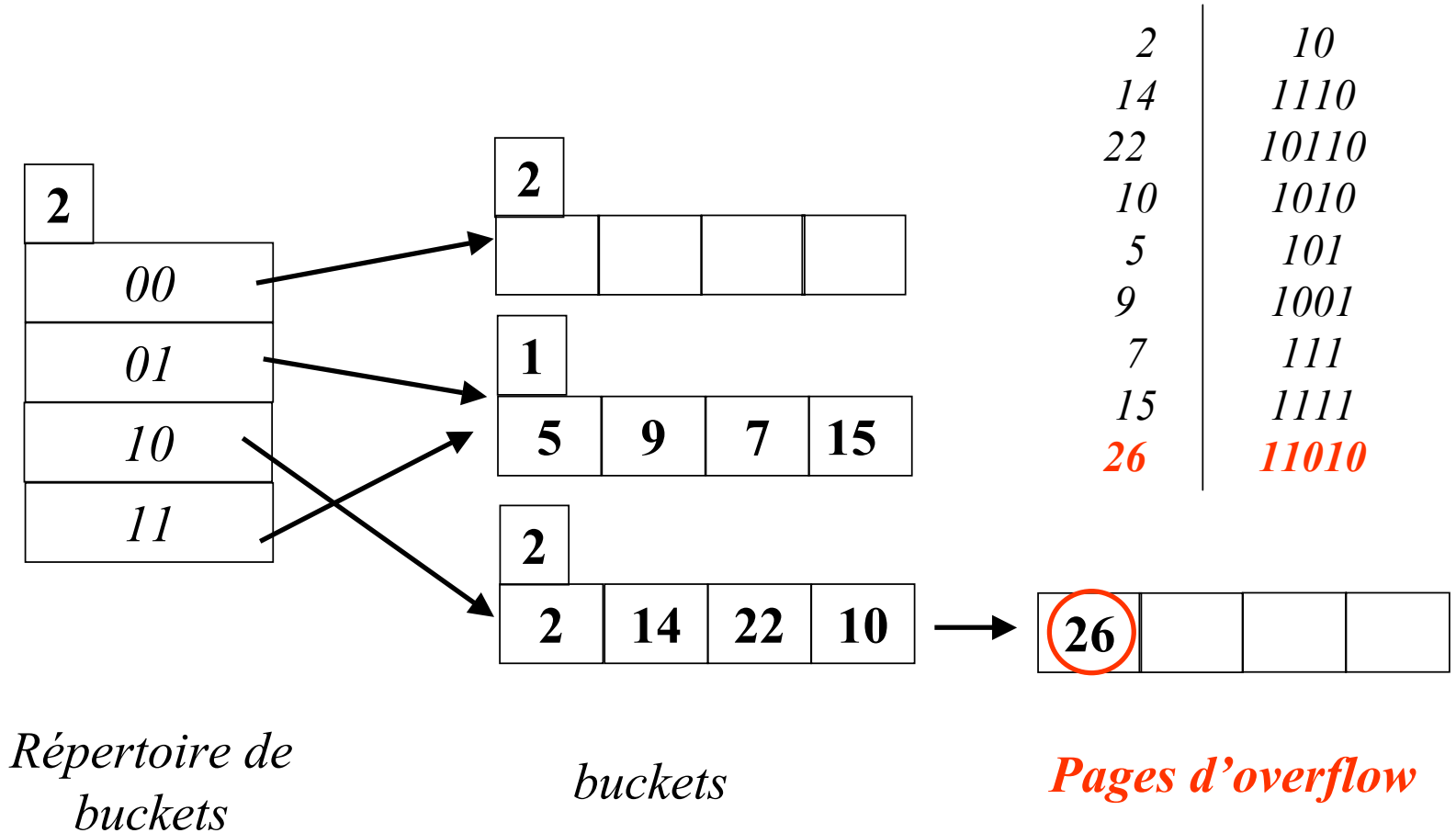
*Répertoire de  
buckets*

*buckets*

**On ajoute l'entrée 26 (11010)**

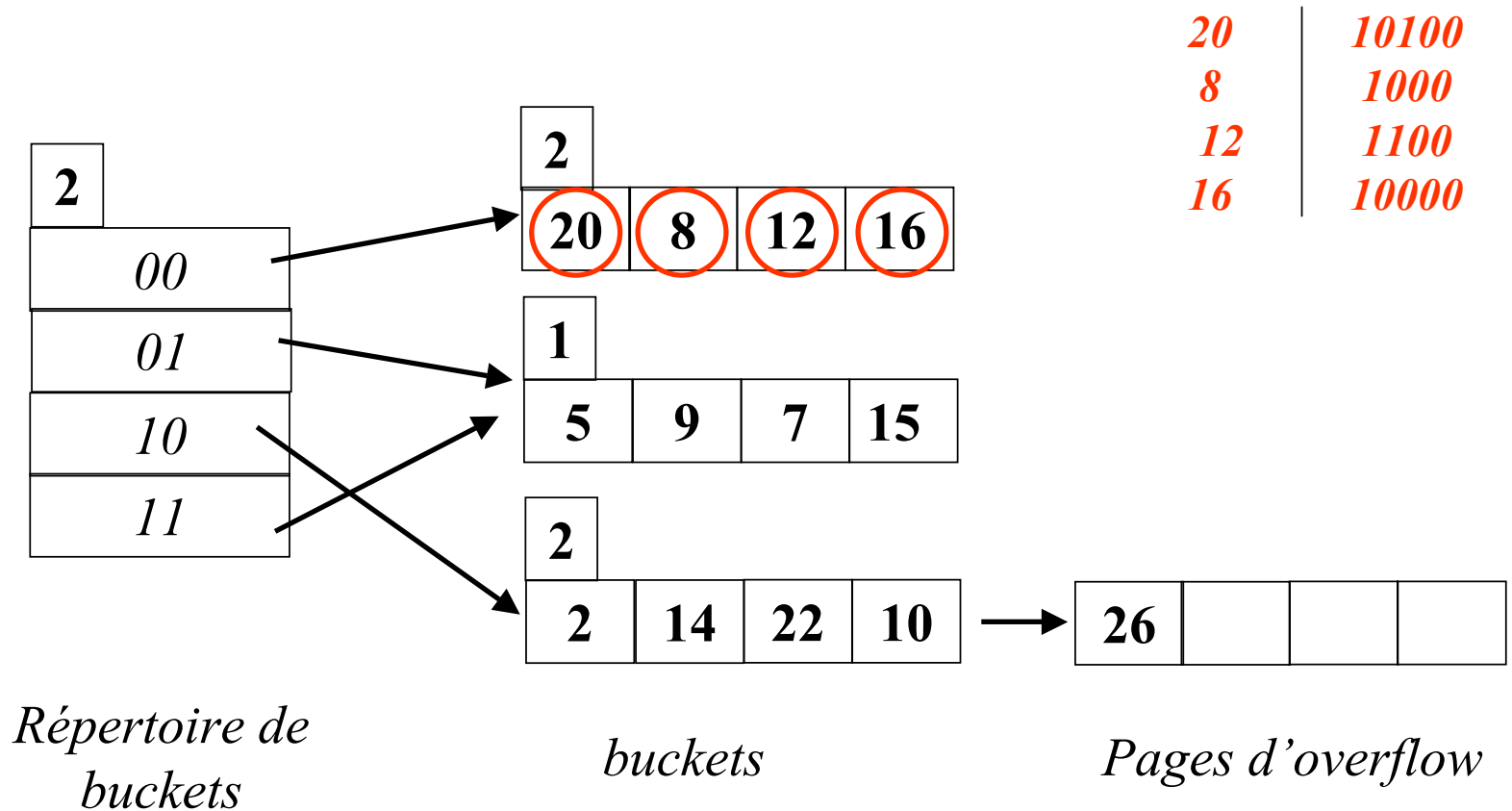
2	10
14	1110
22	10110
10	1010
5	101
9	1001
7	111
15	1111

# Autre exemple de hachage extensible (2/4)

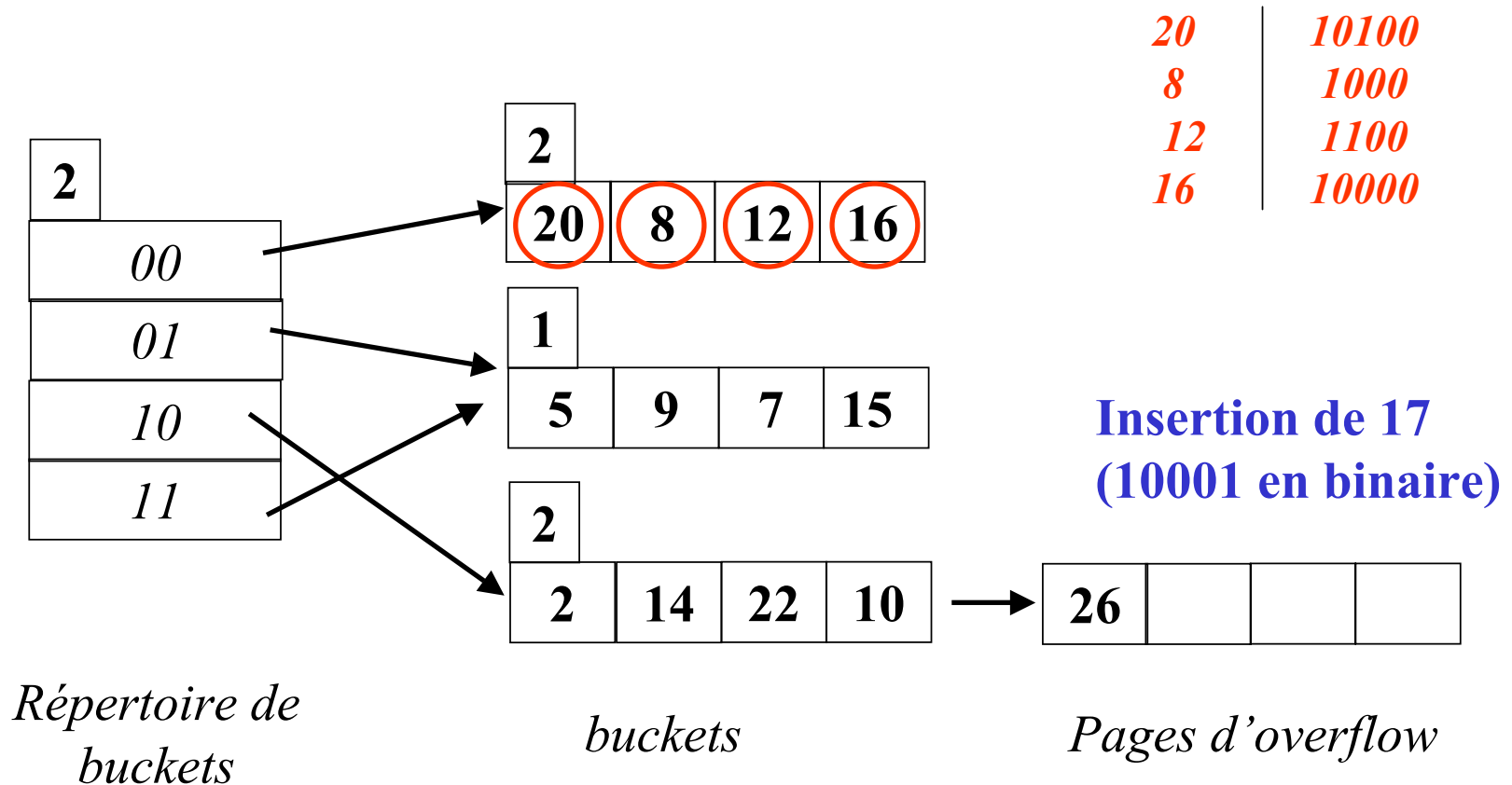




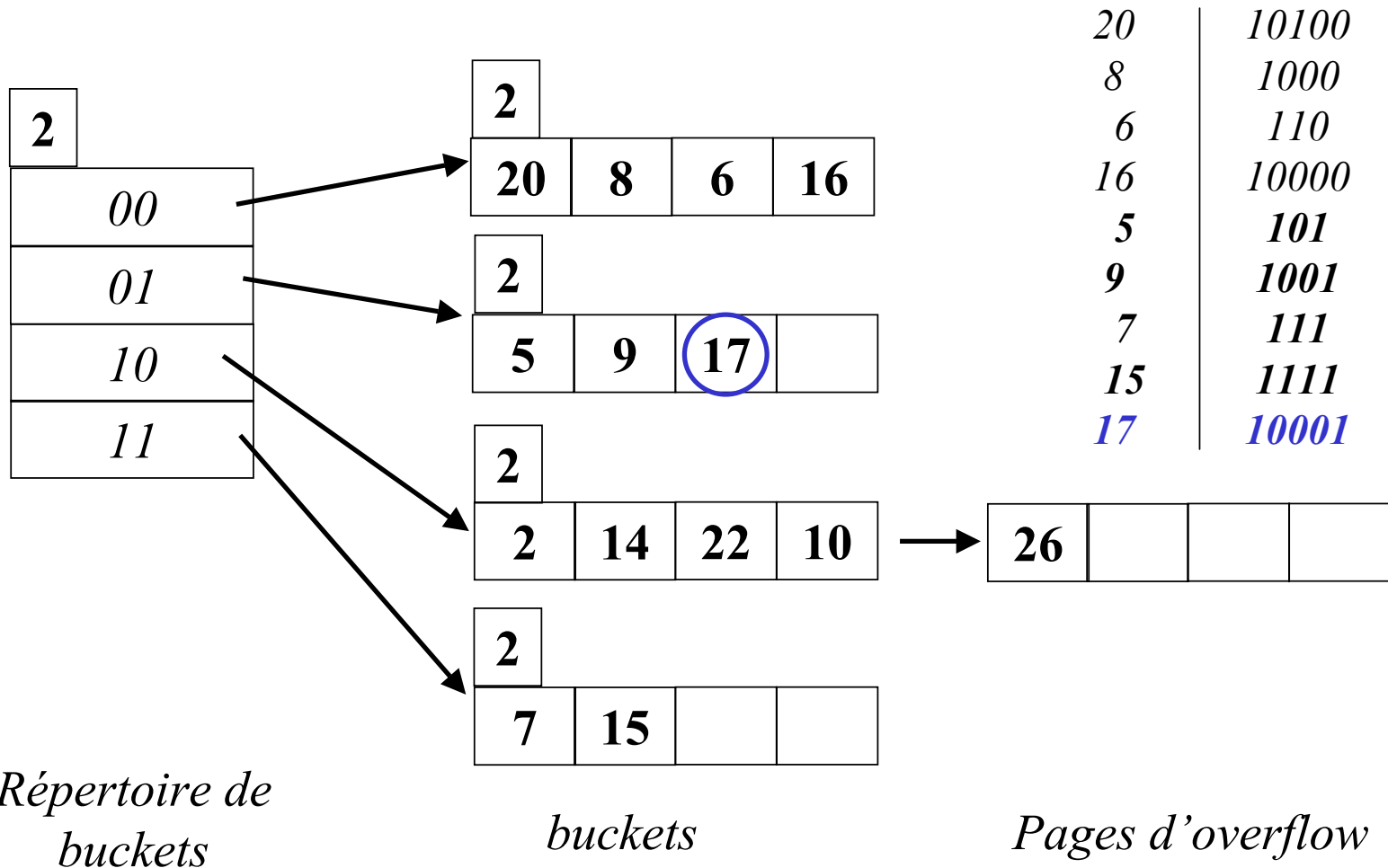
# Autre exemple de hachage extensible (3/4)



# Autre exemple de hachage extensible (3/4)



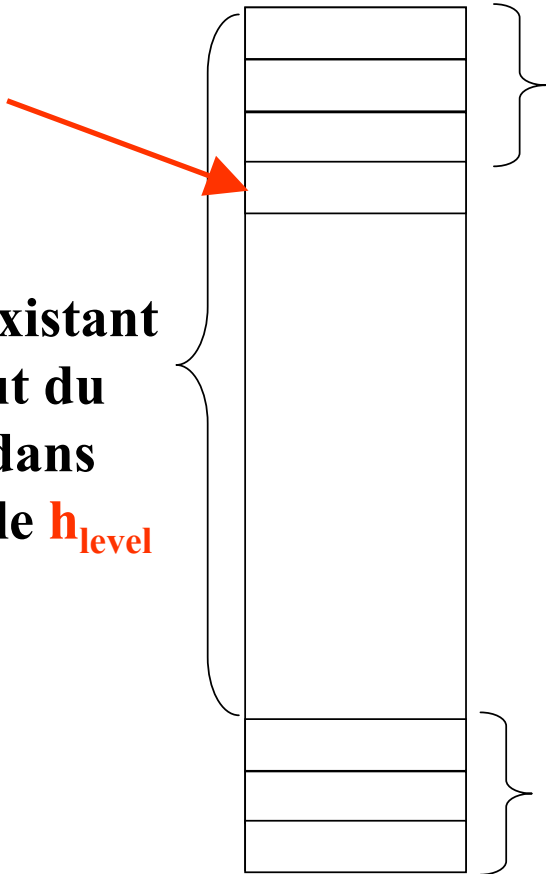
# Autre exemple de hachage extensible (4/4)



# Hachage linéaire (1/6)

Prochain  
bucket à  
diviser

Buckets existant  
au début du  
*round* dans  
l'intervalle  $h_{\text{level}}$



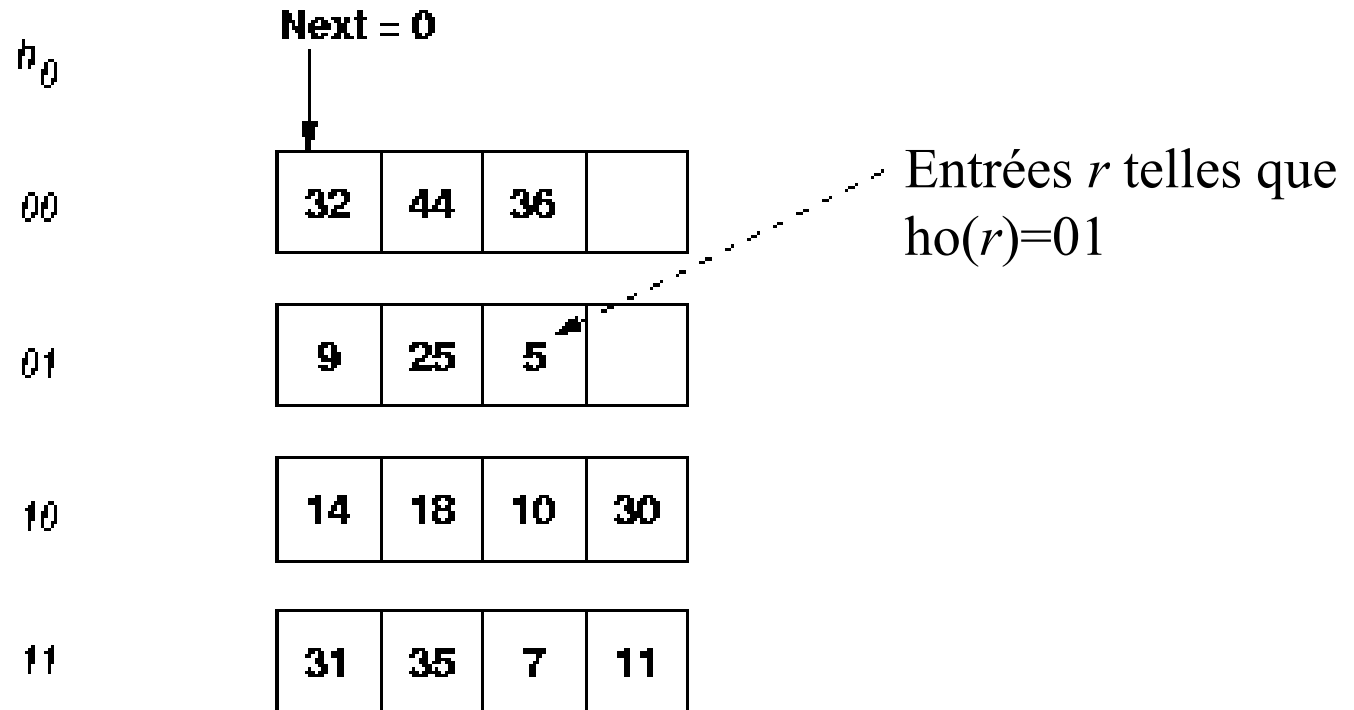
Buckets dédoublés pendant le *round*

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)

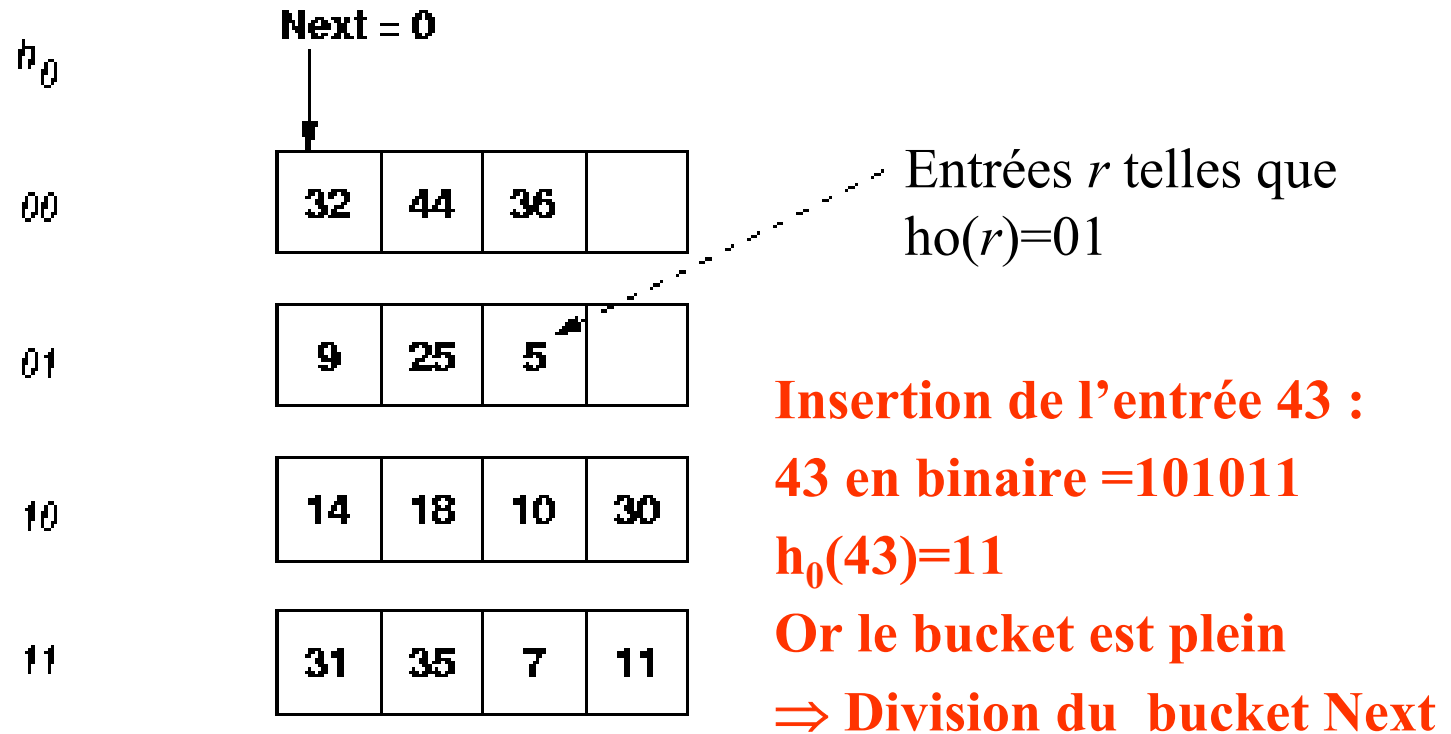
level = 0, N=4



Four info

# Hachage linéaire (2/6)

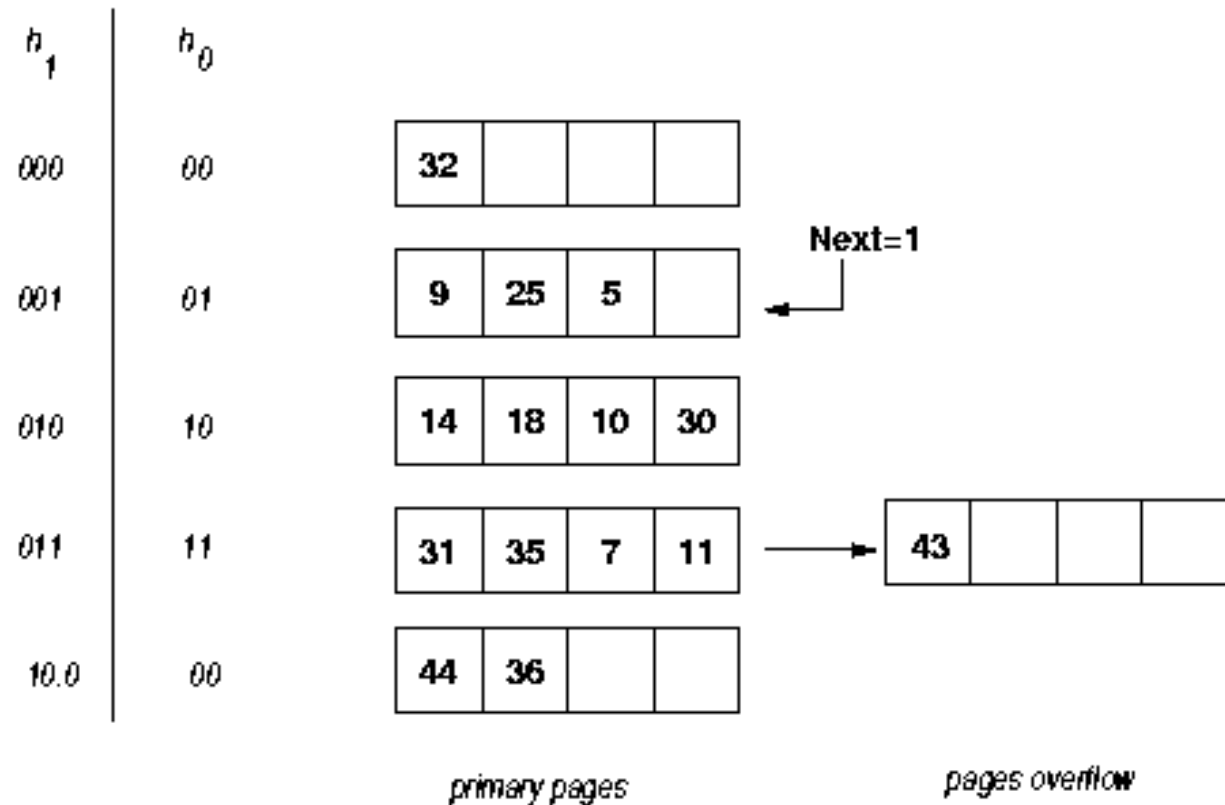
level = 0, N=4



Four info

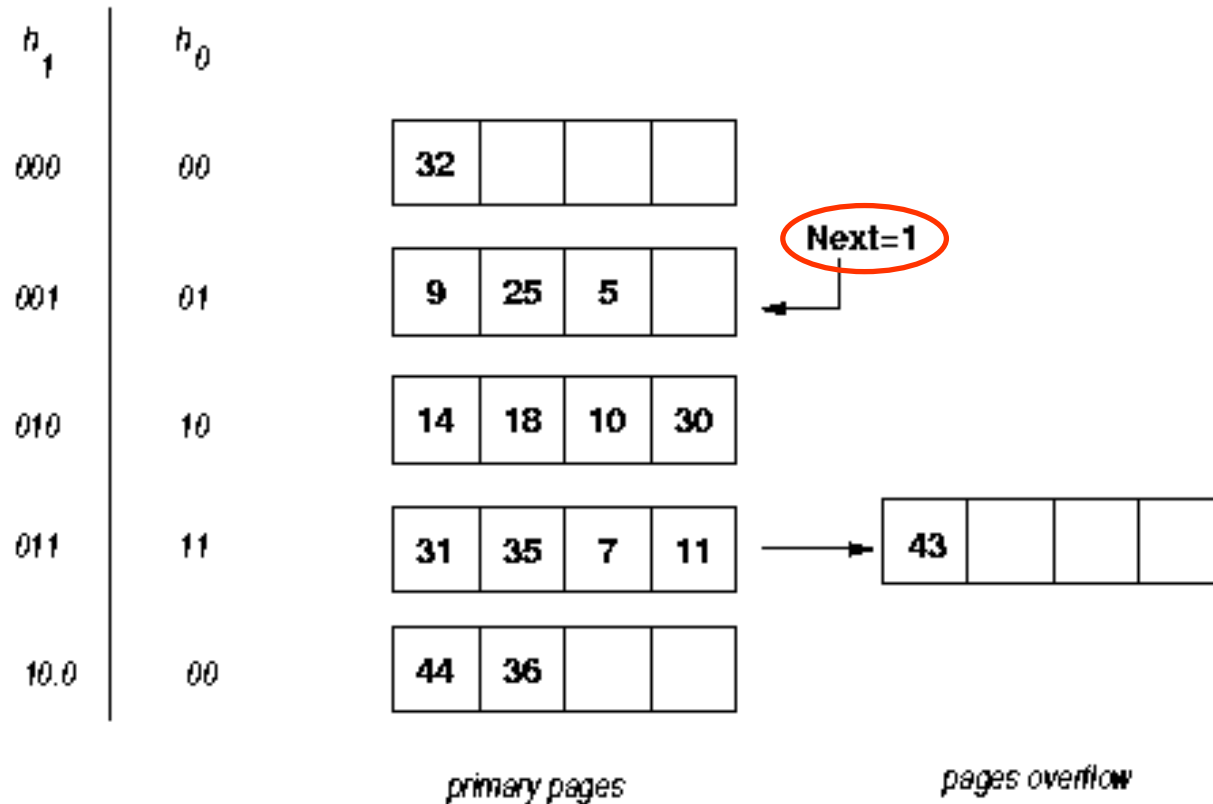
# Hachage linéaire (3/6)

level = 0, N=4



# Hachage linéaire (3/6)

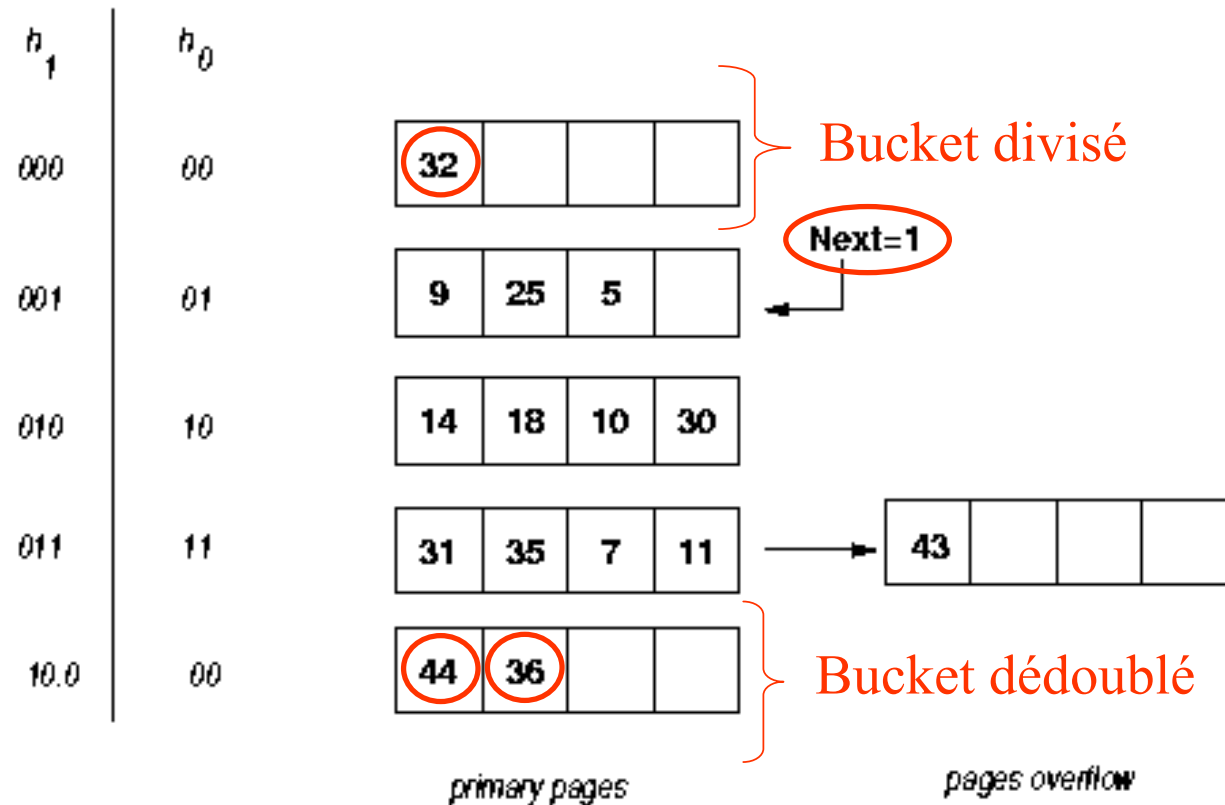
level = 0, N=4





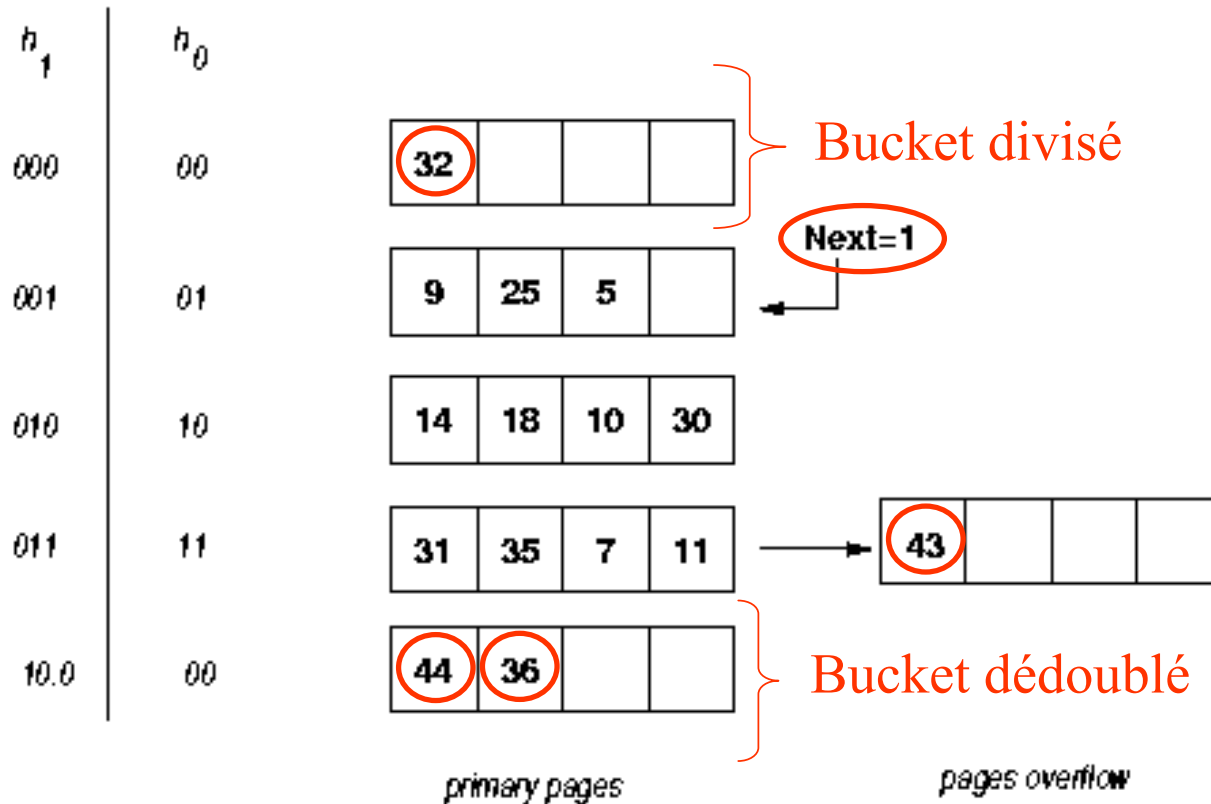
# Hachage linéaire (3/6)

level = 0, N=4



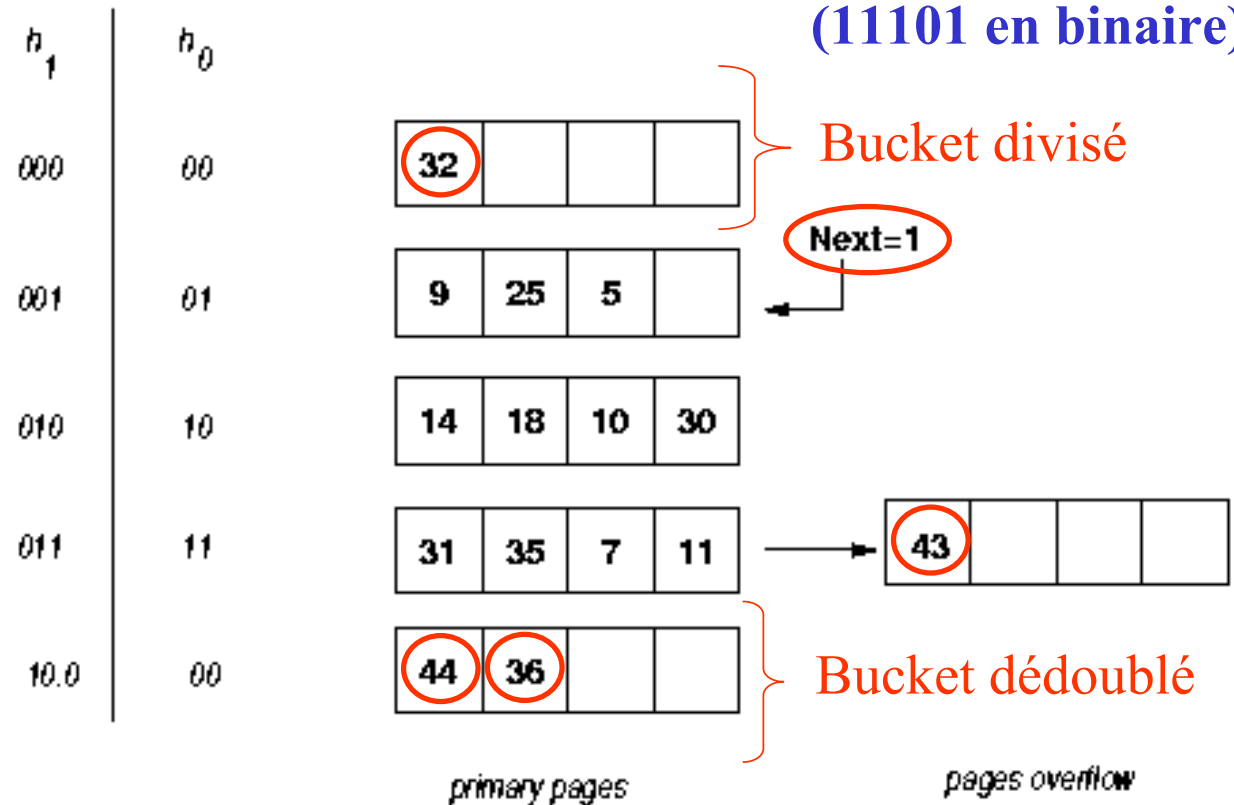
# Hachage linéaire (3/6)

level = 0, N=4



# Hachage linéaire (3/6)

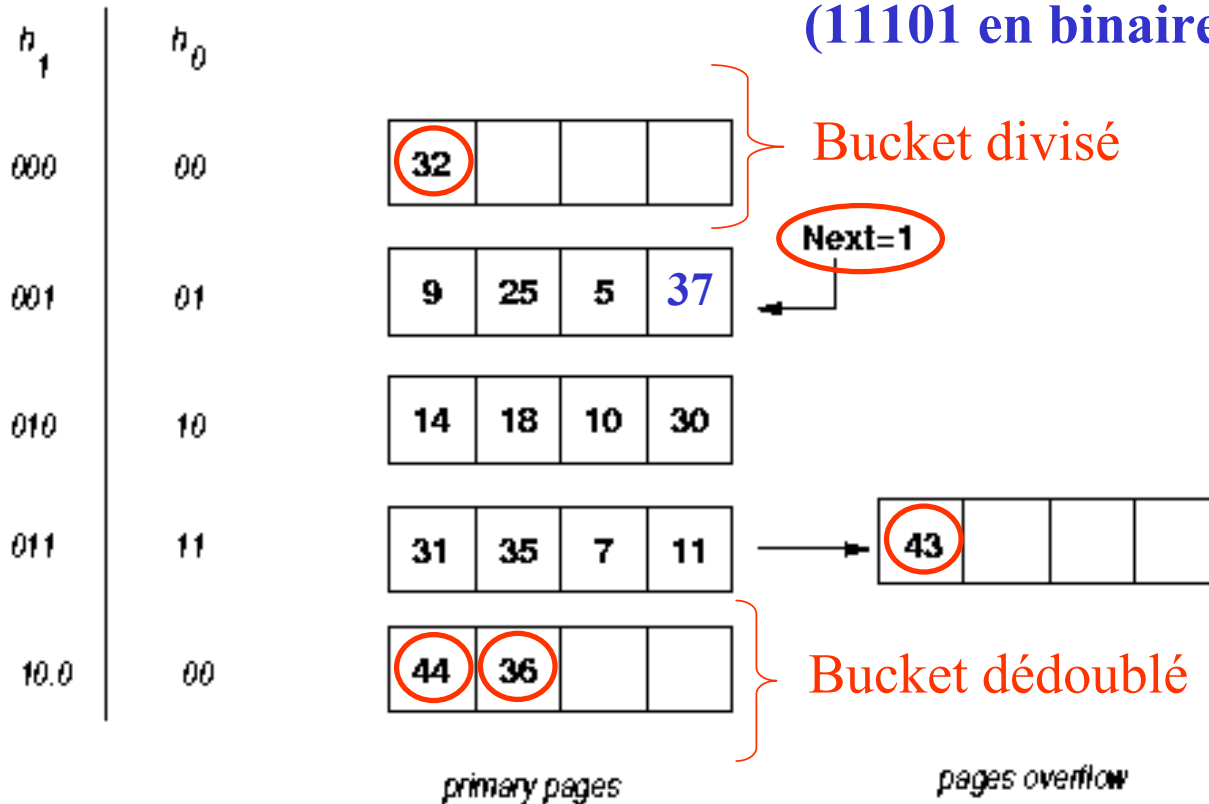
level = 0, N=4



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

# Hachage linéaire (3/6)

level = 0, N=4

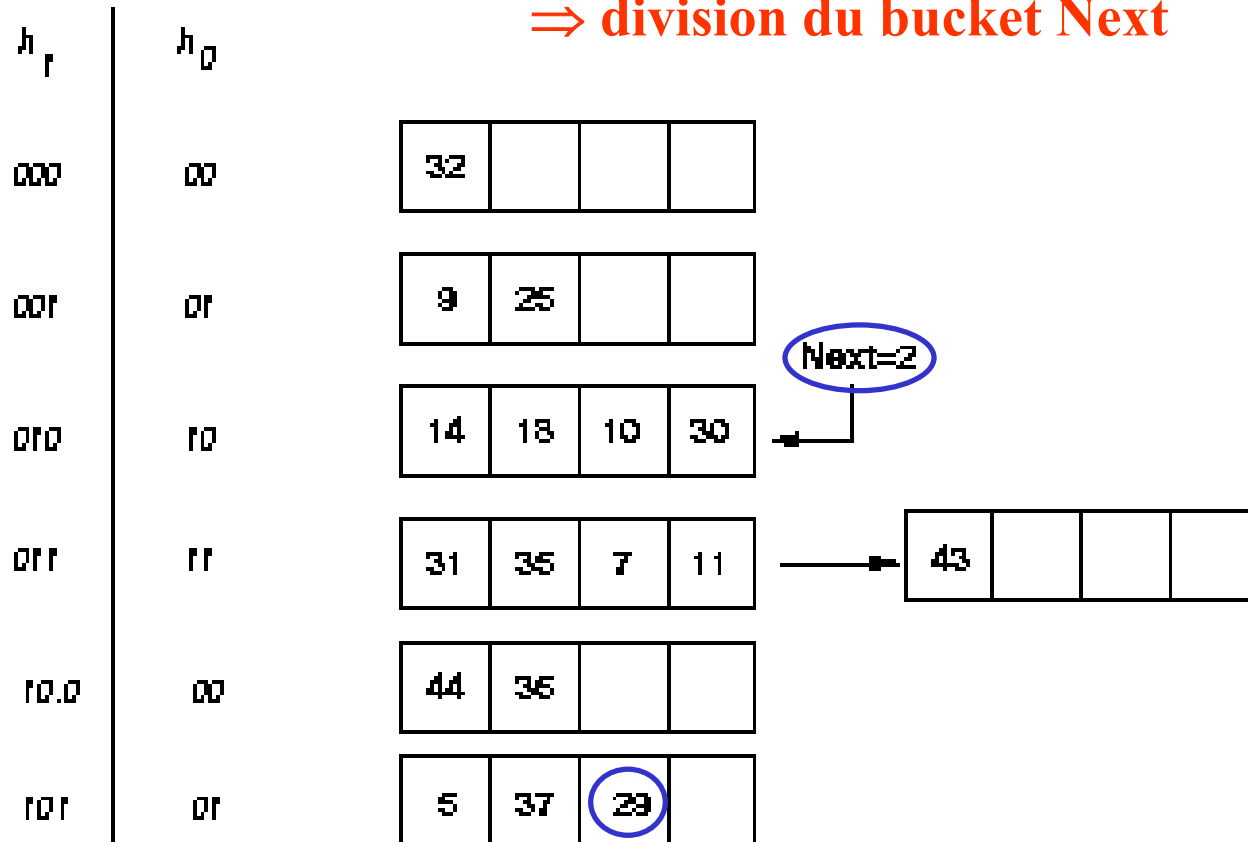


Insertion des entrées 37  
(100101 en binaire) et 29  
(11101 en binaire)

# Hachage linéaire (4/6)

level = 0. N=4

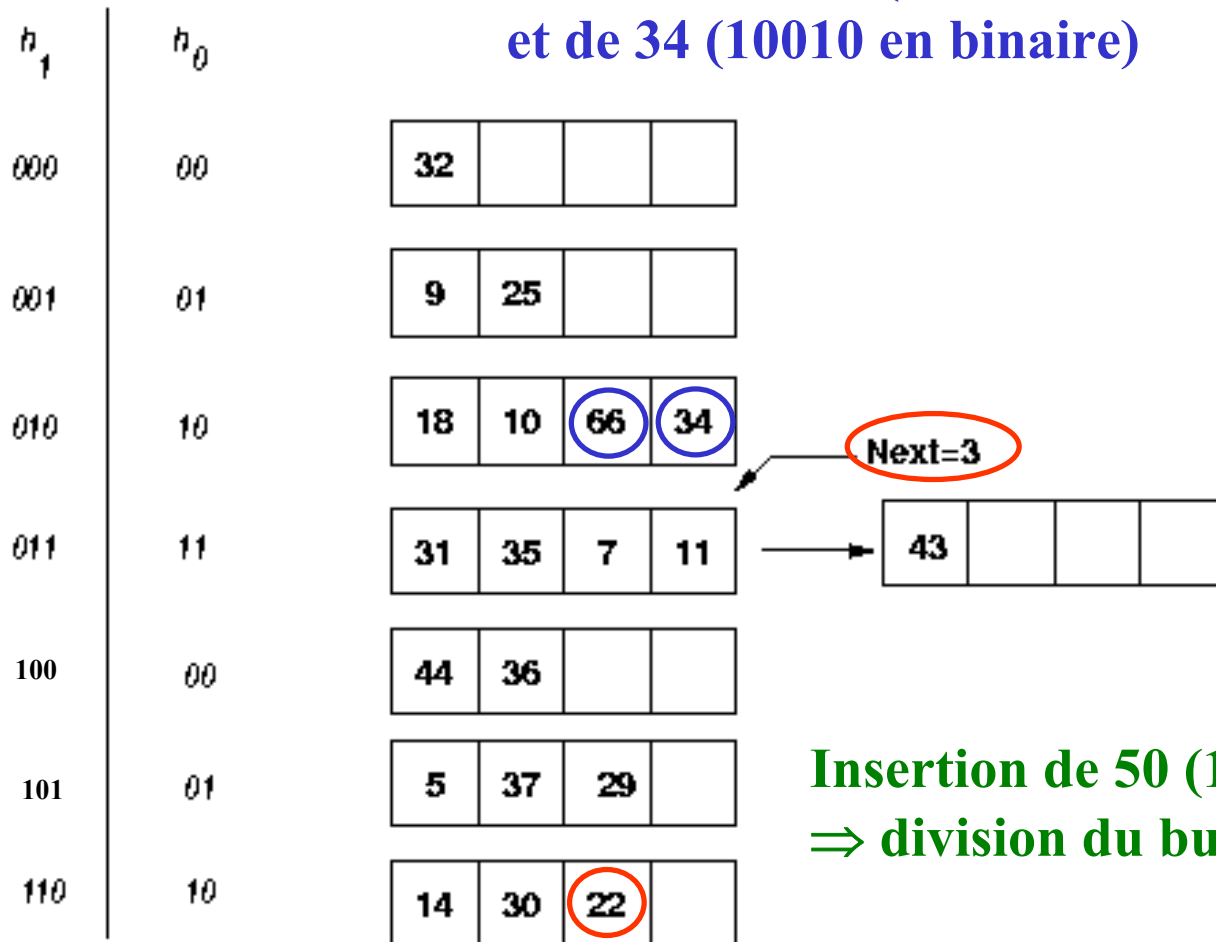
Insertion de 22 (10110 en binaire)  
⇒ division du bucket Next



# Hachage linéaire (5/6)

level = 0, N=4

Insertion de 66 (1000010 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

32			
----	--	--	--

001

9	25		
---	----	--	--

010

18	10	66	34
----	----	----	----



50			
----	--	--	--

011

43	35	11	
----	----	----	--

100

44	36		
----	----	--	--

101

5	37	29	
---	----	----	--

110

14	30	22	
----	----	----	--

111

31	7		
----	---	--	--

Next=0



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**

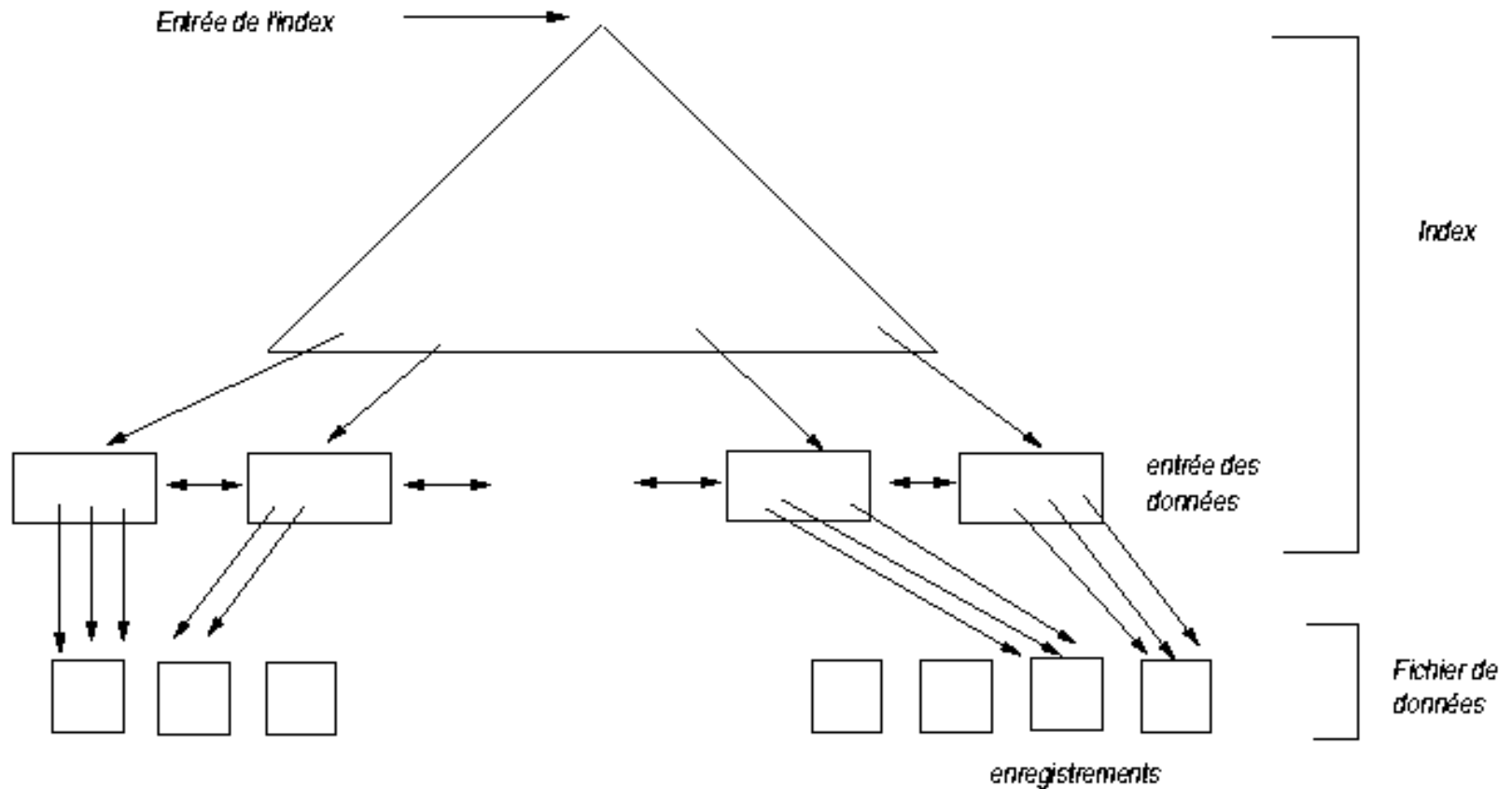
- **Index secondaire**

- *(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**



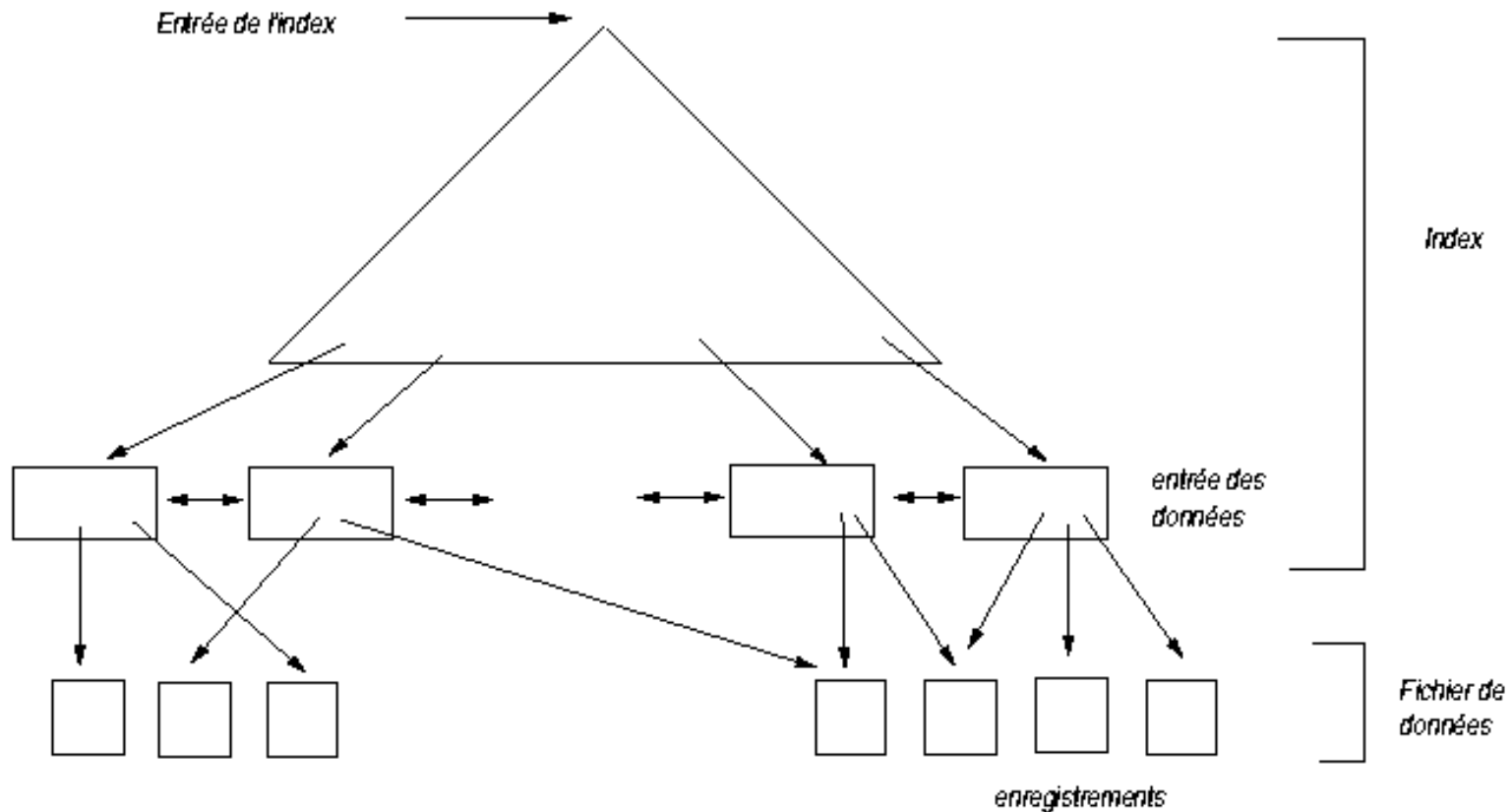
# Index (2/4)

- **Clustered index**



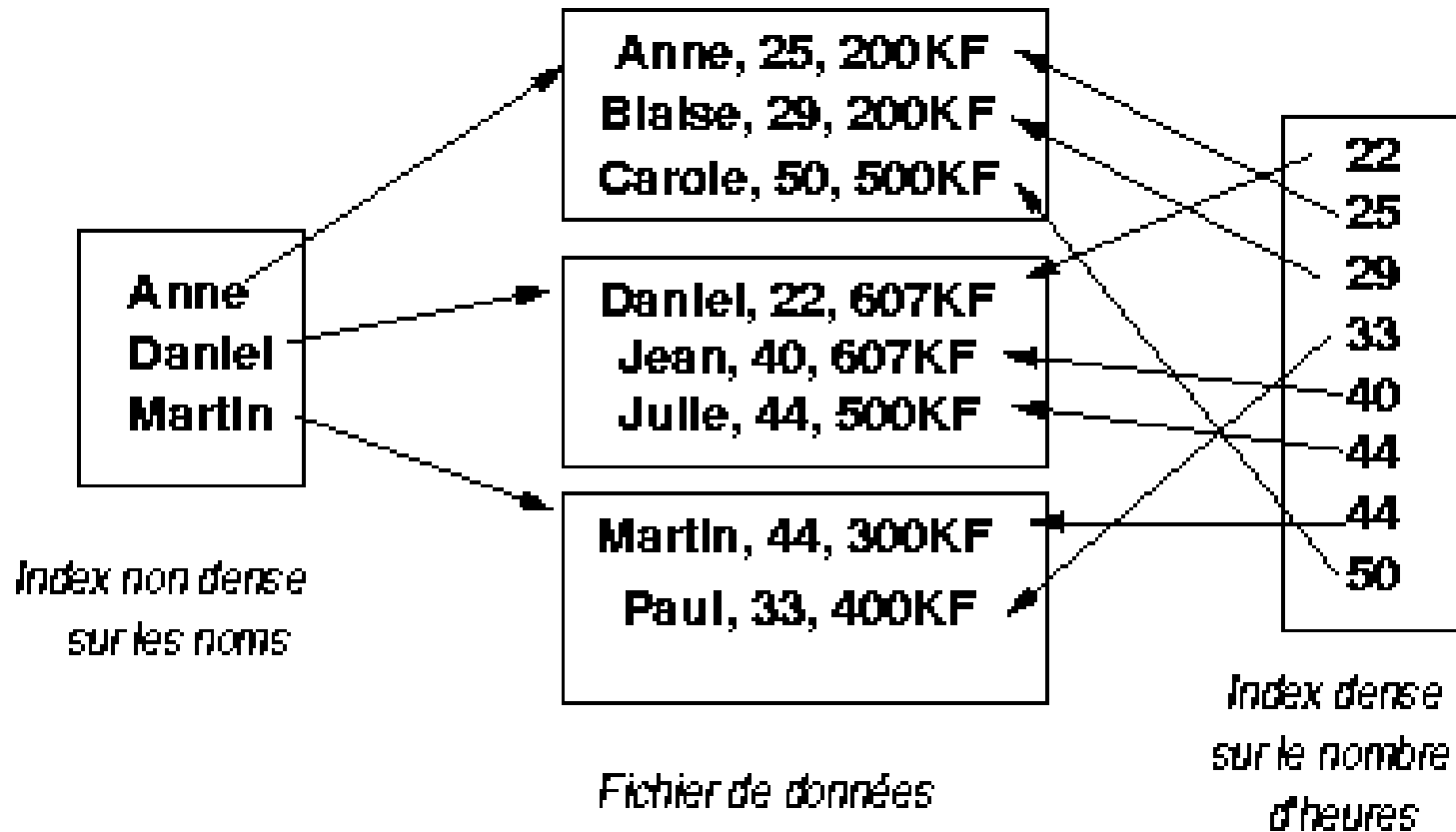
# Index (3/4)

- Unclustered index



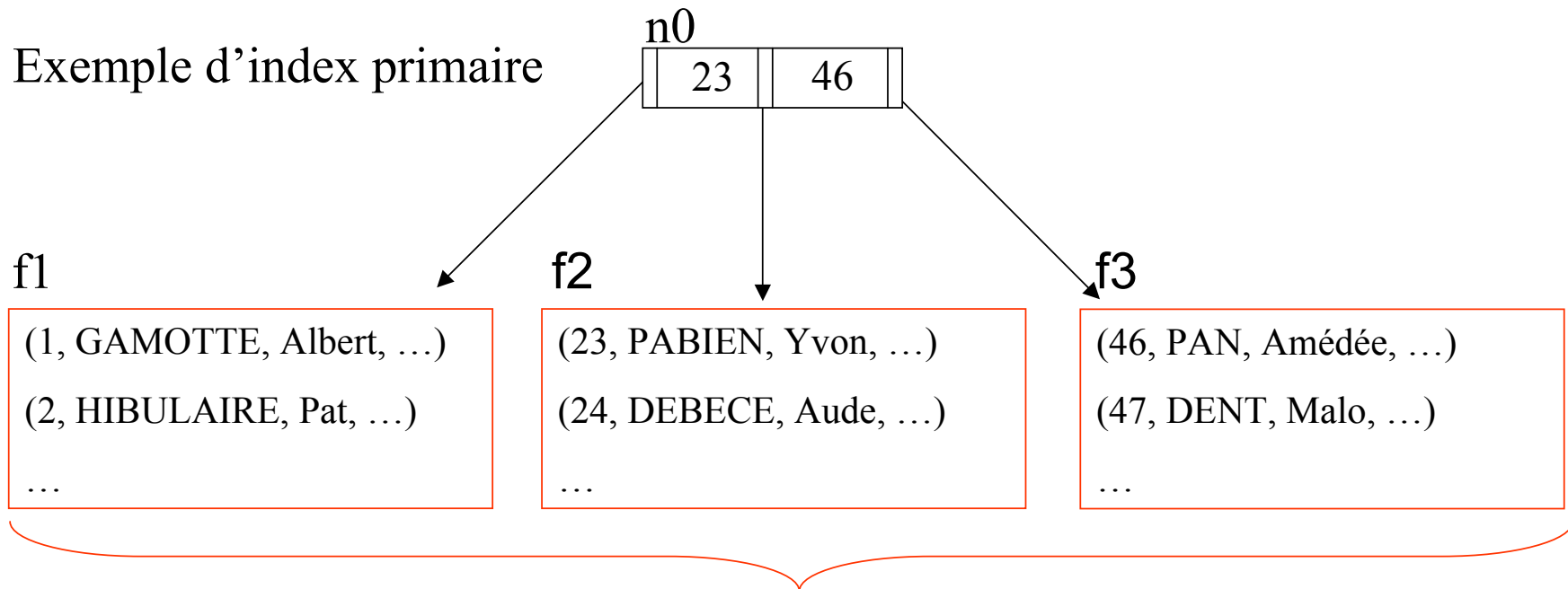
# Index (4/4)

- Index dense / non dense (*sparse*)





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

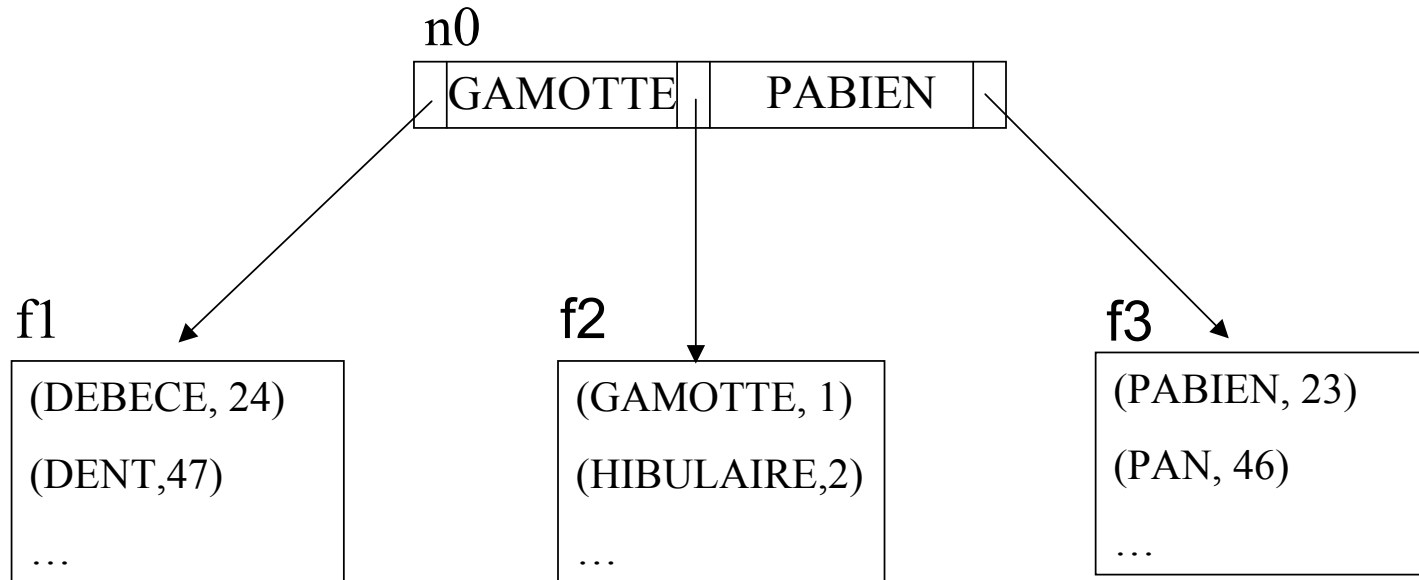


Pages de la relation et feuilles de l'index

# Arbre B+ (2/3)

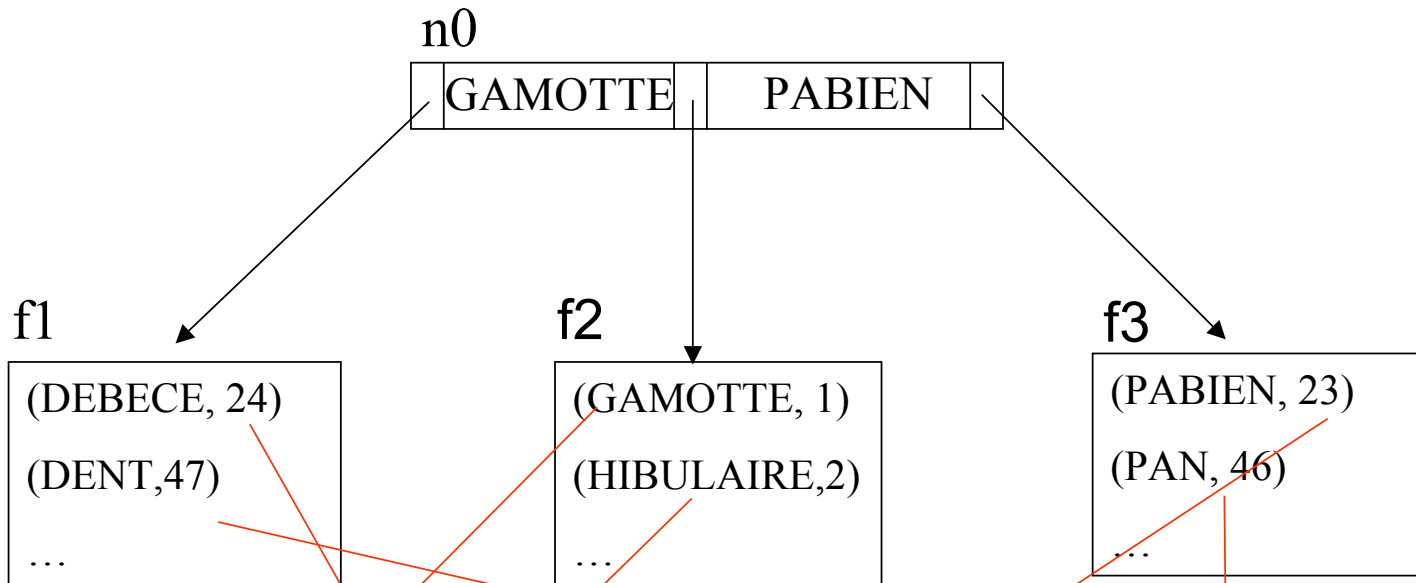


Exemple d'index secondaire



# Arbre B+ (2/3)

## Exemple d'index secondaire



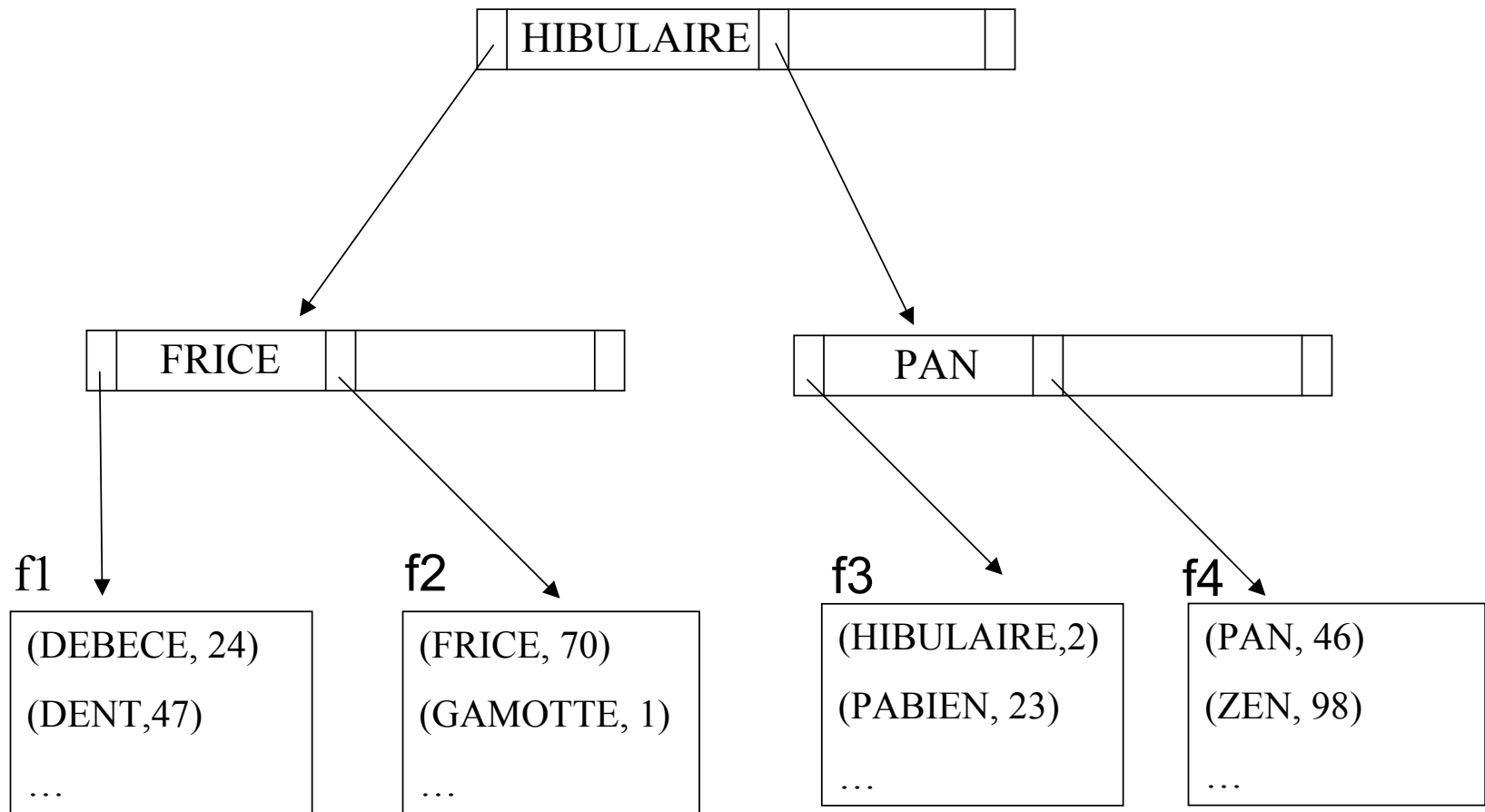
## Pages de la relation

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

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

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

# Arbre B+ (3/3)

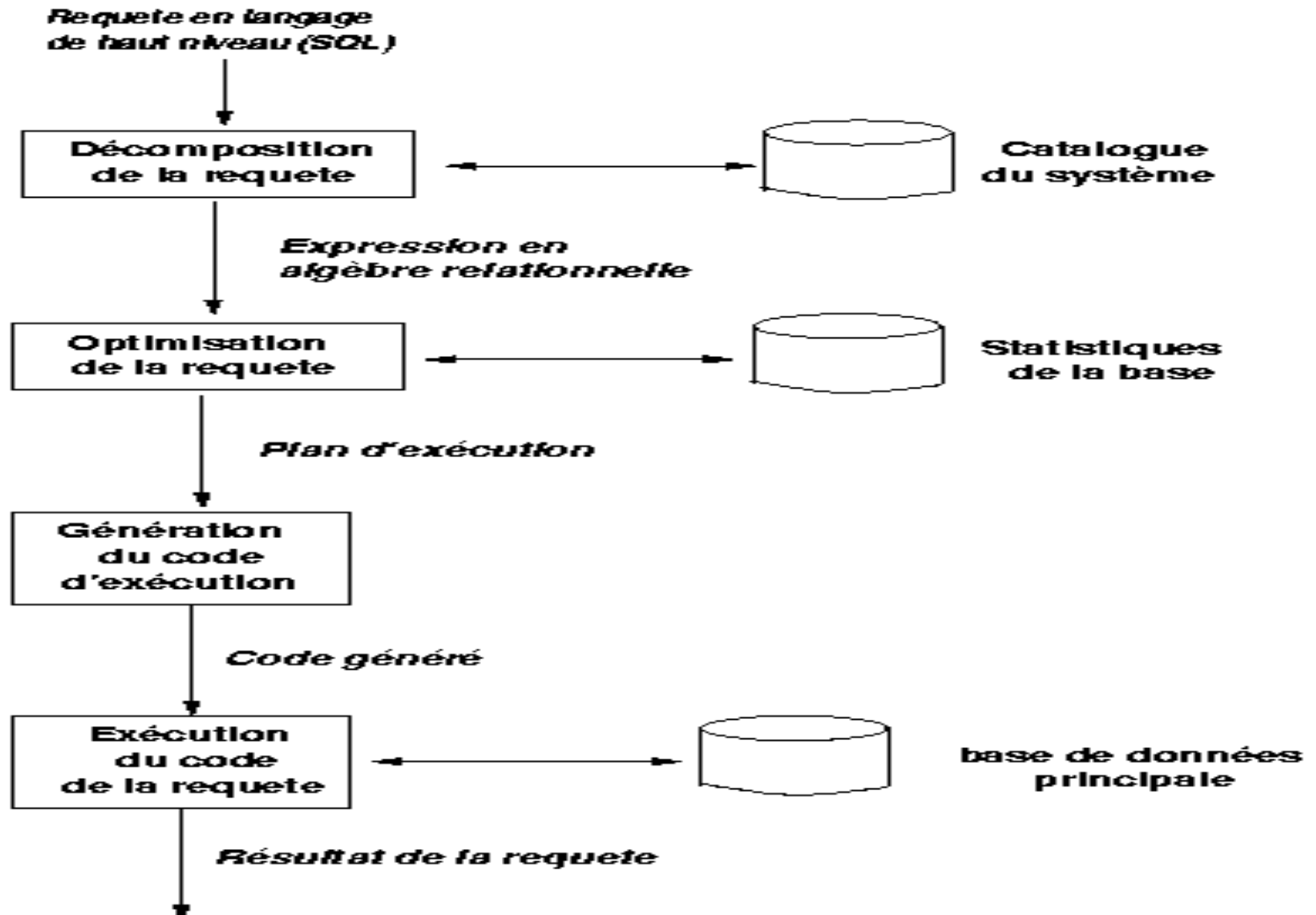


# Chap. III - Optimisation de requêtes

- **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**

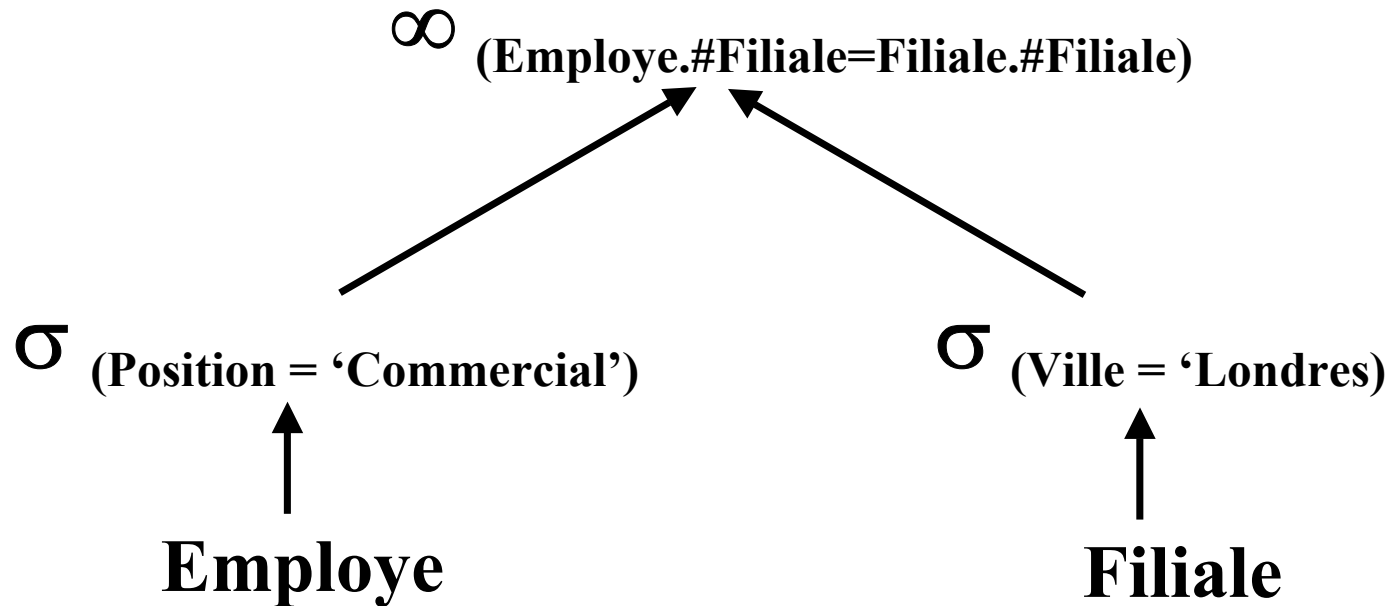
# Catalogue du système (2/2)

- 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 :  $\sigma_{p \wedge q \wedge r} (R) =$

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

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

4) Commutativité des sélections et des projections :

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

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

6) Commutativité des jointures et des sélections

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



## Equivalence d'expressions (2/3)

### 7) Commutativité des jointures et des projections

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

### 8) Commutativité des unions et des intersections

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

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

$$\sigma_p(R \cup S) =$$

$$\sigma_p(R \cap S) =$$

$$\sigma_p(R - S) =$$

## Equivalence d'expressions (3/3)

### 10) Commutativité des projections et des unions

$$\Pi_L(R \cup S) =$$

### 11) Associativité des jointures

$$(R \bowtie S) \bowtie T =$$

### 12) Associativité des unions et des intersections

$$(R \cup S) \cup T =$$

$$(R \cap S) \cap T =$$

# Transformation d'un arbre algébrique

- ① Division des conjonctions de sélections
- ② Ré-ordonnancement 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-ordonnancement des équi-jointures en utilisant la règle 11
- ⑥ Déplacement des projections et création de nouvelles projections en utilisant les règles 4 et

$\Pi_{\text{Nom}}$   
▲

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

$\Pi_{\text{Nom}}$   
▲

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

$\Pi_{\text{Nom}}$   
▲

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

Projet

Employe

Equipe

$\Pi_{\text{Nom}}$   
▲

SELECT Nom

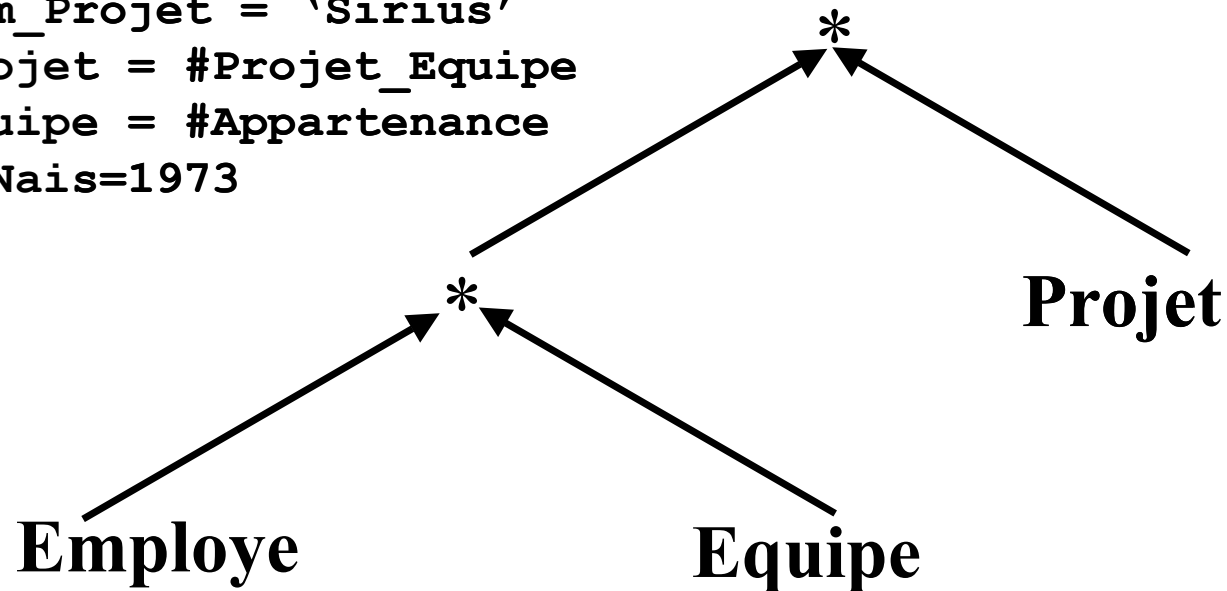
**FROM** Employe, Equipe, Projet

WHERE Nom\_Projet = 'Sirius'

AND #Projet = #Projet\_Equipe

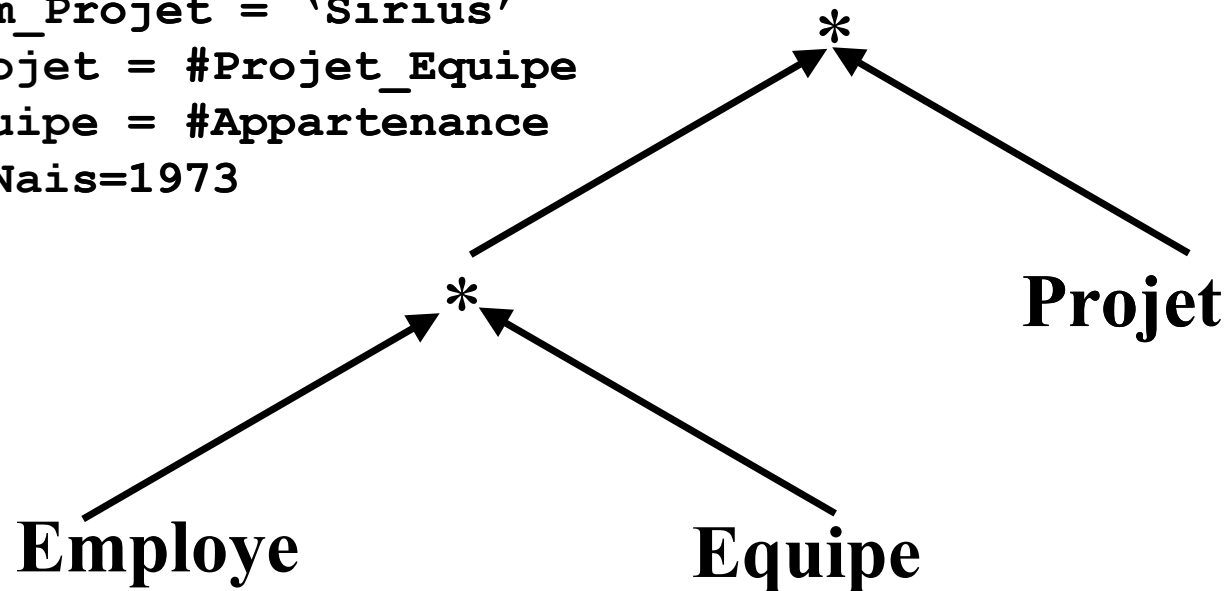
AND #Equipe = #Appartenance

AND DaeNais=1973

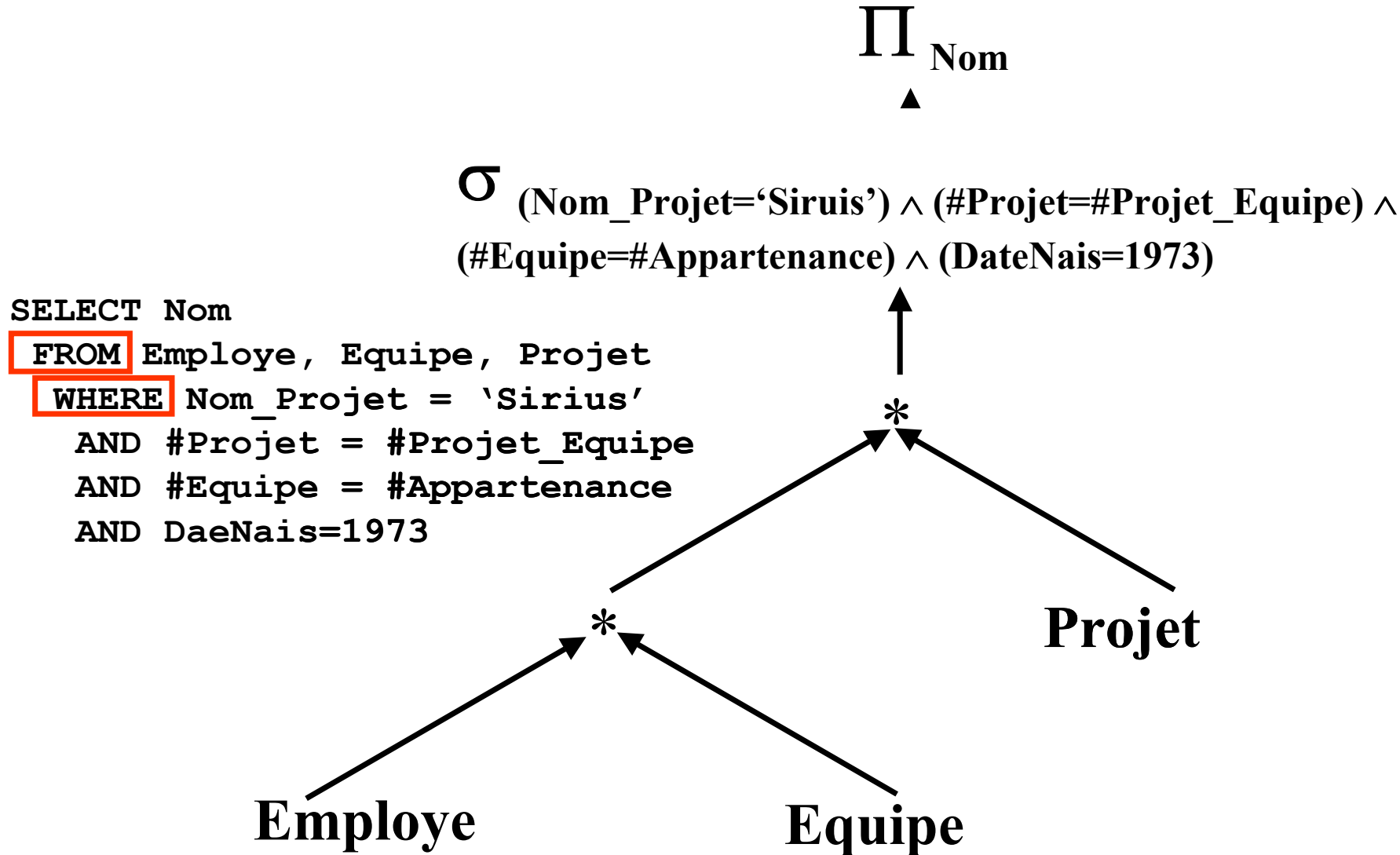


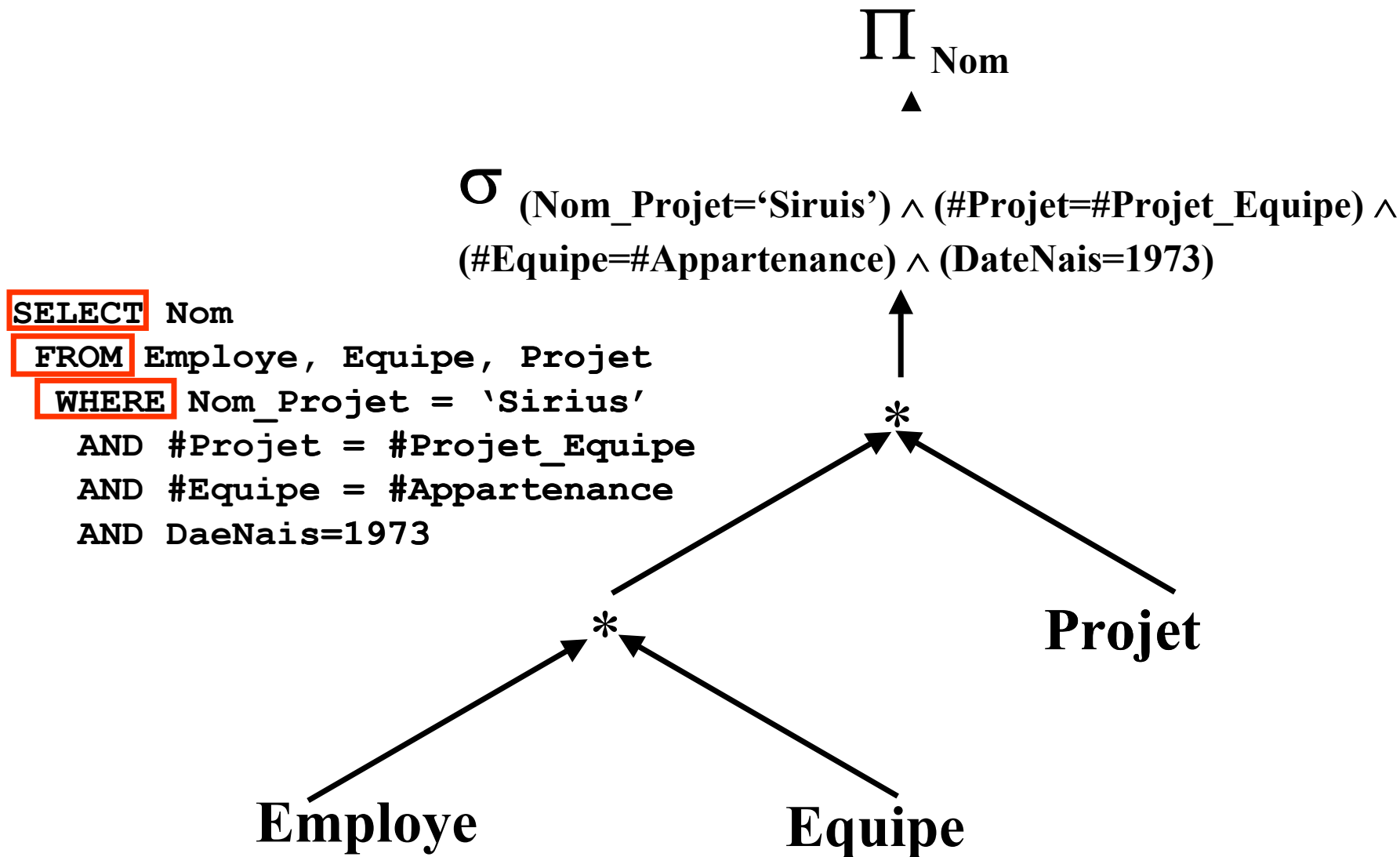
$\Pi_{\text{Nom}}$   
▲

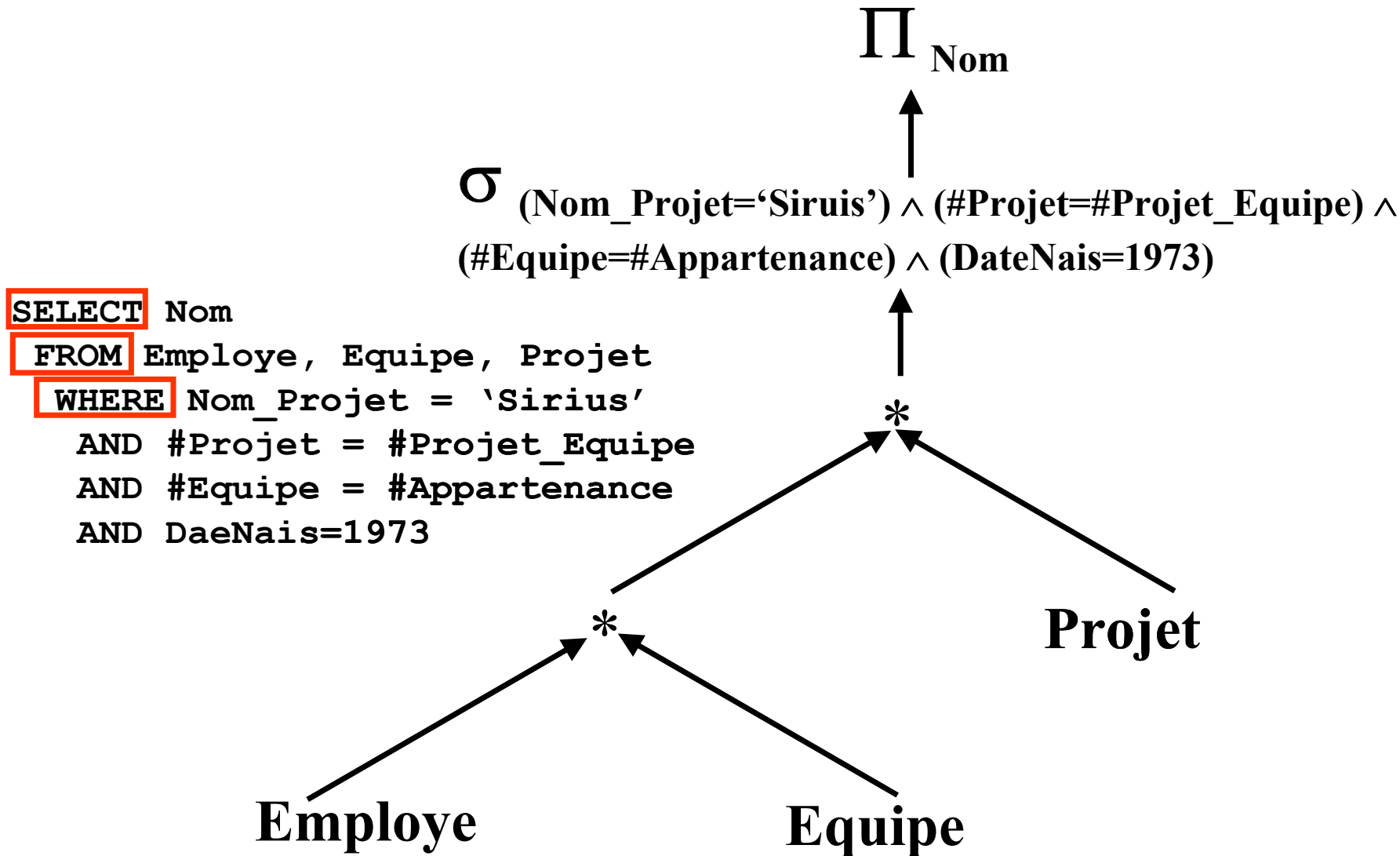
```
SELECT Nom
FROM Employe, Equipe, Projet
WHERE Nom_Projet = 'Sirius'
      AND #Projet = #Projet_Equipe
      AND #Equipe = #Appartenance
      AND DaeNais=1973
```



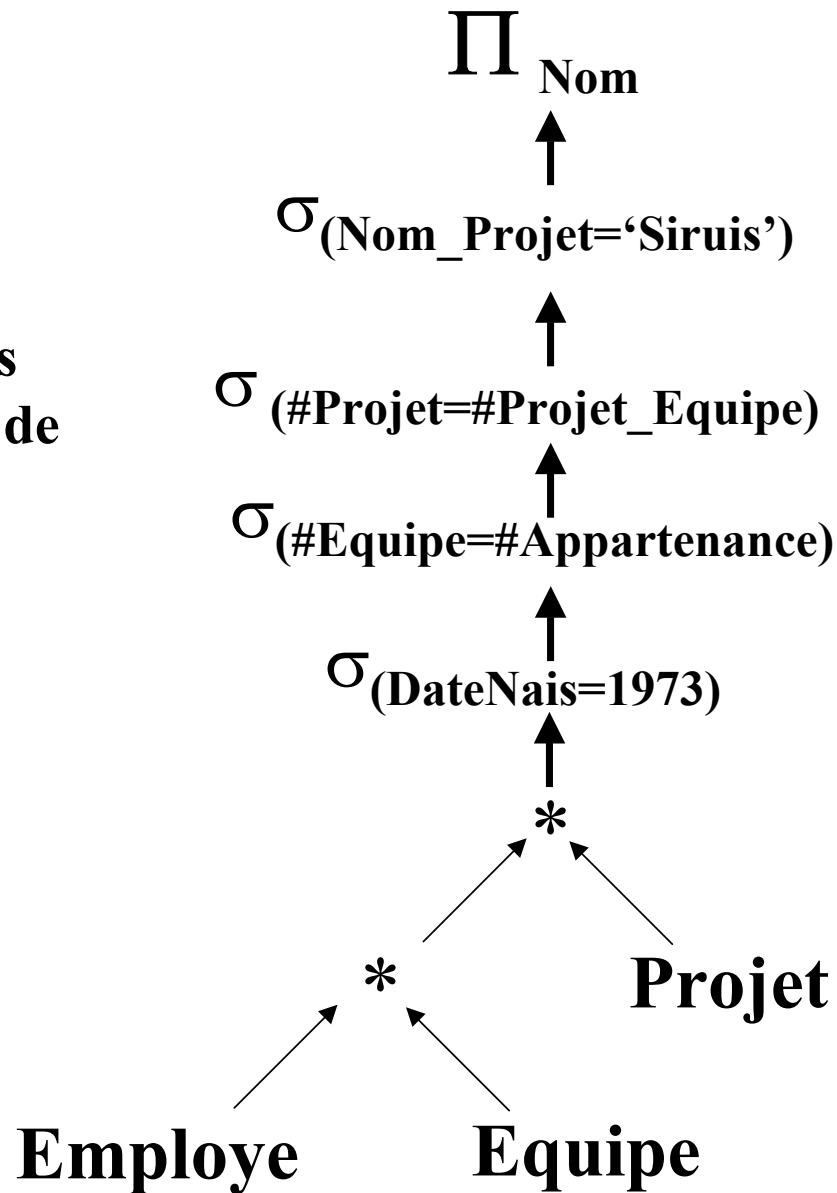




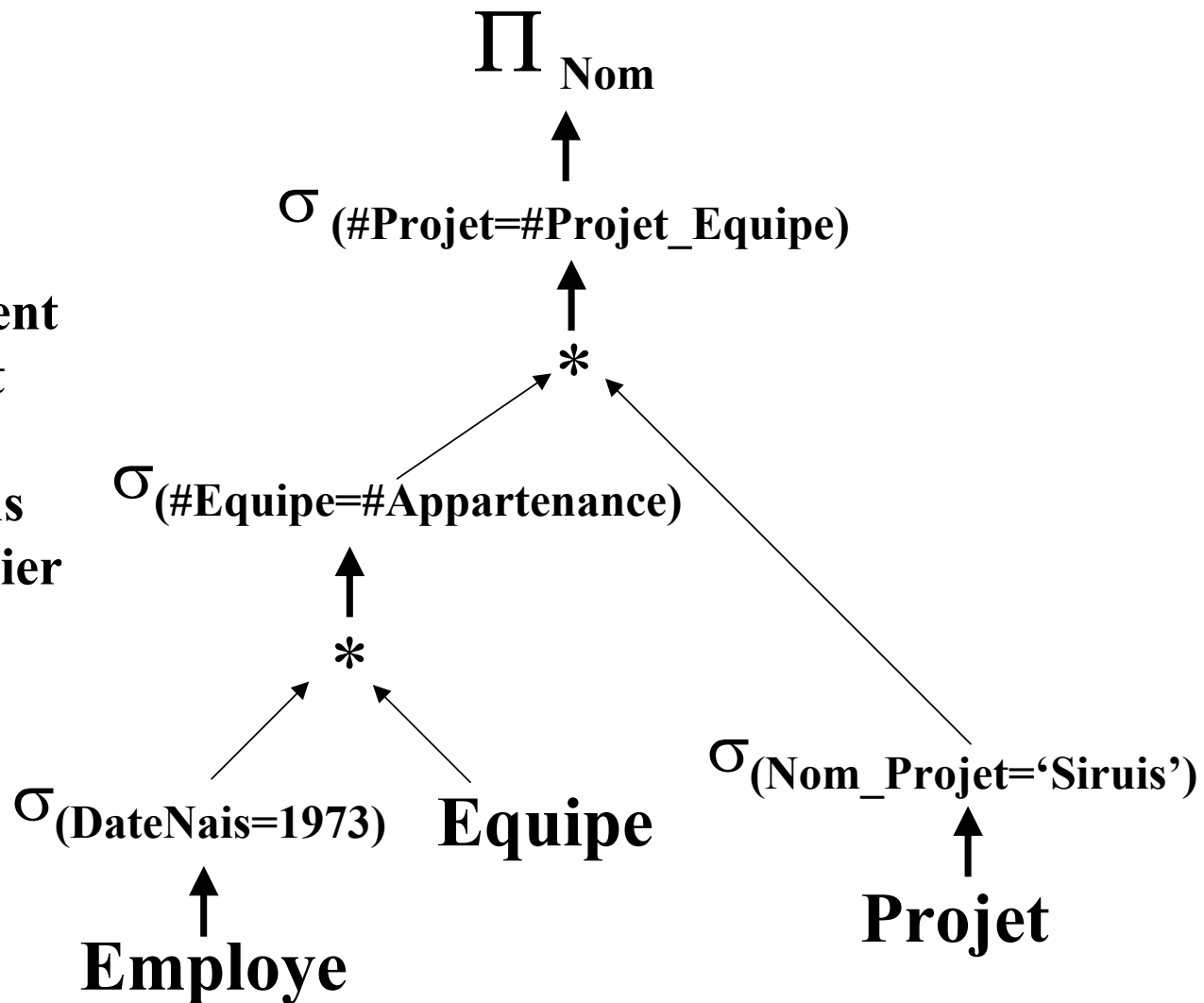




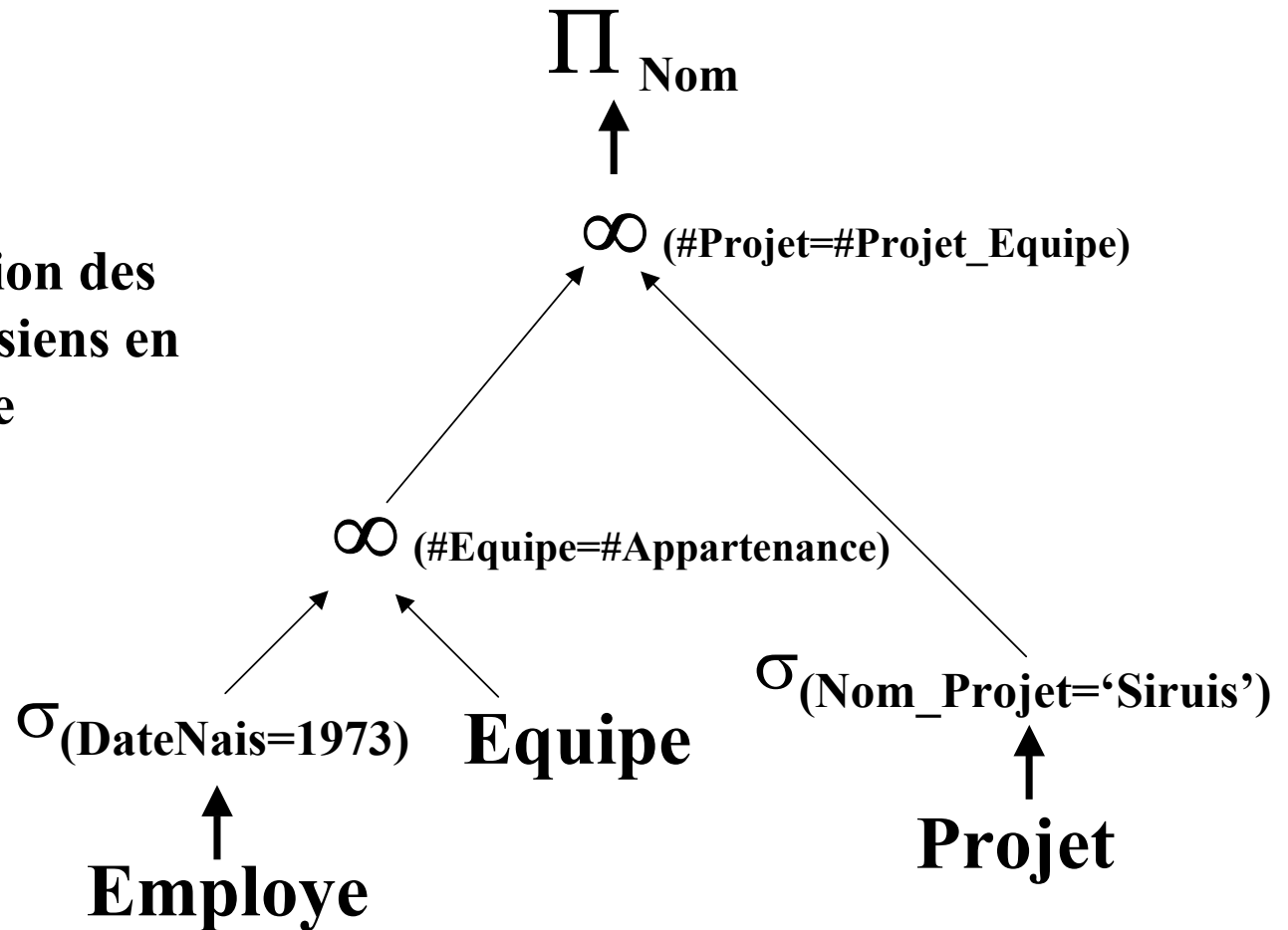
**Division des  
conjonctions de  
sélections**



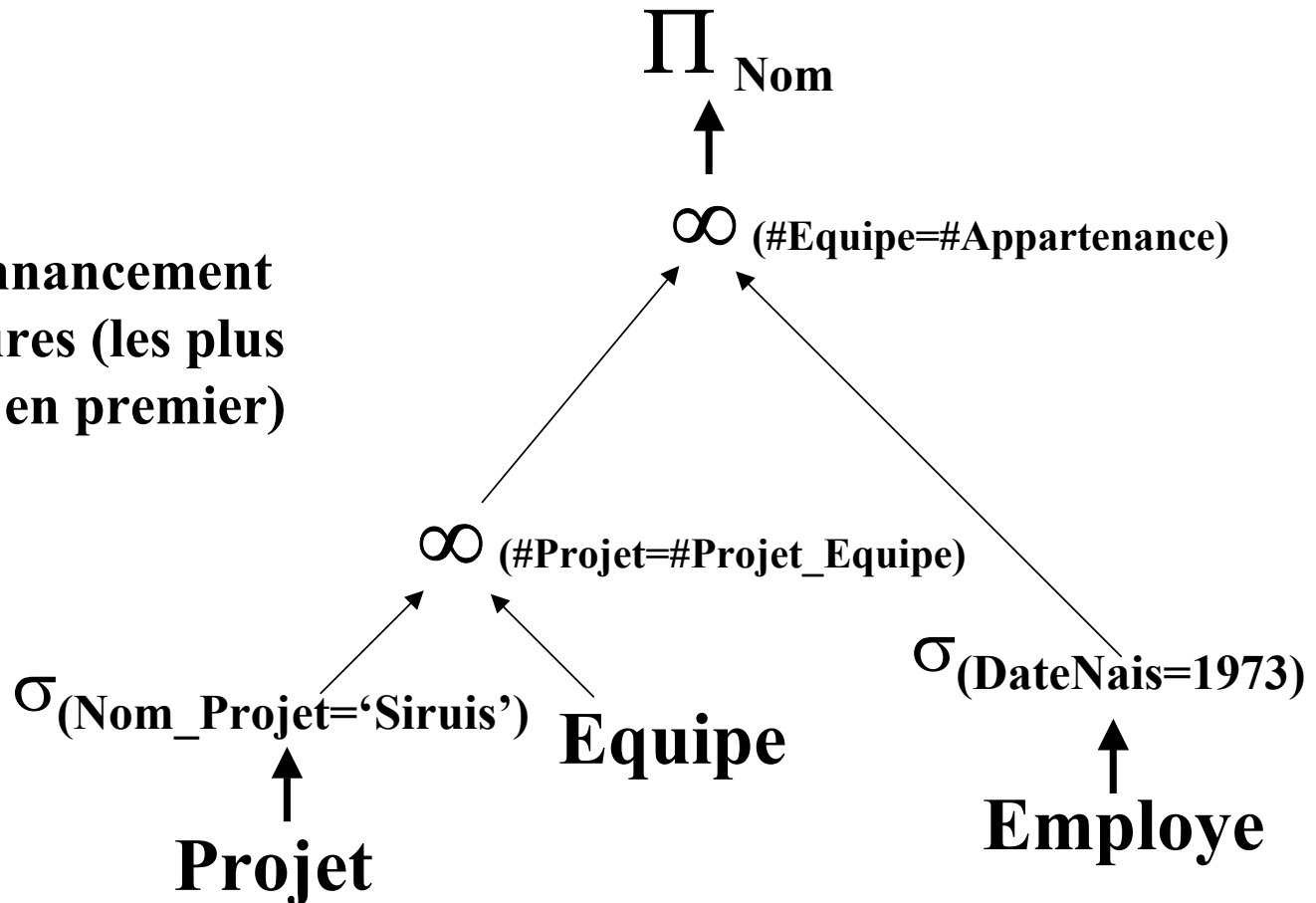
**Ré-ordonnancement  
des sélections et  
application des  
sélections les plus  
sélectives en premier**



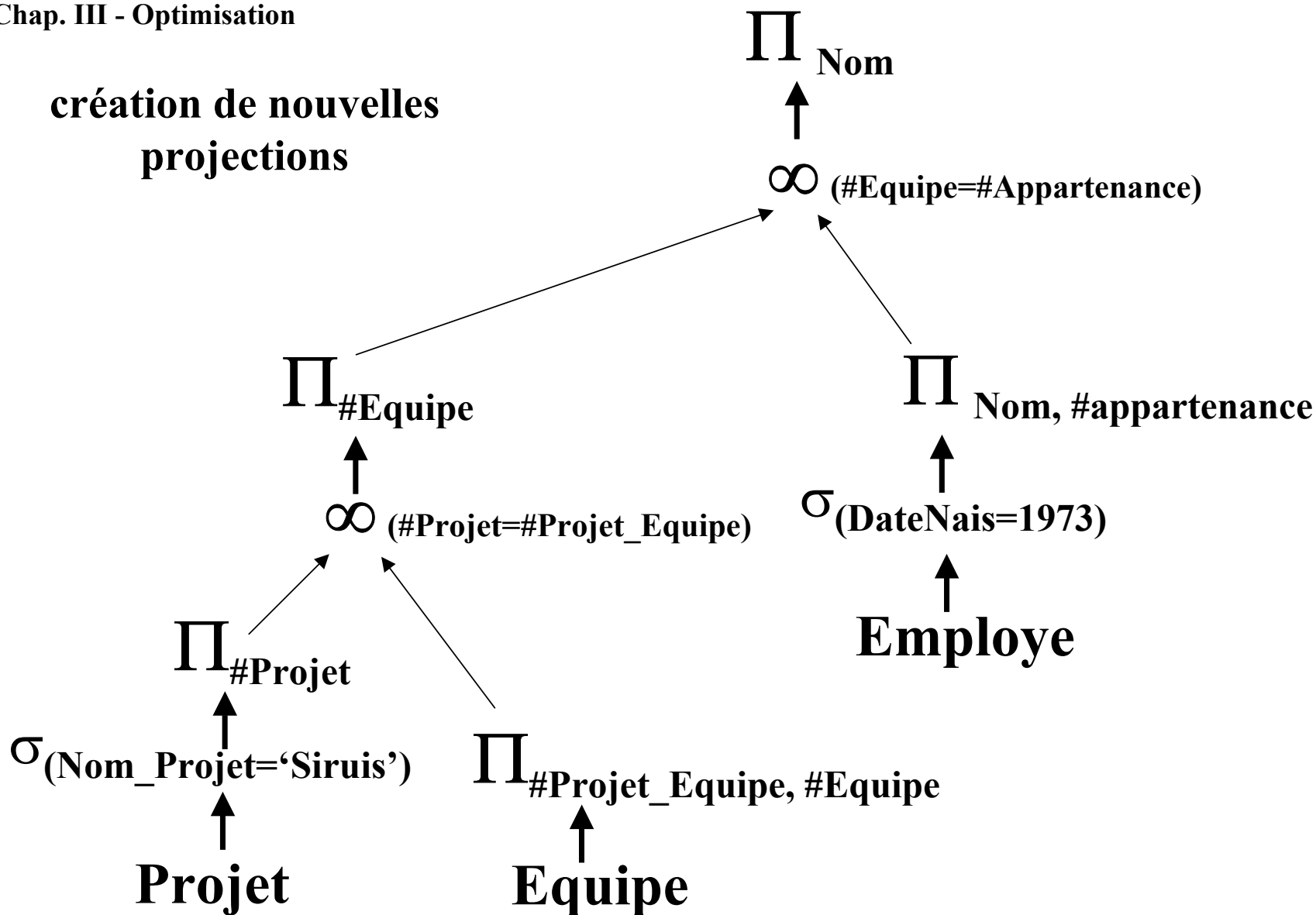
**Transformation des  
produits cartésiens en  
jointure**



**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 $\sigma_p(R)$

## 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<sub>D</sub>(e)=500**

**NDistinct<sub>P</sub>(e)=10**

**NDistinct<sub>S</sub>(e)=500**

**Min<sub>S</sub>(e)=1000**

**Max<sub>S</sub>(e)=5000**

**NLevels<sub>D</sub>(I)=2**

**NLevels<sub>S</sub>(I)=2**

**NBlocks<sub>S</sub>(I)=50**

*Index unclustered*

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

$$\sigma_{Position} = 'Commercial'. (e)$$

$$\sigma_{Departement} = 'RH'. (e)$$

$$\sigma_{Salaire} > 2000. (e)$$

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

# 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**

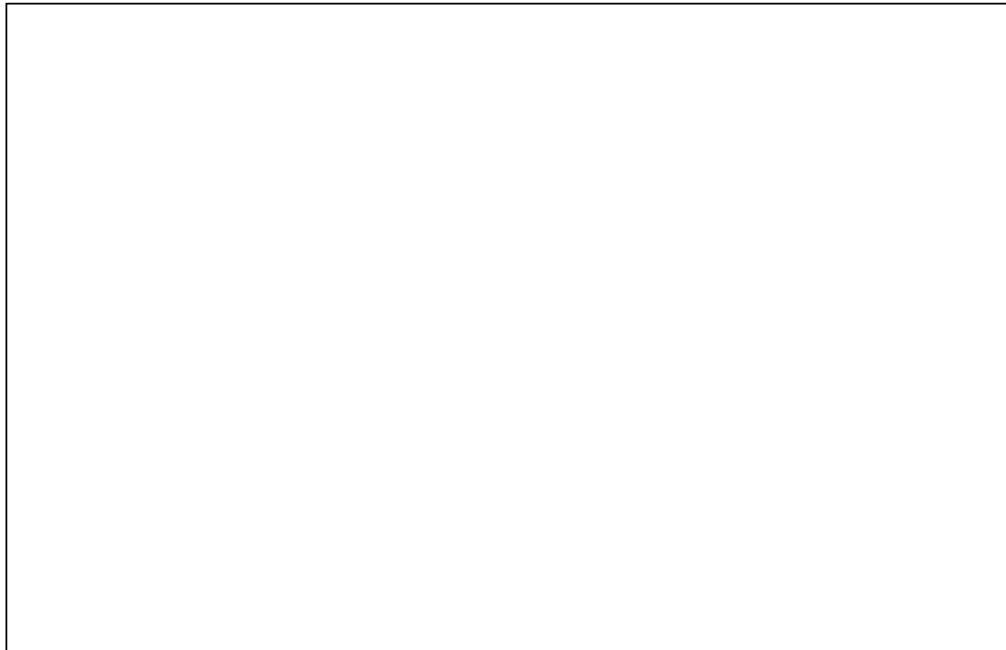
# Cardinalité de la $R \bowtie_F S$

- $NTuples(R \bowtie_F S) \leq NTuples(R) * NTuples(S)$
- 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 :

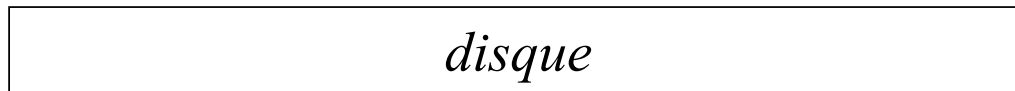
$$NTuples(R \bowtie_F S) \sim \text{Max}(SC_A(R) * NTuples(S), SC_B(S) * NTuples(R))$$

# Boucles imbriquées

*Zone mémoire du buffer*



*disque*



*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*



Page de R

*disque*

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

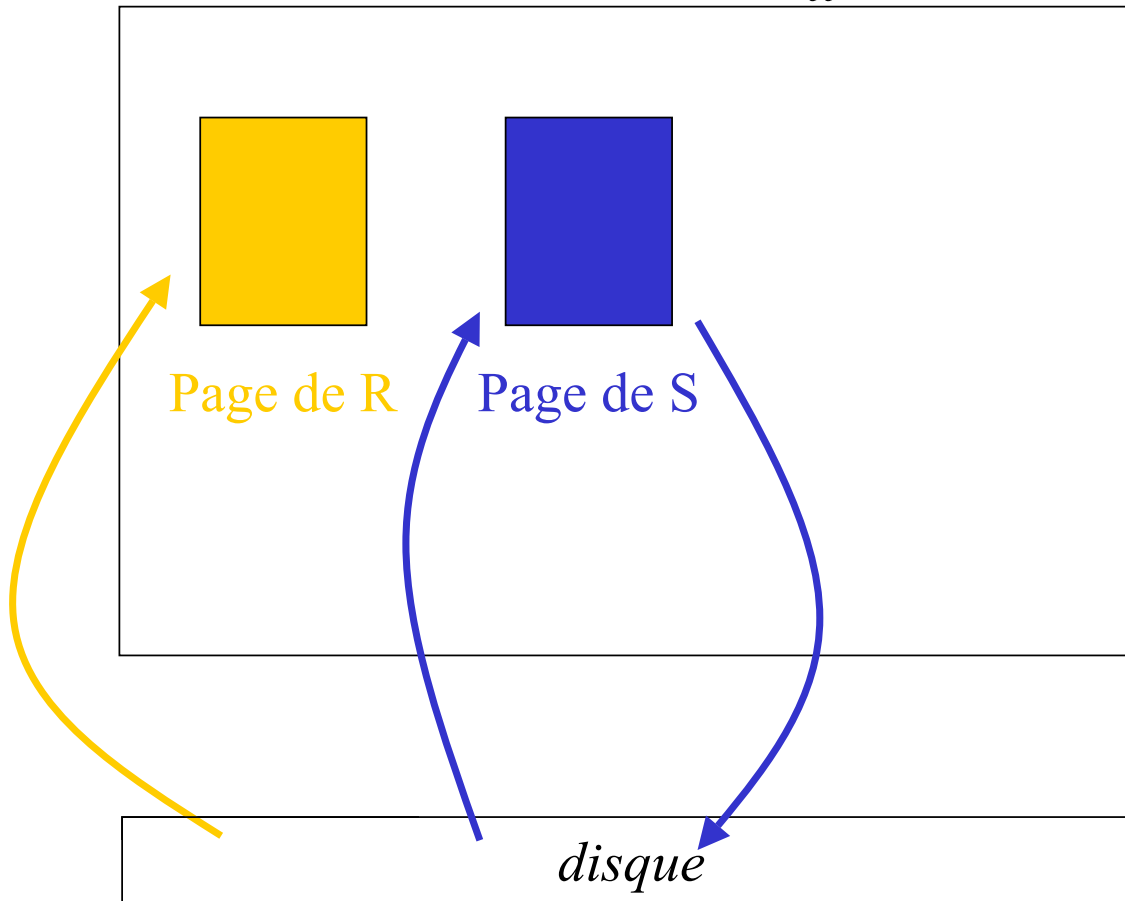
*S'il y a NBuffer places en mémoire, on monte les pages de R en mémoire par paquets de (NBuffer - 2) pages*

*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*



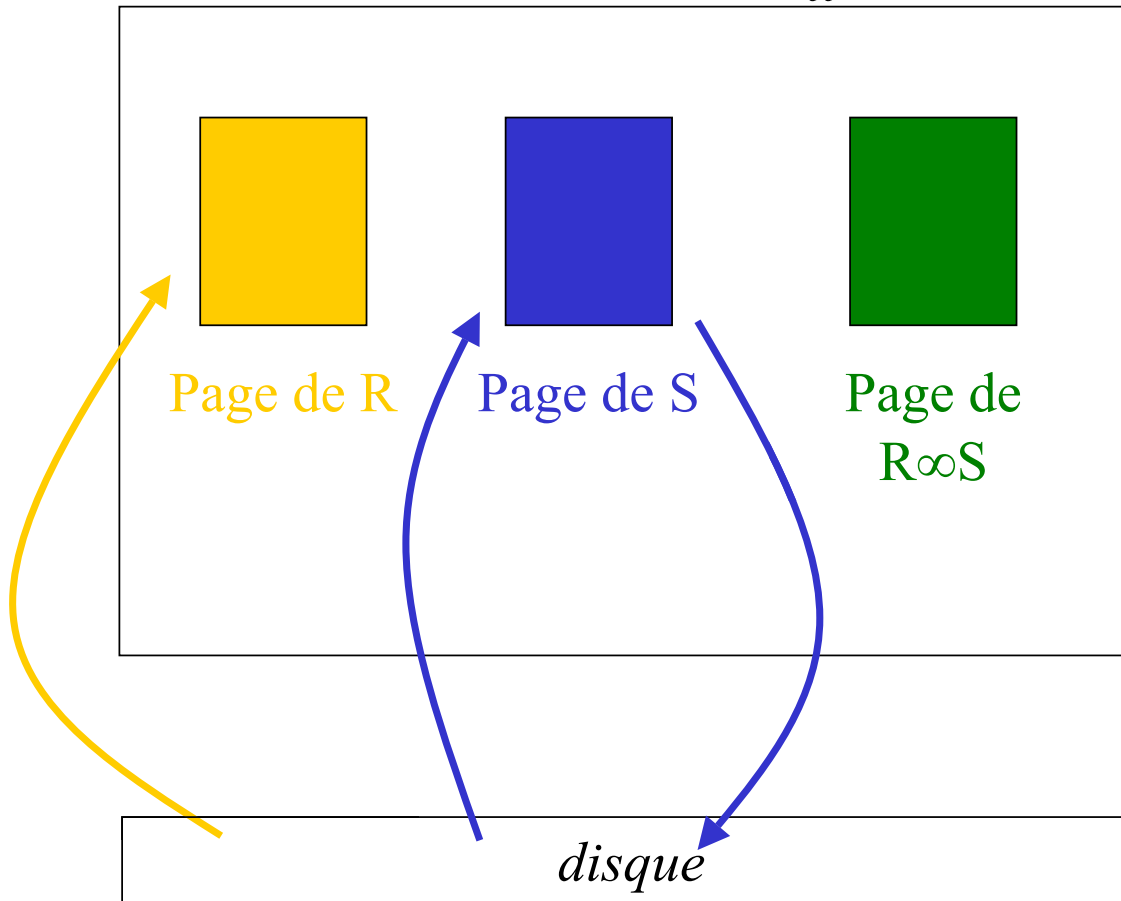
*Les blocs de S sont montés en mémoire autant de fois qu'il y a de blocs pour R*

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

# 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*

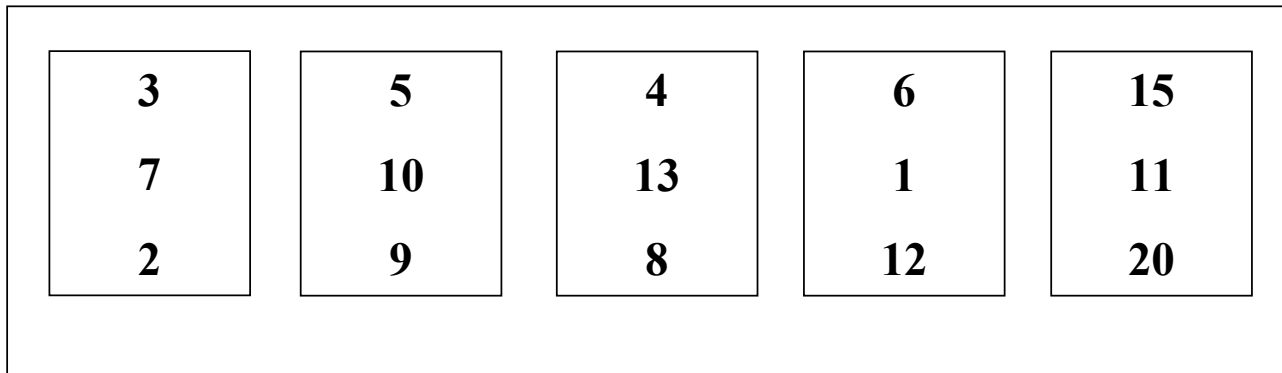
3	5	4	6	15
7	10	13	1	11
2	9	8	12	20

# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

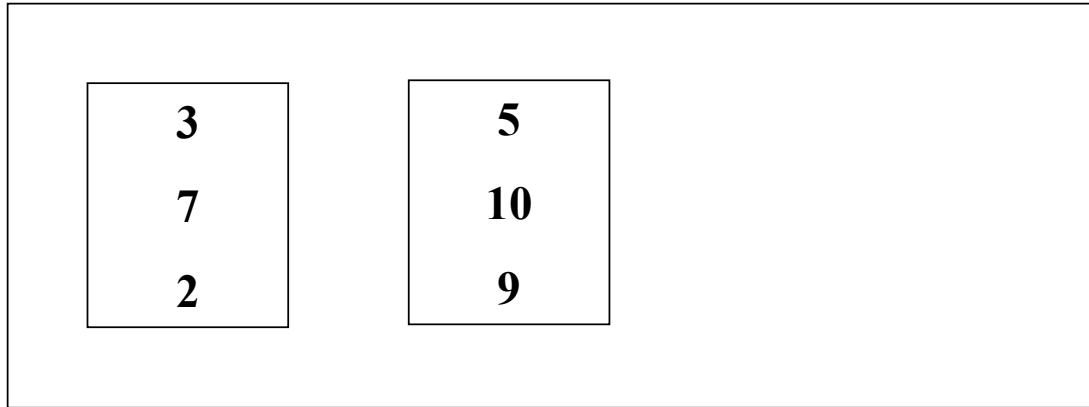


*disque*

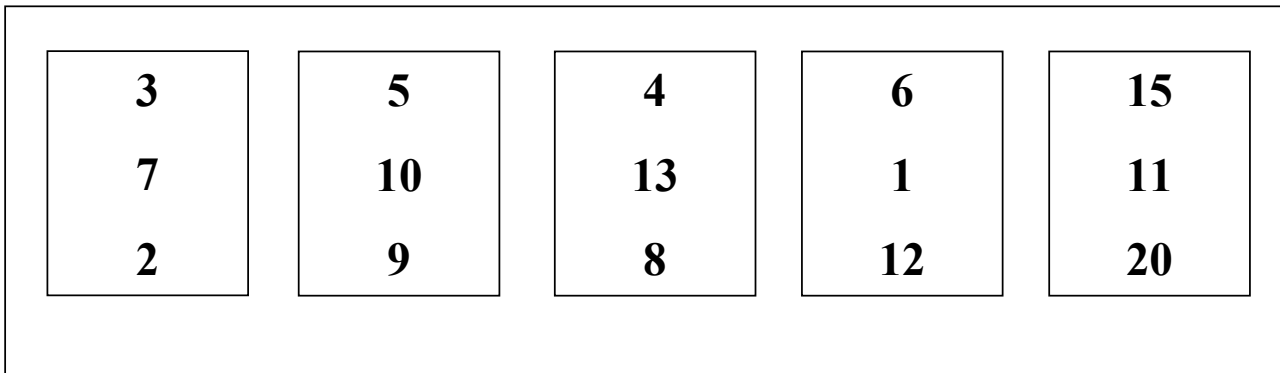


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

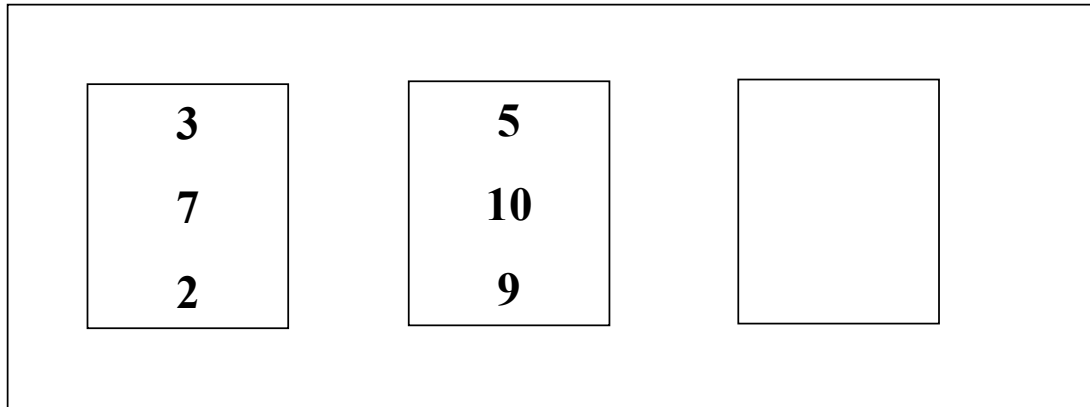


*disque*

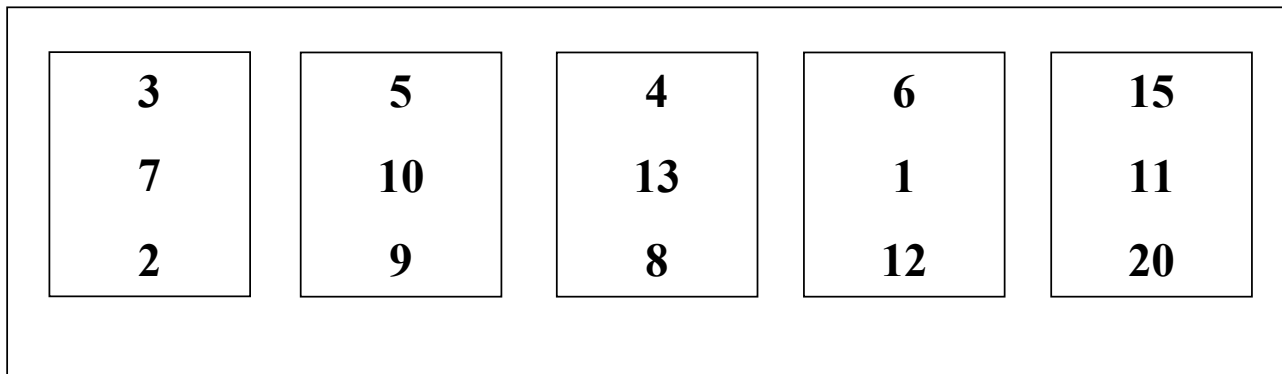


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

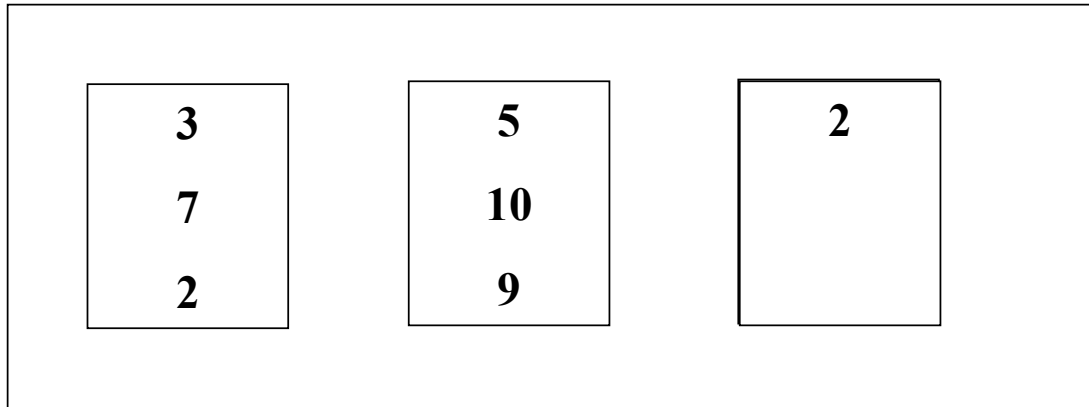


*disque*

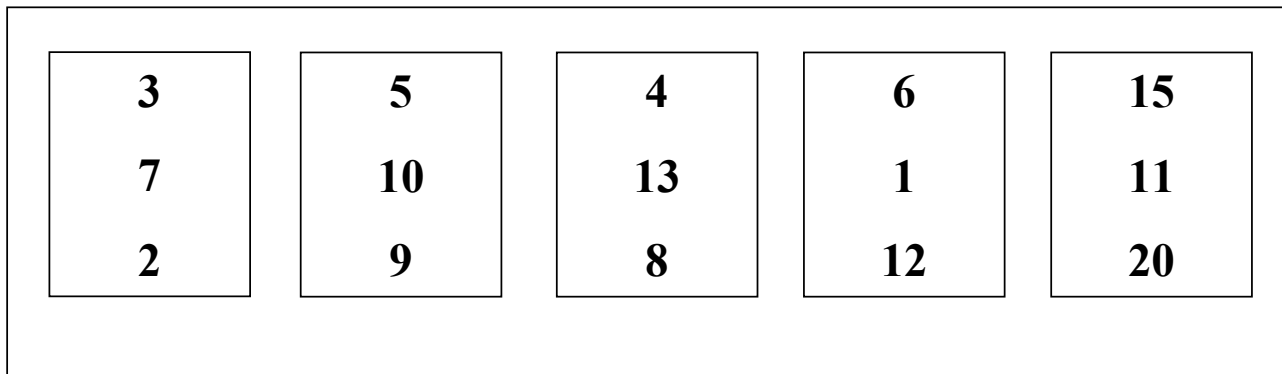


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*



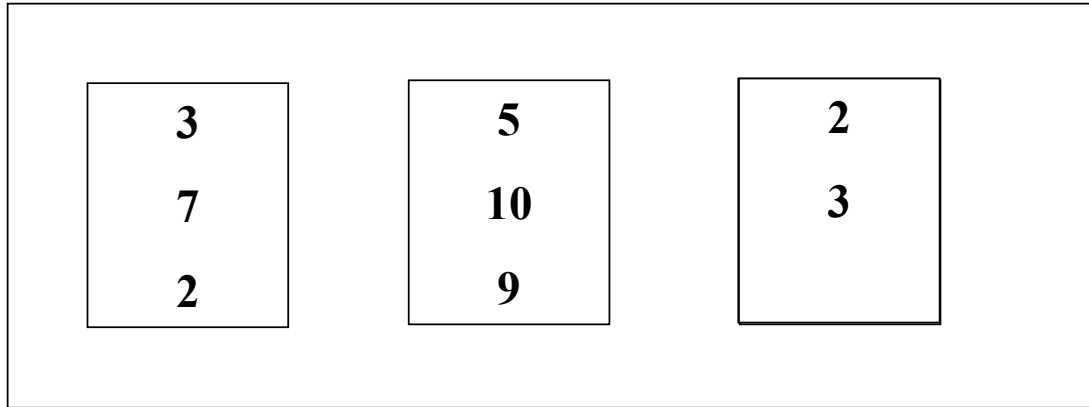
*disque*



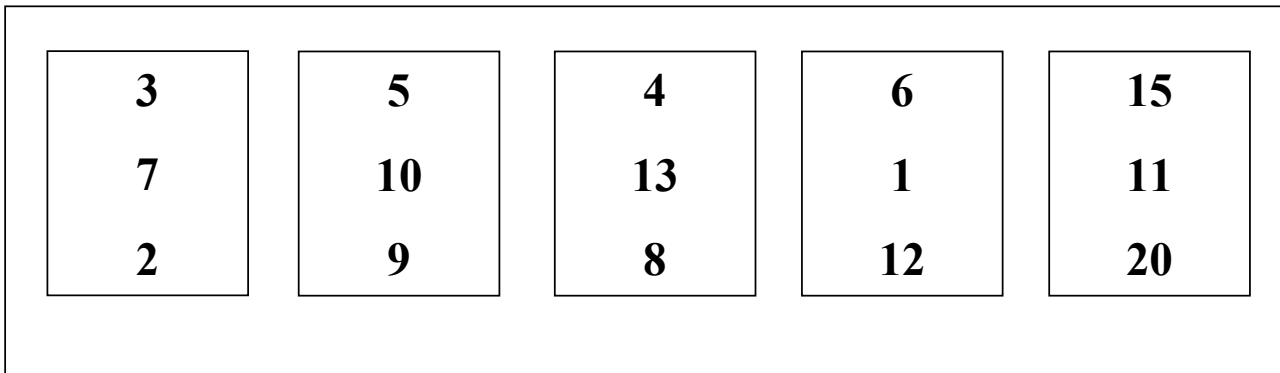


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

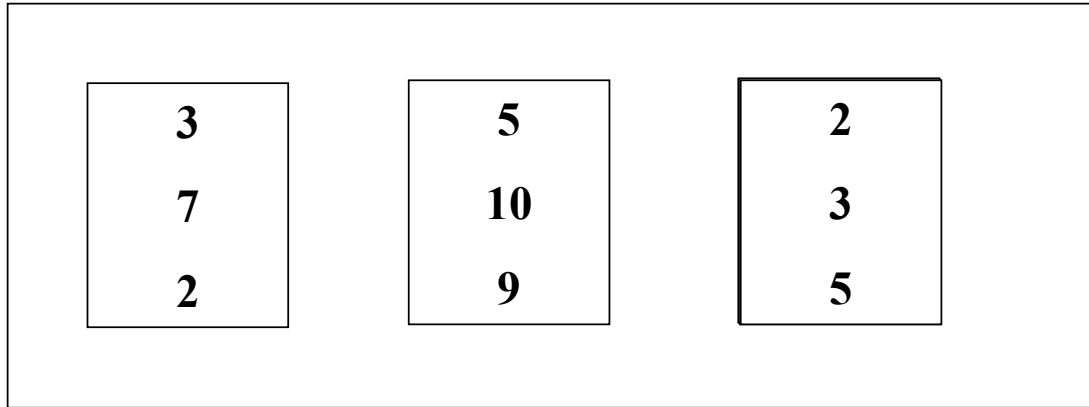


*disque*

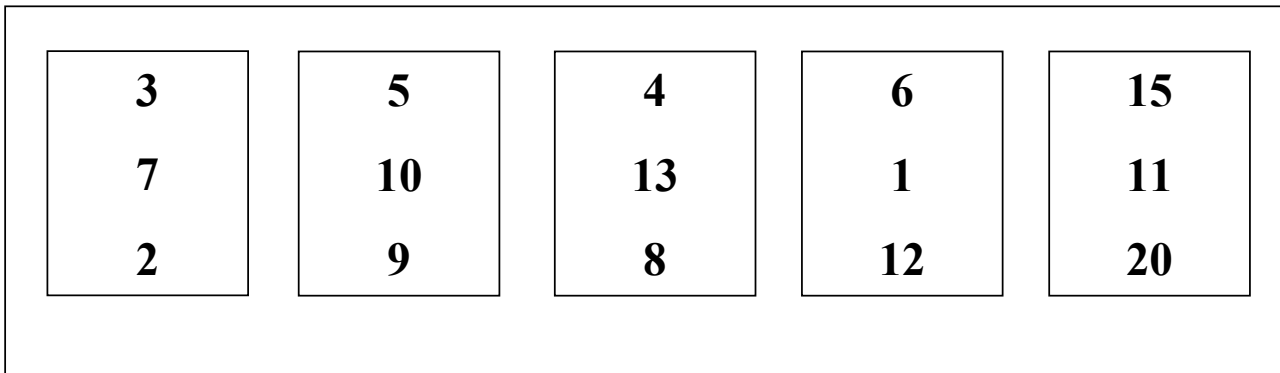


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

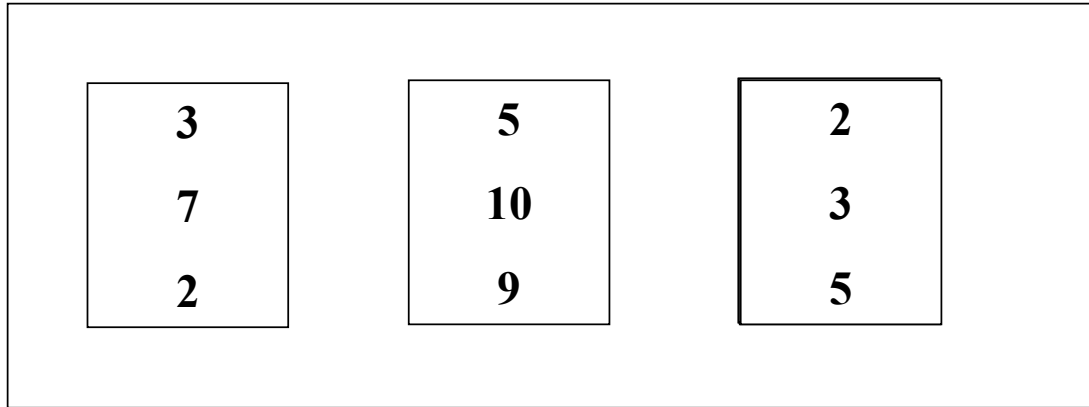


*disque*

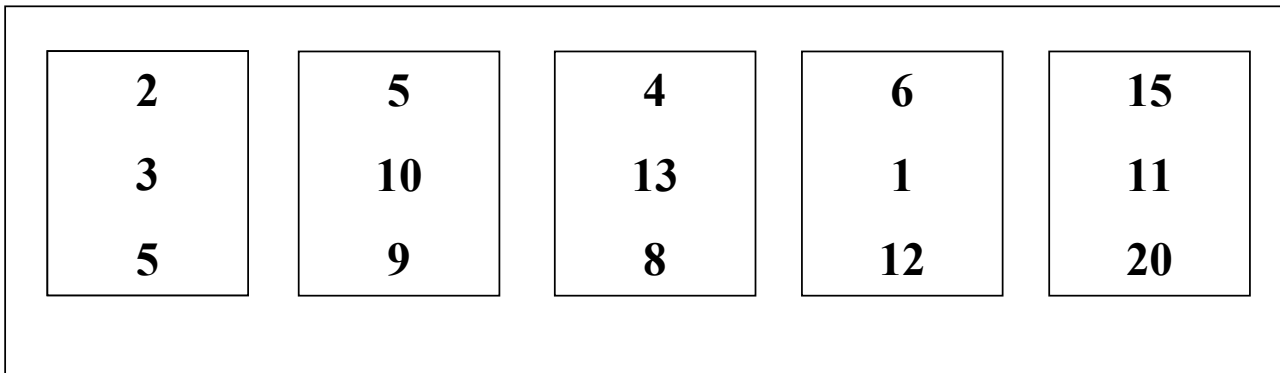


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

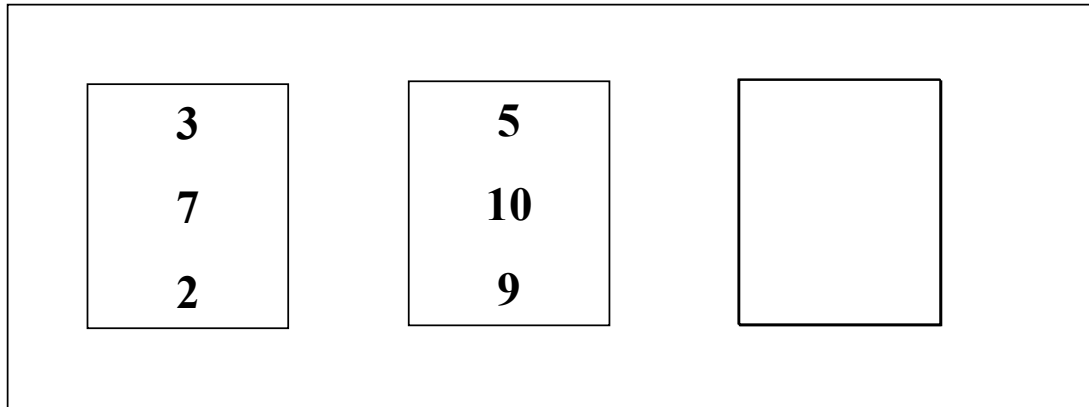


*disque*

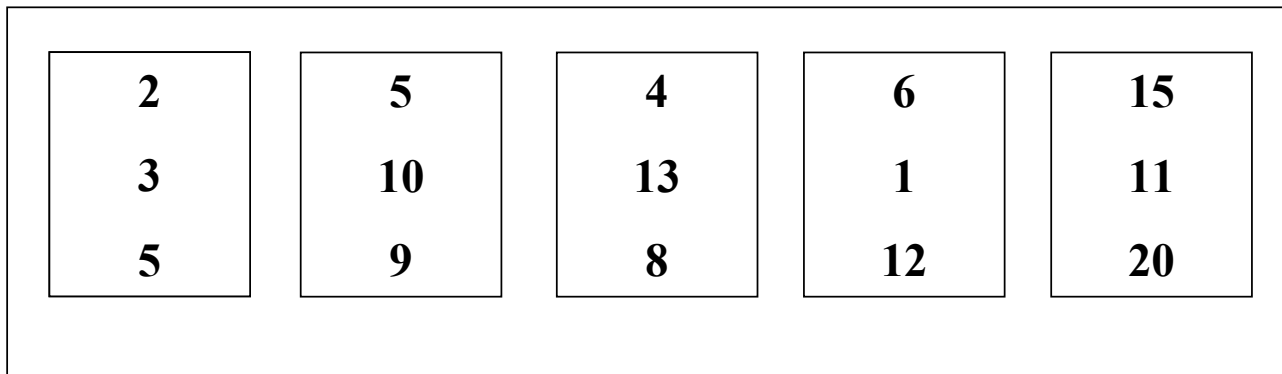


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

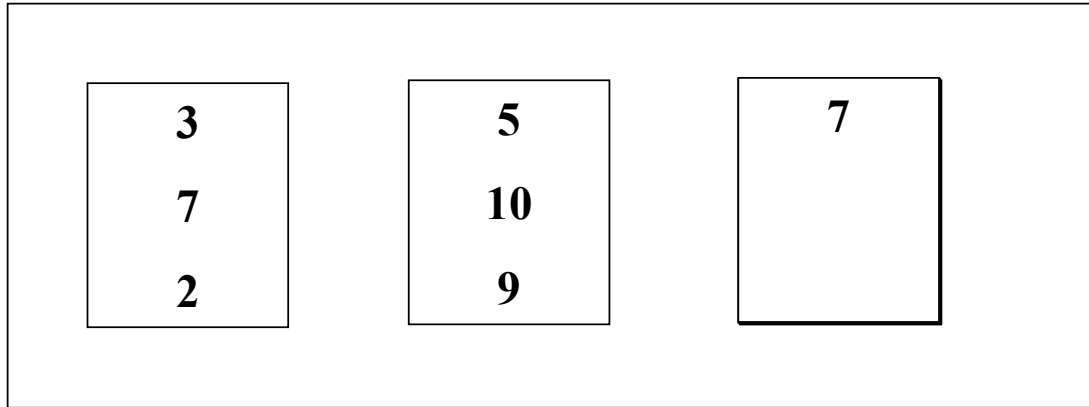


*disque*

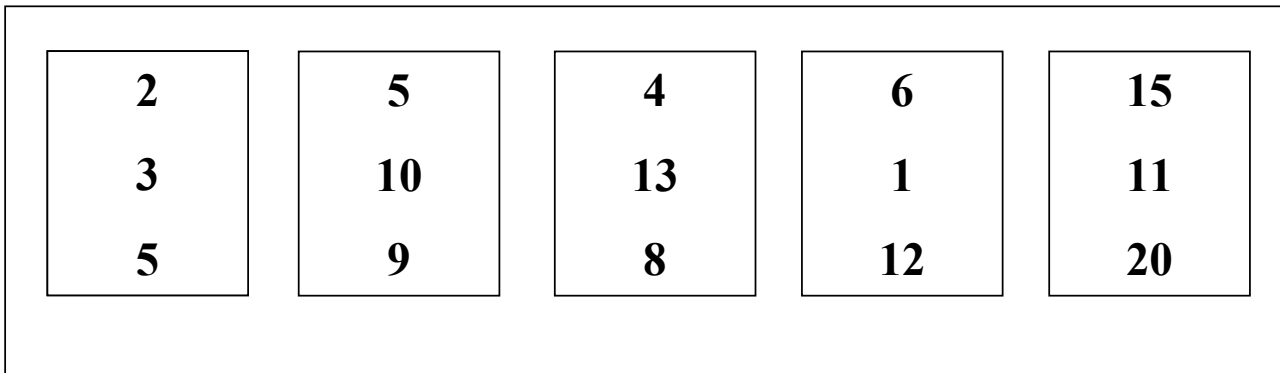


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

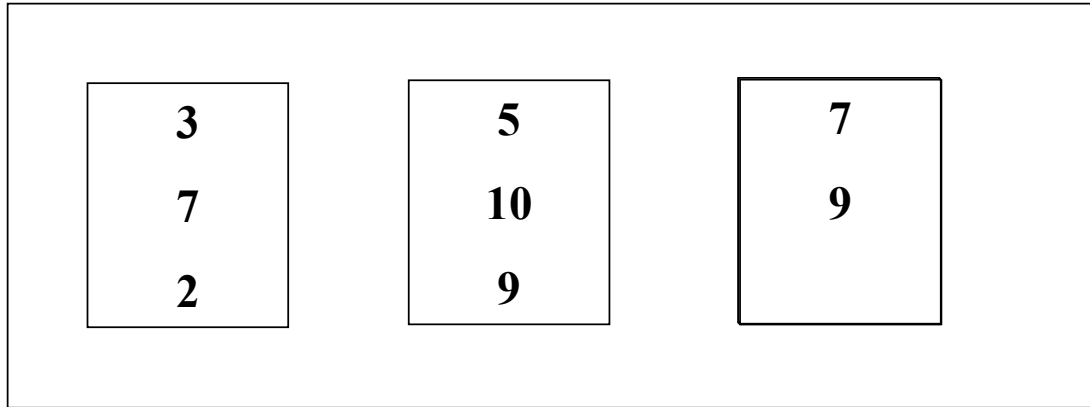


*disque*

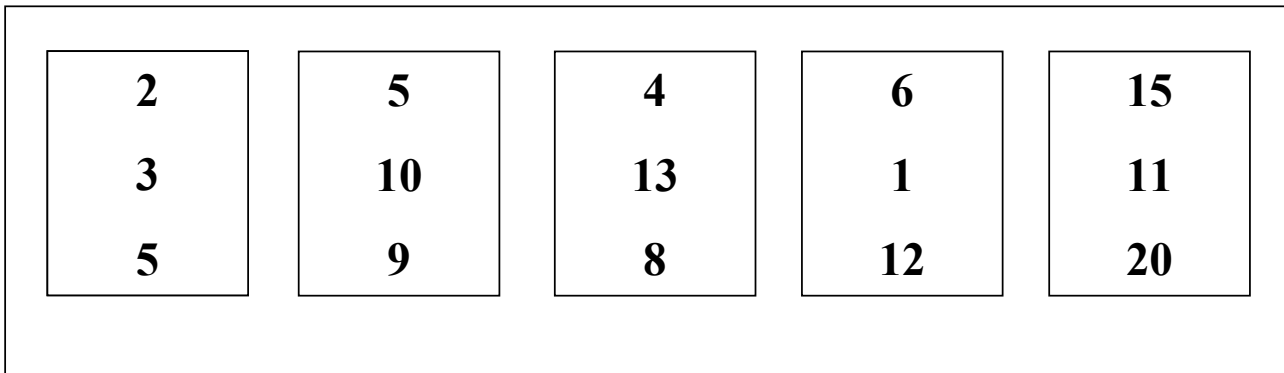


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

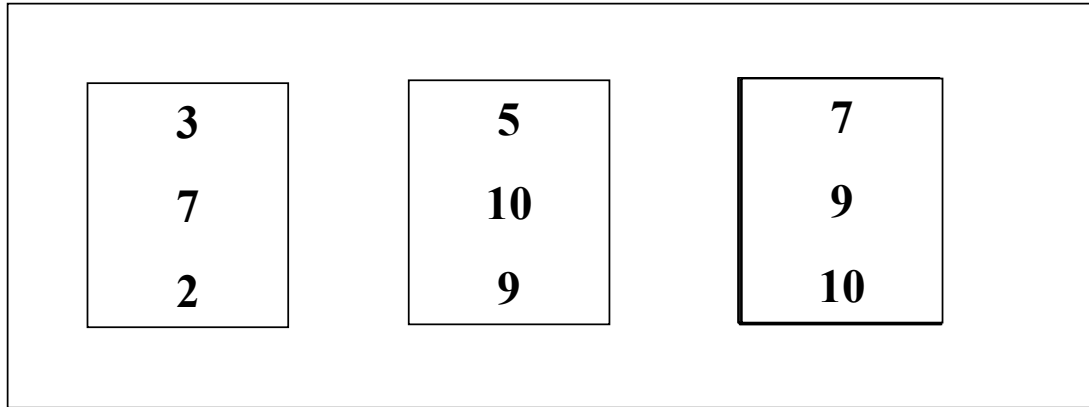


*disque*

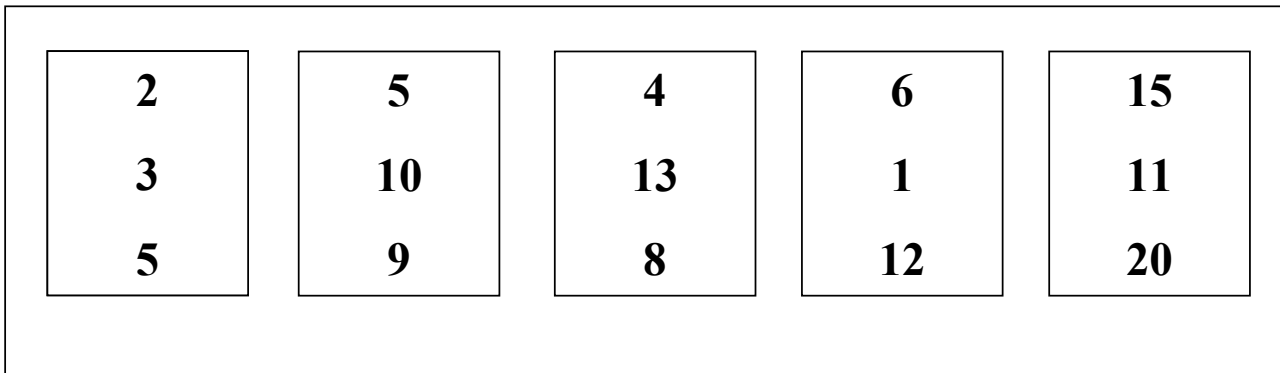


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

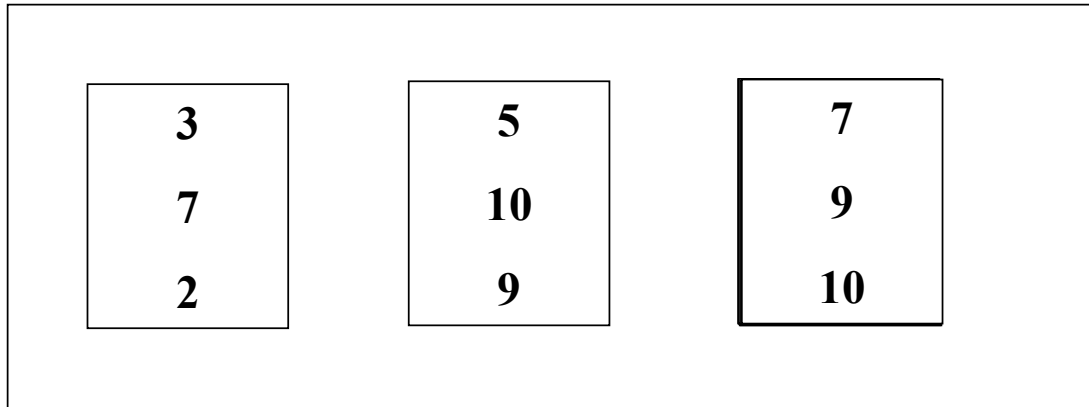


*disque*

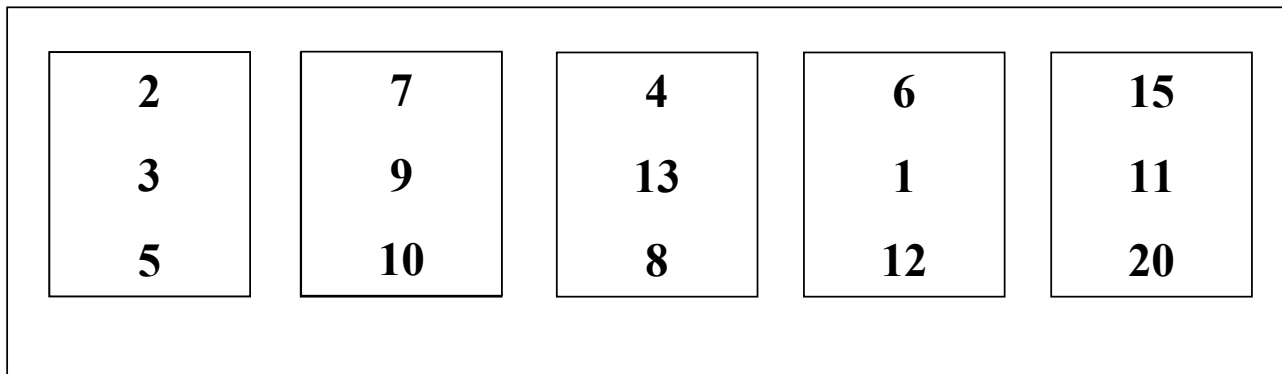


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*



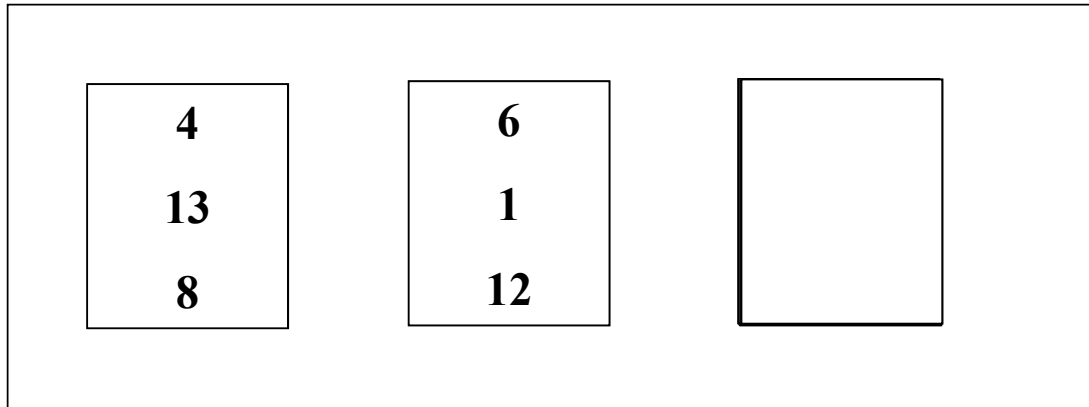
*disque*



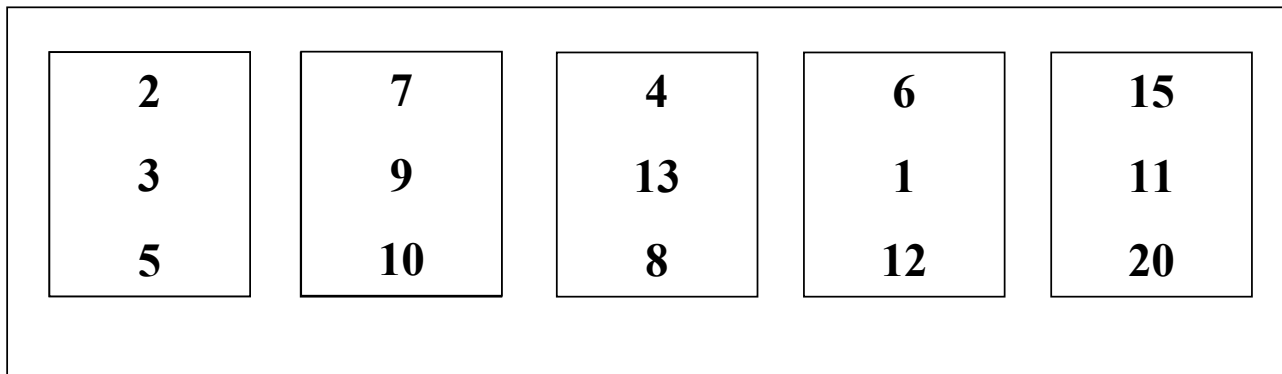


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

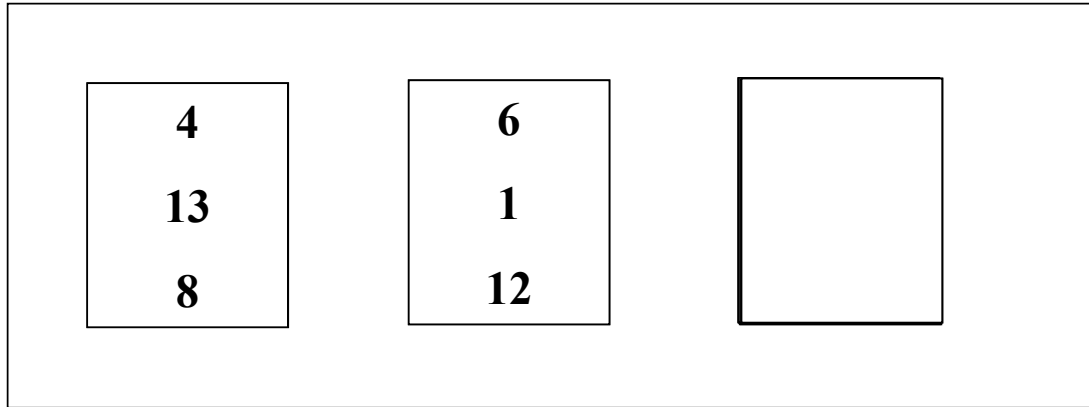


*disque*

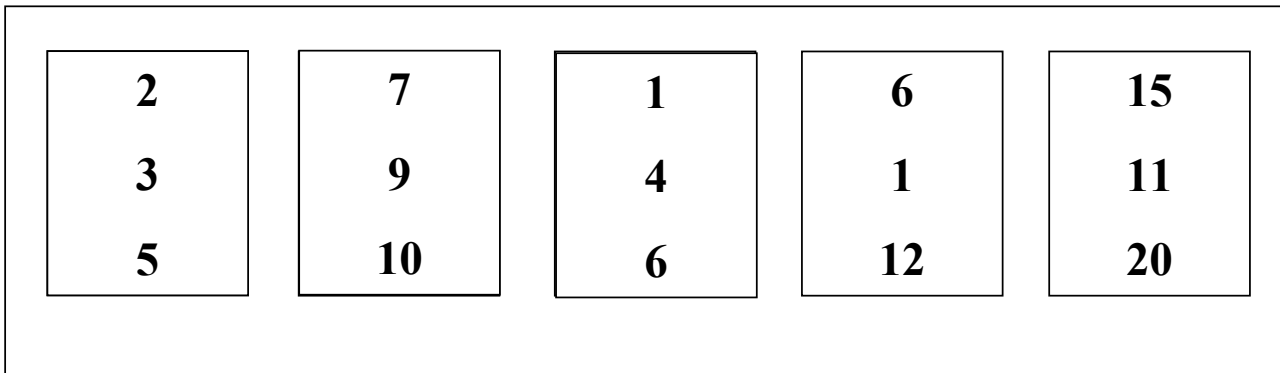


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

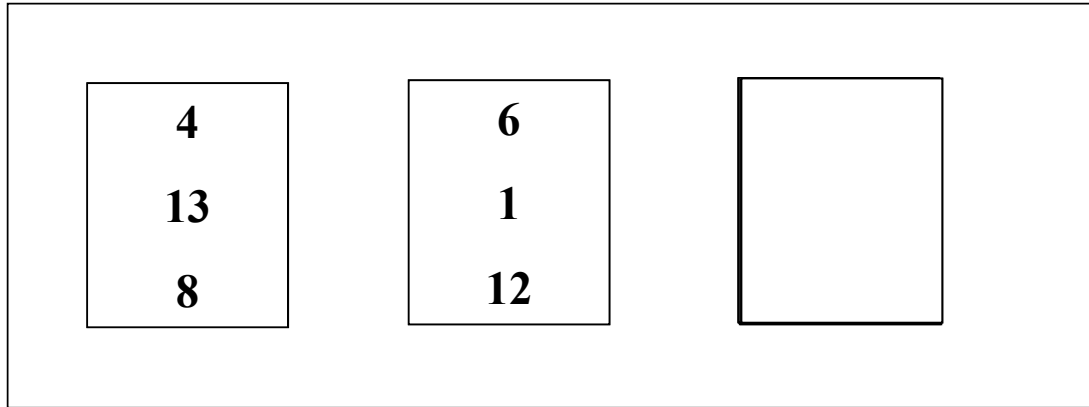


*disque*

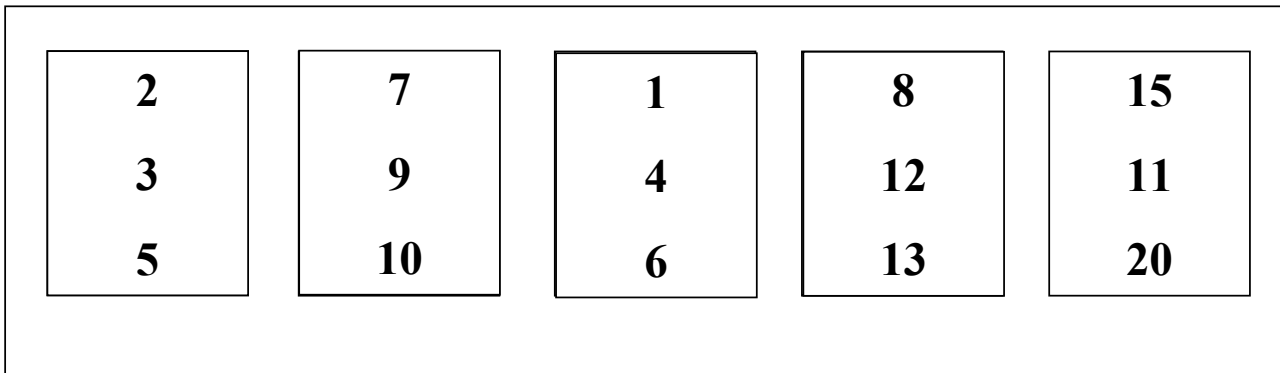


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

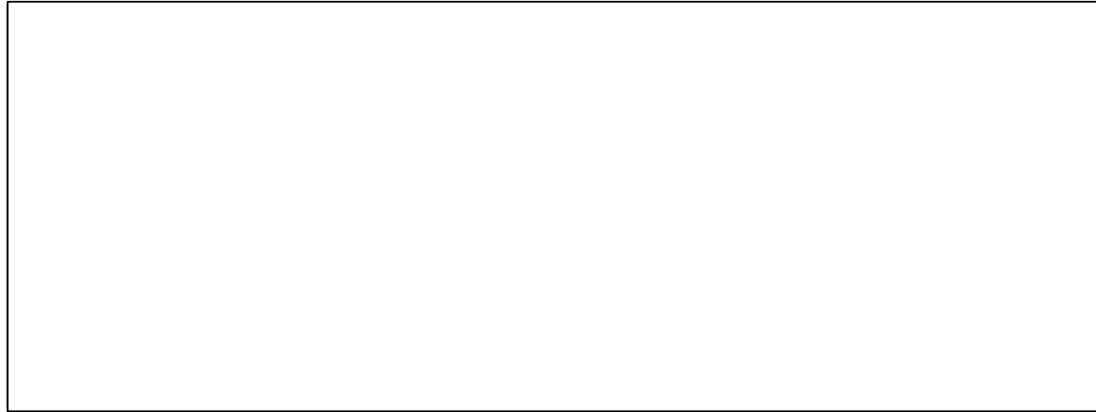


*disque*

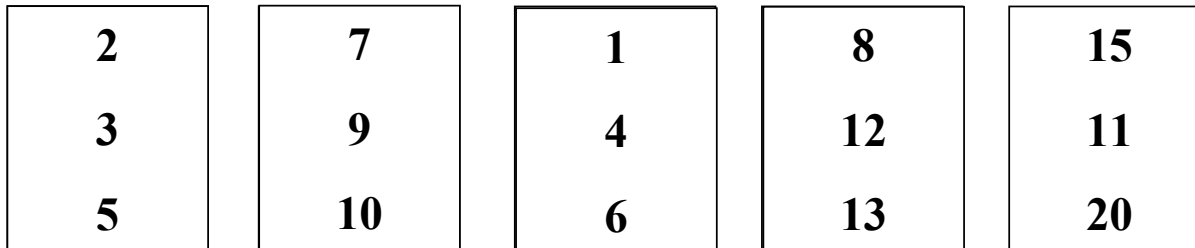


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

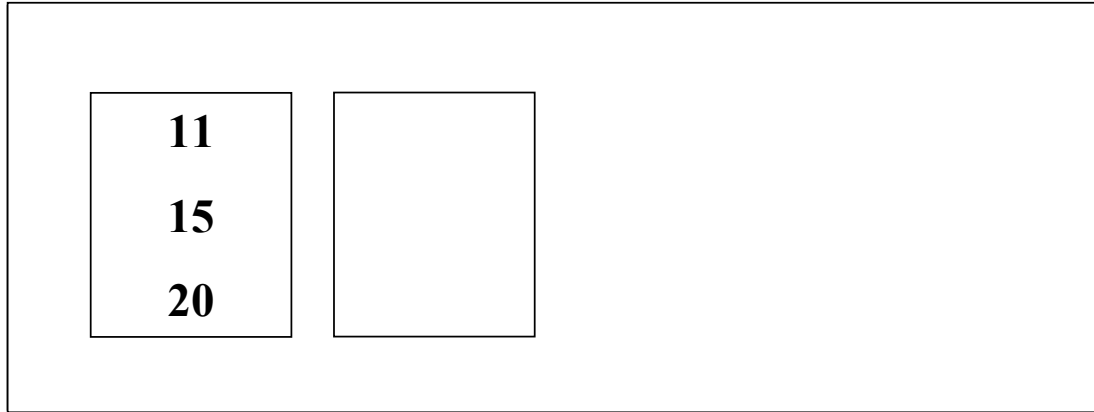


*disque*

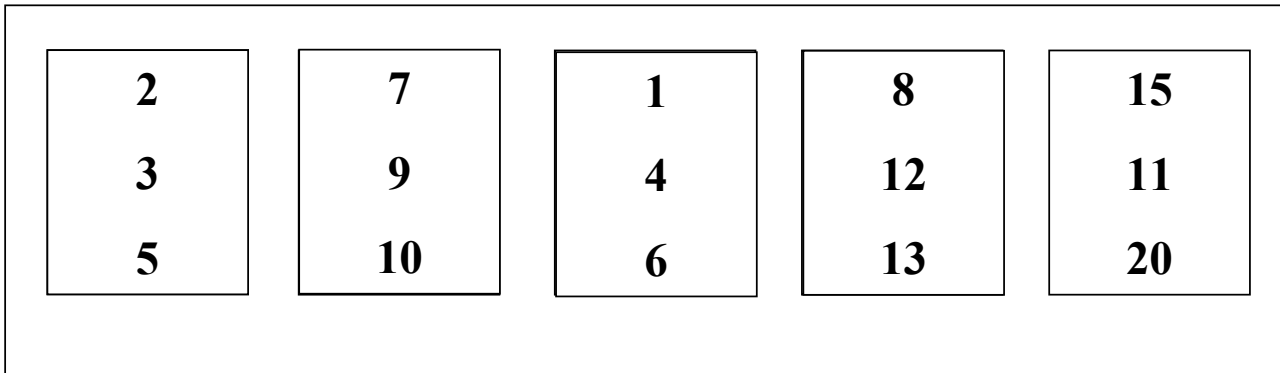


# Tri externe : 1ère étape

*La mémoire  
contient 3  
emplacements*

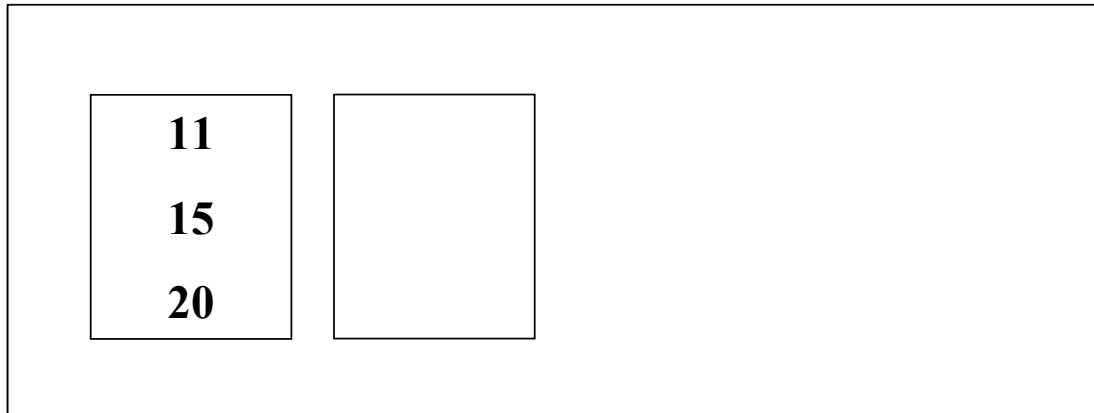


*disque*

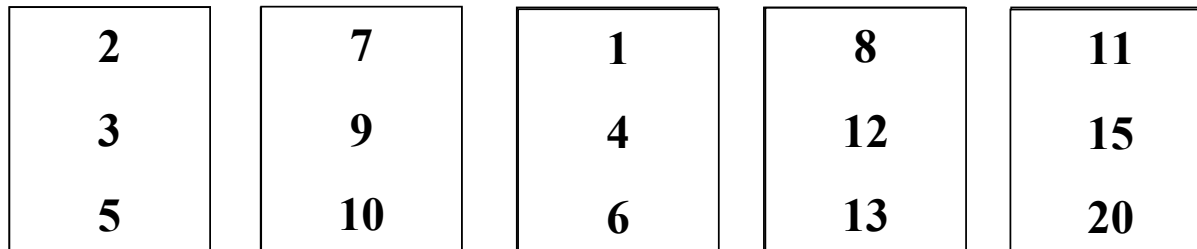


# Tri externe : 1ère étape

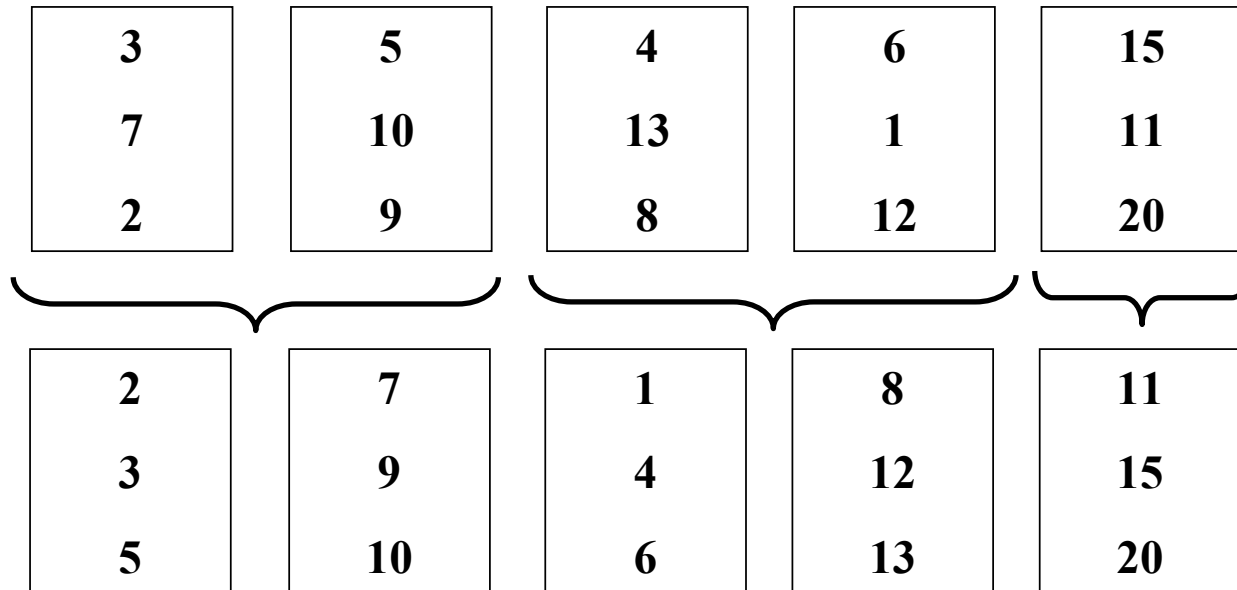
*La mémoire  
contient 3  
emplacements*



*disque*



# Tri externe

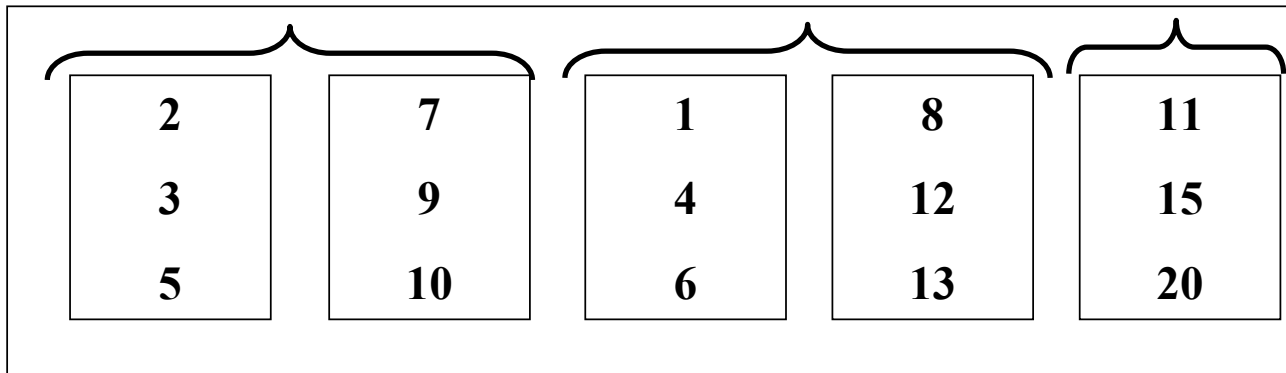


*Ensemble de  
départ*

*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*

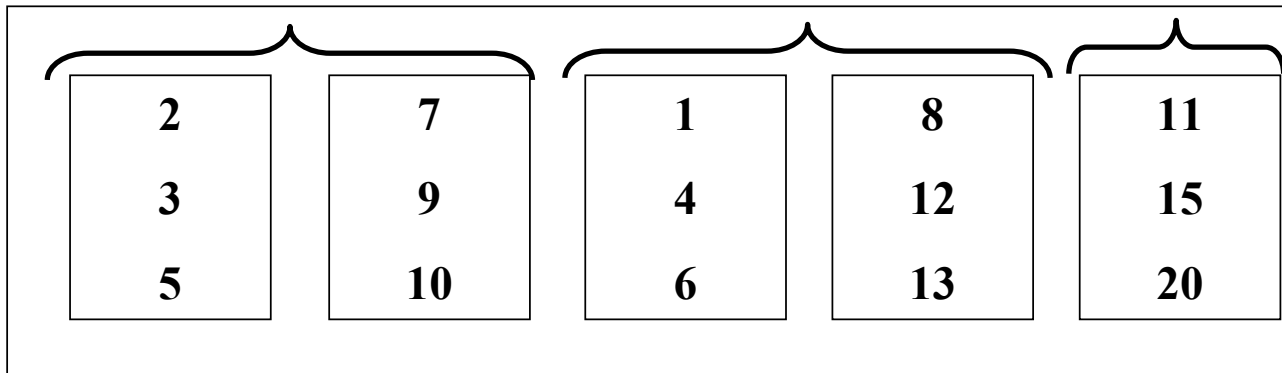


# Tri externe : 2ème étape

*La mémoire  
contient 3  
emplacements*

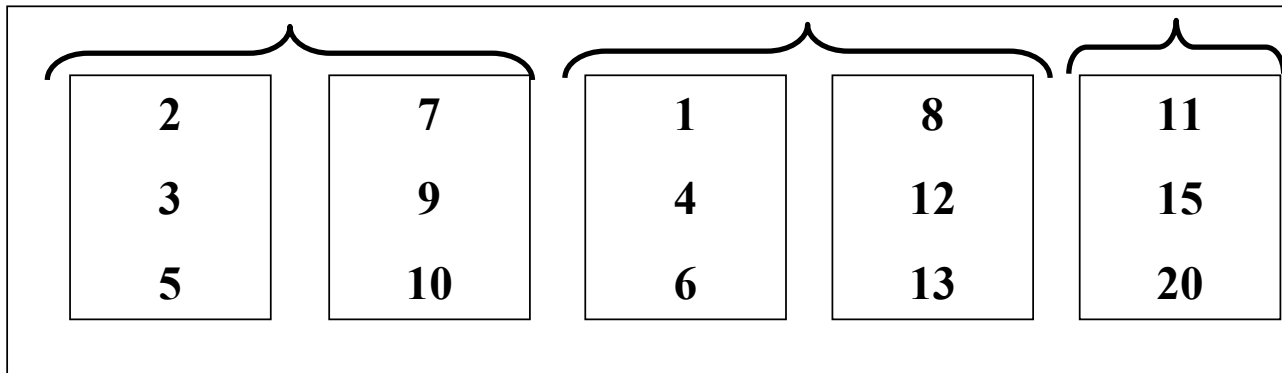
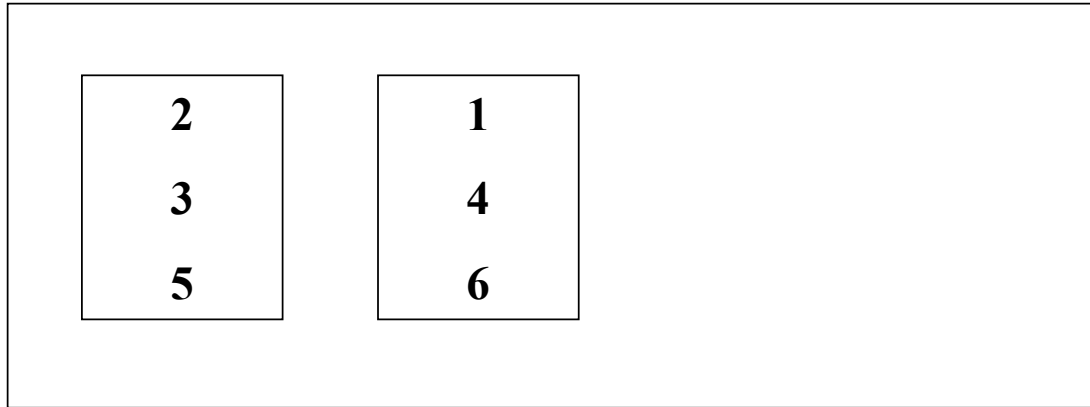


*disque*



# Tri externe : 2ème étape

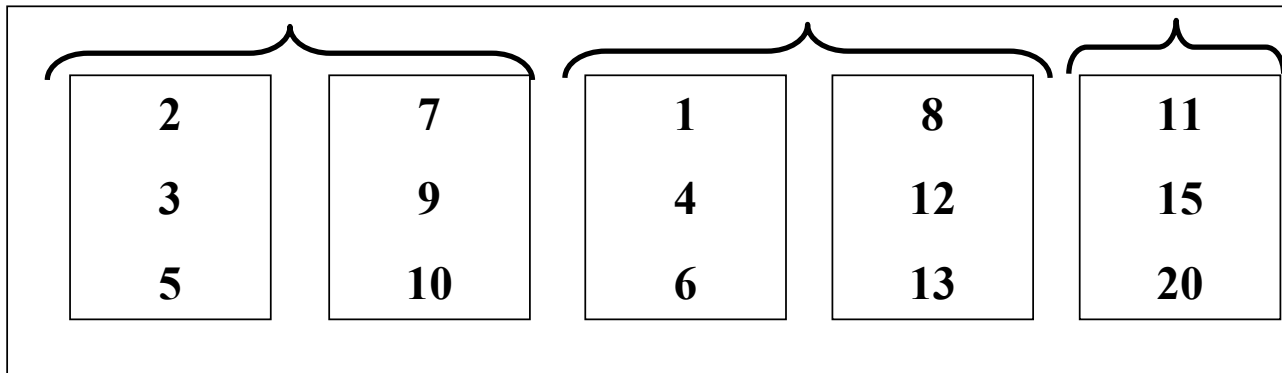
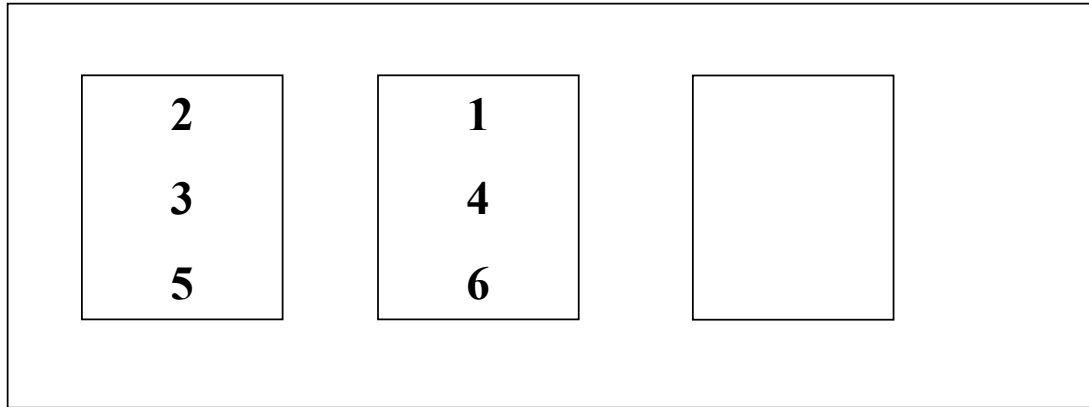
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

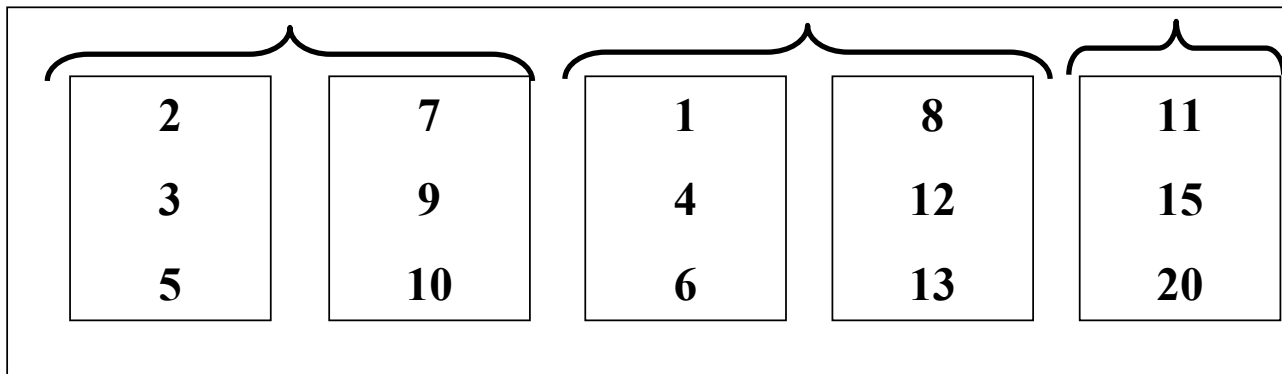
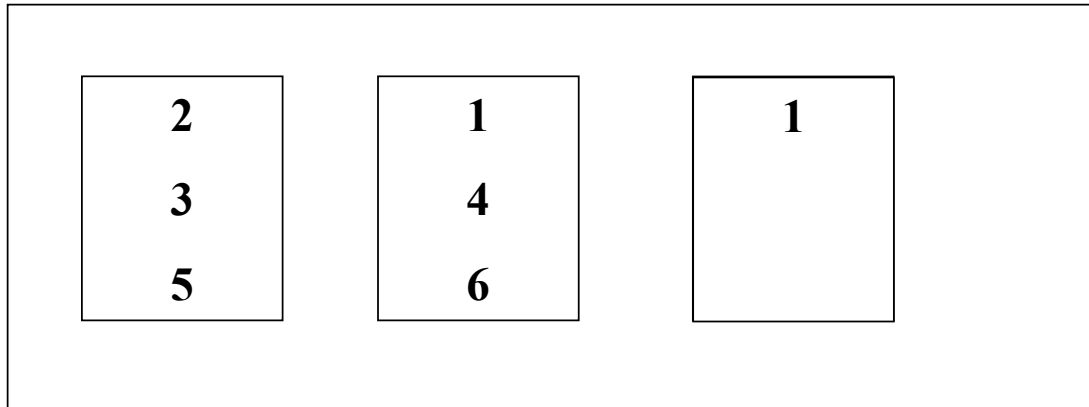
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

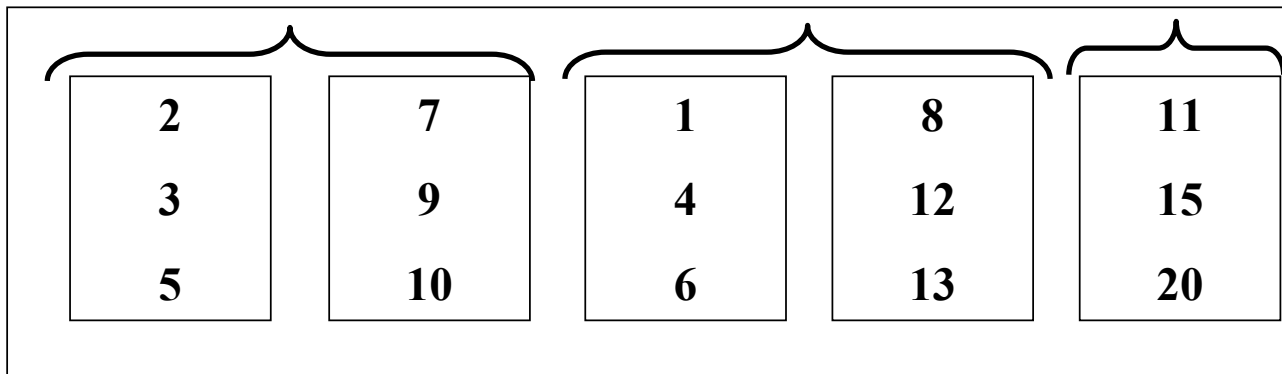
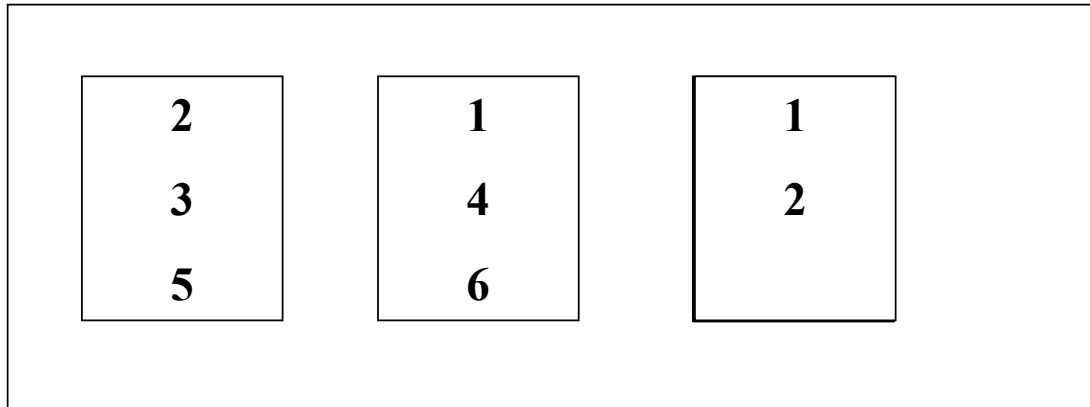
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

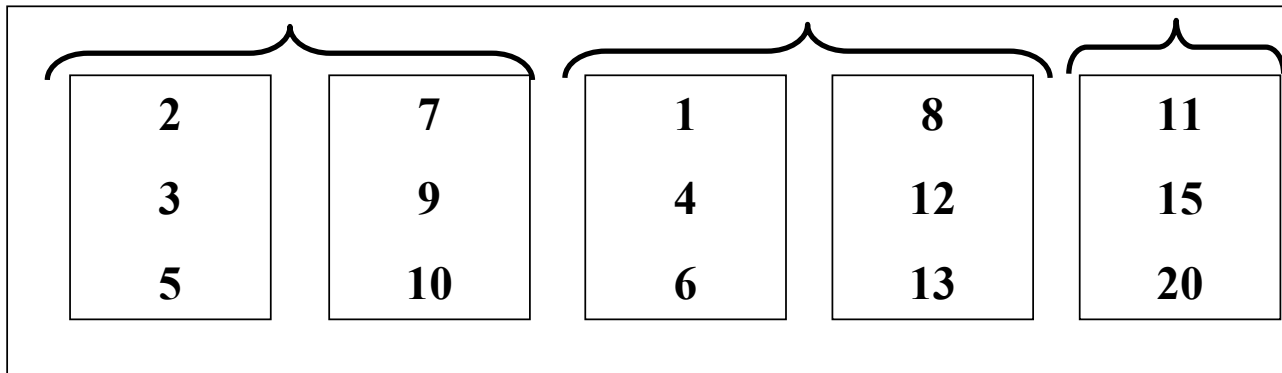
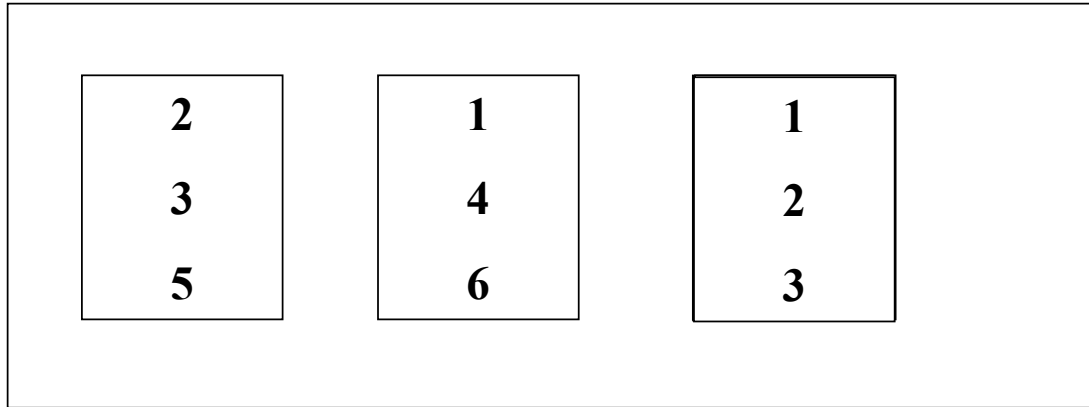
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

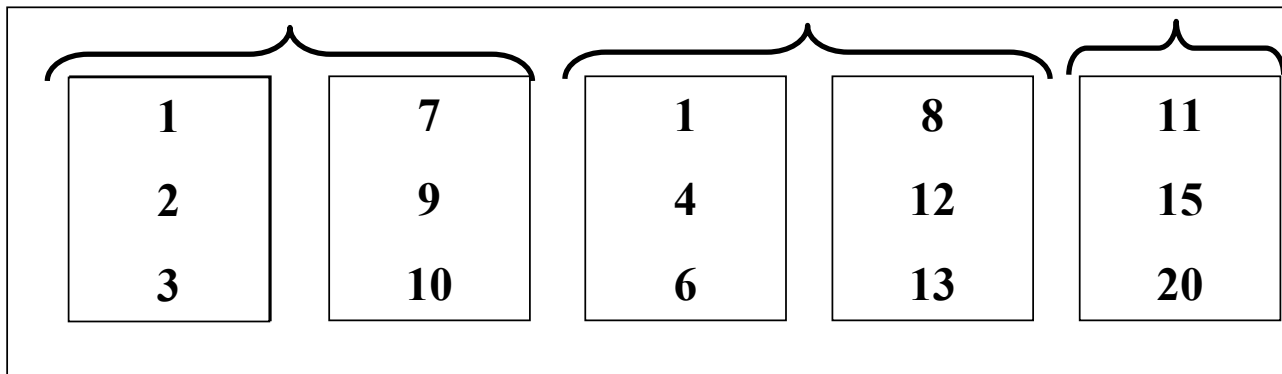
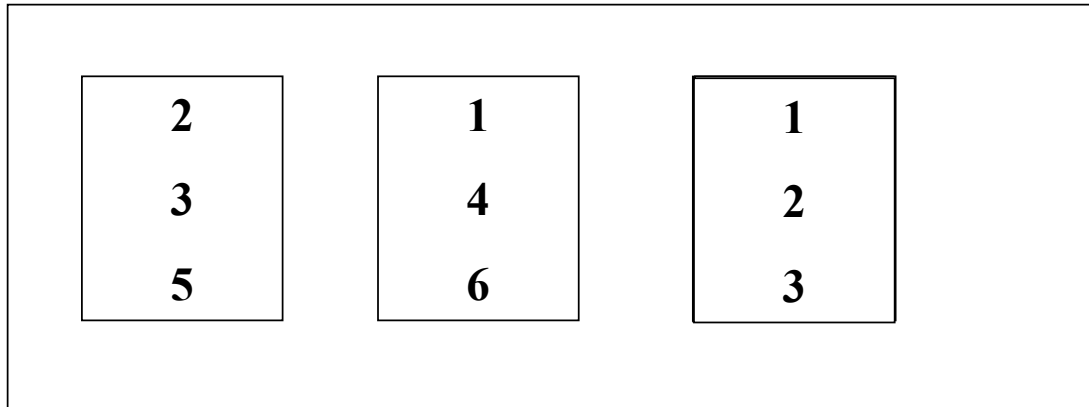
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

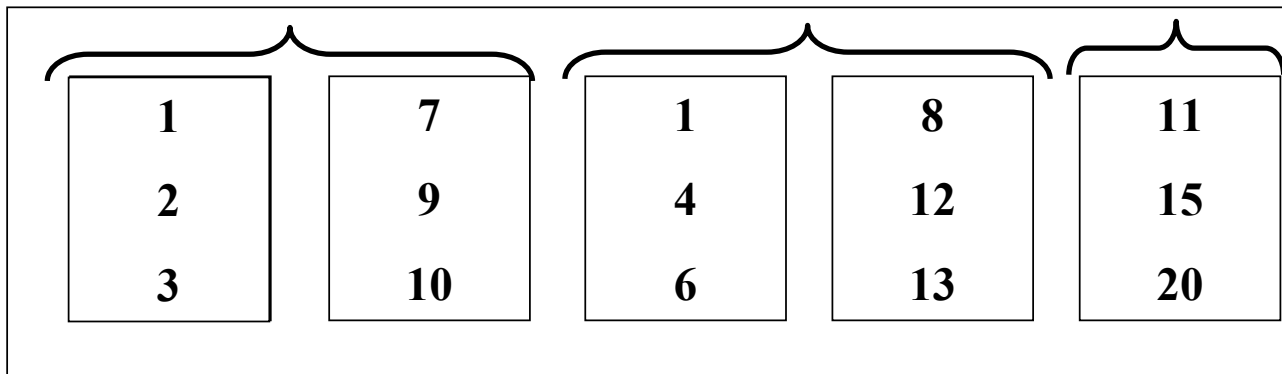
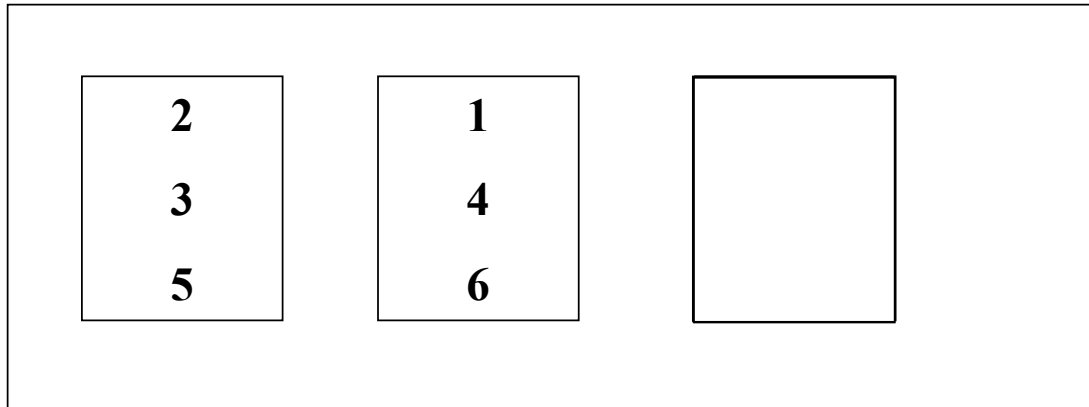
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

*La mémoire  
contient 3  
emplacements*

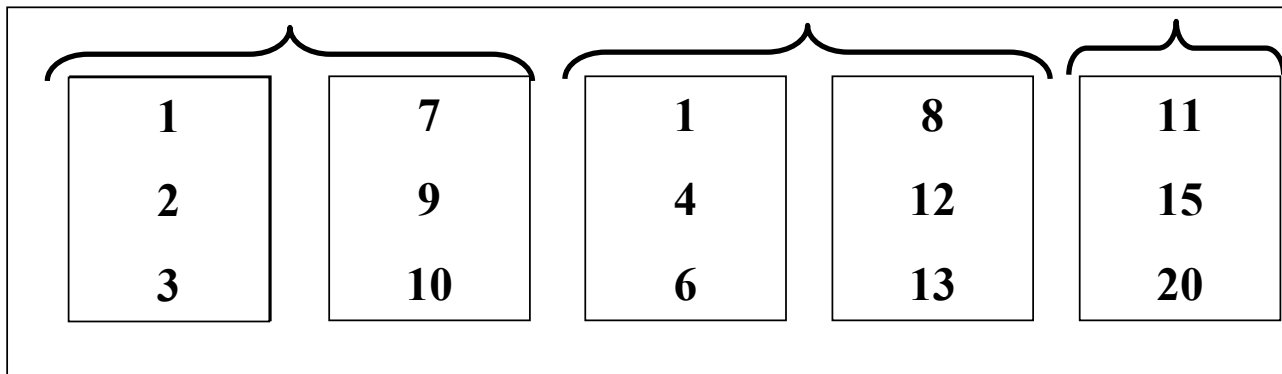
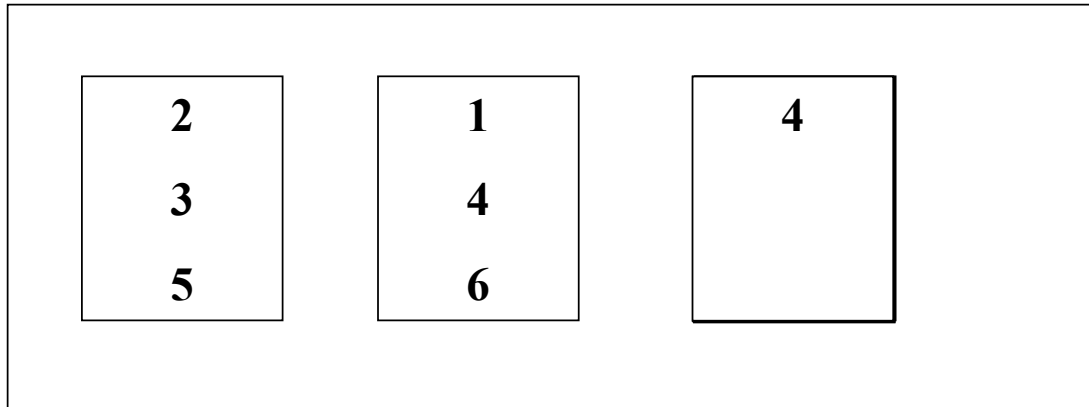


*disque*



# Tri externe : 2ème étape

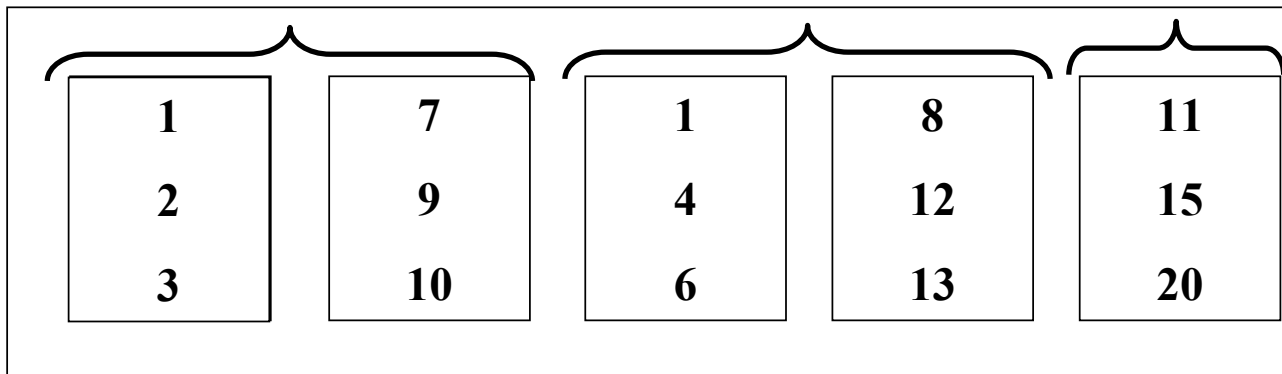
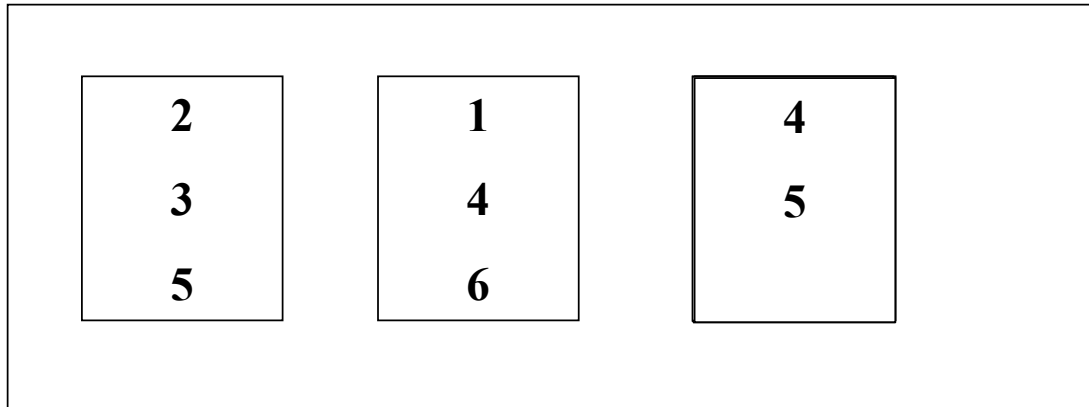
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

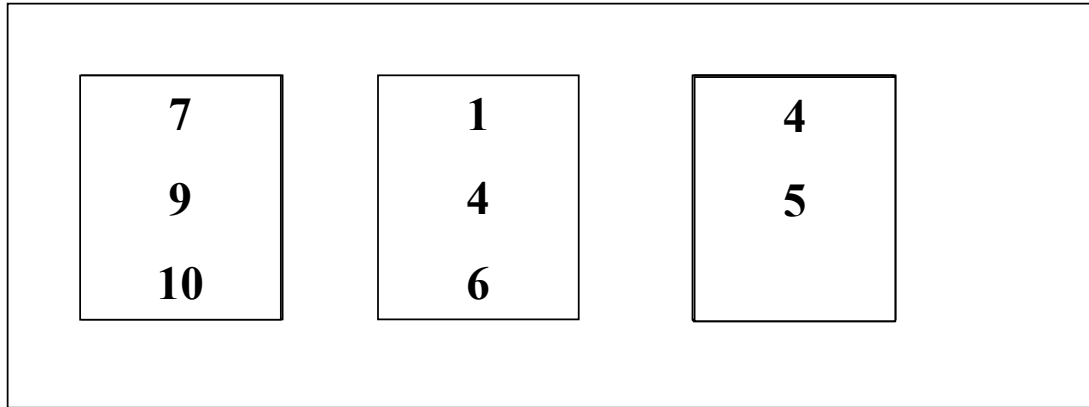
*La mémoire  
contient 3  
emplacements*



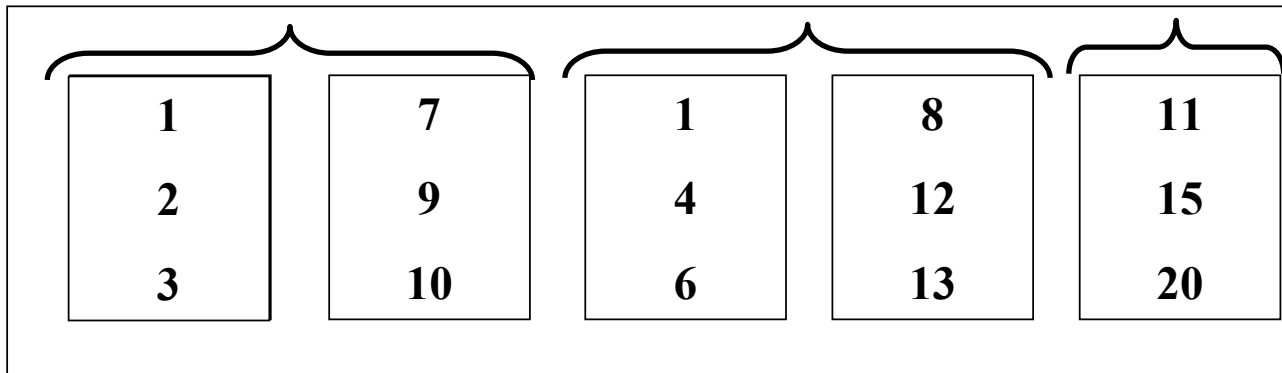
*disque*

# Tri externe : 2ème étape

*La mémoire  
contient 3  
emplacements*

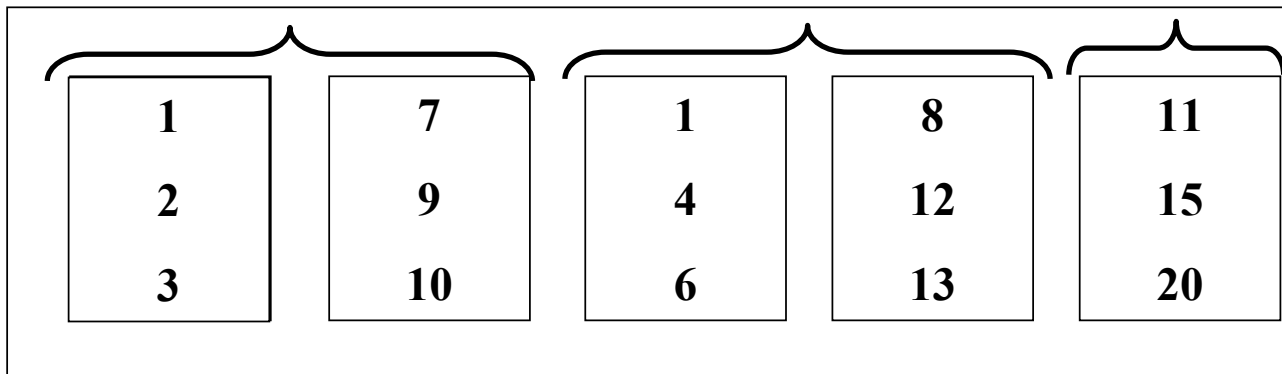
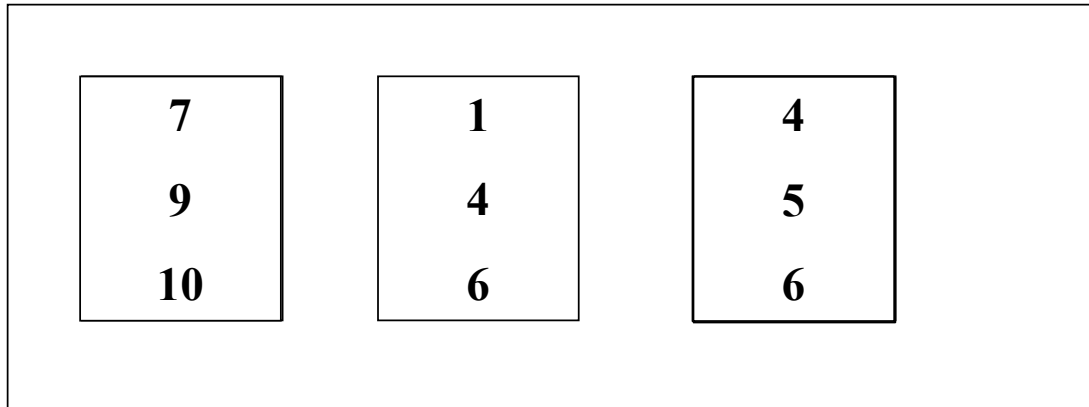


*disque*



# Tri externe : 2ème étape

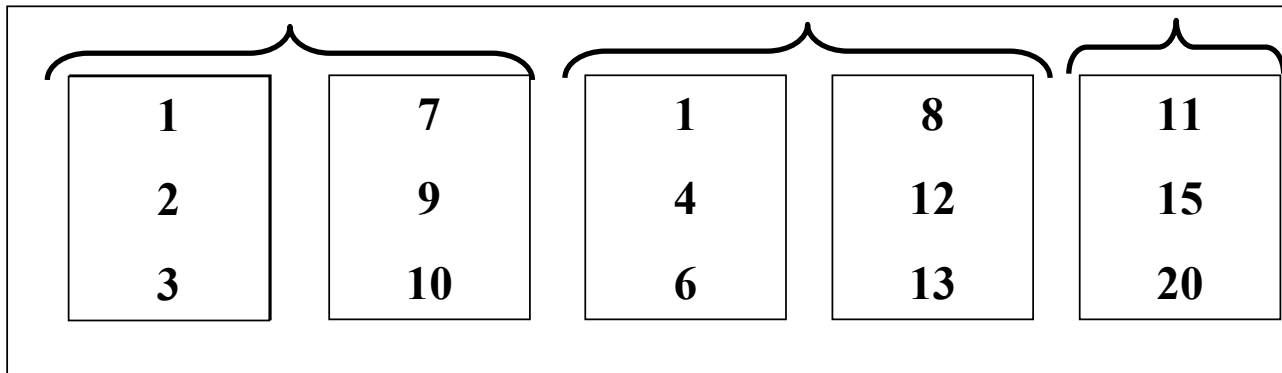
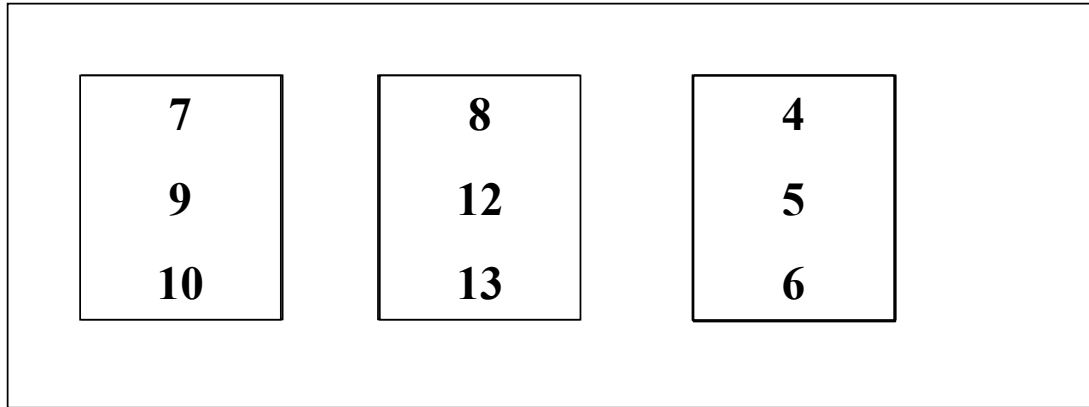
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

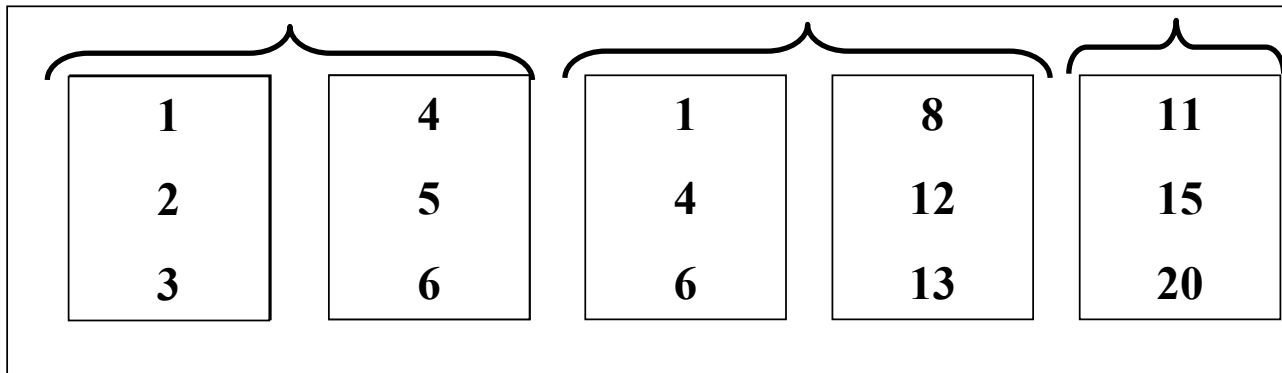
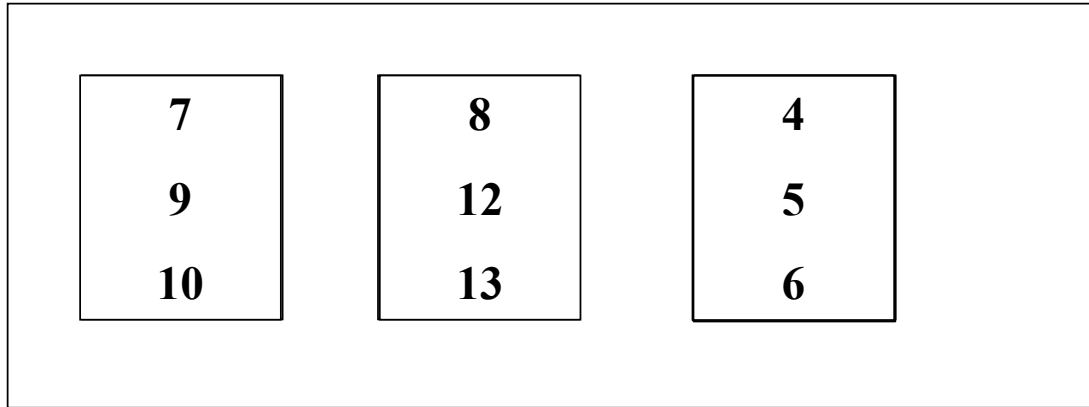
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

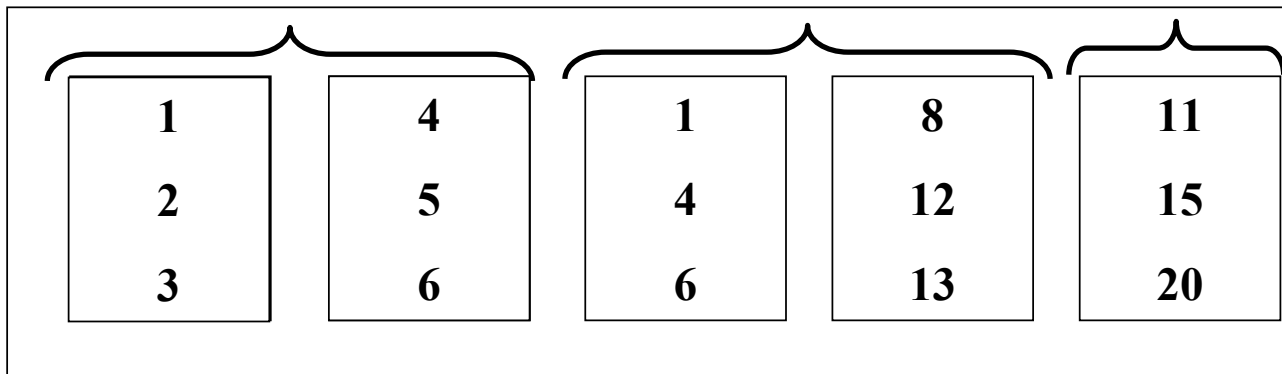
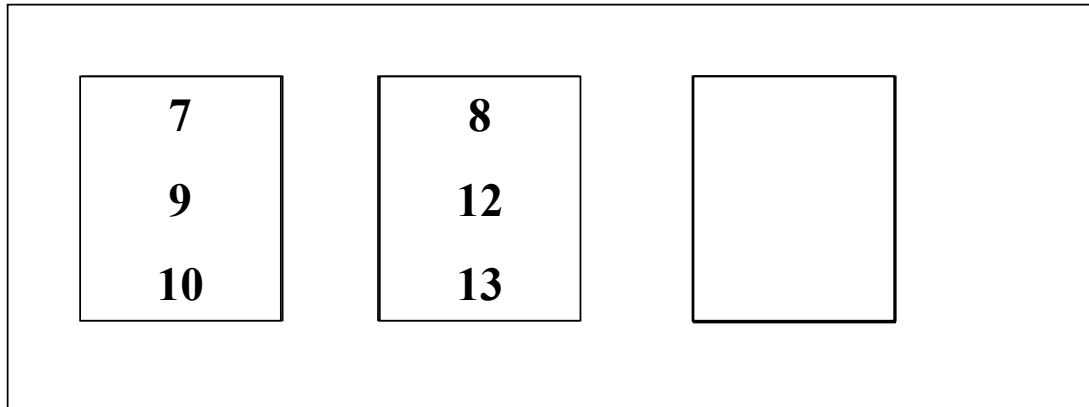
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

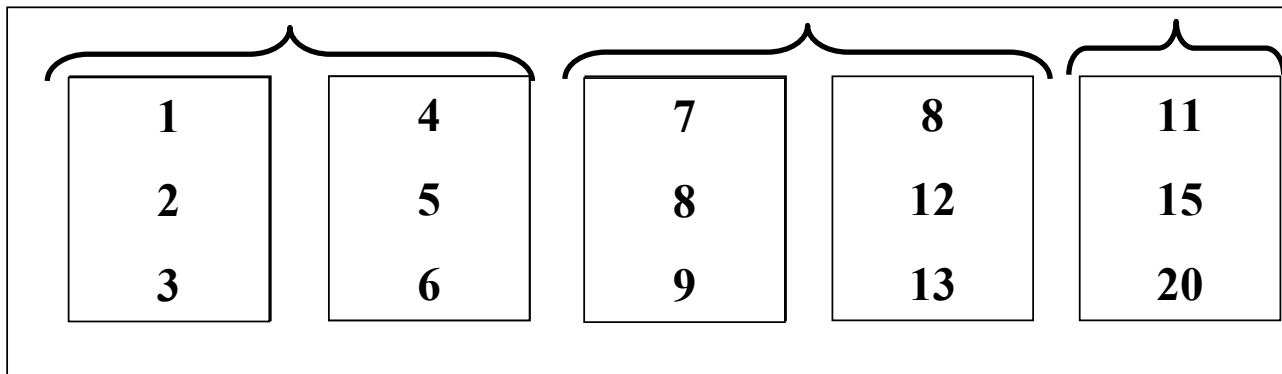
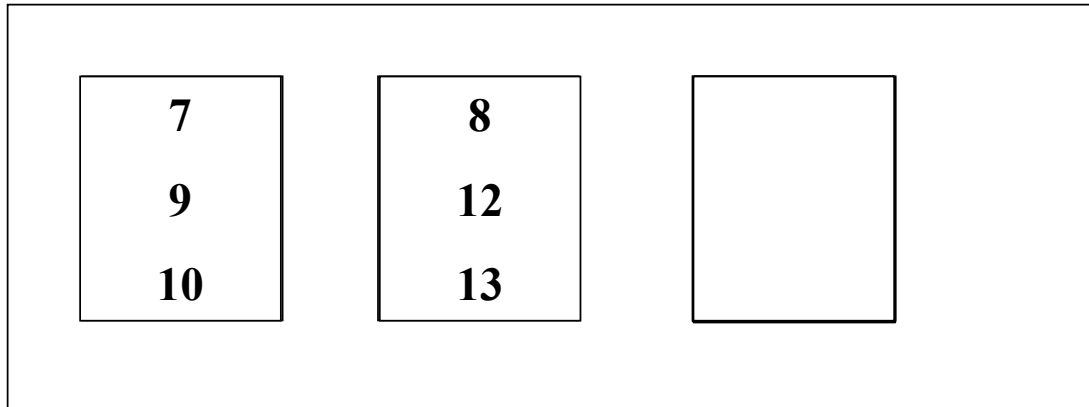
*La mémoire  
contient 3  
emplacements*



*disque*

# Tri externe : 2ème étape

*La mémoire  
contient 3  
emplacements*

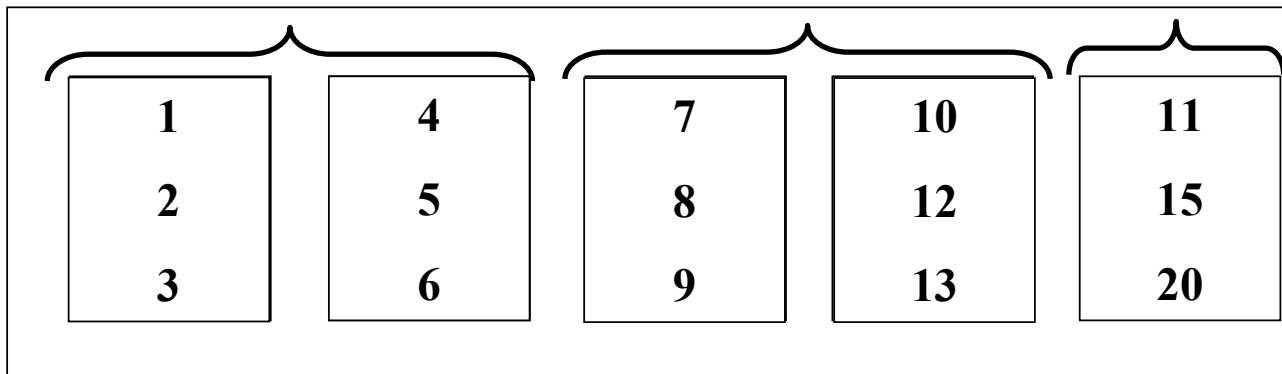
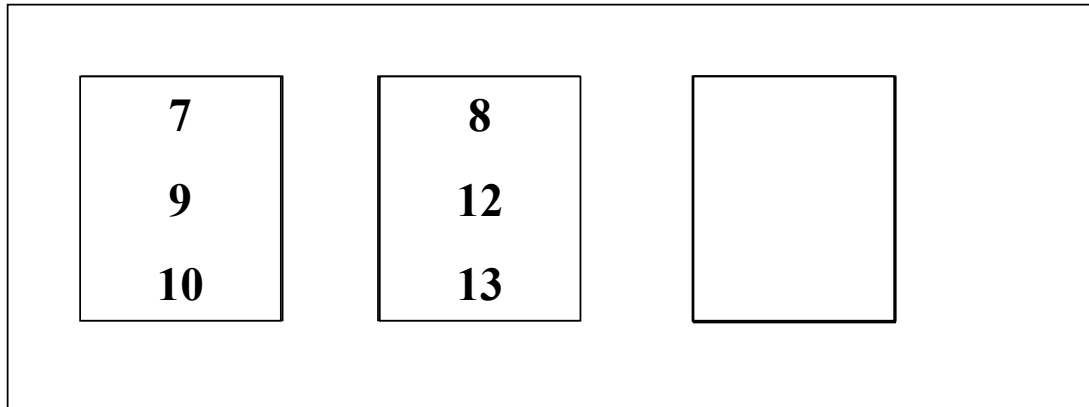


*disque*



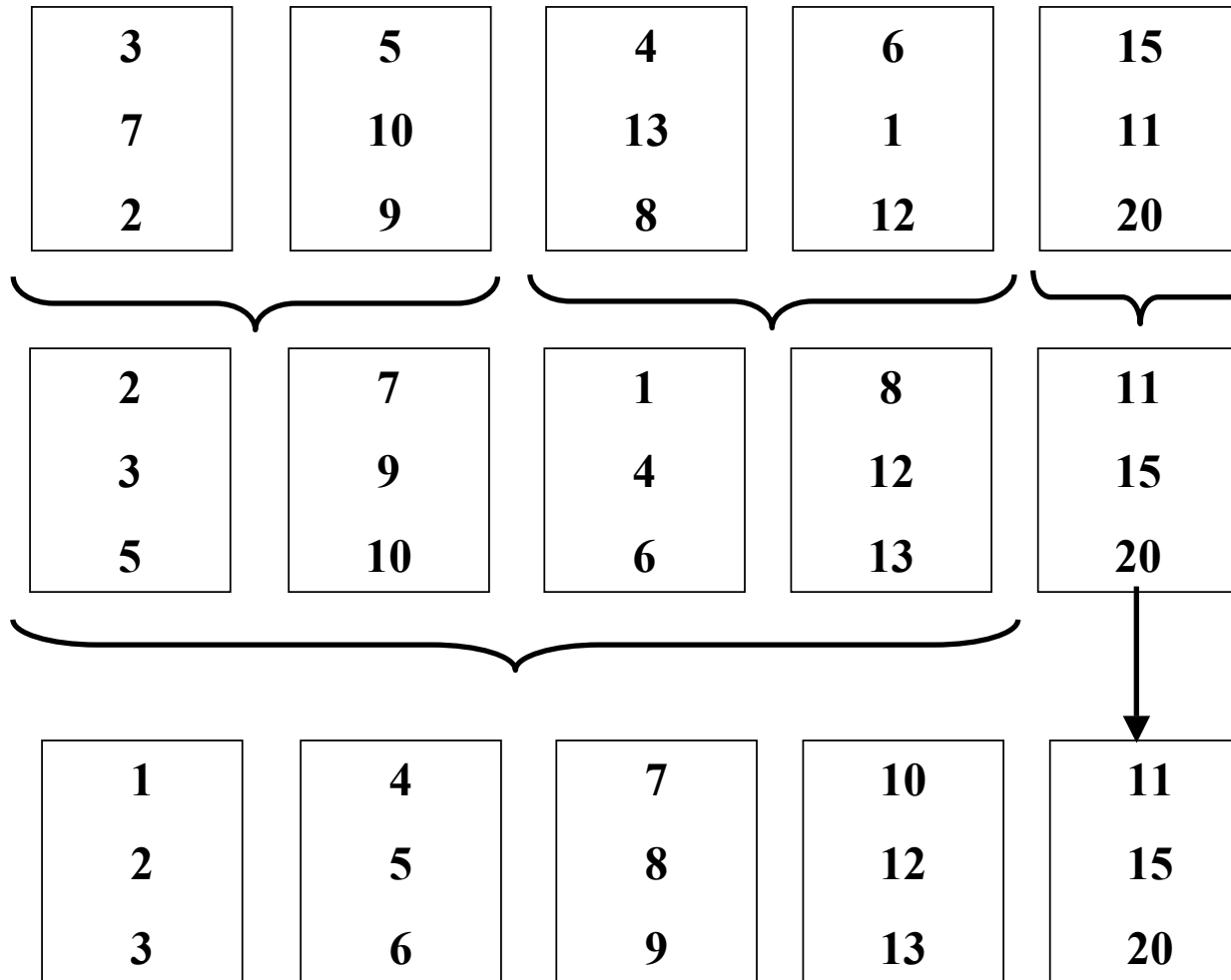
# Tri externe : 2ème étape

*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  $\Rightarrow$  3 paquets triés de 2 pages soit  $2*5$  E/S*

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

# Fusion

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

<b>SID</b>	<b>Département</b>	<b>Jour</b>
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	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
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	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	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	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
31	Paul	12	200KF	Compta	03/02/2001
...	...	...	...	...	...

# 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
31	Paul	12	200KF	Compta	03/02/2001
...	...	...	...	...	...



# 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
31	Paul	12	200KF	Compta	03/02/2001
...	...	...	...	...	...

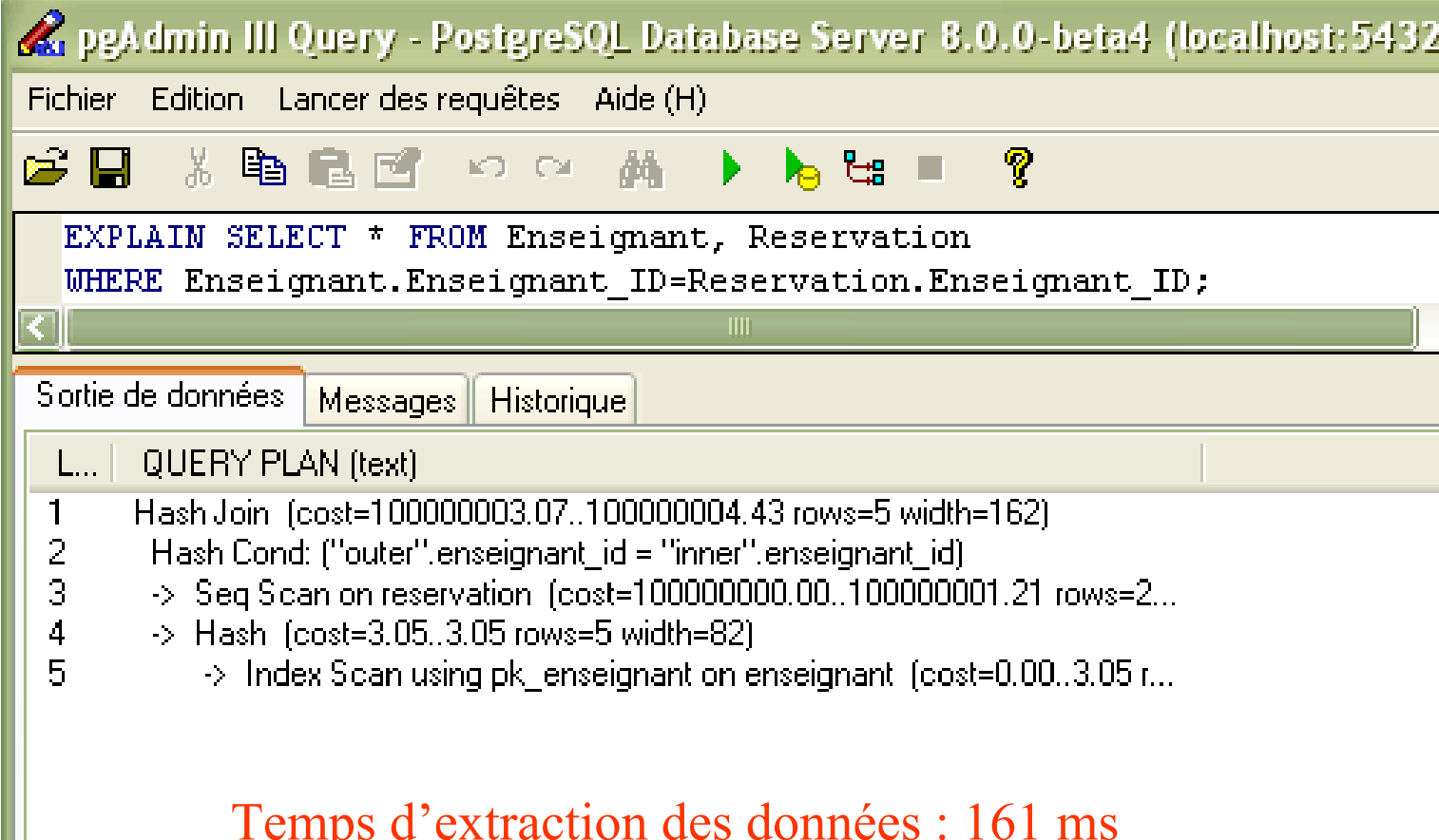
# Exemples en utilisant PostgreSQL

## 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 icons for file operations, query execution, and help. The query editor contains the following SQL query:

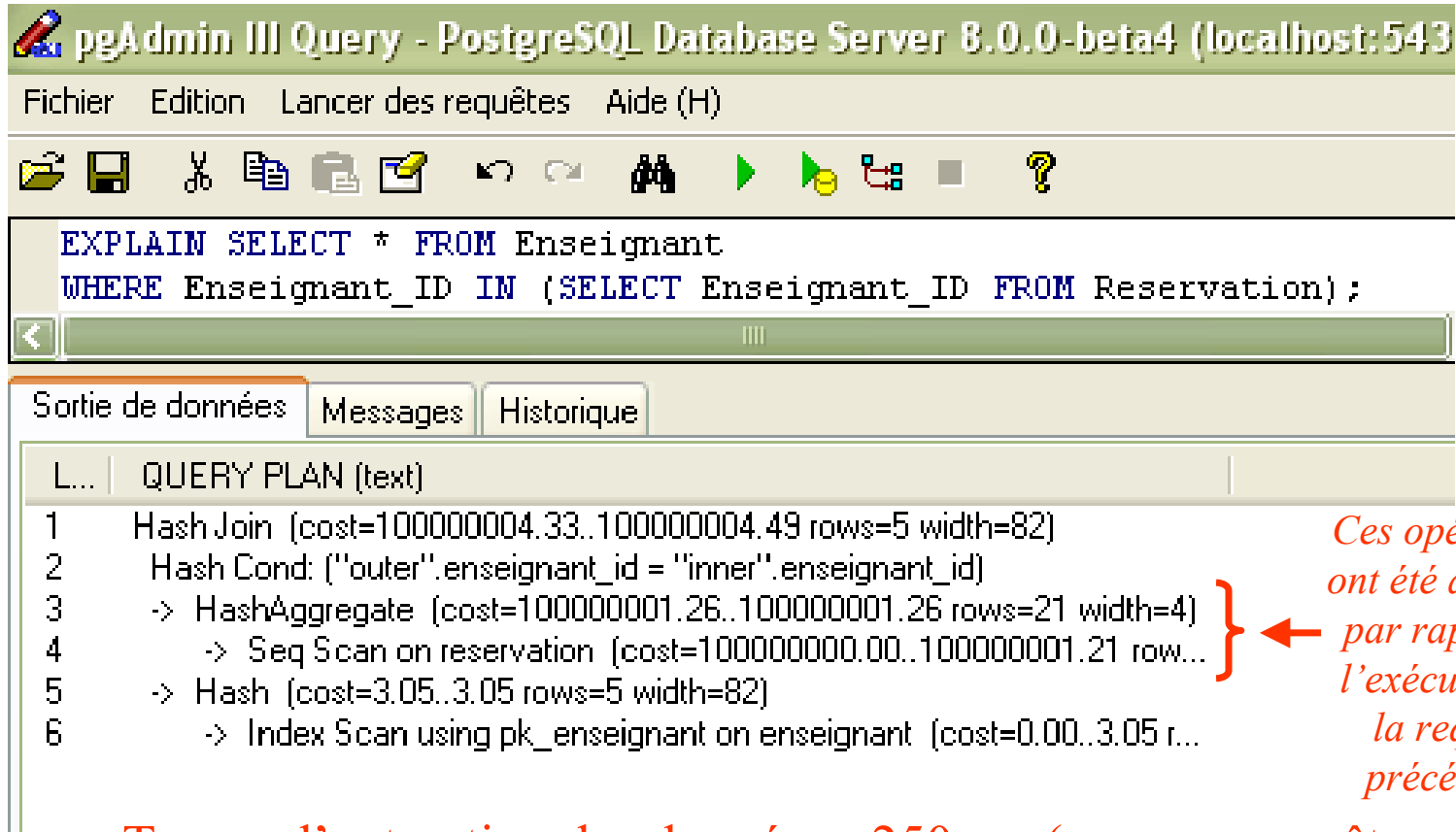
```
EXPLAIN SELECT * FROM Enseignant, Reservation
WHERE Enseignant.Enseignant_ID=Reservation.Enseignant_ID;
```

Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is selected, showing the query plan (text):

```
L... | QUERY PLAN (text)
1   Hash Join (cost=1000000003.07..1000000004.43 rows=5 width=162)
2   Hash Cond: ("outer".enseignant_id = "inner".enseignant_id)
3   -> Seq Scan on reservation (cost=1000000000.00..1000000001.21 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...
```

At the bottom of the screenshot, the text "Temps d'extraction des données : 161 ms" is displayed in red.

# 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)". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, execution, and help. The SQL editor contains the following query:

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

The "Sortie de données" tab is selected, displaying the "QUERY PLAN (text)" output:

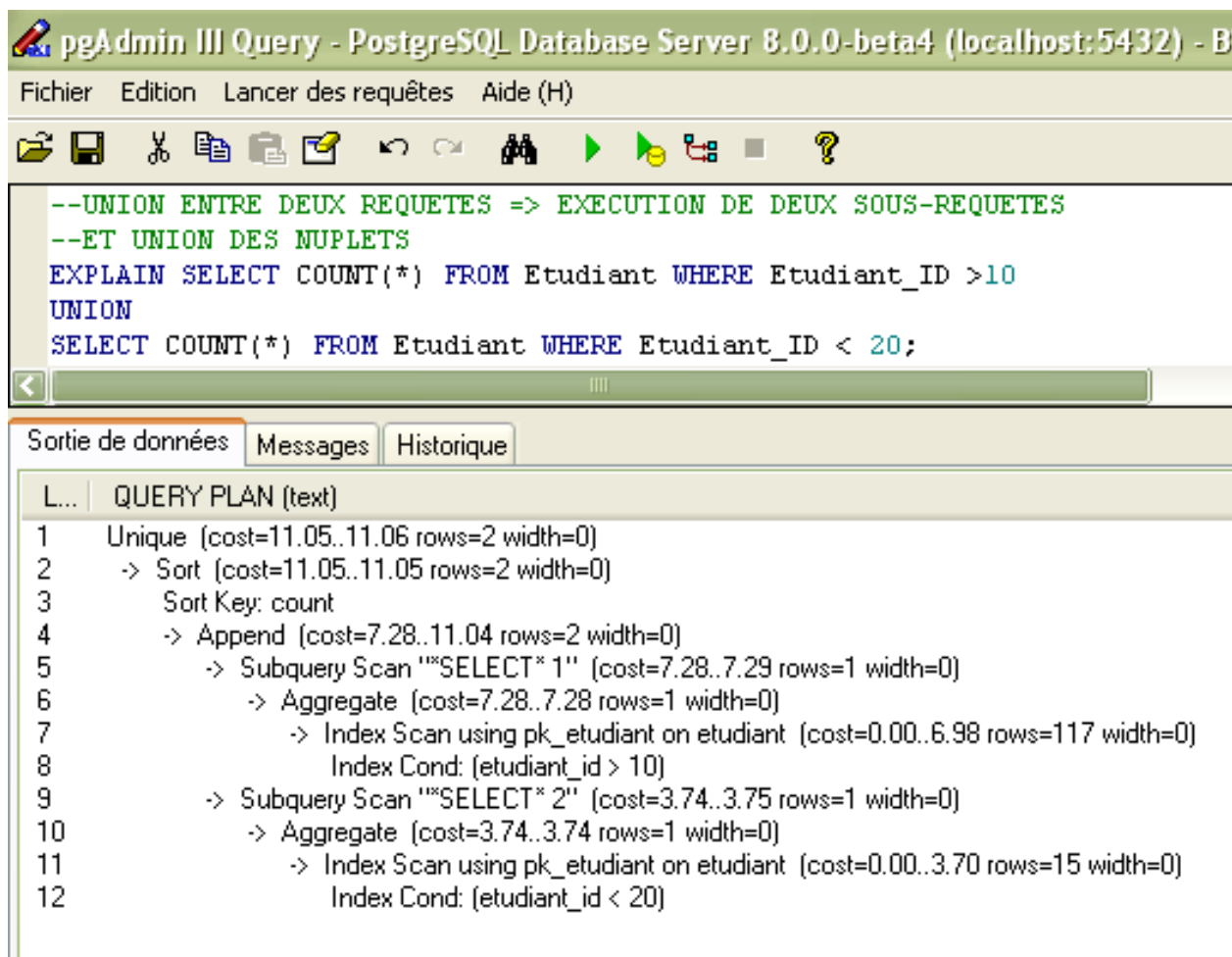
```
1  Hash Join (cost=1000000004.33..1000000004.49 rows=5 width=82)
2  Hash Cond: ("outer".enseignant_id = "inner".enseignant_id)
3  -> HashAggregate (cost=1000000001.26..1000000001.26 rows=21 width=4)
4      -> Seq Scan on reservation (cost=1000000000.00..1000000001.21 row...
5  -> Hash (cost=3.05..3.05 rows=5 width=82)
6      -> Index Scan using pk_enseignant on enseignant (cost=0.00..3.05 r...
```

A red bracket on the right side of the query plan, spanning lines 3, 4, and 5, points to a red text annotation:

*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 indicates it's connected to a PostgreSQL Database Server 8.0.0-beta4 on localhost:5432. The menu bar includes 'Fichier', 'Edition', 'Lancer des requêtes', and 'Aide (H)'. The toolbar contains various icons for file operations and query 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 SQL editor, the 'Sortie de données' tab is active, displaying the 'QUERY PLAN (text)' for the executed query. The plan is as follows:

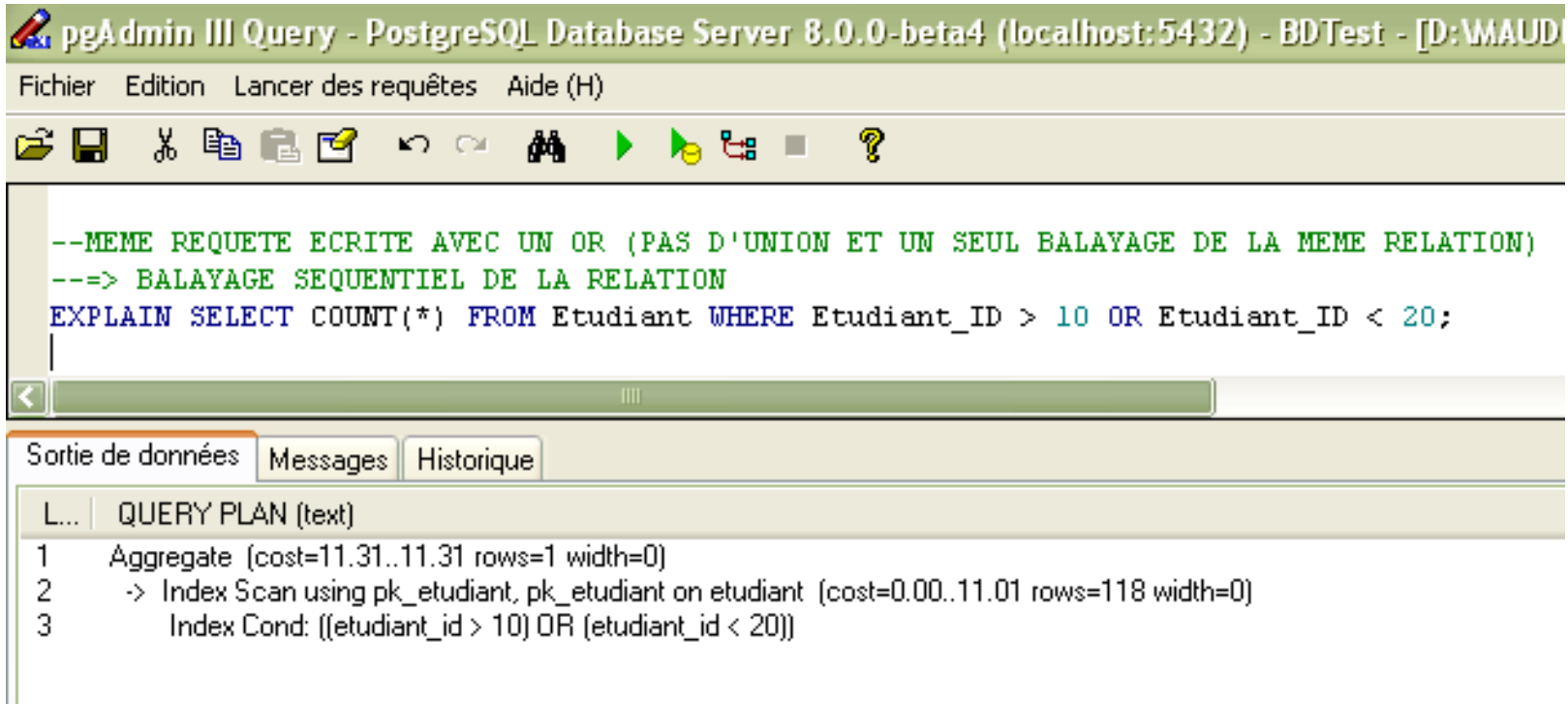
```

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 indicates the connection to '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 various 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;
```

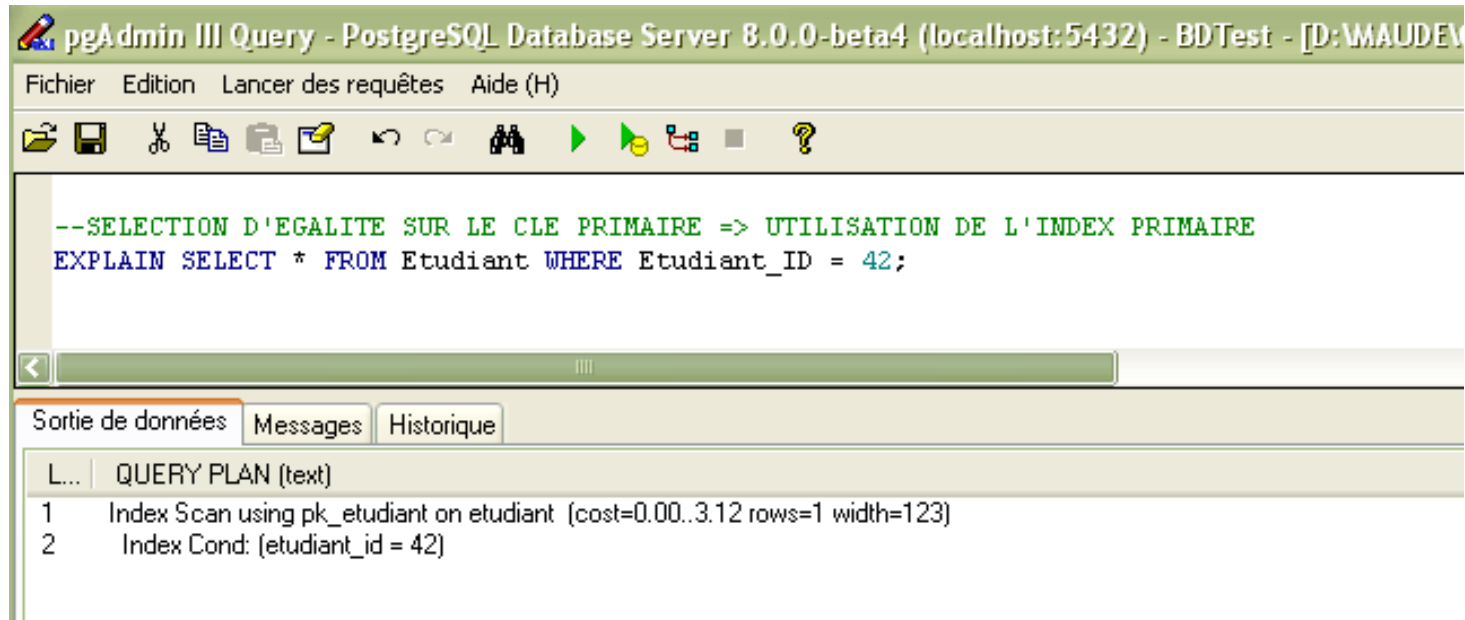
Below the query, the 'Sortie de données' tab is selected, displaying the 'QUERY PLAN (text)' for the executed query:

```
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



The screenshot shows the pgAdmin III Query tool interface. The title bar indicates it's connected to a PostgreSQL Database Server 8.0.0-beta4 on localhost:5432, using the BDTTest database and the user [D:VMAUDEV]. The menu bar includes Fichier, Edition, Lancer des requêtes, and Aide (H). The toolbar contains icons for file operations, query execution, and help. The query editor contains the following SQL code:

```
--SELECTION D'EGALITE SUR LE CLE PRIMAIRE => UTILISATION DE L'INDEX PRIMAIRE  
EXPLAIN SELECT * FROM Etudiant WHERE Etudiant_ID = 42;
```

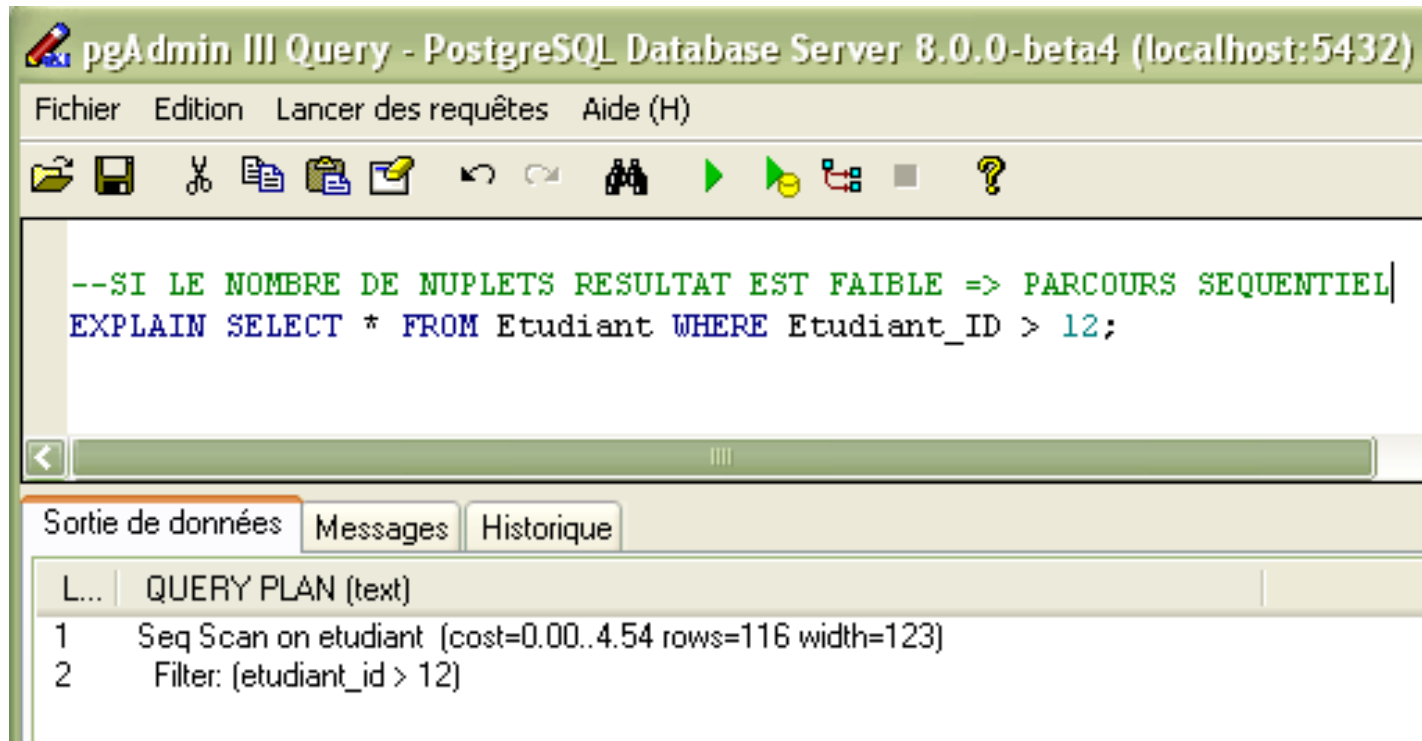
Below the query editor, there are tabs for 'Sortie de données', 'Messages', and 'Historique'. The 'Sortie de données' tab is active, showing the 'QUERY PLAN (text)' for the executed query. The query plan consists of two steps:

- 1 Index Scan using pk\_etudiant on etudiant (cost=0.00..3.12 rows=1 width=123)
- 2 Index Cond: (etudiant\_id = 42)

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

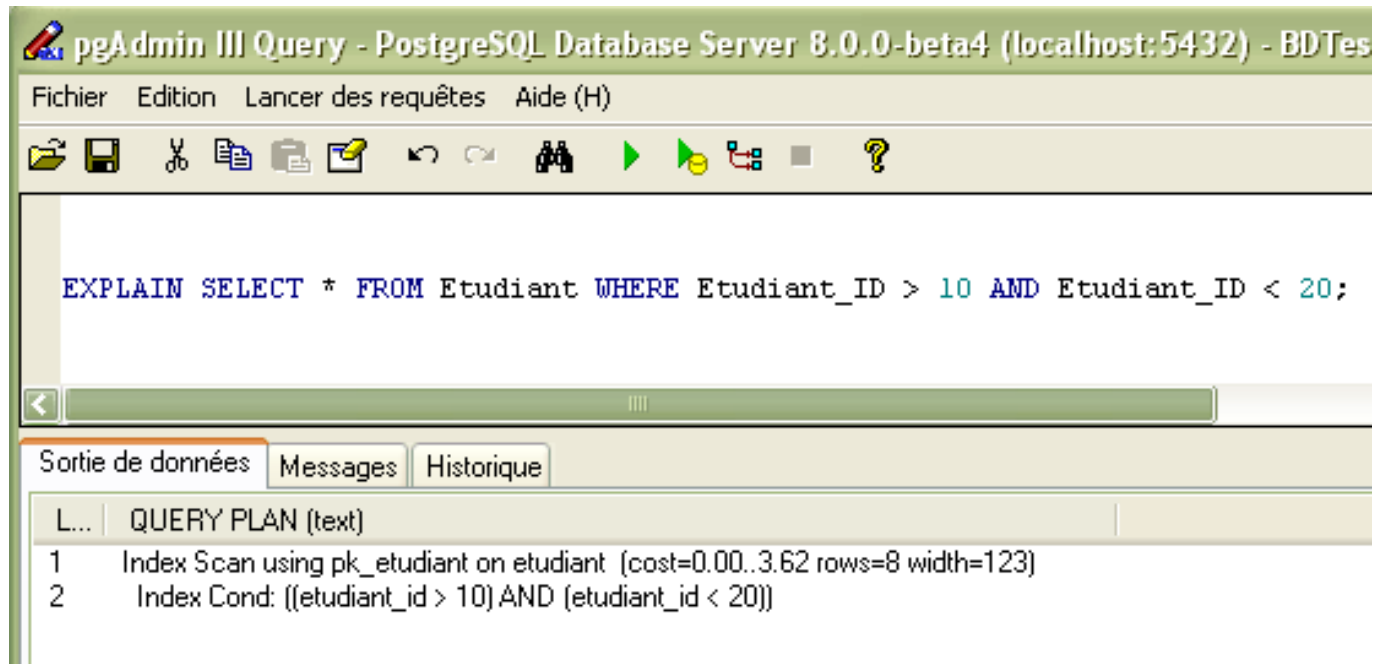


Temps d'extraction des données : 220ms



# 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) - BDTest". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, query execution, and help. The query editor contains the following SQL query:

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

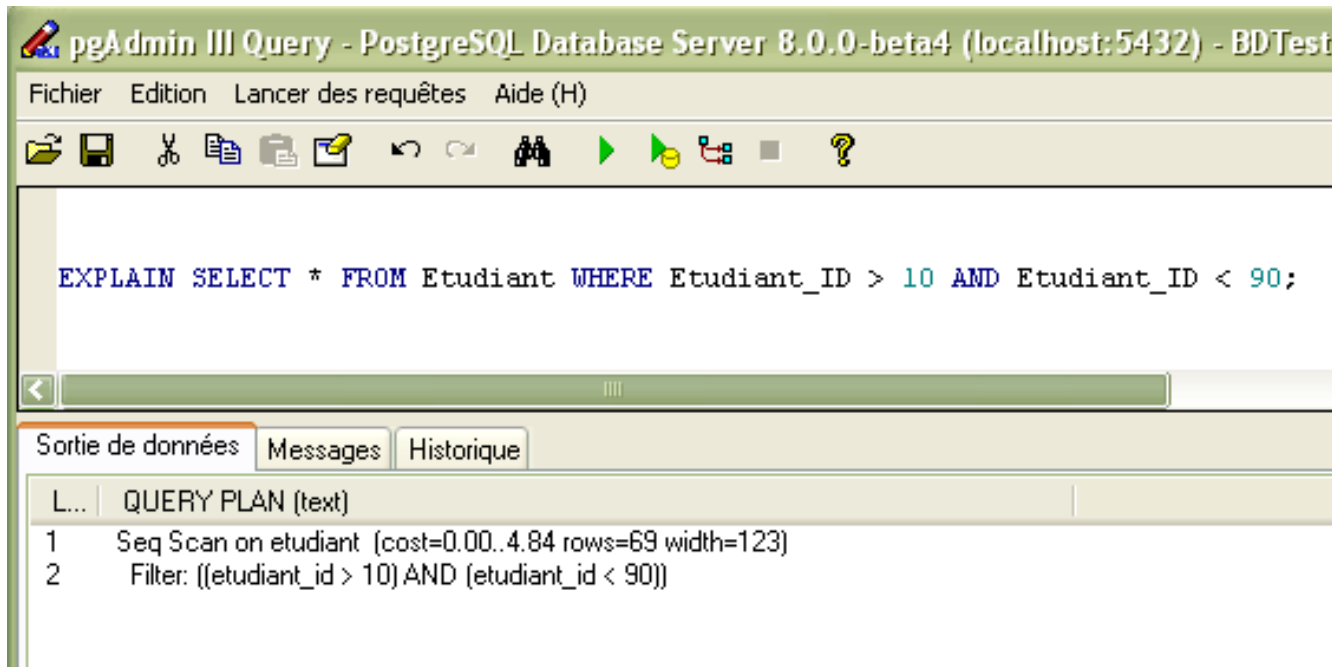
Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, showing the query plan:

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



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". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, query execution, and help. The query editor contains the following SQL query:

```
EXPLAIN SELECT * FROM Etudiant WHERE Etudiant_ID > 10 AND Etudiant_ID < 90;
```

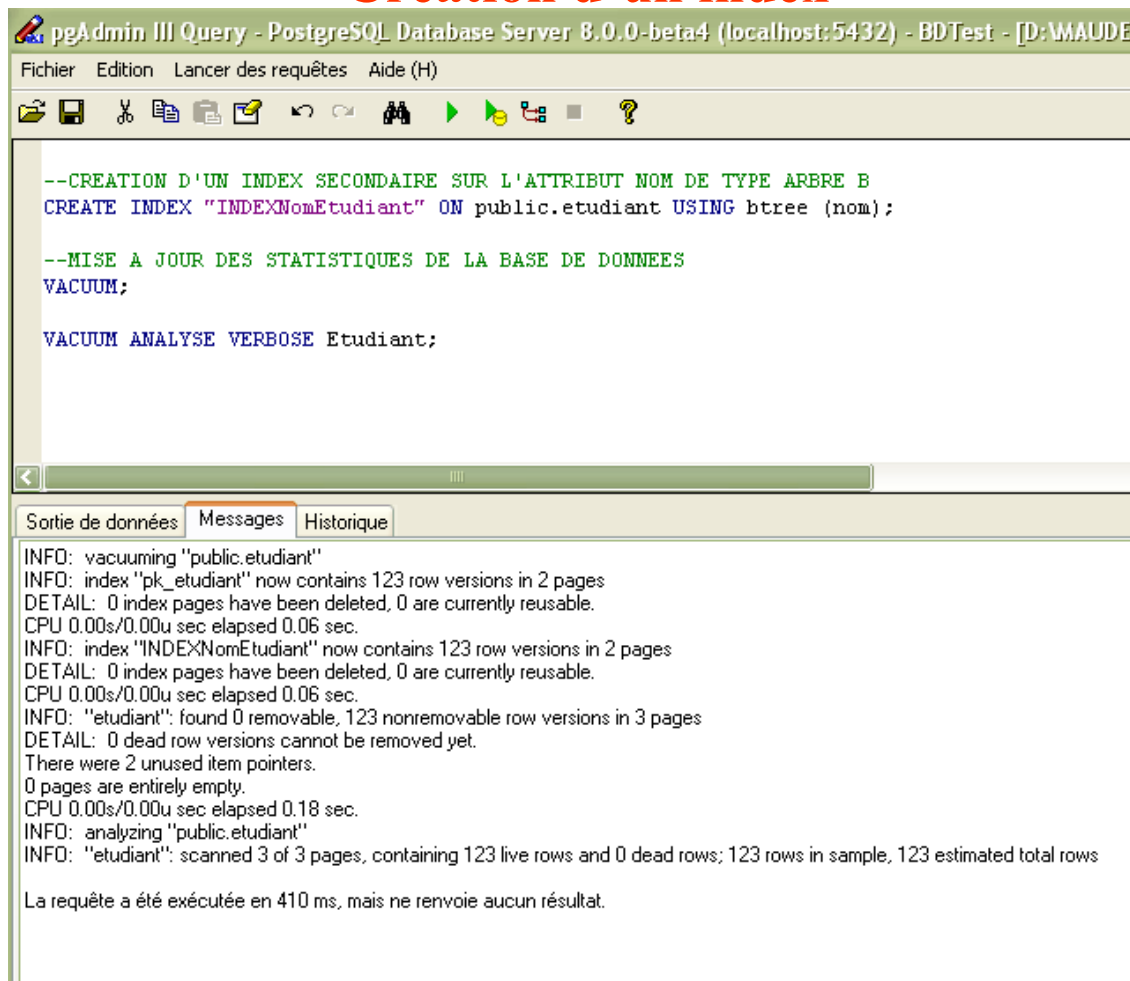
Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, showing the query plan:

L...	QUERY PLAN (text)
1	Seq Scan on etudiant (cost=0.00..4.84 rows=69 width=123)
2	Filter: ((etudiant_id > 10) AND (etudiant_id < 90))

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 it's connected to a PostgreSQL Database Server 8.0.0-beta4 on localhost:5432, using the BDTest database and the user MAUDE. The menu bar includes Fichier, Edition, Lancer des requêtes, and Aide (H). The toolbar contains icons for file operations, execution, and help. 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 ANALYZE VERBOSE Etudiant;
```

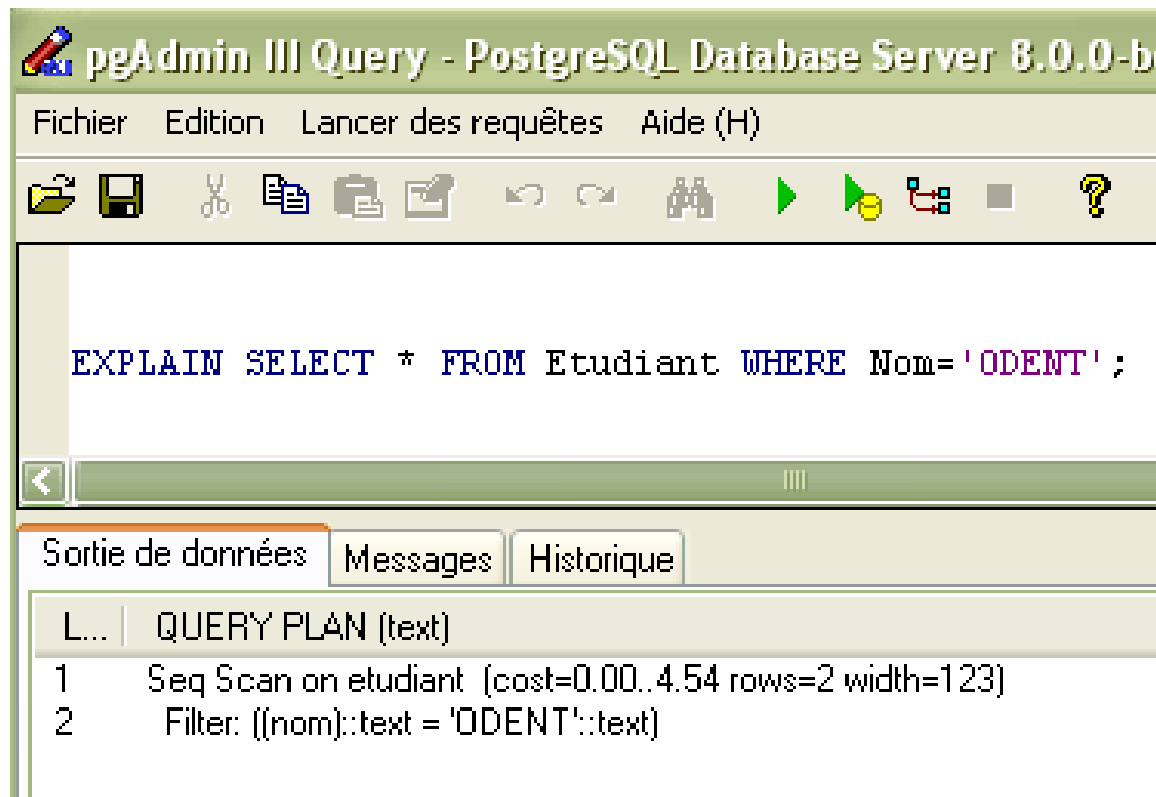
Below the query area, there are tabs for 'Sortie de données', 'Messages', and 'Historique'. The 'Messages' tab is selected, displaying the following output:

```
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
```

At the bottom, a status message reads: 'La requête a été exécutée en 410 ms, mais ne renvoie aucun résultat.'

# Exemples en utilisant PostgreSQL

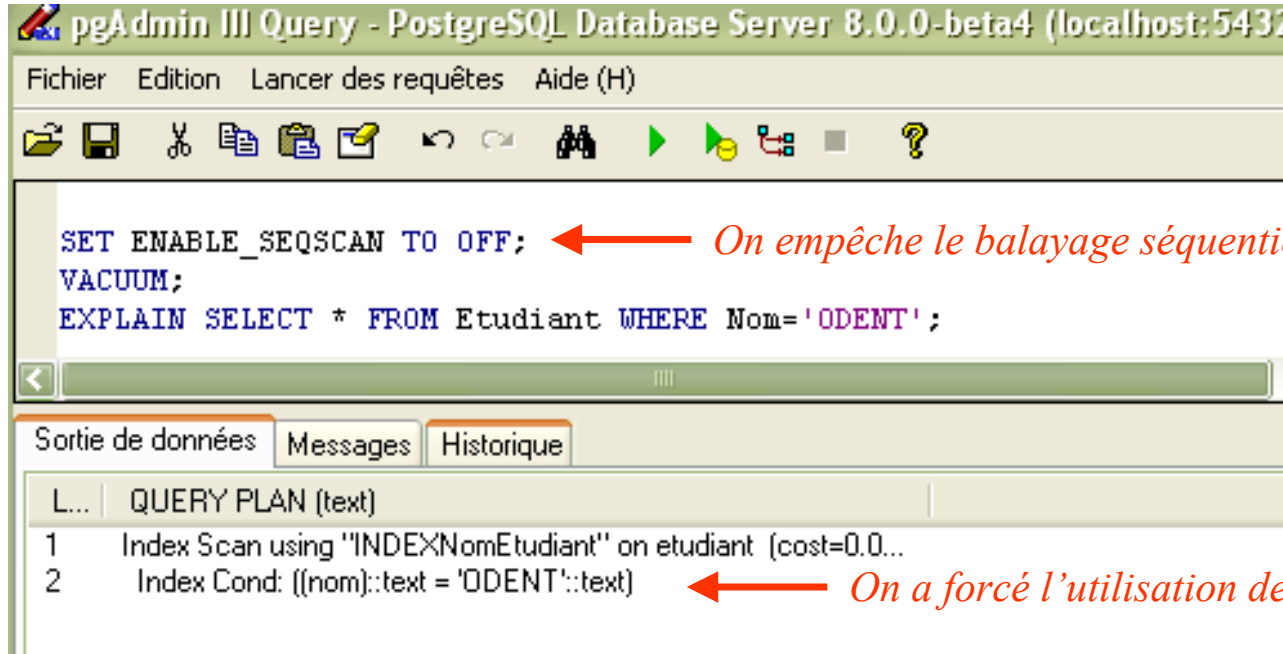
**Le SGBD peut choisir de ne pas utiliser les index**



**Temps d'extraction des données : 221ms**

# Exemples en utilisant PostgreSQL

**On peut forcer l'utilisation des index**



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 icons for file operations, editing, and execution. The SQL editor contains the following commands:

```
SET ENABLE_SEQSCAN TO OFF;
VACUUM;
EXPLAIN SELECT * FROM Etudiant WHERE Nom='ODENT';
```

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

```
1  Index Scan using "INDEXNomEtudiant" on etudiant (cost=0.0...
2  Index Cond: ((nom)::text = 'ODENT'::text)
```

Two red arrows point from the explanatory text to the query plan output. The first arrow points to the "SET ENABLE\_SEQSCAN TO OFF;" command, and the second arrow points to the "Index Cond: ((nom)::text = 'ODENT'::text)" line in the query plan.

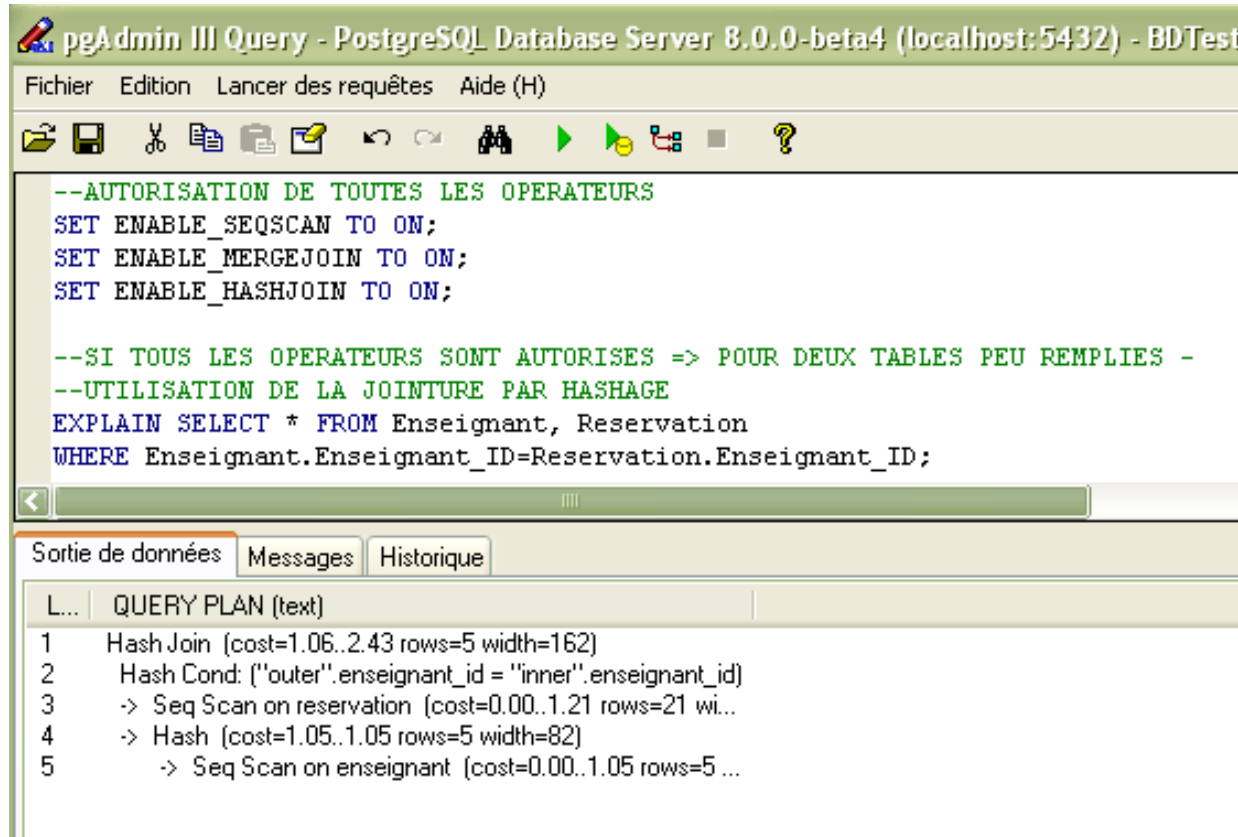
*On empêche le balayage séquentiel*

*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 it is connected 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, execution, and help. The SQL editor contains the following text:

```
--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;
```

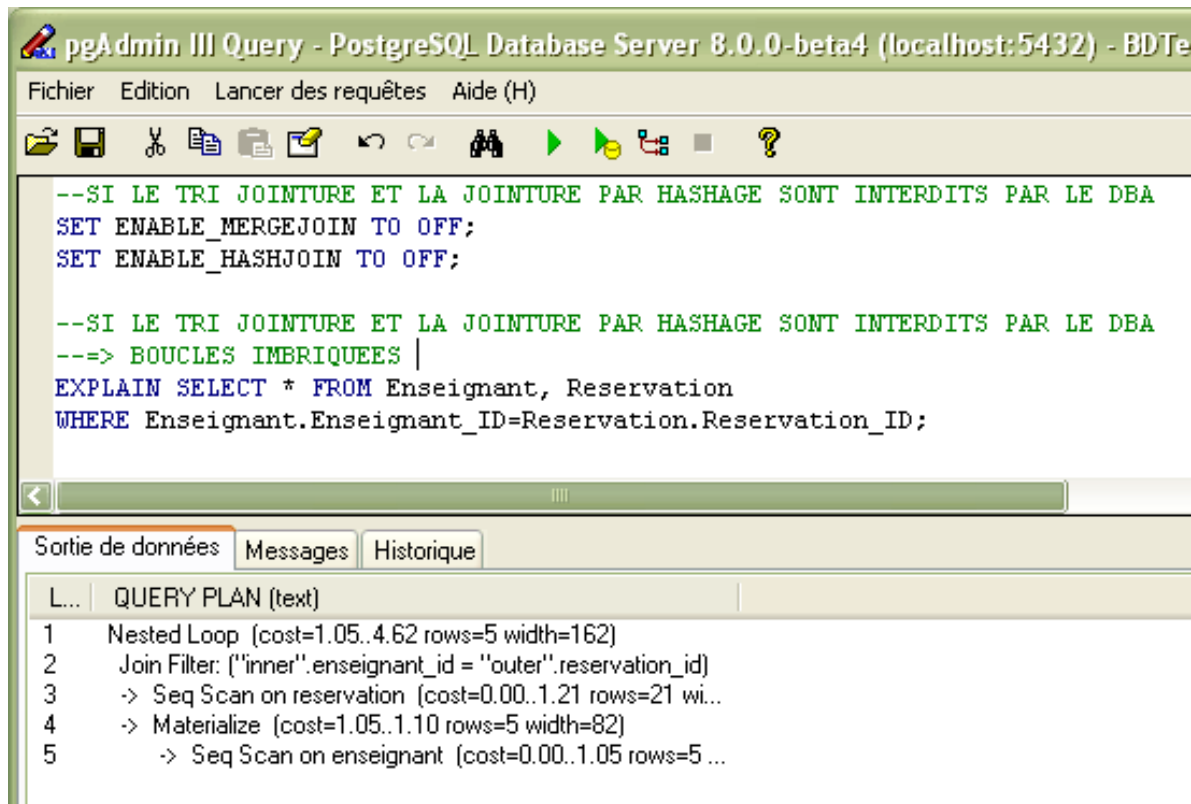
Below the editor, the 'Sortie de données' tab is active, displaying the 'QUERY PLAN (text)' for the executed query. The plan shows a Hash Join operation with a cost of 1.06..2.43, involving a Hash Cond and two Seq Scan operations on the Reservation and Enseignant tables.

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 SQL editor contains the following text:

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

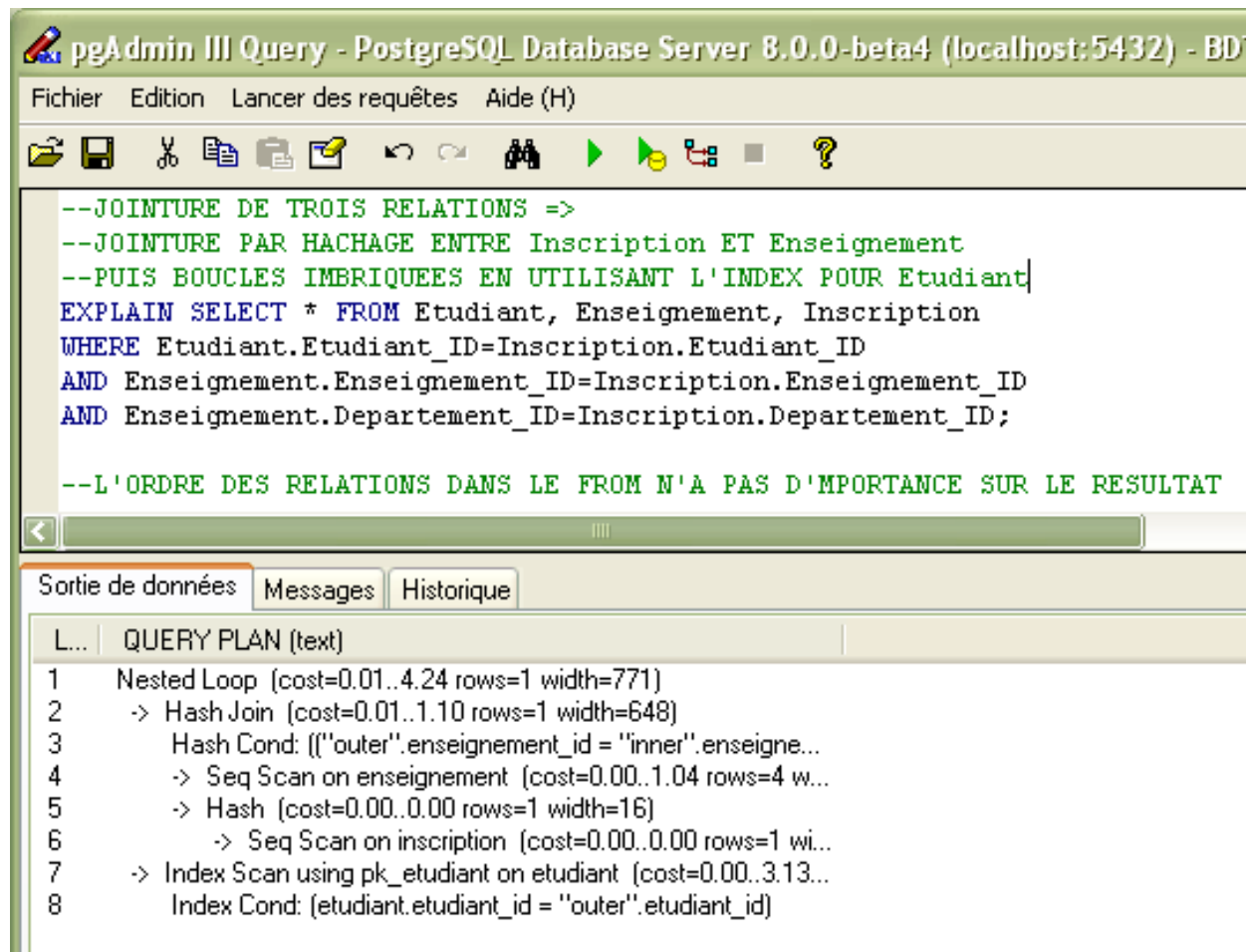
```
1  Nested Loop (cost=1.05..4.62 rows=5 width=162)
2    Join Filter: ("inner".enseignant_id = "outer".reservation_id)
3      -> Seq Scan on reservation (cost=0.00..1.21 rows=21 wi...
4      -> Materialize (cost=1.05..1.10 rows=5 width=82)
5        -> Seq Scan on enseignant (cost=0.00..1.05 rows=5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

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

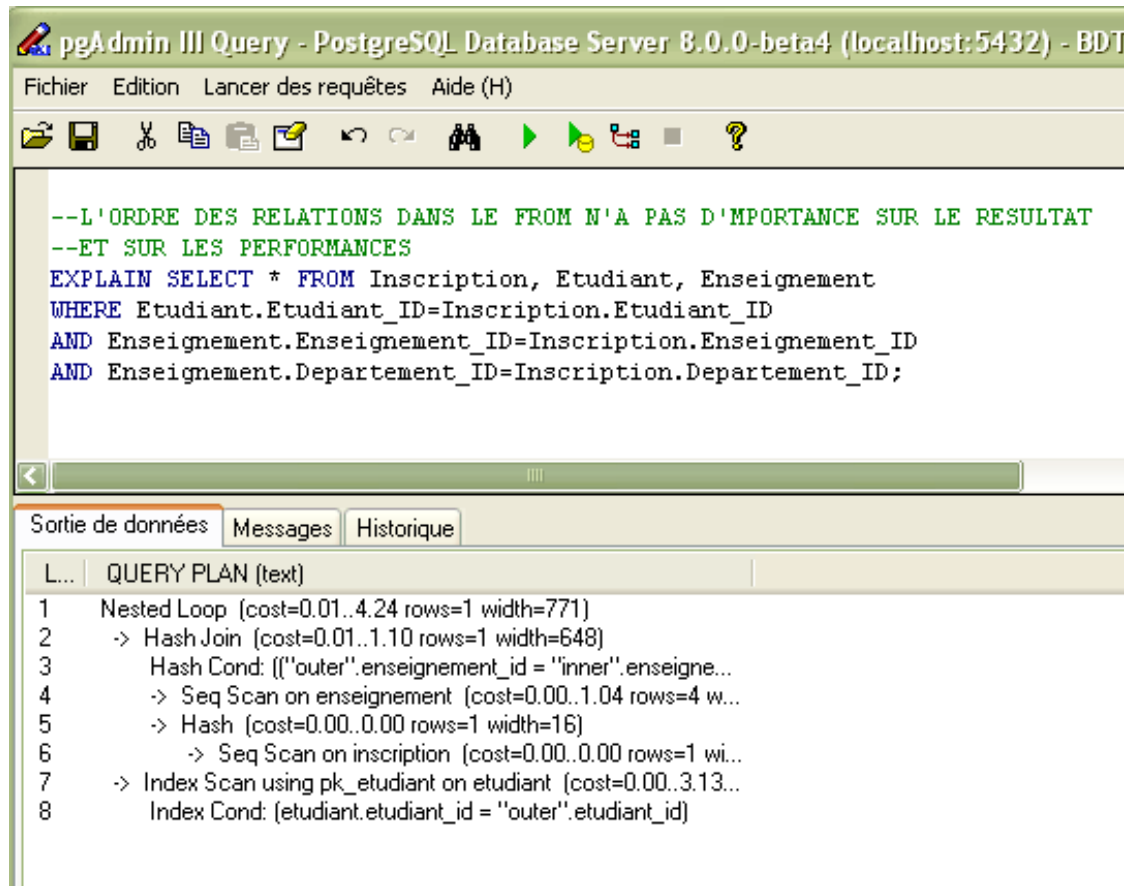
```
1  Nested Loop (cost=0.01..4.24 rows=1 width=771)
2    -> Hash Join (cost=0.01..1.10 rows=1 width=648)
3        Hash Cond: (["outer".enseignement_id = "inner".enseigne...
4        -> Seq Scan on enseignement (cost=0.00..1.04 rows=4 w...
5        -> Hash (cost=0.00..0.00 rows=1 width=16)
6       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



# 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 indicates it's connected to a 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, query execution, and help. 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 selected, 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**

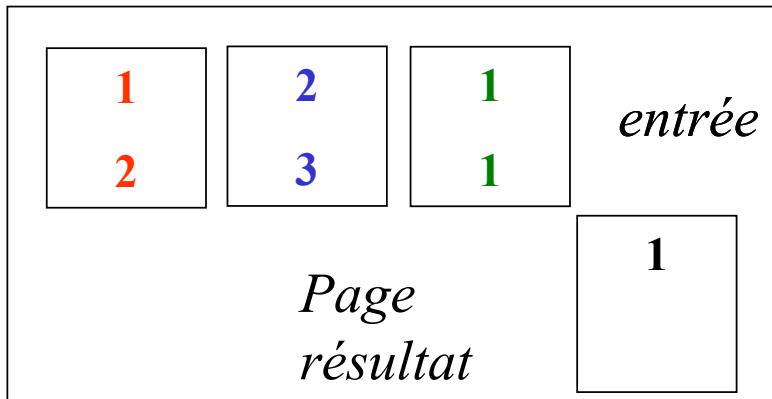
# Élimination des doublons par tri

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

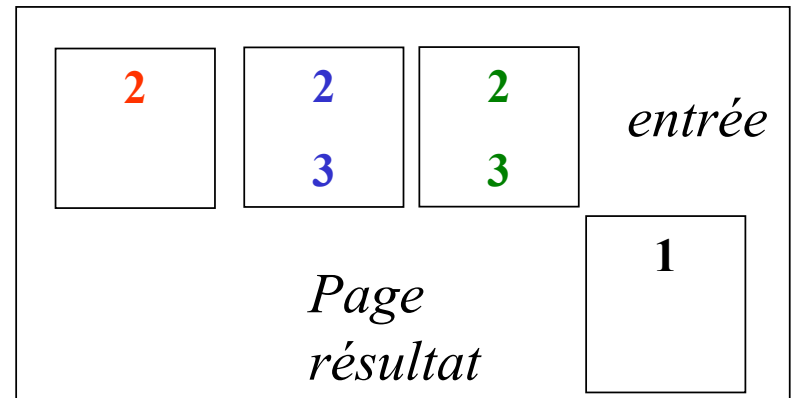
17 nuplets    2 nuplets / bloc  
4 pages en mémoire

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

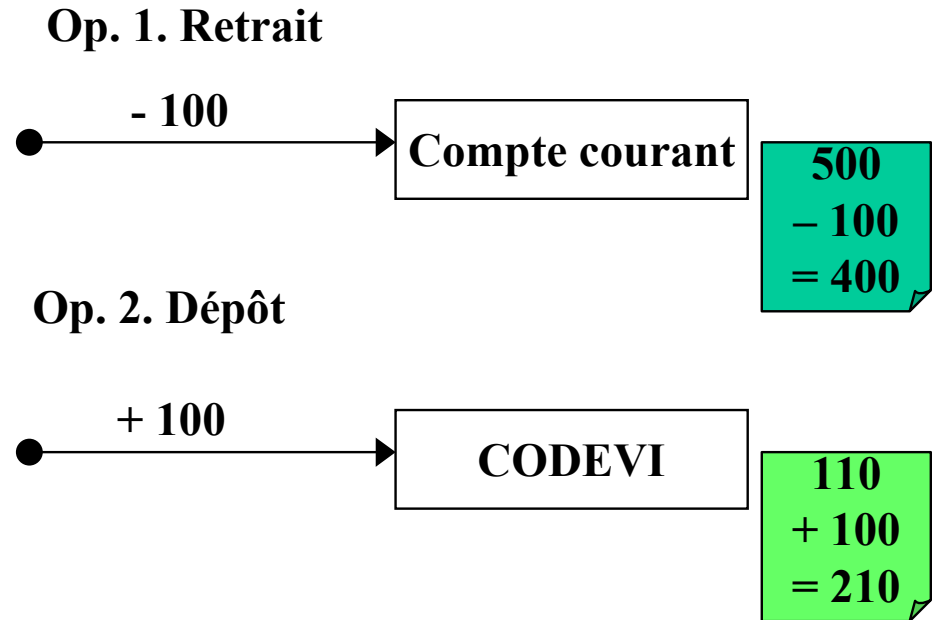
# **Exemple de transaction**

# Exemple de transaction

Virement =  
2 opérations atomiques

# Exemple de transaction

Virement =  
2 opérations atomiques

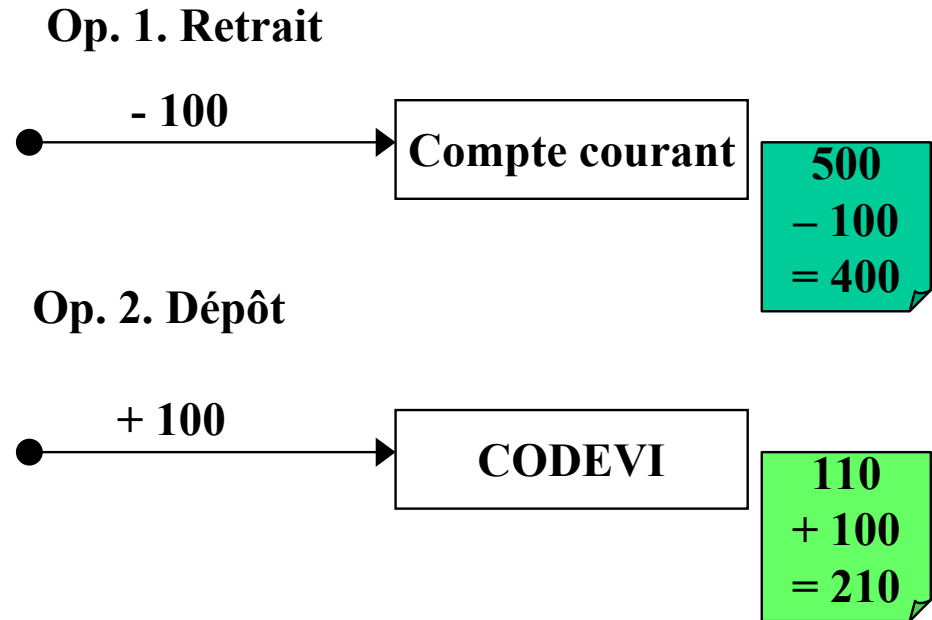




# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques

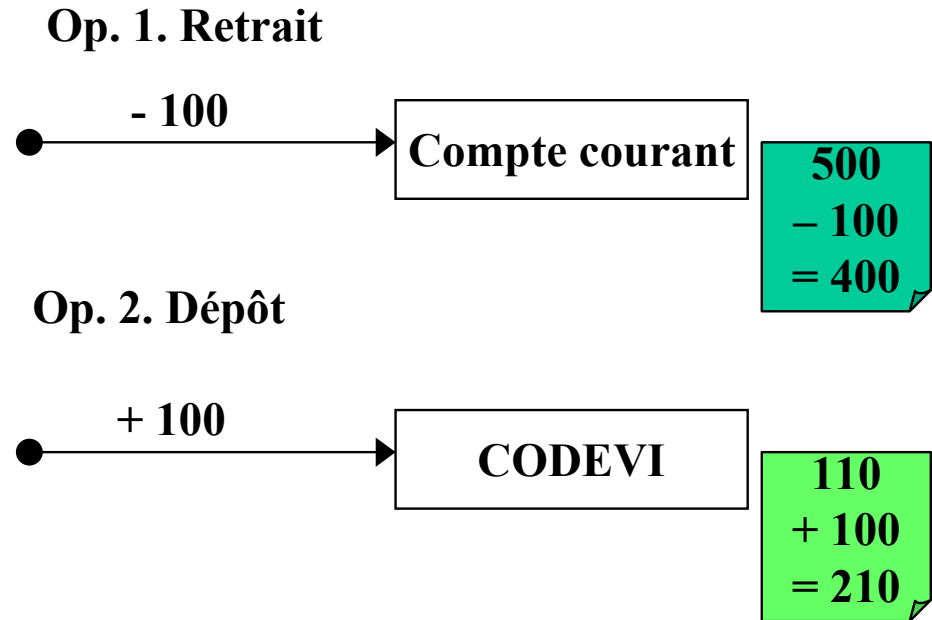


# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques

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

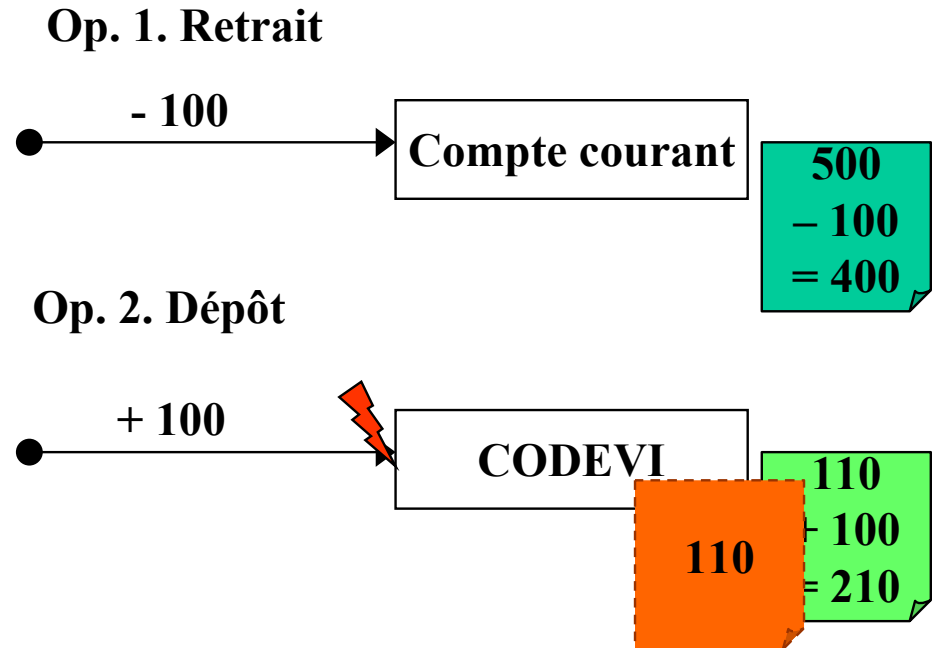


# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques

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



# Exemple de transaction

## Virement bancaire **sans transaction**

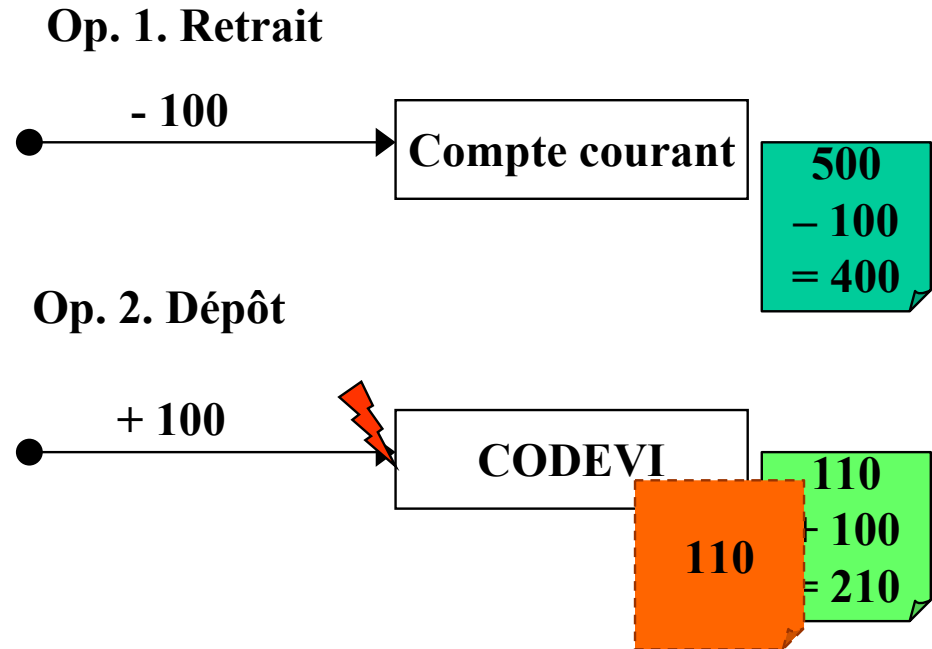
Virement =  
2 opérations atomiques

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

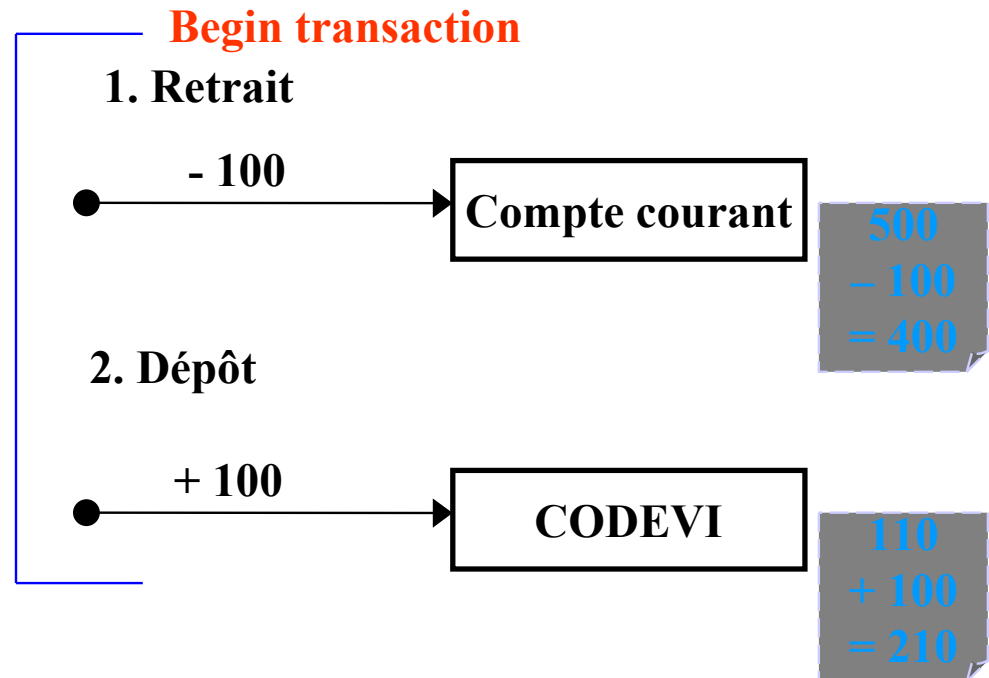
**Virement bancaire dans une transaction (1/2)**

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

# Exemple de transaction

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

Virement = 1 transaction  
de 2 opérations  
atomiques

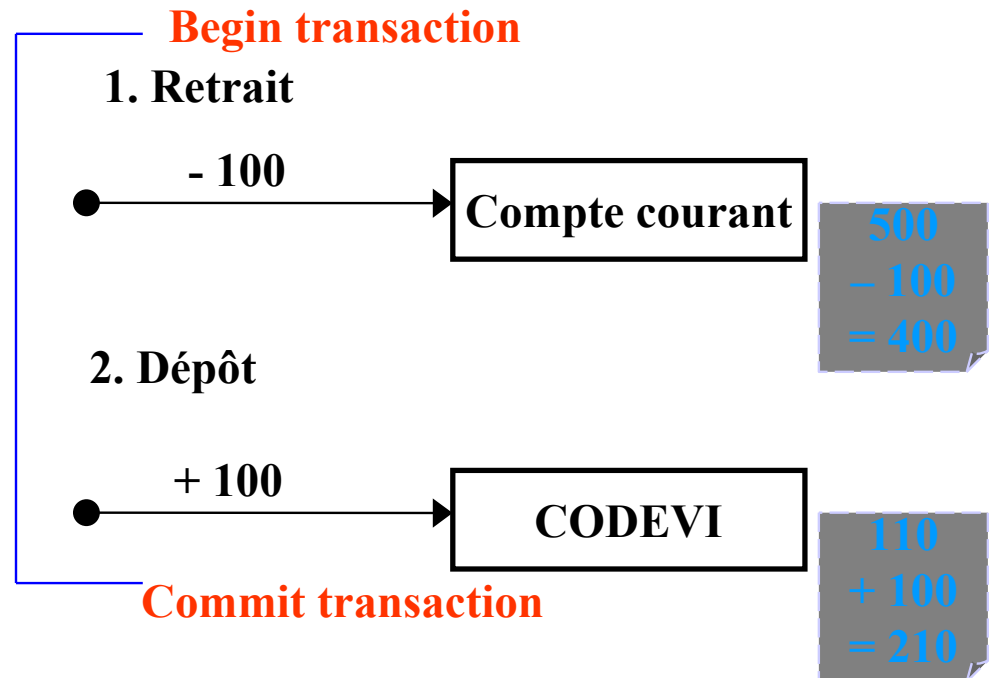




# Exemple de transaction

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

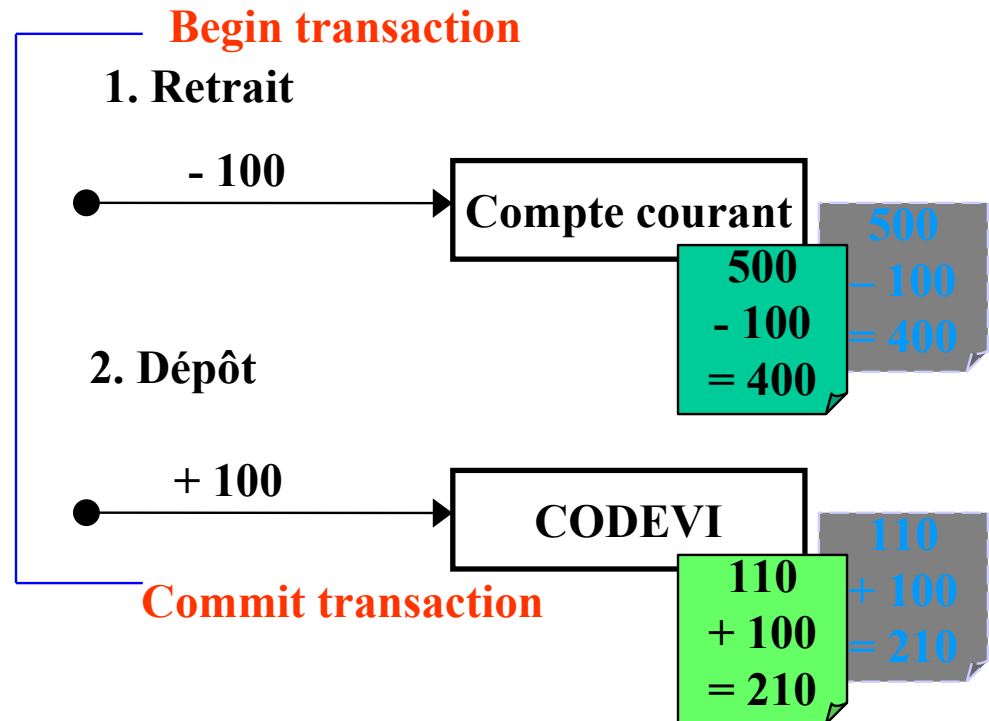
Virement = 1 transaction  
de 2 opérations  
atomiques



# Exemple de transaction

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

Virement = 1 transaction  
de 2 opérations  
atomiques



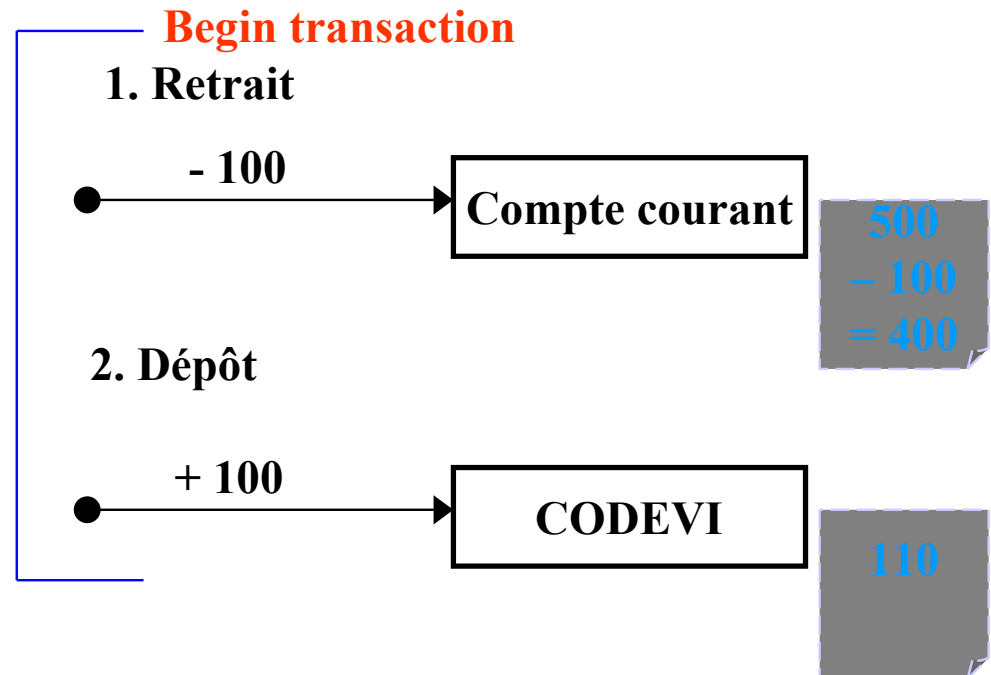
# **Exemple de transaction**

# Exemple de transaction

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

# Exemple de transaction

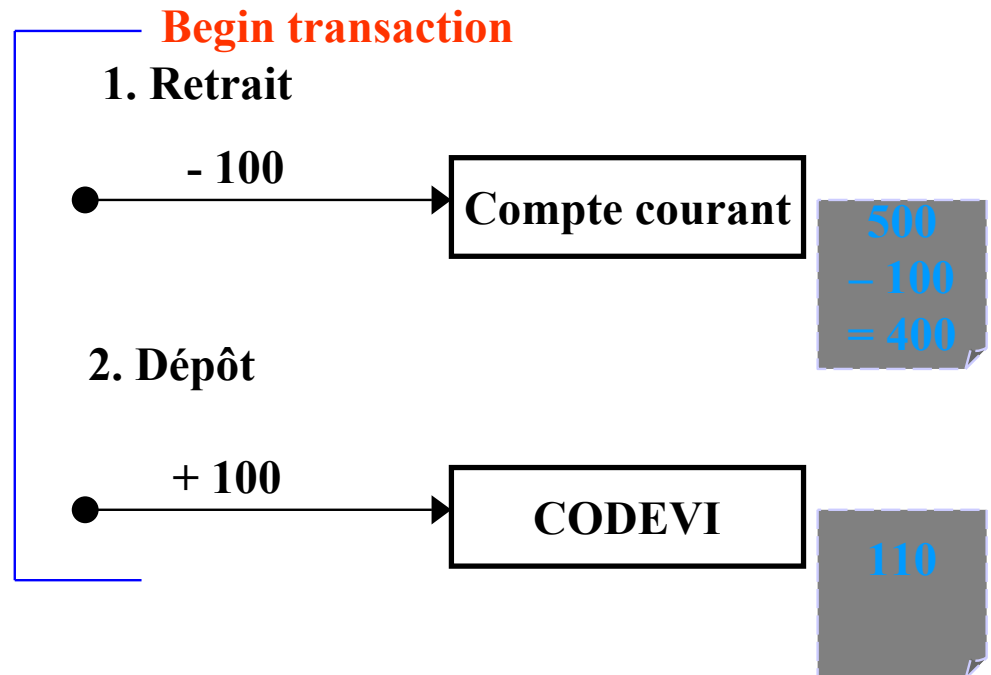
## Virement bancaire **dans une transaction** (2/2)



# Exemple de transaction

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

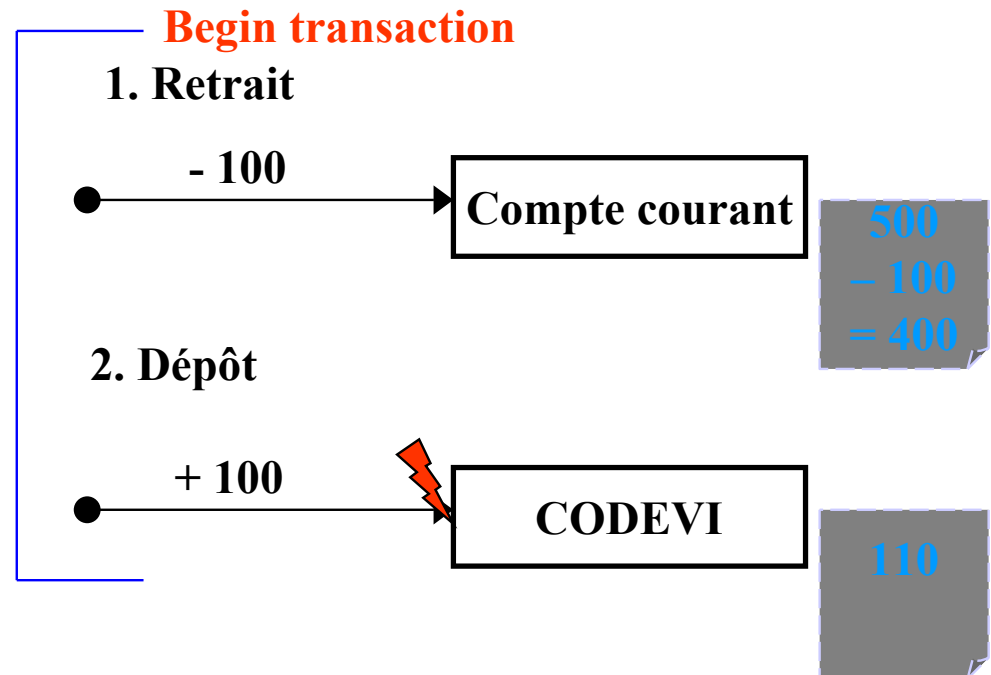
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

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

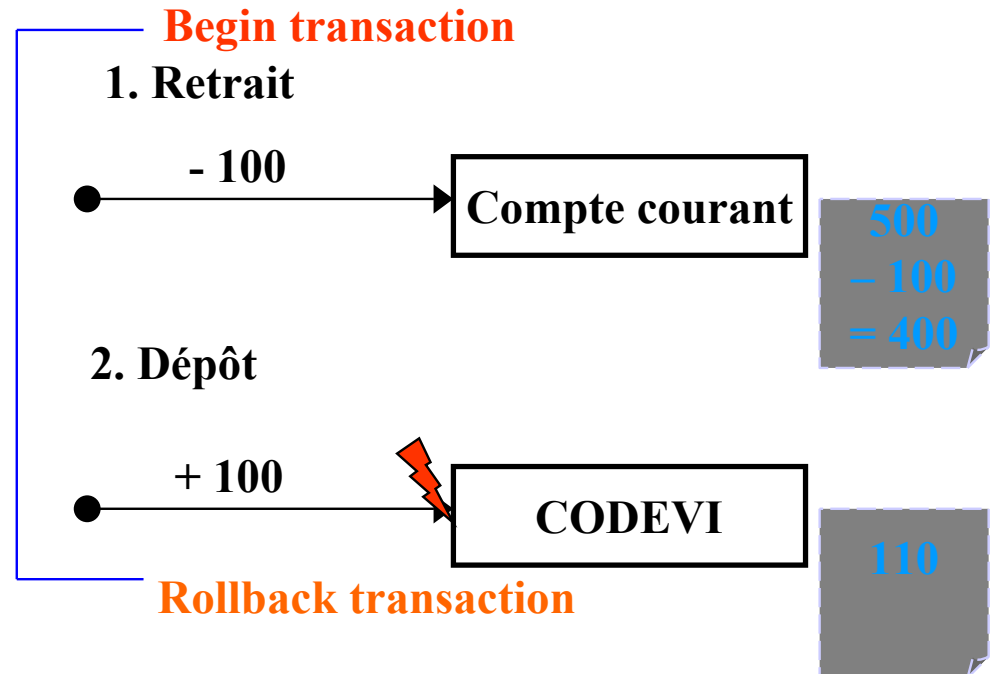
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

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

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

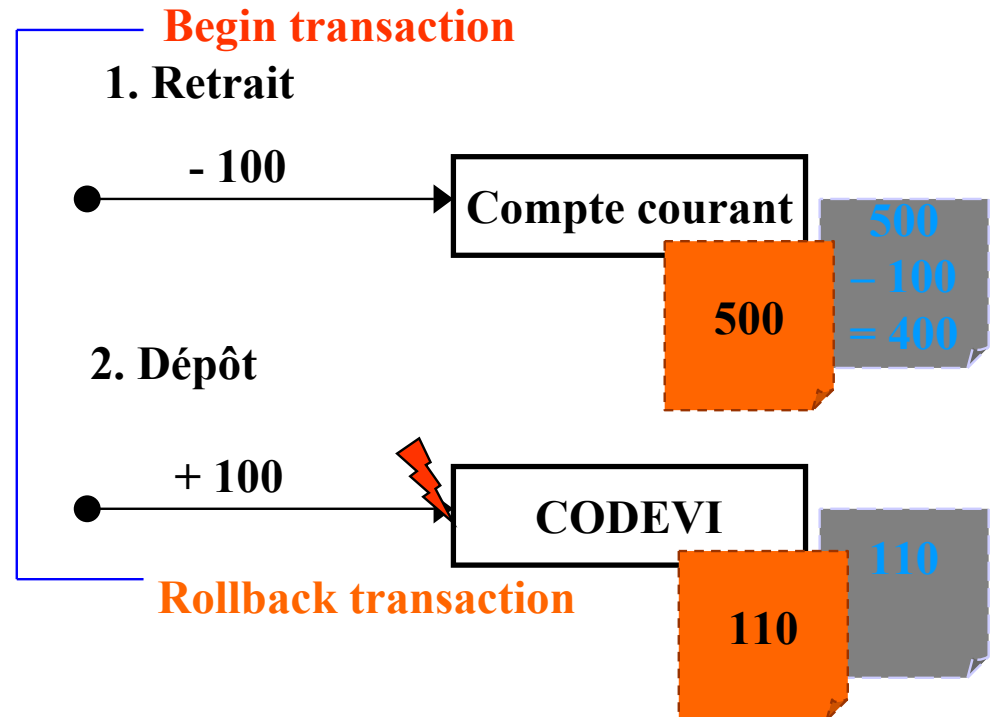




# Exemple de transaction

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

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



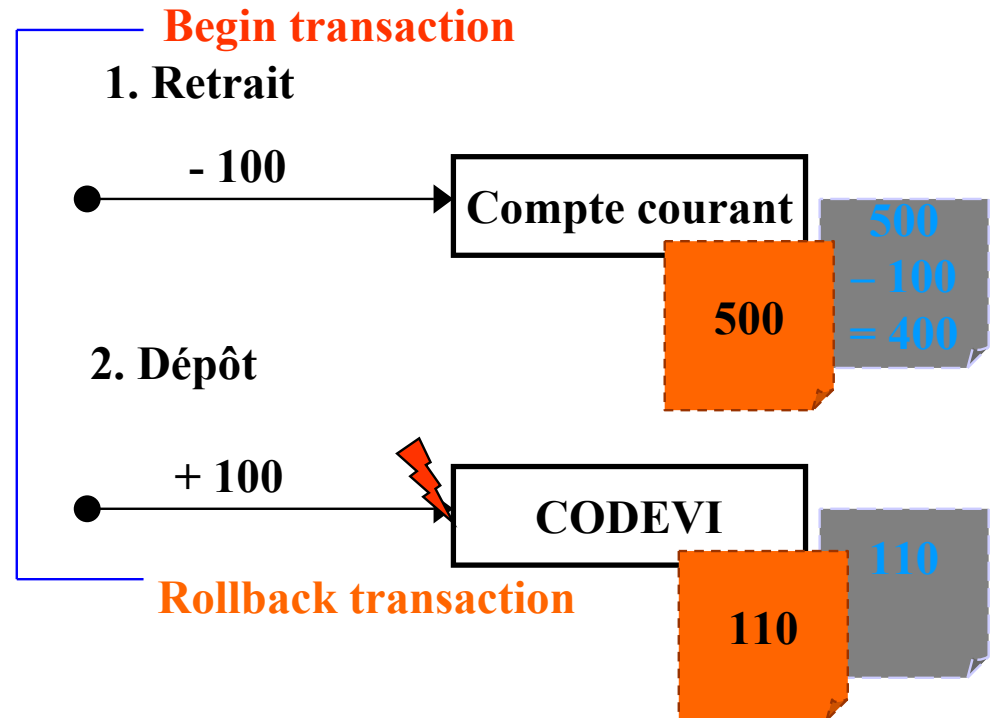
# Exemple de transaction

## 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

# Ordonnancement

- **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  $T$

- **Ordonnancement 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'ordonnancement

- **Ordonnancement 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  $T_1 T_2 T_3$

# Conflits

- **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**

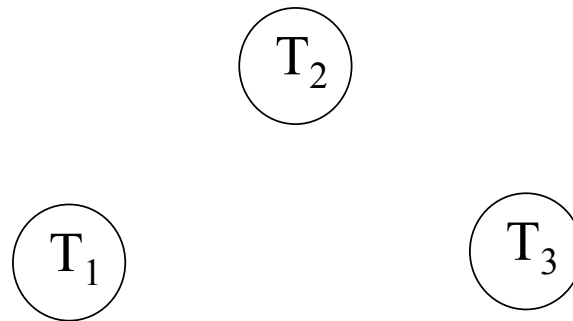
Isolation totale - Mise en attente des transactions en conflit

# 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  $T_j$**

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



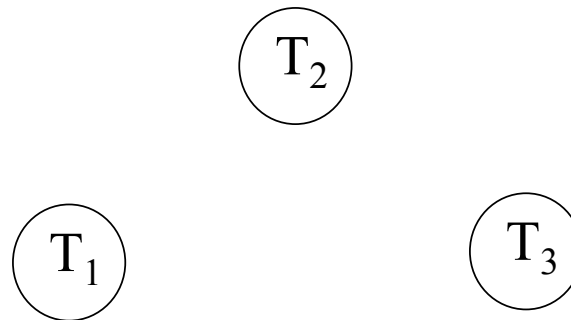
**Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable**

# 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  $T_j$

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



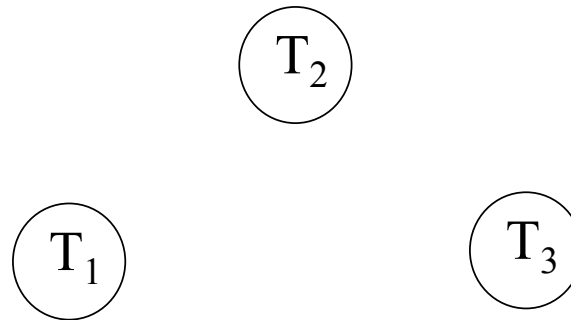
**Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable**

# 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  $T_j$

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



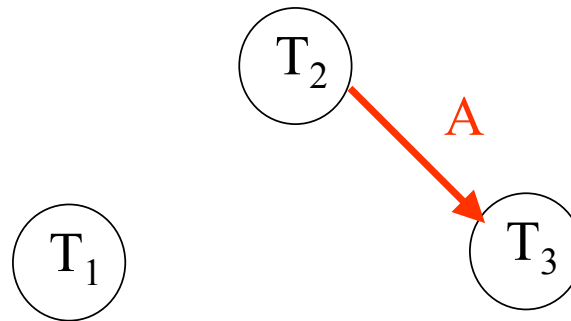
**Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable**

# 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  $T_j$

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



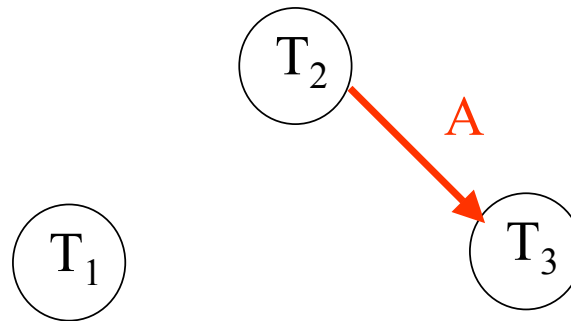
**Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable**

# 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  $T_j$

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



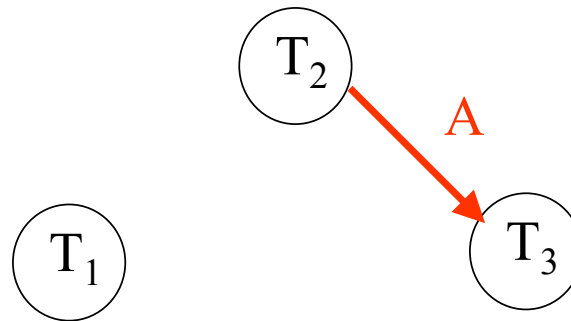
**Si le graphe est sans cycle (tri topologique) alors l'ordonnancement est sérialisable**

# 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**

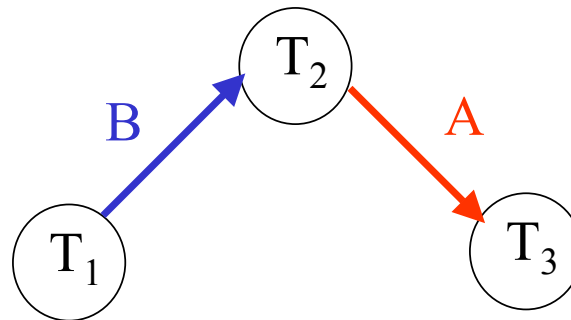


# Graphe de précédence

## 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  $T_j$

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



**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<sup>er</sup> module
- Transmission des transactions au 2<sup>ème</sup> module après insertion des verrous
- Détection des demandes de verrous par le 2<sup>ème</sup> 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<sup>er</sup> module informé par le 2<sup>ème</sup>
- Après libération d'un verrou, transmission du verrou à une transaction en attente par le 2<sup>ème</sup> module

# Inter-blocage



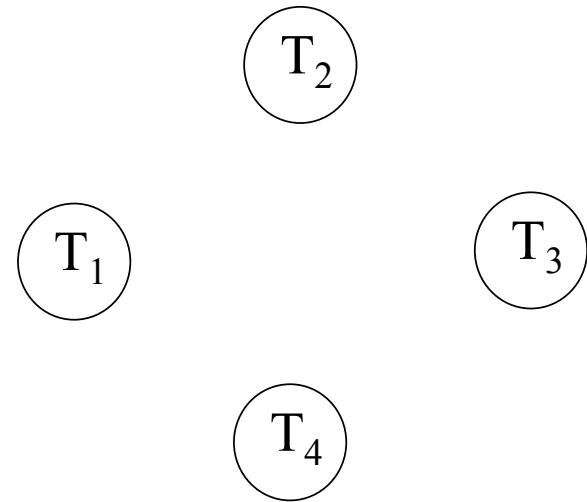
- **Inter-blocage (*Deadlock*)** : Attente mutuelle de deux transactions
- **Détection par un graphe d'attente**
  - ◆ Chaque nœud représente une transaction en cours d'exécution
  - ◆ Un arc de  $T_i$  vers  $T_j$  signifie que  $T_i$  **attend** un verrou détenu par  $T_j$  sur un même item
  - ◆ Ajout 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



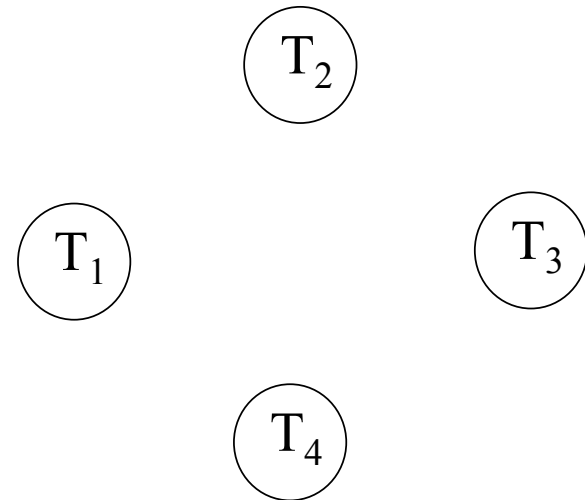
# Graphe d'attente



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

Table des verrous

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



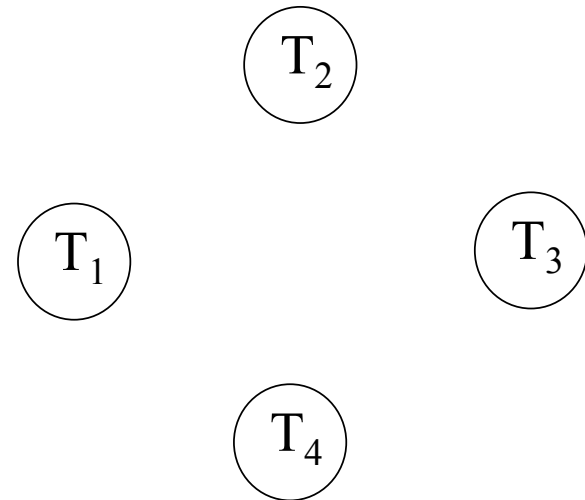
# Graphe d'attente



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

Table des verrous

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



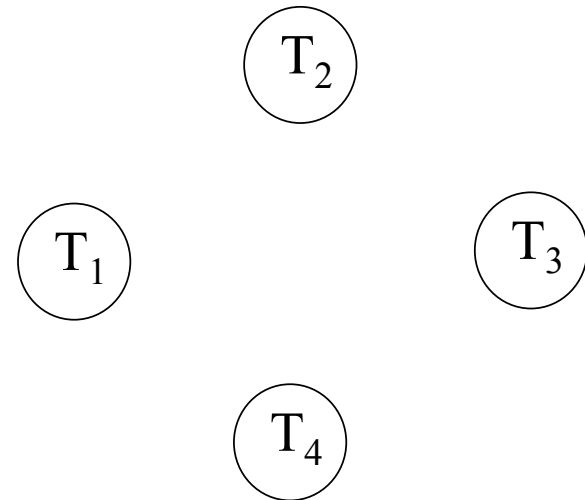
# Graphe d'attente



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

Table des verrous

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





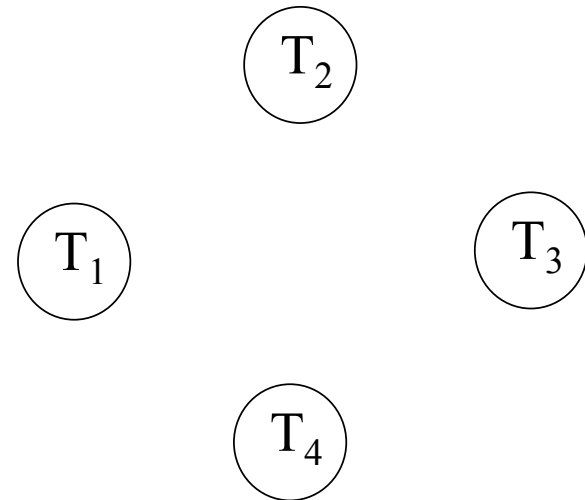
# Graphe d'attente



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

Table des verrous

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



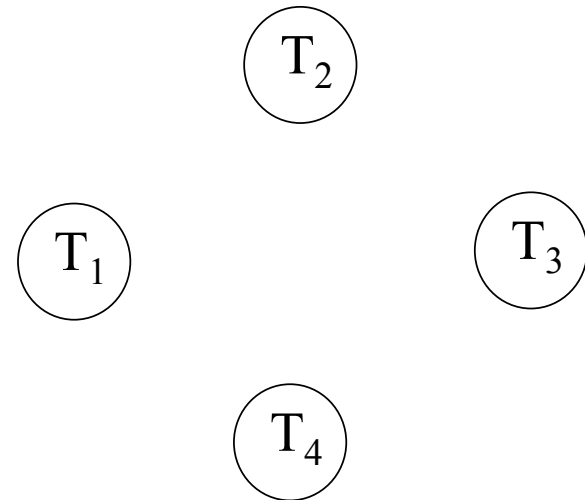
# Graphe d'attente



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

Table des verrous

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



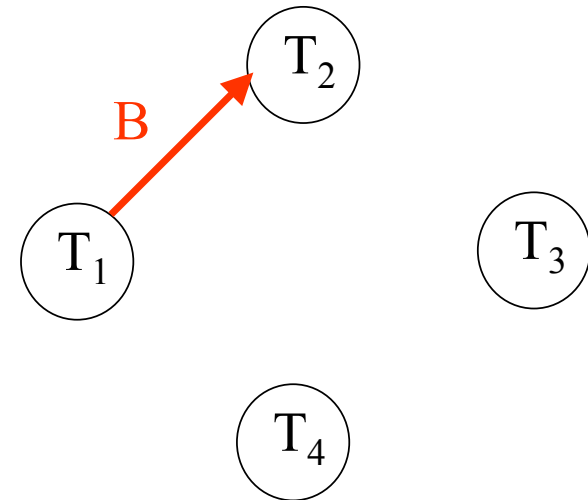
# Graphe d'attente



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

Table des verrous

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



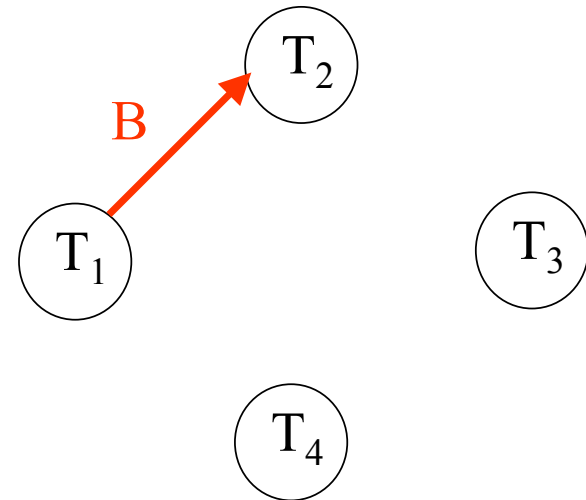
# Graphe d'attente



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

Table des verrous

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



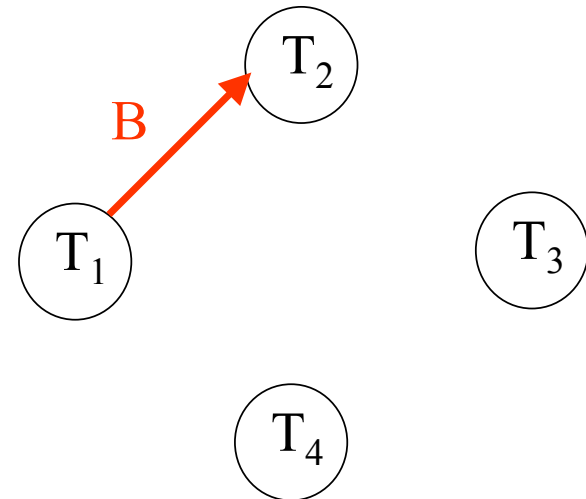
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	



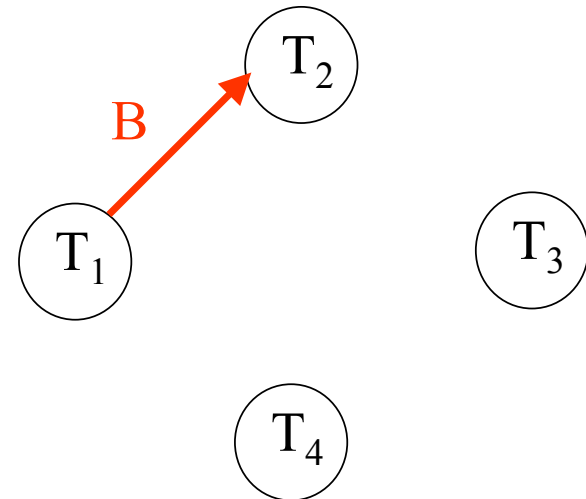
# Graphe d'attente



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

Table des verrous

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



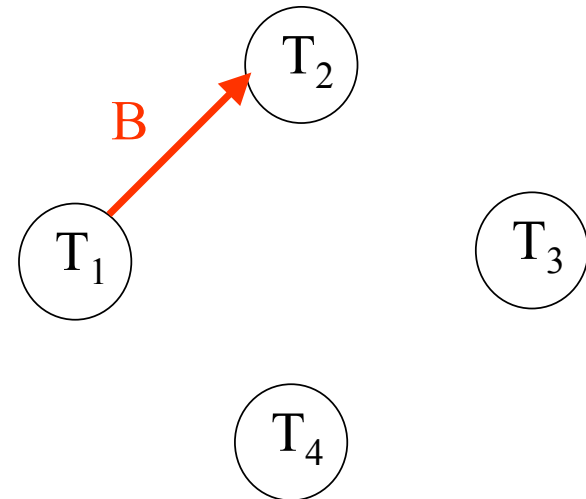
# Graphe d'attente



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

Table des verrous

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



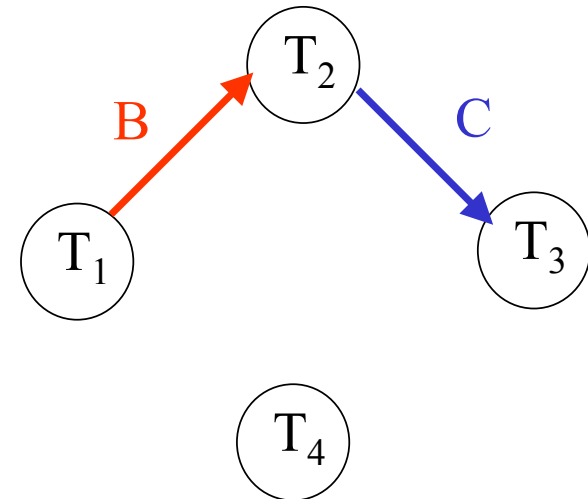
# Graphe d'attente



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

Table des verrous

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





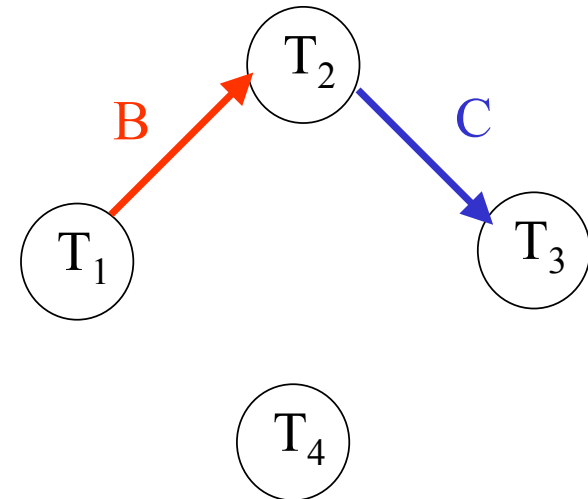
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



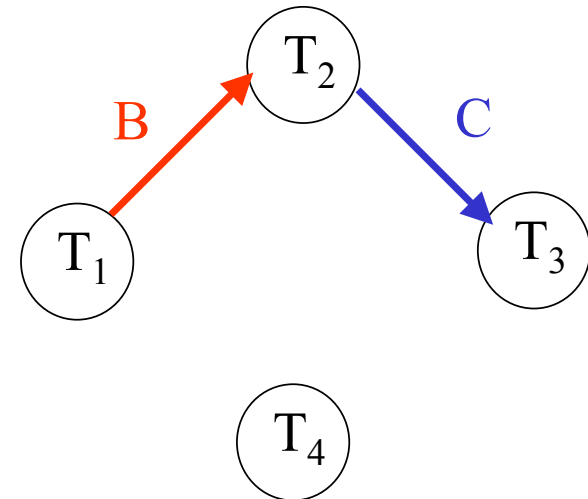
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



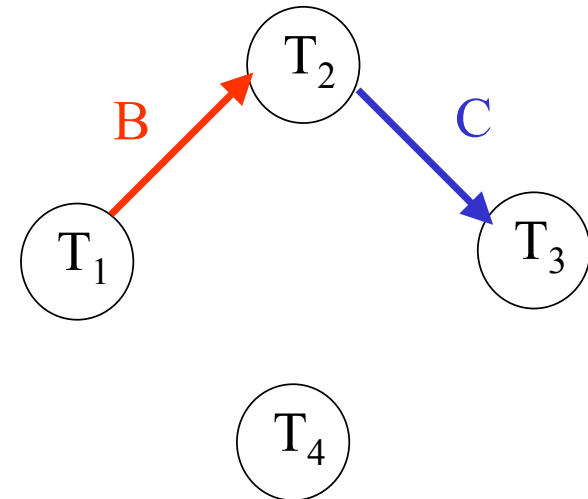
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



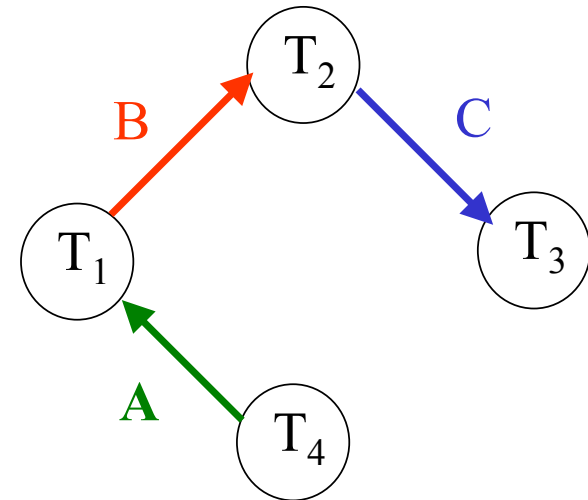
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



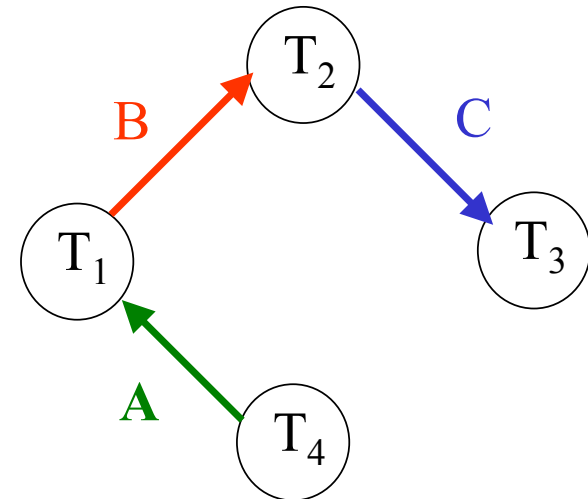
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



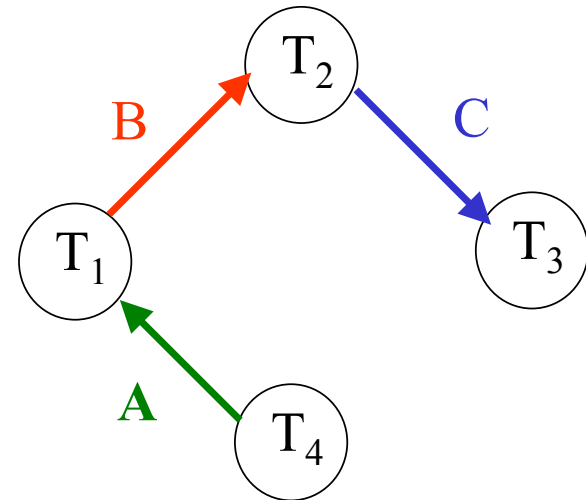
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)



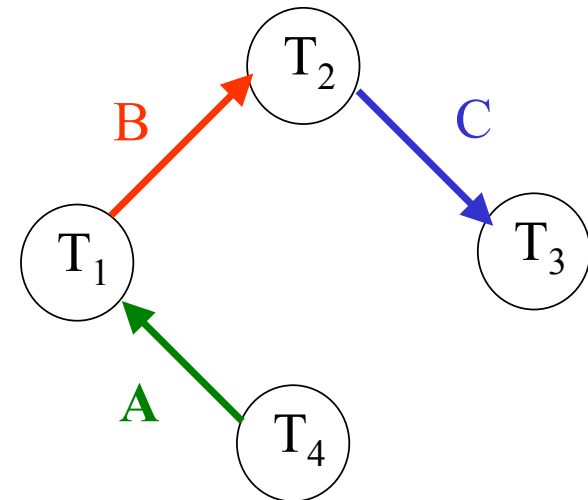
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)	T <sub>3</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)	
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)	



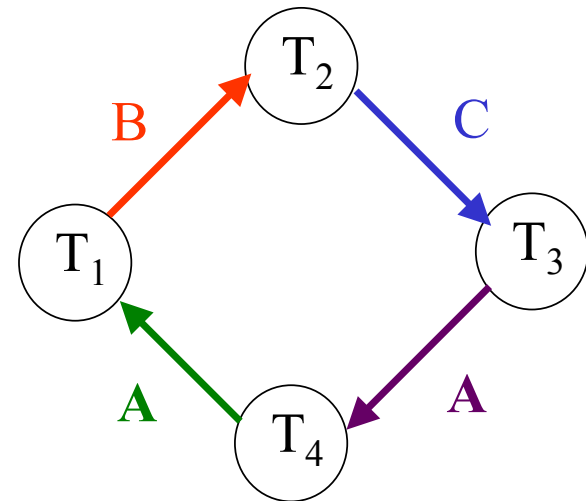
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)	T <sub>3</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)	
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)	





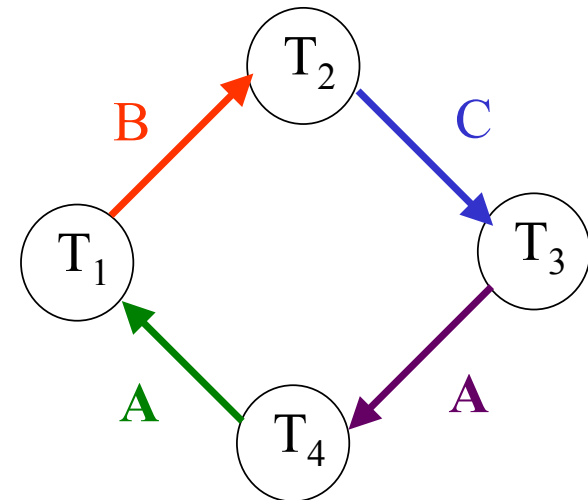
# Graphe d'attente



VP<sub>1</sub>(A), R<sub>1</sub>(A), VX<sub>2</sub>(B), W<sub>2</sub>(B), VP<sub>1</sub>(B), VP<sub>3</sub>(C), R<sub>3</sub>(C), VX<sub>2</sub>(C), VX<sub>4</sub>(A), VX<sub>3</sub>(A)

Table des verrous

A	VP (T <sub>1</sub> )	T <sub>4</sub> (VX)	T <sub>3</sub> (VX)
B	VX (T <sub>2</sub> )	T <sub>1</sub> (VP)	
C	VP (T <sub>3</sub> )	T <sub>2</sub> (VX)	



Cycle dans le graphe d'attente ⇒  
 détection d'un inter-blocage ⇒  
 Annulation de T<sub>3</sub>

# 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 précédence :

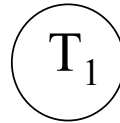
Protocole V2P :

# 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 :



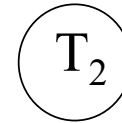
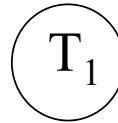
Protocole V2P :

# 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 :



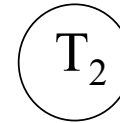
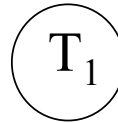
Protocole V2P :

# 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 :



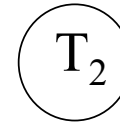
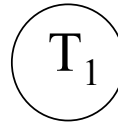
Protocole V2P :

# Protocole de verrouillage en deux phases

## Exemple

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

Graphe de 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 :



Cycle dans le graphe de précédence  $\Rightarrow$  ordonnancement non sérialisable

Protocole V2P :

# Protocole de verrouillage en deux phases

## Exemple

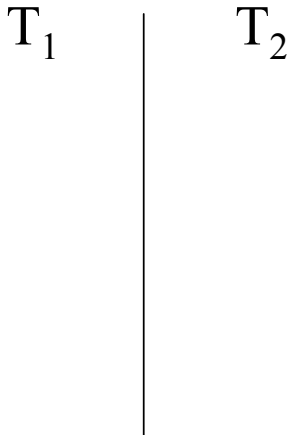
Soit l'ordonnancement :  $R_1(x)$ ,  $R_2(x)$ ,  $W_1(x)$ ,  $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence  $\Rightarrow$  ordonnancement non sérialisable

Protocole V2P :



# Protocole de verrouillage en deux phases

## Exemple

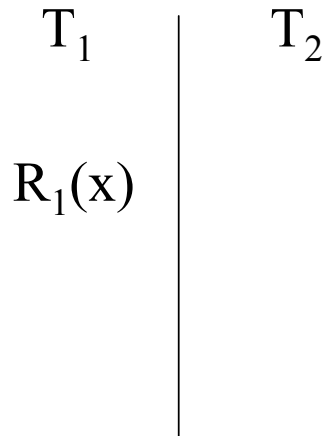
Soit l'ordonnancement :  $R_1(x)$ ,  $R_2(x)$ ,  $W_1(x)$ ,  $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence  $\Rightarrow$  ordonnancement non sérialisable

Protocole V2P :



# Protocole de verrouillage en deux phases

## Exemple

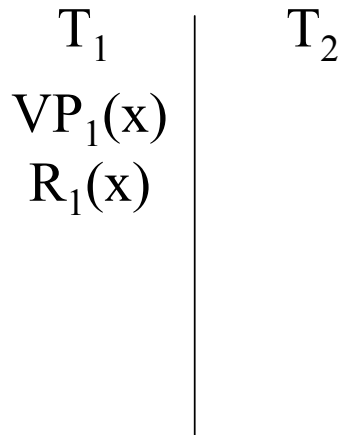
Soit l'ordonnancement :  $R_1(x)$ ,  $R_2(x)$ ,  $W_1(x)$ ,  $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence  $\Rightarrow$  ordonnancement non sérialisable

Protocole V2P :



# Protocole de verrouillage en deux phases

## Exemple

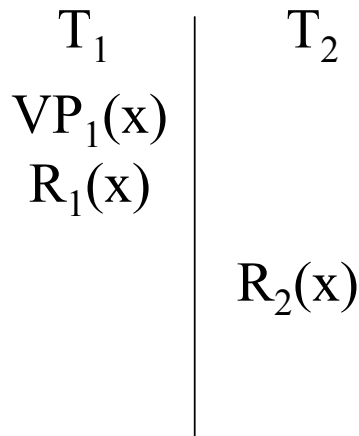
Soit l'ordonnancement :  $R_1(x)$ ,  $R_2(x)$ ,  $W_1(x)$ ,  $W_2(x)$

Graphe de précédence :



Cycle dans le graphe de précédence  $\Rightarrow$  ordonnancement non sérialisable

Protocole V2P :





# Protocole de verrouillage en deux phases

## Exemple

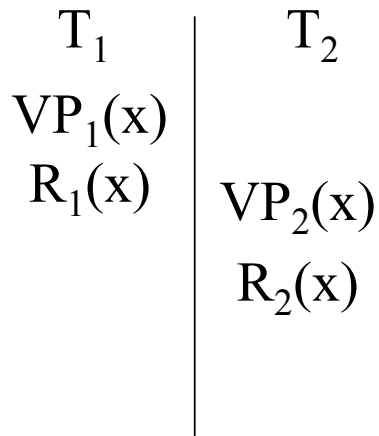
Soit l'ordonnancement :  $R_1(x)$ ,  $R_2(x)$ ,  $W_1(x)$ ,  $W_2(x)$

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 :

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

# 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 précédence :

Protocole V2P strict :

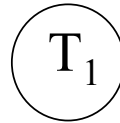
Protocole V2P non strict :

# 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 :

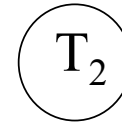
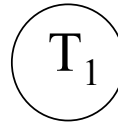
Protocole V2P non strict :

# 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 :

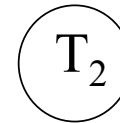
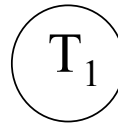
Protocole V2P non strict :

# Protocole de verrouillage en deux phases

## Exemple

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

Graphe de précedence :



Protocole V2P strict :

Protocole V2P non strict :

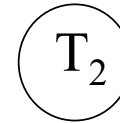
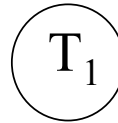


# Protocole de verrouillage en deux phases

## Exemple

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

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

# Protocole de verrouillage en deux phases

## Exemple

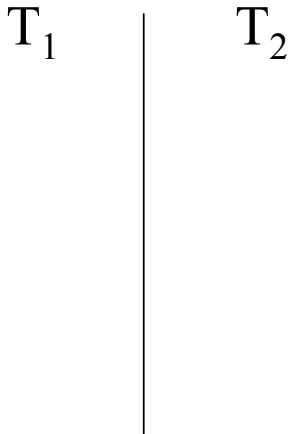
Soit l'ordonnancement :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :



Protocole V2P non strict :

# Protocole de verrouillage en deux phases

## Exemple

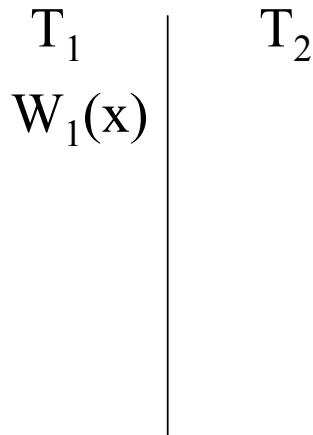
Soit l'ordonnancement :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

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 :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

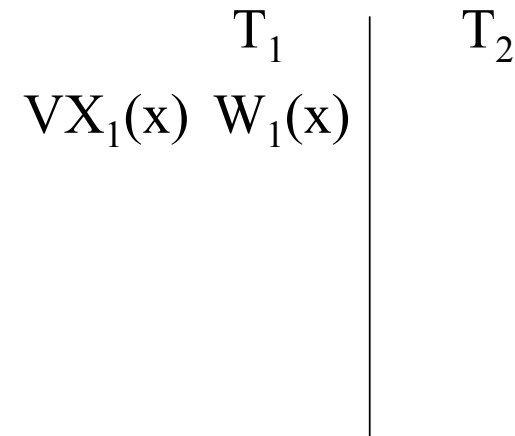
Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



# Protocole de verrouillage en deux phases

## Exemple

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

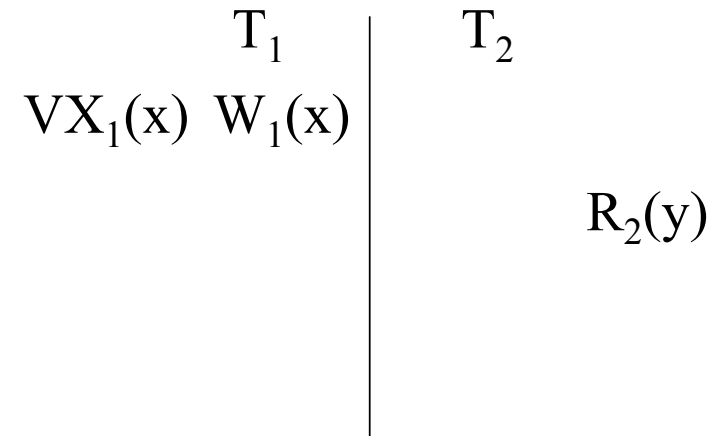
Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



# Protocole de verrouillage en deux phases

## Exemple

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

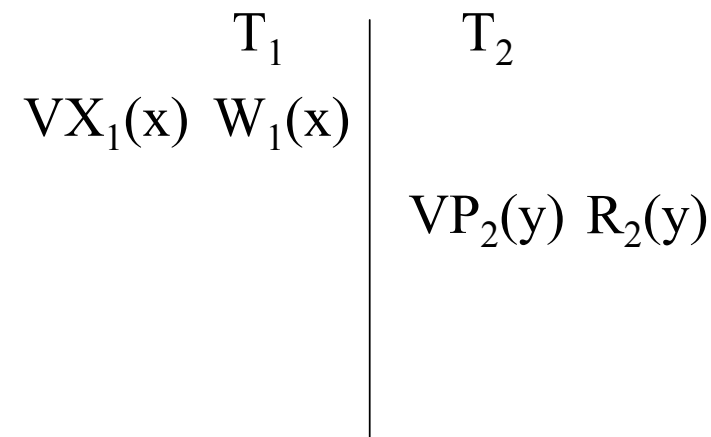
Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :





# Protocole de verrouillage en deux phases

## Exemple

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

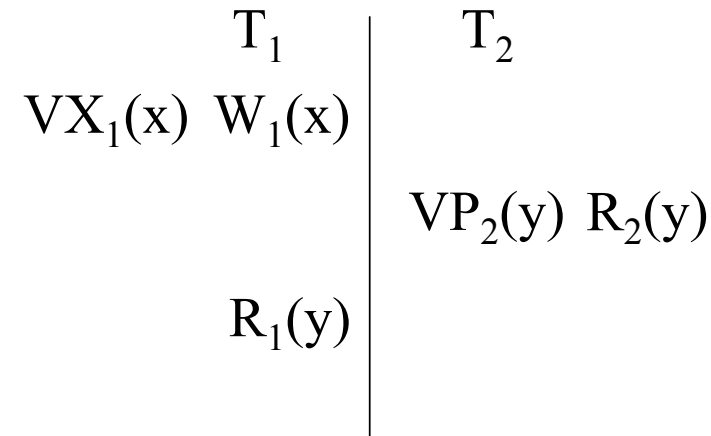
Graphe de précédence :



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 :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

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 :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  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 :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

$T_1$		$T_2$	
$VX_1(x)$	$W_1(x)$		
		$VP_2(y)$	$R_2(y)$
$VP_1(y)$	$R_1(y)$		
		<del><math>VP_2(x)</math></del>	<del><math>R_2(x)</math></del>

# Protocole de verrouillage en deux phases

## Exemple

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

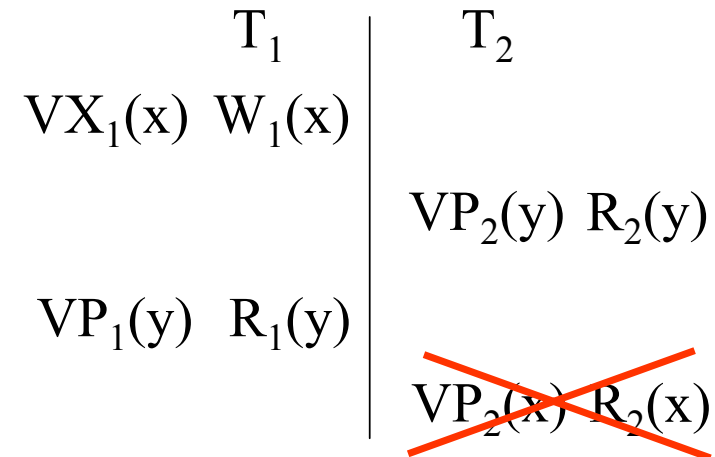
Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



# Protocole de verrouillage en deux phases

## Exemple

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

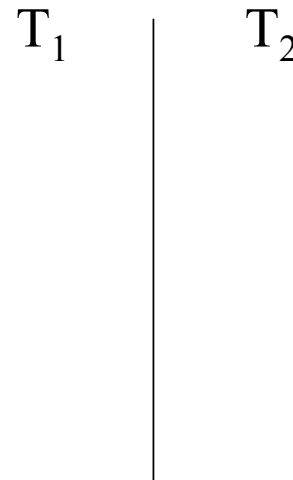
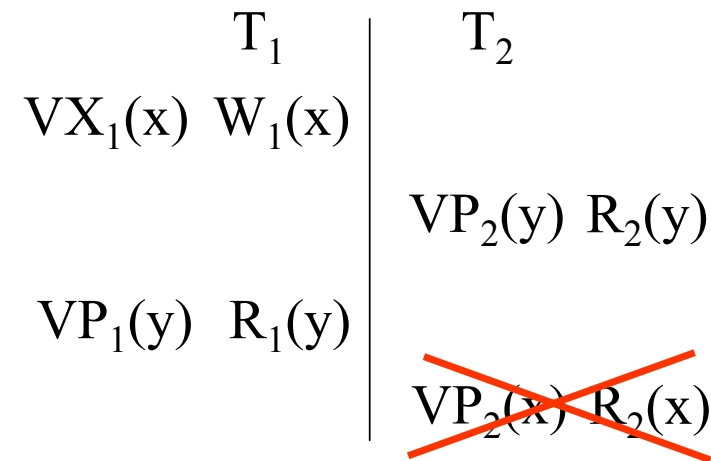
Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :



# Protocole de verrouillage en deux phases

## Exemple

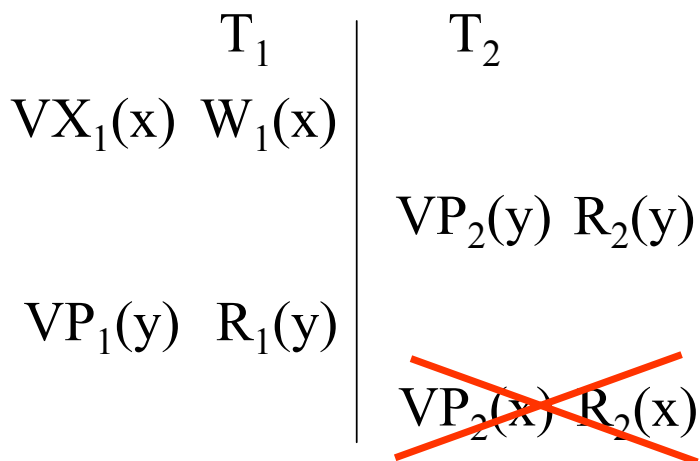
Soit l'ordonnancement :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :

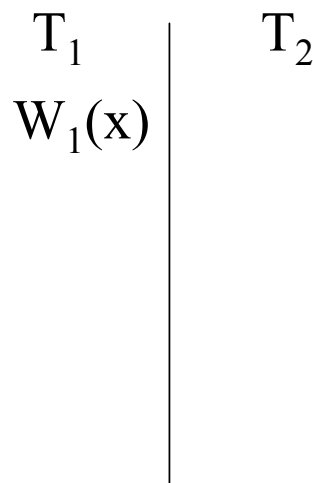


Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	

# 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

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

# 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

Protocole V2P non strict :

$T_1$	$T_2$
$VX_1(x) \quad W_1(x)$	
	$VP_2(y) \quad R_2(y)$
$VP_1(y) \quad R_1(y)$	
	<del><math>VP_2(x) \quad R_2(x)</math></del>

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

# 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

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 :  $W_1(x)$ ,  $R_2(y)$ ,  $R_1(y)$ ,  $R_2(x)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

$T_1$	$T_2$
$VX_1(x) \quad W_1(x)$	
	$VP_2(y) \quad R_2(y)$
$VP_1(y) \quad R_1(y)$	
	<del><math>VP_2(x) \quad R_2(x)</math></del>

Protocole V2P non strict :

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

# 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict :

$T_1$	$T_2$
$VX_1(x) \quad W_1(x)$	
	$VP_2(y) \quad R_2(y)$
$VP_1(y) \quad R_1(y)$	
	<del><math>VP_2(x) \quad R_2(x)</math></del>

Protocole V2P non strict :

$T_1$	$T_2$
$VX_1(x) \quad W_1(x)$	
	$VP_2(y) \quad R_2(y)$
$VP_1(y) \quad R_1(y)$	
$D_1(x)$	
	$VP_2(x) \quad R_2(x)$



# 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

**Protocole V2P strict :**

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

**Protocole V2P non strict :**

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	
	$VP_2(y)$ $R_2(y)$
$VP_1(y)$ $R_1(y)$	
$D_1(x)$	
	$VP_2(x)$ $R_2(x)$

# 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 précédence :

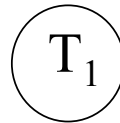
Protocole V2P strict ou non strict :

# 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 :



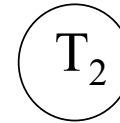
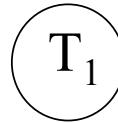
Protocole V2P strict ou non strict :

# 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 :



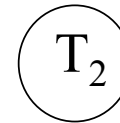
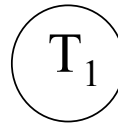
Protocole V2P strict ou non strict :

# 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 précedence :



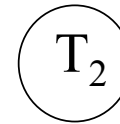
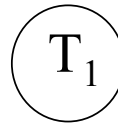
Protocole V2P strict ou non strict :

# 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 précedence :



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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :



# Protocole de verrouillage en deux phases

## Exemple

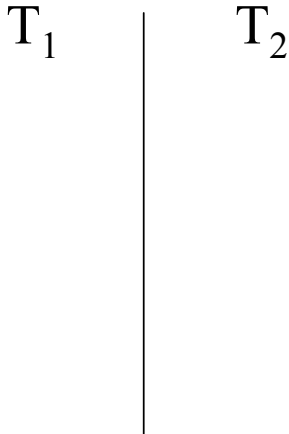
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $\underline{R_1(y)}$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :



# Protocole de verrouillage en deux phases

## Exemple

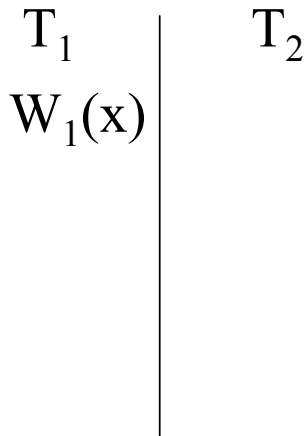
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

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

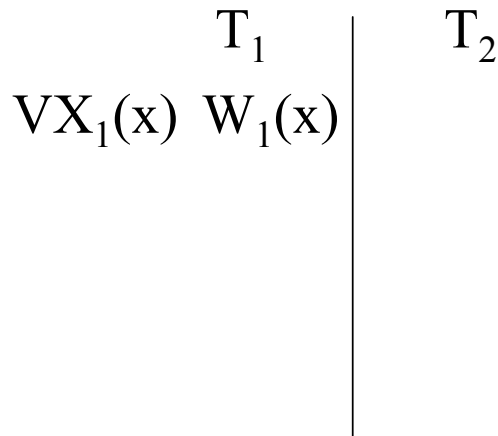
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :



# Protocole de verrouillage en deux phases

## Exemple

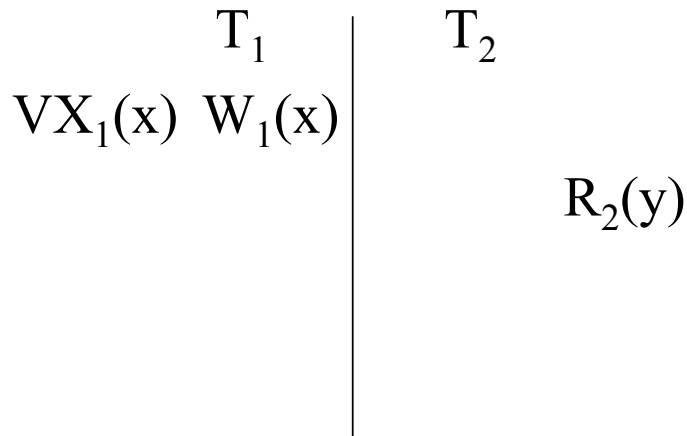
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $\underline{R_1(y)}$ ,  $C_1$ ,  $C_2$

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

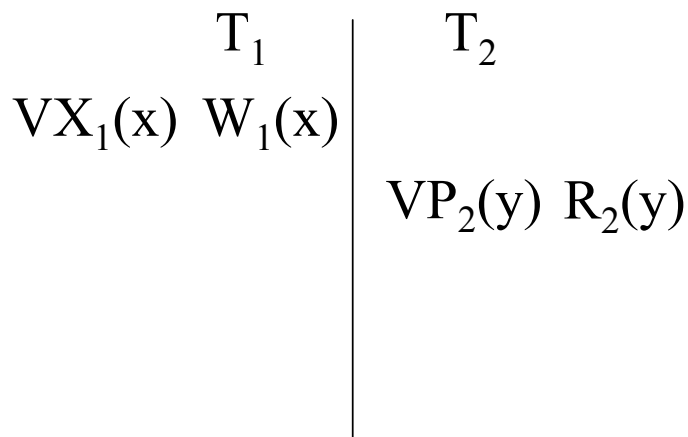
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

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

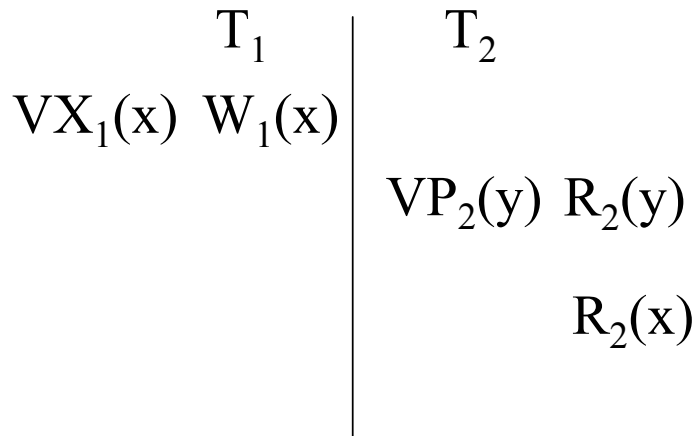
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :

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

# Protocole de verrouillage en deux phases

## Exemple

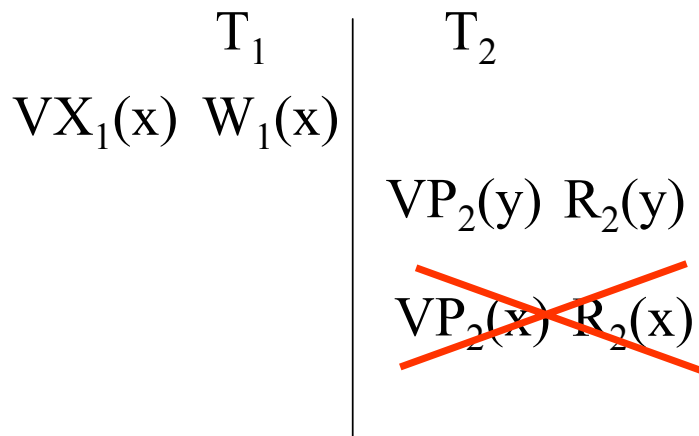
Soit l'ordonnancement :  $W_1(x)$ ,  $R_2(y)$ ,  $R_2(x)$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

Graphe de 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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	$VP_2(y)$ $R_2(y)$
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>

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

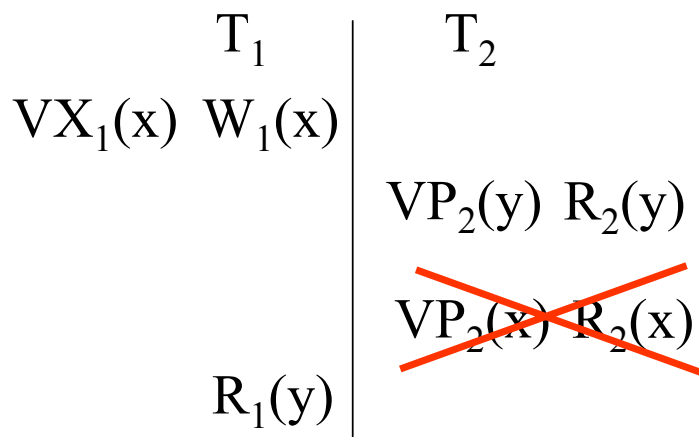
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $\underline{R_1(y)}$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  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 :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	$VP_2(y)$ $R_2(y)$
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>
$VP_1(y)$ $R_1(y)$	

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 précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	$VP_2(y)$ $R_2(y)$
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>
$VP_1(y)$ $R_1(y)$	

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

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  ordonnancement sérialisable

Protocole V2P strict ou non strict :

$T_1$	$T_2$
$VX_1(x)$ $W_1(x)$	$VP_2(y)$ $R_2(y)$
	<del><math>VP_2(x)</math> <math>R_2(x)</math></del>
$VP_1(y)$ $R_1(y)$	

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

**V2P  $\Rightarrow$  sérialisable**

# Protocole de verrouillage en deux phases

## Exemple

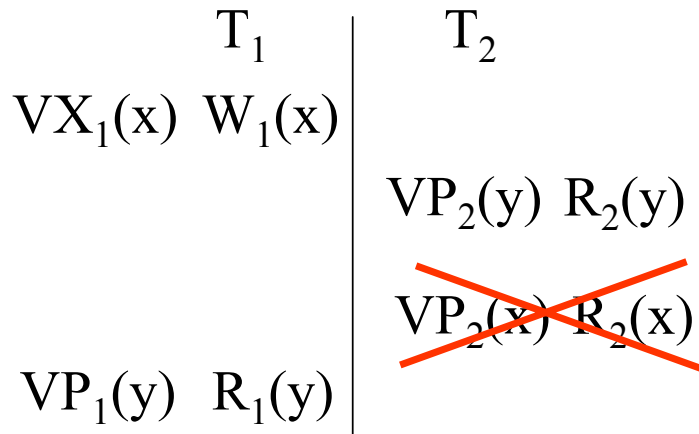
Soit l'ordonnancement :  $\underline{W_1(x)}$ ,  $R_2(y)$ ,  $\underline{R_2(x)}$ ,  $R_1(y)$ ,  $C_1$ ,  $C_2$

Graphe de précedence :



Pas de cycle dans le graphe de précedence  $\Rightarrow$  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**

**V2P  ~~$\nRightarrow$~~  sérialisable**

# Ordonnancement 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 : ...  $W_i(x)$ ,  $R_j(y)$ ,  $R_i(y)$ ,  $R_j(x)$  ...

$T_i$	$T_j$	$x$		$y$		
RTS	WTS	RTS	WTS	RTS	WTS	
100	125	0	0	0	0	$T_i + \text{ancienne que } T_j$



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

$T_i$  + ancienne que  $T_j$



# Ordonnancement par estampillage

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

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

$T_i$  + ancienne que  $T_j$

# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	$RTS^x$ 0	$WTS^x$ 0	$RTS^y$ 0	$WTS^y$ 0	$T_i + \text{ancienne que } T_j$
$W_i(x)$	$R_j(y)$		100	125		

# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	$x$		$y$		$T_i + \text{ancienne que } T_j$
		RTS 0	WTS 0	RTS 0	WTS 0	
$W_i(x)$	$R_j(y)$		100			
$R_i(y)$				125		

# Ordonnancement par estampillage

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

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

# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	$RTS^x$ 0	$WTS^x$ 0	$RTS^y$ 0	$WTS^y$ 0	$T_i$ + ancienne que $T_j$
$W_i(x)$			100			
	$R_j(y)$			125		
$R_i(y)$				125		<p>← <math>TS(T_i) &gt; WTS(y)</math> et la dernière transaction la plus récente ayant lu <math>y</math> est toujours <math>T_j</math></p>

# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	$x$		$y$		$T_i + \text{ancienne que } T_j$
		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 $T_j$
	$R_j(x)$					



# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	RTS <sup>x</sup> 0	WTS <sup>x</sup> 0	RTS <sup>y</sup> 0	WTS <sup>y</sup> 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 $T_j$
	$R_j(x)$					TS( $T_j$ ) > WTS(x) $\Rightarrow$ la dernière transaction ayant mis à jour x est plus ancienne que $T_j \Rightarrow$ écriture autorisée

# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	RTS <sup>x</sup> 0	WTS <sup>x</sup> 0	RTS <sup>y</sup> 0	WTS <sup>y</sup> 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

# Ordonnancement par estampillage

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

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

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

$T_i$  + récente que  $T_j$

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

$T_i$ 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		



# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

$T_i$ 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 : ...  $W_i(x)$ ,  $R_j(y)$ ,  $R_i(y)$ ,  $R_j(x)$  ...

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

$T_i$  + récente que  $T_j$

$T_j$  aurait du lire la valeur de  $x$  avant que  $T_i$ , plus récente, n'ait modifié la valeur de  $x$   
 $\Rightarrow$  annulation de  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + ancienne que  $T_j$

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

$T_i$  + ancienne que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + ancienne que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + ancienne que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + ancienne que  $T_j$



# Ordonnancement par estampillage

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

$T_i$ 100	$T_j$ 125	$RTS^x$ 0	$WTS^x$ 0	$RTS^y$ 0	$WTS^y$ 0	$T_i + \text{ancienne que } T_j$
$R_i(x)$	$R_j(y)$	100				
$R_i(y)$				125		

# Ordonnement par estampillage

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

$T_i$ 100	$T_j$ 125	$x$		$y$		$T_i + \text{ancienne que } T_j$
RTS 0	WTS 0	RTS 0	WTS 0	RTS 0	WTS 0	
$R_i(x)$	$R_j(y)$	100				
				125		
$R_i(y)$						

←  $TS(T_i) > WTS(y) \Rightarrow$  la dernière transaction la plus récente ayant lu  $y$  est toujours  $T_j$

# Ordonnancement par estampillage

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

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

# Ordonnement par estampillage

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

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

# Ordonnement par estampillage

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

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

# Ordonnement par estampillage

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

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

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

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

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

$T_i$  + récente que  $T_j$



# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

$T_i$	$T_j$	$x$		$y$	
		RTS	WTS	RTS	WTS
99	80	0	0	0	0
$R_i(x)$		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 lise  $\Rightarrow$  annulation de  $T_j$

# Ordonnancement par estampillage

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

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



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

$T_i$  + récente que  $T_j$

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

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

$T_i$  + récente que  $T_j$

# Ordonnancement par estampillage

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

$T_i$ 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		
	$W_j(x)$					

# Ordonnancement par estampillage

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

$T_i$ 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	
	$W_j(x)$				

$T_i$  + 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)



# Ordonnancement par estampillage

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

$T_i$ 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	
	$W_j(x)$				

$T_i$  + 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	$RTS=0$ $WTS=0$
$R_i(A)$ $W_i(A)$	$R_j(A)$ $W_j(A)$			$RTS=150$ $WTS=0$ $RTS=150$ $WTS=150$ $RTS=200$ $WTS=150$ $RTS=200$ $WTS=200$
		$R_k(A)$ 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>			
$R_i(A)$ $W_i(A)$	$R_j(A)$ $W_j(A)$	$R_k(A)$ <del>Abort</del>	$R_n(A)$	Lue	Crée Lue Lue	Crée  Lue

$T_k$  lit une ancienne version de l'item

# Verrouillage hiérarchique

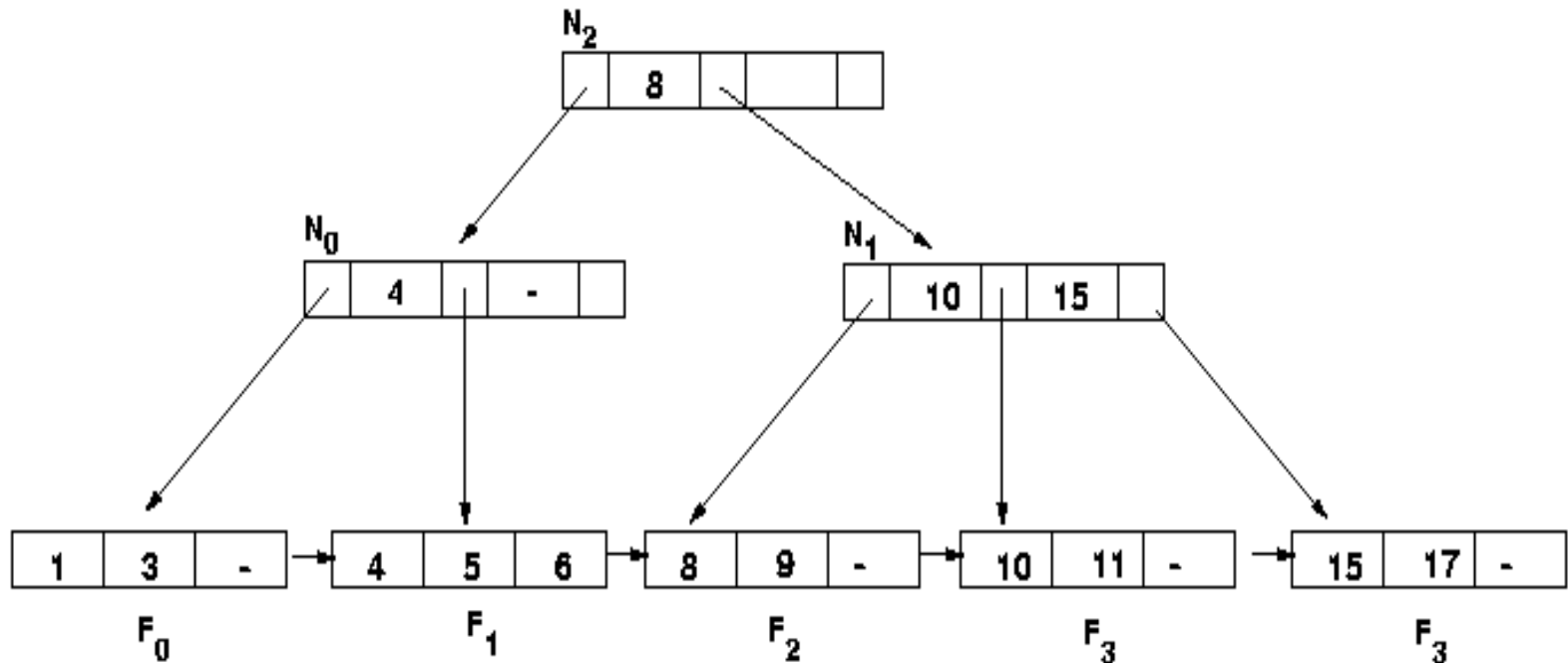
- **Hiérarchie : relation, page, nuplet**
- **Intention d'obtenir un verrou partagé (IP) ou exclusif (IX)**
- **Protocole garantissant la sérialisation et évitant les inter-blocages**

	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+

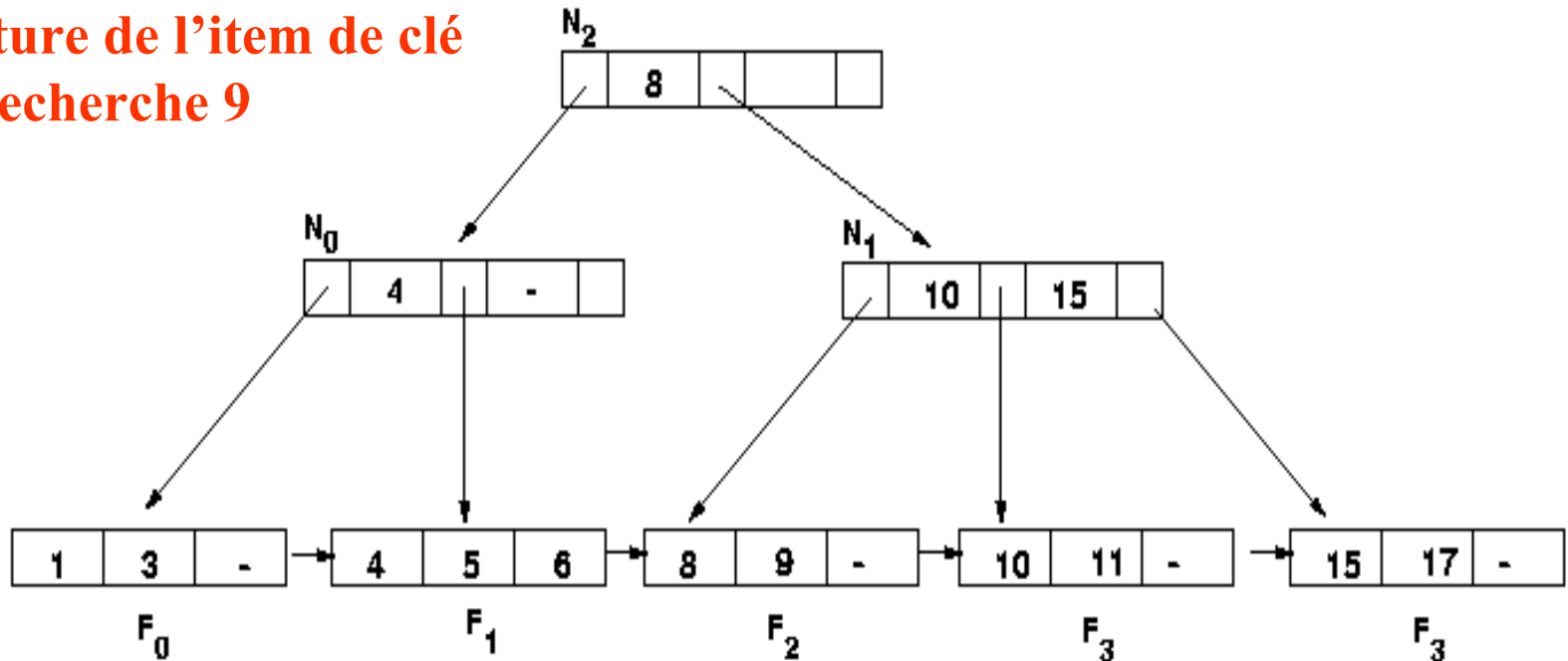
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



# 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

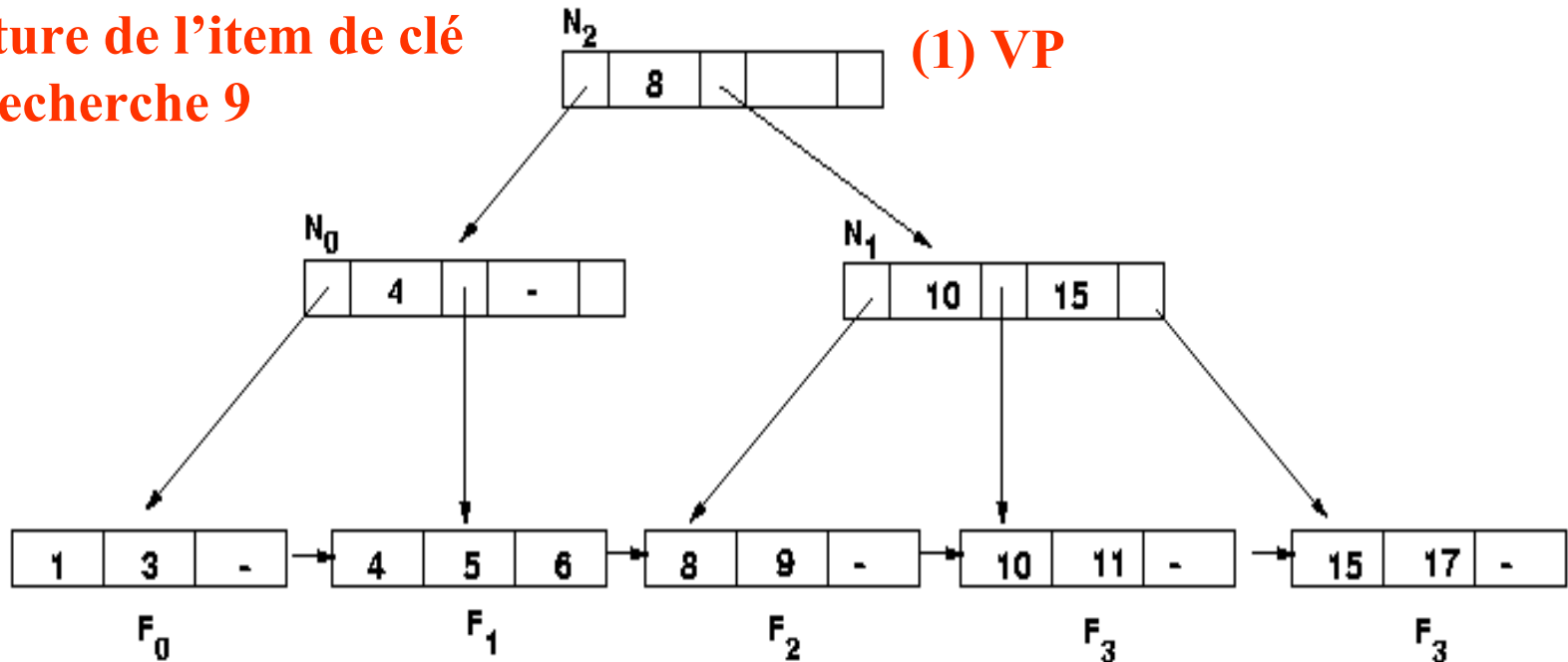
Lecture de l'item de clé  
de recherche 9



# 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

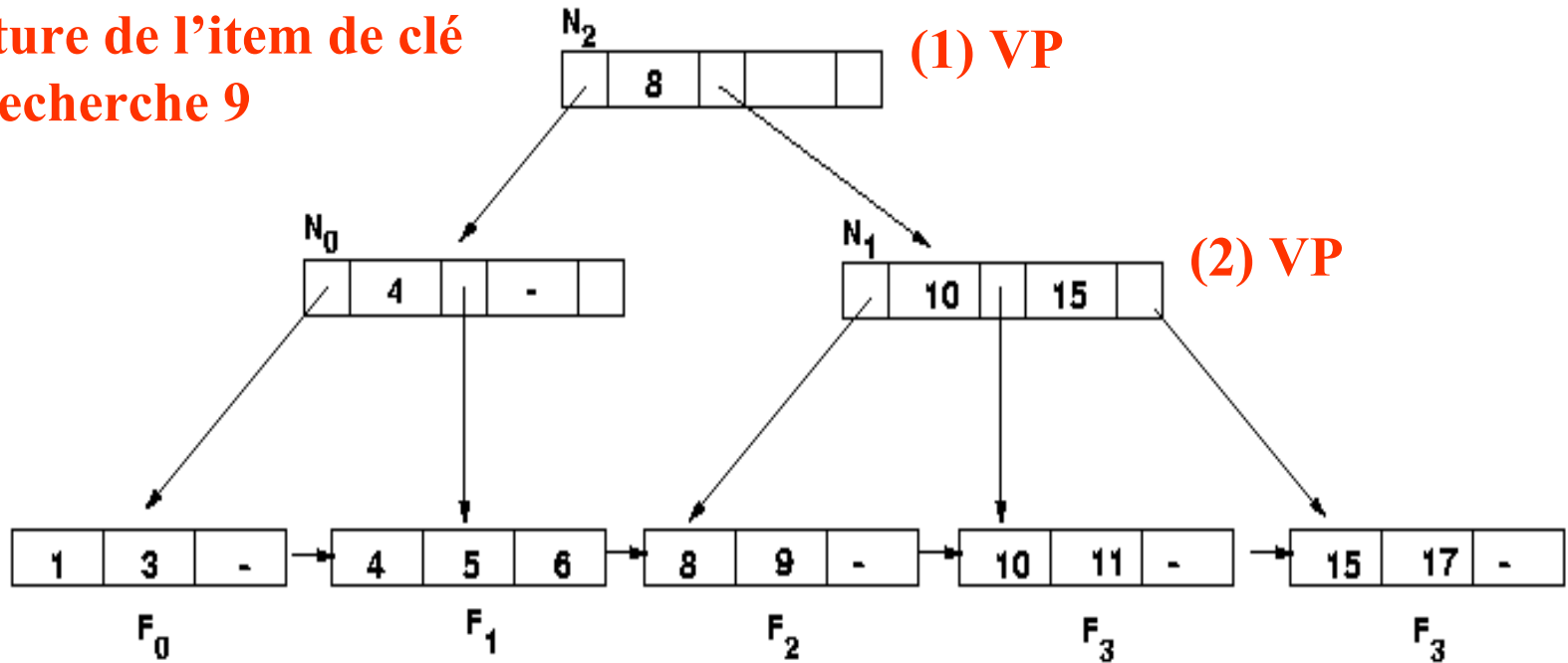
Lecture de l'item de clé  
de recherche 9



# 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

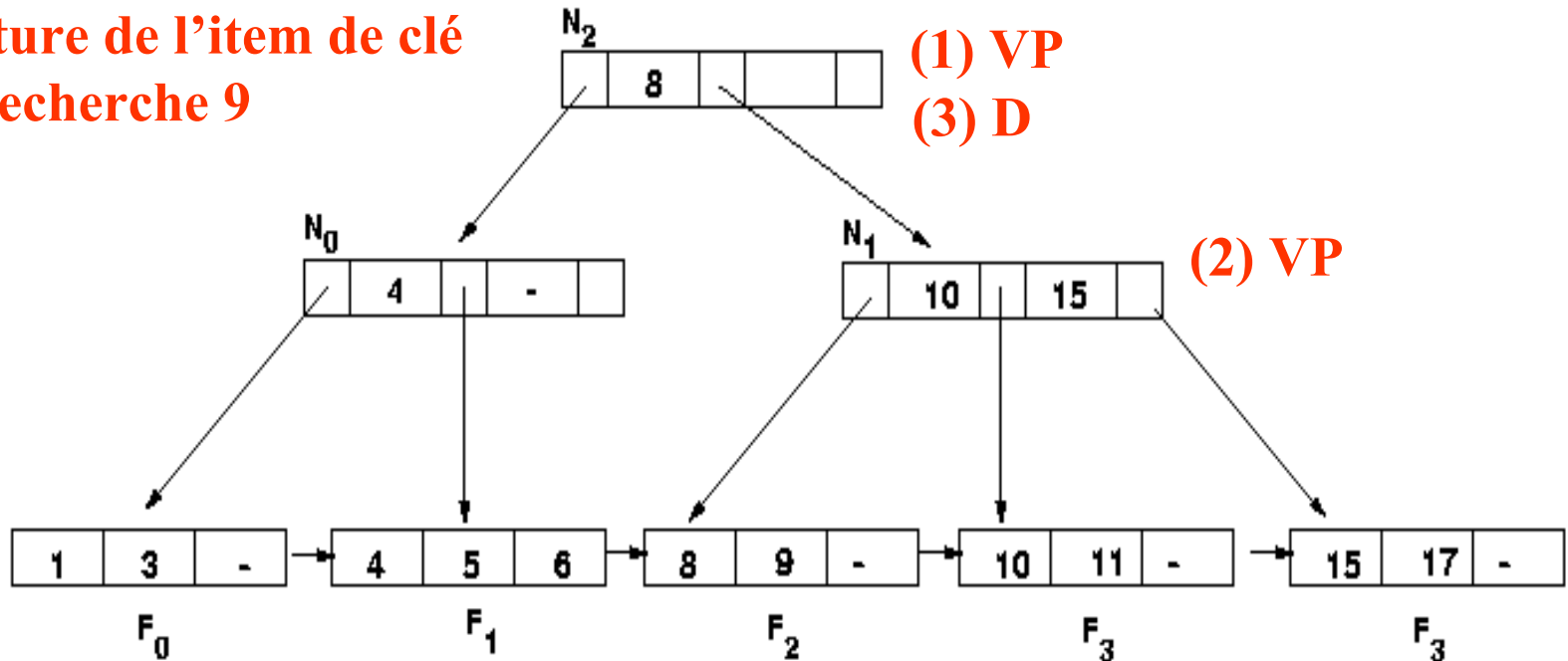




# Concurrency dans un arbre B+

- 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

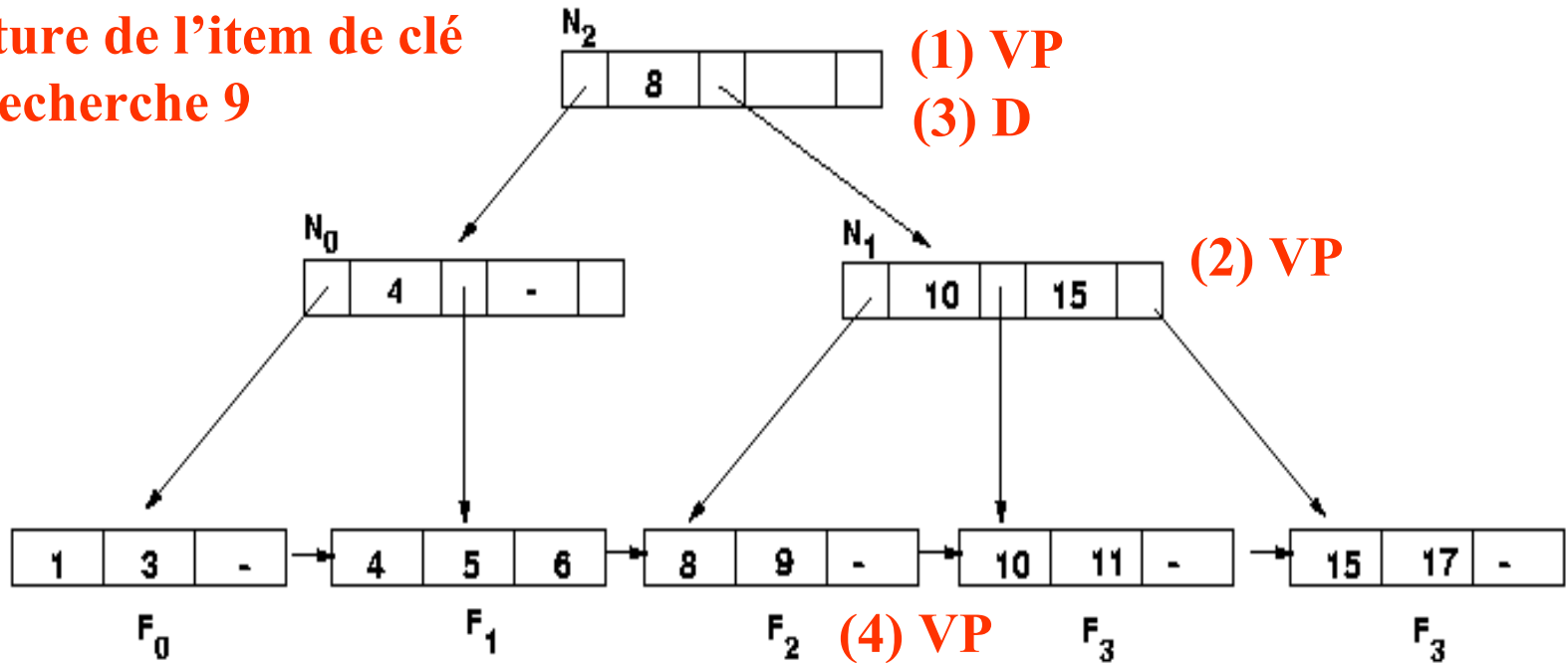
Lecture de l'item de clé  
de recherche 9



# 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

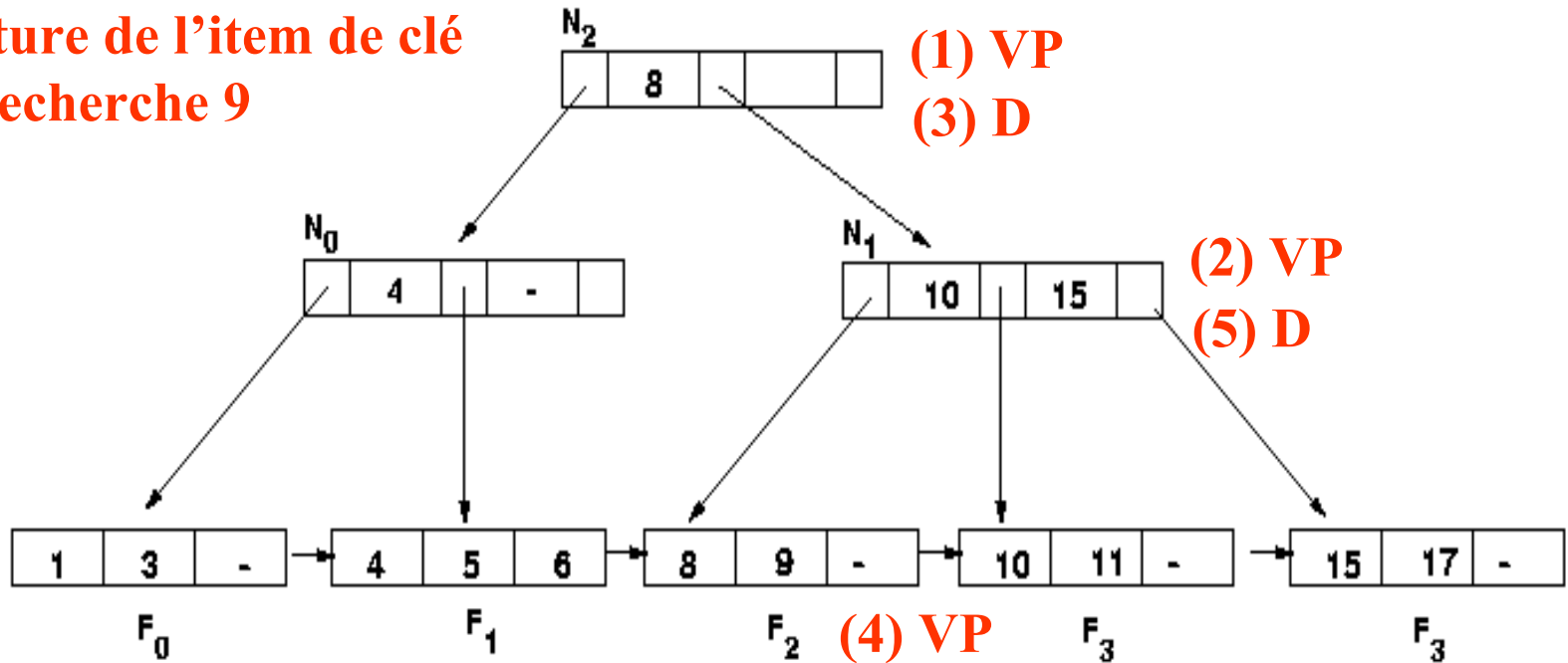
Lecture de l'item de clé  
de recherche 9



# Concurrency dans un arbre B+

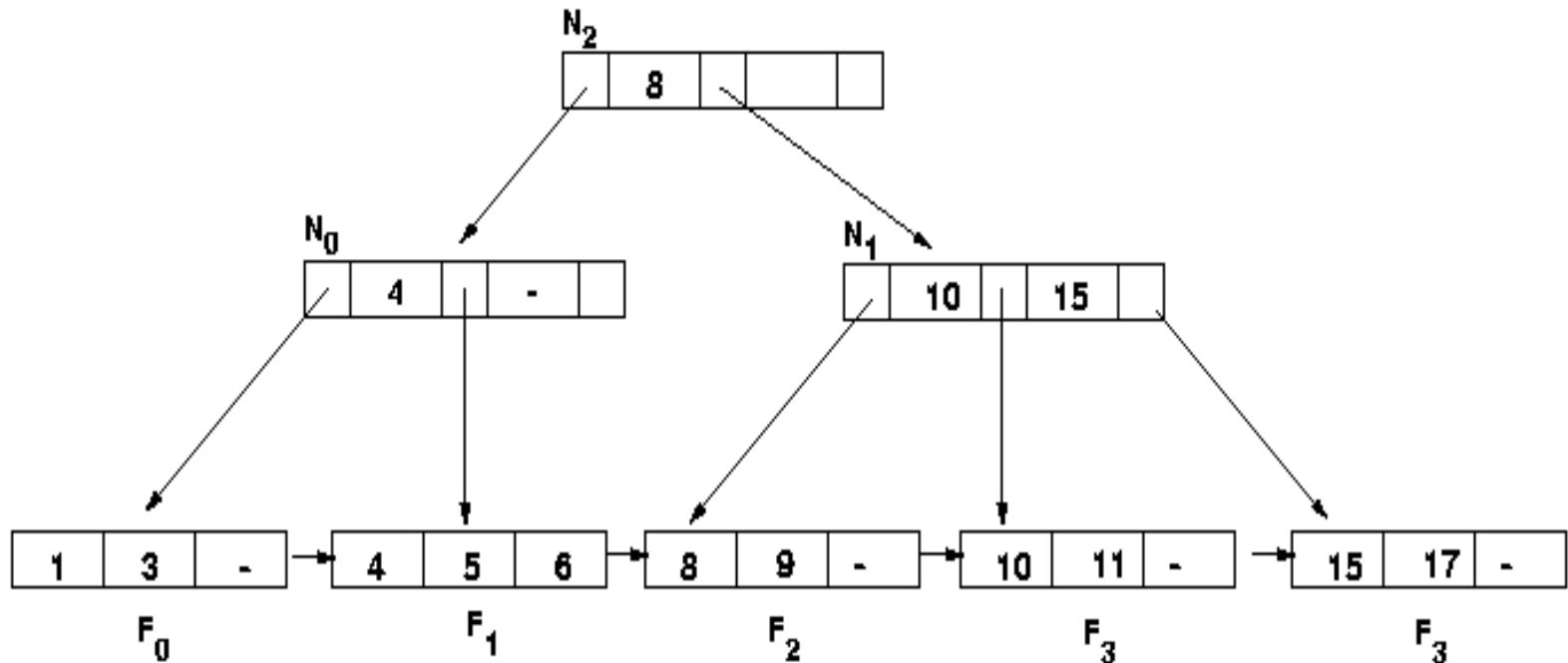
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

Lecture de l'item de clé  
de recherche 9



# Concurrency dans un arbre B+

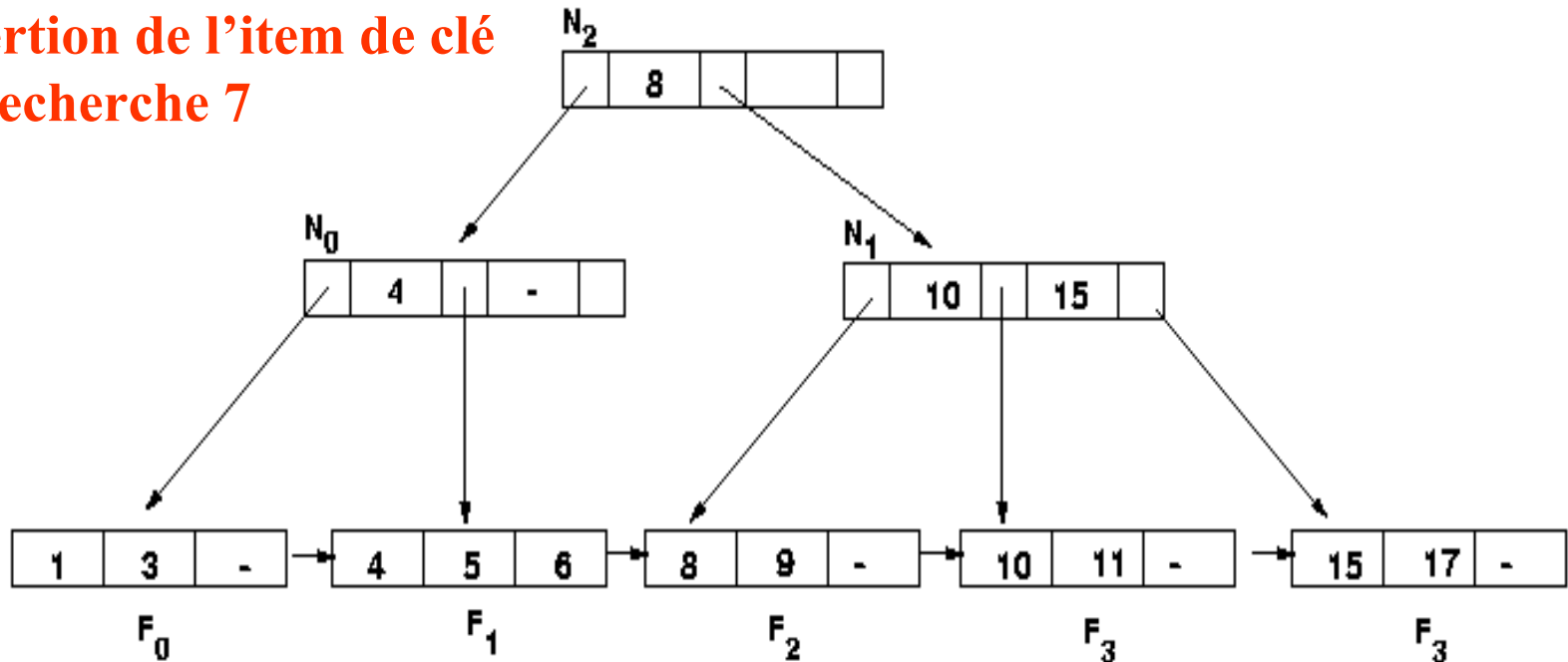
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



# 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

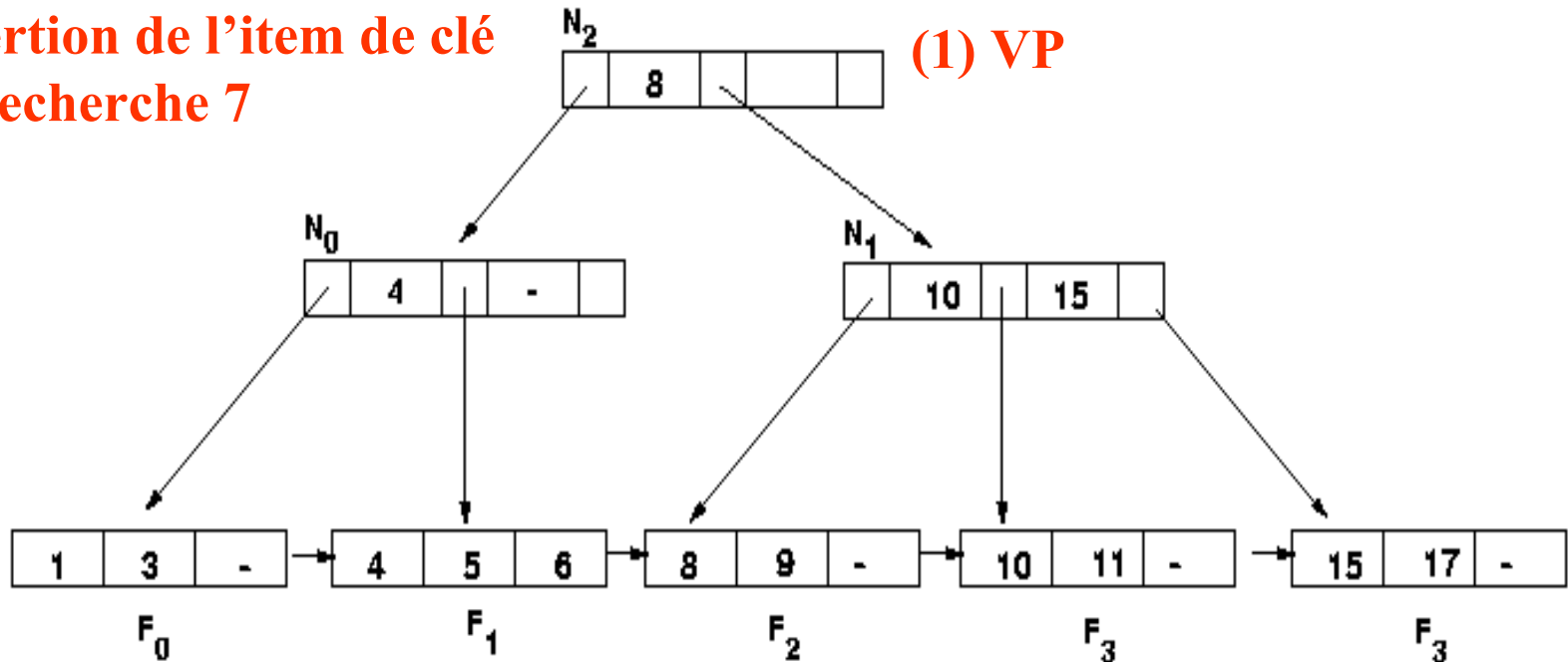
Insertion de l'item de clé de recherche 7



# 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

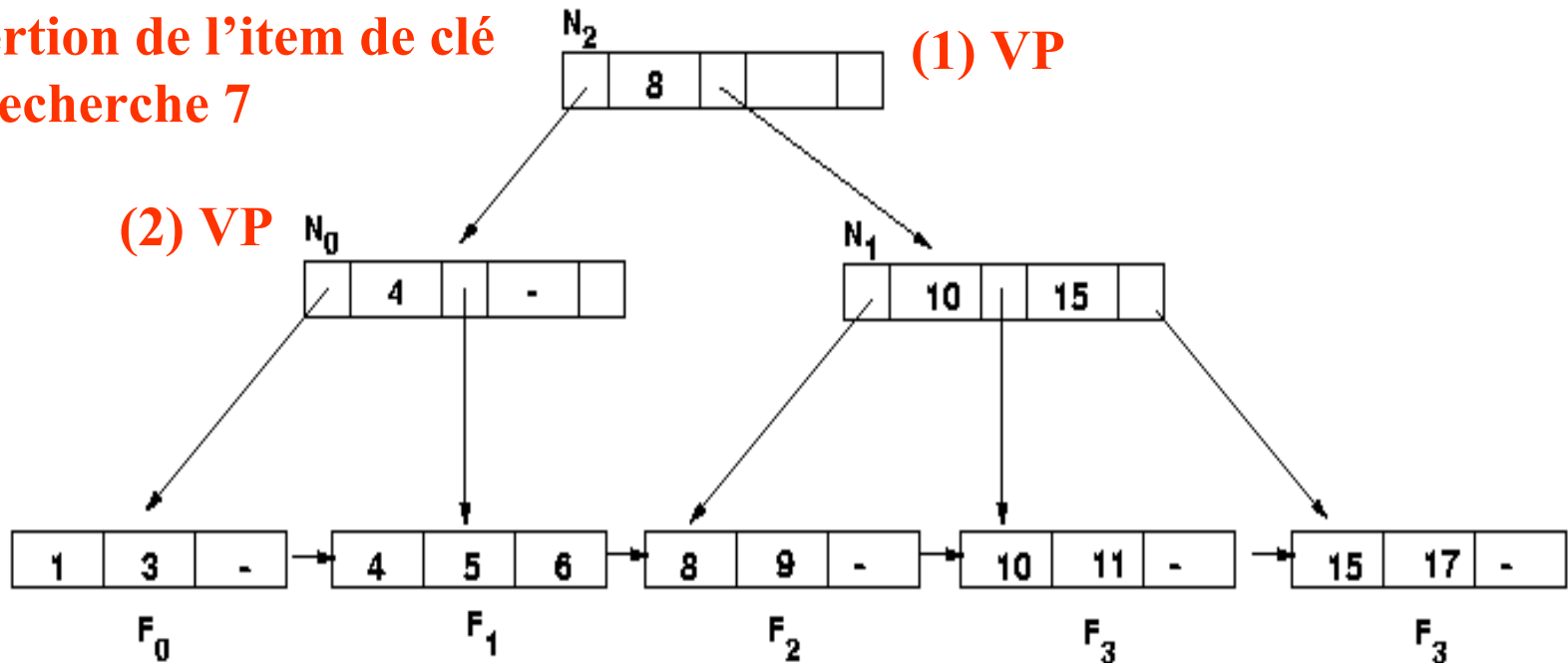
Insertion de l'item de clé  
de recherche 7



# 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

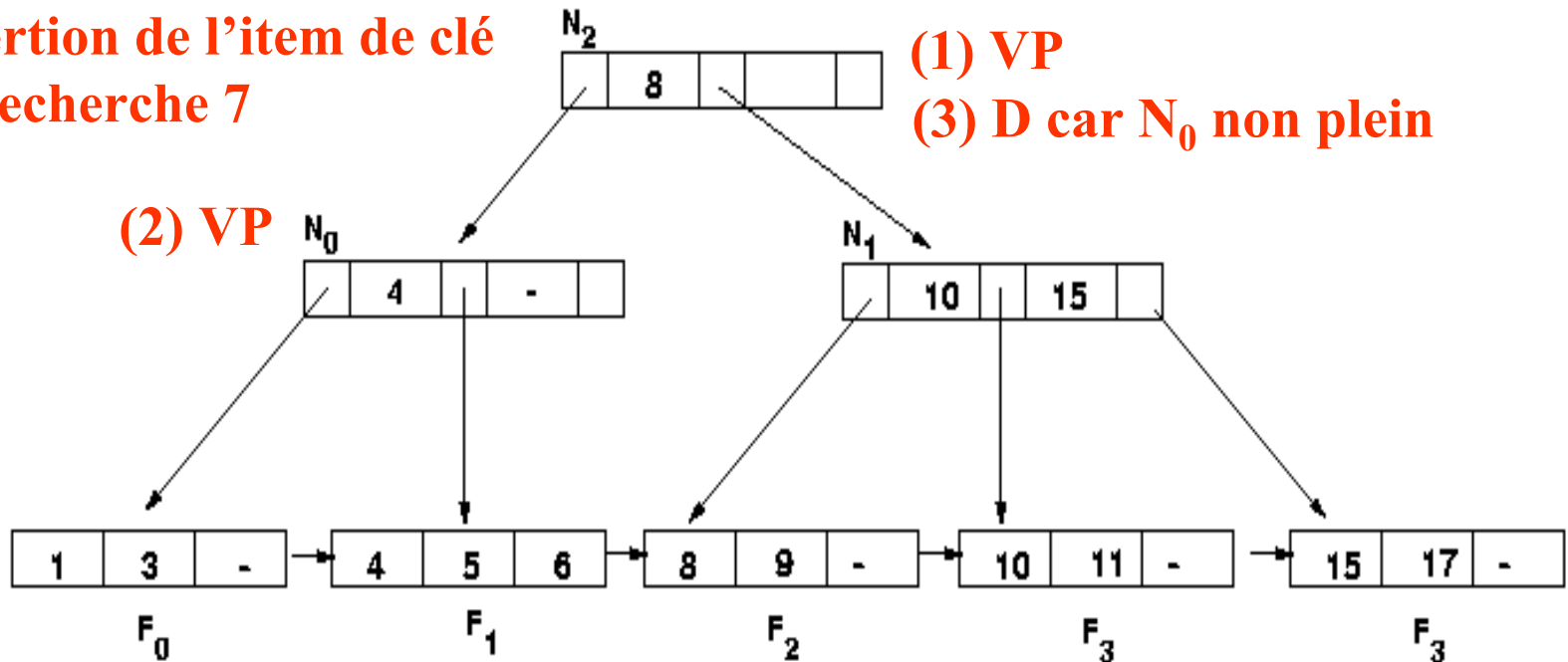
Insertion de l'item de clé  
de recherche 7



# 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

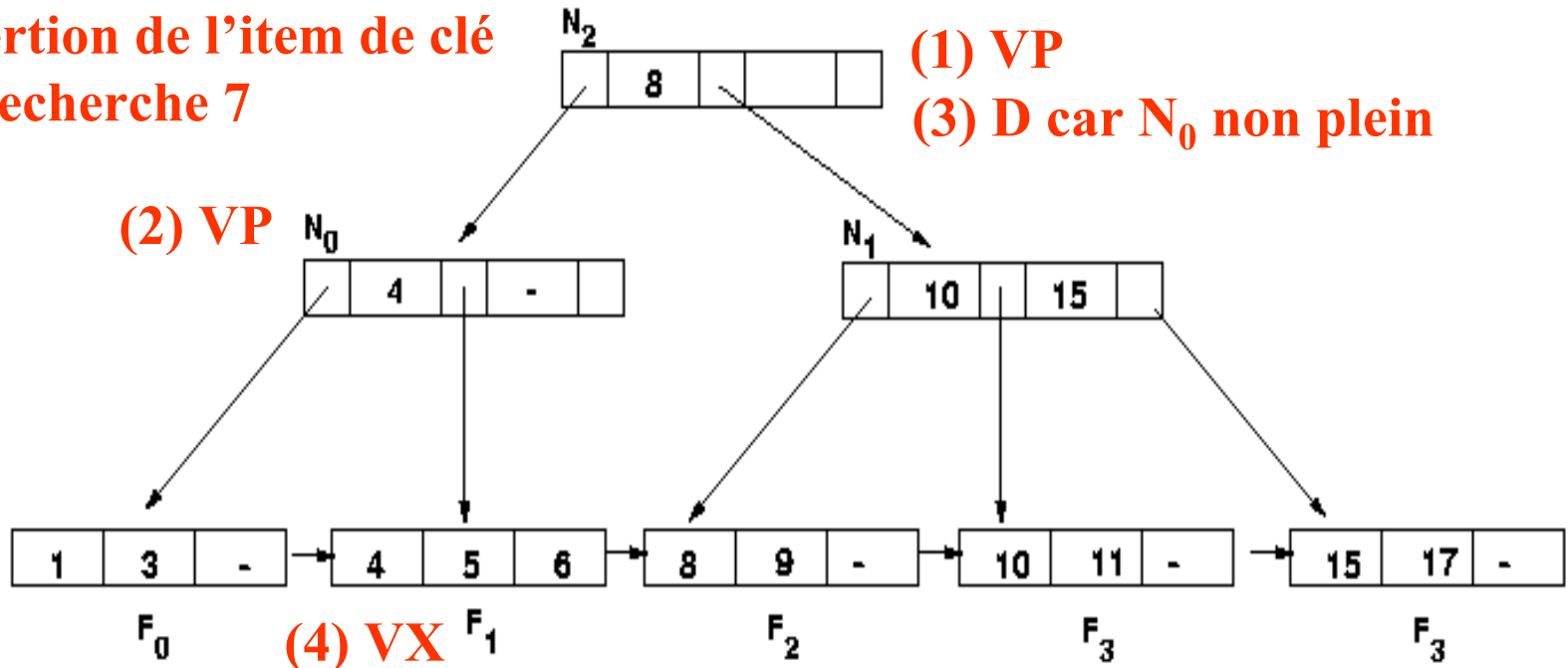




# Concurrency dans un arbre B+

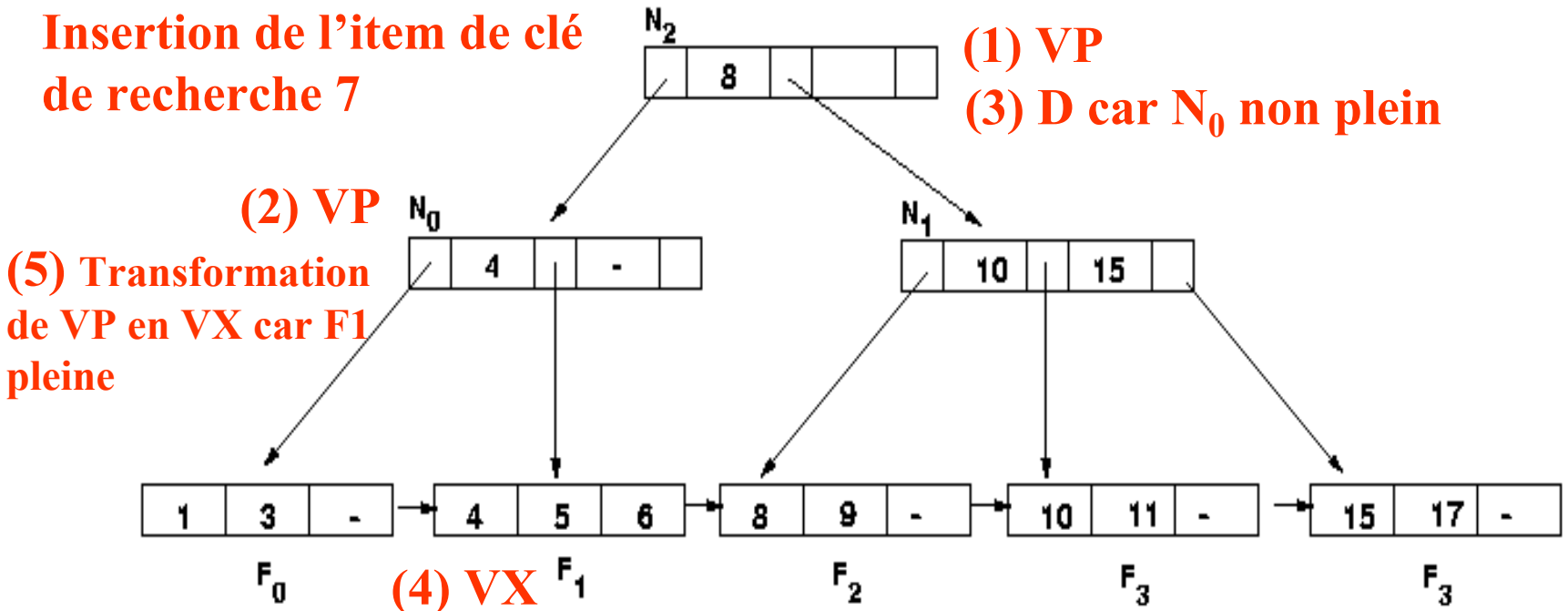
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour

Insertion de l'item de clé  
de recherche 7



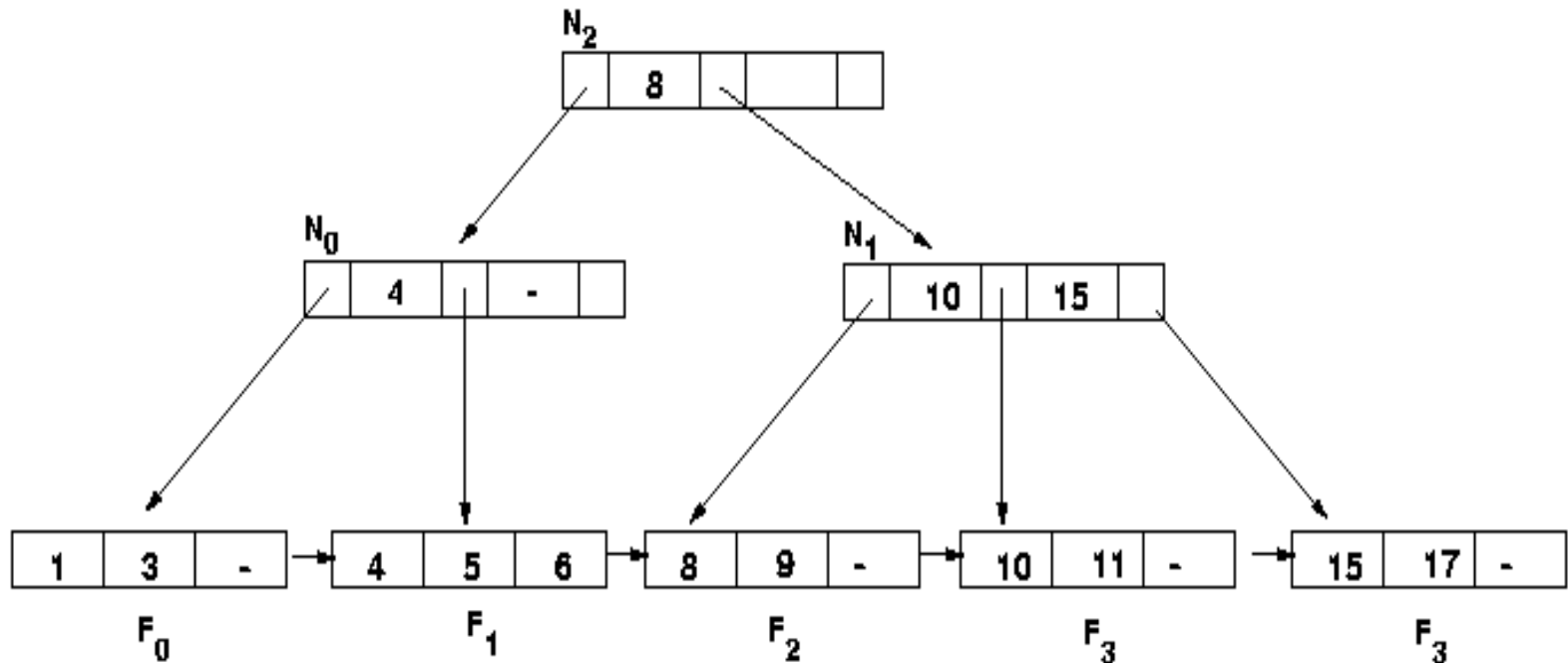
# Concurrency dans un arbre B+

- 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+

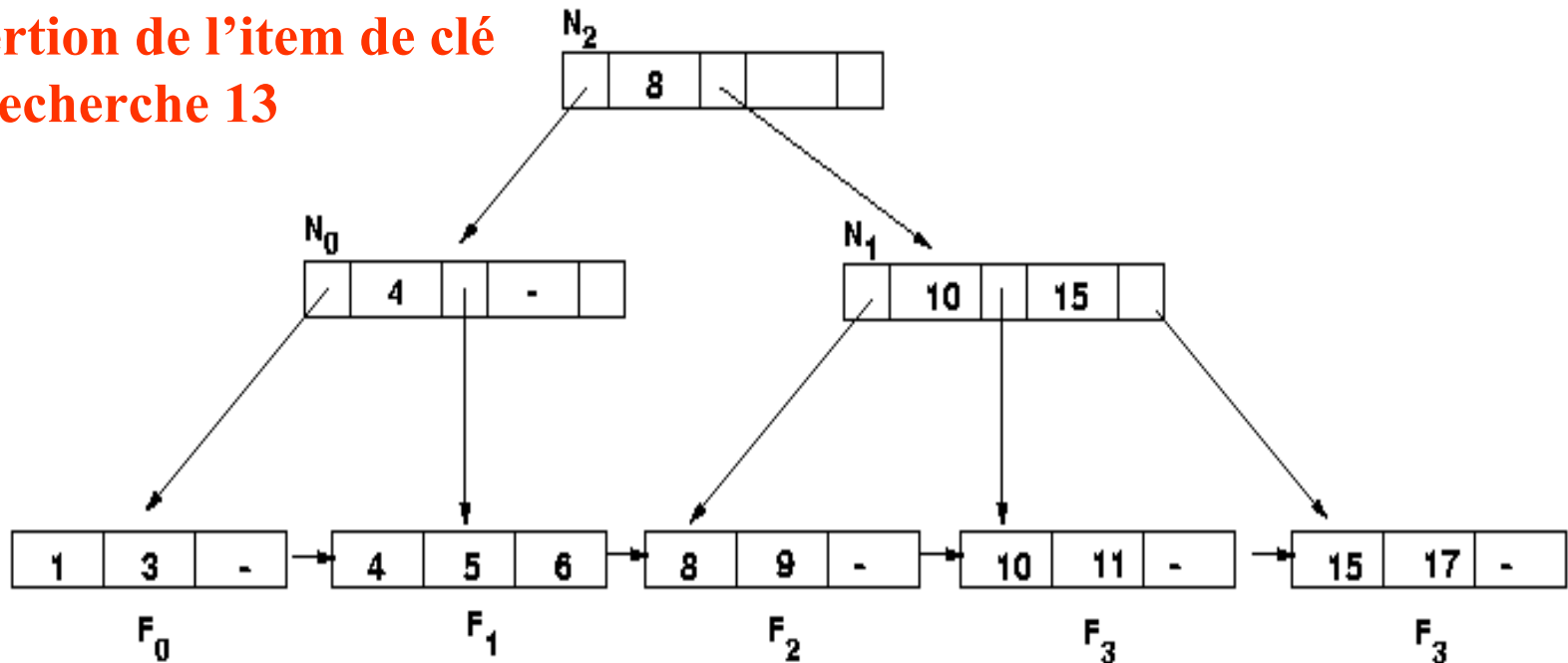
- Les niveaux hauts de l'arbre servent pour les recherches
- Pour les modifications : verrouillage exclusif d'un nœud s'il risque d'être touché par la mise à jour



# 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

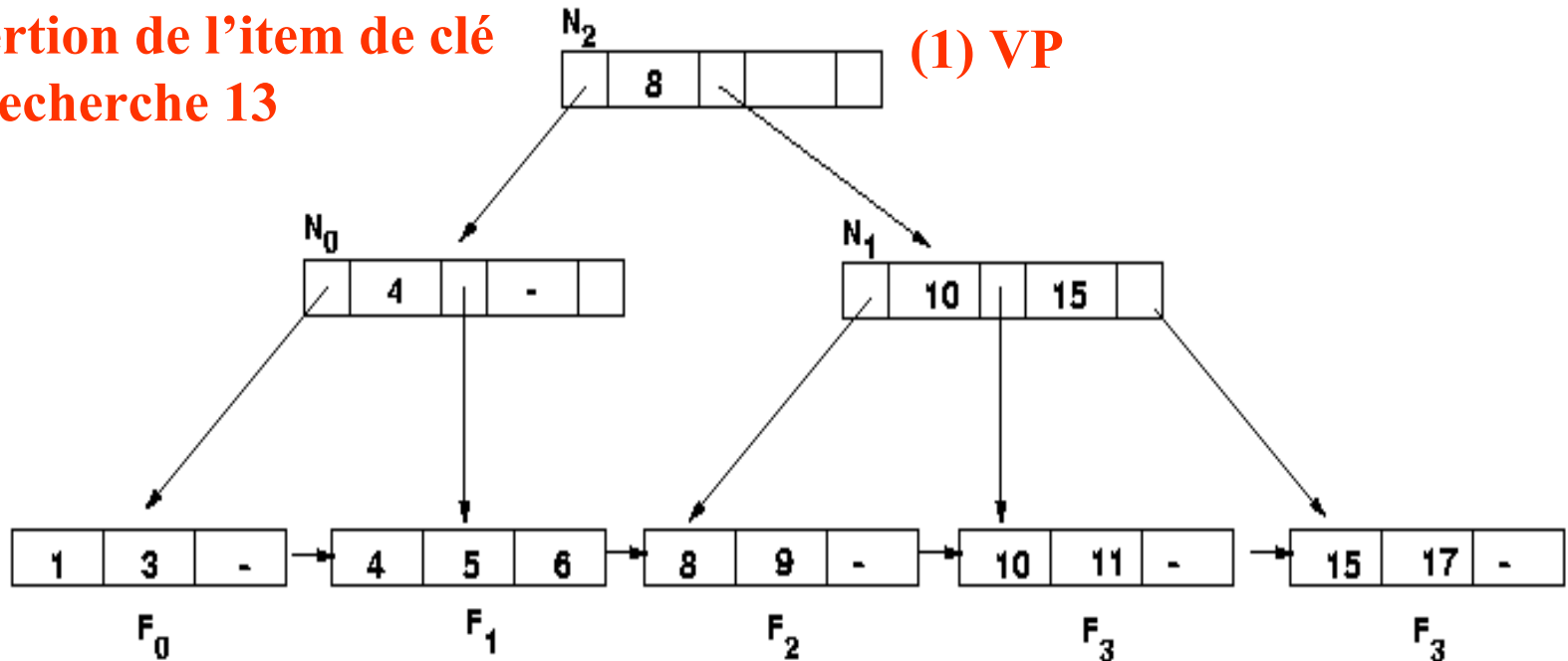
Insertion de l'item de clé de recherche 13



# 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

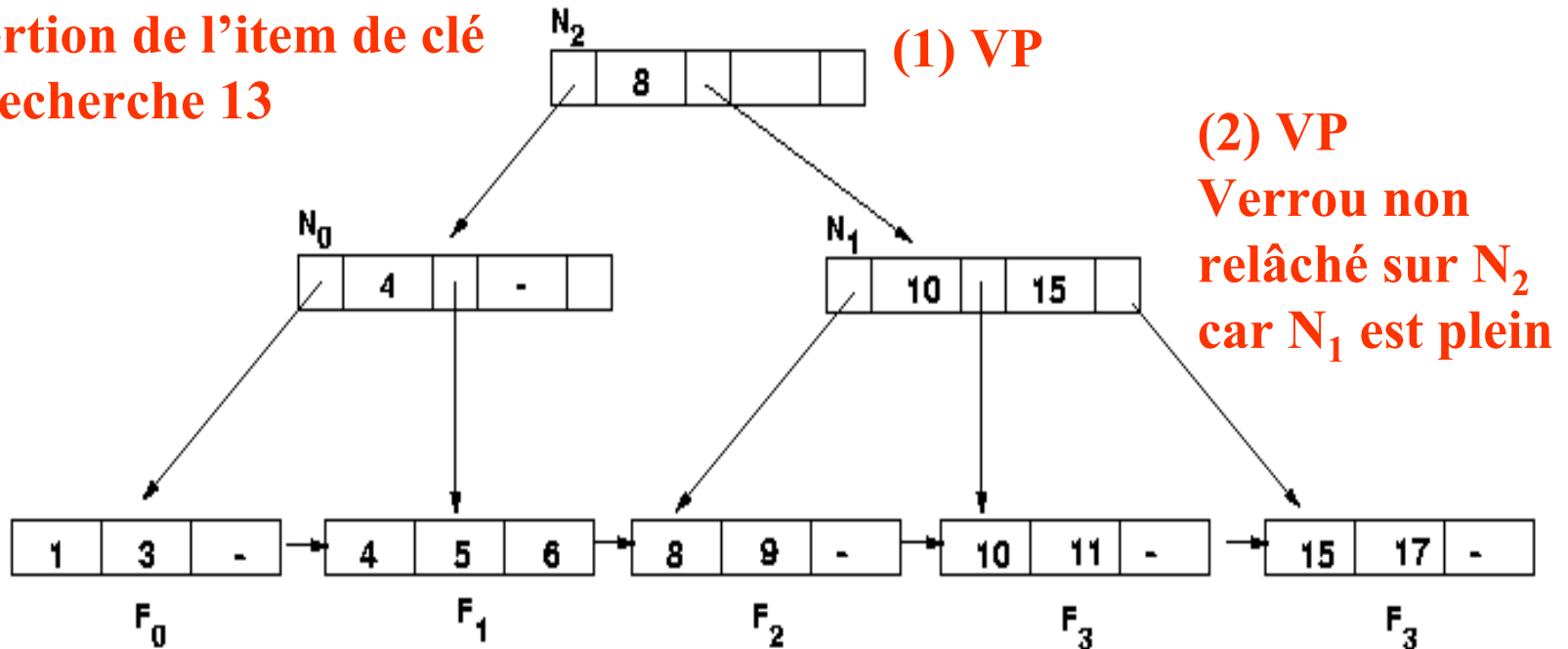
Insertion de l'item de clé  
de recherche 13



# 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

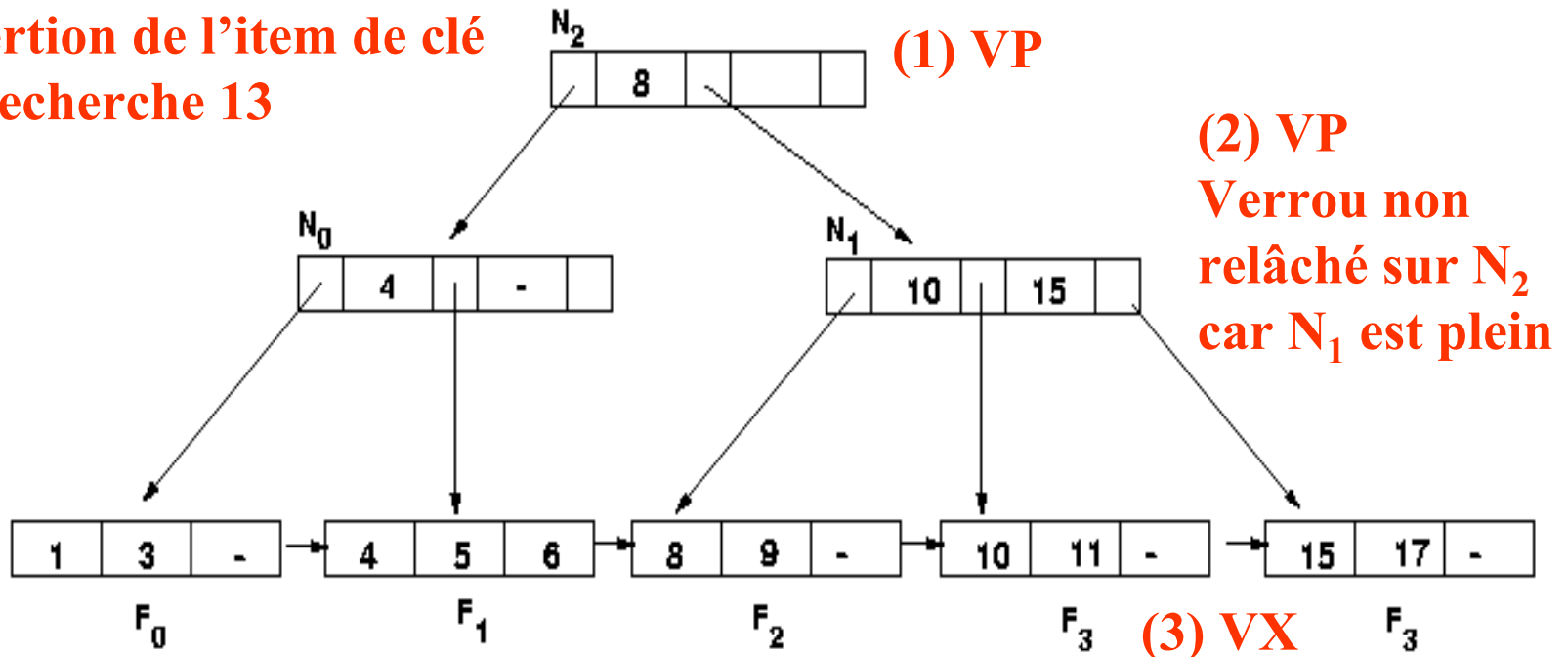
Insertion de l'item de clé  
de recherche 13



# 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

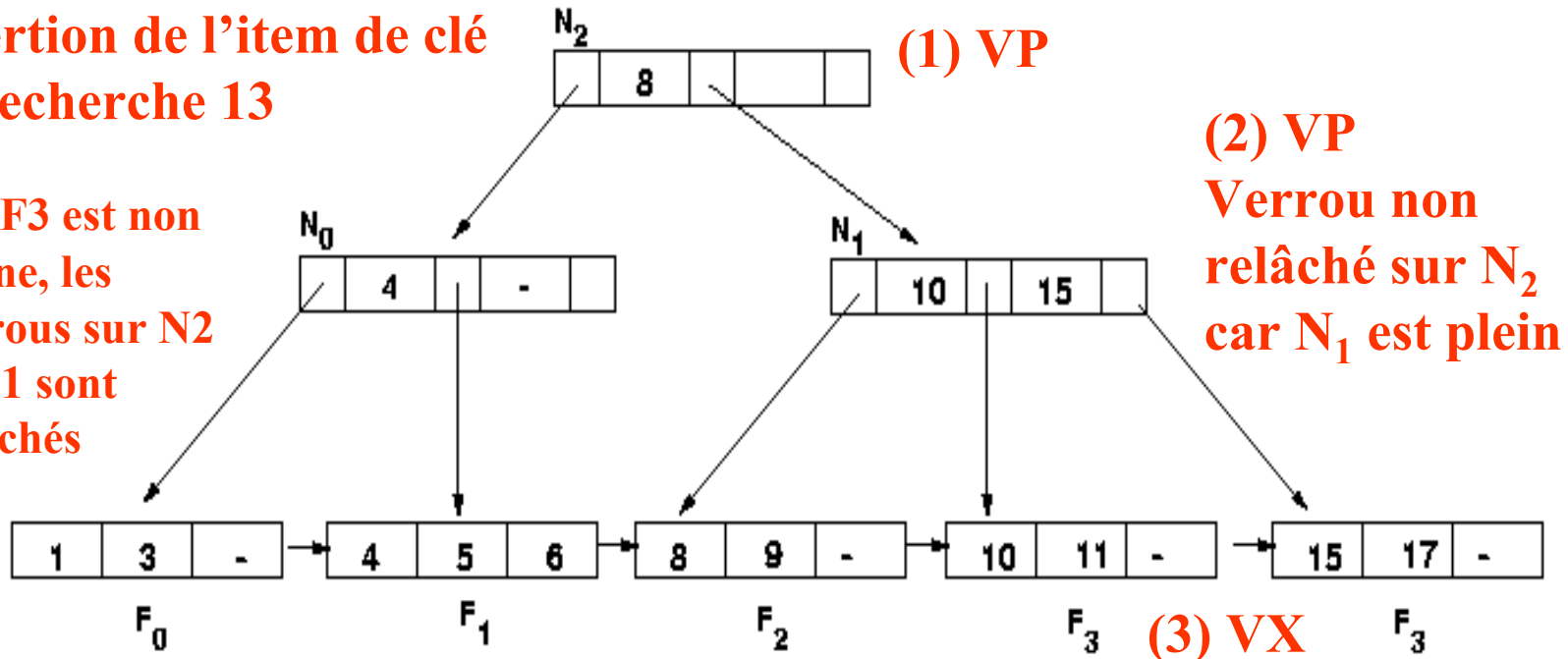


# Concurrency dans un arbre B+

- 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

(4) F3 est non pleine, les verrous sur N2 et N1 sont relâchés





# Transactions et SQL2

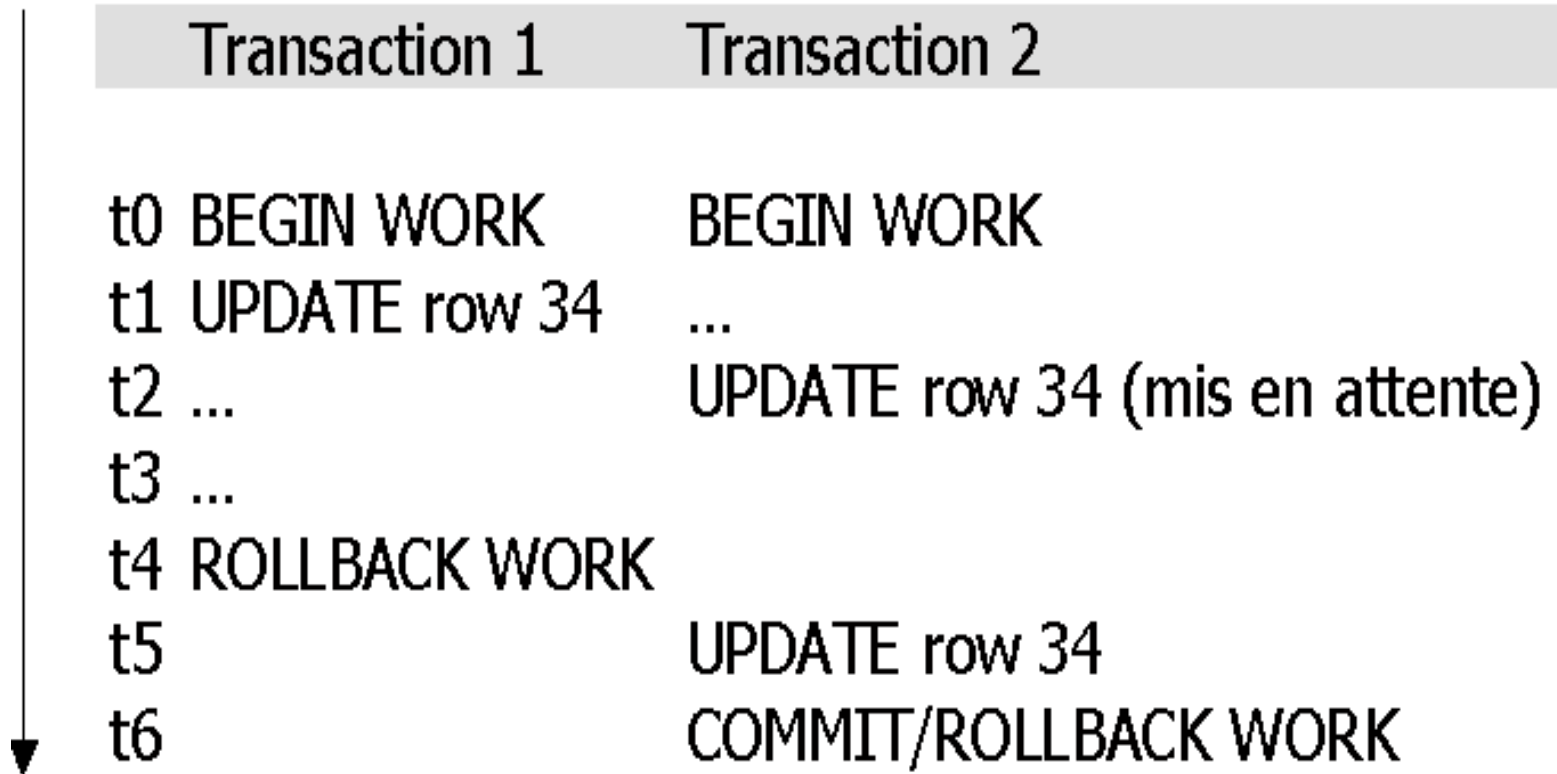
- Une transaction commence dès la 1ère requête ou tout de suite après un *COMMIT* ou un *ROLLBACK*
- Propriétés *READ ONLY* ou *READ WRITE*
- Degrés d'isolation

Degré	Lecture impropre	Lecture non reproductible	Références fantômes
<b>READ UNCOMMITTED</b>	<b>OUI</b>	<b>OUI</b>	<b>OUI</b>
<b>READ COMMITTED</b>	<b>NON</b>	<b>OUI</b>	<b>OUI</b>
<b>REPEATABLE READ</b>	<b>NON</b>	<b>NON</b>	<b>OUI</b>
<b>SERIALIZABLE</b>	<b>NON</b>	<b>NON</b>	<b>NON</b>

- *SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY*

# Exemple de PostgreSQL

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



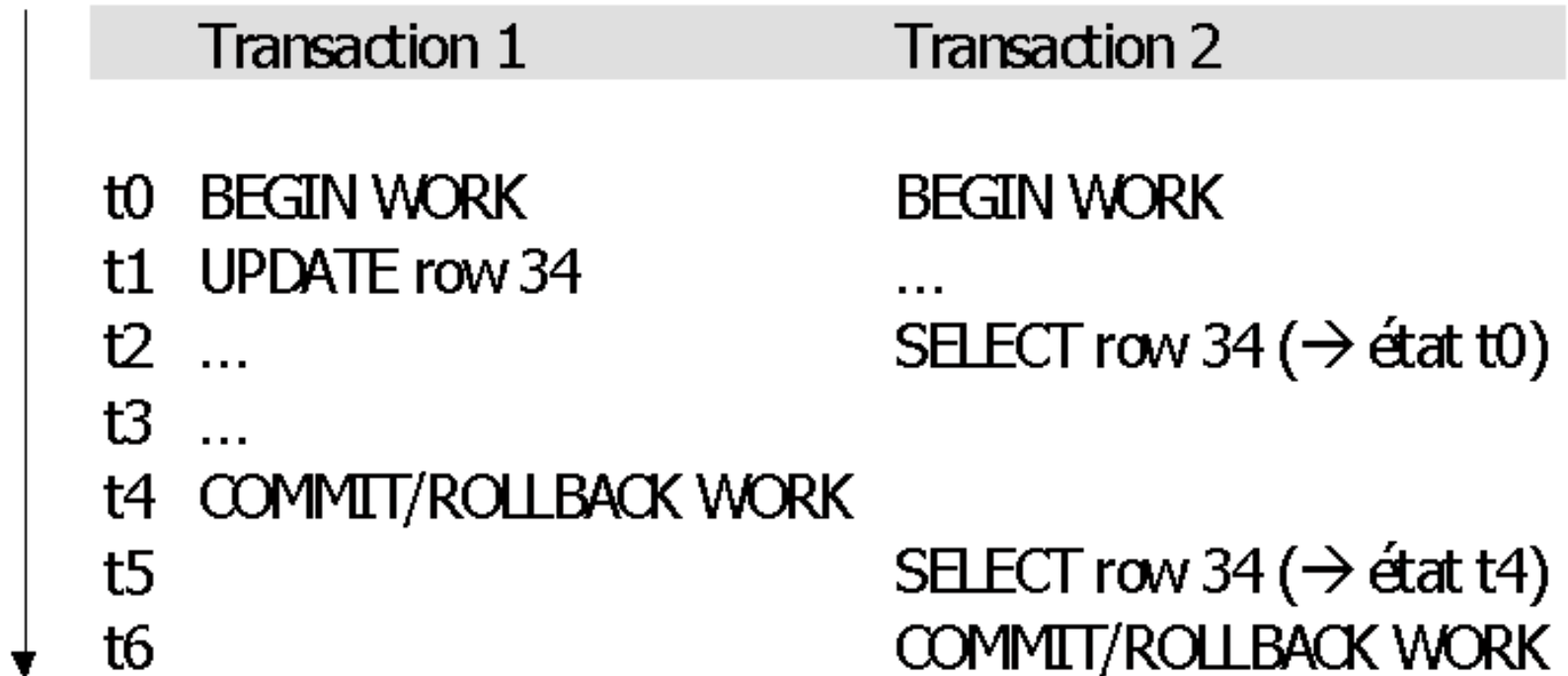
# Exemple de PostgreSQL

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

	Transaction 1	Transaction 2
t0	BEGIN WORK	BEGIN WORK
t1	UPDATE row 34	...
t2	...	UPDATE row 34 (mis en attente)
t3	...	
t4	COMMIT WORK	
t5		UPDATE row 34 ( <b>Ré-exécute la</b> condition de recherche)
t6		COMMIT/ROLLBACK WORK

# 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

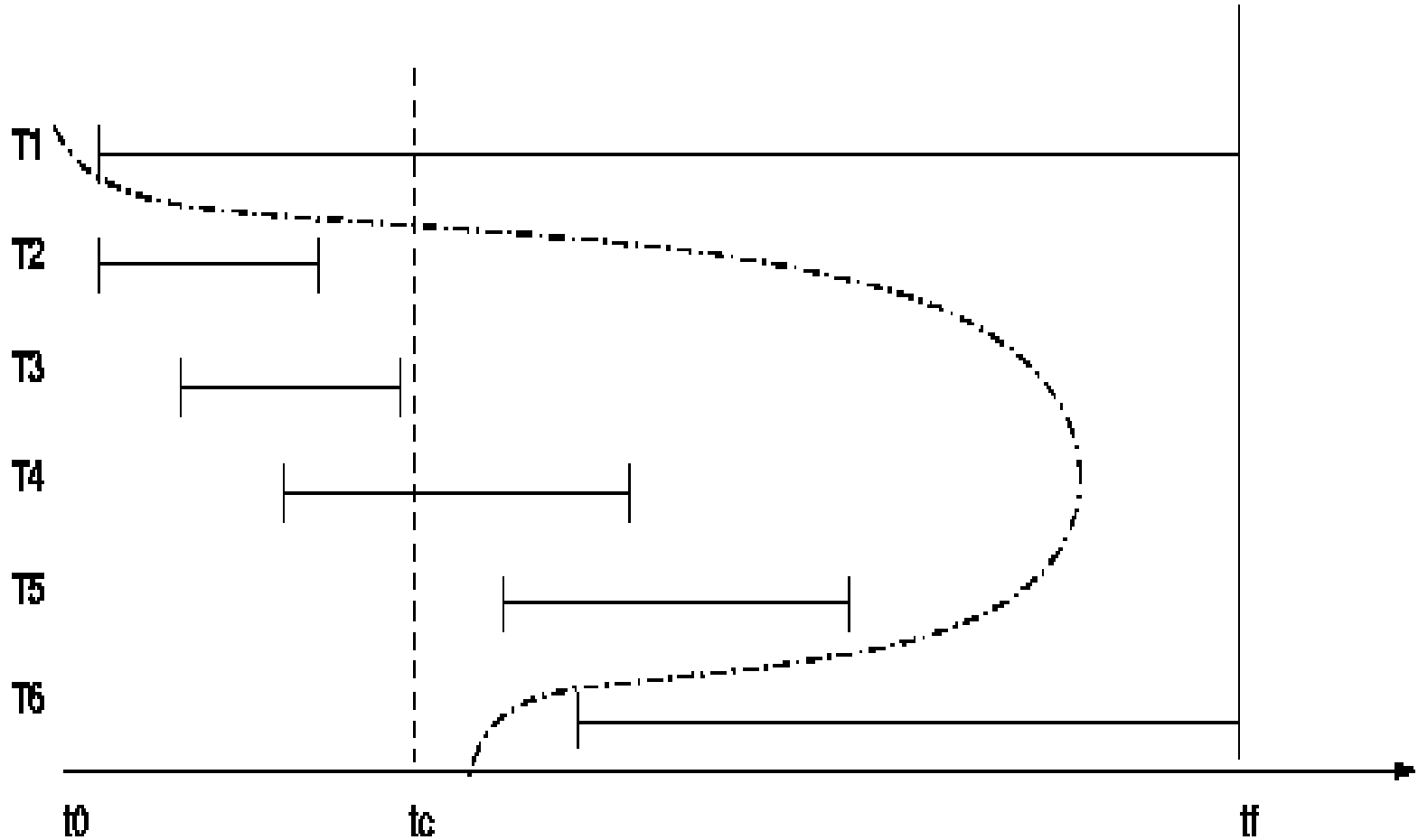
# **Chap. V - Reprise après panne**

- **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

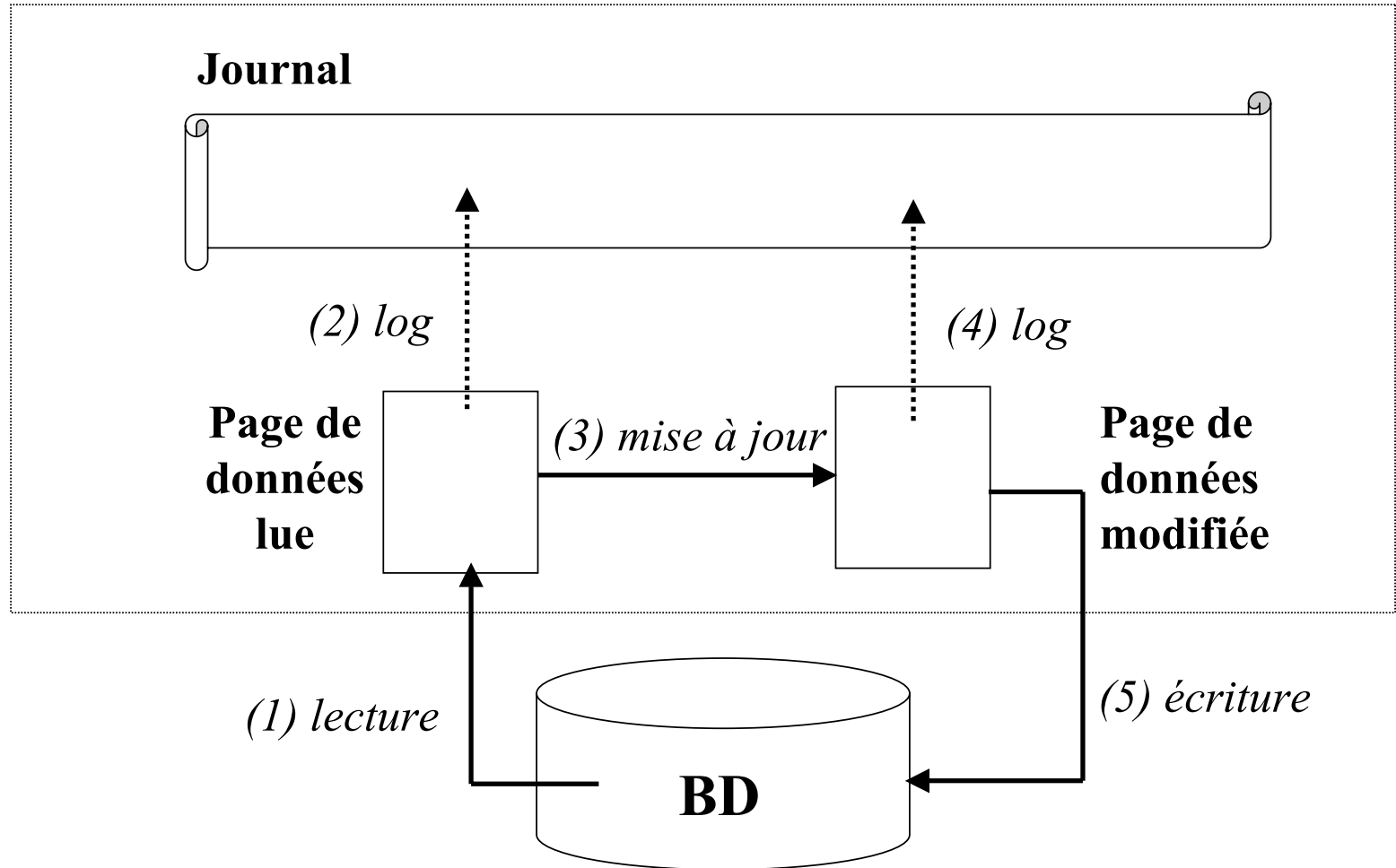
- **Journal des images après (*redo log*)**

- ♦ 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

- **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
- ♦ 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**