

Licence MI2E - 3ème année  
Mention Informatique  
et Mention Mathématiques Mineure Informatique

# **Notions d'architecture des SGBD**

**Maude Manouvrier**

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

# BIBLIOGRAPHIE

## Ouvrages de référence utilisés pour le cours :

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

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

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

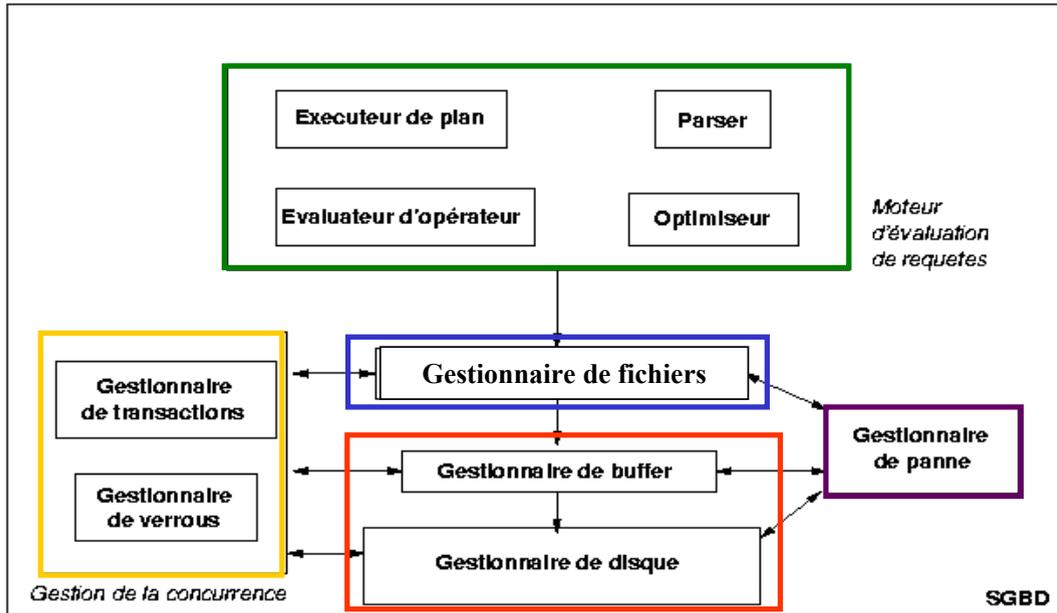
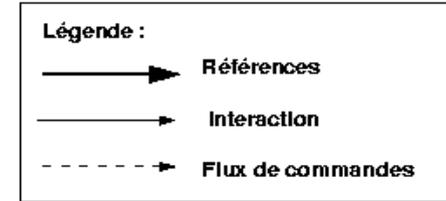
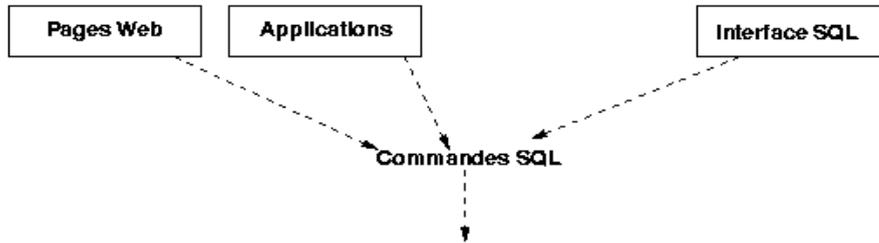
**T. Connolly, C. Begg et A. Strachan, *Database Systems A Pratical Approach to Desigh, Implementation and Management*, 1998, disponible à la BU 055.7 CON**

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

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

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

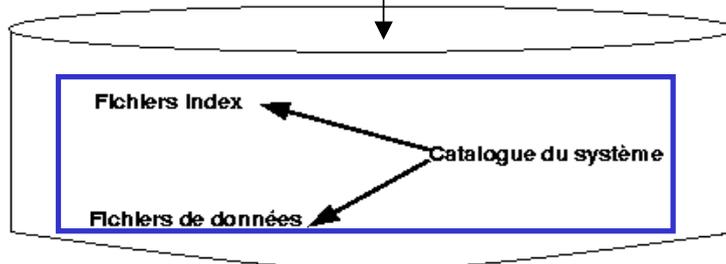
**G. Gardarin, *Bases de Données - objet/relationnel*, Eyrolles, 1999, disponible à la BU 005.74 GAR + *Le client - serveur*, Eyrolles, 1996004.21 GAR**



Chapitre 3

Chapitre 4

Chapitre 1 Chapitre 5



Chapitre 2

# Chap. I - Architecture d'un SGBD

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

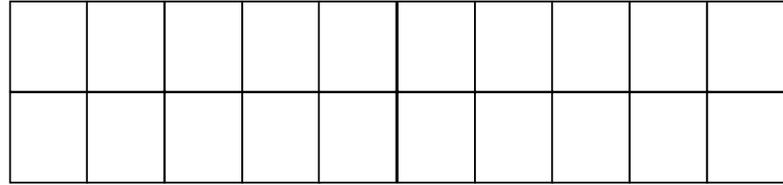
# Gestionnaire de buffer

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

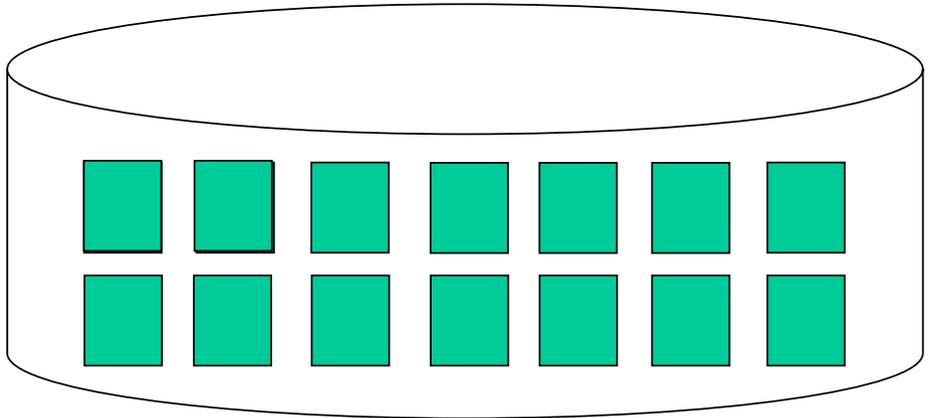
- Politique de remplacement (ex. LRU)
- Gestion des pages mises à jour
- Partition de la mémoire
- Vérification des droits sur les pages

# Gestionnaire de buffer

Mémoire



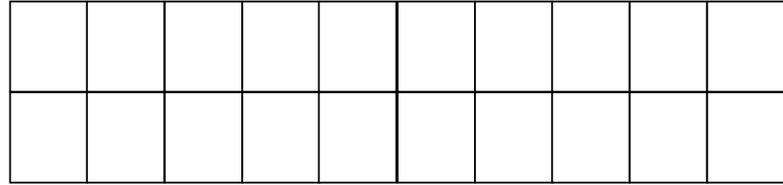
Disque



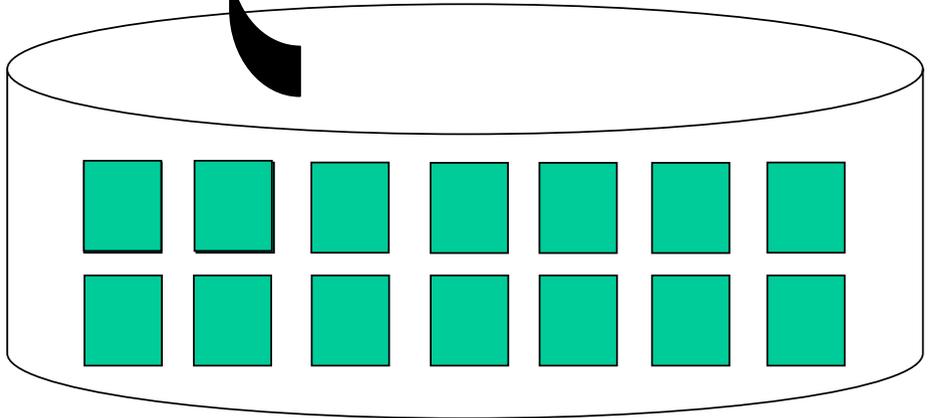
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

# Gestionnaire de buffer

Mémoire



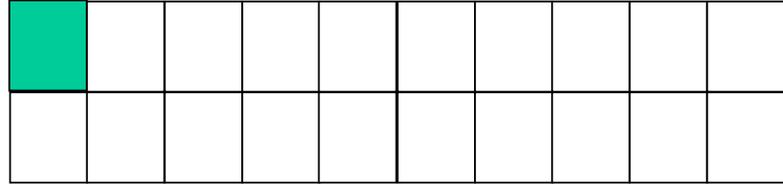
Disque



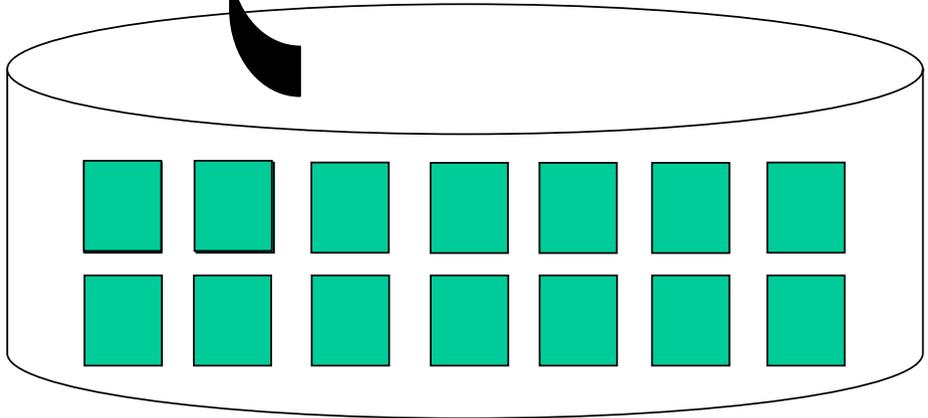
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

# Gestionnaire de buffer

Mémoire



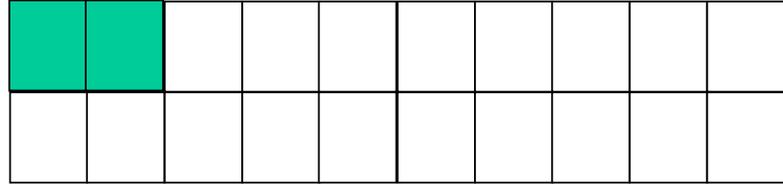
Disque



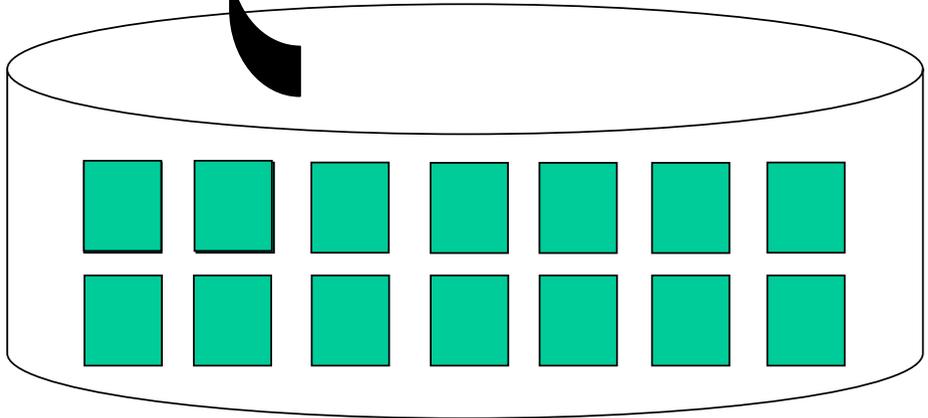
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

# Gestionnaire de buffer

Mémoire



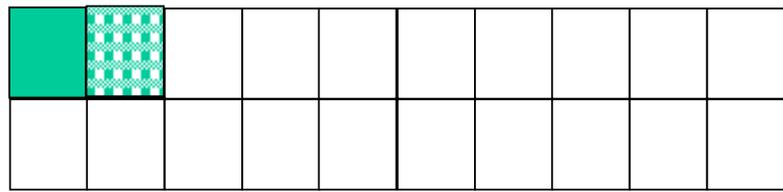
Disque



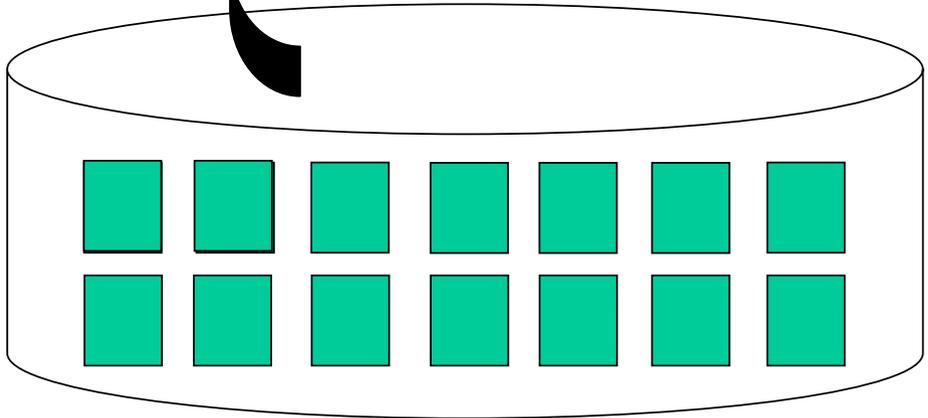
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

# Gestionnaire de buffer

Mémoire



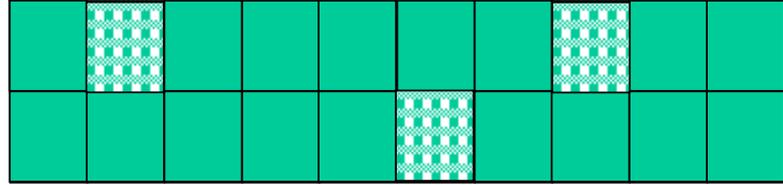
Disque



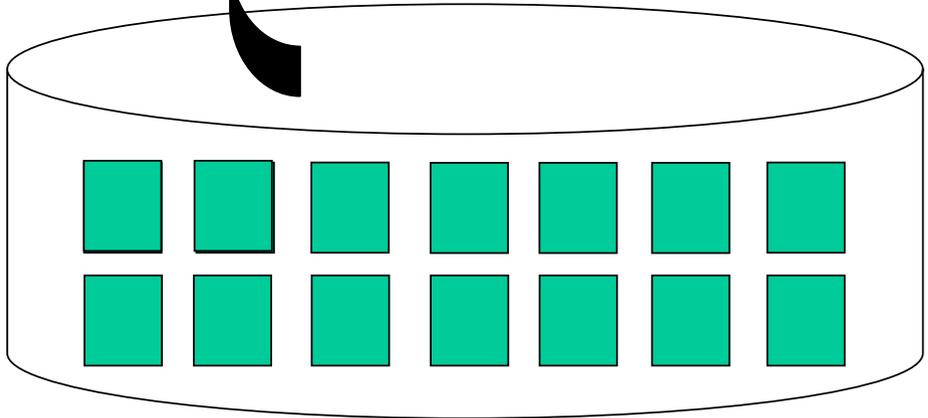
-  Page mémoire libre
-  Page de données
-  Page de données modifiées

# Gestionnaire de buffer

Mémoire



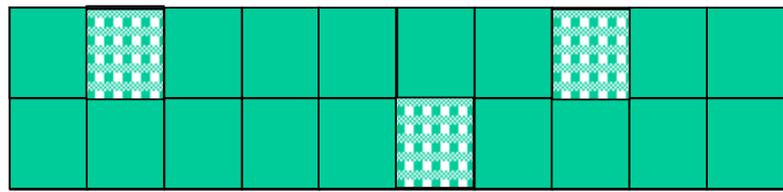
Disque



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

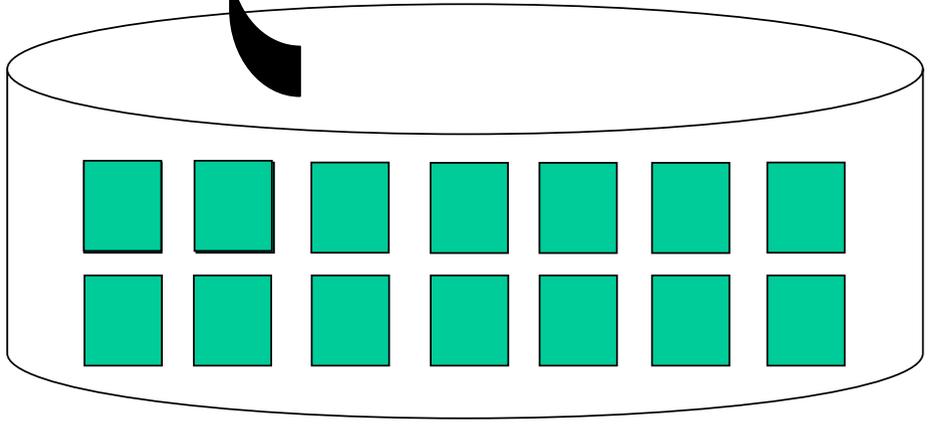
# Gestionnaire de buffer

Mémoire



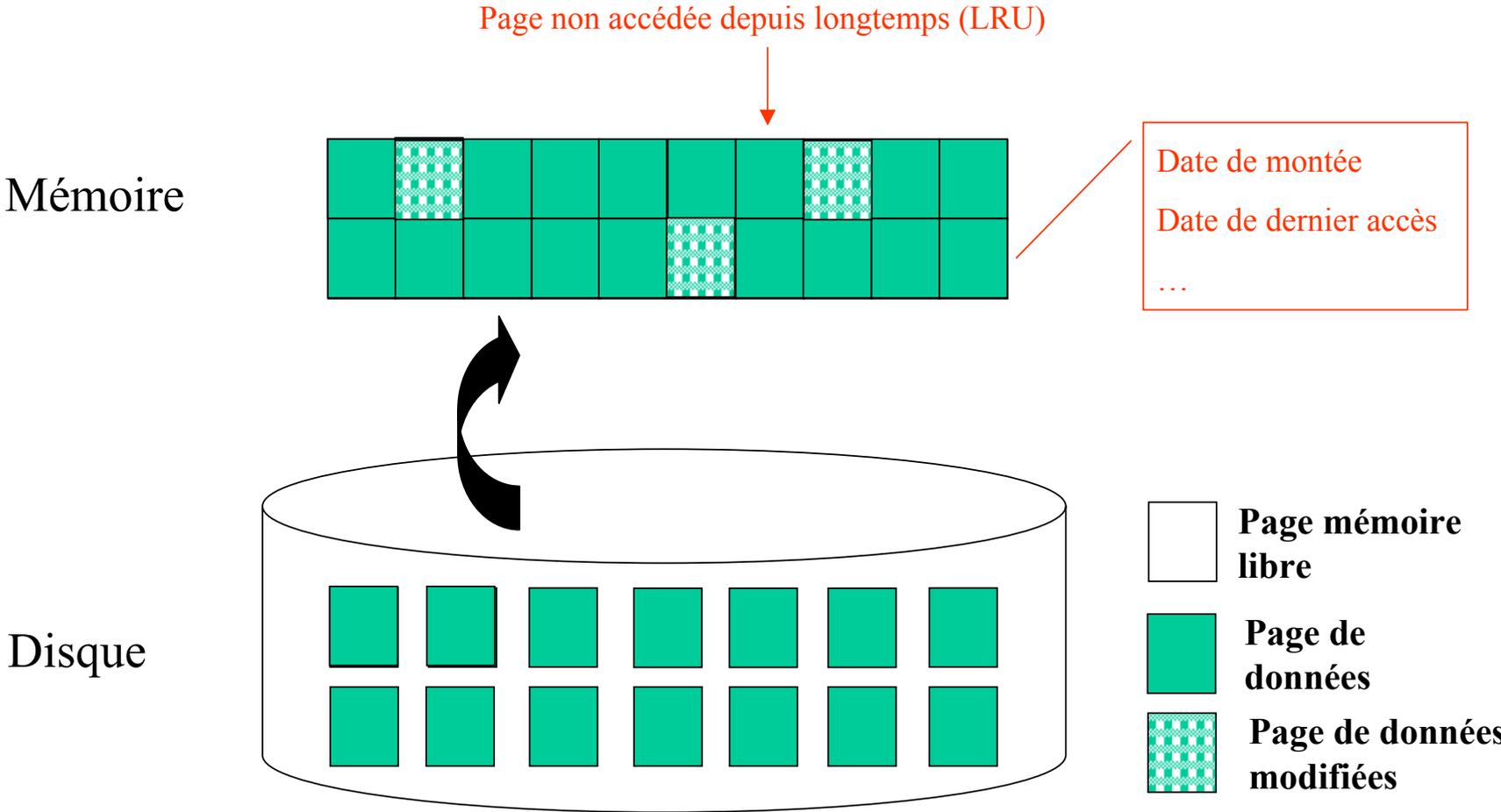
Date de montée  
Date de dernier accès  
...

Disque



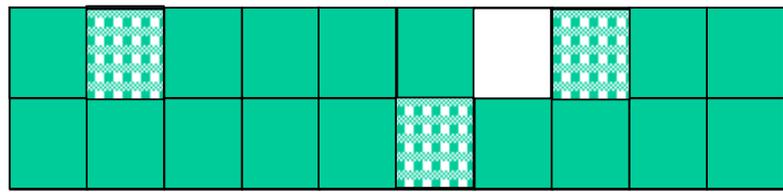
-  Page mémoire libre
-  Page de donnée
-  Page de données modifiées

# Gestionnaire de buffer



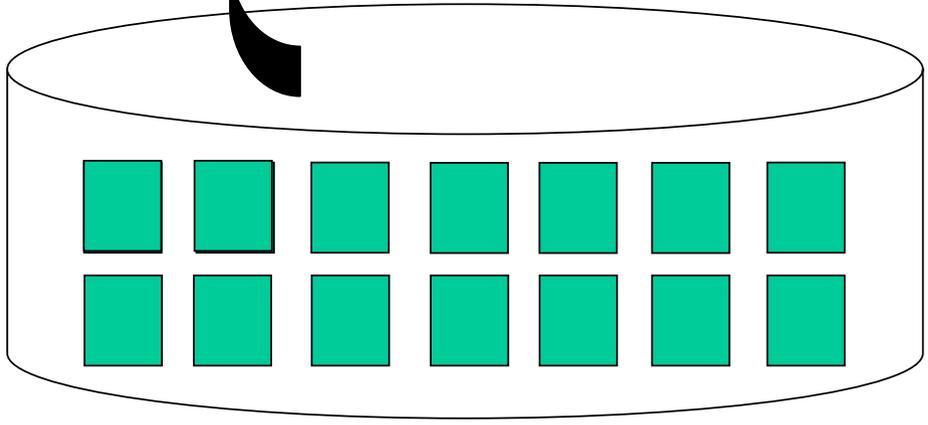
# Gestionnaire de buffer

Mémoire



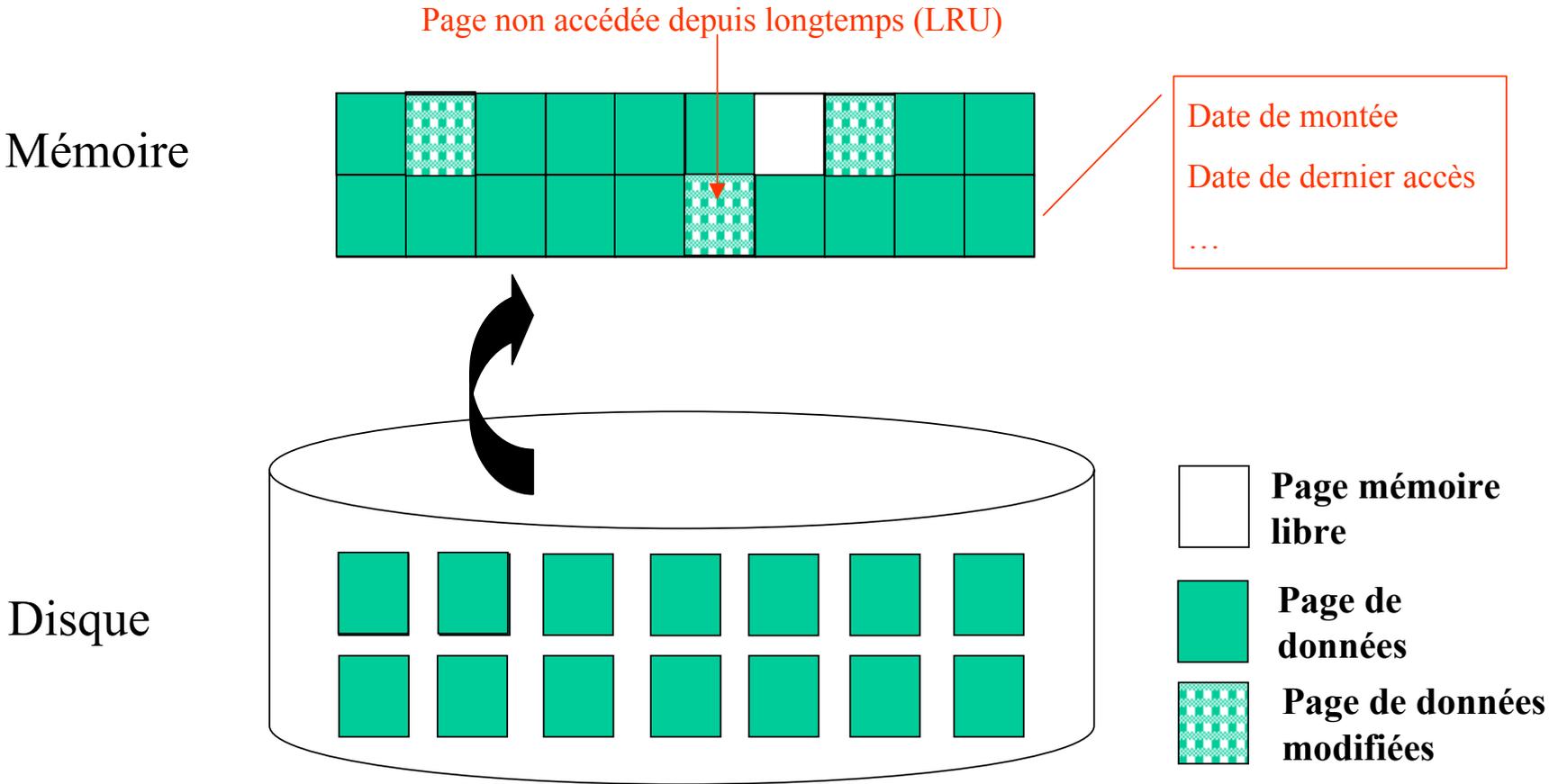
- Date de montée
- Date de dernier accès
- ...

Disque

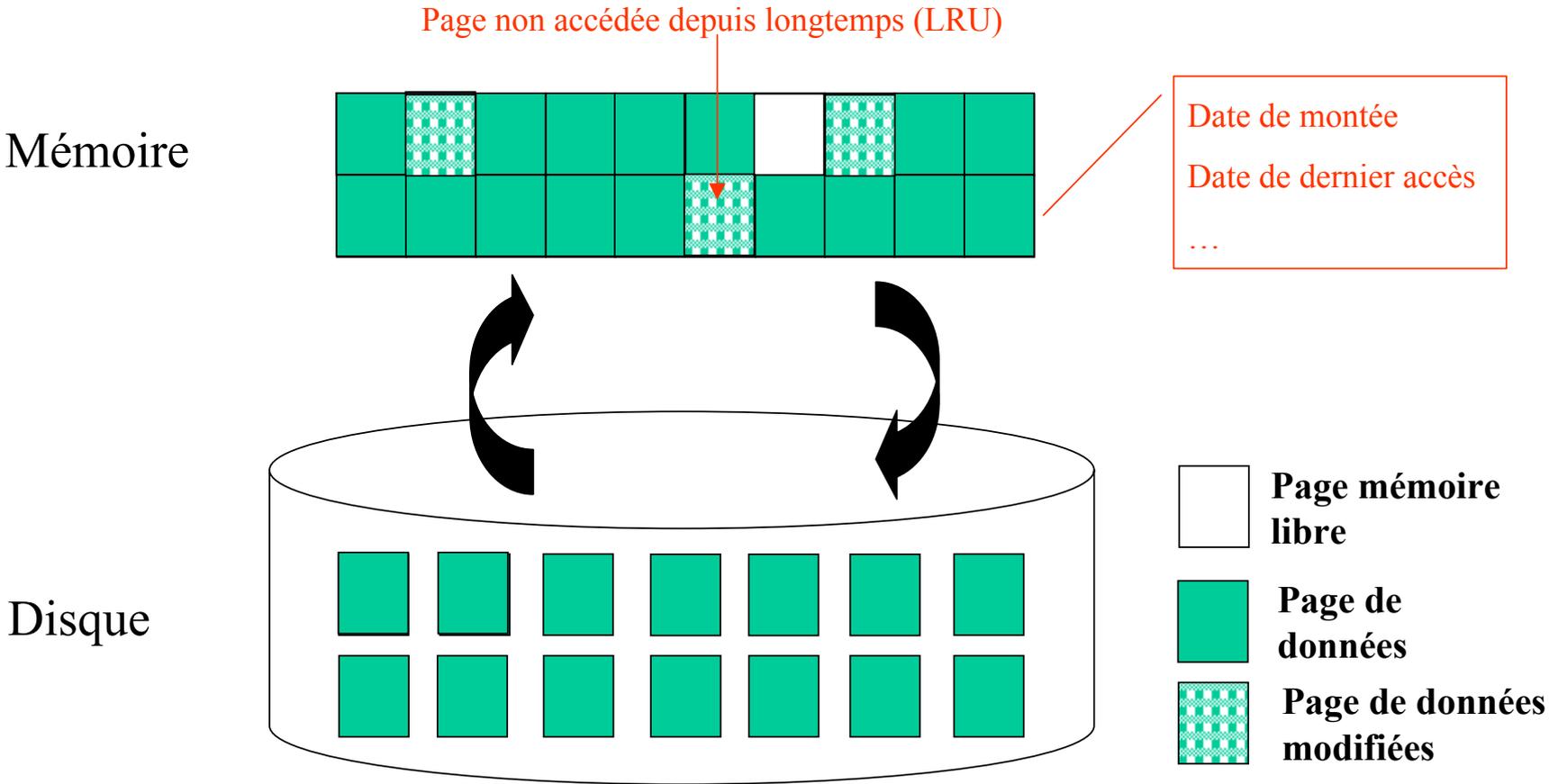


- Page mémoire libre
- Page de données
- Page de données modifiées

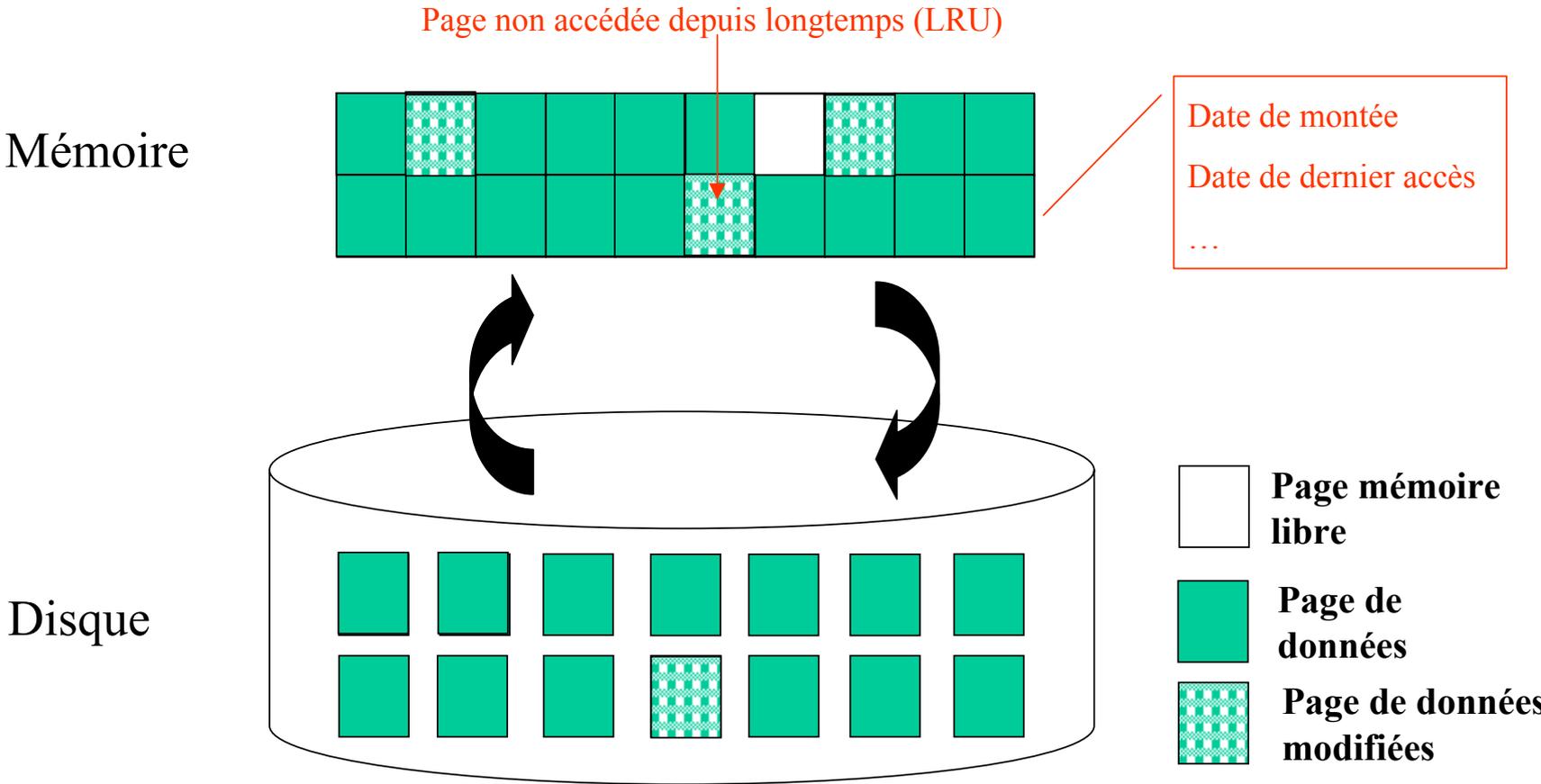
# Gestionnaire de buffer



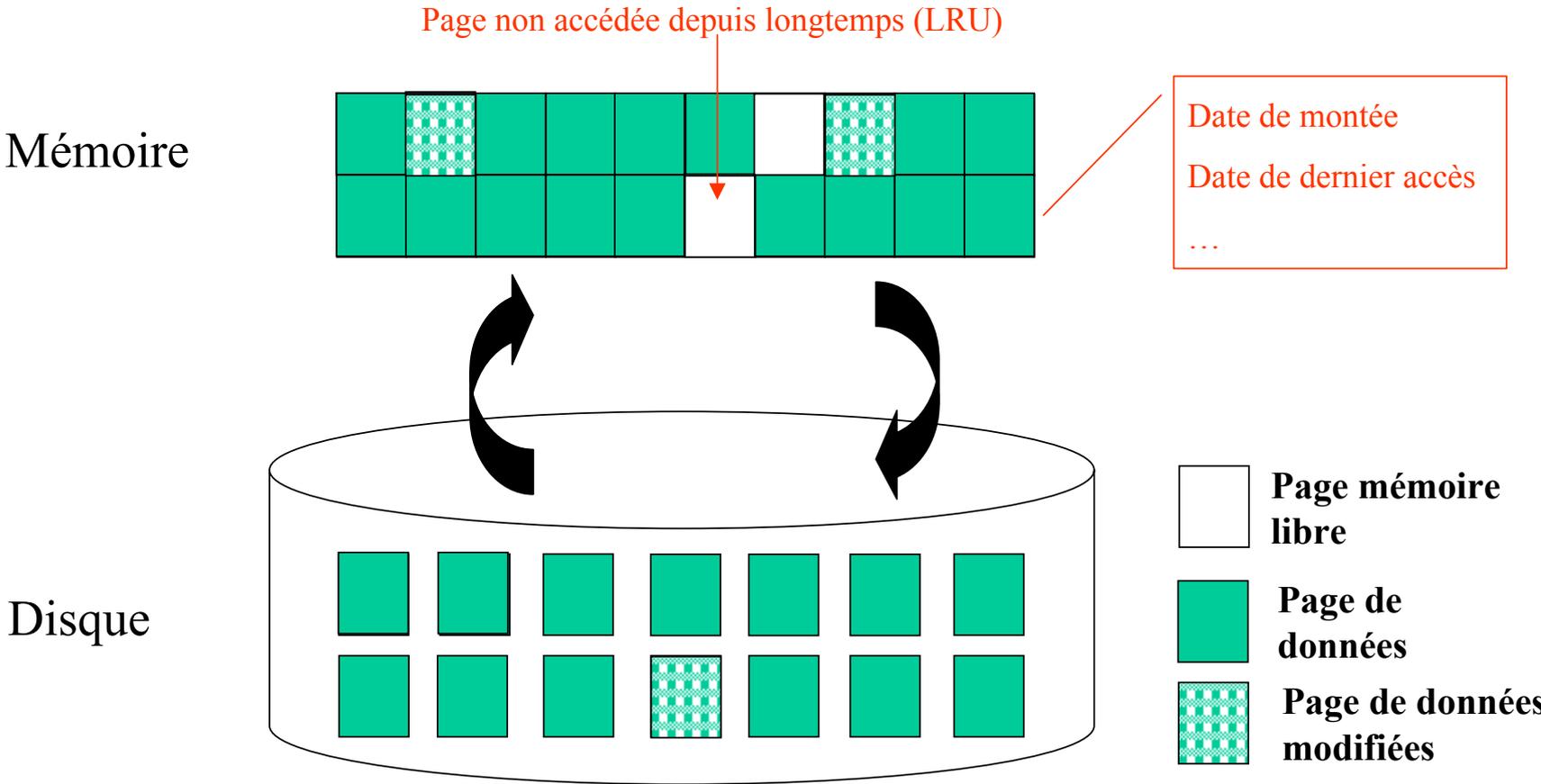
# Gestionnaire de buffer



# 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**

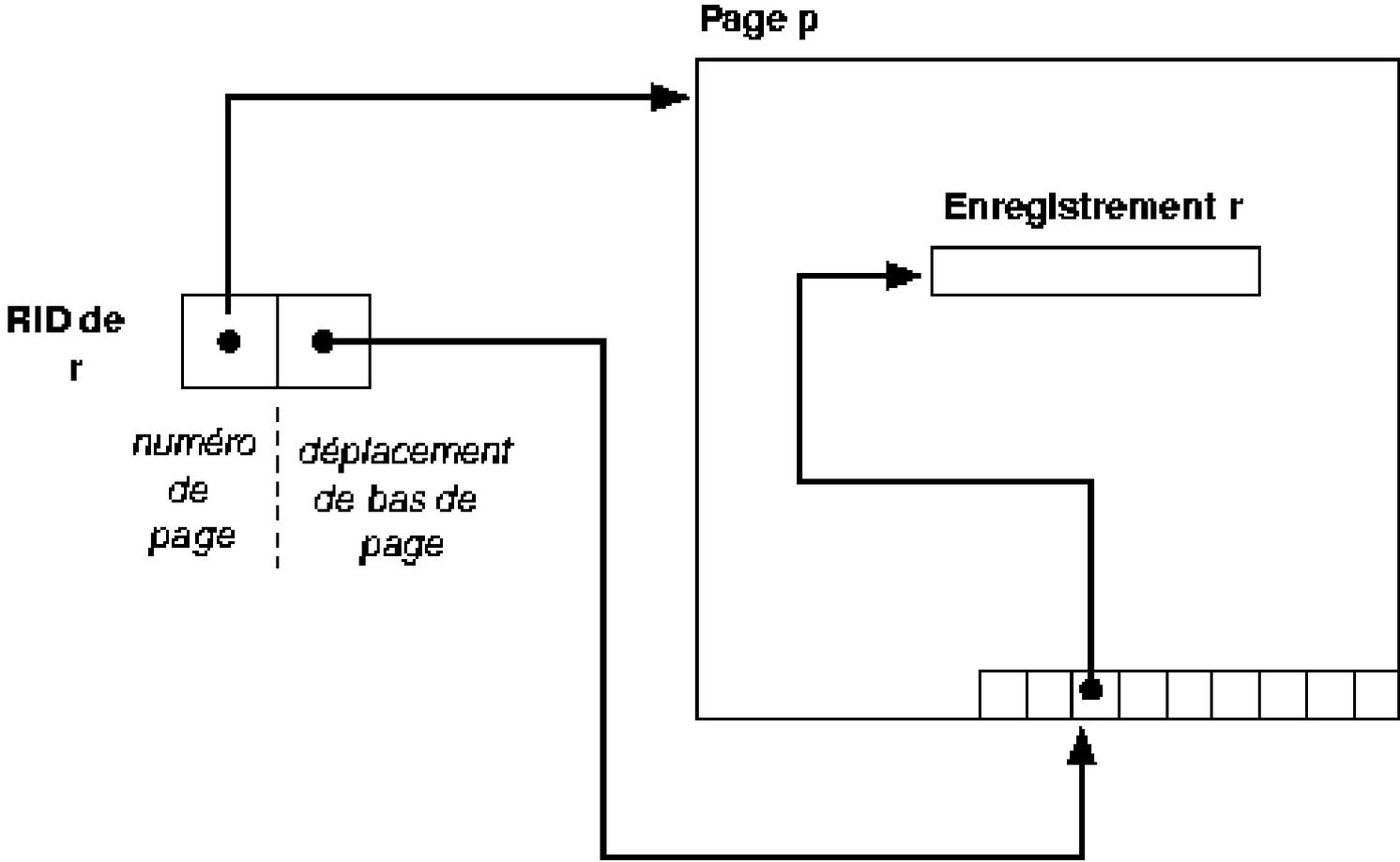
# Chap. II - Organisation des données

- 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



# 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**

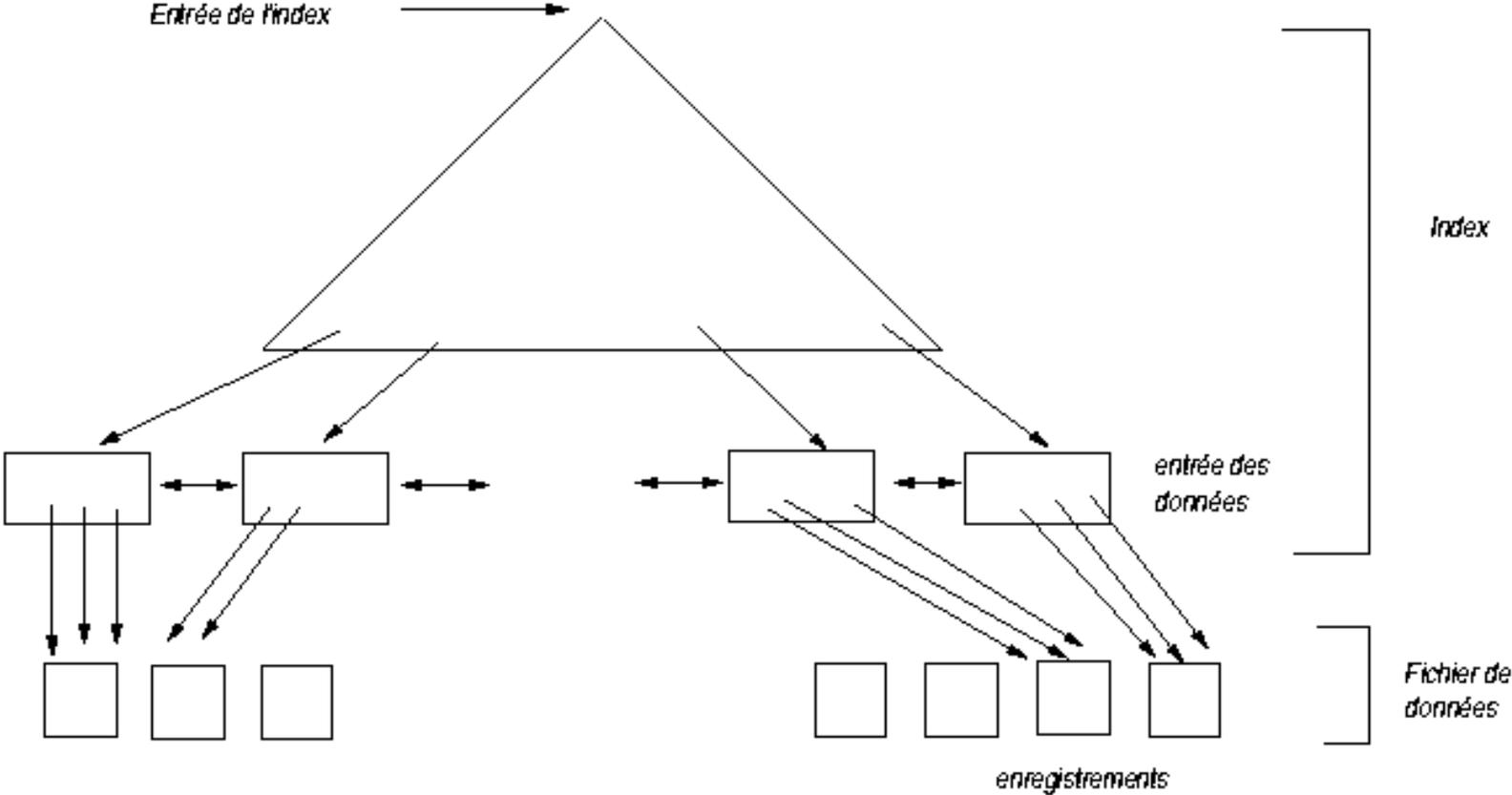
- *(clé de recherche, valeur(s) de clé primaire)*

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

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

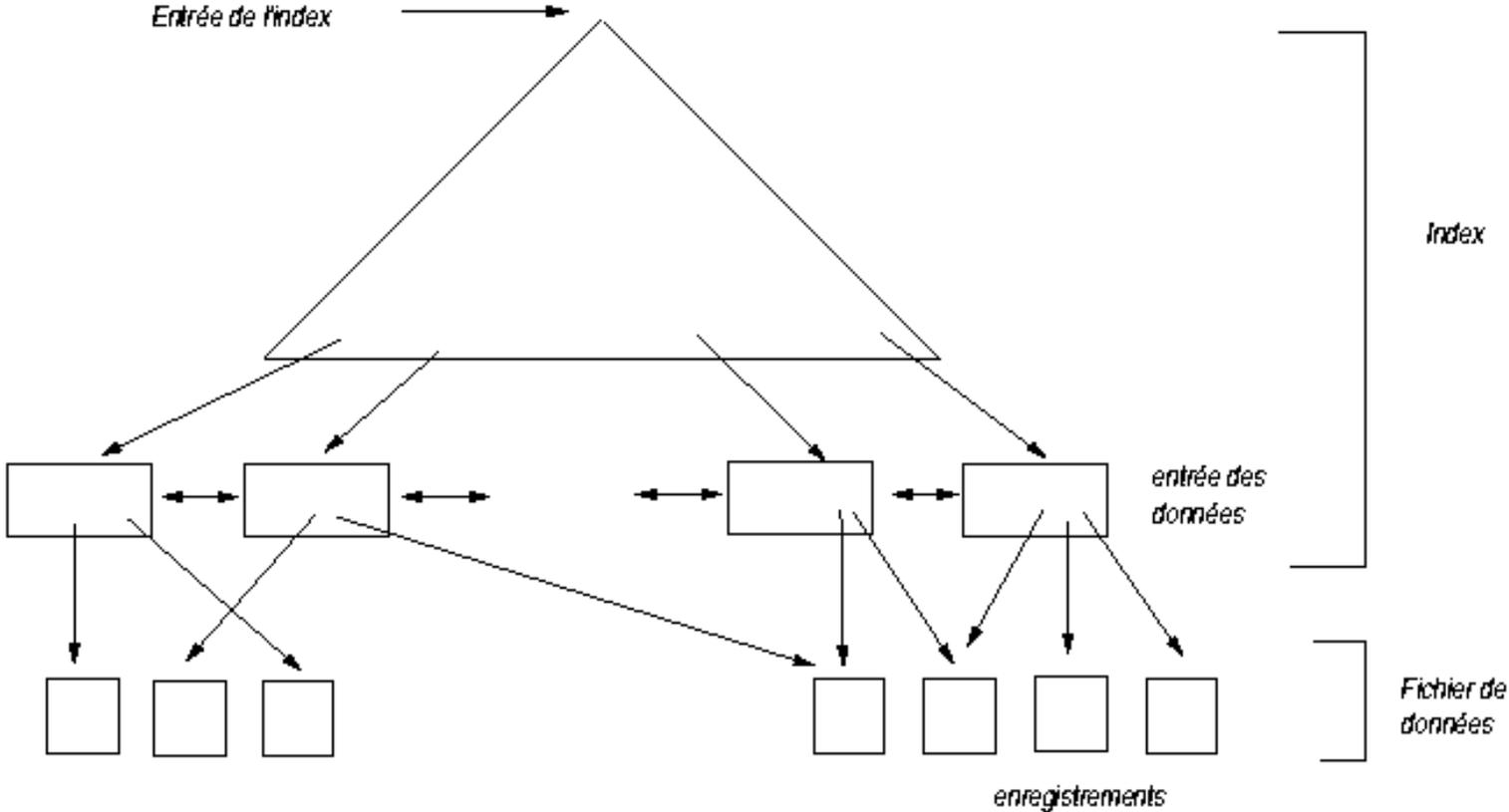
# Index (2/4)

- **Clustered index**



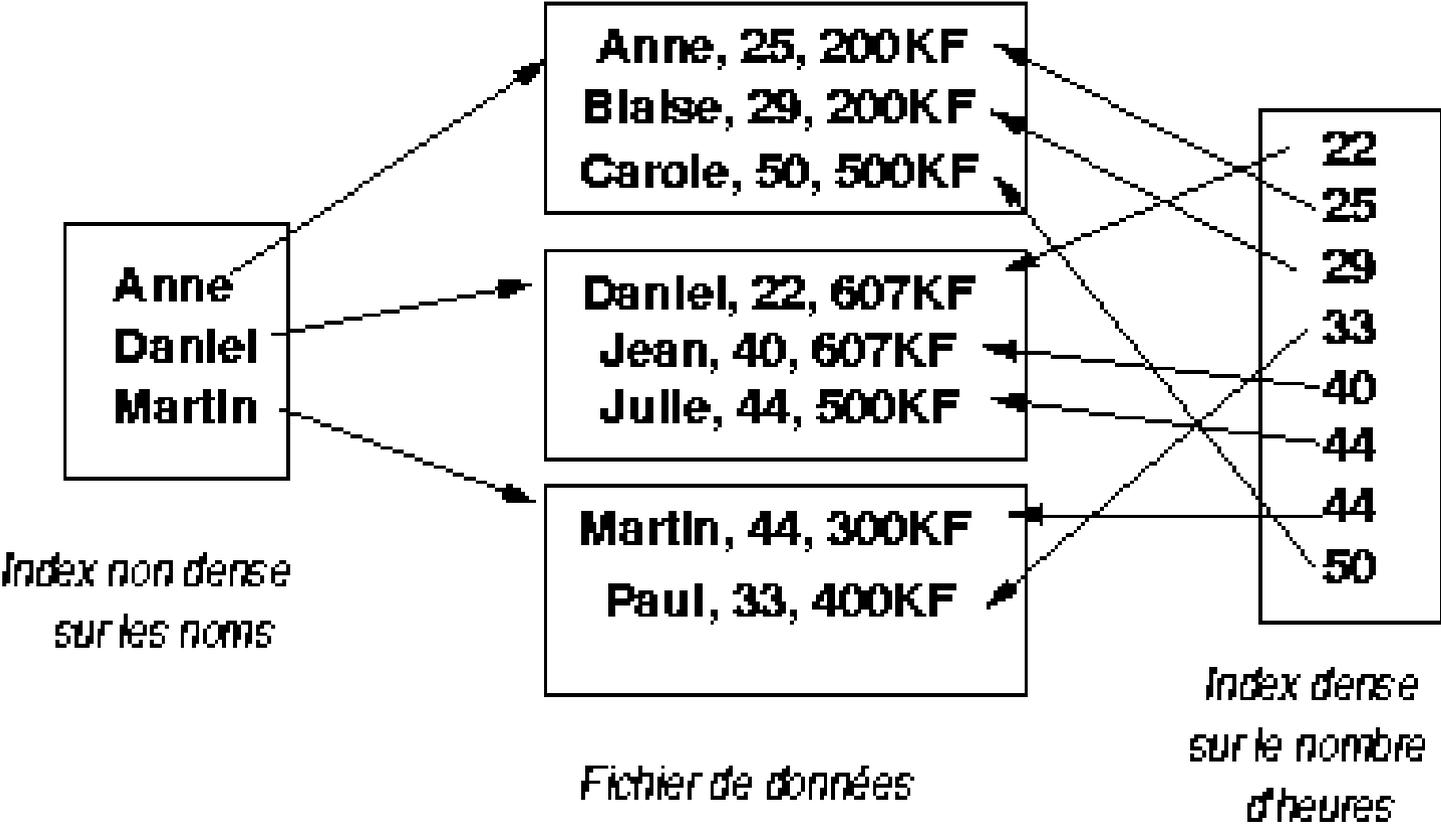
# Index (3/4)

- Unclustered index



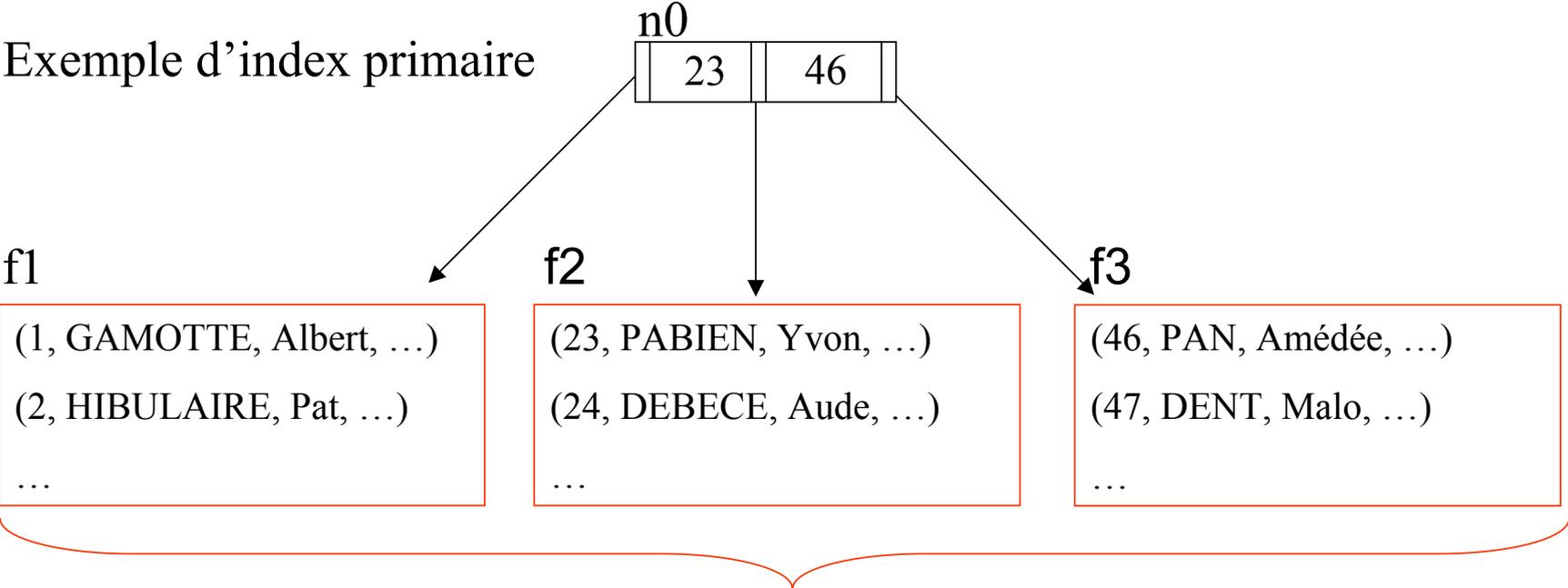
# Index (4/4)

- Index dense / non dense (*sparce*)





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

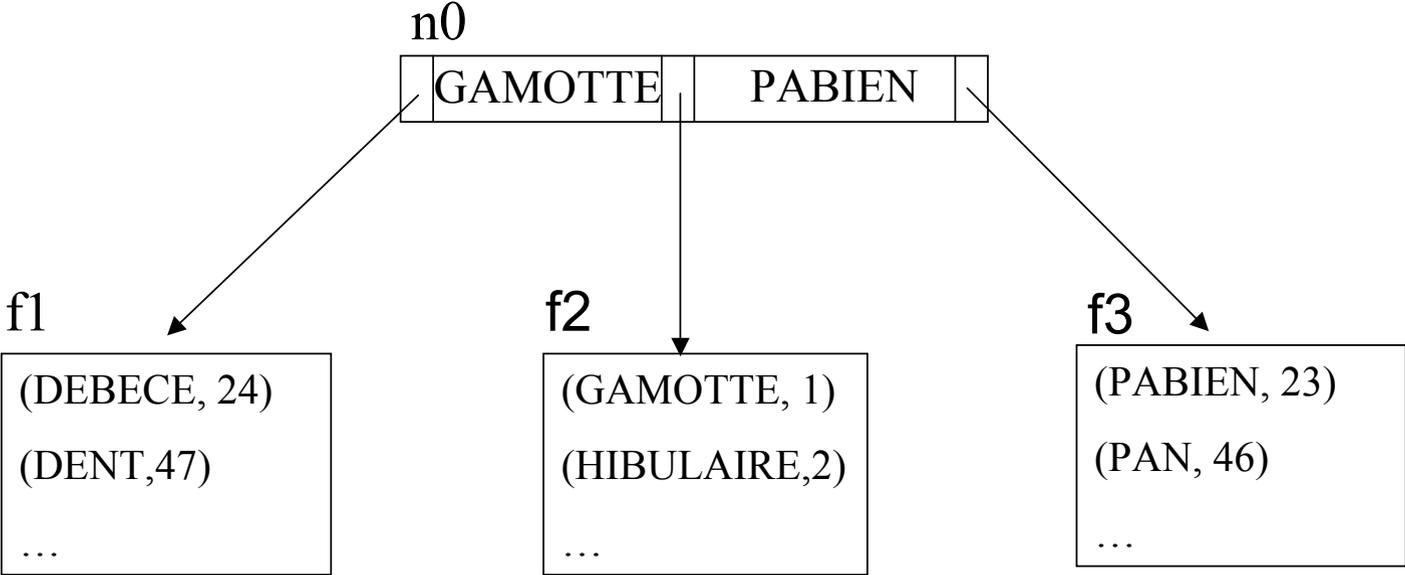


Pages de la relation et feuilles de l'index



# Arbre B+ (2/3)

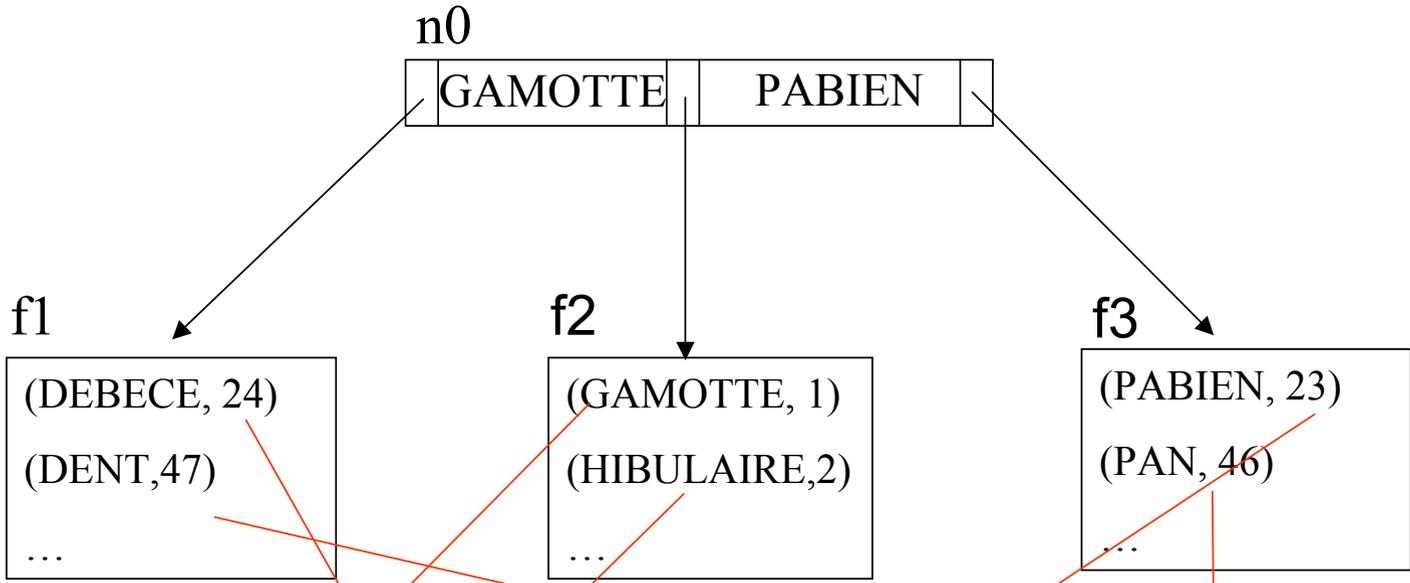
Exemple d'index secondaire



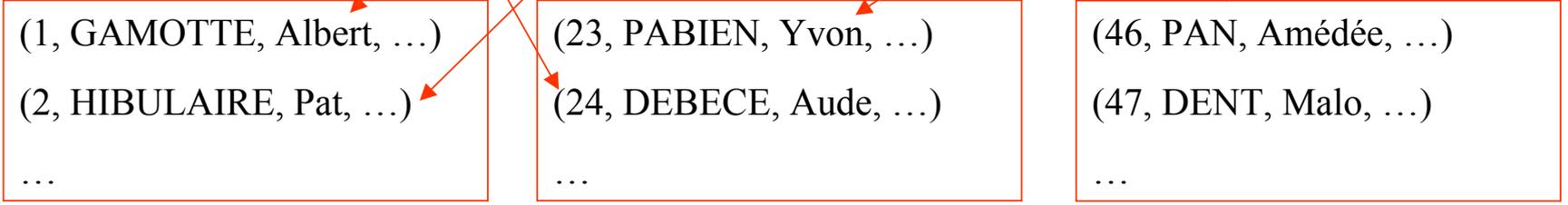


# Arbre B+ (2/3)

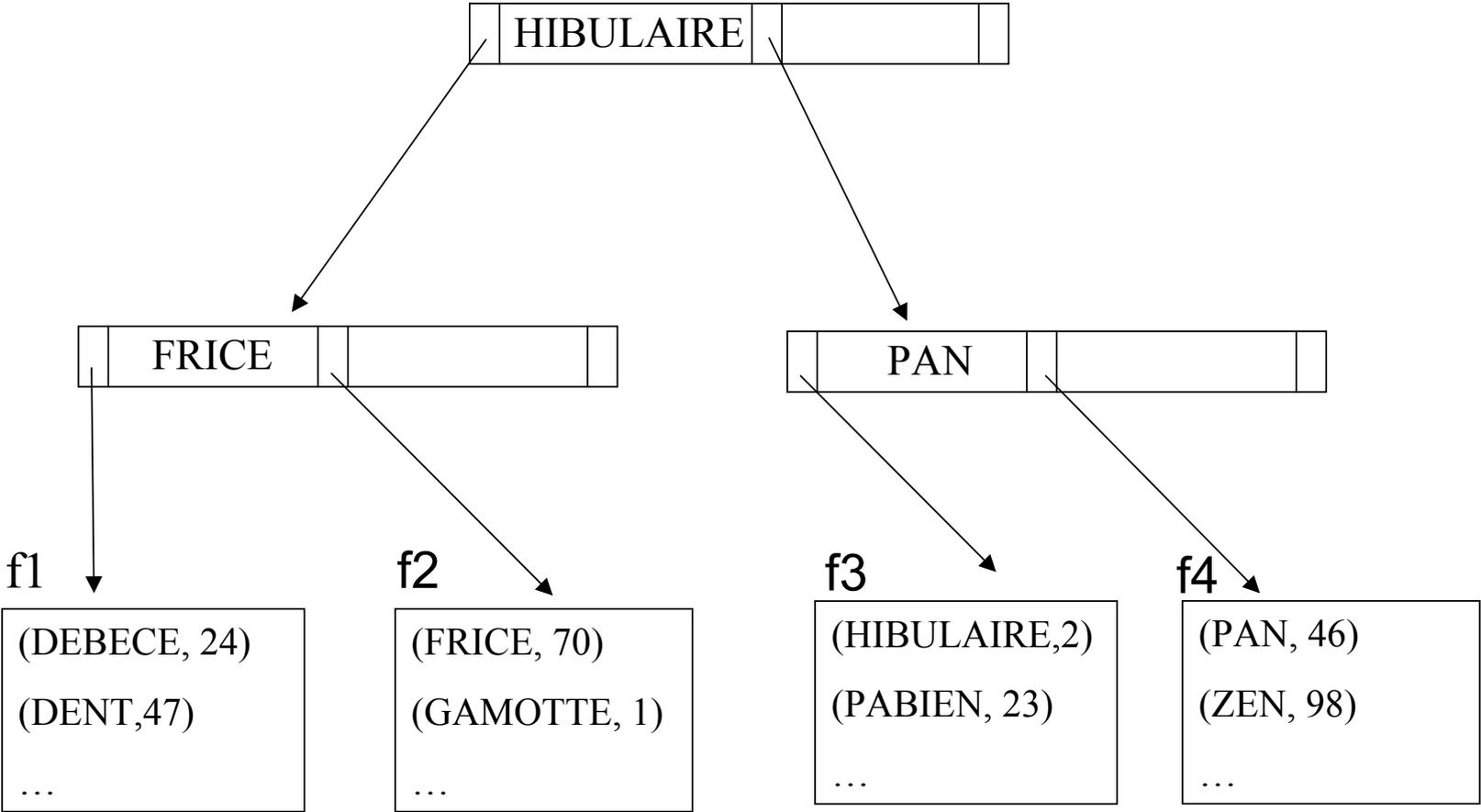
Exemple d'index secondaire



## Pages de la relation



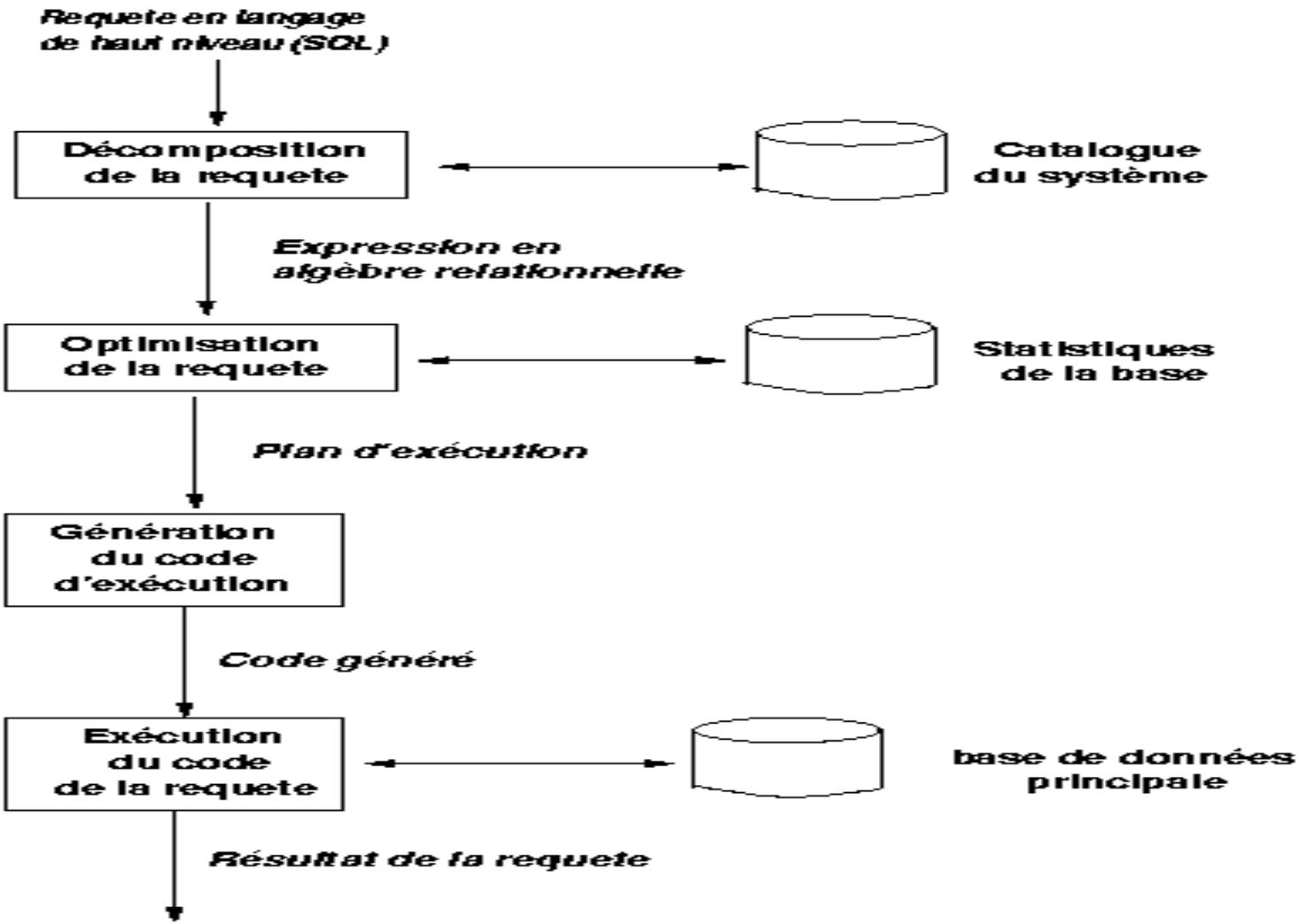
# Arbre B+ (3/3)



# 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**

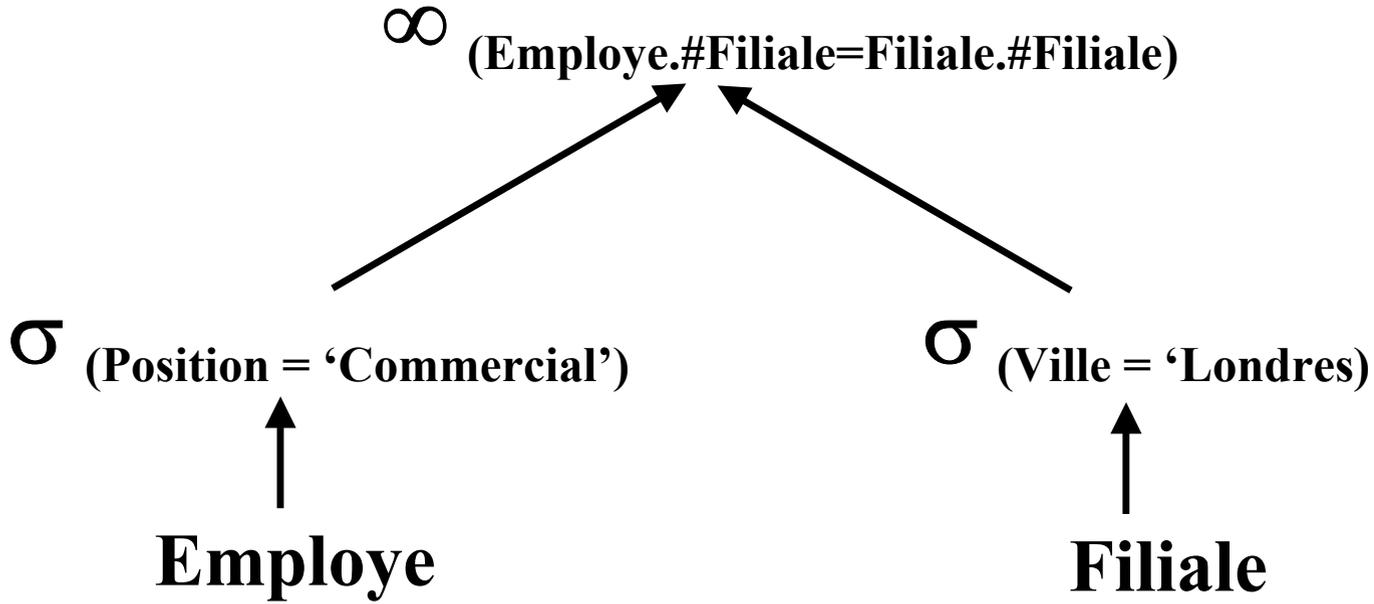
# Catalogue du système (2/2)

- 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

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

### 8) Commutativité des unions et des intersections

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

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

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

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

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

## Equivalence d'expressions (3/3)

### 10) Commutativité des projections et des unions

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

### 11) Associativité des jointures

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

### 12) Associativité des unions et des intersections

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

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

# Transformation d'un arbre algébrique

- ① **Division des conjonctions de sélections**
- ② **Ré-ordonnement des sélections en utilisant les règles 2 et 4**
- ③ **Application des sélections les plus sélectives en premier**
- ④ **Transformation des produits cartésiens en jointure**
- ⑤ **Ré-ordonnement des équi-jointures en utilisant la règle 11**
- ⑥ **Déplacement des projections et création de nouvelles projections en utilisant les règles 4 et 7**

$\Pi_{\text{Nom}}$   
▲

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

$\prod$  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
```

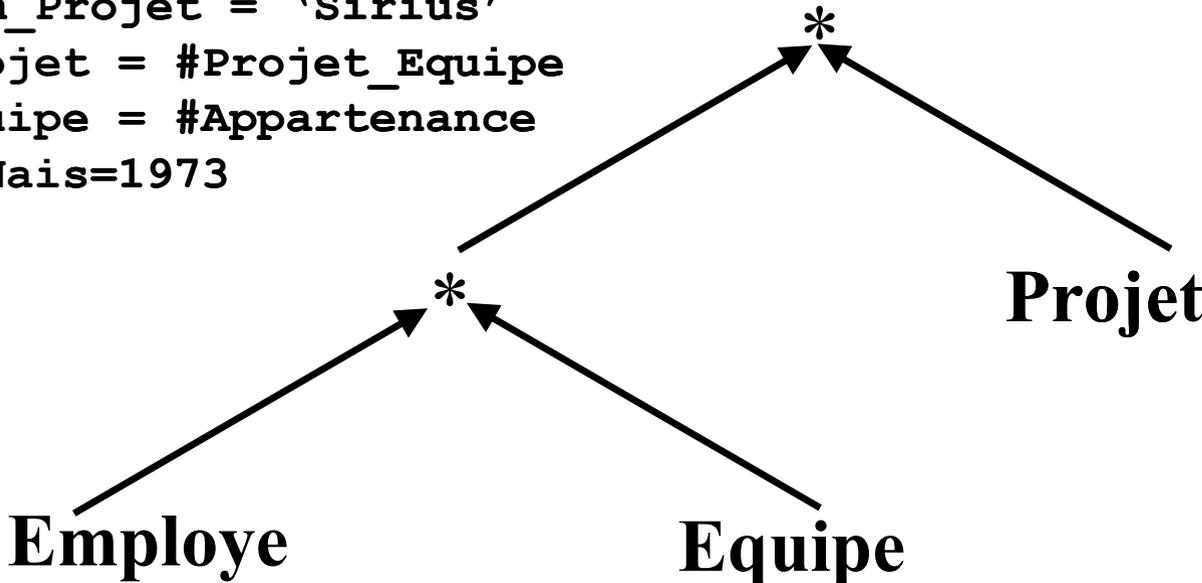
Projet

Employe

Equipe

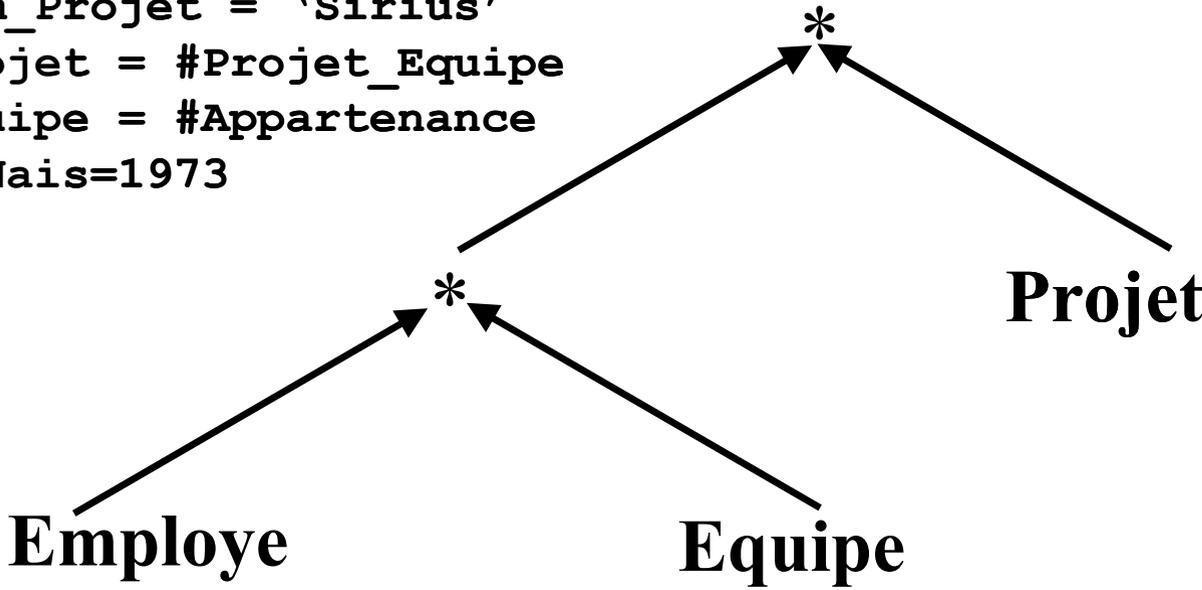
$\Pi_{\text{Nom}}$   
▲

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



$\Pi_{\text{Nom}}$   
▲

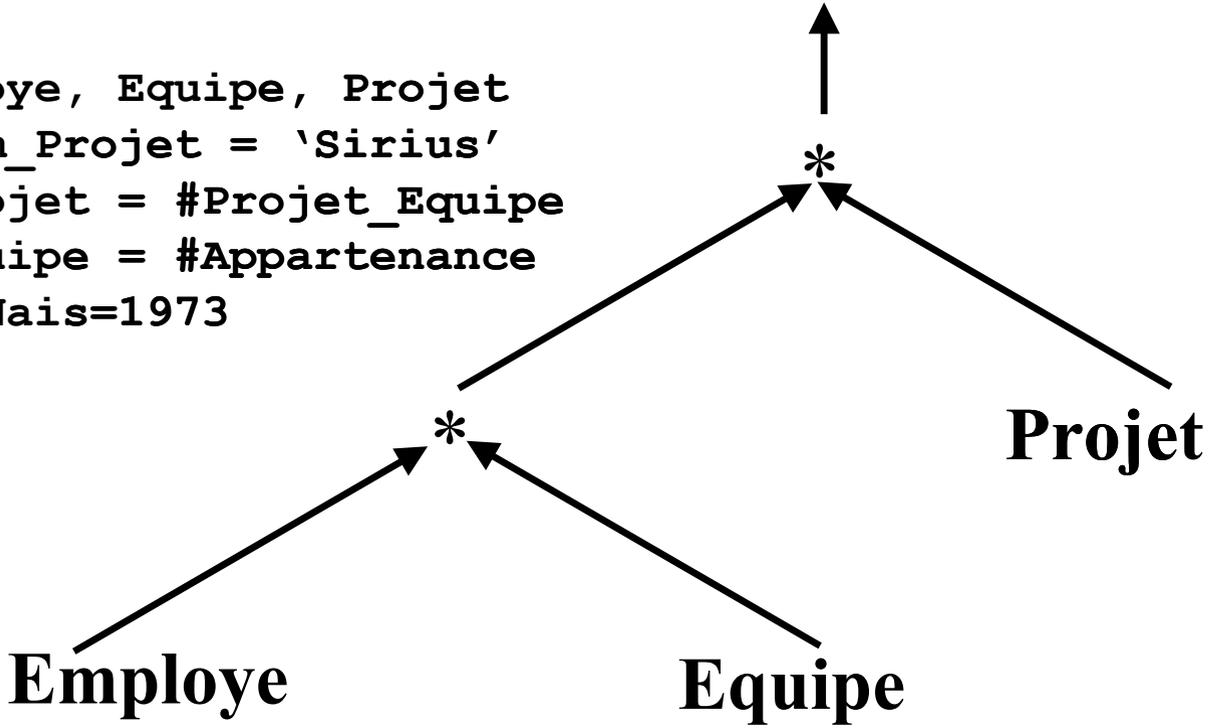
```
SELECT Nom  
FROM Employe, Equipe, Projet  
WHERE Nom_Projet = 'Sirius'  
AND #Projet = #Projet_Equipe  
AND #Equipe = #Appartenance  
AND DaeNais=1973
```



$\Pi_{\text{Nom}}$   
▲

$\sigma_{(\text{Nom\_Projet}=\text{'Siruis'}) \wedge (\#\text{Projet}=\#\text{Projet\_Equipe}) \wedge (\#\text{Equipe}=\#\text{Appartenance}) \wedge (\text{DateNais}=1973)}$

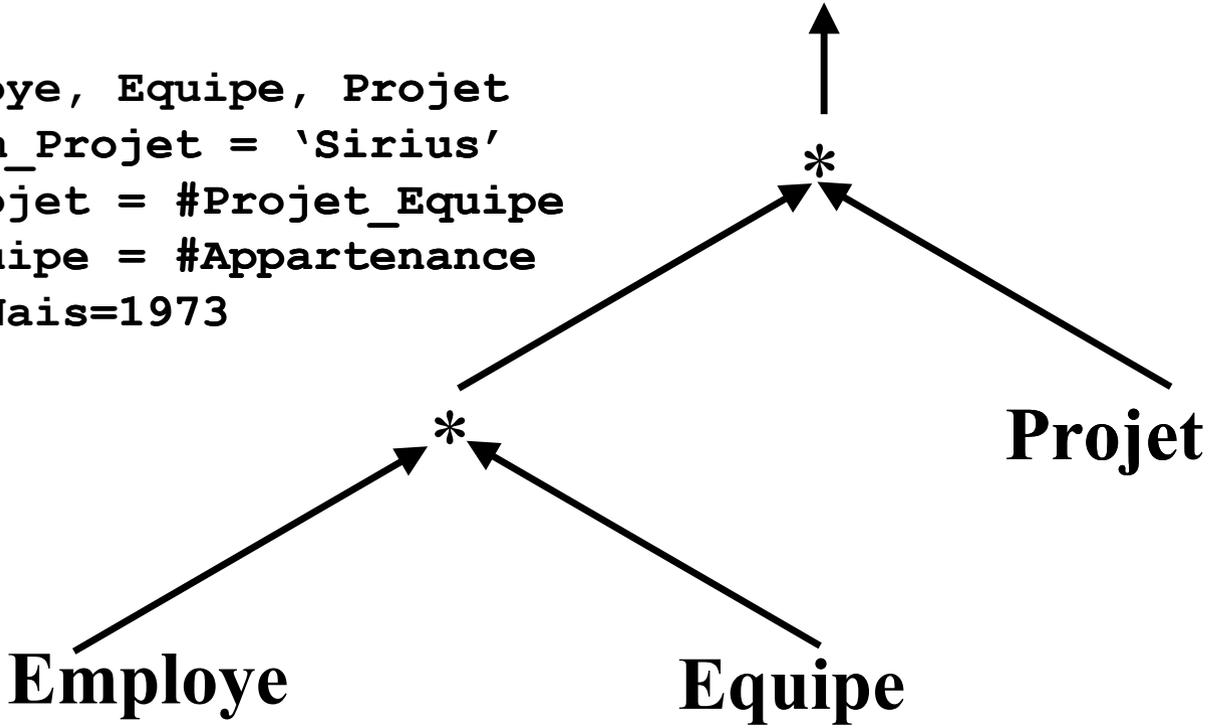
```
SELECT Nom  
FROM Employe, Equipe, Projet  
WHERE Nom_Projet = 'Sirius'  
AND #Projet = #Projet_Equipe  
AND #Equipe = #Appartenance  
AND DaeNais=1973
```

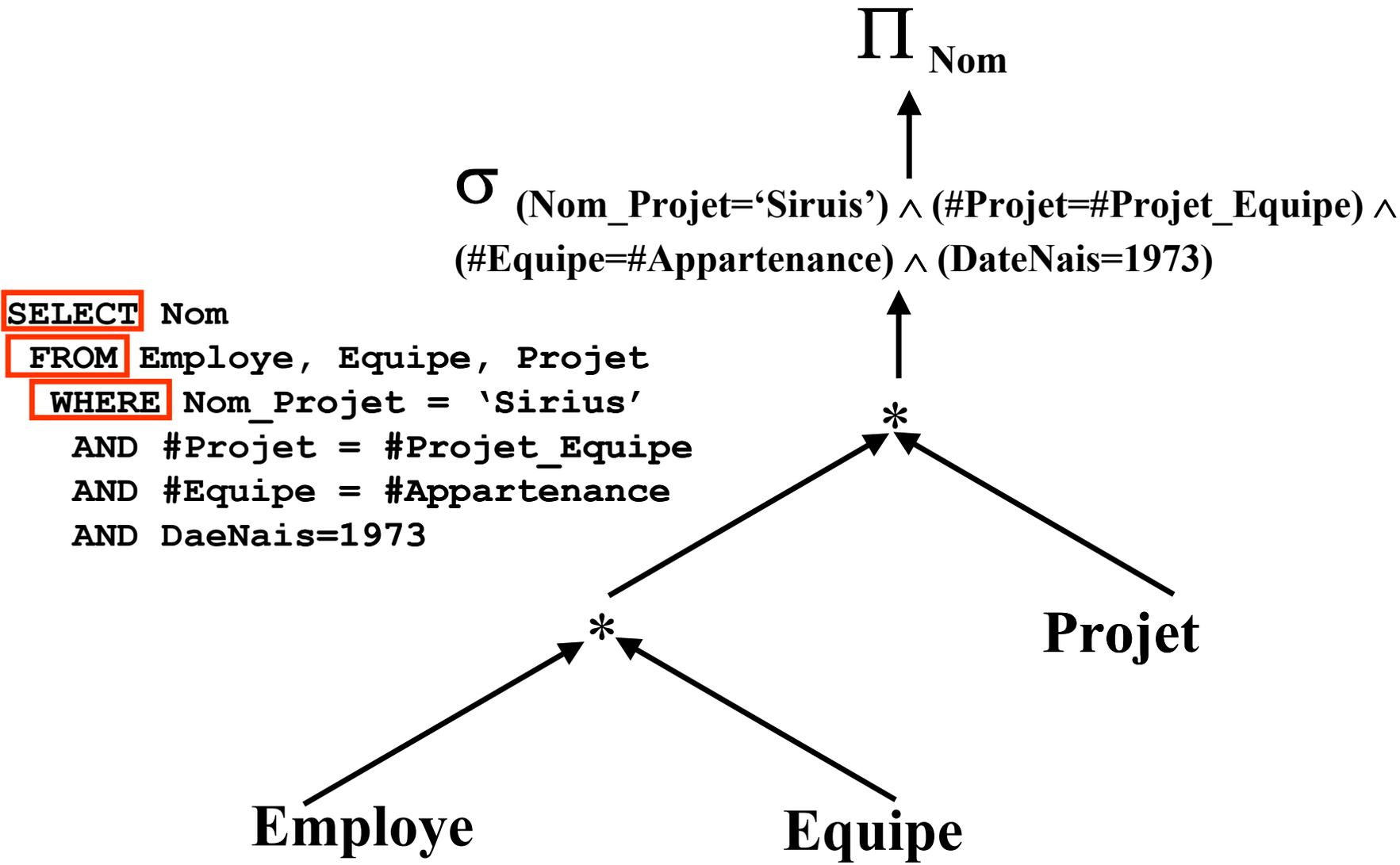


$\Pi_{\text{Nom}}$   
▲

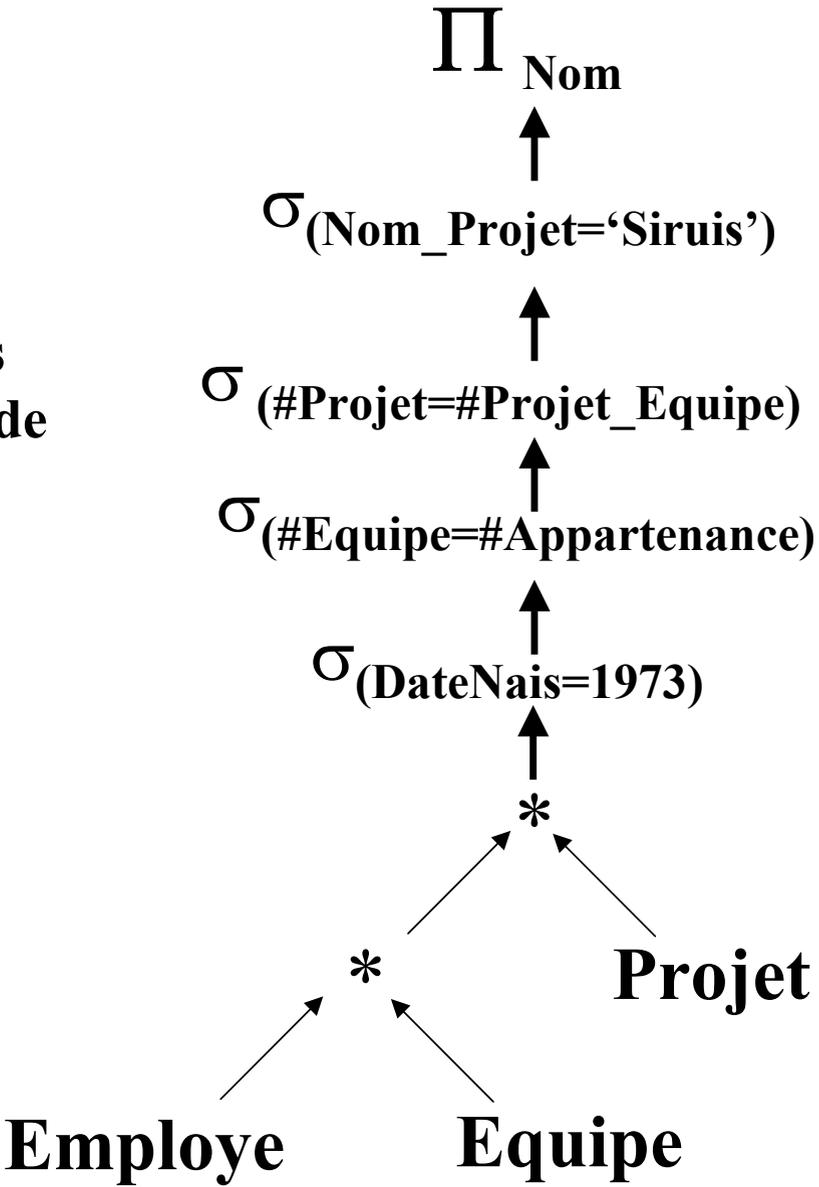
$\sigma_{(\text{Nom\_Projet}=\text{'Siruis'}) \wedge (\#\text{Projet}=\#\text{Projet\_Equipe}) \wedge (\#\text{Equipe}=\#\text{Appartenance}) \wedge (\text{DateNais}=1973)}$

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

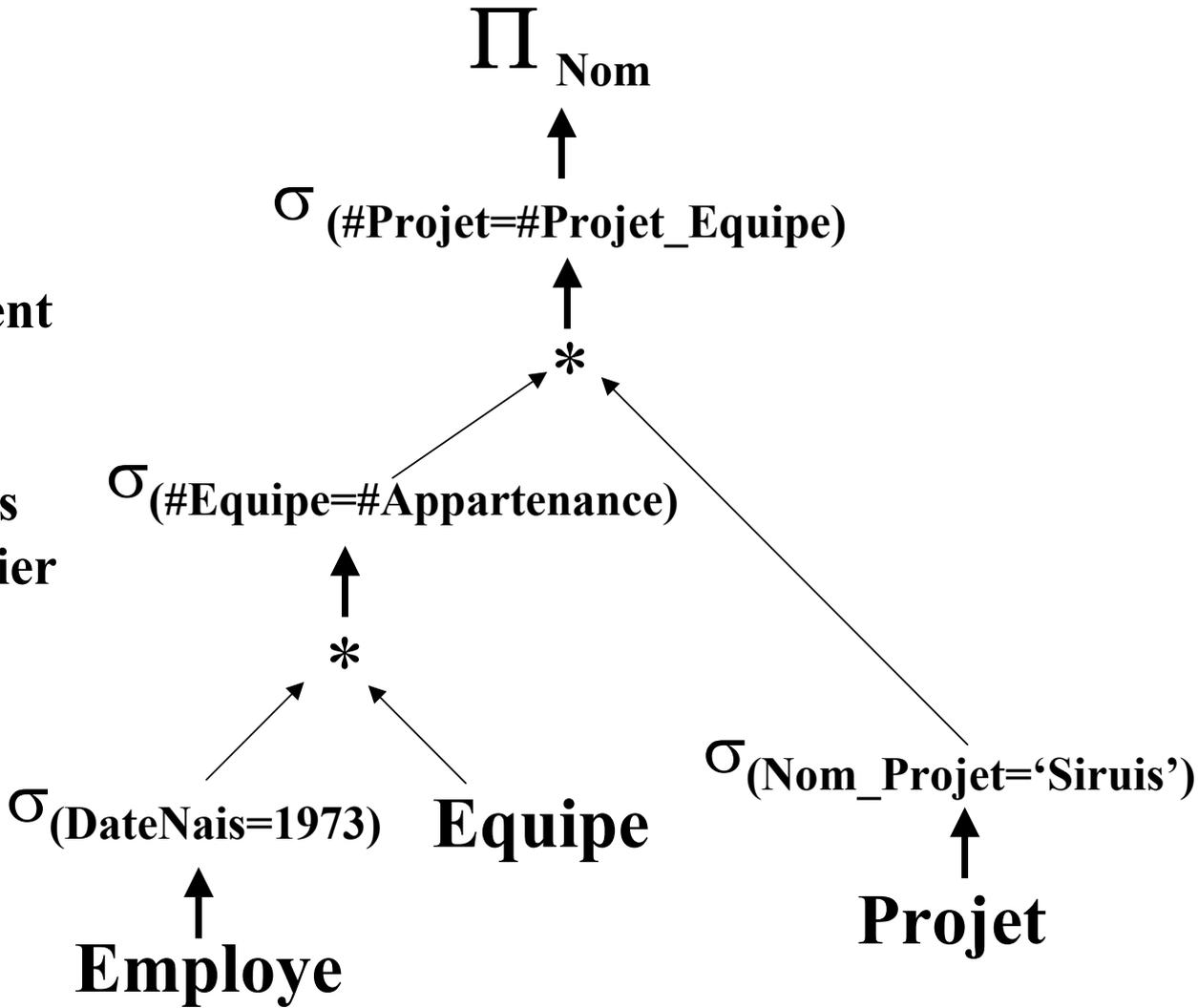




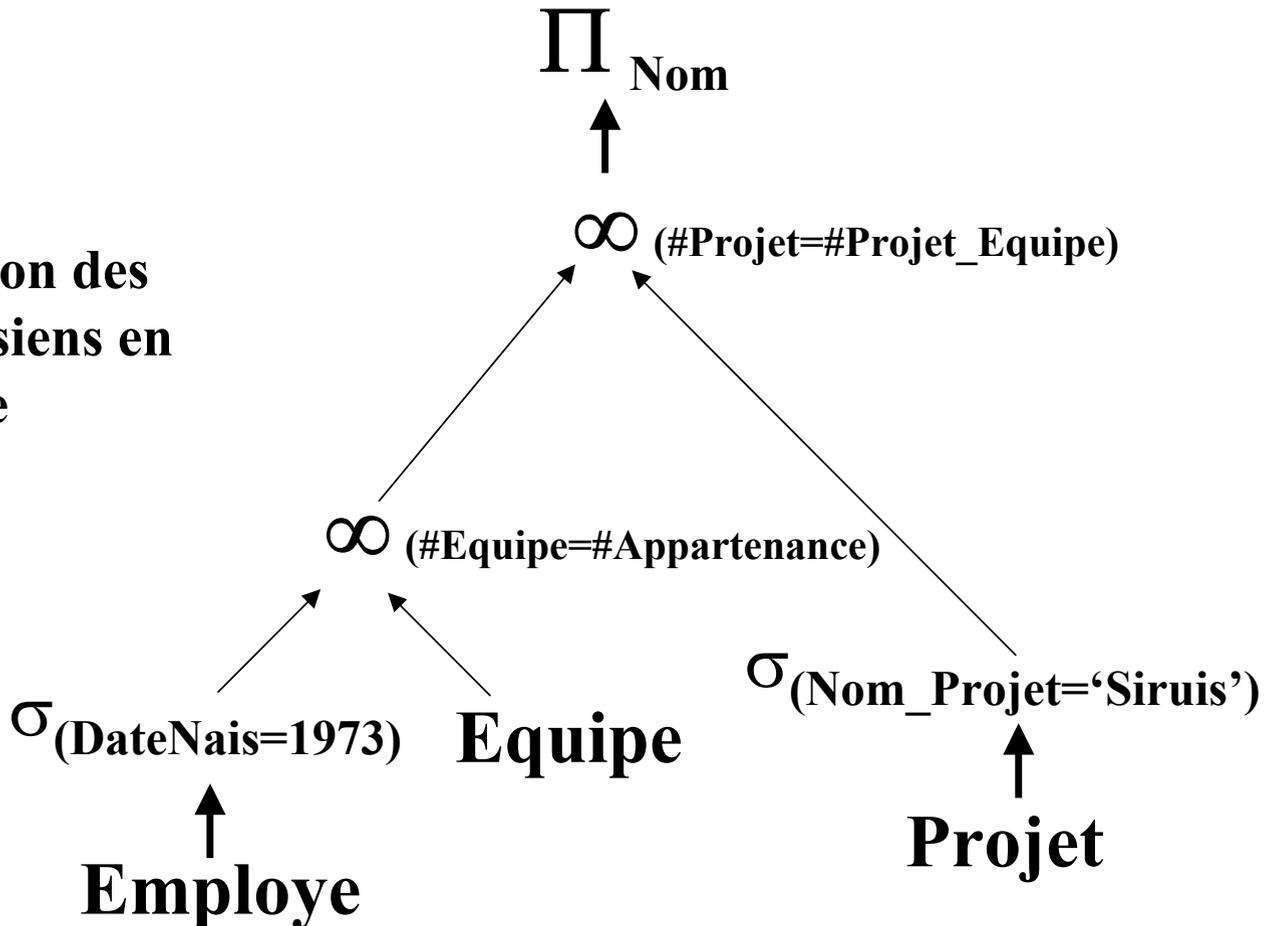
**Division des  
conjonctions de  
sélections**



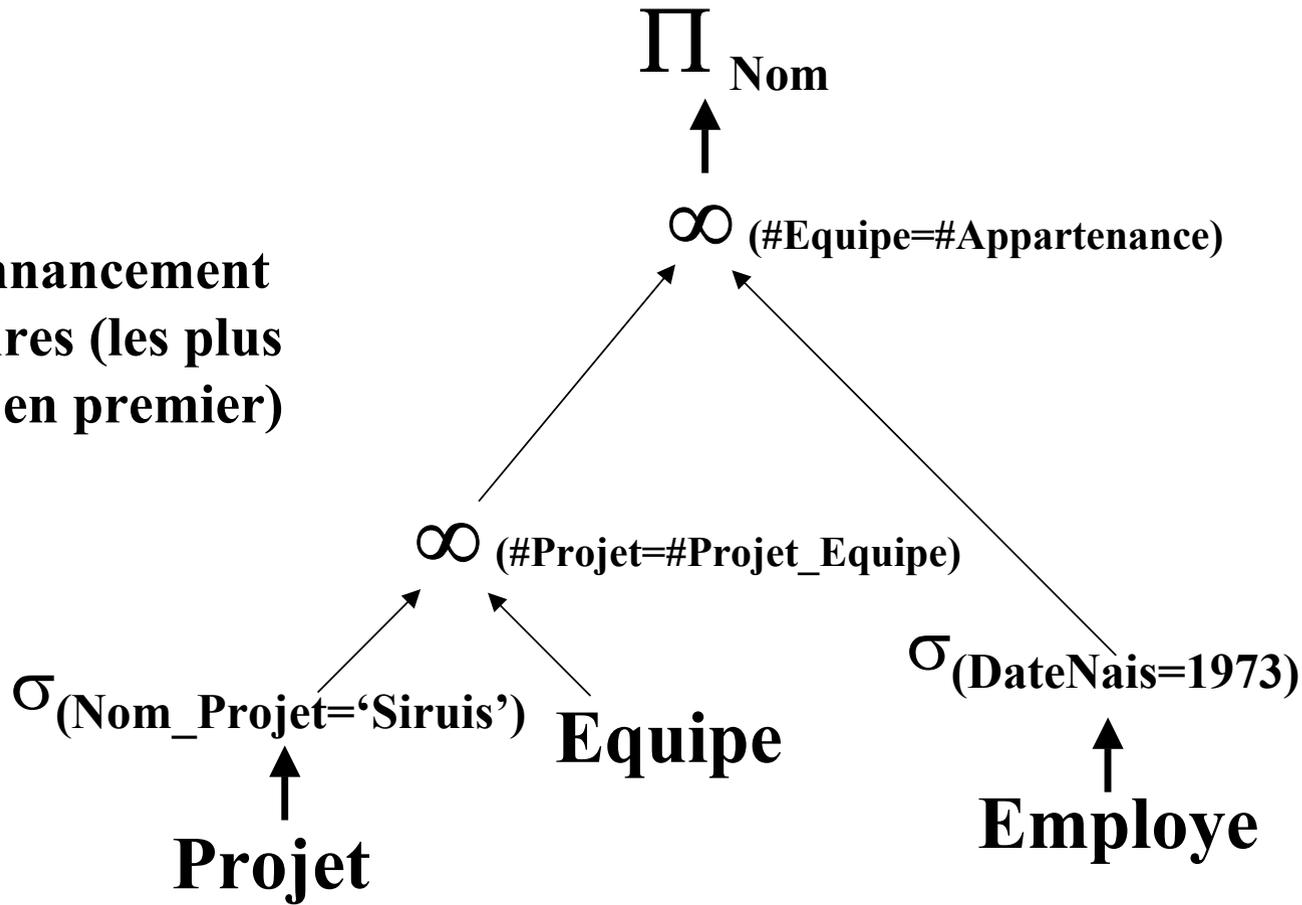
Ré-ordonnancement  
des sélections et  
application des  
sélections les plus  
sélectives en premier



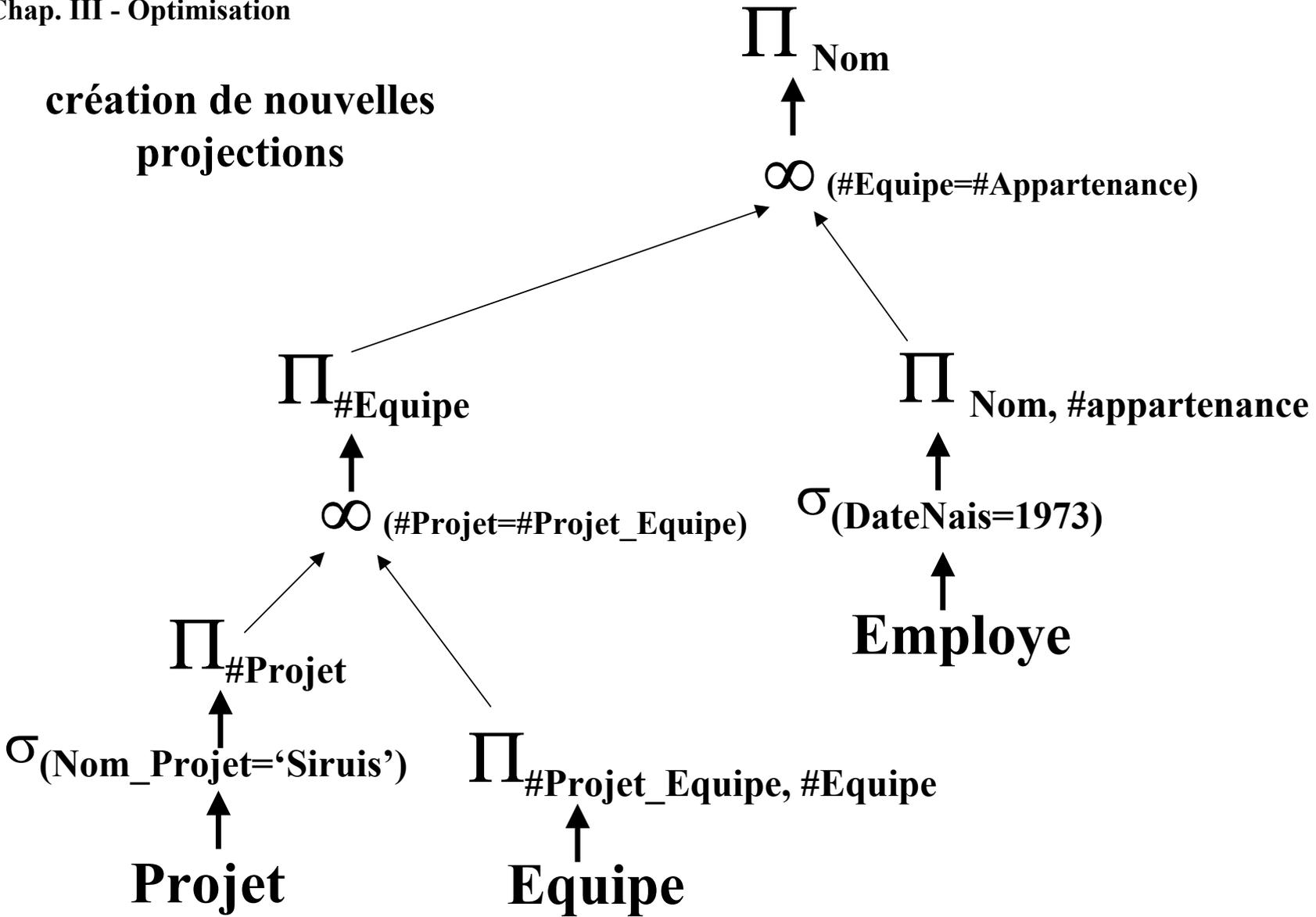
**Transformation des produits cartésiens en jointure**



**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  $I$

# 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

# 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, editing, and execution. The query editor contains the following SQL code:

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

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

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

Temps d'extraction des données : 161 ms

# Exemples en utilisant PostgreSQL

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

Fichier Edition Lancer des requêtes Aide (H)

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

Sortie de données Messages Historique

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

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

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

# Exemples en utilisant PostgreSQL

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

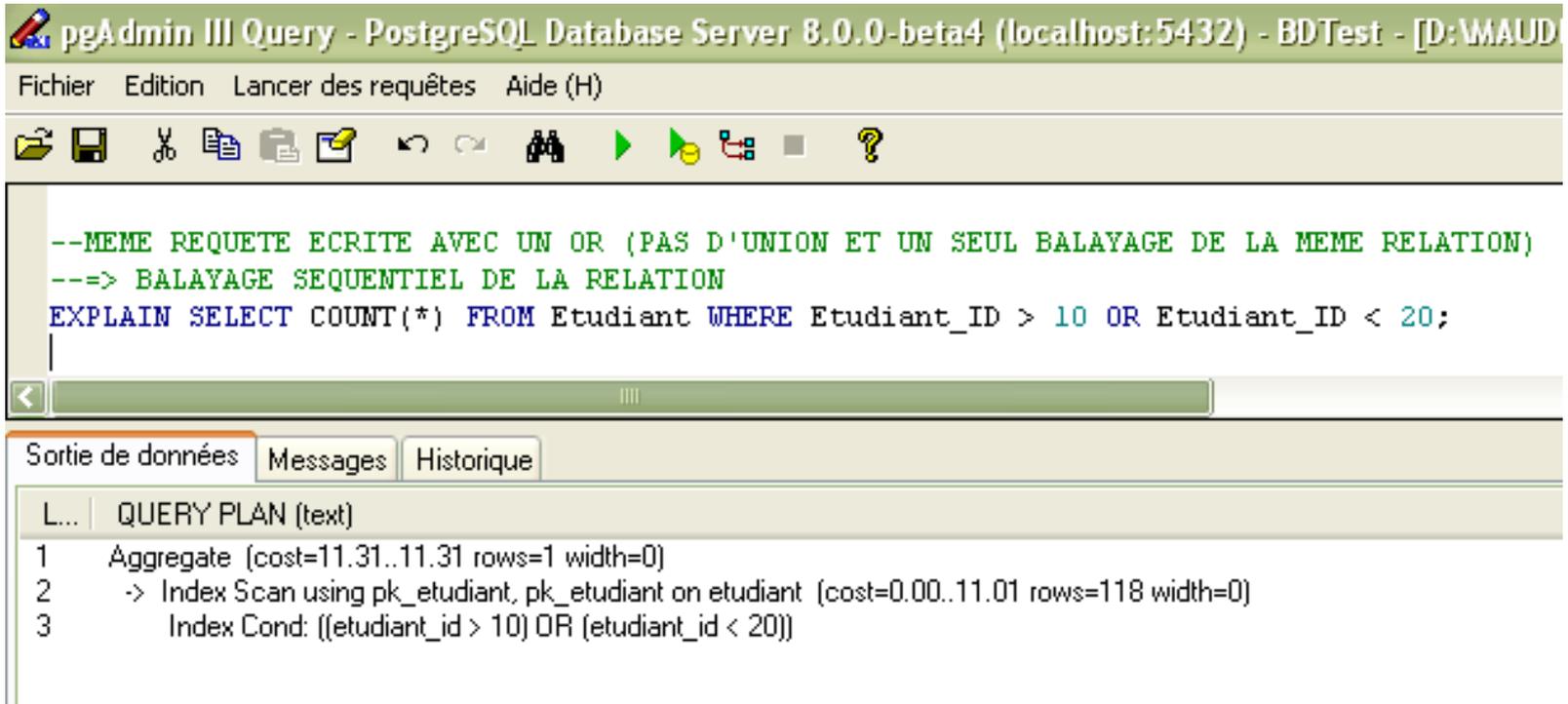
```
--UNION ENTRE DEUX REQUETES => EXECUTION DE DEUX SOUS-REQUETES
--ET UNION DES NUPLETS
EXPLAIN SELECT COUNT(*) FROM Etudiant WHERE Etudiant_ID >10
UNION
SELECT COUNT(*) FROM Etudiant WHERE Etudiant_ID < 20;
```

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

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

Temps d'extraction des données : 280ms

# Exemples en utilisant PostgreSQL



The screenshot shows the pgAdmin III Query tool interface. The title bar reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTest - [D:VMAUD]". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations, navigation, 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;
```

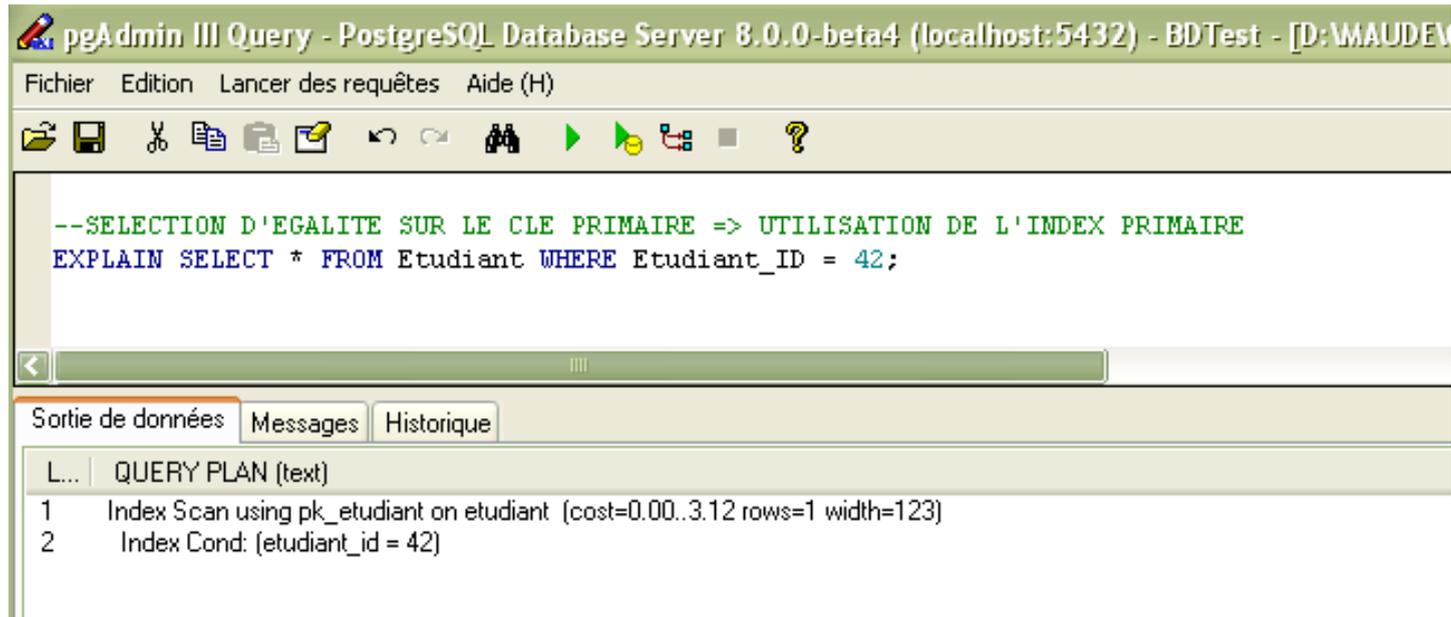
Below the query, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, showing the following query plan:

```
L... | QUERY PLAN (text)  
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 the connection to a PostgreSQL Database Server 8.0.0-beta4 on localhost:5432. The main 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;
```

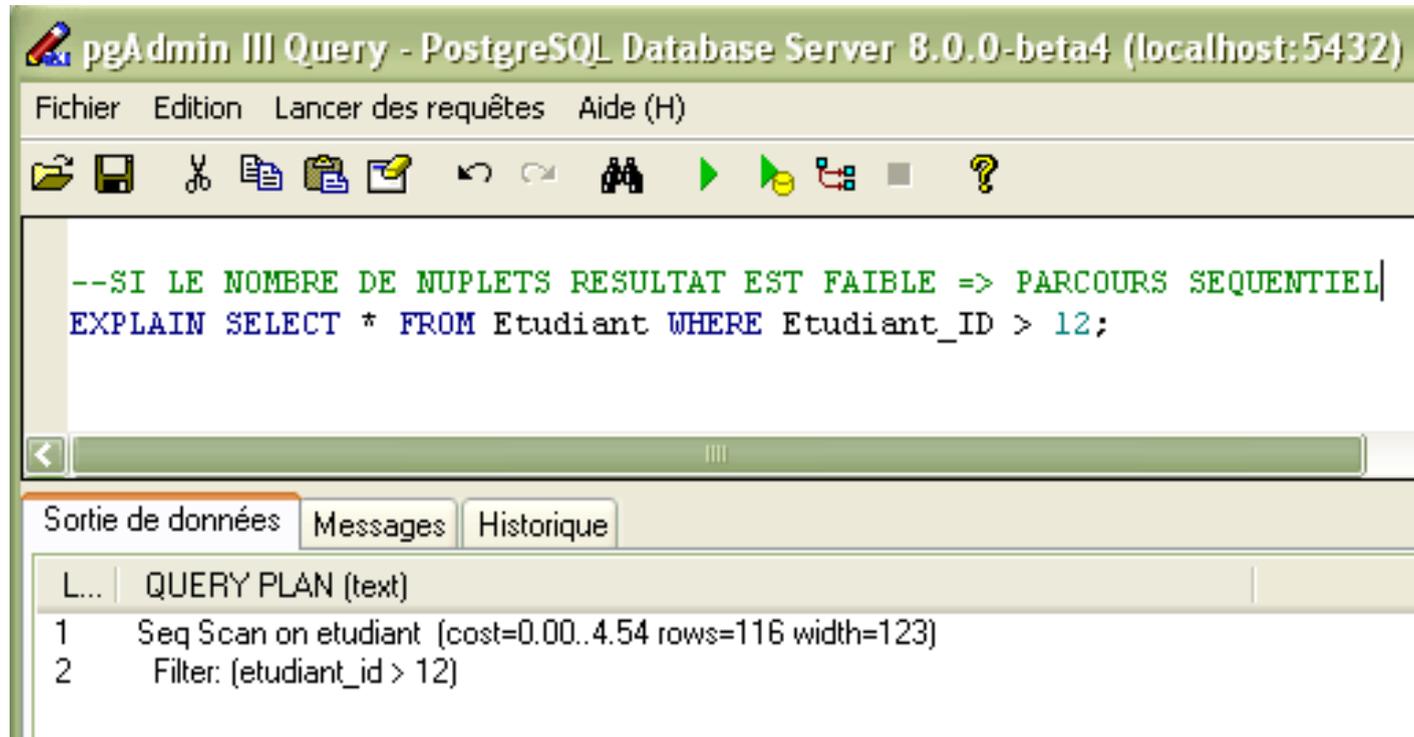
Below the query editor, the 'Sortie de données' (Data Output) tab is active, displaying the query plan:

L...	QUERY PLAN (text)
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



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 and execution. The query editor contains the following SQL code:

```
--SI LE NOMBRE DE NUPLETS RESULTAT EST FAIBLE => PARCOURS SEQUENTIEL|  
EXPLAIN SELECT * FROM Etudiant WHERE Etudiant_ID > 12;
```

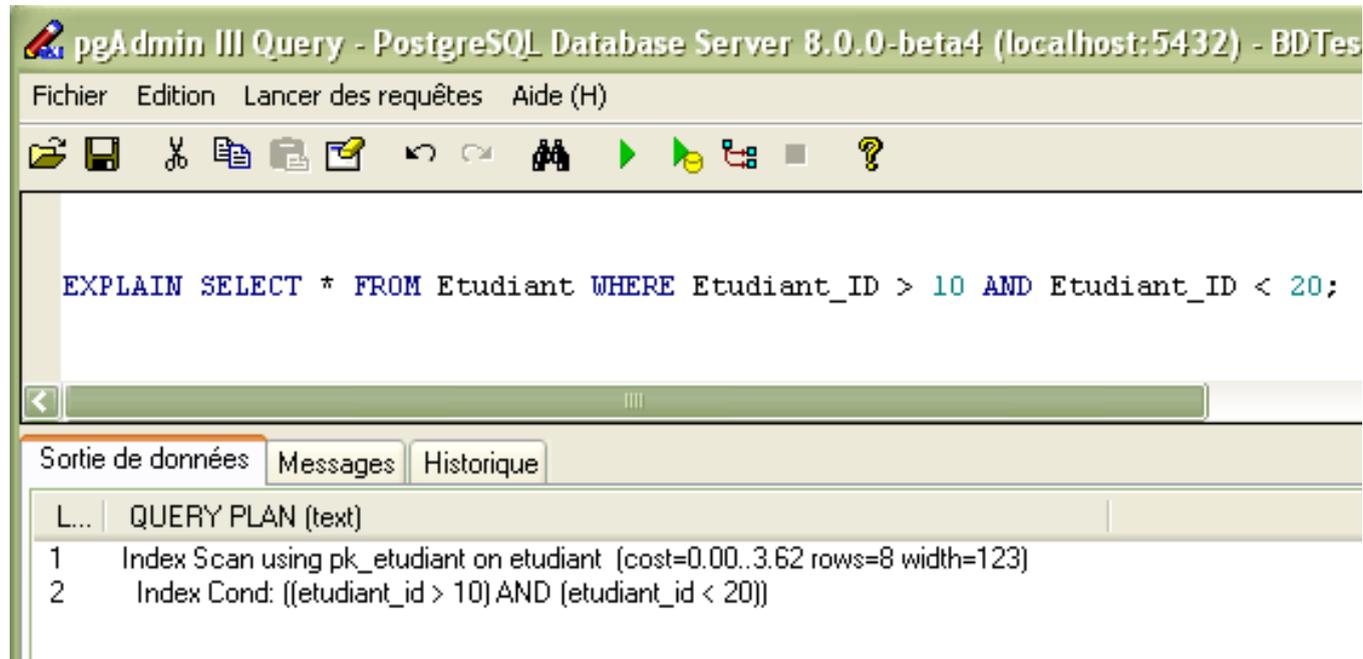
Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the following query plan:

L...	QUERY PLAN (text)
1	Seq Scan on etudiant (cost=0.00..4.54 rows=116 width=123)
2	Filter: (etudiant_id > 12)

Temps d'extraction des données : 220ms

# Exemples en utilisant PostgreSQL

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



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

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

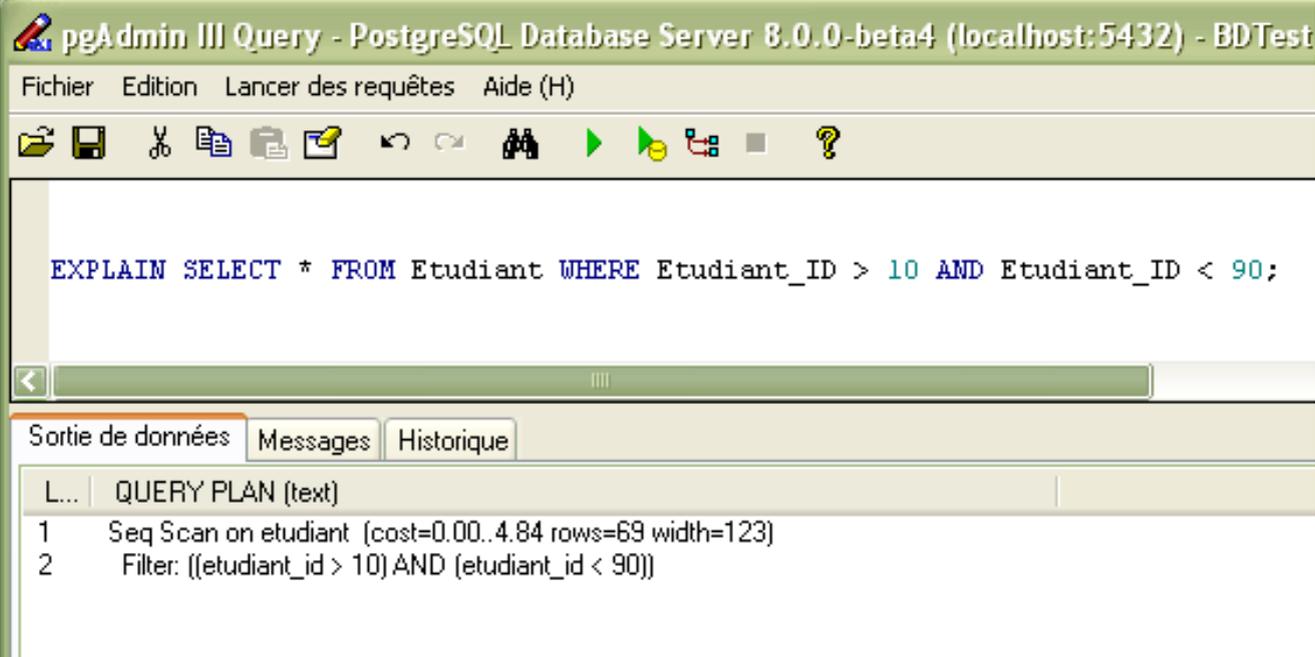
Below the query editor, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the query plan:

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

Temps d'extraction des données : 230ms

# Exemples en utilisant PostgreSQL

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



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, navigation, and execution. The main text area contains the following SQL query:

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

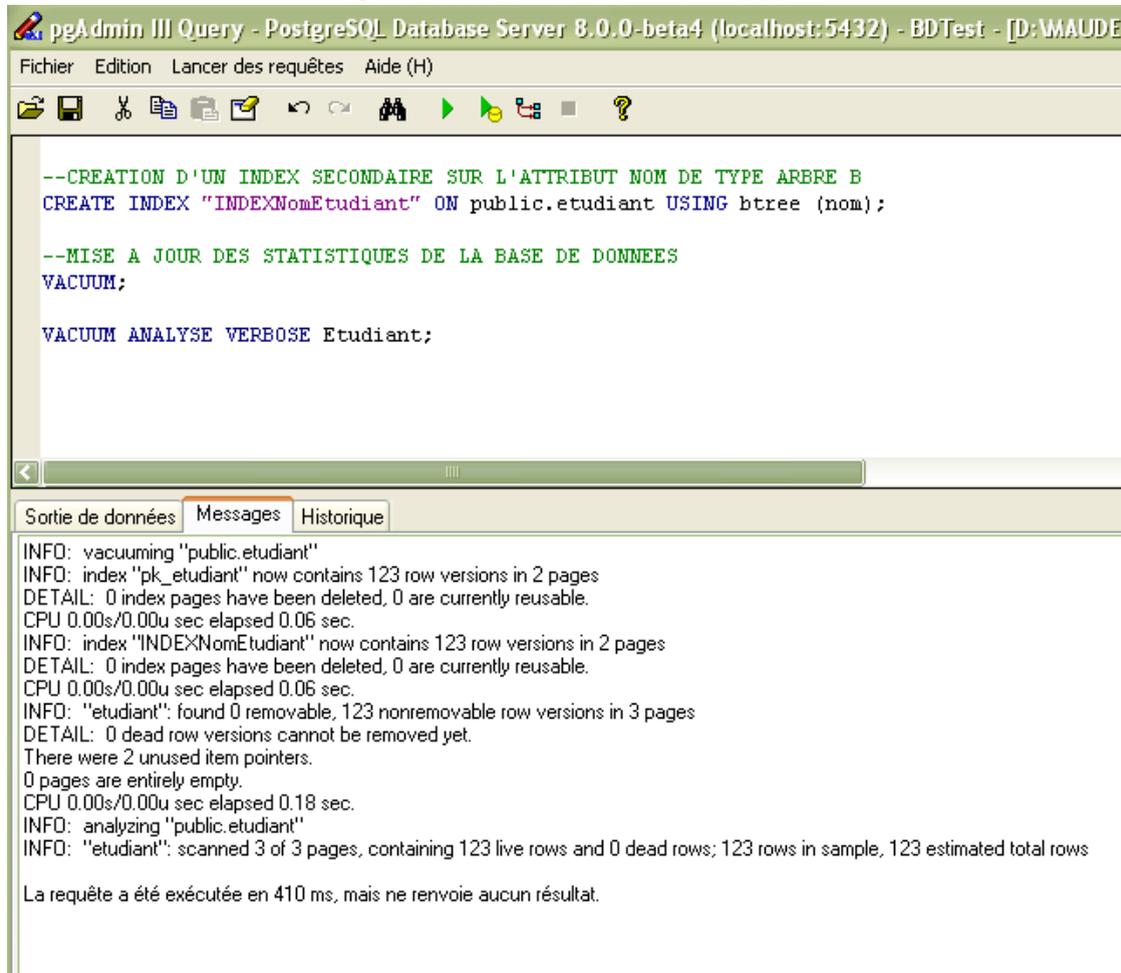
Below the query area, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying the following query plan:

L...	QUERY PLAN (text)
1	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 reads "pgAdmin III Query - PostgreSQL Database Server 8.0.0-beta4 (localhost:5432) - BDTest - [D:WMAUDE]". The menu bar includes "Fichier", "Edition", "Lancer des requêtes", and "Aide (H)". The toolbar contains icons for file operations and execution. The main text area contains the following SQL commands:

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

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

VACUUM ANALYSE VERBOSE Etudiant;
```

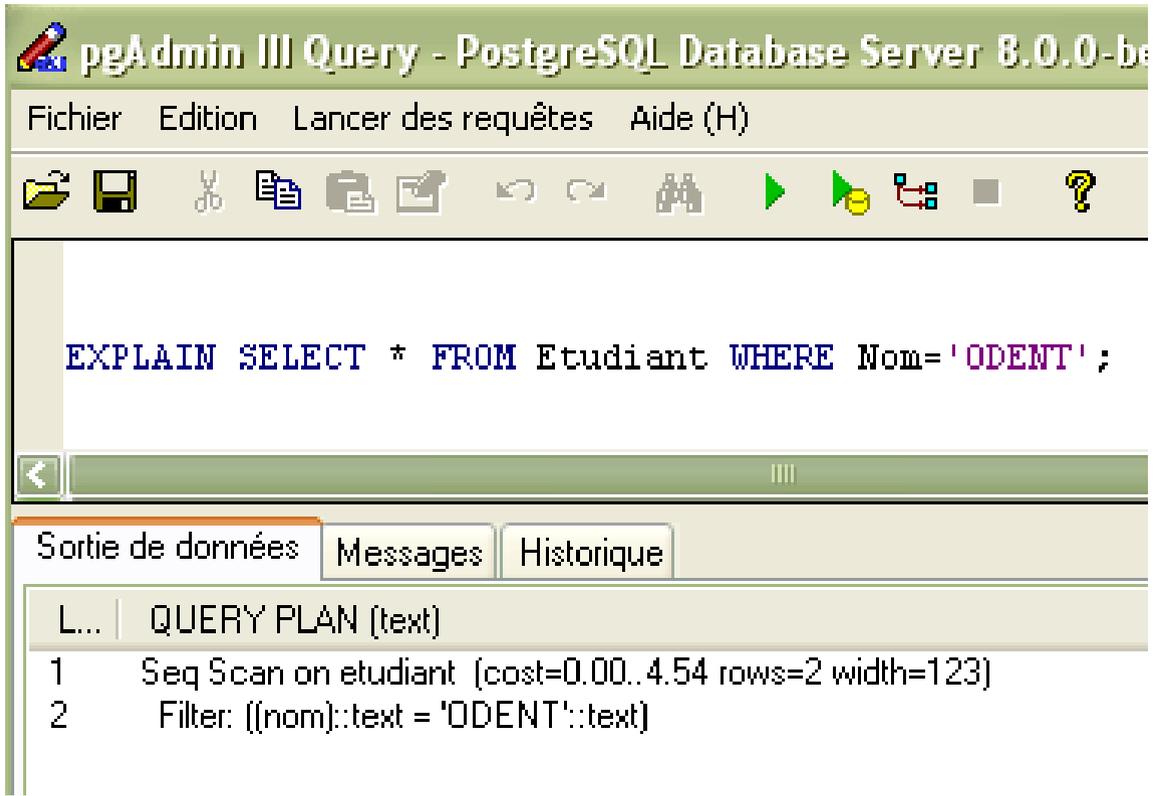
The output pane at the bottom shows the following messages:

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

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

# Exemples en utilisant PostgreSQL

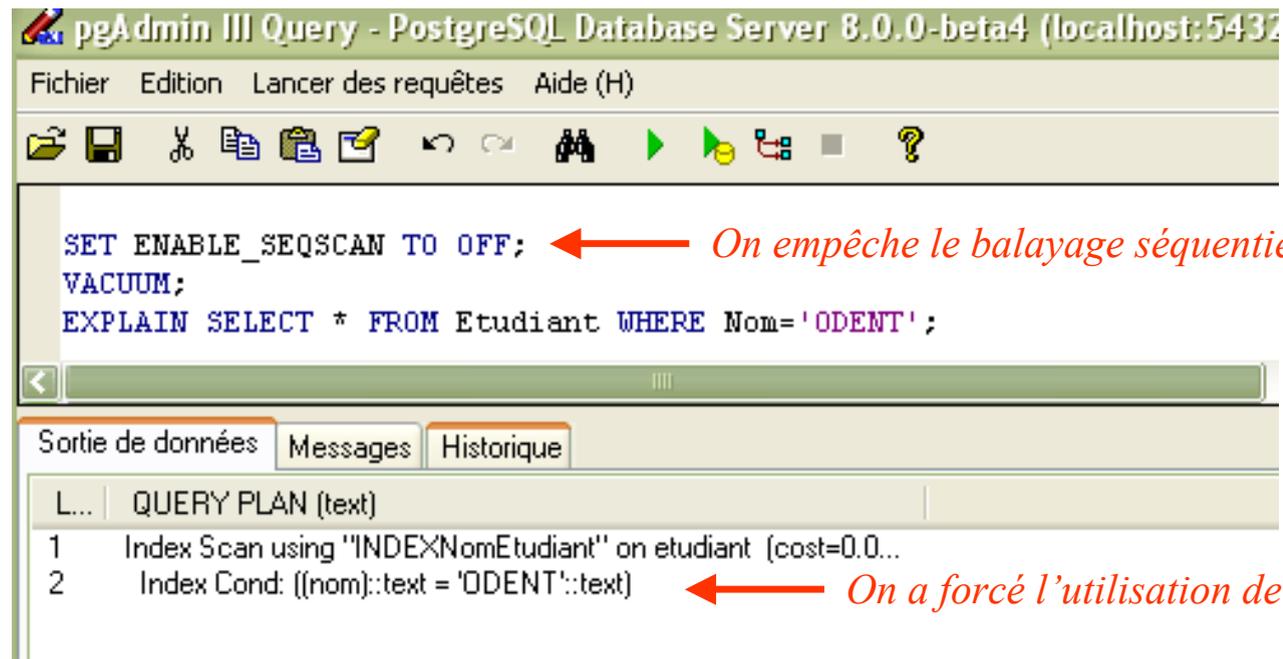
**Le SGBD peut choisir de ne pas utiliser les index**



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

# Exemples en utilisant PostgreSQL

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



← *On empêche le balayage séquentiel*

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

**Temps d'extraction des données : 220ms**

# Exemples en utilisant PostgreSQL

## Algorithmes de jointure

The screenshot shows the pgAdmin III Query tool interface. The title bar 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 and execution. The main text area contains the following SQL commands:

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

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

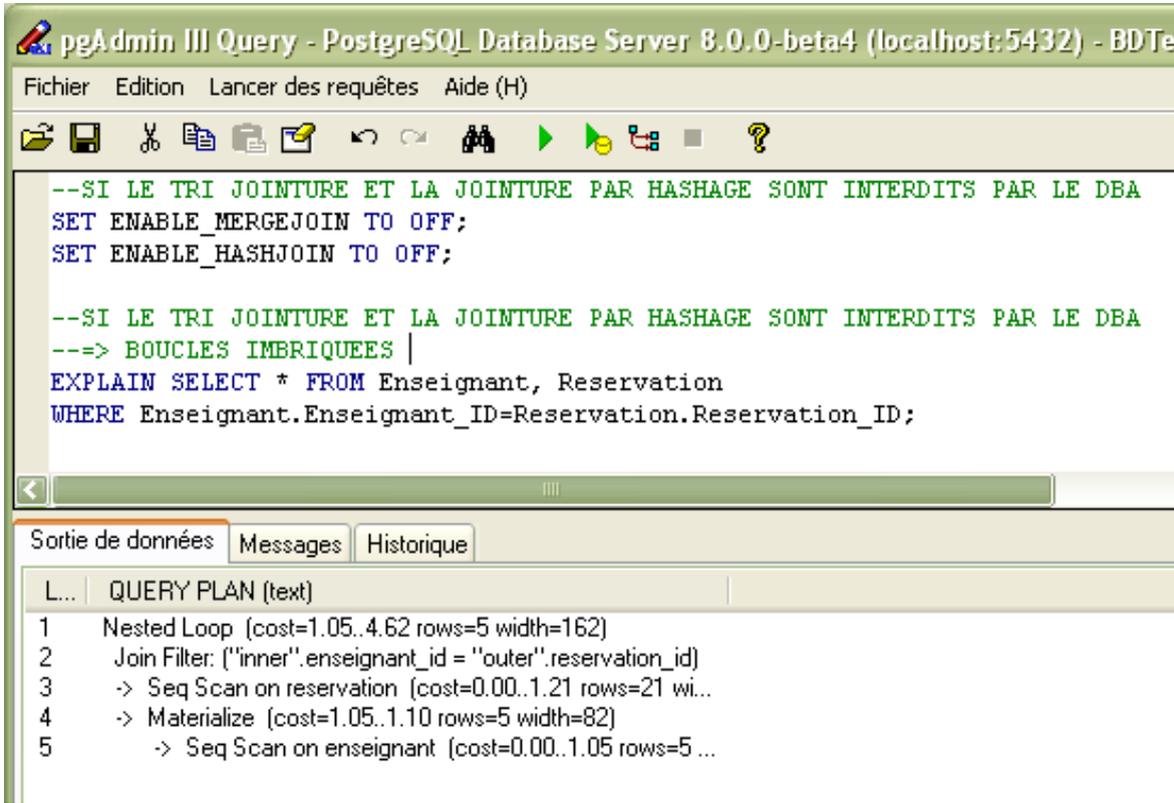
Below the text area, there are tabs for "Sortie de données", "Messages", and "Historique". The "Sortie de données" tab is active, displaying a query plan:

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

Temps d'extraction des données : 201ms

# Exemples en utilisant PostgreSQL

## Algorithmes de jointure



Temps d'extraction des données : 190ms

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

```
--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'IMPORTANT SUR LE RESULTAT
```

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

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

Temps  
d'extraction  
des données :  
200ms

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

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

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

```
1  Nested Loop (cost=0.01..4.24 rows=1 width=771)
2  -> HashJoin (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)
```

Temps  
d'extraction  
des données :  
200ms

# Chap. IV - Gestion de la concurrence

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

**[BEGIN TRANSACTION]**

...

**COMMIT / ROLLBACK**

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

# Propriétés des transactions

- **Atomicité** : Tout ou rien

Une transaction effectue toutes ses actions ou aucune.

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

- **Cohérence** : Intégrité des données

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

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

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

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

Les modifications effectuées sont garanties même en cas de panne

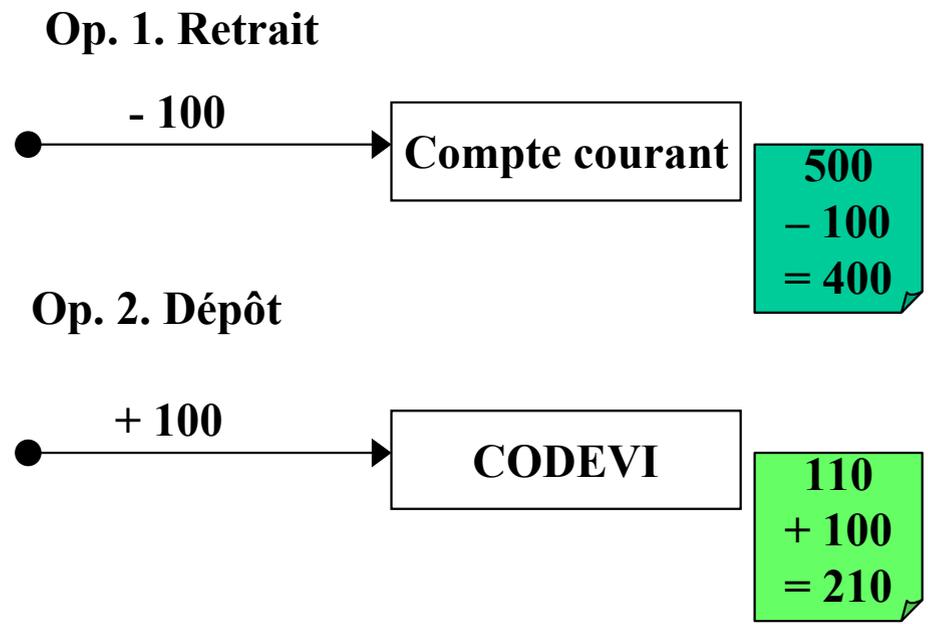
# Exemple de transaction

# Exemple de transaction

Virement =  
2 opérations atomiques

# Exemple de transaction

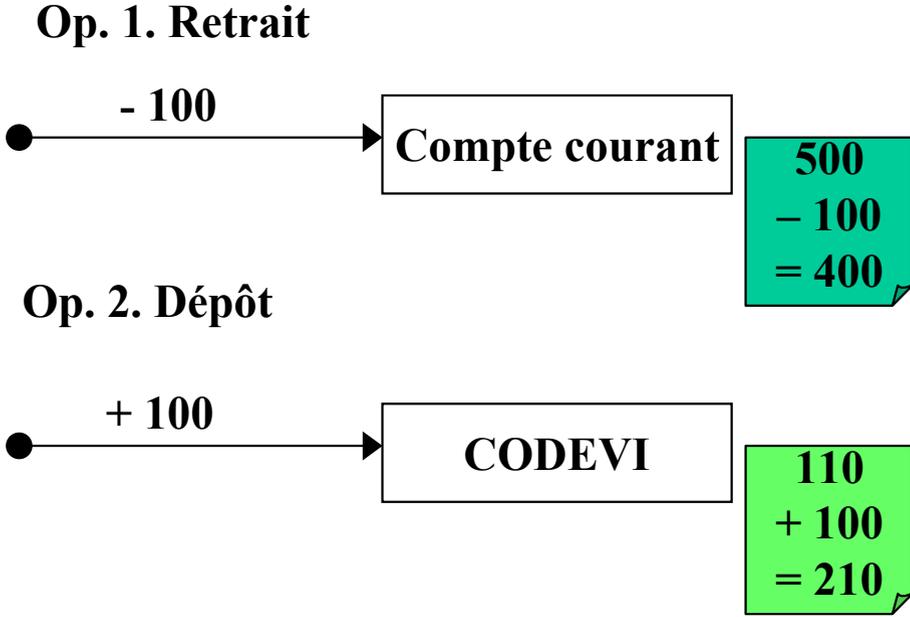
Virement =  
2 opérations atomiques



# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques

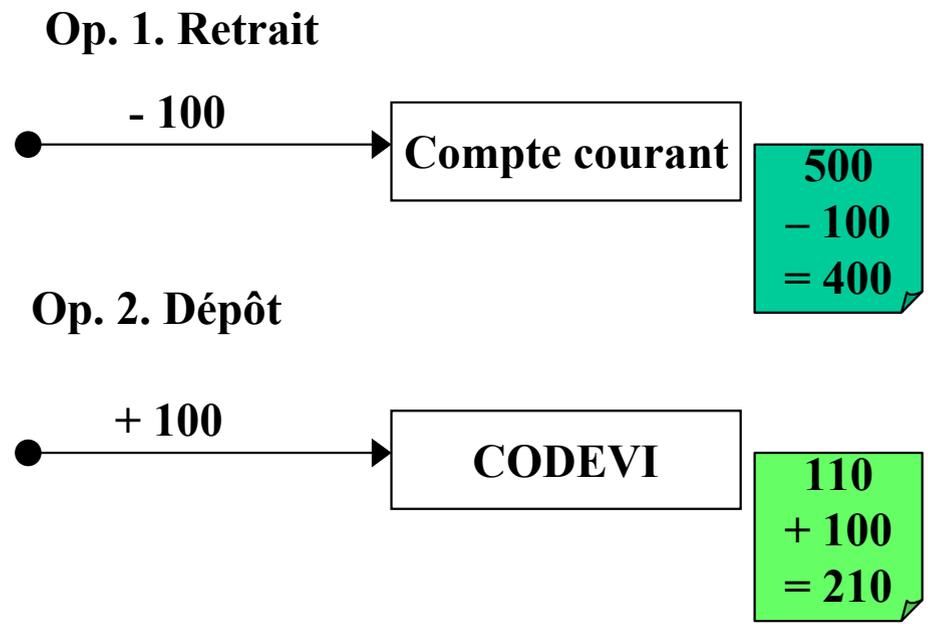


# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques

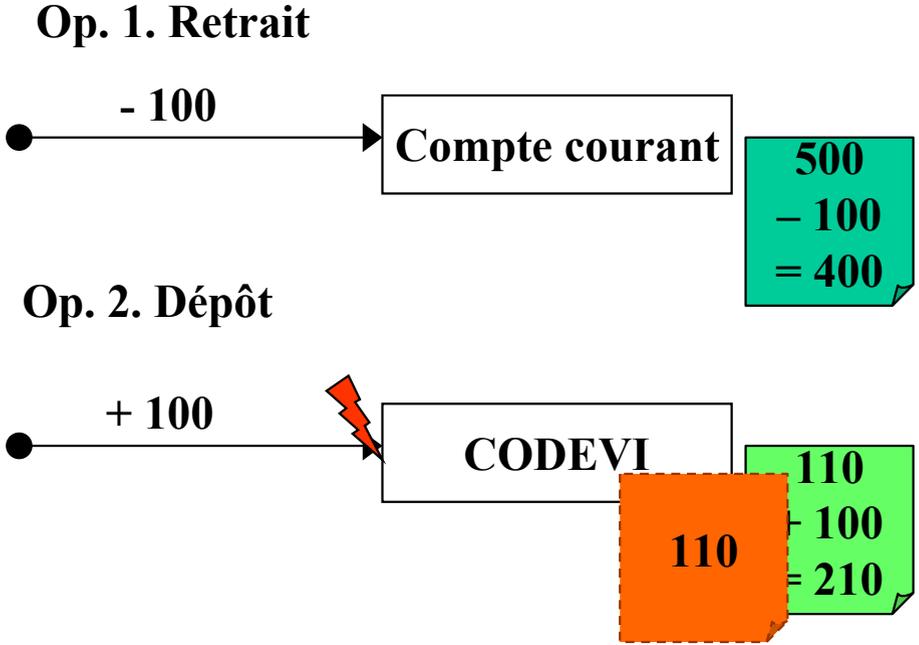
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

## Virement bancaire **sans transaction**

Virement =  
2 opérations atomiques  
  
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

## Virement bancaire sans transaction

Virement =

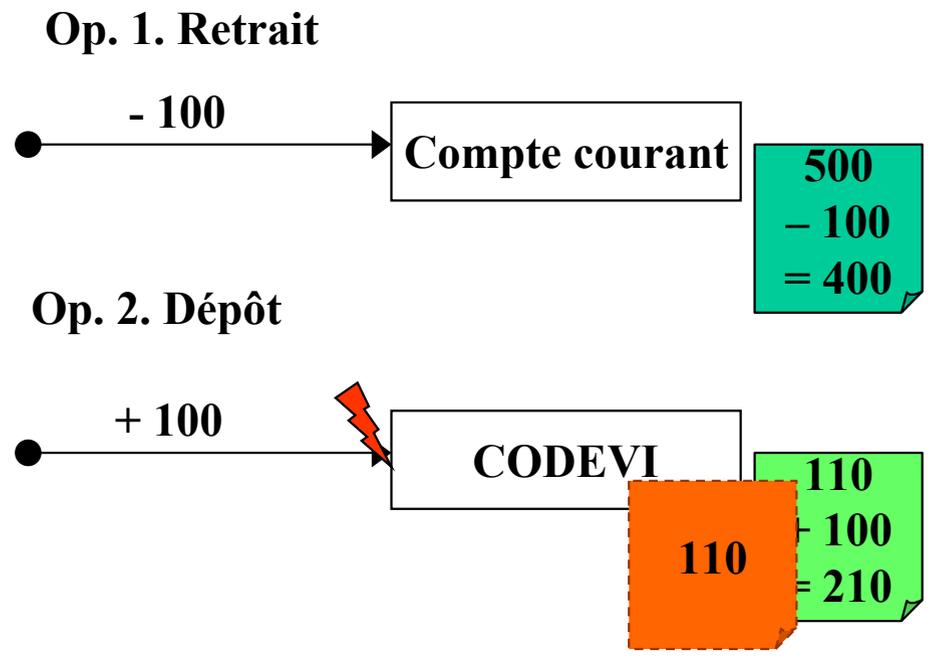
2 opérations atomiques

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

**Compte courant = 400**

**CODEVI = 110**

**Appelez la banque !!!**



# Exemple de transaction

# Exemple de transaction

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

# Exemple de transaction

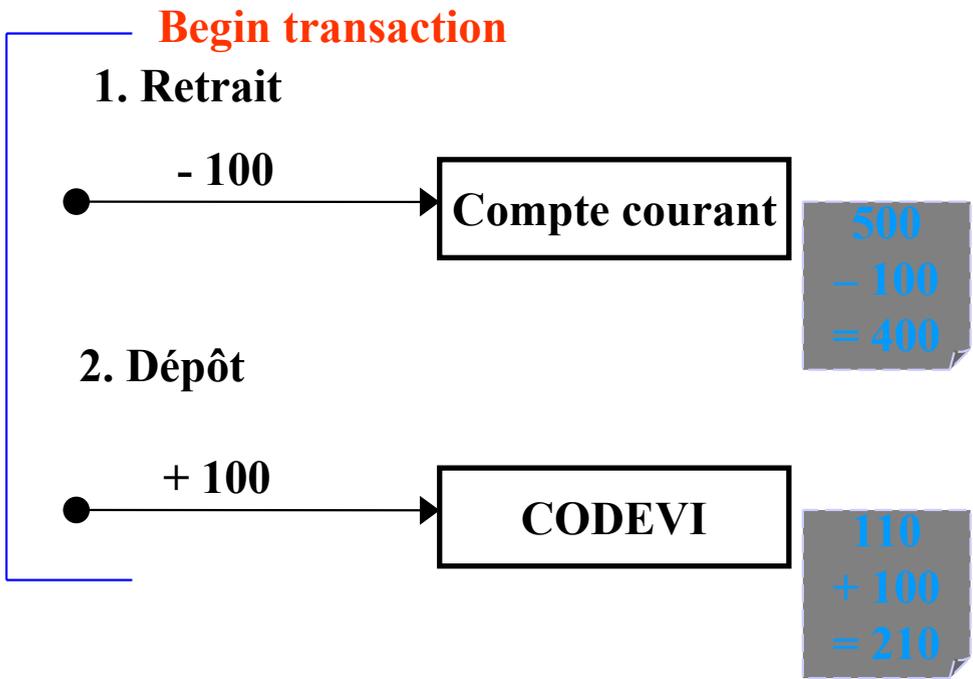
**Virement bancaire dans une transaction (1/2)**

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

# Exemple de transaction

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

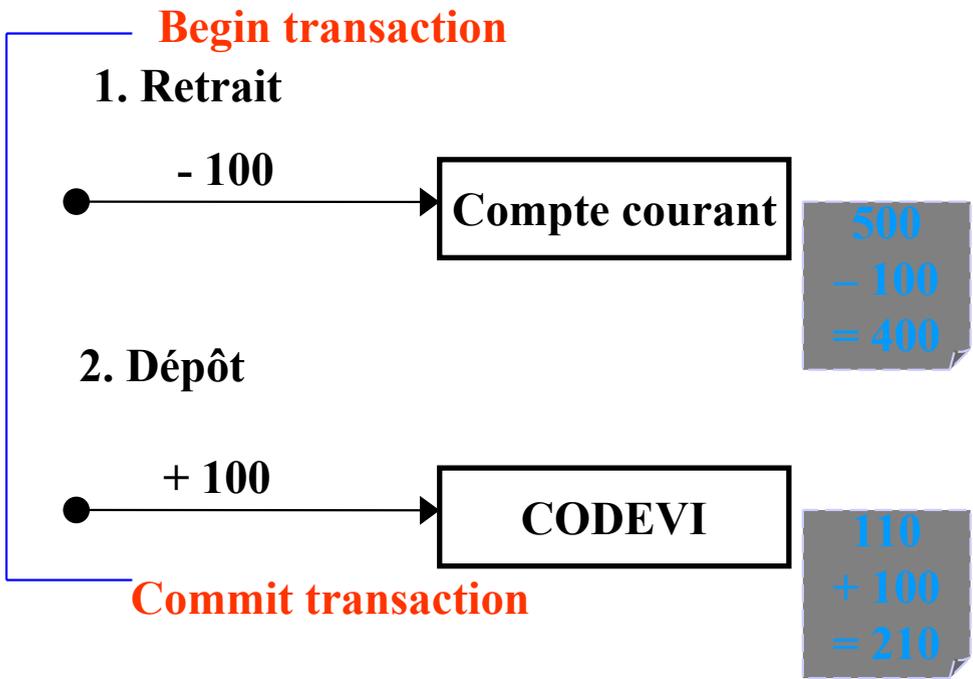
Virement = 1 transaction  
de 2 opérations  
atomiques



# Exemple de transaction

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

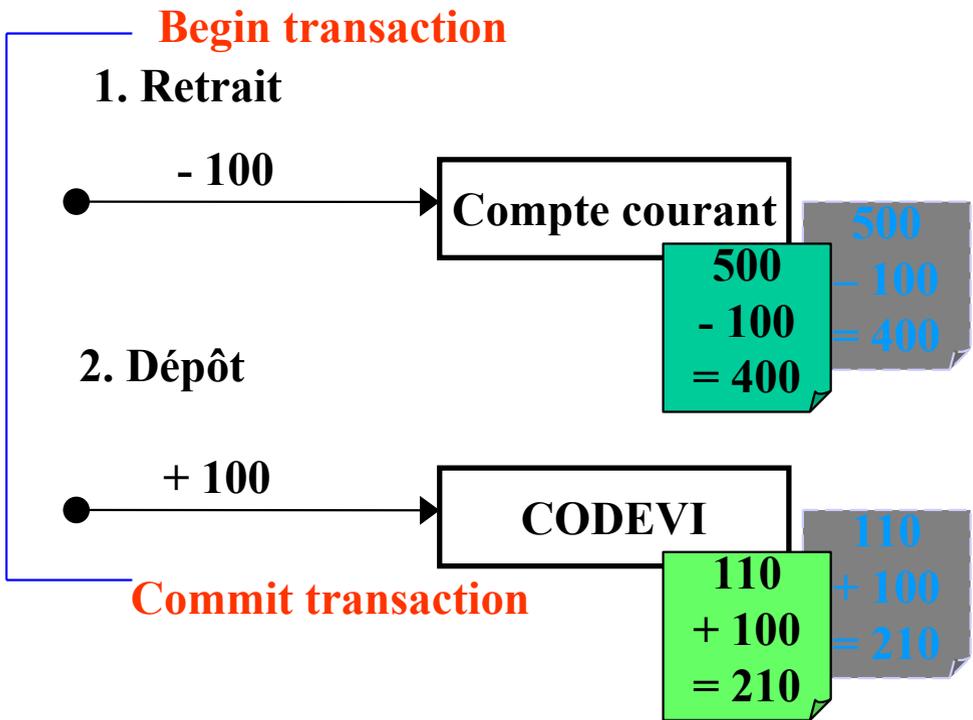
Virement = 1 transaction  
de 2 opérations  
atomiques



# Exemple de transaction

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

Virement = 1 transaction de 2 opérations atomiques



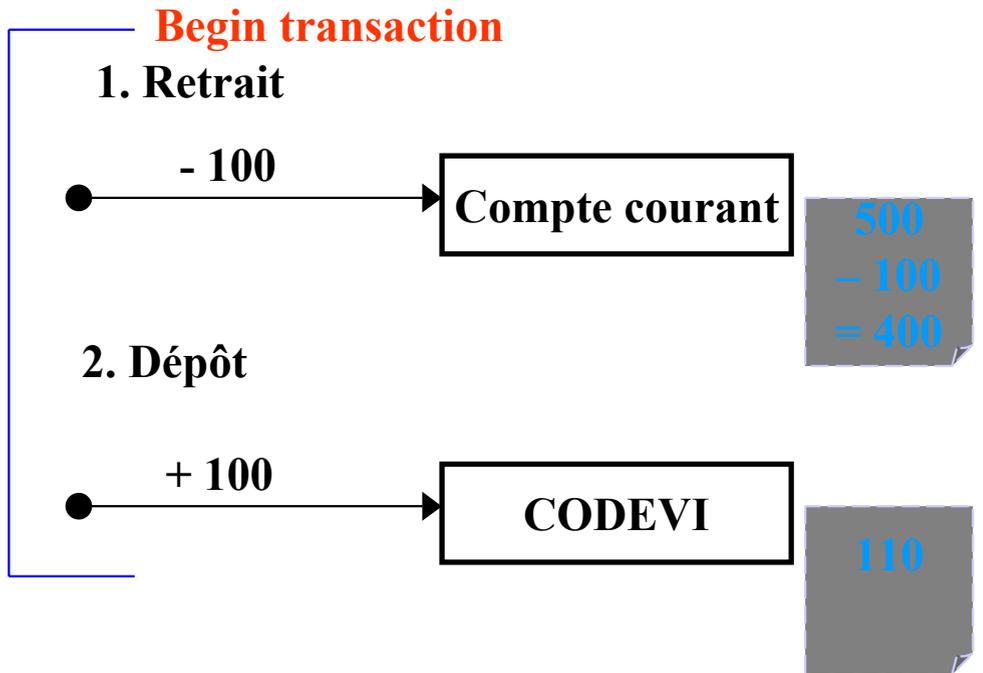
# Exemple de transaction

# Exemple de transaction

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

# Exemple de transaction

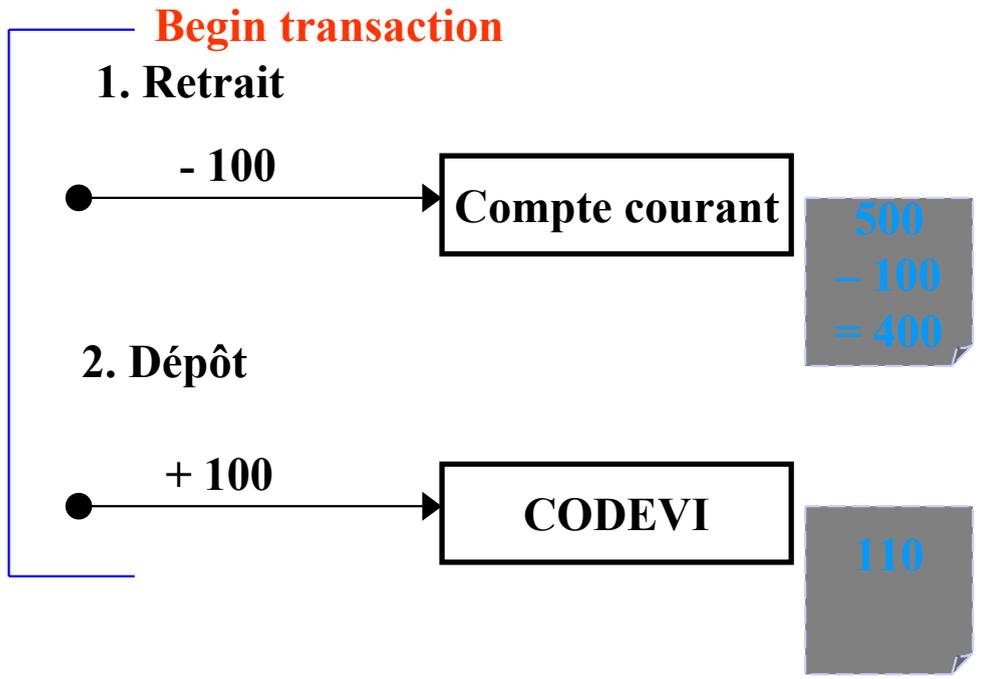
## Virement bancaire dans une transaction (2/2)



# Exemple de transaction

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

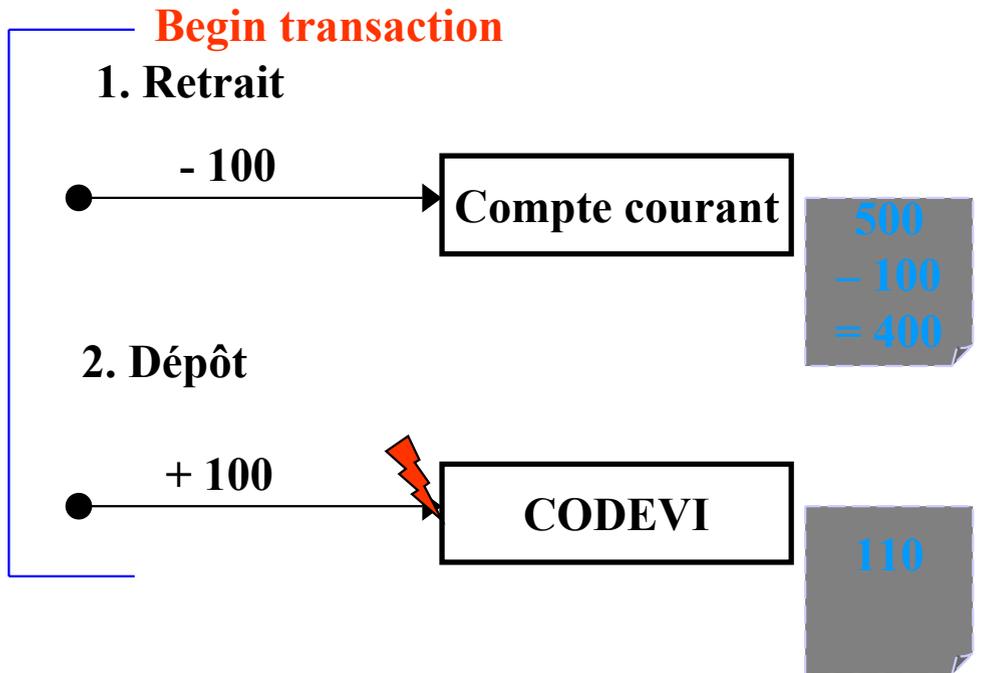
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

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

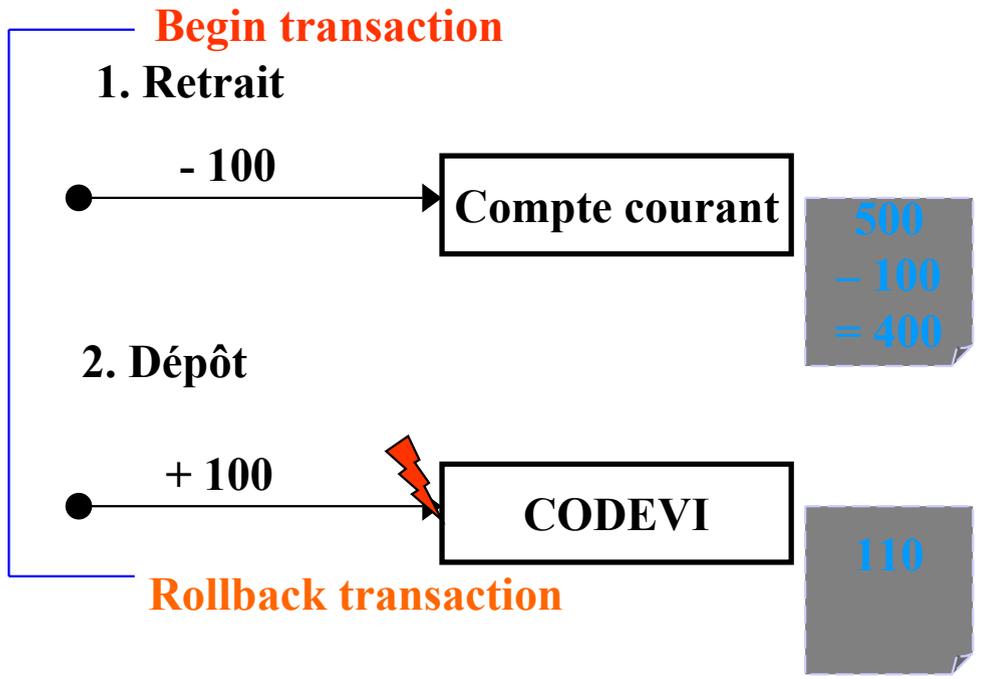
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

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

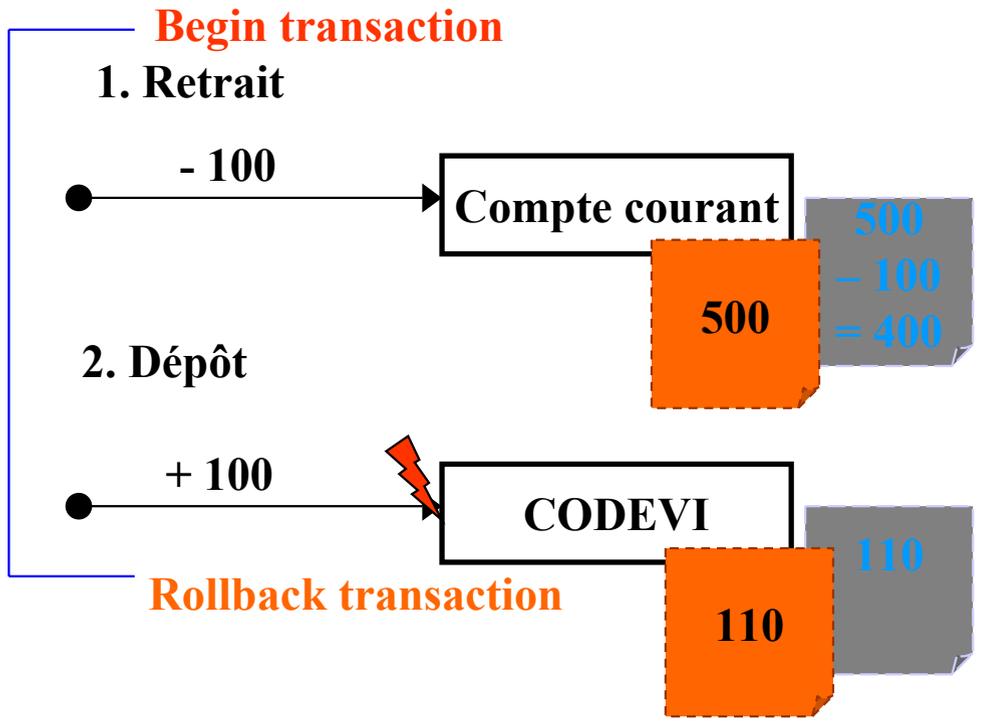
Que se passe-t-il si le  
Dépôt échoue ?



# Exemple de transaction

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

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



# Exemple de transaction

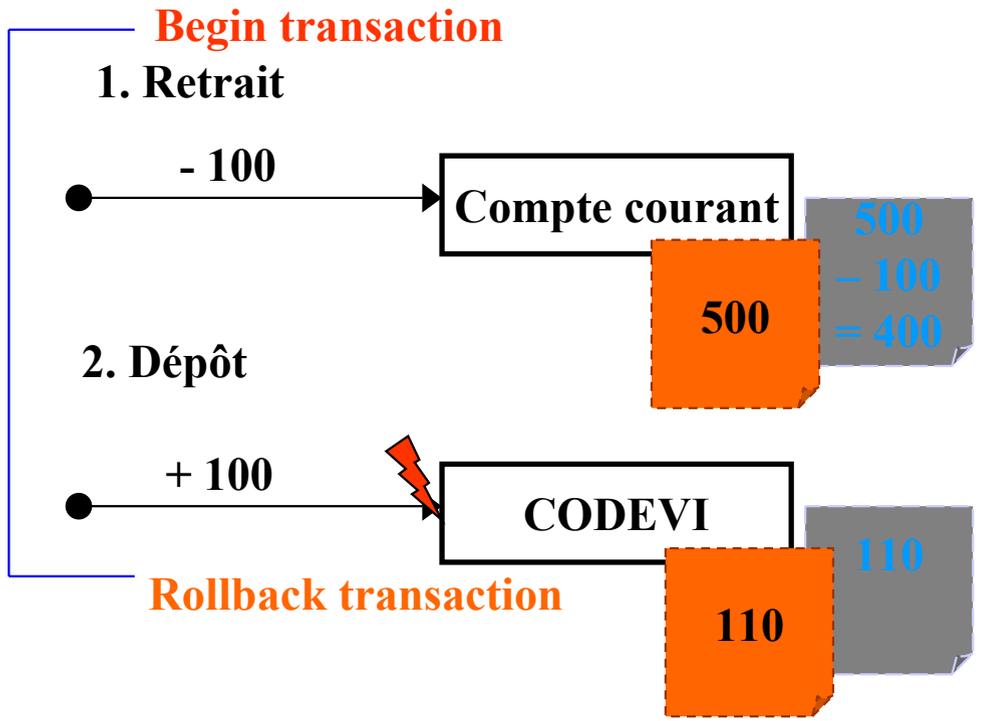
## Virement bancaire dans une transaction (2/2)

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

Compte courant = 500

CODEVI = 110

Recommencez !



# Degrés d'isolation sous SQL2

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

# Architecture du système de transactions

## Missions du système de transactions

Gérer les transactions, maintenir la cohérence, gérer les pannes

- **Gestionnaire de transactions :**

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

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

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

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

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

# Ordonnement

- **Opération d'une transaction  $T$**

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

- **Ordonnement de transactions**

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

- **Ordonnement séquentiel**

Pas d'entrelacement des actions des différentes transactions

# Concurrency

- **Transactions concurrentes**

Deux transactions accédant en même temps aux mêmes items

- **Ordonnancement sérialisable**

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

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

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

# Odonnancement sérialisable

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

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

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

Exécution séquentielle :  $T_1 T_2 T_3$

Une exécution sérialisable équivalente à  $T_1 T_2 T_3$ :

Solde A - x;

Solde A - y;

Solde B + x;

Solde A - z;

Solde B + y;

Solde B + z;

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

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

# Conflicts

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

# Transactions et SQL2

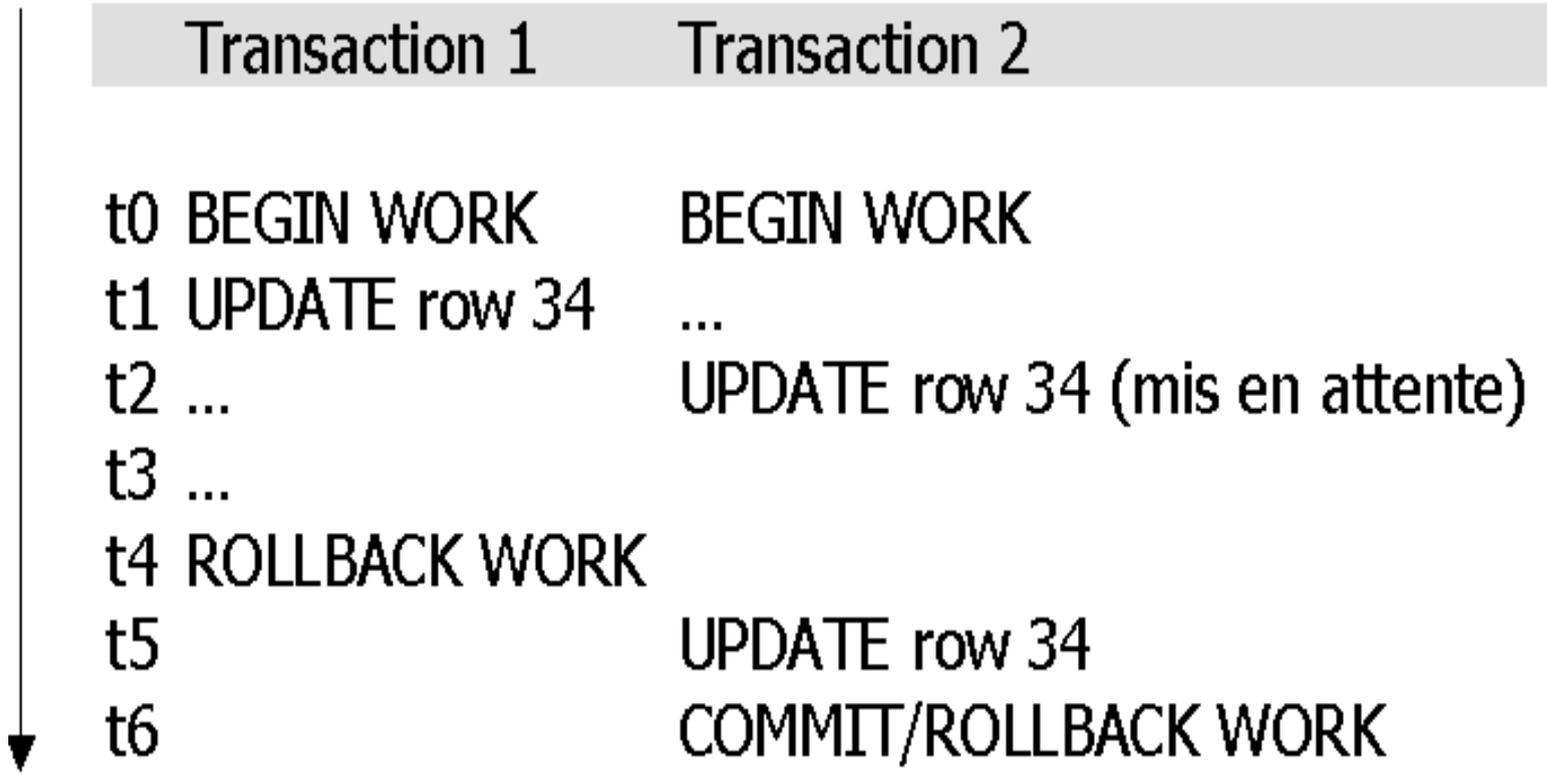
- Une transaction commence dès la 1ère requête ou tout de suite après un *COMMIT* ou un *ROLLBACK*
- Propriétés *READ ONLY* ou *READ WRITE*
- Degrés d'isolation

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*

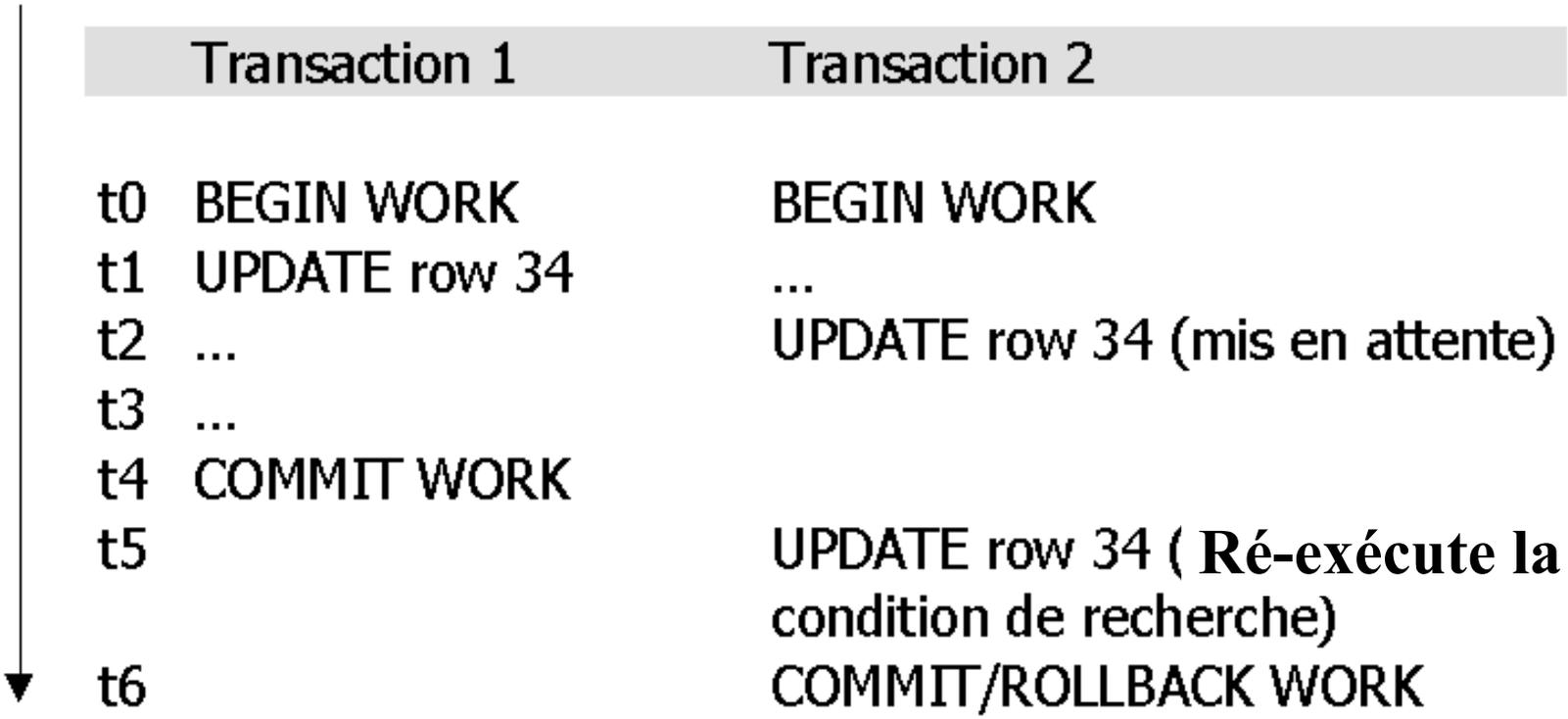
# Exemple de PostgreSQL

Pour le niveau d'isolation par défaut : READ COMMITTED



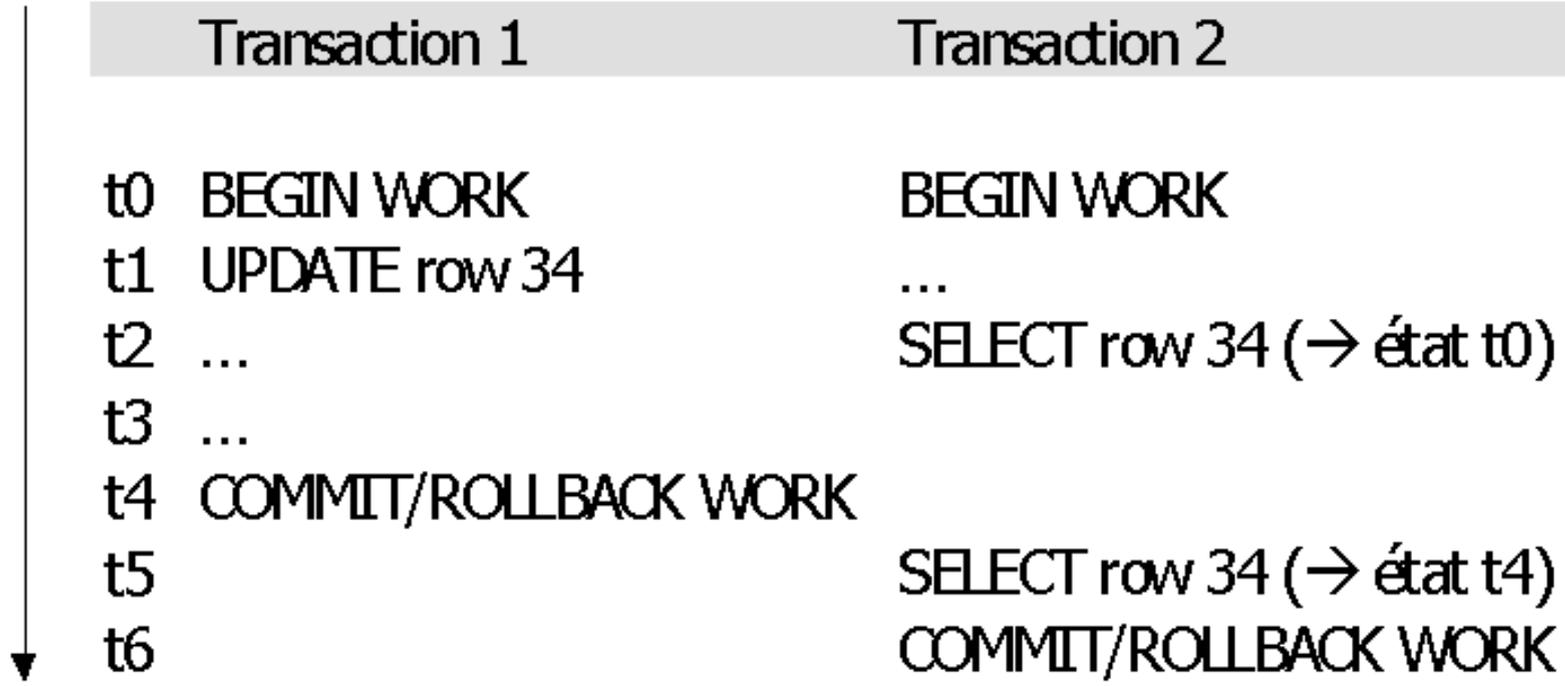
# Exemple de PostgreSQL

Pour le niveau d'isolation par défaut : READ COMMITTED



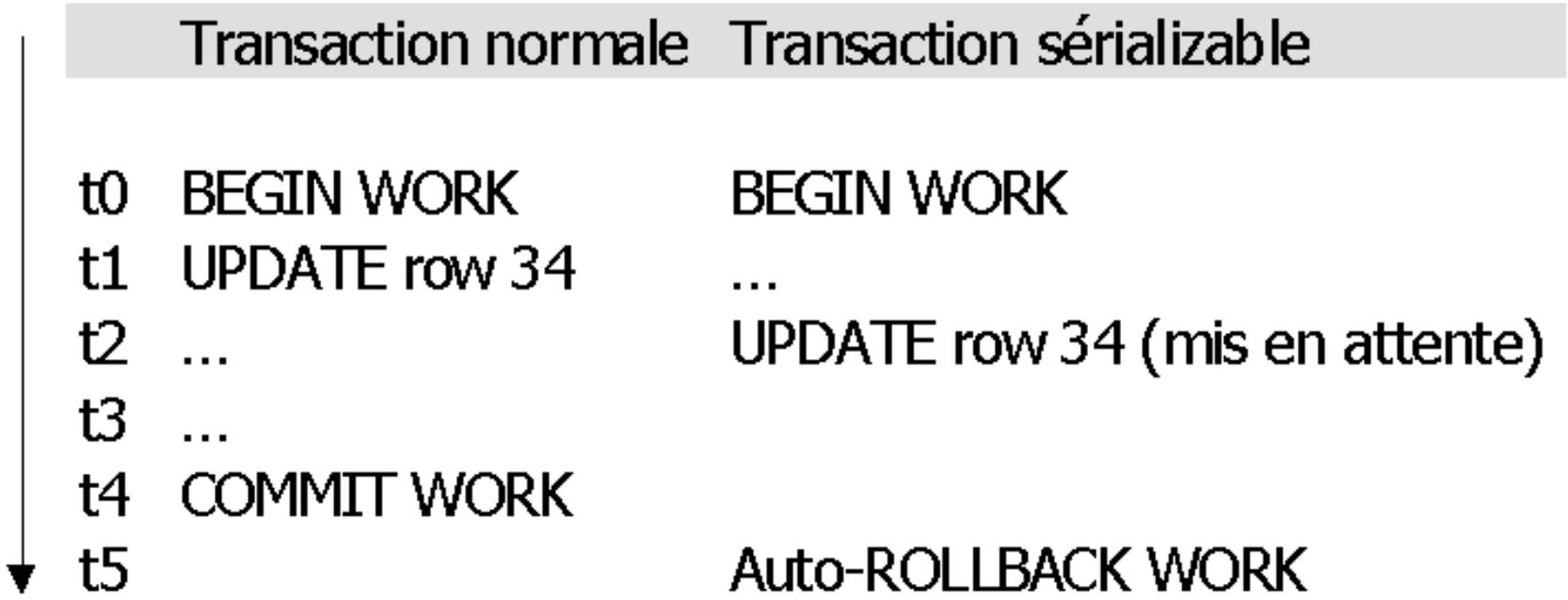
# Exemple de PostgreSQL

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



# Exemple de PostgreSQL

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



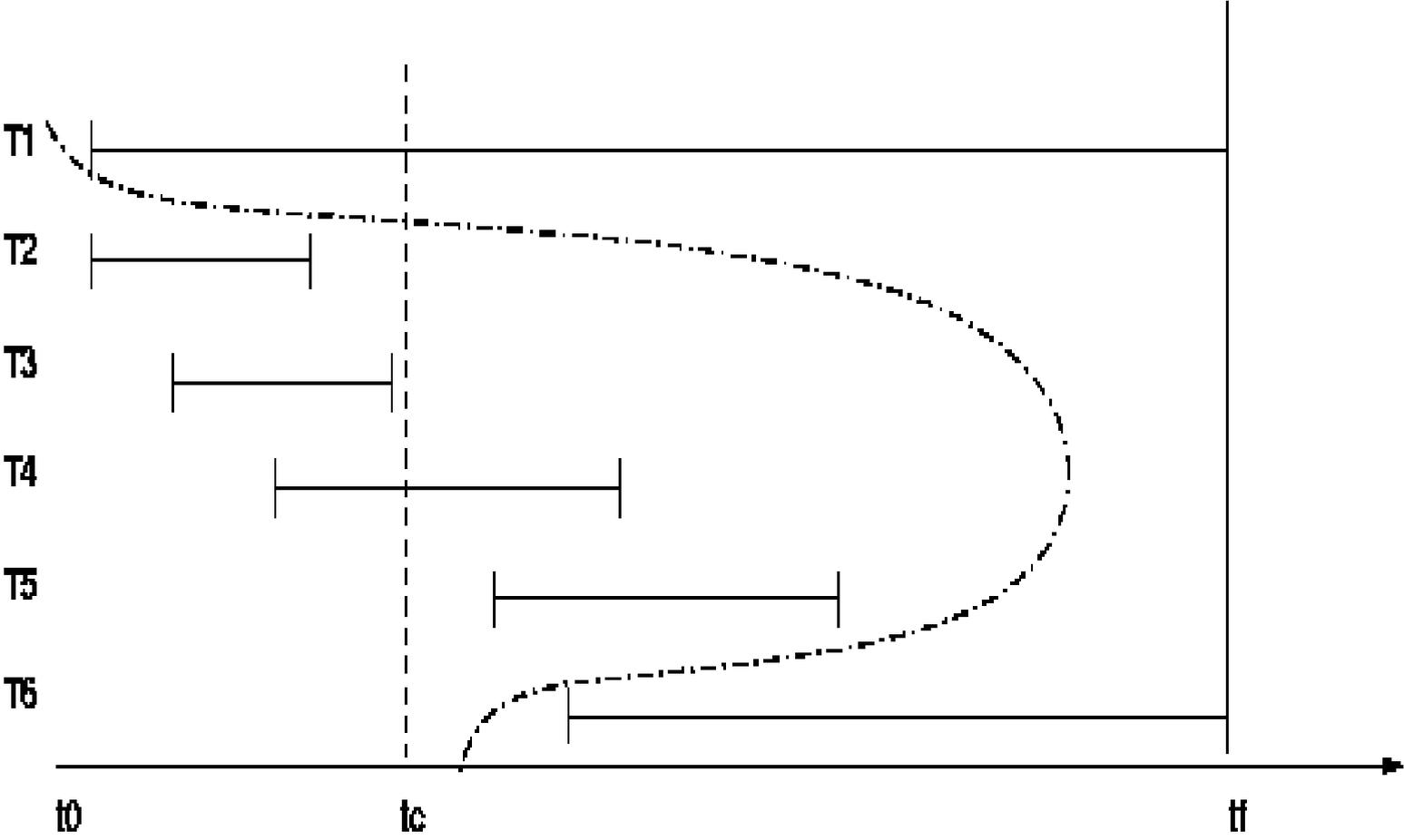
# **Chap. V - Reprise après panne**

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

# 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

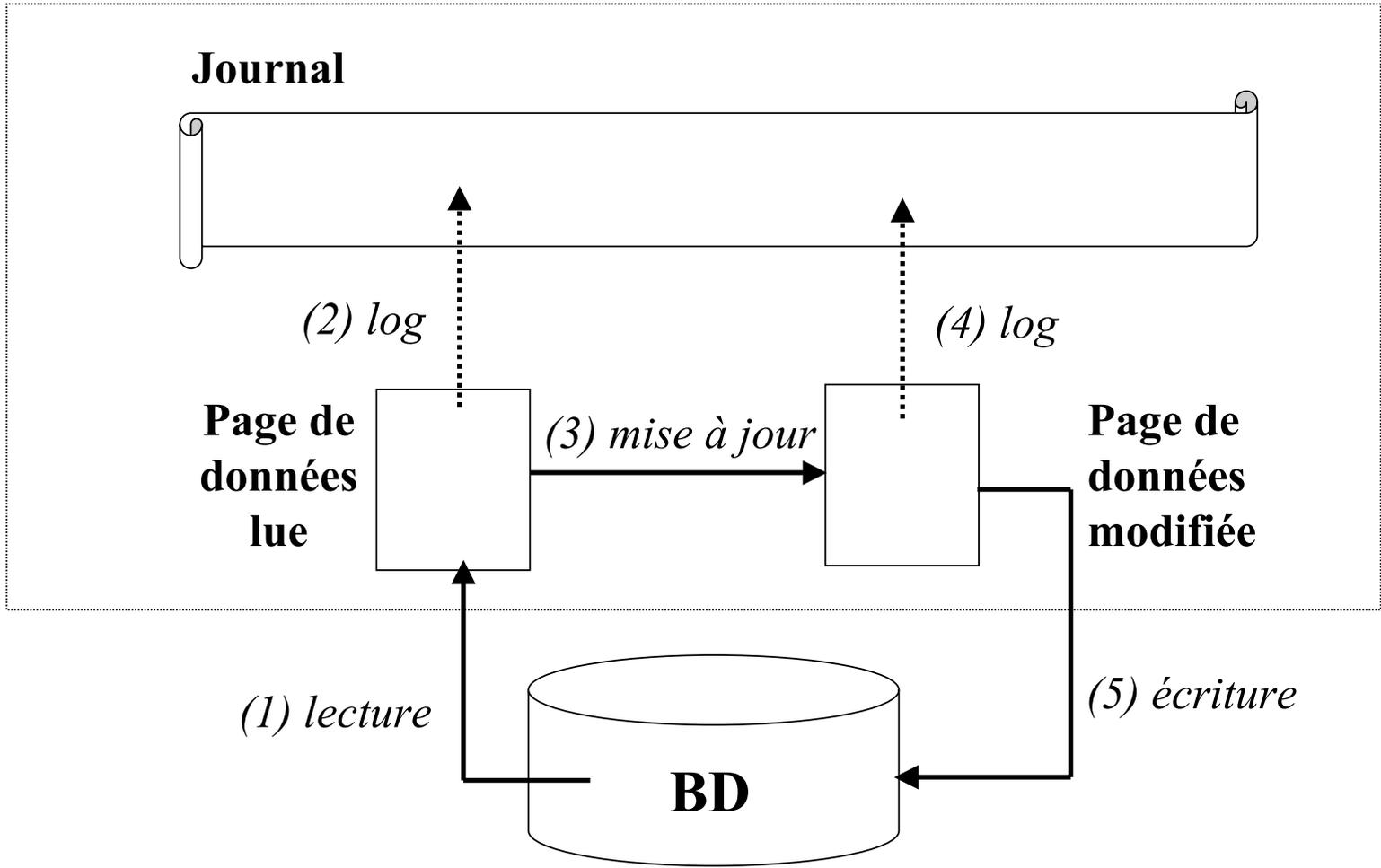
- **Journal des images après (*redo log*)**

- ◆ 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

# 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**