

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

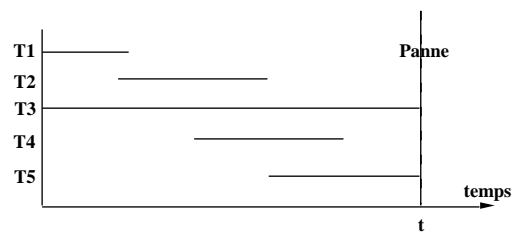
- Résistance aux pannes
- Sécurité de données
- Gestion de mémoire de travail

- 1 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.1- Principe de la résistance aux pannes : exemple



- Une panne se produit au temps t
 - les deux transactions T3 et T5 sont en cours d'exécution
 - les transactions T1, T2 et T4 s'étaient terminées correctement avant la panne
- Reprise :
 - Les effets de T1, T2 et T4 doivent survivre à la panne : "refaire" une partie ou totalement s'il y a un endommage de leurs modifications par la panne
 - Ceux de T3 et T5 doivent être éliminés par l'action "défaire"

- 2 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.1- Principe de la résistance aux pannes : types de pannes

- **PANNES D'UNE ACTION**
Les pannes surviennent quand une commande au SGBD est mal exécuté. En général, une telle panne est détectée par système qui retourne un code d'erreur au programme d'application
- **PANNES D'UNE TRANSACTION**
Les pannes surviennent quand une transaction ne peut continuer par suite d'une erreur de programmation, d'un verrou mortel ou d'un mauvais ordonnancement des accès concurrents, d'une panne d'action non corrigéable
- **PANNES DU SYSTÈME**
Les pannes systèmes nécessite l'arrêt du système et son redémarrage. La mémoire secondaire n'est pas affectée par ce type de panne. La mémoire centrale est effacée
- **PANNES DE LA MÉMOIRE SECONDAIRE**
Les pannes de la mémoire socondaire peut survenir soit suite à une défaillance matérielle, soit suite à une défaillance matérielle, soit suite à une défaillance logicielle impliquant de mauvaises écritures. Une partie de la mémoire secondaire est perdue

- 3 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.1- Principe de la résistance aux pannes : objectifs et notions

- **OBJECTIFS**
 - Minimiser le travail perdu tout en assurant un retour à des données cohérentes après pannes
 - fournir un protocole aux applications permettant de FAIRE, REFAIRE ou DÉFAIRE une transaction. Ces protocoles constituent 3 actions atomiques de SGBD
- **NOTION DE VALIDATION DE TRANSACTION**
La validation d'une transaction est confirmée par l'exécution d'une action atomique, appelée COMMIT, généralement en fin de transaction, provoquant l'intégration définitive de toutes les mises à jour effectuées par la transaction dans la bases
- **NOTION D'ANNULATION DE TRANSACTION**
L'annulation d'une transaction est confirmée par l'exécution d'une action atomique, appelée ABORT, généralement après une panne, provoquant l'annulation de toutes les mises à jour effectuées par la transaction dans la bases
- **NOTION DE REPRISE DE TRANSACTION**
La reprise d'une transaction est l'annulation de cette transsaction en exécutant une action atomique, appelée RECOVERY, qui refait les mises à jour de cette transaction dans la base

- 4 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.1- Principe de la résistance aux pannes : outils

- **MÉMOIRE SÛRE (stable Memory)** : Mémoire découpée en page dans laquelle une écriture de page est soit correctement exécutée, soit non exécuté
- **JOURNAL DES IMAGES AVANT (Before Image Log)**
Fichier système contenant d'une part les valeurs (image) avant modifications des pages modifiées, dans l'ordre des modifications avec les identifiants des transactions modifiantes, et d'autre part des enregistrements indiquant les débuts, validation et annulation de transactions
- **JOURNAL DES IMAGES APRÈS (After Image Log)**
Fichier système contenant d'une part les valeurs (image) après modifications des pages modifiées, dans l'ordre des modifications avec les identifiants des transactions modifiantes, et d'autre part des enregistrements indiquant les débuts, validation et annulation de transactions

Ordre d'enregistrement

Enregistrement d'un Journal

(Tampon)	↓ Journal	IDT	IDT	= Identifiant de transaction
(Base)		HMO	HMO	= Heure de la modification
		FMO	FMO	= Fichier modifié
		ADP	ADP	= Adresse de la page modifiée
		IMA	IMA	= Valeur avant modification
		IMP	IMP	= Valeur après modification

- 5 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.1- Principe de la résistance aux pannes : outils

- **Sauvegarde (Backup copy)**
Copie cohérente d'une base locale effectuée périodiquement alors que cette base est dans un état cohérent
- **Point de reprise système (System checkpoint)**
Etat d'avancement du système sauvegardé sur mémoires secondaires à partir duquel il est possible de repartir après un arrêt.
 - Les informations sauvegardées sur disque comportent en général l'image de la mémoire, l'état des travaux en cours, les pointeurs courants sur des fichiers séquentiels ...
 - Un enregistrement "point de reprise système" est écrit dans le journal. Lors d'une reprise, on repart en général du dernier point de reprise système
 - Il existe des études de la fréquence optimum des points de reprise (Gelenbe 79)

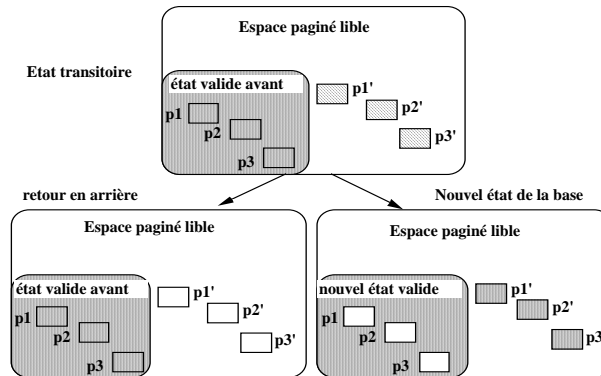
- 6 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.2- Validation et annulation de transactions

- Principe de la validation et l'annulation par le mécanisme de l'écriture à double



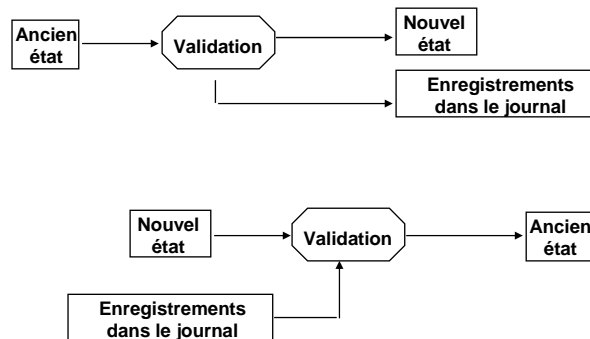
- 7 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.2- Validation et annulation de transactions

- Principe de la validation et l'annulation avec journal (GRAY81)



- 8 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

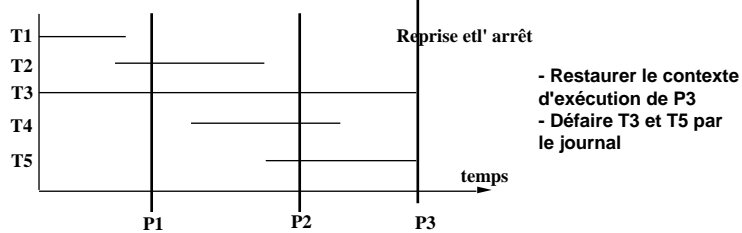
1.3- Procédures de reprise

- Procédure de reprise (Recovery procedure)

Procédure système exécutée lors du redémarrage du système ayant pour objectif de reconstruire une base cohérente aussi proche que possible de l'état attendu lors de la panne ou de l'arrêt

- La reprise normale

La reprise normale a lieu après un arrêt normal de la machine, dans lequel un point de reprise système a été écrit comme dernier enregistrement du journal. La reprise consiste à restaurer le contexte d'exécution sauvegardé lors de ce point de reprise



- 9 -

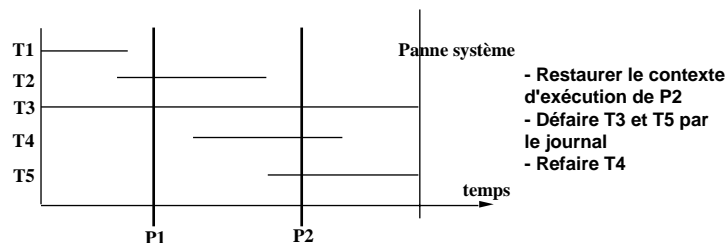
V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.3- Procédures de reprise

- La reprise après une panne du système (Reprise à chaud)

La reprise à chaud est effectuée après une panne système qui entraîne la perte de la mémoire centrale sans perte de données sur mémoires secondaires. À partir du contexte du dernier point de reprise, on défait les transactions "perdantes" (non terminées au moment de panne), refait les transactions "gagnantes" (a été validée après le point de reprise et avant la panne)



- 10 -

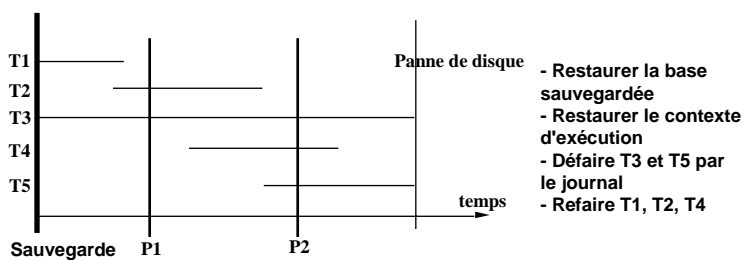
V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

1- RÉSISTANCE AUX PANNES

1.3- Procédures de reprise

- La reprise après une panne de mémoire secondaire (Reprise à froid)

La reprise à froid est exécutée lorsqu'une partie des données est perdue, ou lorsque la base de données est devenue incohérente. On commence à partir de la dernière sauvegarde et refait les transactions gagnantes jusqu'au dernier point de reprise



- 11 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

2- SECURITÉ DE DONNÉES

2.1- Identification et Autorisation : Concepts

- Objectifs : Assurer que les opérations non autorisées ou mal intentionnées doivent être rejetées
- Sujet et Objet : Une opération doit être effectuée par un SUJET (utilisateur par exemple) et elle doit appliquer sur certains objets de la base (les données)
- Identification : Procédé constant à associer à un sujet un nom ou un numéro qui le désigne de manière unique (compte d'utilisateurs par exemple)
- Authentification : Procédé permettant de vérifier qu'un sujet est bien qui prétend être (Mot de passe par exemple)
- Autorisation : Droit d'exécution d'une opération par un sujet sur un objet
L'attribut d'une autorisation peut dépendre :
 - du sujet : ses privilèges d'accès, le terminal utilisé
 - de l'objets : son nom, son état actuel, son contenu ou sa valeur
 - de l'opération effectuée : lecture, écriture

- 12 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

2- SECURITÉ DE DONNÉES

2.2- Identification et Autorisation : Contrôle

- Mécanisme de matrice d'autorisation

Une matrice d'autorisations est une matrice dont les lignes correspondent aux sujets et les colonnes aux objets, définissant pour chaque couple "sujet-objet" les opérations autorisées

- En pratique, la matrices d'autorisation peut être stockée :

- par ligne : à chaque sujet est alors associée la liste des objets auxquels il peut accéder ainsi que les droits d'accès qu'il possède
- par colonne : à chaque sujet est alors associée la liste des sujets pouvant l'accéder avec les droits d'accès associés
- par élément : à chaque couple "sujet-objet" sont associée les droits d'accès du sujet sur l'objet

- Niveau d'autorisation

Un niveau d'autorisation est un nombre associé à chaque objet ou à chaque sujet tel que un accès est autorisé ssi le niveau du sujet accédant est supérieur ou égal au niveau de l'objet accédé

- 13 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

2- SECURITÉ DE DONNÉES

2.2- Identification et Autorisation : Contrôle

- Exemple (Gardarin 84)

Sujets : ETUDIANT - SECRETAIRE - PROFESSEUR

Objet : NOM - RESULTAT

Atribut de droit LECTURE - ECRITURE

- Matrice d'autorisation :

	ETUDIANT	SECRETAIRE	PROFESSEUR
NOM	10	11	11
RESULTAT	10	10	11

- Niveau d'autorisation : ETUDIANT = 1 - SECRETAIRE = 2 - PROFESSEUR = 3
NOM = 1 - RESULTAT = 3

La matrice d'autorisation équivalente est :

	ETUDIANT	SECRETAIRE	PROFESSEUR
NOM	11	11	11
RESULTAT	00	00	11

- 14 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

2- SECURITÉ DE DONNÉES

2.3- Autres types de contrôle de sécurité

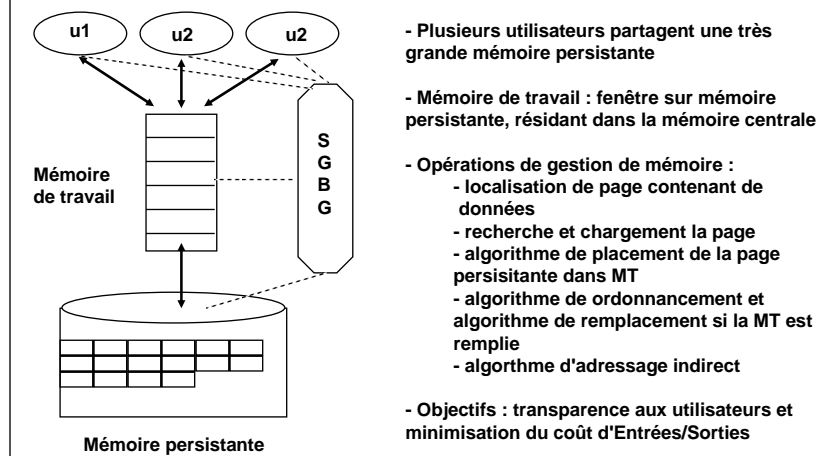
- **Contrôle du flux de l'information** : Les contrôles de flux consistent à surveiller les chemins qu'emprunte l'information afin qu'elle n'arrive entre les mains de personnes mal intentionnées.
 - Exemple : X a droit sur l'objet O, mais pas Y. X crée l'objet O' qui est une copie de l'objet O et donne le droit à Y sur l'objet O'
 - Technique de contrôle : les objets du système sont regroupés par classe et à chaque classe est associé un niveau d'autorisation
- **Contrôle d'inférence** : Les contrôles d'inférence ont pour but d'éviter qu'un utilisateur déduise, à partir de données auxquelles il peut accéder, des informations qu'il ne doit pas connaître (par un calcul statistique par exemple)
 - Technique de contrôle : imposer la limite de cardinalité d'un résultat
- **Cryptographie** : La cryptographie a pour but de stocker ou de transporter l'information sous une forme telle que seuls les utilisateurs en possession du code sont susceptibles de la comprendre
 - Technique de contrôle : Codage / Décodage des données

- 15 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.1- Principe et objectifs



- 16 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.1- Principe et objectifs

Gestionnaire de MT

- facteur essentiel dans la détermination de la performance du SGBD

- rôles de GMT :
- allouer, libérer, remplacer des pages persistantes en MT
 - protéger mutuellement des données appartenant aux différentes transactions
 - coordonner le partage des données entre transactions
 - fournir la possibilité de reprise sur pannes

Fonctionnalités de GMT

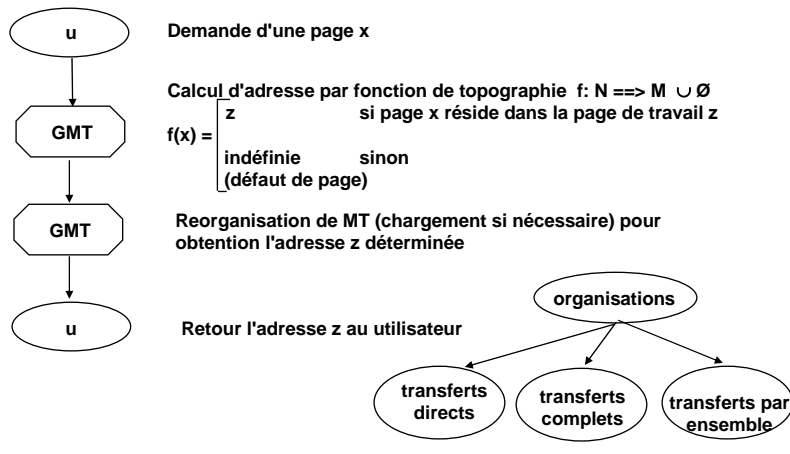
- distribuer les pages persistantes en MT
-> problème de l'organisation des transferts
- allouer la MT entre plusieurs transactions
-> méthode d'allocation
- déterminer le moment de chargement
-> mécanisme d'optimisation d'E/S
- placer la page persistante dans la MT
-> technique de placement
- déterminer la page à sacrifier et le moment de sacrifice
-> stratégie de remplacement de pages

- 17 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.2- Organisation des transferts



- 18 -

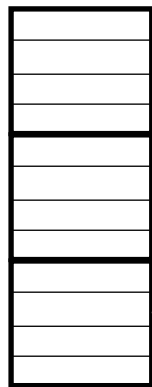
V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

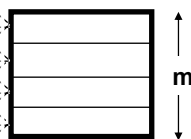
3.2- Organisation des transferts

Transferts directs

$$f(x) = (x \text{ mod } m) \rightarrow \text{Partitionnement de MP}$$



Mémoire persistante



Mémoire de travail

- Simple : maximum une page par partition se présente dans MT

- Si le temps de transferts de pages est négligeable, l'organisation est la moins coûteuse et la plus rapide (souvent utilisé dans la gestion de Mémoire Cache de OS)

- Inconvénients : E/S même si MT, est pas plein, technique aveuglée, besoin de configuration adéquate

- souvent utilisée dans les OS
- aucun SGBD l'utilise

- 19 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

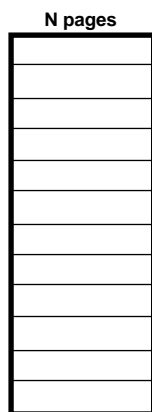
3- Gestion de mémoire de travail

3.2- Organisation des transferts

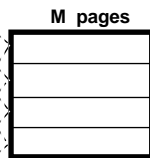
Transferts complets

Correspondance complète de N à M

$$f(x) = z, x \in [0, \dots, N-1], z \in [0, \dots, M-1]$$



Mémoire persistante



Mémoire de travail

- simple : on peut charger une page x à une adresse z libre

- Pas de remplacement de page s'il y a des pages libres dans MT

- liberté pour la mise en oeuvre un mécanisme de remplacement

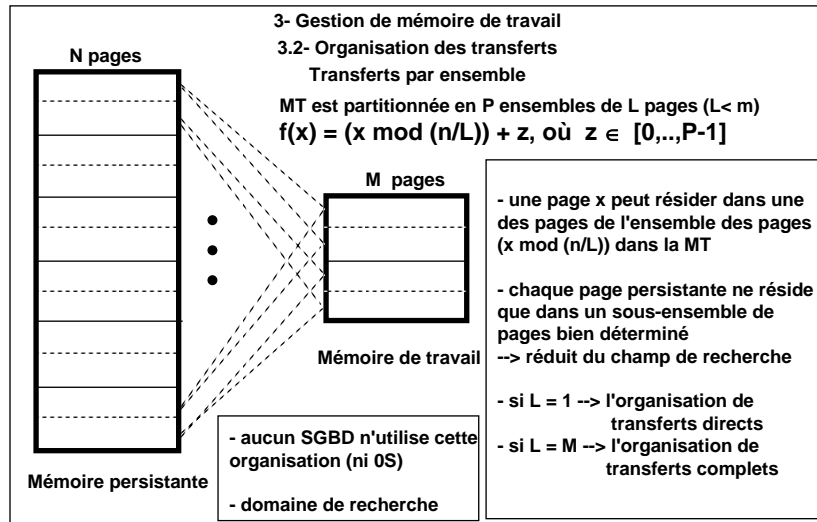
- La plupart des SGBD utilise cette organisation

- la MT, dans ce cas, se présente comme une pile des pages

- la recherche l'existence d'une page est couteuse
- le comportement du SGBD n'est pas pris en compte

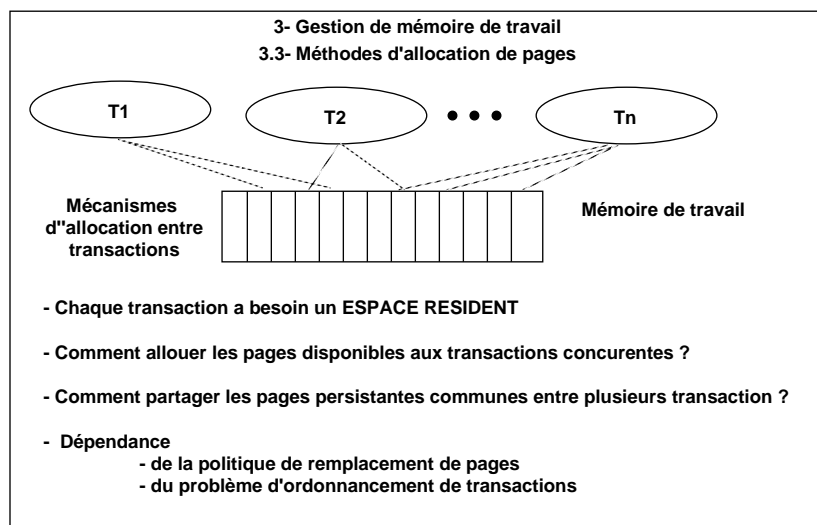
- 20 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES



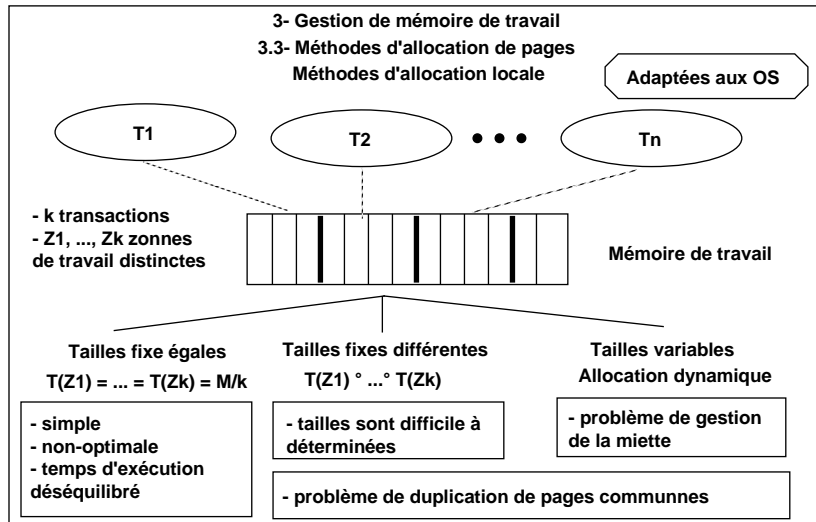
- 21 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

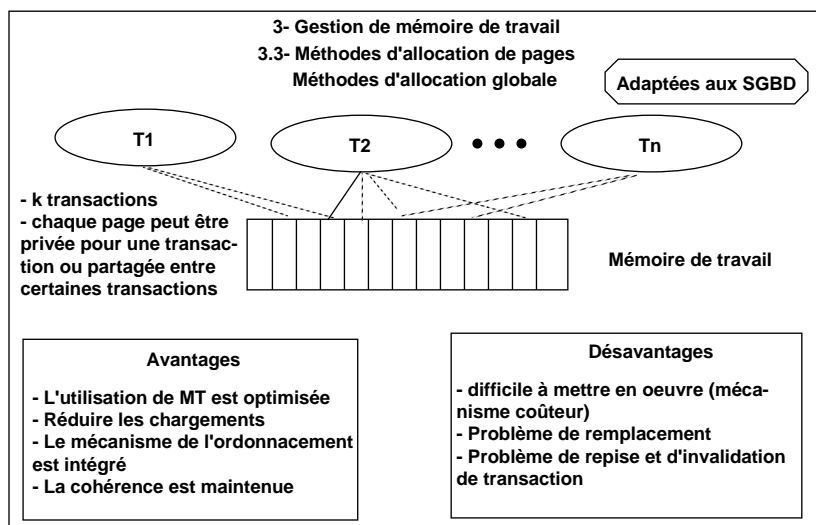


- 22 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES



V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

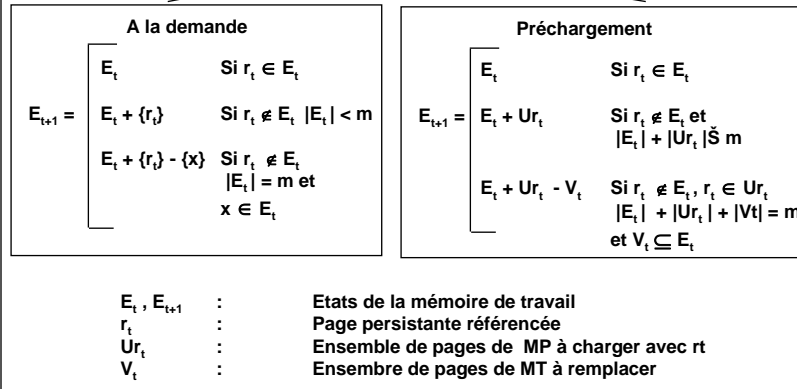


V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.4- Mécanismes de chargement

- Quand il faut charger une page persistante ?



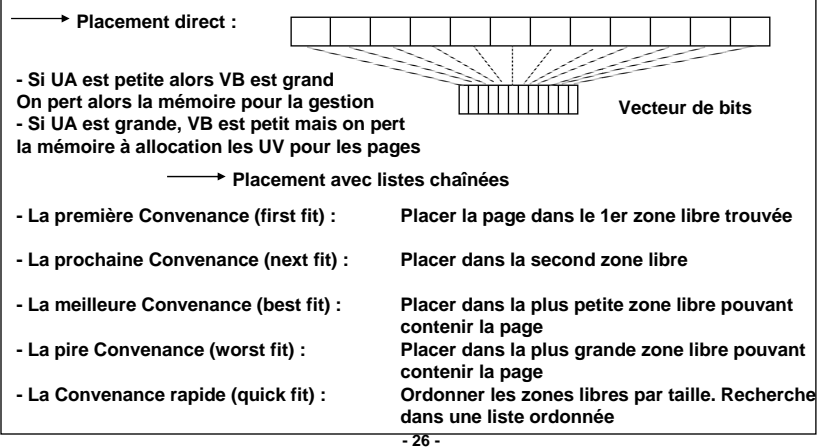
- 25 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.5- Techniques de Placement de pages de taille variable

Comment gérer la position des unités d'allocation dans la mémoire de travail ?



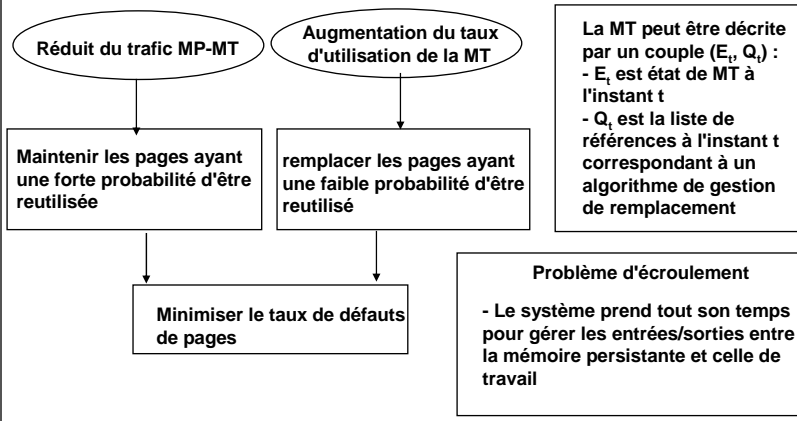
- 26 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Quelles sont les pages à sacrifier s'il y a un défauts de pages dans MT ?



- 27 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Algorithme RAND [BELA86]

$$(E_{t+1}, Q_{t+1}) = G_{\text{RAND}}(E_t, Q_t, r_t)$$

$$E_{t+1} = \begin{cases} E_t & \text{Si } r_t \in E_t \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ E_t + \{r_t\} - \{p_k\} & \text{Si } r_t \notin E_t \\ & |E_t| = m \text{ et } p_k \in E_t \end{cases} \quad Q_{t+1} = \begin{cases} Q_t & \text{Si } r_t \in E_t \\ \{p_1, \dots, p_{k-1}, r_t\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ \{p_1, \dots, p_{k-1}, r_t, p_{k+1}, \dots, p_m\} & \text{Si } r_t \notin E_t, |E_t| = m \\ & \text{et } p_k \in E_t \end{cases}$$

Description

- Toutes les pages sont distribuées et référencées avec une même probabilité
- Remplacement aléatoire :
 - générer un nombre aléatoire k entre 1 et m
 - remplacer la page à la position k en MT par la nouvelle page

- 28 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Algorithme FIFO (First-In-First-Out) [BELA86]

$$(E_{t+1}, Q_{t+1}) = G_{FIFO}(E_t, Q_t, r_t)$$

$$E_{t+1} = \begin{cases} E_t & \text{Si } r_t \in E_t \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ E_t + \{r_t\} - \{p_m\} & \text{Si } r_t \notin E_t \\ & |E_t| = m \text{ et} \end{cases} \quad Q_{t+1} = \begin{cases} Q_t & \text{Si } r_t \in E_t \\ \{r_t, p_1, \dots, p_j\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ \{\{r_t, p_1, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_{m-1}\}\} & \text{Si } r_t \notin E_t, |E_t| = m \end{cases}$$

Description

- Organisation comme une file d'attente simple :
 - les pages sont classées par l'ordre de son temps arrivés,
 - la page en tête est la dernière arrivée, et celle en queue est la première arrivée
- Remplacement : la page de la queue sort de la MT, la page chargée prend la tête

- 29 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Algorithme LRU (Least-Recently-Used) [MATT 70]

$$(E_{t+1}, Q_{t+1}) = G_{LRU}(E_t, Q_t, r_t)$$

$$E_{t+1} = \begin{cases} E_t & \text{Si } r_t \in E_t \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ E_t + \{r_t\} - \{p_m\} & \text{Si } r_t \notin E_t \\ & |E_t| = m \end{cases} \quad Q_{t+1} = \begin{cases} Q_t & \text{Si } r_t \in E_t \text{ et } r_t = p_1 \\ \{r_t, p_1, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_{m-1}\} & \text{Si } r_t \in E_t \text{ et } r_t = p_m \\ \{r_t, p_1, \dots, p_{k-1}, p_{k+1}, \dots, p_m\} & \text{Si } r_t \in E_t \text{ et } r_t = p_k \\ \{r_t, p_1, \dots, p_j\} & \text{Si } r_t \notin E_t \text{ et } |E_t| < m \\ \{r_t, p_1, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_{m-1}\} & \text{Si } r_t \notin E_t, |E_t| = m \end{cases}$$

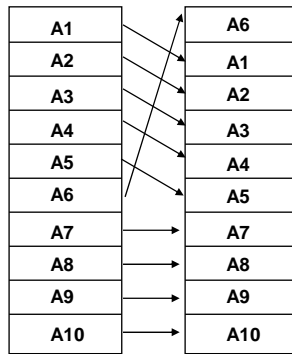
Description

- Les pages récemment référencées ont plus de chance d'être re-référencées
- La MT organisation comme une pile dont les éléments sont classés par l'ordre de son dernier moment de référence : la dernière référence est au sommet, la moins récemment référencée est au fond de pile
- Lors d'une référence à une page, si celle-ci est dans la MT, elle sera placée en tête de la pile. Si un défaut de page se produit, la page chargée est placée au sommet de la pile et si la MT est pleine, la page au fond de la pile est sacrifiée

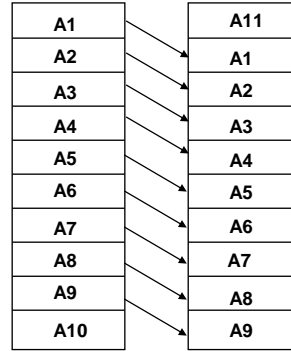
- 30 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages



Référence à une page dans MT



Référence à une page hors de MT

- 31 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages

Algorithme LFU Least-Frequently-Used
(E_{t+1}, Q_{t+1}) = $G_{LRU}(E_t, Q_t, r_t)$

- Chaque page a
un Compteur de
Fréquences (CR)

$$E_{t+1} = \begin{cases} E_t & \text{Si } r_t \in E_t \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t, |E_t| < m \\ E_t + \{r_t\} - \{p_m\} & \text{Si } r_t \notin E_t, |E_t| = m \end{cases} \quad Q_{t+1} = \begin{cases} \{r_t, p_2, \dots, p_k, \dots, p_m\} & \text{Si } r_t = p_1, CR(r_t)+1 > CR(p_2) \\ \{p_1, \dots, p_k, \dots, p_{m-1}, r_t\} & \text{Si } r_t = p_m, CR(r_t)+1 < CR(p_{m-1}) \\ \{r_t, p_1, \dots, p_{k-1}, r_t, p_{k+1}, \dots, p_m\} & \text{Si } r_t \in E_t \text{ et } \\ & \exists k (1 < k < m) CR(p_k) < CR(r_t)+1 < CR(p_{k+1}) \\ \{p_1, \dots, p_k, r_t\} & \text{Si } r_t \notin E_t, |E_t| < m \\ \{p_1, \dots, p_k, \dots, p_{m-1}, r_t\} & \text{Si } r_t \notin E_t, |E_t| = m \end{cases}$$

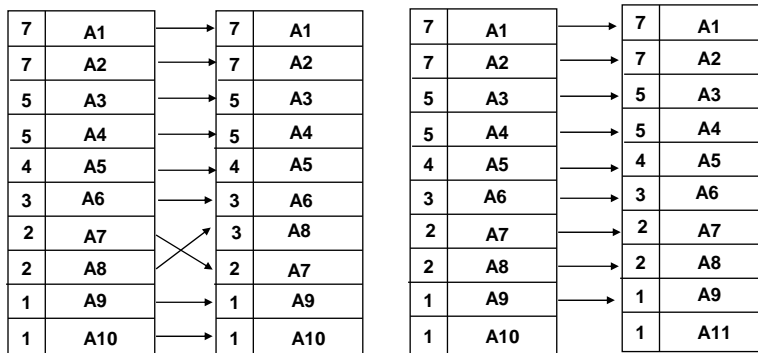
Description

- Les pages plus fréquemment référencées ont plus de chance d'être re-férencées
- La MT organisation comme une pile dont les éléments sont classés par l'ordre de fréquence de référence: la moins fréquentée est au fond, la plus fréquentée est au sommet de pile
- Lors d'une référence à une page, si celle-ci est dans la MT, son compteur CR est incrémenté et elle sera placée à la place correspondante à cette valeur. Si un défaut de page se produit, le compteur de la page chargée est initialisé à 1, et elle sera placée au fond de la pile. Si la MT est pleine, la page au fond de la pile sera sacrifiée

- 32 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages



Référence à une page dans MT (A8)

Référence à une page hors de MT(A11)

- 33 -

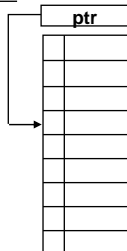
V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages Algorithme CLOCK (seconde chance)

$$(E_{t+1}, Q_{t+1}) = G_{\text{CLOCK}}(E_t, Q_t, r_t)$$

$$E_{t+1} = \begin{cases} E_t & \text{Si } r_t \in E_t \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t, |E_t| < m \\ E_t + \{r_t\} - \{p_m\} & \text{Si } r_t \notin E_t, |E_t| = m \end{cases}$$

$$Q_{t+1} = \begin{cases} Q_t & \text{Si } r_t \in E_t \\ \{p_1, \dots, p_{k-1}, r_t\} & \text{Si } r_t \notin E_t, |E_t| = k < m \\ \{p_{j+1}, \dots, p_m, p_1, p_{k-1}, p_k, p_{k+1}, \dots, p_{j-1}, r_t\} & \text{Si } r_t \notin E_t, |E_t| = m, \text{ptr} = k, \text{bit}(i) = 1, \text{bit}(j) = 0 \\ & (0 < k \leq i < j \leq m) \\ \{p_{j+1}, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_m, p_1, \dots, p_{j-1}, r_t\} & \text{Si } r_t \notin E_t, |E_t| = m, \text{ptr} = k, \text{bit}(i) = 1, \text{bit}(j) = 0 \\ & (0 \leq i < j \leq m) \end{cases}$$



- Une variance de FIFO :

- organisation comme une liste circulante

- un pointeur ptr qui pointe sur élément suivant de la page la plus récente chargée

- chaque page est associée à un bit dont la valeur est à :

- 1 si la page n'est pas prête à sortir
- 0 si la page est à état de prêt à sortir

- 34 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

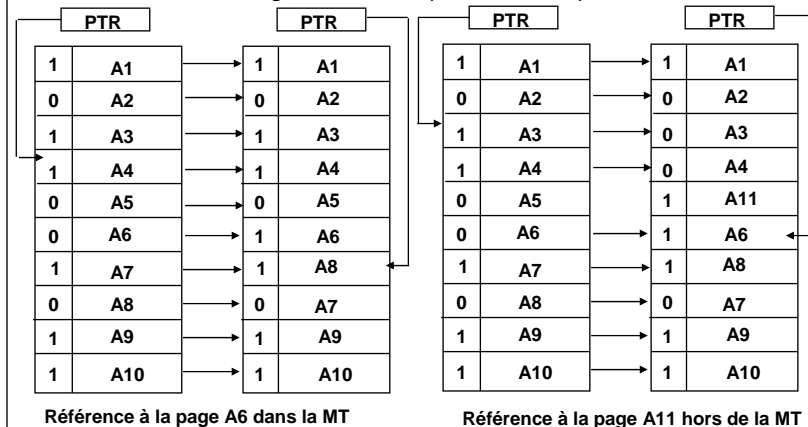
3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages Algorithme CLOCK (seconde chance) Description

- Les pages sont chargées dans MT seulement à la demande
 - Lors d'une référence à une page située dans MT, son bit de référence est mis à 1, et le pointeur PTR est pointé sur la page suivante dans la liste
 - Lors d'une référence à une page hors de la MT, s'il y a de la place libre (PTR point sur un emplacement libre), on charge la page dans cet emplacement, initialise le bit associé à 1 et PTR à l'emplacement suivant
 - Si la MT est pleine lors d'une référence : ARRET := FAUX
répéter
examiner la page PTR : si son bit de référence est à 0;
ARRET = VRAI; Affectuer le chargement;
mettre le bit de référence à 1;
sinon mettre le bit à valeur 0;
- PTR := PTR + 1;
Jusqu'à ARRET;

- 35 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail 3.6- Stratégie de remplacement de pages Algorithme CLOCK (seconde chance)



- 36 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Algorithme WS (Working Set)

$$(E_{t+1}(T), Q_{t+1}(T)) = G_{WS}(E_t(T), Q_t(T), r_t)$$

$$E_{t+1}(T) = \begin{cases} E_t(T) & \text{Si } r_t \in E_t(T) \text{ et } t-CT(p_i) < T \\ E_t(T) - \{p_j\} & \text{Si } r_t \in E_t(T) \text{ et } t-CT(p_j) = T \\ E_t + \{r_t\} & \text{Si } r_t \notin E_t(T) \text{ et } t-CT(p_i) < T \\ E_t + \{r_t\} - \{p_j\} & \text{Si } r_t \notin E_t(T) \text{ et } t-CT(p_j) = T \end{cases}$$

$$Q_{t+1}(T) = \begin{cases} Q_t(T) & \text{Si } r_t \in E_t(T) \text{ et } t-CT(p_i) < T \\ \{p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_{w(t,T)}\} & \text{Si } r_t \in E_t(T) \text{ et } t-CT(p_j) = T \\ \{p_1, \dots, p_{w(t,T)}, r_t\} & \text{Si } r_t \notin E_t(T) \text{ et } t-CT(p_i) < T \\ \{p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_{w(t,T)}, r_t\} & \text{Si } r_t \notin E_t(T) \text{ et } t-CT(p_j) = T \end{cases}$$

- A chaque page est associé un compteur CT

- Si la page référencée $\in W(t,T)$, son compteur s'est mis à t. Toute page y de $W(t,T)$, telle que $t-CT(y) = T$ sera évacuée de $W(t,T)$

- Sinon, on évacue toutes les pages y de $W(t,T)$ telles que $t-CT(y) = T$. La page demandée sera chargée et son compteur s'est mis à t

- **Chaque transaction** a un espace de travail $W(t,T)$ défini dans l'intervalle $[t-T, t]$: $W(t,T) = \{p \in N \mid p \in \{r_{t-T}, r_{t-T+1}, \dots, r_t\}\}$. Le nombre d'éléments à l'instant t est noté par $w(t,T)$

- Lorsqu'un défaut de page, une page présente peut être remplacée ssi elle n'a pas été référencée dans les T références précédentes

- 37 -

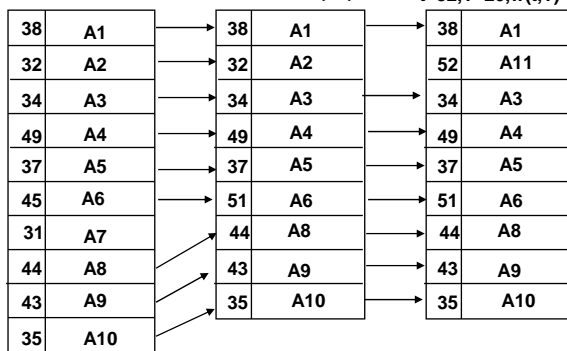
V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages

Algorithme WS

$t=50, T=20, w(t,T)=15$ $t=51, T=20, w(t,T)=14$ $t=52, T=20, w(t,T)=14$



référence à la page A6

référence à la page A11

- 38 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages Algorithme PFF(Page Fault Frequency)

- Basé sur le principe de minimiser la fréquence de défaut de pages
 - Soit P un taux de défaut de pages autorisé entre (0-1). Soit t' l'instant du dernier défaut lié à une transaction. Soit ptr un pointeur sur la page produisant le dernier défaut de pages. Soit $V_i(t',P)$ les pages demandées avant l'instant t' (t' inclus). L'espace de travail est organisé comme une pile classé par l'ordre des instants de références décroissants.
 On a : $V_i(t',P) = E_i(P) - \{r_{t-t'+1}, r_{t-t'+2}, \dots, r_j\} = \{r_j, r_{j+1}, \dots, r_{|E_i(P)|}\}$ (avec $ptr = j$ à l'instant t') et :

$$(E_{t+1}(P), Q_{t+1}(P)) = G_{PFF}(E_i(P), Q_i(P), r_t)$$

$$E_{t+1}(P) = \begin{cases} E_i(P) & \text{Si } r_t \in E_i(P) \\ E_i(P) + \{r_t\} & \text{Si } r_t \notin E_i(P) \text{ et } t+1-t' \leq 1/P \\ E_i(P) + \{r_t\} - \{V_i(t',P)\} & \text{Si } r_t \notin E_i(P) \text{ et } t+1-t' > 1/P \end{cases}$$

$$Q_{t+1}(P) = \begin{cases} Q_i(P) & \text{Si } r_t \in E_i(P) \text{ et } r_t = p_i \\ \{r_t, p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_{|E_i(P)|}\} & \text{Si } r_t = p_i \in E_i(P) \\ \{r_t, p_1, \dots, p_{|E_i(P)|}\} & \text{Si } r_t \notin E_i(P) \text{ et } t+1-t' \leq 1/P \\ \{r_t, p_1, \dots, p_{j-1}\} & \text{Si } r_t \notin E_i(P) \text{ et } t+1-t' > 1/P \text{ et } ptr = j \end{cases}$$

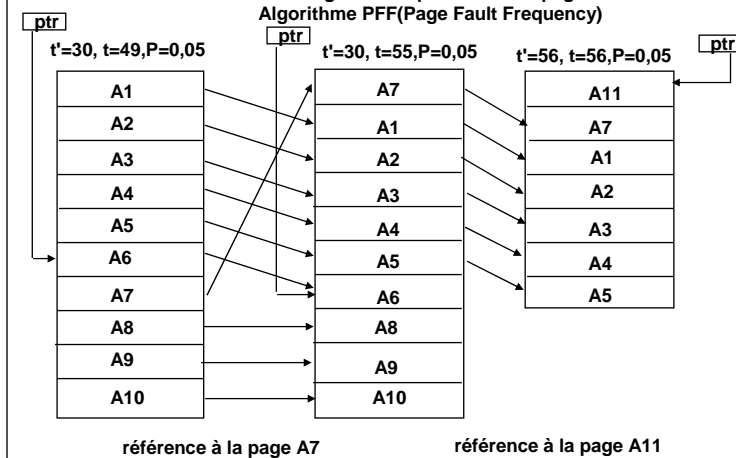
- Si la page demandée se situe dans MT, elle se place à la tête de la pile
 - Si elle n'est pas dans MT, et si $t+1-t' \leq 1/P$ alors on décale la pile vers le fond. Si $t+1-t' > 1/P$ les pages au dessus de la page ptr sont décalées vers le fond, les autres pages sont évaluées de MT. La page demandée est chargée au sommet

- 39 -

V- SÉCURITÉ DE DONNÉES ET RÉSISTANCE AUX PANNES

3- Gestion de mémoire de travail

3.6- Stratégie de remplacement de pages Algorithme PFF(Page Fault Frequency)



- 40 -

VI- ÉVALUATION DE REQUÊTES

- Concepts de base
- Analyse de requêtes
- Ordonnement relationnel
- Ordonnement par décomposition

- 41 -

VI- ÉVALUATION DE REQUÊTES

1- CONCEPTS DE BASE

- Evaluation de requête : Algorithme d'accès optimisé (ou optimal) du système pour obtenir le résultat d'une requête utilisateur
- Phases d'évaluation :
 - Analyse de requête : qui consiste à étudier syntaxiquement, et parfois sémantiquement, la requête de sorte à vérifier sa correction et à simplifier le critère de recherche
 - Ordonnement d'opérations élémentaires : qui consiste à décomposer la requête en une séquence d'opérations élémentaires et à déterminer un ordre plus ou moins optimal de ces opérations. Le parallélisme entre certaines opérations est possible
 - Exécution des opérations élémentaires : qui consiste à faire exécuter en parallèle et/ou en séquence les opérations élémentaires dans le plan d'exécution, afin d'élaborer le résultat de la requête
- Plan d'exécution : (Request schedule)
Programme parallèle d'opérations élémentaires à exécuter pour évaluer le résultat d'une requête

- 42 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.1- Analyse syntaxique

- Objectifs :
 - Contrôle lexical, syntaxique : mots clés, la syntaxe des expressions, ...
 - Contrôle des objets -référéncés : l'existence des attributs, des relations cités dans la requête,
 - La correction de la qualification de la requête : la mise en forme normale conjontive (ET de OU) ou disjontive (OU de ET) selon le besoin de l'opération élémentaire concernée
 - Création d'un premier arbre syntaxique

- 43 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Objectifs et Outils

- Objectifs :
 - Détermination de la correction de la requête
 - Recherche de requêtes équivalentes par manipulation de la qualification
 - Recherche de requêtes équivalentes à l'aide des contraintes d'intégrité
- Exemple illustré (Gardarin 85) : une BD composée de quatre relations
ABUS (NOM, NV, QUANTTTITÉ) PRODUCTEUR (NP, NOM, REGION)
VINS (NV, MILLÉSIME, DEFREÉ) PRODUIT (NV, NP)
- Q1 : "quels sont les crus des vins produits par un producteur bordelais en 1975 ayant un degré inférieur ou égal à 14° ?"
- Q1 en QUEL :
RANGE OF P,V, R IS PRODUCTEUR, VINS, PRODUIT
RETIEVE V.CRU INTO W
WHERE (V.MILLESIME = "1976") ^ (V.DEGRÉ ≤ 14) ^
(P.REGION = "Bordeaulais") ^ (P.NP = R.NP) ^ (R.NV = V.NV)

- 44 -

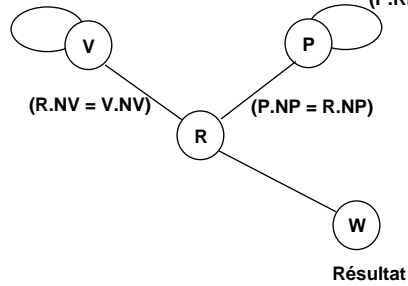
VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Objectifs et Outils

- Graphe de connexion des relations (Wong 76, Ullman 80) :
Graphe dans lequel un sommet est associé à chaque référence de relation, où une jointure est représentée par un arc entre les deux relations participantes et une restriction par une boucle sur la relation à laquelle elle s'applique

$(V.MILLESIME = "1976") \wedge (V.DEGRE \checkmark 14)$ $(P.REGION = "Bordealais")$



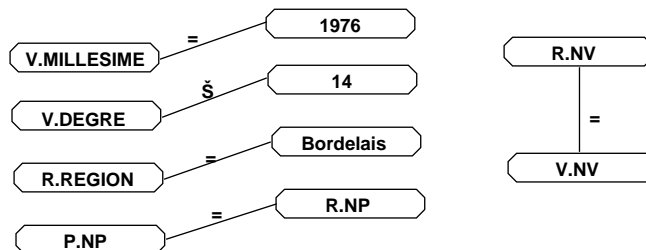
- 45 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Objectifs et Outils

- Graphe de connexion des attributs (Hevner 79) :
Graphe dans lequel un sommet est associé à chaque référence d'attribut ou de constante, où une jointure est représentée par un arc entre les attributs participants et une restriction par un arc entre un attribut et une constante



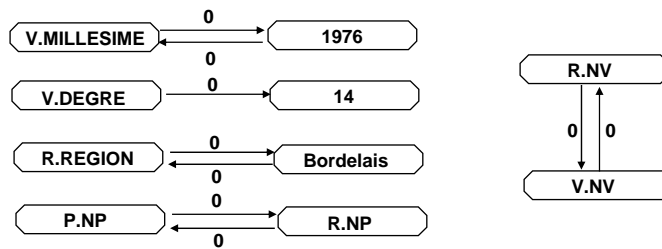
- 46 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Objectifs et Outils

- Graphe de connexion des attributs normalisé (Rosenkrantz 80) :
Graphe de connexion des attributs orienté dans lequel les arcs sont valués par des entiers. Un arc du sommet x vers le sommet y valué par entier c représente l'inégalité $x \leq y + c$. Une égalité de type $x = y$ est représentée par deux arcs valués par 0, de x à y et de y à x



- 47 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Principes de Correction

- Requetes incorrectes
Une requête est incorrecte si :
 - soit elle est mal formulée car certaines parties apparaissent inutiles dans la requête (exemple : si l'utilisateur oublie une jointure)
 - soit elle est contradictoire car la qualification ne peut être satisfaite par aucun tuple (exemple : $(V.DEGRE \leq 12) \wedge (V.DEGRE \leq 14)$)
- Théorème 1 (Wong 76) :
Une requête est mal formulée si son graphe de connexion des relations n'est pas connexe
- Théorème 2 (Rosenkrantz 80) :
Une requête est contradictoire si son graphe normalisé de connexion des attributs présente un cycle dont la somme des valuations est négative

- 48 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Requêtes équivalentes par transitivité

- **Requêtes équivalentes**
Deux requêtes sont équivalentes si et seulement si elles donnent le même résultat pour toute extension possible de la base de données
- **Fermeture transitive du graphe normalisé de connexion des attribut**
Considérons le graphe normalisé de connexion des attributs. On appelle R une relation binaire définie comme la suivante :
pour tout couple d'attributs (x,y) xRy si et seulement si x et y, reliés par un chemin de x à y de somme de valuation c, vérifient $x \dot{S} y + c$. R est une relation transitive : si xRy et yRz alors xRz .
On appelle FT(R) la fermeture transitive de R.
- **Théorème de requêtes équivalentes (Gomez 78, Bernstein 79)**
Deux requêtes sont équivalentes si les fermetures transitives de leur graphe normalisé (R1 et R2 respectivement) sont identiques
$$FT(R1) = FT(R2)$$

- 49 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Requêtes équivalentes par transitivité

- **Exemple (Gardarin 85) :**
Q2 : Quels sont les noms des buveurs ayant bu un Bordeaux de degré supérieur ou égale à 14
- R1) RANGE OF A, R, P, V IS ABUS, PRODUIT, PRODUCTEUR, VINS
RETRIEVE A.NOM
WHERE (A.NV = R.NV) \wedge (R.NV = V.NV) \wedge (R.NP = P.NP) \wedge
(P.REGION = "Bordelais") \wedge (V.DEGRE 14)
- R2) RANGE OF A, R, P, V IS ABUS, PRODUIT, PRODUCTEUR, VINS
RETRIEVE A.NOM
WHERE (A.NV = R.NV) \wedge (A.NV = V.NV) \wedge (R.NP = P.NP) \wedge
(P.REGION = "Bordelais") \wedge (V.DEGRE 14)
- La différence entre R1 et R2 est le choix des jointures

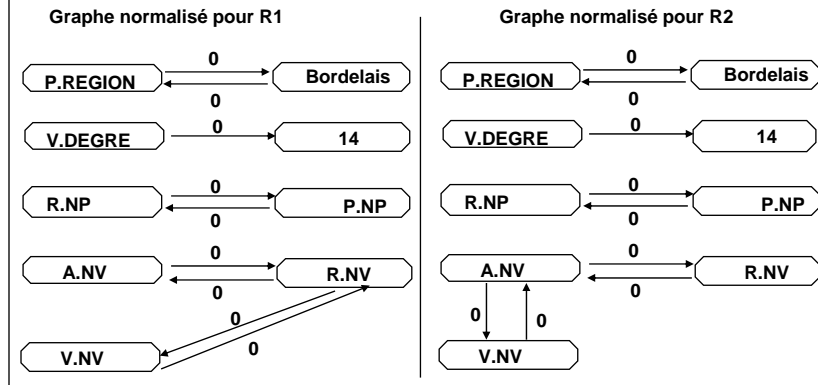
- 50 -

VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Requetes équivalentes par transitivité

- Exemple (Gardarin 85) :



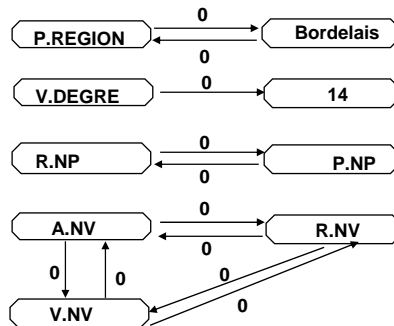
VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Requetes équivalentes par transitivité

- Exemple (Gardarin 85) :

Fermeture transitive des Graphes normalisés pour R1 et R2



VI- ÉVALUATION DE REQUÊTES

2- ANALYSE DES REQUÊTES

2.2- Analyse sémantique : Requêtes équivalentes par intégrité

- Principe (Grant 80 - Hammer 80 - King 81 - La Chimia 82) : Soient
 - une requête de qualification Q
 - un ensemble de contraintes d'intégrité I1, ..., In.
- Si Q est contradictoire à une contrainte Ij, la réponse à cette requête est vide.
- Sinon, on cherche une qualification Q' "meilleure" que Q telle que :

$$I1 \wedge I2 \wedge \dots \wedge In \wedge Q' \rightarrow Q$$
- Exemple (Gardarin 85)
 - Si on a

$$(A.NV = R.NV) \wedge (R.NV = V.NV) \wedge (R.NP = P.NP) \wedge (P.REGION = "Bordelais") \Rightarrow (V.DEGRE > 14)$$
 - Alors :
 - Q1 a une réponse vide (requête contradictoire)
 - Q2 peut être remplacée par Q2' dont la qualification est "simplifiée" par la suppression de (V.DEGRE > 14)

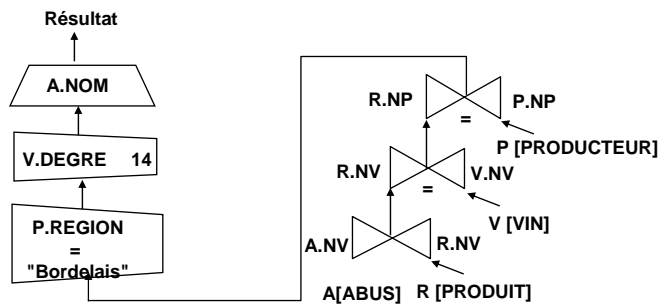
- 53 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.1- Arbre algébrique

- Arbre algébrique (Relational algebra tree) :
Arbre représentant une requête dont les noeuds terminaux représentent les relations, les noeuds intermédiaires des opérations de l'algèbre relationnelle, le noeud racine le résultat de la requête, et les arcs les flux de données entre les opérations
- Exemple (Gardarin 85) : un arbre algébrique pour Q2



- 54 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.1- Arbre algébrique

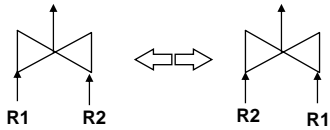
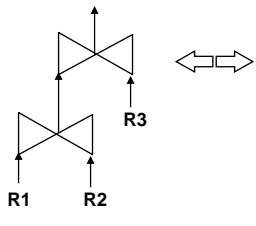
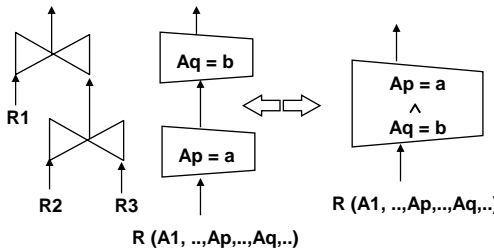
- **Plan d'exécution algébrique :**
Plan d'exécution généré à partir d'un arbre algébrique en parcourant l'arbre, des feuilles vers la racine. Une opération peut être exécutée dès que ses opérandes sont disponibles, et si l'opération A1 n'utilise pas les résultats de l'opération A2, A1 et A2 peuvent être exécutées en parallèle
- **Optimiser simultanément :**
 - Le nombre d'opérations d'Entrées / Sorties
 - Le parallélisme entre les Entrées / Sorties
 - La taille des tampons nécessaires à l'exécution
 - Le temps unité central nécessaire
- **Génération du plan d'exécution optimal :**
Optimisation dépend essentiellement de l'ordre des opérations apparaissant dans l'arbre algébrique. Il est nécessaire d'établir des règles permettant de générer, à partir d'un arbre initial, tous les arbres possibles afin de choisir celui conduisant au meilleur plan d'exécution (souvent par heuristique pour éviter la complexité)

- 55 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- **R1 : Commutativité des jointures :**

- **R2: Associativité des jointures :**

- **R3 : Regroupement des restrictions**


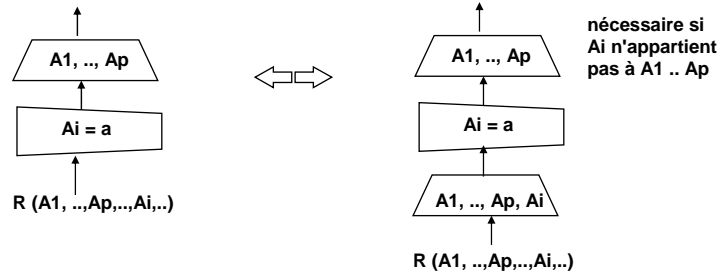
- 56 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- R4 : Commutation des restrictions et projections :



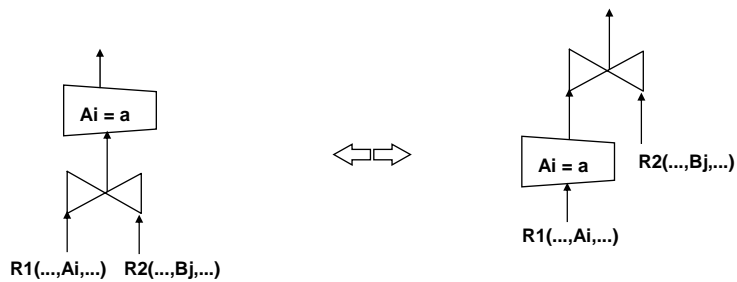
- 57 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- R5 : Commutation des restrictions et jointures :



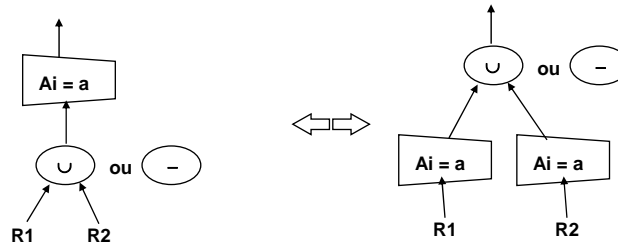
- 58 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- R6 : Commutation des restrictions et unions (ou différences) :



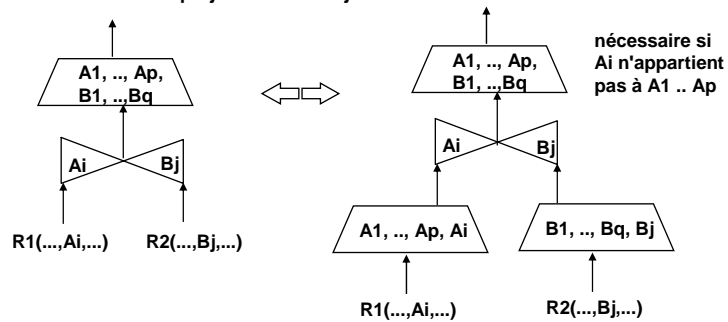
- 59 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- R7 : Commutation des projections et des jointures :



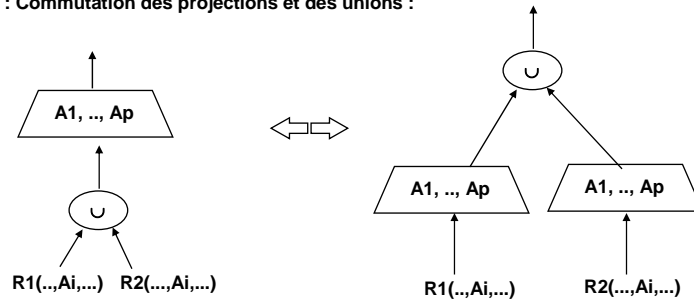
- 60 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.2- Règles de transformation des arbres [Ullman 80]

- R8 : Commutation des projections et des unions :



- 61 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.3- Optimisation par descente des opérateurs unaires [Ullman 80]

- Principe
 - 1- Séparer les restrictions comportant plusieurs prédicats à l'aide de la règle 3
 - 2- Descendre les restrictions aussi bas que possible à l'aide des règles 4, 5 et 6
 - 3- Regrouper les restrictions successives portant sur une même relation
 - 4- Descendre les projections aussi bas que possible à l'aide des règles 7 et 8
 - 5- Regrouper les projections successives en conservant les attributs restants et éliminer d'éventuelles projections inutiles qui auraient pu apparaître (ex Projection sur tous les attributs d'une relations)
- En règle générale, une restriction ou une jointure suivie par une projection sont exécutées simultanément par un même opérateur de sélection

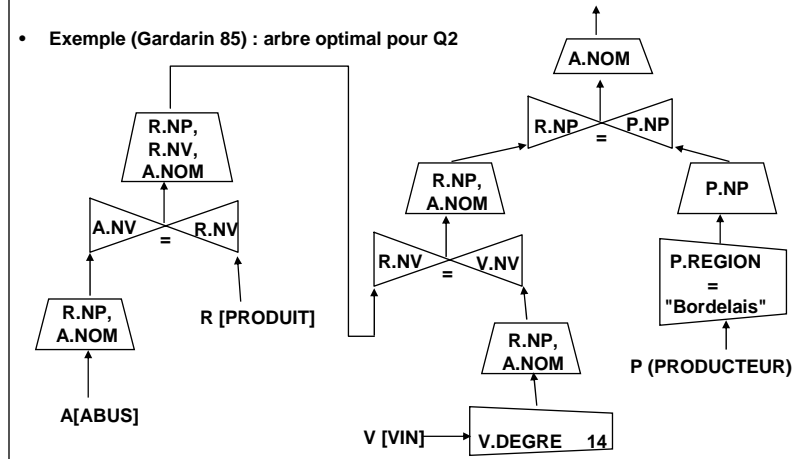
- 62 -

VI- ÉVALUATION DE REQUÊTES

3- ORDONNANCEMENT RELATIONNEL

3.3- Optimisation par descente des opérateurs unaires [Ullman 80]

- Exemple (Gardarin 85) : arbre optimal pour Q2



VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76)

4.1- Détachement des sous-requêtes

- Détachement**
Transformation de requête consiste à diviser une requête en deux sous-requêtes ayant une seule variable commune

- Condition du détachement :** Soit une requête Q de la forme

(Q)	RANGE OF (X1,...,Xn) IS (R1,...,Rn)
	RETRIEVE T (X1,...,Xm)
où B1, B2 sont des qualifications	WHERE B2 (X1,...,Xm) and
et T est le résultat de projection	B1 (Xm,...,Xn)

Cette requête peut être décomposée en deux requêtes succesives Q1, Q2 par détachement :

- | | | | |
|----|-------------------------------------|----|--------------------------------------|
| Q1 | RANGE OF (Xm,...,Xn) IS (Rm,...,Rn) | Q2 | RANGE OF (X1,...,Xm) IS (R1,...,R'm) |
| | RETRIEVE INTO R'm(T'(Xm)) | | RETRIEVE T(X1,...,Xm) |
| | WHERE B1 (Xm,...,Xn) | | WHERE B2 (X1,...,Xm) |

Où T'(Xm) contient les informations de Xm nécessaires au requête Q2

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76)

4.1- Détachement des sous-requêtes

- **Semi-jointure**
La semi-jointure de la relation R par la relation S, noté $R \bowtie S$, est une jointure de R et S projetée sur des attributs de R
 - Autrement dit, la semi-jointure de R par S est composée des tuples (ou partie de tuples) de R appartenant à la jointure avec S
 - On peut voir la semi-jointure de R par S comme une généralisation de la restriction de R sur les valeurs de l'attribut de jointure de S
- **Utilisation de détachement**
Le détachement permet de faire apparaître
 - les restrictions et
 - les semi-jointures

- 65 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76)

4.1- Détachement des sous-requêtes

- **Propriété 1**
Toutes les sélections sont détachable
 - Par détachement des sélections, on peut éliminer tous les boucles contenant un seule noeud (boucle de sélection)
- **Propriété 2 (Bernstein 79)**
Une condition suffisante et nécessaire pour qu'une requête soit détachable en séquence de semi-jointures (non nécessairement unique), est que son graphe de connexion des relations, après l'élimination des boucles de sélection, est un arbre
- **Requête irréductible**
Une requête est irréductible si l'on ne peut pas la décomposer par le détachement
- **Propriété 3**
Toute question irréductible présente des cycles dans son graphe de connexion des relations
 - Il peut exister, pour une requête irréductible, une requête équivalente décomposable

- 66 -

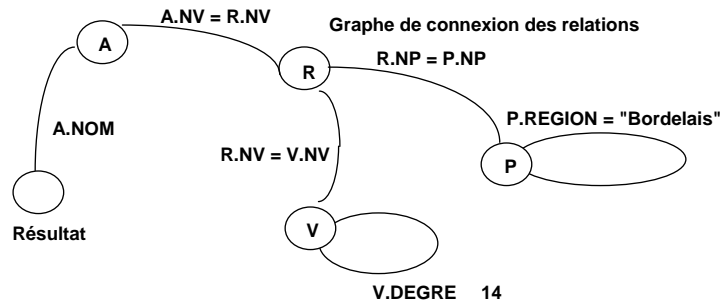
VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76) 4.1- Détachement des sous-requêtes

- Exemple : Requête Q2

```

RETRIEVE A.NOM
WHERE (A.NV = R.NV) ∧ (R.NV = V.NV) ∧ (R.NP = P.NP) ∧
      (P.REGION = "Bordelais") ∧ (V.DEGRE = 14)
    
```



- 67 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76) 4.1- Détachement des sous-requêtes

- Exemple : Requête Q2 :
- PHASE 1 : Détachement pour l'obtention des sélections :

Il existe deux sélections qui peuvent être détachées

- une sélection sur P pour avoir la variable P.NP

```

(Q21)  RETRIEVE P.NP INTO P'
        WHERE (P.REGION = "Bordelais")
    
```

- une sélection sur V pour avoir la variable V.NV

```

(Q22)  RETRIEVE V.NV INTO V'
        WHERE (V.DEGRE = 14)
    
```

- 68 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76)

4.1- Détachement des sous-requêtes

- Exemple : Requête Q2 :
- PHASE 2 : Détachement pour l'obtention des semi-jointures :
 - En remplaçant V par V' et P par P' et éliminant les sélections de Q2, on obtient
(Q'2) RETRIEVE A.NOM
WHERE (A.NV = R.NV) \wedge (R.NV = 'V.NV) \wedge (R.NP = P'.NP)
 - Il existe trois semi-jointures qui peuvent être détachées
 - une semi-jointure de R par P' pour avoir la variable R.NP et R.NV
(Q23) RETRIEVE R.NP, R.NV INTO R'
WHERE (R.NP = P'.NP)
 - une semi-jointure de R' par V' pour avoir la variable R'.NV
(Q24) RETRIEVE R'.NV INTO R''
WHERE (V'.NV = R'.NV)
 - une semi-jointure de A par R'' pour avoir la variable résultat A.NOM
(Q25) RETRIEVE A.NOM
WHERE (A.NV = R''.NV)

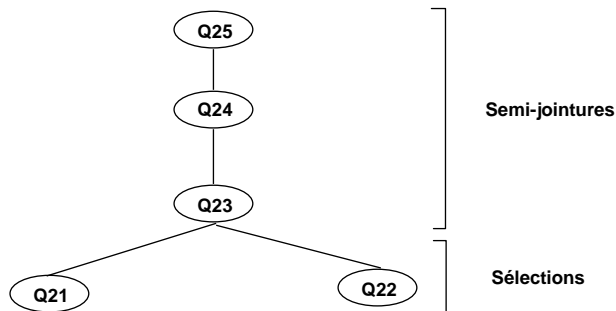
- 69 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76)

4.1- Détachement des sous-requêtes

- Exemple : Requête Q2 :
- PHASE 3 : Etablir l'arbre final d'exécution !
L'arbre d'exécution contient 2 sélections et 3 semi-jointures



- 70 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76) 4.2- Substitution de tuples

- Objectif :
La substitution de tuple est pour but de transformer une partie d'une requête irréductible en un sous-requête décomposable par détachement
- Algorithme SUBTUP (R : Requête)
Début
 - Choix d'une variable X de relation à balayer
 - Si l'arbre de connexion des relations R', en substituant les valeurs de X, est irréductible alors SUBTUP (R')
 - Sinon
Parcourir séquentiellement dans X
Pour chaque tuple x de X :
 - substituer la variable X par la valeur du tuple x
 - obtenir une sous-requête décomposable Rx
 - traiter cette requête par le détachement
 - traiter le résultat obtenu
 Fin Algorithme

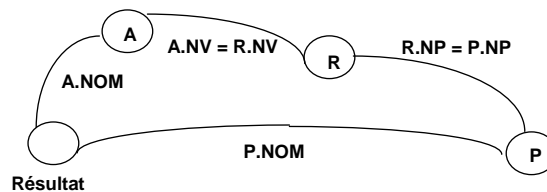
- 71 -

VI- ÉVALUATION DE REQUÊTES

4- ORDONNANCEMENT PAR DÉCOMPOSITION (Ingres - Wong 76) 4.2- Substitution de tuples

- Exemple (Gardarin 85) : Requête irréductible
R3 : Quels sont les noms de buveurs et les noms de producteurs de vins tels que le buveur ait bu un vin du producteur
- | | |
|----------|--------------------------------------|
| RANGE OF | A, R, P IS ABUS, PRODUIT, PRODUCTEUR |
| RETIEVE | A.NOM, P.NOM |
| WHERE | (A.NV = R.NV) \wedge (R.NP = P.NP) |

- Arbre de connexion des relations correspondant à Q3



- 72 -