Cours 2

La Gestion des Transactions

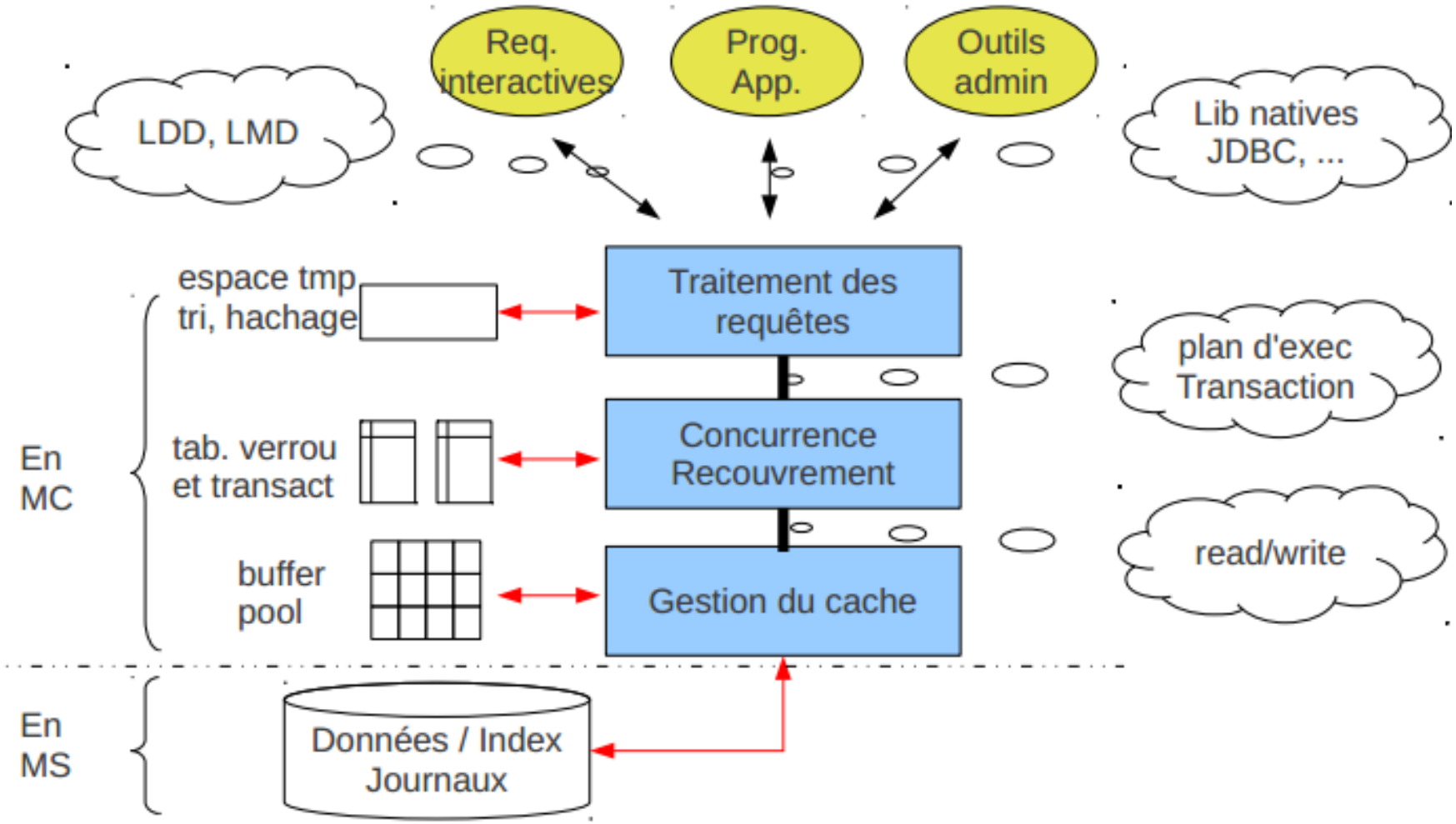
BDA
Master GL/RT - 2019- 2020

OUARED Abdelkader

Plan du cours

- Le concept de transaction
- Fonctionnement du système de base de données
- Contrôle de concurrence
- Reprise après une panne

Architecture système de base de données



Architecture système

Traitement de requêtes (Gestionnaire de transaction):

- Optimisation de requêtes (plan d'exécution)
- Petit espace de RAM pour réaliser des opérations < **join, tri** >

Contrôleur de concurrence:

- Envoie les opérations suivant un ordre sérialisable et strict

Gestionnaire de recouvrement :

- Gère le stockage et l'indexation
- Le Gestionnaire de recouvrement (GR) fait en sorte qu'il y ait toujours assez d'informations sur mémoire stable pour qu'une reprise correcte soit possible après l'occurrence d'une panne
  - **Les opérations du gestionnaire de recouvrement**
    read, *write*, *commit*, *rollback* et *restart* (en cas de reprise)
- **Retour en arrière:** Restaurer la BD dans le cas d'annulation de la transaction par user, ou le système(ABORT, COMMIT)

Architecture système

Gestionnaire de cache :

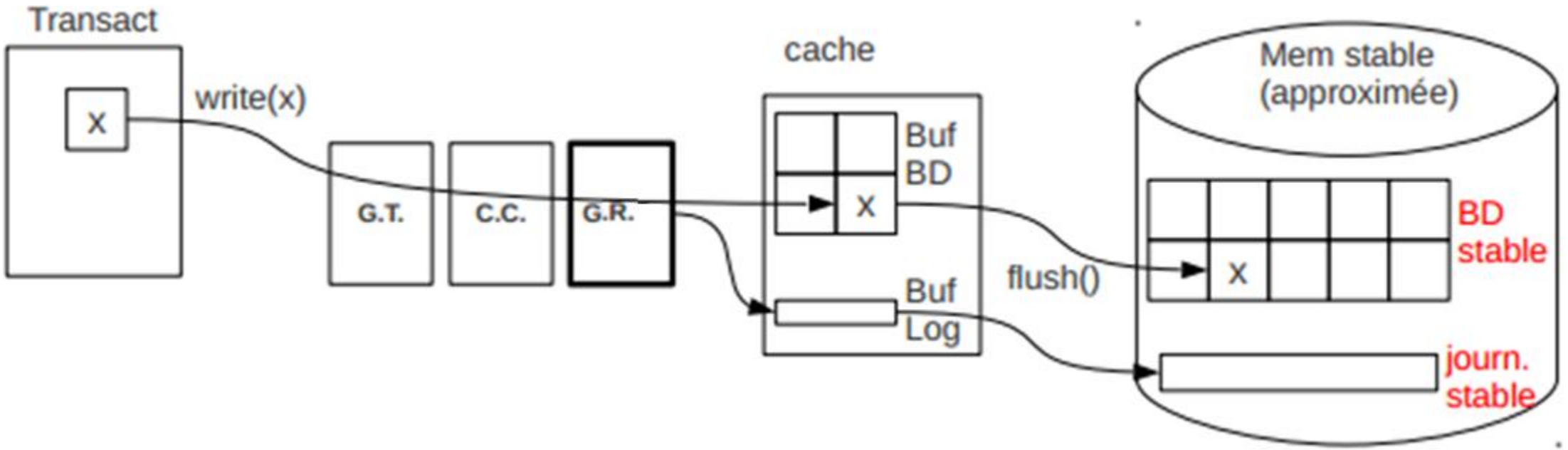
- La BD stable et le journal stable résident sur disque
- Les accès transitent par le cache en MC

- Synchronise la page de cache avec la page de la BD stable
- Choix de la page victime → besoin de l'espace !!

Les opérations du gestionnaire de cache

Fetch (pour obliger une lecture physique)
Flush (pour obliger une écriture physique)
Pin (pour bloquer une page dans le cache)
Unpin (pour débloquer une page du cache)

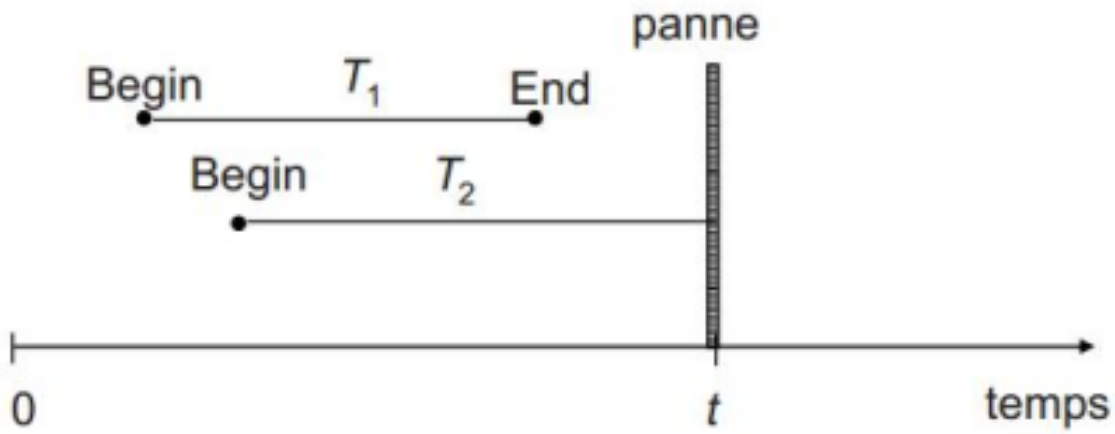
Accès aux données



Reprise après une panne

Pourquoi journaliser?

- Lors de la reprise



– toutes les mises-à-jour de T1 doivent être faites dans la BD

– aucune mise-à-jour de T2 ne doit être faite dans la BD

La journalisation

- Un journal est une séquence d'enregistrements décrivant les mises à jours effectuées par les transactions
- C'est l'historique d'une exécution sur fichier séquentiel
- Un journal est dit « **physique** » s'il garde la trace des modifications au niveau octet à l'intérieur des pages

  Ex: **<Ti, numPg, Depl, Long, img_avant, img_après>**

- Un journal est dit « **logique** » s'il garde la trace de la description de haut niveau des opérations de mise à jour

  Ex: « insérer le tuple x dans la table T et mettre à jour les index »

Utilisations du journal

Lors de l'annulation

remettre les images avant (Undo) en parcourant le journal en arrière

Lors de la validation

sauvegarder toutes les images après en mémoire stable (journal)

- optimisation : « group commit + precommit »

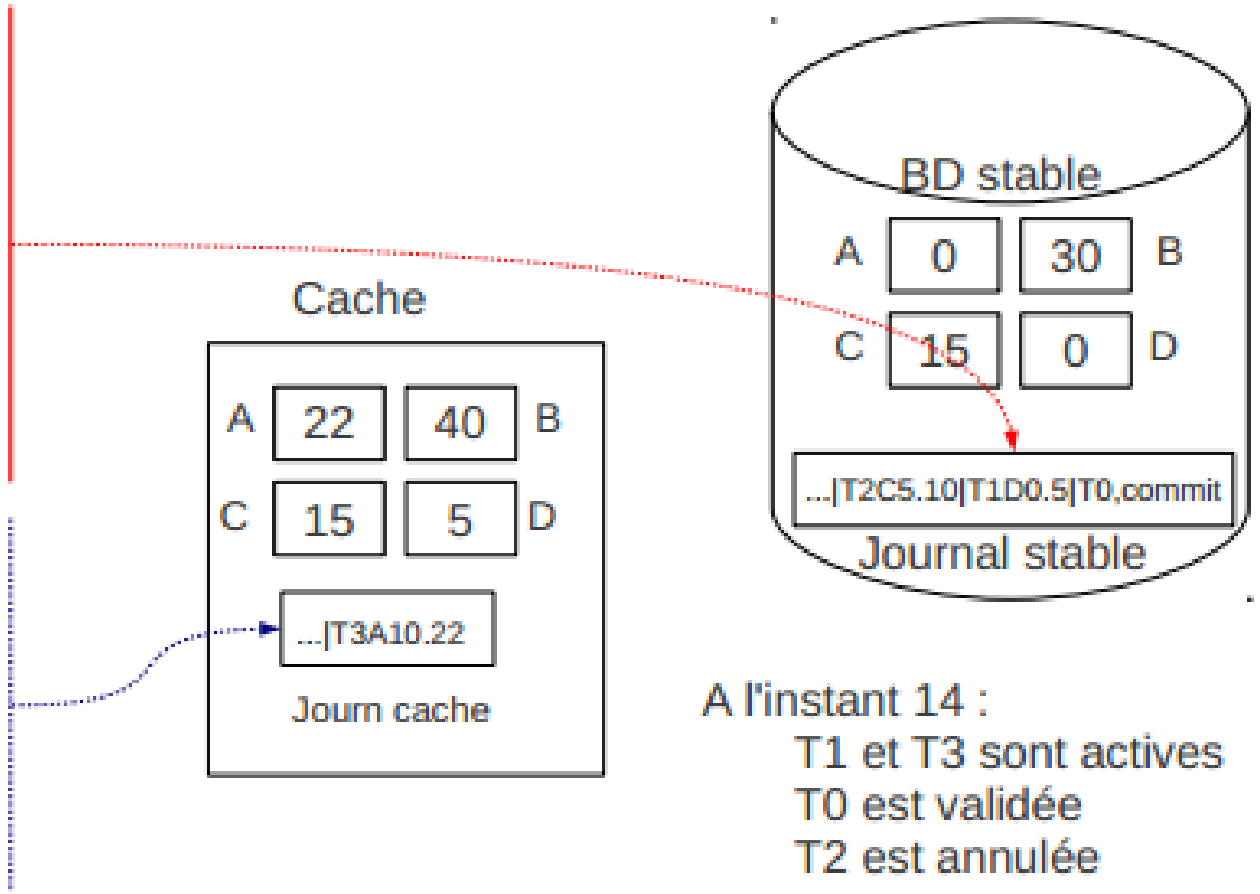
Lors d'une reprise après panne

Refaire l'historique en exécutant les opérations du journal du début jusqu'au moment de la panne (dernier enregistrement)

- très coûteux et beaucoup de travail inutile

Exemple

1 <T0, start>
2 <T0, A, 0, 10>
3 <T1, start>
4 <T0, B, 20,30>
5 <T2, start>
6 <T2, C,5 ,10>
7 <T1, D, 0, 5>
8 <T0, commit>
9 <T2, A, 10, 15>
10 <T1, B, 30, 40>
11 <T2, abort>
12 <T1, C, 5, 15>
13 <T3, start>
14 <T3, A, 10, 22>
15 ...



A l'instant 14 :
T1 et T3 sont actives
T0 est validée
T2 est annulée

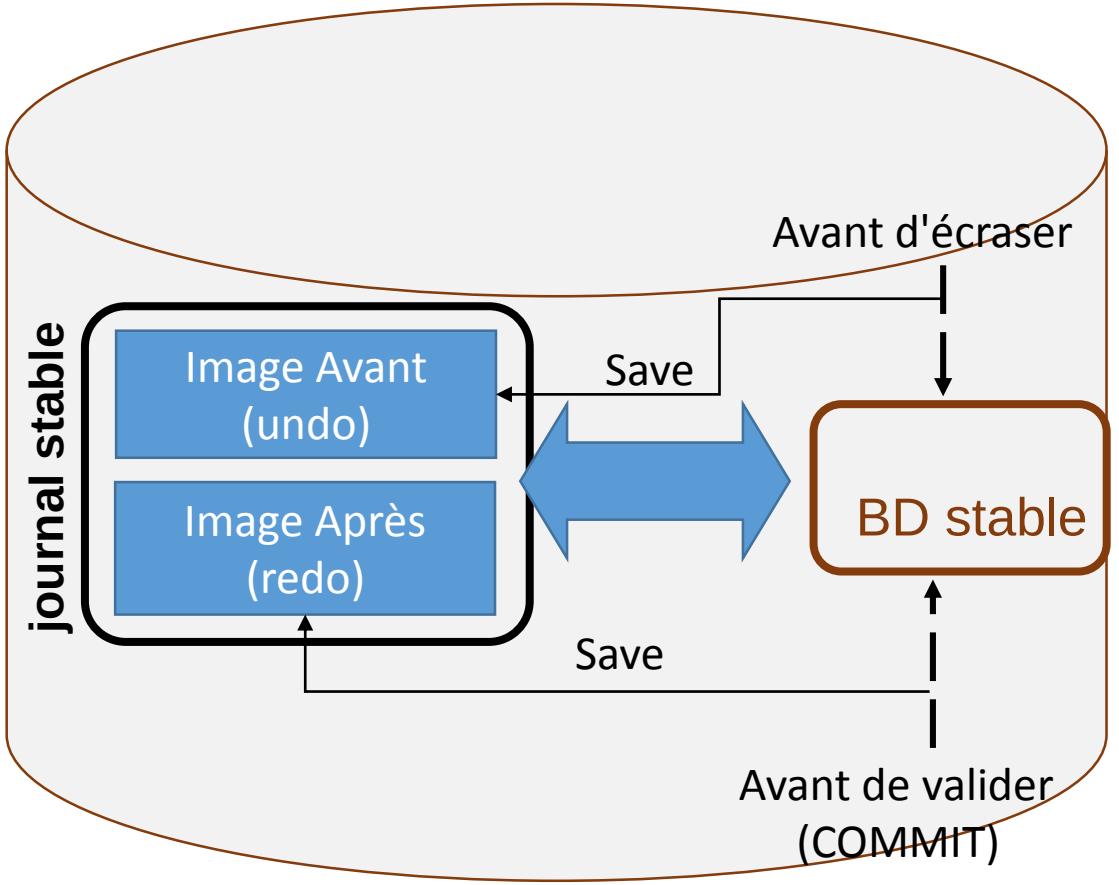
Règles du undo et du redo

Règle du Avant (Undo)

Avant d'écraser une ancienne valeur par une nouvelle, dans la BD stable, sauvegarder l'ancienne (l'image avant) dans un autre emplacement stable (dans le journal stable)

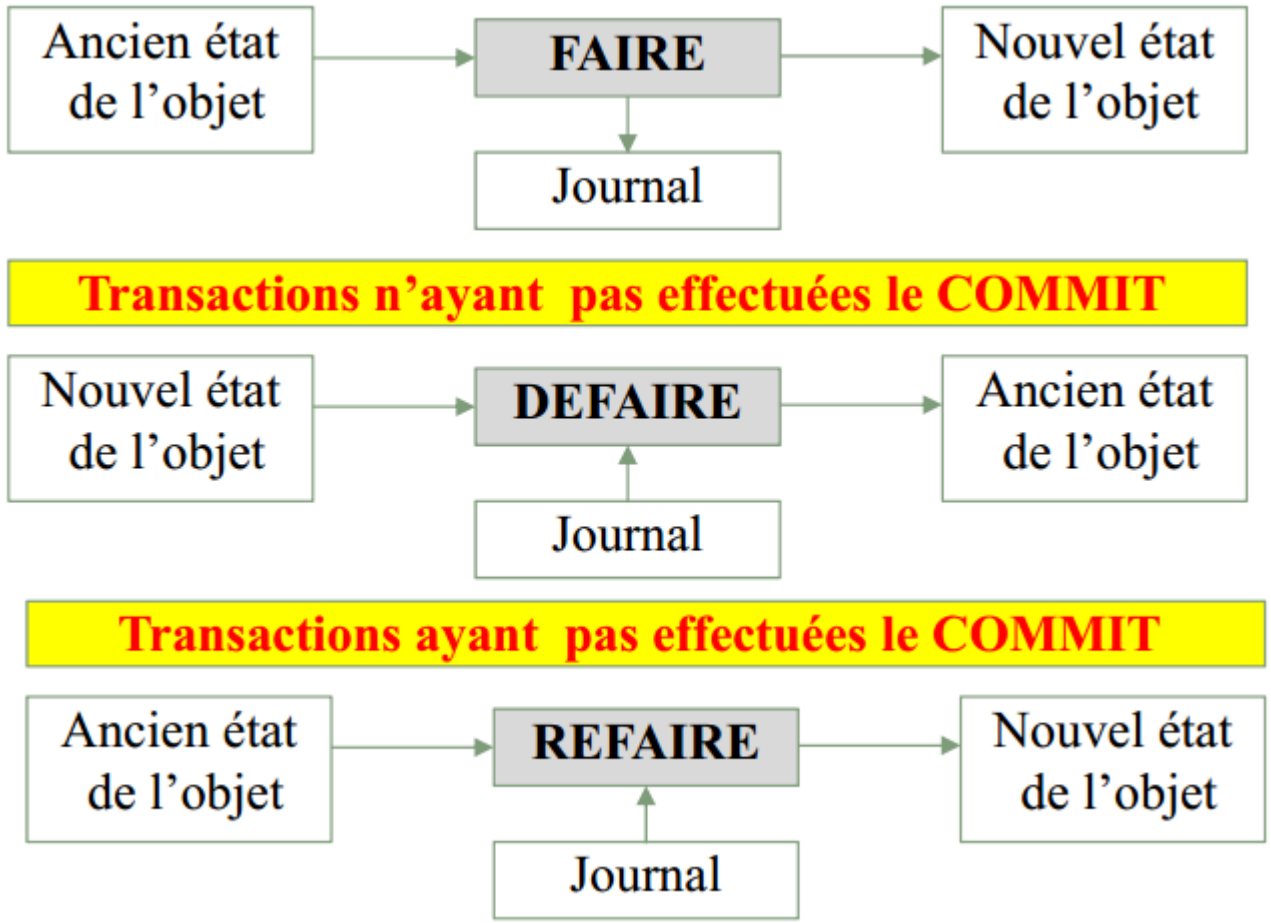
Règle du Après (redo)

Avant d'autoriser une transaction à validée, toutes les valeurs générées (les images- après de son journal) doivent d'abord être sauvegardées en mémoire stable (journal stable)



La journalisation

Paradigme faire-refaire-défaire



Utilisations du journal

Lors de la validation

sauvegarder toutes les images après en mémoire stable (journal)

- optimisation : « group commit + precommit »

Lors de l'annulation

remettre les images avant (Undo) en parcourant le journal en arrière

Lors d'une reprise après panne

Refaire l'historique en exécutant les opérations du journal du début jusqu'au moment de la panne (dernier enregistrement)

- très coûteux et beaucoup de travail inutile

Points de reprise

- Réduit la quantité de travail à refaire ou défaire lors d'une panne
- Un point de reprise enregistre une liste de transactions actives
- Pose d'un point de reprise:

– écrire un enreg. **begin_checkpoint** dans le journal

– écrire les buffers du journal et de la BD sur disque

– écrire un enreg. **end_checkpoint** dans le journal

Checkpoints (Points de Sauvegardes ou Points de reprise

C'est la sauvegarde (périodique) de certaines informations en **mémoire stable** durant le déroulement normal des transactions**, afin de réduire la quantité de travail** que doit faire la procédure **'Restart'** après une panne.

Checkpoint consistant

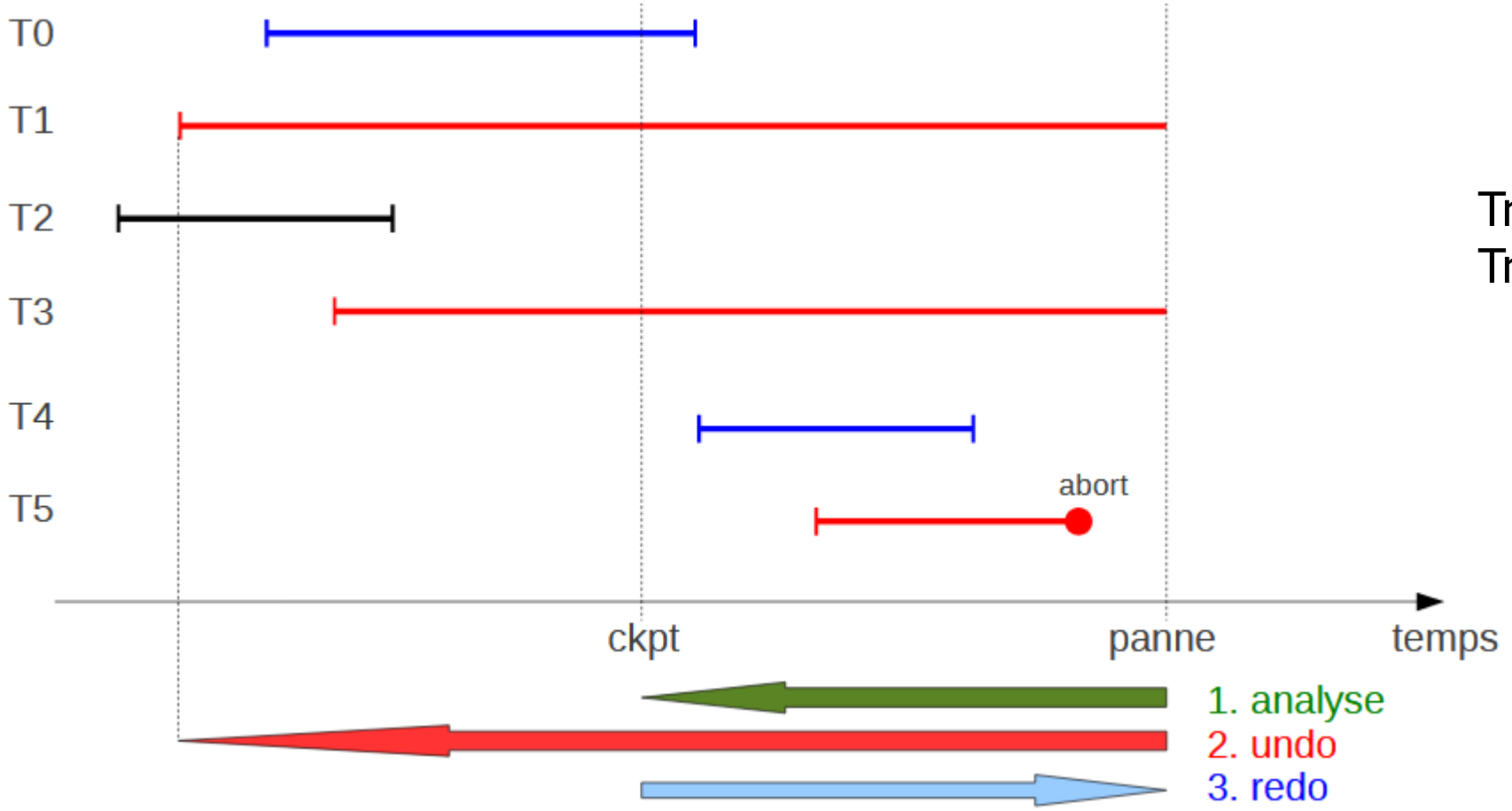
- Stopper l'acceptation de nouvelles transactions
- Attendre que les transactions actives se terminent
- Sauvegarder toutes les pages modifiées du cache
- Marquer le journal stable par un enreg <**ckpt**>

Lors de la reprise (Restart)

- Parcourir le journal en arrière jusqu'au <**ckpt**> le plus récent et défaire les opérations des transactions annulées
- Parcourir le journal en avant depuis la marque du <**ckpt**> et refaire les opérations des transactions validées

➲ Quel est inconvénient de cette solution ?
➲ Comment je peux optimiser ?

Exemple



Transaction refaites ??
Transaction defaites ??

reprise: T0 et T4 seront refaites. T1, T3 et T5 seront défaites

Question ?

Checkpoint consistant

➲ Quel est inconvénient de cette solution ?
➲ Comment je peux l'optimiser ?

Procédures de reprise

• **Reprise à chaud**

– perte de données en mémoire, mais sur disque

– à partir du dernier point de reprise , déterminer les transactions

validées : REDO

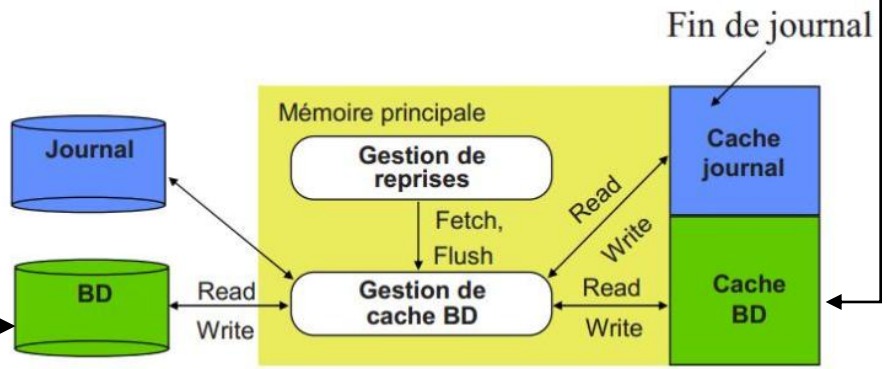
si Commit T, alors REDO := REDO + T ;

non validée : UNDO

– si Abort T, alors UNDO := UNDO + T ;

– si Begin T et T' ∉ REDO, alors UNDO := UNDO + T

•



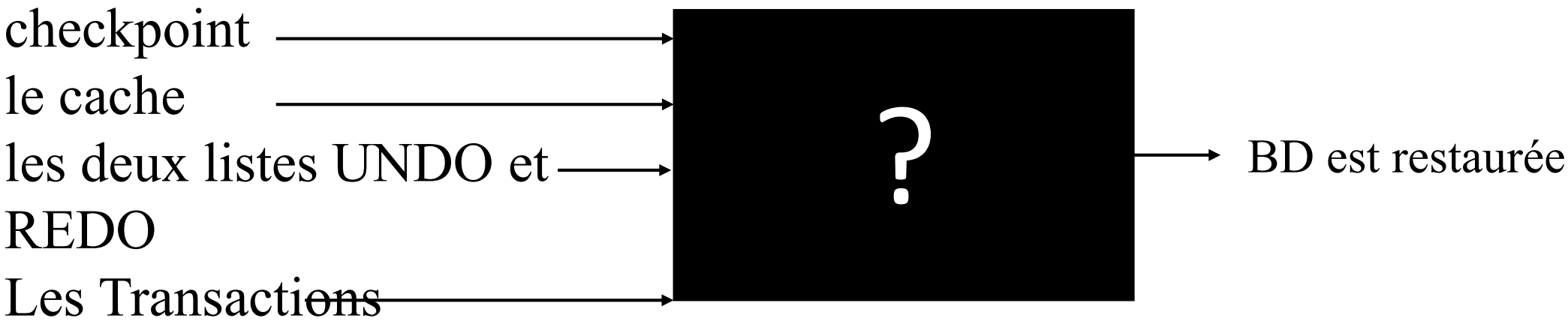
Reprise à froid

– perte de données sur disque

– à partir de la dernière sauvegarde et du dernier point de reprise, faire REDO des transactions validées

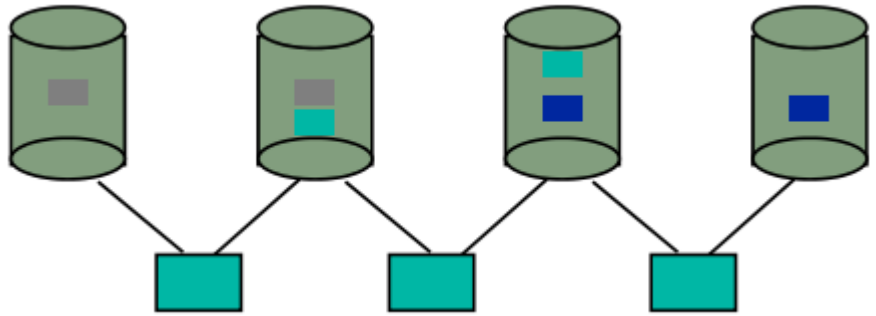
– UNDO inutile

Algo. général de reprise



Tolérance aux pannes

- Possibilité de fonctionnement malgré les pannes
  - en général avec une reprise à chaud
- Approche traditionnelle
  - **duplication** en miroir du matériel et des données
- Avantage suppl. : le partage de charge



Multi-ordinateurs

- On peut stocker les données redondant d'une BD sur plusieurs sites
  - même distants
- données d'une SDDS
  - Structure de Données Distribuée et Scalable
- Une meilleure protection contre une panne catastrophique
  - explosion, panne de courant, grève...
- Une meilleure sécurité
  - il peut être nécessaire de pénétrer plusieurs sites pour avoir une donnée