Reprise sur Panne
FIP - BDD

Nicolas Travers

Équipe Vertigo & ISI
Laboratoire CEDRIC
Conservatoire National des Arts & Métiers, Paris, France

Plan

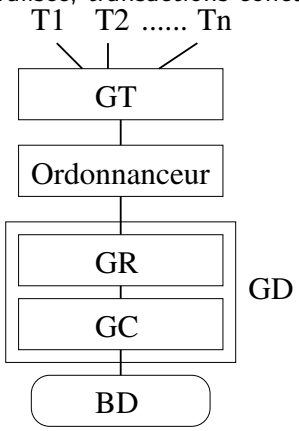
Problématique

- Une base de données est constamment modifiées
- Modifications en mémoire pour plus d'efficacité
- Le contenu de la base de données n'est pas toujours à jour

⇒ Définir une stratégie de Reprise en cas de panne

Rappel : Modèle abstrait de la BD

BD centralisée, transactions concurrentes :



Composantes

- **Gestionnaire de transactions** (GT) : reçoit les transactions et les prépare pour exécution
- **Ordonnanceur** (Scheduler) : contrôle l'ordre d'exécution des opérations (séquences sérialisables et recouvrables)
- **Gestionnaire de reprise** (GR) : Commit + Abort
- **Gestionnaire du Cache** (GC) : gestion de la mémoire volatile et de la mémoire stable

GR + GC = GD (**Gestionnaire de données**) : assure la tolérance aux pannes

Objectif

- dernière valeur validée de x : dernière valeur écrite en x par une transaction validée
- état validé de la BD : l'ensemble des dernières valeurs validées pour tous les enregistrements
- panne : mémoire volatile perdue

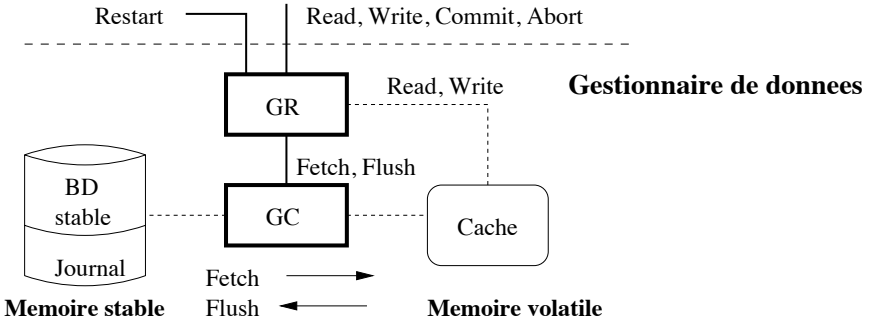
$\Longrightarrow$ *Restart* doit ramener la BD à l'état validé avant la panne

- problèmes
  - annuler l'effet des transactions non-validées
  - terminer les transactions validées
  - structures à garder en mémoire stable pour assurer la reprise

2. Architecture

Les composantes du Gestionnaire de données (GD)

- **Gestionnaire du Cache** (GC) : gère les deux mémoires
- **Gestionnaire de reprise** (GR) : opérations BD + Restart



Gestionnaire du Cache

- utilisation de la mémoire volatile : rapidité
- idéal : copie de toute la BD
- en réalité : caching, car taille mémoire volatile limitée

Cache

- zone de mémoire volatile divisée en *cellules* : 1 enreg./cellule
- en réalité, le Cache stocke des *pages disque*

Opérations

- Flush (c), c cellule

si c **inconsistante alors**
copier c **sur disque**
rendre c **consistante**
sinon rien ;

- Fetch (x), x enregistrement (pas dans le Cache)

sélectionner c **cellule vide**
si toutes les cellules occupées alors
vider une cellule c **avec Flush et l'utiliser comme cellule vide**
copier x **du disque en** c
rendre c **consistante**
mettre à jour le répertoire du cache avec (x, c)

Gestionnaire de reprise

Opérations

- GR_Read (T_i, x)
- GR_Write (T_i, x, v)
- GR_Commit (T_i)
- GR_Abort (T_i)
- Restart

Journalisation

Journal

- historique des écritures dans la mémoire stable
- **journal physique** : liste de [T_i, x, v]
  - préserve l'ordre des écritures : fichier séquentiel
  - souvent on stocke aussi *l'image avant* de l'écriture
- **journal logique** : opérations de plus haut niveau
  Ex. insertion x dans R et mise-à-jour index
  - moins d'entrées, mais plus difficile à interpréter
- autres informations : listes de transactions actives, validées, annulées

Exemple : journal physique

[T_1,x,2], [T_2,y,3], [T_1,z,1], [T_2,x,8], [T_3,y,5], [T_4,x,2], [T_3,z,6]
c_1 a_2 c_4

liste_active={T_3}
liste_commit={T_1, T_4}
liste_abort={T_2}

Ramasse-miettes

- recyclage de l'espace utilisé par le journal
- règle :
  entrée [T_i, x, v] recyclée $\Leftrightarrow$
  - T_i annulée ou
  - T_i validée, mais une autre T_j validée a écrit x après T_i

Stratégies pour la reprise

Types de GR

- GR peut forcer ou non GC d'écrire des cellules du Cache sur disque
- GR lance l'annulation
  - permet aux transactions non-validées d'écrire sur disque
  - *Restart* doit annuler ces écritures (annulation)
- GR lance la répétition
  - permet aux transactions de valider avant d'écrire sur disque
  - *Restart* doit refaire ces écritures (répétition)
- 4 catégories de GR (combinaisons annulation - répétition)

Règles défaire/refaire

- règles de journalisation, nécessaires pour que le GR puisse faire correctement annulation/répétition
- Règle "défaire" (pour annulation) : si x sur disque contient une valeur validée, celle-ci doit être journalisée avant d'être modifiée par une valeur non-validée
- Règle "refaire" (pour répétition) : les écritures d'une transaction doivent être journalisées avant son Commit
- *Remarque* : ces règles sont naturellement respectées si l'on écrit dans le journal avant toute écriture dans la BD

Idempotence de Restart

- une panne peut interrompre toute opération, même *Restart*
- idempotence : *Restart* interrompu et relancé donne le même résultat que le *Restart* complet
- optimisation : journalisation des opérations de *Restart* pour ne pas tout recommencer

Checkpointing

- ajouter des informations sur disque en fonctionnement normal afin de réduire le travail de *Restart*
- *point de contrôle* ("checkpoint") : point (marqué dans le journal) où l'on réalise les actions supplémentaires

Quelques techniques

- marquer dans le journal les écritures déjà **réalisées**/annulées dans la BD stable
  - pas besoin de refaire/annuler ces écritures à la reprise
- marquer toutes les écritures **validées**/annulées dans la BD stable
  - pas besoin de refaire/annuler à la reprise les transactions validées/annulées

Algorithme annulation/répétition

Principes

- GR qui demande annulation et répétition : le plus complexe
- écrit les valeurs dans le Cache et ne demande pas de Flush
- avantages : flexibilité, minimise I/O

Opérations

- GR-Write (T_i, x, v)

liste_active = **liste_active** $\cup \{T_i\}$
si **x** n'est pas dans le cache alors allouer cellule pour **x**
journal = **journal** + [T_i, x, v]
cellule(**x**) = **v**
confirmer Write à l'ordonnanceur

- GR-Read (T_i, x)

si **x** n'est pas dans le cache alors **Fetch(x)**
retourner la valeur de *cellule*(**x**) à l'ordonnanceur

- GR-Commit (T_i)

liste_commit = **liste_commit** $\cup \{T_i\}$
confirmer le Commit à l'ordonnanceur
liste_active = **liste_active** - T_i

- GR-Abort (T_i)

pour chaque **x** écrit par T_i
    si **x** n'est pas dans le cache alors allouer cellule pour x
    cellule(**x**) = **image_avant(x, T_i)**
liste_abort = **liste_abort** $\cup \{T_i\}$
confirmer Abort à l'ordonnanceur
liste_active = **liste_active** - $\{T_i\}$

- Restart

marquer toutes les cellules comme vides
refait = {}, **annulé** = {}
pour chaque [T_i,x,v] ∈ **journal** (à partir de la fin) où **x** ∉ **annulé**∪**refait**
    si **x** n'est pas dans le cache alors allouer cellule pour **x**
    si T_i ∈**liste_commit** alors
        cellule(**x**)=v
        refait=**refait** ∪ {x}
    sinon
        cellule(**x**)= **image_avant(x, T_i)**
        annulé=**annulé** ∪ {x}
    si **refait** ∪ **annulé** = BD alors stop boucle
pour chaque T_i ∈ **list_commit**
    list_active=**list_active**-{T_i}
confirmer Restart à l'ordonnanceur

Algorithme annulation/sans-répétition

- GR ne demande jamais répétition
- enregistre écritures avant le Commit
- GR-Write, GR-Read, GR-Abort pareil
- GR-Commit pareil, mais d'abord :
  - pour chaque x écrit par T_i, si $x \in$ Cache alors Flush(x)
- Restart pareil, sauf que "refait" n'existe pas

Algorithme sans-annulation/répétition

- GR ne demande jamais annulation
- écritures des T_i non-validées retardées après Commit
- GR-Write : ajoute juste [T_i, x, v] au journal
- GR-Read : si T_i a déjà écrit x, lecture dans le journal, sinon dans la BD
  - si T écrit x, les autres transactions ne peuvent lire x qu'après la fin de T (exécution stricte)
- GR-Commit : chaque x écrit par T_i est calculé à partir du journal et écrit dans le cache
- GR-Abort : juste ajoute T_i à liste_abort
- Restart : pareil, sauf que "annulé" n'existe pas

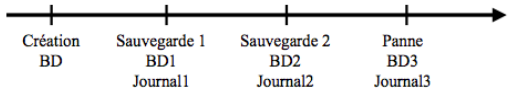
Algorithme sans-annulation/sans-répétition

les écritures de T_i réalisées sur disque en une seule opération atomique, au Commit

Types de pannes

- Normale
- Après une panne du système : reprise à chaud
  - on a perdu la mémoire centrale mais pas la mémoire secondaire
- Après une panne de mémoire secondaire : reprise à froid
  - perte de mémoire secondaire
  - principe :
    - Reprendre les sauvegardes sur bande
    - Utiliser le journal s'il est disponible
- panne catastrophique

Exemple de situation



- **Reprise à chaud** : on réapplique le journal 3 sur BD3 pour défaire les transactions non validées
- **Reprise à froid** : si BD2 est endommagée, on reprend BD1 et on réapplique journal2 et Journal3

Recovery Manager

- Outils standard spécialisé dans la sauvegarde et la restauration de données
- Cas de pertes de la base de données : définition des points de sauvegarde
- Sauvegardes :
- Contrôles :

Recovery Manager

- Outils standard spécialisé dans la sauvegarde et la restauration de données
- Cas de pertes de la base de données : définition des points de sauvegarde
- Sauvegardes :
  - À froid (SGBD inactif)
    À chaud (SGBD en cours de traitements)
  - De fichiers
    De bases entières
  - Incrémentales
    Différentielles
- Contrôles :

Recovery Manager

- Outils standard spécialisé dans la sauvegarde et la restauration de données
- Cas de pertes de la base de données : définition des points de sauvegarde
- Sauvegardes :
  - À froid (SGBD inactif)
    À chaud (SGBD en cours de traitements)
  - De fichiers
    De bases entières
  - Incrémentales
    Différentielles
- Contrôles :
  - Sauvegardes
  - Restauration (blocs corrompus)

Possibilités

- Simulation de restauration
- Évite les blocs vides car opération au niveau blocs de données
- Mode console
- Interfaçage avec différent logiciels de sauvegarde

Sauvegardes Totales et Partielles

- Sauvegarde totale :
  - Copie toutes les données
  - *backup database*
- Sauvegarde partielle :
  - Sous-ensemble sélectionné
  - *backup datafile* / *backup tablespace*

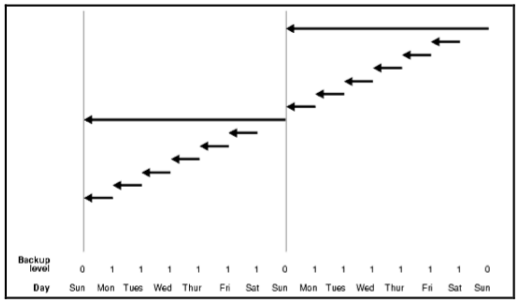
Sauvegarde incrémentales

- Permet d'échelonner des sauvegardes (alléger le SGBD)
- Besoin d'une première sauvegarde (dites de niveau 0)
  `RMAN> backup incremental level 0 database`
- Toutes les autres sont dites de niveau 1
  `RMAN> backup incremental level 1 database`
- Stockent les modifications depuis la dernière sauvegarde

Introduction Reprise Stratégies Procédures RMAN Sauvegardes Catalogue Restauration

Sauvegarde incrémentales

- Permet d'échelonner des sauvegardes (alléger le SGBD)
- Besoin d'une première sauvegarde (dites de niveau 0)
  `RMAN> backup incremental level 0 database`
- Toutes les autres sont dites de niveau 1
  `RMAN> backup incremental level 1 database`
- Stockent les modifications depuis la dernière sauvegarde

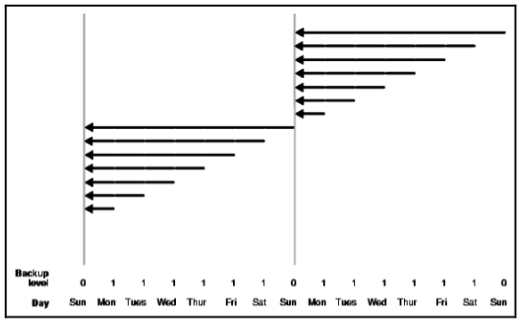


43 / 51
le cnam
N. Travers Reprise sur Panne

Introduction Reprise Stratégies Procédures RMAN Sauvegardes Catalogue Restauration

Sauvegarde incrémentales cumulatives

- Sauvegarde des modificationss depuis le dernier niveau 0
  `RMAN> backup incremental level 1 cumulative database`
- Perte en temps de sauvegarde, gain en temps de reprise



44 / 51
le cnam
N. Travers Reprise sur Panne

Recovery Catalog

- Le catalogue de récupération est optionnel mais fortement recommandé
- Contient :
  - Ensemble des sauvegardes
  - Copies des fichiers de données
  - Structure physique de la base de données
  - Archives de logs
  - Scripts de travail
- Doit être créé sur un serveur distinct

Gestion du catalogue

- Création du schéma du propriétaire RMAN :

```
SQL> CREATE TABLESPACE rman_data datafile 'RMAN/rd.dbf' size 50M;
CREATE USER rman identified by toto DEFAULT TABLESPACE rman_data;
GRANT CONNECT, RESOURCE, RECOVERY_CATALOG_OWNER TO rman;
```

- Création du Catalogue RMAN :

```
RMAN> create catalog tablespace rman_data;
```


Enregistrement de la base de données

- Connexion à RMAN :

```
RMAN> target u1/p1[@db] catalog u2/p2[@catalog]
```

- Enregistrement :

```
RMAN> register database ;
```

Restauration

- Avant de restaurer, il faut mettre le tablespace cible *offline* :
  `RMAN> SQL ''ALTER TABLESPACE TOTO OFFLINE'';`
- Puis lancer la dernière sauvegarde (en lien avec le catalogue) :
  `RMAN> RESTORE TABLESPACE TOTO ;`
- Puis relancer le tablespace :
  `RMAN> SQL ''ALTER TABLESPACE TOTO ONLINE'';`
- Restauration complète :
  `RMAN> RECOVER TABLESPACE TOTO ;`
- Restauration incomplète :
  ```
  set UNTIL TIME ‘‘to_date(’23/10/2012 12:00’,
  ’DD/MM/YYYY HH24:MI’)’’;
  ```


Informations

- Liste des différentes sauvegardes :
  ```
  LIST BACKUP OF DATABASE;
  LIST BACKUP OF CONTROLFILE;
  LIST BACKUP OF SPFILE;
  LIST BACKUP OF TABLESPACE USERS;
  LIST BACKUP OF DATAFILE 1;
  LIST CTROLFILECOPY 1
  ```