Technique de récupération des bases de données

Hala Skaf-Molli

L3

skaf@loria.fr

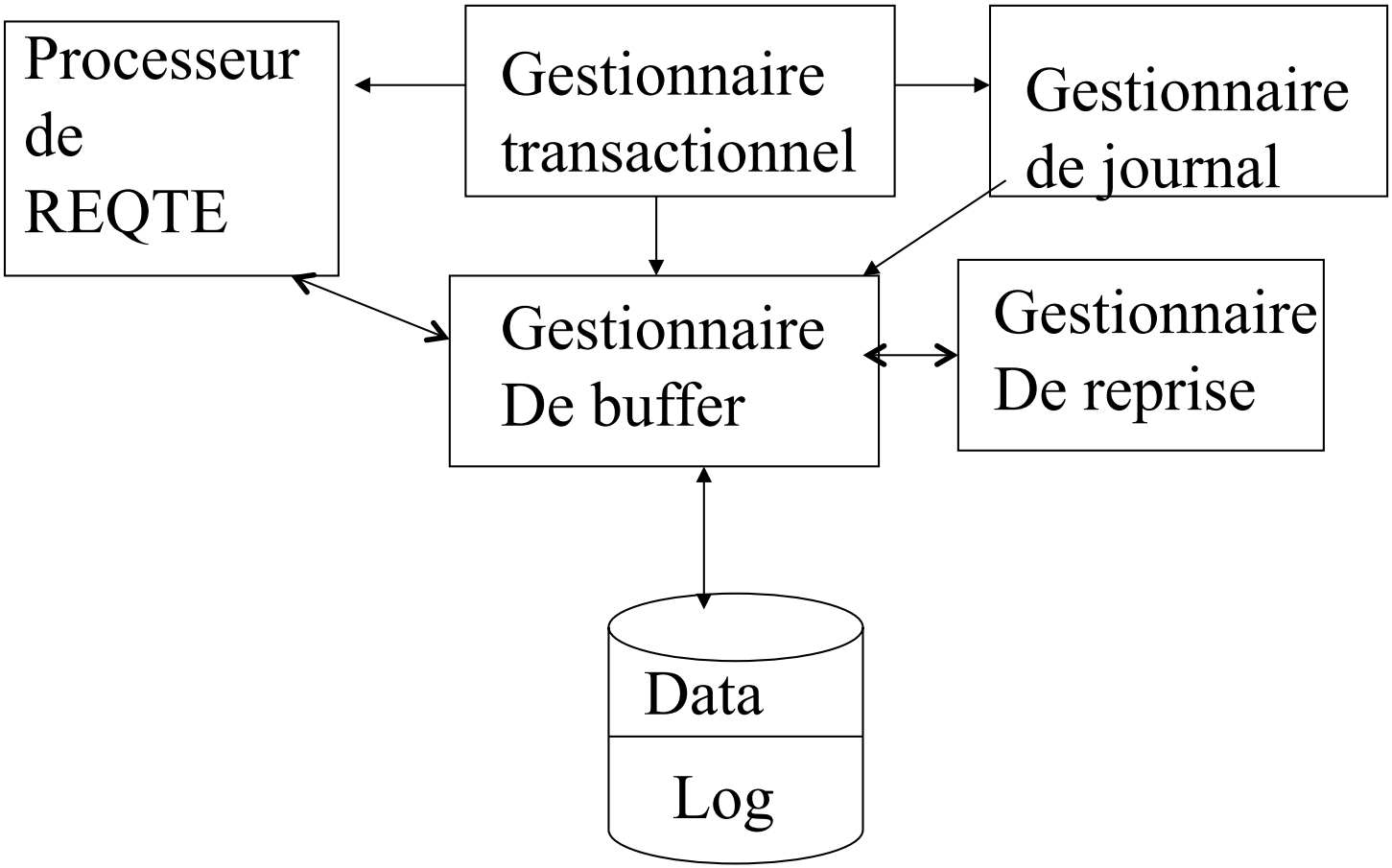
Récupération des bases de données

- Une base de données utilise:
  - Mémoire secondaire, non volatile (disque)
  - Mémoire centrale, volatile (*database cache*)
- Pendant son exécution une transaction lit et modifie les données dans la cache et à sa terminaison, l'idéal est que la transaction écrit ses modifications sur le disque
- Problème: que se passe t-il en cas du panne?
  - assurer l'atomicité de transactions

Concepts de la récupération

- La récupération (reprise ou recouvrement) des transactions consiste à restaurer la base de données à l'état cohérent *le plus récent* avant la survenue de la panne.
- Deux types de panne:
  - Panne catastrophique : panne disque, système..
  - Panne non catastrophique: erreur de la transaction, exception dans la transaction, pb d'accès concurrent, coupure d'électricité

Gestionnaire transactionnel et gestionnaire de log



Stratégies de récupération (1)

- Panne catastrophique: si une grande partie de la BD a été endommagée (panne disque):
  - Restaurer une copie (sauvegarde) antérieure de la base
  - Reconstruire un nouvel état en réappliquant (refaisant) les opérations validées et sauvegardées dans le journal
- Une copie complète de **l'archive** de la base de données + le **journal** …

Stratégie de récupération (2)

- Panne non catastrophique :Si la BD n'est pas physiquement endommagée mais elle est devenue incohérente :
  - Annuler les opérations qui sont à l'origine de l'incohérence (défaire)
  - Refaire certaines opérations
- Pas besoin d'une copie complète de l'archive de la base de données, le **journal** est suffisant …

Panne non catastrophiques

- Trois techniques de reprise:
  - Mises à jour immédiates :UNDO LOG
  - Mises à jour différées : REDO LOG
  - UNDO/REDO LOG

Les techniques de mise à jour immédiate
UNDO LOG

- La base de données est actualisée immédiatement après une mise à jour d'une transaction **avant** sa validation
  - Bien sûr, les opérations sont enregistrées d'abord de manière **persistante** dans le journal (avec le protocole de journalisation avant écriture, WAL Write-Ahead Logging)

Les techniques de mise à jour immédiate

- Si une transaction échoue avant d'atteindre son point de validation :
  - Il faut annuler les effets de la transaction (ROLLBACK)
  - défaire (avec UNDO) les opérations
- Algorithme: UNDO

UNDO LOG

- Un tuple <T, x, v> dans le log signifie que la transaction T a modifié l'élément x de la base de données et l'ancienne valeur de x est v.
- Attention, pas d'indication sur la nouvelle valeur de x.

Règles d'UNDO LOG

- U1: Si une transaction T modifie un élément x de la base de données, alors l'enregistrement <T,x,v> du log doit être écrit sur disque avant que la nouvelle valeur de x soit écrit sur le disque.
- U2: Si une transaction T est validée, alors l'enregistrement <Commit T> du log doit être écrit sur le disque **après** que tous les éléments modifiés par la transaction soient copiés sur le disque (le plutôt possible).

UNDO LOG

- L’ordre d’écriture sur le disque :
  1. L’enregistrement du log concernant l’élément modifié : <T, x, v> , v est l’ancienne valeur de x.
  2. Les éléments modifiés de la base de données
  3. L’enregistrement de la validation de la transaction <COMMIT T>

Exemple: UNDO LOG

T1 : A := A* 2 ; B := B*2 avec A=8 ; B=8 CI: A=B

étape	action T1	t	M-A	M-B	log	D-A	D-B
1	begin T1				<begin T1>	8	8
2	t := lire(A)	8	8			8	8
3	t:=t*2	16	8			8	8
4	écrire(A,t)	16	16		<T1,A,8>	8	8
5	t := lire(B)	8	16	8		8	8
6	t := t*2	16	16	8		8	8
7	écrire(B,t)	16	16	16	<T1,B,8>		
8	FLUSH LOG						
9	OUTPUT(A)	16	16	16		16	8
10	OUTPUT (B)	16	16	16		16	16
11	COMMIT T1			16	<Commit T1>		
12	FLUSH LOG						

Reprise avec mise à jour immédiate

1. Identifier les transactions validées (committées) et les transaction non validées.
2. Lire le journal de la fin vers le début (de la dernière enregistrement écrit vers le premier écrit), pour chaque tuple <T, x, v> :
   a. Si T est valide i.e. <Commit T> apparaît sur le disque alors
      rien (Règle U2)
   b. Si T n'est pas valide (incomplète ou annulée) alors
      mettre la valeur de x dans la base de données à v
3. Pour chaque transaction T incomplète, ajouter <rollback T> dans le journal et copier le journal sur le disque.

Retour à l'exemple d'UNDO LOG

- Panne après l'étape 12 :
  - <commit T1> est copié sur le disque . T est identifiée comme valide alors, rien à faire pour T1.
- Panne entre l'étape 11 et 12:
- Panne entre l'étape 10 et 11:
- Panne avant 8 :

Technique de reprise avec mises à jour immédiate

- Avantages:
  - Elle dispense de la nécessité de disposer d'un grand espace tampon pour stocker toutes les pages mises à jour en mémoire
- Désavantage:
  - Il peut augmenter le nombre de I/O sur le disque (une pages est fréquemment mise à jour par de nombreuses transactions)

Les techniques de mises à jour différées

- Actualisent physiquement la BD sur le disque après que la transaction a atteint son point de validation
  - Avant la validation de la transaction toutes les modifications sont enregistrées dans l'espace local (tampon) des transactions
  - Lors de la validation les mises à jour sont enregistrées de manière **persistante** dans le journal
  - Puis écrites dans la base de données

Les techniques de mises à jour différées

- Si une transaction échoue avant d'atteindre son point de validation:
  - Pas besoin de défaire (avec UNDO) les opérations
- Besoin de refaire (REDO) une transaction validée dans le journal et ses effets n'ont pas été pris en compte dans la base de données
- Algorithme: (REDO LOG)

REDO LOG

- L'enregistrement <T, x, v> dans le log signifie que la transaction T a modifié l'élément x de la base de données et la nouvelle valeur de x est v.
- Attention, pas d'indication sur l'ancienne valeur de x.

La règle de REDO LOG

- R1: avant de modifier n'importe quel élément x sur le disque, il est nécessaire que tous les enregistrement concernent les modifications de x:
  - l'enregistrement de mise à jour <T,x,v>
  - et l'enregistrement <COMMIT T>

  doivent apparaître sur le disque.

REDO LOG

- L'ordre d'écriture sur le disque :
  1. L'enregistrement du log concernant l'élément changé : <T, x,v>, v est la nouvelle valeur de x.
  2. L'enregistrement concernant la validation de la transaction <commit T>
  3. la nouvelle valeur de x.

Exemple: REDO LOG

T1 : A := A* 2 ; B := B*2 avec A=8 ; B=8 CI: A=B

étape	action T1	t	M-A	M-B	log	D-A	D-B
1	begin T1				<begin T1>	8	8
2	t := lire(A)	8	8			8	8
3	t:=t*2	16	8			8	8
4	écrire(A,t)	16	16		<T1,A,16>	8	8
5	t := lire(B)	8	16	8		8	8
6	t := t*2	16	16	8		8	8
7	écrire(B,t)	16	16	16	<T1,B,16>	8	8
8	Commit T1				<Commit T1>		
9	FLUSH LOG						
10	OUTPUT(A)	16	16	16		16	8
11	OUTPUT (B)	16	16	16		16	16

Écrire(A,t) : copier la valeur de la variable t dans l'emplacement de l'élément A dans la mémoire centrale

OUTPUT(A): copier le bloc mémoire contenant A vers le disque.

Fulsh LOG: copier les entrées du log vers le disque si elles ne sont pas déjà copiées.

Exemple: REDO LOG

étape	action T1	t	M-A	M-B	log	D-A	D-B
1	begin T1				<begin T1>	8	8
2	t := lire(A)	8	8			8	8
3	t:=t*2	16	8			8	8
4	écrire(A,t)	16	16		<T1,A,16>	8	8
5	t := lire(B)	8	16	8		8	8
6	t := t*2	16	16	8		8	8
7	écrire(B,t)	16	16	16	<T1,B,16>	8	8
8	Commit T1				<Commit T1>		
9	FLUSH LOG						
10	OUTPUT(A)	16	16	16		16	8
11	OUTPUT (B)	16	16	16		16	16

Reprise avec mise à jour différées

1. Identifier les transactions validées (committées) et les transaction non validées.
2. Lire le journal du début, pour chaque enregistrement <T, x, v> :
   a. Si T n' est pas valide i.e. <Commit T> n'apparaît pas sur le disque alors
      rien
   b. Si T est valide alors
      écrire la valeur v dans la base
3. Pour chaque transaction T incomplète, ajouter <rollback T> dans le journal et copier le journal sur le disque.

Retour à l'exemple (1)

- Panne après l'étape 9 :
  - <commit T1> est copié sur le disque . T est identifiée comme valide alors, on lit le journal du début , on copie la nouvelle valeur de A et de B sur le disque.
- Panne entre l'étape 8 et 9,
  - <commit T1> n'a pas encore arrivé au disque ?? la transaction est marquée comme incomplète : rien à faire pour A et B, ajouter <rollback T1> dans le log et copier le log sur le disque.
- Panne avant 8 : comme 2.

Techniques de reprise avec mises à jour différées

- Avantage:
  - Récupération facile: Si une transaction échoue avant d'atteindre son point de validation: rien à faire
- Désavantage:
  - Il peut être mis en œuvre pour les transactions brèves et ne modifient que peu d'éléments, pourquoi ?

Comparaison de UNDO et REDO journal

Undo Log	Redo Log
Ancienne valeur	Nouvelle valeur
Ecrire les éléments modifiés sur le disque **avant** d'écrire le <commit T> sur le disque	Ecrire les éléments modifiés sur le disque **après** d'écrire le <commit T> sur le disque
Parcours de la fin vers le début	Parcours du début vers la fin
Transaction valide: rien	Transaction valide : REDO
Transaction incomplète : UNDO	Transaction incomplète : rien

UNDO/REDO LOG

- Un enregistrement <T, X, v, w> dans le log signifie que la transaction T a changé l'élément x de la base de données, son ancienne valeur est v et sa nouvelle valeur est w.

Règle d'UNDO/REDO Log

- UR1 : avant de modifier un élément x de la base de données, il faut que l'enregistrement <T, x, v, w> du journal soit copié sur le disque.

- La règle UR1 respect les contraintes **communes** d'Undo et Redo log.
- Pas de contraintes sur le <COMMIT T> , il peut précéder ou suivre les modifications réalisées sur les éléments de la base de données ..

Scénario

étape	action T1	t	M-A	M-B	log	D-A	D-B
1	begin T1		8	8	<begin T1>	8	8
2	t := lire(A)	8					
3	t:=t*2	16					
4	écrire(A,t)	16	16		<T1,A,8,16>		
5	t := lire(B)	8					
6	t := t*2	16					
7	écrire(B,t)	16		16	<T1,B, 8,16>		
8	FLUSH LOG						
9	OUTPUT(A)					16	
10	commit T1				<commit t1>		
11	OUTPUT (B)			16			16

Attention : problème avec la validation retardée

- Une transaction apparaît comme valide (commit) à l'utilisateur (je réserve un ticket d'avion sur le web et je me déconnecte) avant que le commit de ma transaction soit copié sur le disque un crash a lieu = > alors ma transaction est annulée (Undo) au lieu qu'elle soit rejouer (Redo). Pour éviter ce problème, on peut ajouter une nouvelle règle au Undo/Redo log
- UR2 : l'enregistrement <commit T> doit être copié sur le disque dès qu'il apparaît dans le log.
- On doit ajouter FLUSH LOG après l'étape 10.

Reprise avec UNDO/REDO LOG

1. **Redo** toutes les transactions validées dans l'ordre la première d'abord
2. **Undo** toutes les transactions incomplètes dans l'ordre la dernière d'abord.

Retour à l'exemple

- Une panne arrive après <commit T> est copié sur le disque :
  - T est identifiée comme valide, on écrit la valeur 16 pour A et B (sur le disque) .. Peut-être A est déjà 16 ..
- Une panne arrive avant que <commit T> soit copié sur le disque :
  - T est identifiée comme transaction incomplète. Les anciennes valeurs de A et B (8) sont copiées sur le disque.
- Une panne arrive entre l'étape 9 et 10 … la valeur de B n'a pas besoin d'être modifiée.. Dans le cas général, on ne sait pas, la règle générale, on fait d'Undo

Exercice (1)

- Soit le journal:

  <begin T> ;<T,A,10,11> ;<T,B,20,21> ;<commit T> ;
- Donner l'ordre (s) d'écriture des événements suivants sur le disque:
  - LA : copier <T,A,10,11> sur le disque
  - LB : copier <T,B,20,21> sur disque
  - D-A: écrire la nouvelle valeur de A sur le disque
  - D-B: écrire la nouvelle valeur de B sur le disque
  - CT: copier <Commit T> sur le disque

- En supposant que LA; LB; CT

Exercice (2)

La séquence suivante de UNDO/REDO est écrite par deux transactions T et U.

<begin T> ;

<T,A,10,11> ;

<begin U> ;

<U,B,20,21> ;

<T,C,30,31>;

<U,D,40,41>;

<commit U>;

<T,E,50,51>;

<Commit T>

Décrire les actions de gestionnaire de reprise si une panne est arrivée et le dernier enregistrement copié sur le disque est:

a) <begin U>

b) <Commit U>