

Systèmes de Gestion des Bases de Données

Accès concurrents et Sécurité de fonctionnement

Nacer Boudjlida

Introduction à la notion de transaction

- Partage des ressources dans les systèmes d'exploitation
- Base de données accédée par plusieurs utilisateurs (ou programmes)
- Sous-système de gestion des accès concurrents
- Techniques : Verrouillage, exclusion mutuelle
- Ressources partagées : Données et méta-données
- Transaction \simeq Programme (accès, modification)
- Accès concurrents \rightarrow Problèmes

Transactions : problème 1 : (Non) Atomicité

- Virement de compte à compte : $\{C1 = C1 - m; C2 = C2 + m\}$
 - 1)
 - 2)
 - 3)
 - 4)
 - 5)
 - 6)
- Si arrêt du programme avant *Ecrire(y)* : Incohérence
- Atomicité : "Tout ou rien"
- "Tout" \implies Durabilité (Permanence des modifications)
- "Rien" \implies Annulation des modifications

Transactions : problème 2 : (Non) Isolation

- Transaction T1 : Débiter X de N
- Transaction T2 : Créditer X de M

T1	T2
$x \leftarrow Lire(X)$	$y \leftarrow Lire(X)$
$x \leftarrow x - N$	$y \leftarrow y + M$
$X \leftarrow Ecrire(x)$	$X \leftarrow Ecrire(y)$

Problème 2 : (Non) Isolation

- $X = 100; N = 10; M = 50$

Exécution avec entrelacement	X_{base}	x_{T1}	y_{T2}
T1 :	100		
T1 :			
T2 :			
T2 :			
T1 :			
T2 :			

- Perte de mise à jour : $X = X - N$
- \implies Isolation : pas d'interférence pendant l'exécution

Problème 3 : (Non) Cohérence

- Contrainte : $A = B$

T1 : $\{A = A + 1; B = B + 1\}$	T2 : $\{A = A \times 2; B = B \times 2\}$
t11 : $A_{T1} \leftarrow Lire(A)$	t21 : $A_{T2} \leftarrow Lire(A)$
t12 : $A_{T1} \leftarrow A_{T1} + 1$	t22 : $A_{T2} \leftarrow A_{T2} \times 2$
t13 : $A \leftarrow Ecrire(A_{T1})$	t23 : $A \leftarrow Ecrire(A_{T2})$
t14 : $B_{T1} \leftarrow Lire(B)$	t24 : $B_{T2} \leftarrow Lire(B)$
t15 : $B_{T1} \leftarrow B_{T1} + 1$	t25 : $B_{T2} \leftarrow B_{T2} \times 2$
t16 : $B \leftarrow Ecrire(B_{T1})$	t26 : $B \leftarrow Ecrire(B_{T2})$

- $\langle t11; t12; t13; t21; t22; t23; t14; t15; t16; t24; t25; t26 \rangle$: Correct
- $\langle t21; t22; t11; t12; t23; t13; t14; t15; t16; t24; t25; t26 \rangle$: Incorrect

Problème 4 : (Non) Permanence des modifications

a) (Non) Répétabilité des lectures

Transaction T1	Transaction T2
$a \leftarrow Lire(A)$	$b \leftarrow Lire(A)$
	$Imprimer(b)$
$a \leftarrow a + 100$	
$A \leftarrow Ecrire(a)$	$b \leftarrow Lire(A)$
	$Imprimer(b)$

- Impressions de T2 : valeurs différentes

Problème 4 : (Non) Permanence des modifications

b) Tuples "fantômes"

Transaction T1	Transaction T2
	1. $V \leftarrow Lire(A = \{a < 4000\})$
	2. $Imprimer(V)$
3. Supprimer tout a tel que $a < 4000$	
	4. $V \leftarrow Lire(A = \{a < 4000\})$
	5. $Imprimer(V)$

- Deuxième impression de V vide

Transactions et niveaux d'isolation : SQL 92

- Niveau 0 : lectures "sales" permises
 - D en cours de modification par T
 - Transactions autres que T, avant validation par T :
 - * bloquées en modification
 - * autorisées à lire D
- Niveau 1 : lectures "sales" interdites
- Niveau 2 : non répétabilité des lectures avant validation
- Niveau 3 (Par défaut) : empêcher les tuples fantômes

Transactions et reprise

- Nécessité de contrôler les accès concurrents
- Causes d'échec des transactions :
 - Erreur logiciel ou matériel
 - Avortement par le sous-système de gestion de la concurrence
 - "Catastrophe naturelle", etc.
- Nécessité de rétablir la cohérence : Reprise
 - Revenir à l'état cohérent le plus proche de l'instant de l'incident
 - Mécanisme : Journal ou Log ("Histoire des transactions")

Schéma d'exécution des transactions

- Transactions de mise à jour : séquences
 1. Lecture (Base → Mémoire centrale)
 2. Traitement (en mémoire centrale)
 3. Écriture (Mémoire centrale → Base)
- Politique "du tout ou rien" (Atomicité) :
 - $Action_1$
 - ...
 - $Action_n$

Si toutes les actions se sont bien passées
Alors Valider la transaction (COMMIT)
Sinon Annuler ses effets (ROLLBACK)
Finsi

Schéma d'exécution des transactions

- Validation : rendre effectives les modifications
- Annulation : défaire les actions de la transaction
- Diagramme d'états d'une transaction

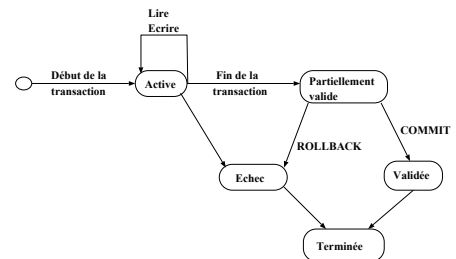


Schéma d'exécution des transactions

- *Validité partielle* :
 - Des techniques testent l'absence d'interférences
 - Des protocoles de reprise s'assurent qu'ils peuvent effectivement enregistrer les modifications
- Parfois : Annulation/Ré-exécution (UNDO/REDO) d'une opération

Journalisation, points de validation et de contrôle

- *Journal* :
 - Trace des opérations effectuées sur la base
 - Support non volatile + Archivage périodique
- *Types d'informations du journal*, pour chaque transaction :
 - <début_transaction, identification de T> ;
 - <écriture, identification de T, donnée concernée, ancienne valeur, nouvelle valeur> ;
 - <lecture, identification de T, donnée concernée>^a ;
 - <COMMIT, identification de T>.

^aCertains systèmes ne conservent pas de trace des opérations de lecture.**Journalisation, points de validation et de contrôle**

- *Point de validation* :
 - Instant de journalisation de <COMMIT, identification de T>
 - Les actions de T ont été réussies
 - + Écritures dans le journal réussies
 - Au-delà de ce point : T est valide et ses MaJ sont permanentes
 - En cas d'incident : le système sait quoi
 - * annuler ("histoire en arrière")
 - * relancer, éventuellement ("histoire en avant")

Journalisation, points de validation et de contrôle

- *Points de contrôle*
 - Inscrits périodiquement dans le *Log*
 - *Période* : intervalle de temps, nombre donné de mise à jour ou combinaison des deux
 - *Prise de point de contrôle* :
 - 1.
 - 2.
 - 3.
 - 4.
- ⇒ les actions des transactions dont le point de validation dans le journal précède un point de contrôle pourront ne pas être ré-exécutées en cas d'incident

Points de validation et Points de contrôle**Points de validation et Points de contrôle**

Transactions : Sérialisabilité et plan d'exécution

- Gestion de la concurrence fondée sur la *sérialisabilité* :

L'exécution concurrente de n transactions doit être équivalente à (avoir le même effet pour chaque transaction que) leur exécution séquentielle

- *Propriétés (ACID)* à assurer par un gérant de transactions :
 1. *Atomicité* : politique du "tout ou rien" ;
 2. *Cohérence* : satisfaction des contraintes d'intégrité ;
 3. *Isolation* : pas d'interférence entre les transactions ;
 4. *Durabilité* : permanence des modifications effectuées.

Sérialisabilité et plan d'exécution

- A et I : par la méthode de reprise
- C : par le programmeur et le sous-système d'intégrité
- D : par les mécanismes de reprise et de contrôle de la concurrence
- *Sérialisabilité* : par le contrôle de la concurrence
- *Plan d'exécution* \mathcal{P} (ou *schedule*) de n transactions :

Ordre sur les opérations des transactions tel que pour toute transaction T_k de \mathcal{P} , si une opération O_i précède une opération O_j dans T_k , alors O_i précède aussi O_j dans \mathcal{P}

- Possibilité d'entrelacement des transactions $\implies \mathcal{P}$ non unique \implies n'autoriser que les plans d'exécution corrects

Sérialisabilité et plan d'exécution

- Exécution séquentielle = *Succession* = Plan correct

Un plan d'exécution \mathcal{P} des transactions T_1, \dots, T_n est une *succession* (ou *serial schedule*) s'il existe une permutation Π de $(1, \dots, n)$ telle que $\mathcal{P} = \langle T_{\Pi(1)}; \dots; T_{\Pi(n)} \rangle$

- *Exécution sérialisable* : Plan correct

Une exécution de T_1, \dots, T_n est *sérialisable* si et seulement si elle donne, pour chaque T_i , le même résultat qu'une succession

- *Problème* : n'autoriser que les exécutions sérialisables
 - Protocoles pessimistes : verrouillage, estampillage
 - Protocoles optimistes : contrôle à la fin d'une transaction

Contrôle de la concurrence : Verrouillage

- *Verrou* : Variable associée à un *granule* et dont la valeur indique le type d'opération possible sur un granule
- *Granule* : unité de verrouillage
 - Base, Relation, Partie de relation, etc.
- *Petite taille* :
 - (+) \nearrow degré concurrence
 - (-) \nearrow temps de son contrôle
- *Grande taille* :
 - (-) \nearrow temps d'attente

Contrôle de la concurrence : Types de verrous

1. *Verrou binaire* :

- Deux états : Libre | Occupé
- (-) Une seule transaction détient un granule

2. *Verrou partagé (Share) et verrou exclusif (Exclusive)* :

- *Accès multiples en lecture*
- *En mise à jour* : Un accès [+ des attentes]
- *Compatibilité des verrous* :

	Partagé	Exclusif
Partagé		
Exclusif		

Contrôle de la concurrence : Types de verrous

3. *Verrou d'intention* :

- Transaction demande un verrou de mise à jour \simeq verrou de lecture avec *intention* d'écriture
- *Compatibilité* :

	Partagé	Exclusif	Mise à jour
Partagé	Oui	Non	Non
Exclusif	Non	Non	Non
Mise à jour	Oui	Non	Non

Gestion des transactions : Verrouillage à deux phases

- *Two Phase Locking Protocol* : le plus souvent implanté
 1. *Phase 1* (expansion) : acquisition
 2. *Phase 2* (rétrécissement) : libération et plus aucune autre acquisition
- Garantie de la sérialisabilité
- Mais verrouillage \implies Risques :
 1. Interblocage (*Deadlock*)
 2. Famine (*Livelock*)
- **Note** : *Two Phase Locking strict*

Verrouillage et Interblocage

- Attentes mutuelles
- *Prévention* : Imposer à toute transaction de verrouiller en avance tous les éléments dont elle a besoin et n'apposer aucun verrou si un de ces éléments n'est pas libre
- *Détection* : Cycle dans un graphe d'attente
 - *Nœuds* : Transactions
 - *Arc* $T1 \rightarrow T2$: $T1$ attend un granule détenu par $T2$
 - *Présence de cycle* :
 1. *Tuer* une transaction
 2. Défaire ses actions
 3. Libérer ses ressources

Verrouillage, Interblocage et Famine

- *Exemple d'interblocage* :
- *Famine* : Attente infinie
 - *Exemple* : Priorité toujours aux mêmes transactions
 - *Solutions* : Priorités dynamiques ou *FIFO*

Contrôle de la concurrence : Estampillage

- *Estampille* : Identifiant de transaction (date de début)
- Technique fondée sur un ordre sur les *estampilles* : sérialisabilité sans interblocage
 - Plan d'exécution sérialisable s'il se conforme à l'ordre
 - *Succession* équivalente = transactions ordonnées sur les estampilles
- Granule G "marqué" par la date t de la dernière transaction qui y a accédé
- Actions sur G estampillées t' , $t' \geq t$: autorisées
- Actions estampillées t'' , $t'' < t$: violation de la sérialisabilité \implies Rollback

Contrôle de la concurrence : Techniques "optimistes"

- Pas de contrôle pendant l'exécution des transactions
- Mises à jour locales à la transaction
- Fin de transactions : test de non violation de la sérialisabilité
- Adaptées aux transactions ayant peu d'interférence

Reprise en cas d'incident (Recovery)

- Reconstruire un état cohérent à partir "du passé" (*Log*)
- État le plus proche possible de l'instant de l'incident
- Stratégie de reprise dépend de la gravité de l'incident
 1. *Reprise "à froid"* : si dégâts importants
 - (a) Charger une sauvegarde de la base
 - (b) Re-exécuter les transactions valides des journaux
 2. *Reprise "à chaud"* : Défaire [et refaire] des actions

Reprise "à chaud", Technique 1 : Mise à jour différée

- Mise à jour effective après le point de validation de la transaction
- \simeq Modifications de données "au brouillon" (copies)
- Cas échec : base inchangée
- Cas succès : Substitution Copie/"Original" (*Shadow paging*)
- Cas incident au moment de la substitution : Refaire certaines actions de transactions validées

Reprise "à chaud", Technique 1 : Mise à jour différée**Reprise "à chaud" : Technique 2 : Mise à jour immédiate**

- Journalisation des modifications avant écriture dans la base avant le point de validation
- Write Ahead Log Protocol :
 - Implanté dans la plupart des SGBD
 - Permet la reprise même en cas d'incident entre l'écriture dans le journal et l'écriture dans la base

Write Ahead Log Protocol**WAL et reprises**

- Cas reprise à chaud : Annuler les transactions non validées (cf. Log)
- Cas reprise à froid :
 1. Charger une base cohérente (exemple : B1)
 2. Charger les journaux adéquats (exemple : J2, J3)
 3. Ré-exécuter automatiquement les transactions validées
- Remarque : cascade de ROLLBACK
 - Rollback T1
 - T2 a utilisé des valeurs modifiées par T1

Concurrence et Reprise : Conclusion

- Sériabilité et ACID"-ité"
- Verrouillage à deux phases (Interblocage)
- Write Ahead Log Protocol
- Shadow Paging
- Reprise en cas d'incident
- Sécurité par duplication (*Mirroring*)
- Organisation de l'exploitation des bases (*officier de sécurité*)
- Transactions plates : même comportement sur tous les SGBD
- Transactions imbriquées : pas de modèle d'exécution universel