Gestion de l'accès concurrent

ELMARI ET NAVATHE, 6TH ED.
CHAPITRE 22

Plan

- Introduction
- Problématique
- Modèle
- Solutions par verrouillage
- Solutions par estampillage
- Synthèse
- Conclusion

Introduction

Objectif

- Garantir l'isolation des transactions sans violer l'atomicité, la cohérence ni la durabilité
- Corollaires
  - Assurer l'exclusion mutuelle des transactions conflictuelles
  - Résoudre les conflits d'accès (rw et ww)
- Exemple
  - Si deux transactions T1 et T2 sont en conflit relativement à l'élément X, il faut déterminer laquelle des deux transactions accède à X et si l'autre doit être annulée ou suspendue

Rappels

- Concurrence : les processus se partagent un même processeur, leurs exécutions s'entrelacent.
- Parallélisme : les processus sont simultanément exécutés sur des processeurs différents.
- En général, les deux en même temps !

Note sur la concurrence et parallélisme

- En général, un modèle adéquat pour la concurrence l'est aussi pour le parallélisme, mais pas l'inverse.
- Les *Concurrent sequential processes* (CSP) de Hoare demeurent un des modèles les plus pertinents.

Références

- ELMASRI, Ramez ; NAVATHE, Shamkant B.; *Fundamentals of database systems ;* Sixth edition, Pearson Addison Wesley, 2011. ISBN 978-0-13-608620-8.
  - chapitre 22
- George Coulouris, Jean Dollimore, Tim Kindberg ; *Distributed systems : concepts and design* ; Forth edition, Addison-Wesley, 2005. ISBN 0-321-26354-5.
  - pour un approfondissement, le livre au complet!

Verrouillage

Verrouillage
(à la recherche de l'isolation)

- verrous binaires
- verrous partagés
  - non convertibles
  - convertibles

Verrous binaires

Règles

1. verrouillé(x) préalable à lire(x) dans T
2. verrouillé(x) préalable à écrire(x) dans T
3. verrouillé(x) préalable à déverrouiller(x) dans T
4. ¬verrouillé(x) préalable à verrouiller(x) dans T
5. lire(x) et écrire(x) se terminent par déverrouiller(x)

Verrous partagés non convertibles

Règles en situation de non-convertibilité

1. verrouiller_*m*(x) avant lire(x) dans T,
   en spécifiant m ∈ {partagé, exclusif}
2. verrouiller_exclusif(x) avant écrire(x) dans T
3. lire(x) et écrire(x) se terminent par déverrouiller(x)
4. déverrouiller(x) que si verrouillé(x) dans T
5. verrouiller_partagé(x) que si ¬verrouillé(x) dans T
6. verrouiller_exclusif(x) que si ¬verrouillé(x) dans T

Verrous partagés convertibles

Règles en situation de convertibilité

- Assouplir les règles 5 et 6

  5bis verrouiller_partagé(x) que si
  ¬verrouillé(x) ou
  possède verrou exclusif dans T

  6bis verrouiller_exclusif(x) que si
  ¬verrouillé(x) ou
  possède le seul verrou partagé dans T

- Nécessite le maintien de la liste des détenteurs de verrous (en plus des verrous eux-mêmes).

Interblocage
(étreinte fatale – *dead lock*)

Définition

- lorsque chacune des transactions d'un ensemble de transactions attend la libération d'un élément verrouillé par une autre transaction du même ensemble

Exemple

T1	T2
v_par(y)	
	v_par(x)
v_ex(x)	
	v_ex(y)

Verrouillage bi-phase

- Modèle 2PL : garantir a priori l'exécution en séparant les actions en deux phases :
  - acquisition
  - libération
- Variantes
  - basique : tel quel;
  - conservatrice : prédéclaration des verrous;
  - stricte : libération des seuls verrous partagés;
  - rigoureuse : libération des verrous à la fin seulement.
- Limite
  - certains programmes sérialisables sont refusés.

Verrouillage bi-phase conservateur

- Méthode
  - prédéclaration des verrous
- Comportement
  - empêche d'entrer en transaction à moins d'obtenir tous les accès,
  - une transaction est exécutée en entier durant la phase de libération.
- Corolaires
  - risque de famine,
  - augmentation de l'écart-type des délais,
  - difficulté voire impossibilité de mise en oeuvre dans un contexte de SGBD.

Verrouillage bi-phase strict

- Méthode
  - libération des seuls verrous partagés
- Comportement
  - …
- Corolaires
  - …

Verrouillage bi-phase rigoureux

- Méthode
  - libération des verrous à la fin seulement
- Comportement
  - …
- Corolaires
  - …

Techniques de gestion des transactions

- Prévention 2PL
  - strict, ok
  - autres ⇒ ordonnancement strict des attributs
  - non praticable dans un contexte de SGBD (calculabilité, tri des attributs)
- Prévention avec estampilles
  - BTO - WD (wait die)
  - STO - WW (wound wait)
- Prévention conservatrice
  - NW (no wait)
  - CW (cautious wait)
- Détection de l'interblocage
  - Maintien et analyse du graphe des attentes
  - Mise hors délai (solution pratique)
- Famine
  - Mise hors délai (problématique d'équitabilité)

Estampillage

Principe

- Toutes les transactions sont estampillées en ordre strictement croissant
- Estampille de lecture d'un élément : la plus grande des estampilles des transactions courantes ayant effectivement lu l'élément
- Respectivement pour l'écriture

Algorithme de base

BTO

- 1. Transaction T issues a write_item(X) operation:
  - If read_TS(X) > TS(T) or if write_TS(X) > TS(T), then an younger transaction has already read the data item so abort and roll-back T and reject the operation.
  - If the condition in part (a) does not exist, then execute write_item(X) of T and set write_TS(X) to TS(T).
- 2. Transaction T issues a read_item(X) operation:
  - If write_TS(X) > TS(T), then an younger transaction has already written to the data item so abort and roll-back T and reject the operation.
  - If write_TS(X) ≤ TS(T), then execute read_item(X) of T and set read_TS(X) to the larger of TS(T) and the current read_TS(X).

Algorithme strict
STO

- Transaction T issues a write_item(X) operation:
  - If TS(T) > read_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).
- Transaction T issues a read_item(X) operation:
  - If TS(T) > write_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).

Règle de Thomas

TTO

- If read_TS(X) > TS(T) then abort and roll-back T and reject the operation.
- If write_TS(X) > TS(T), then just ignore the write operation and continue execution. This is because the most recent writes counts in case of two consecutive writes.
- If the conditions given in 1 and 2 above do not occur, then execute write_item(X) of T and set write_TS(X) to TS(T).

Synthèse

	B2PL	C2PL	S2PL	R2PL	S2PLWD	S2PLWW	S2PLNW	S2PLCW	BTO	STO	TTO
récupérabilité (s-prog)	oui	oui	oui	oui	oui	oui	oui	oui	oui	oui	oui
c-ser garantie	oui	oui	oui	oui	oui	oui	oui	oui	oui	oui	non
c-ser complète	non	non	non	non	non	non	non	non	non	non	non
fermeture a priori	non	oui	non	non	non	non	non	non	non	non	non
annulation et reprise	oui+	oui	oui	oui+	oui	oui	oui	oui-	oui	oui	oui
annulation en cascade	non-	non	non-	non-	non-	non-	non-	non-	oui	oui	oui
interblocage	oui+	non	oui=	oui-	non-	non-	non	non	non	non	non
famine	oui+	non	oui=	oui-	non	non	non	non	non	non	non
facilité de meo	x=	x+	x=	x-	x-	x-	x-	x-	x++	x++	x+
applicable en pratique	non	non	oui	non	oui	oui	oui	oui	oui	oui	oui
contexte								t+f-	t-f+	t-f+	

Fin