Gestion de l'accès concurrent



ADAPTATION DE

ELMASRI ET NAVATHE, 6TH ED. CHAPITRE 22

PAR LUC LAVOIE

UTILISÉ CONFORMÉMENT AUX CONDITIONS ÉTABLIES PAR LES AUTEURS

2023-09-05

1

Plan



- Introduction
- Problématique
- Modèle
- Solutions par verrouillage
- Solutions par estampillage
- Synthèse
- Et les autres solutions?
- Conclusion

2023-09-05

Introduction

2023-09-05

3

Objectif



- Garantir l'isolation des transactions sans violer l'atomicité, la cohérence ni la durabilité
- Corollaires
 - Assurer l'exclusion mutuelle des transactions conflictuelles
 - o 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

2023-09-05

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.

2023-09-05

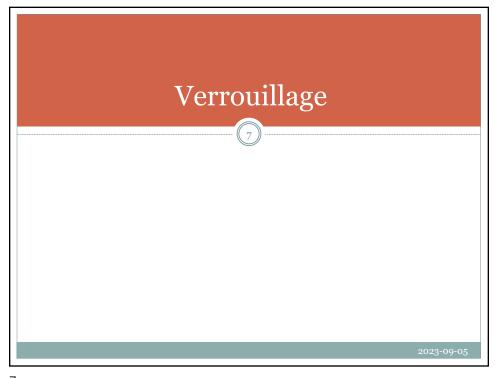
5

Références



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

023-09-05



/

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)

2023-09-05

9

Verrous partagés non convertibles



Règles en situation de non-convertibilité

- verrouiller_m(x) avant lire(x) dans T, en spécifiant m ∈ {partagé, exclusif}
- 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

2023-09-05

Verrous partagés convertibles

11

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).

2023-09-05

11

Interblocage (étreinte fatale – *dead lock*) Définition **Exemple** T1 T₂ • lorsque chacune des v_par(y) transactions d'un v_par(x) ensemble de v_ex(x) transactions attend la v_ex(y) libération d'un élément verrouillé par une autre transaction du même ensemble

Verrouillage bi-phase



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

2023-09-05

13

Verrouillage bi-phase conservateur



- Méthode
 - o 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
 - o risque de famine,
 - o augmentation de l'écart-type des délais,
 - o difficulté voire impossibilité de mise en oeuvre dans un contexte de SGBD.

2023-09-05

Verrouillage bi-phase strict

15

- Méthode
 - o libération des seuls verrous partagés
- Comportement
 - o ...
- Corolaires
 - o ...

2023-09-05

15

Verrouillage bi-phase rigoureux



- Méthode
 - o libération des verrous à la fin seulement
- Comportement
 - o ...
- Corolaires
 - o ...

2023-09-05

Techniques de gestion des transactions



- Prévention 2PL
 - o strict, ok
 - o autres ⇒ ordonnancement strict des attributs
 - o non praticable dans un contexte de SGBD (calculabilité, tri des attributs)
- Prévention avec estampilles
 - o BTO WD (wait die)
 - o STO WW (wound wait)
- Prévention conservatrice
 - o NW (no wait)
 - o 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)

2023-09-05

17

Estampillage



2023-09-05

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

2023-09-05

19

Algorithme de base



- 1. La transaction T lance l'opération write_item(X,v):
 - × Si read_TS(X) > TS(T) ou write_TS(X) > TS(T) alors
 - o une transaction concurrente a déjà lu ou modifié X, conséquemment : rejeterl'opération write_item(X,v) et annuler la transation T.
 - × Sinon
 - excécuter l'opération write_item(X,v) et affecter la valeur de TS(T) à l'estampille write_TS(X).
- 2. La transaction T lance l'opération read_item(X,v) :
 - \times Si write_TS(X) > TS(T) alors
 - o une transaction concurrente a déjà modifié X, conséquemment : rejeter l'opération read_item(X,v) et annuler la transaction T.
 - × Sinon
 - exécuter read_item(X,v) et affecter la plus grande des valeurs parmi TS(T) et read_TS(X) à l'estampille read_TS(X).

2023-09-05

Algorithme strict



- La transaction T lance l'opération write_item(X,v) :
 - \times Si TS(T) > read_TS(X) alors
 - o reporter T jusqu'à ce que la transaction T' ayant lu ou modifié X soit terminée (complétée ou annulée)
 - × sinon
 - o procéder.
- La transaction T lance l'opération read_item(X,v) :
 - × Si TS(T) > write_TS(X) alors
 - o reporter T jusqu'à ce que la transaction T' ayant lu ou modifié X soit terminée (complétée ou annulée)
 - × sinon
 - o procéder.

2023-09-05

21

Règle de Thomas

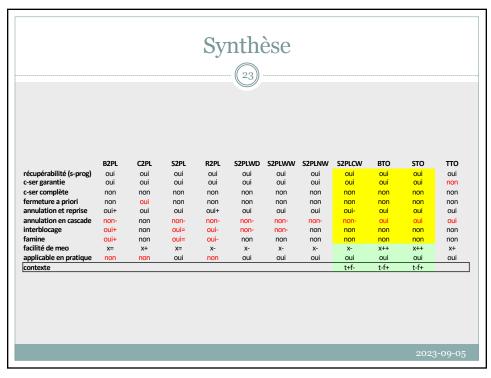
(TTO: une variante de STO)



La règle d'écriture est modifiée ainsi

- \circ si read TS(X) > TS(T) alors
 - × rejeter l'opération et annuler T.
- o sinon
 - \times si write_TS(X) > TS(T)
 - /* Il s'agit d'une deuxième écriture annulant la première */
 - o ignorer l'opération et poursuivre l'execution.
 - o /* s'il devait en résulter un conflit, la règle précédente le détectera */
 - sinon
 - \circ /* read_TS(X) \leq TS(T) **et** write_TS(X) \leq TS(T) */
 - \circ executer write_item(X) et affecter la valeur write_TS(X) à TS(T).

2023-09-05



23

Et les autres solutions?



- MVCC en général, voir
 - o https://en.wikipedia.org/wiki/Multiversion_concurrency_control
 - o https://fr.wikipedia.org/wiki/Multiversion_Concurrency_Control (traduction perfectible de la référence précédente)
 - o https://en.wikipedia.org/wiki/List_of_databases_using_MVCC
- MVCC et PostgreSQL, voir
 - https://momjian.us/main/writings/pgsql/mvcc.pdf

2023-09-05

