INF4410

Module 6 : Transactions

6.1 Un serveur gère plusieurs valeurs a1, ...an. Ce serveur offre deux opérations à ses clients: value = Read(i), Write(i, value). Les transactions T et U sont définies comme suit. Donnez deux sérialisations équivalentes de ces transactions?

T: x=Read(j); y=Read(i); Write(j,44); Write(i,33);

U: x=Read(k); Write(i,55); y=Read(j); Write(k,66);

Si T procède entièrement avant U, le résultat final sera:

T x=aj0; T y=ai0; U x=ak0; U y=44; ai=55; aj=44; ak=66;

L'entrelacement suivant des opérations produit le même résultat:

T x=Read(j); T y=Read(i); U x=Read(k); T Write(j,44); T Write(i,33); U Write(i,55); U y=Read(j); U Write(k,66);

Si U procède avant T, le résultat final sera:

T x=aj0; T y=55; U x=ak0; U y=aj0; ai = 33; aj = 44; ak = 66;

L'entrelacement suivant des opérations produit le même résultat:

T x=Read(j); U x=Read(k); U Write(i,55); T y=Read(i); U y=Read(j); U Write(k,66); T Write(j,44); T Write(i,33);

6.2 Montrez pourquoi lorsqu'une transaction a relâché un verrou elle ne doit plus en obtenir afin de permettre l'équivalence avec la sérialisation?

Soit deux transactions T et U:

T: x=Read(i); Write(j,44);
U: Write(i,55); Write(j,66);

Avec T avant U, le résultat est: x=ai0; ai=55; aj=66;

Si T commence avant U mais les verrous sont relâchés trop tôt, nous pourrions avoir:

Lock T i; x=Read(i); unlock T i; lockU i; Write(i,55); lockU j; Write(j,66); CommitU; unlockU i,j; lockT j; Write(j,44); unlockT j; CommitT;

6.3 Soit deux transactions T et U, lesquelles de ces sérialisations sont valides? Sont possibles avec un verrouillage en deux phases?

T: x = read(i); write(j,44);
U: write(i,55); write(j,66);

a) T x=Read(i); U write(i,55); T write(j,44); U write(j,66);

b) U write(i,55); U write(j,66); T x=Read(i); T write(j,44);

c) T x=Read(i); T write(j,44); U write(i,55); U write(j,66);

d) U write(i,55); T x=Read(i); U write(j,66); T write(j,44);

Les 4 sérialisations proposées sont valides (a et c pour T avant U, b et d pour U avant T). En a et d, le verrou sur i bloquerait la deuxième opération et ces sérialisations ne pourraient se produire.

6.4 Expliquez pourquoi il ne faut pas relâcher les verrous avant de se commettre, même après la fin des opérations.

Si une autre transaction utilise les valeurs modifiées et la première transaction échoue, l'autre transaction aura utilisé des valeurs incorrectes. Il serait possible cependant de relâcher les verrous de lecture.

6.5 Une synchronisation optimiste est appliquée aux transactions T et U qui sont actives en même temps. Discutez ce qui arrive dans chacun des cas suivants:

T: x=Read(i); Write(j,44);
U: Write(i,55); Write(j,66);

a) T est prête en premier et la validation en reculant est utilisée? b) En avançant?

c) U est prête en premier et la validation en reculant est utilisée? d) En avançant?

a) Reculant: Read(i) est vérifié avec les écritures de transactions concurrentes qui sont terminées. Correct car U qui veut écrire i n'as pas encore terminé. Pour U, aucune lecture n'est effectuée et il n'y a donc pas de conflit.

b) Avançant: Write(j,44) est vérifié avec les lectures de transactions concurrentes actives (pas de lecture dans U). Lorsque U termine, les écritures ne posent pas problème car T est terminée.

c) Reculant: Correct, U ne fait pas de lecture. Read(i) est vérifié avec les écritures de transactions concurrentes terminées, un problème est détecté car la valeur a été modifiée par U; T doit être repris.

d) Avançant: Write(i,55) est vérifié avec les lectures de transactions concurrentes actives, un problème est détecté car T veut lire i; la transaction U est annulée. Pour T, Write(j,44) est vérifié avec les lectures de transactions concurrentes actives, ce qui est correct.

6.6 Deux transactions T et U utilisent des compteurs de temps pour vérifier leur cohérence. Initialement ai = 10 au temps t0, et aj = 20 au temps t0.

Voici deux séquences possibles à vérifier pour les opérations de T et U:

T: Début;
U: Début;
U: Write(i,55);
U: Write(j,66);
T: x=Read(i);
T: Write(j,44);
U: Compléter;

T: Début; T obtient la valeur t1;
U: Début; U obtient la valeur t2;
U: Write(i,55); t2 >= t0 temps dernière lecture, et t2 > t0 écriture de ai. Nouvelle version tentative de ai à 55 avec temps d'écriture t2.
U: Write(j,66); t2 >= t0 temps dernière lecture, et t2 > t0 écriture de aj. Nouvelle version tentative de aj à 66 avec temps d'écriture t2.
T: x=Read(i); t1 >= t0 temps dernière écriture de ai; x = 10, dernière lecture = t1;
T: Write(j,44); t1 >= t1 temps dernière lecture, et t1 > t0 temps de dernière écriture de aj. Nouvelle version tentative de aj à 44 avec temps d'écriture t1;
U: Compléter; ai = 55 à t2, dernière lecture t1. aj a une version tentative à t1. La version de U à 66 avec temps t2 doit attendre.
T: Compléter; aj a une version tentative à 44 avec temps t1 et temps de lecture maximum t0. Correct suivi de la version tentative de aj à 66 avec temps t2.

T: Début;
U: Début;
U: Write(i,55);
U: Write(j,66);

U: Compléter;
T: x=Read(i);
T: Write(j,44);

Ce second ordonnancement complète la transaction U avant que ai ne soit lu. A ce moment, ai a la valeur 55 au temps t2 et dernière lecture en t0 et aj a la valeur 66 au temps t2 et dernière lecture t0.

Lorsque T veut lire ai, il est trop tard. t1 >= t2 temps de dernière écriture de ai est faux et la transaction T doit être annulée.

6.7 Expliquez comment une transaction imbriquée s'assure que toutes les sous-transactions sont correctement commises ou annulées.

Chaque sous-transaction maintient une liste des transactions prêtes à se commettre. La transaction de premier niveau peut donc faire une recherche en profondeur des transactions prêtes pour la validation finale.

6.8 Les transactions T, U, et V sont définies comme suit. Décrivez le journal produit par cette application si un verrouillage en deux phases est utilisé et U acquiert i et j avant T? Décrivez comment ce journal peut être utilisé pour récupérer en cas de panne?

T: x=Read(i); Write(j,44);
U: Write(i,55); Write(j,66);
V: Write(k,77); Write(k,88);

P0: ...; P1: Write(i,55);
P2: Write(j,66);
P3: Prépare U(i:P1, j:P2), P0;
P4: Compléter U, P3;
P5: Write(j,44);
P6: Préparer T(j:P5), P4;
P7: Compléter T, P6;
P8: Write(k,88);
P9: Préparer V(k:P8), P7;
P10: Compléter V, P9;

Pour récupérer, les variables a1,...an sont replacées à leur valeur par défaut. Ensuite le journal est lu en reculant. Le récupérateur voit que V est complété et à P9 que V contient P8; il met donc ak à 88. Il continue à reculer à P7, voit que T est complété et peut ensuite lire aj à 44 en voyant P6 qui pointe à P5. La chaîne remonte ensuite à P4, où U est marqué complété; ceci fait reculer à P3 qui pointe vers une valeur de j qui est ignorée car la valeur la plus récente est déja lue, et vers une valeur pour i ce qui permet de trouver que ai devient 55.

6.9 Les transactions T et U utilisent un contrôle optimiste de la concurrence basé sur les compteurs de temps. Décrivez l'information écrite dans le journal si T débute en premier.

T x=Read(i);
U Write(i,55);
U Write(j,66);
T Write(j,44);
U Commit;
T Commit;

P0;
P1 i=55;
P2 j=66;
P3 j=44;
P4 Prépare U(i:P1,j:P2), P0;
P5 U attente, P4;
P6 Prépare T(j:P3), P5;
P7 Commettre T, P6;
P8 Commettre U, P7;

L'entrée P5 permet d'assurer que U termine, même si elle doit avoir lieu après T, au cas où T serait annulée. Les entrées sont toujours par ordre de temps car la récupération se fait en lisant en reculant le fichier de journal. Si le serveur tombe avant les deux commettre, la transaction U sera terminée et T sera annulée. Si le serveur tombe après P7, les deux transactions termineront.

Dans la mesure où les transactions ne peuvent pas vraiment bloquer, il n'y a pas d'inconvénient à écrire tôt les informations dans le journal. L'avantage est de mieux répartir la charge d'écriture dans le journal.

6.10 Les transactions T et U utilisent un contrôle optimiste de la concurrence basé sur la validation en reculant. Décrivez l'information écrite dans le journal si T débute en premier.

T x=Read(i);
U Write(i,55);
U Write(j,66);
T Write(j,44);
U Commit;
T Commit;

P0;
P1 i=55;
P2 j=66;

P3 Prépare U(i:P1,j:P2), P0;
P4 Commettre U, P3;
P5 j=44;
P6 Prépare T(j:P5), P4;
P7 Commettre T, P6;

Lorsqu'une transaction est validée, le numéro de transaction apparaît dans le journal. U est validée car elle ne fait aucune lecture. Par contre, la transaction T est annulée car elle lit i qui est écrit par U. T est redémarrée et peut finalement compléter.