

Master Informatique Université Bordeaux I
Bases de Données Avancées
Exercices (Retoré)



Pour résumer ce cours:

1. Trois techniques de reprise sur panne avec à chaque fois la possibilité de checkpoints sans accord *non quiescent*
 - (a) undo loggin
 - (b) redo loggin
 - (c) undo/redo loggin
2. Trois techniques de gestion des accès concurrents:
 - (a) énoncé du problème: sérialisabilité, sérialisabilité par permutation, test par construction du graphe de précedence;
 - (b) verrous simples: schémas cohérents (actions précédées de verrouillage; déverrouillage ultérieur), schéma légaux (au plus un verrou sur un élément); verrous de plusieurs types (partagés et exclusifs) , modification des notions de cohérence et de légalité en conséquences
 - (c) estampillage (time stamps) simple ou avec copies multiples
 - (d) validation

Voici quelques exercices types sur ce programme (les premiers avaient été distribués en cours). Pour les exercices de réflexion, pas d'exercice type, mais vous pouvez réfléchir aux questions suivantes: quel type de correction garantissent les techniques de gestion des accès concurrents? comment les garantissent-elles?

I. Reprise sur panne

Exercice A

Voici les expressions enregistrées sur un fichier undo log par deux transactions T et U:

```
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<U,D,40>
<COMMIT U>
<T,E,50>
<COMMIT T>
```

Décrire les opérations d'une reprise sur panne, pour chacune des fins suivantes du fichier log.

1. <START U>
2. <COMMIT U>
3. <T,E,50>>
4. <COMMIT T>

Exercice B

Voici les expressions enregistrées sur un fichier undo log:

```
<START S>
<S,A,60>
<COMMIT S>
<START T>
<T,A,10>
<START U>
<U,B,20>
<T,C,30>
<START V>
<U,D,40>
<V,F,70>
<COMMIT U>
<T,E,50>
<COMMIT T>
<V,B,80>
<COMMIT V>
```

Supposons qu'on démarre un checkpoint sans accord après l'écriture de:

1. <S,A,60>
2. <T,A,10>

3. <U, B, 20>>
4. <U, D, 40>
5. <T, E, 50>

Dans chacun des cas,

1. déterminer la place de la fin du checkpoint.
2. pour chaque moment possible de panne jusqu' où doit on remonter dans le fichier log?

Exercice C

Les données et les questions dont sont les mêmes que dans le premier exercice, mais il s'agit d'un fichier log de type redo.

Exercice D

Les données et les questions dont sont les mêmes que dans le second exercice, ainsi que les poitn de panne possibles, mais il s'agit d'un fichier log de type redo. Poru chacun de spoitns de panne envisagé, dire:

(D.i) A quel endroit le <END CKPT> peut-il avoir été écrit?

(D.ii) Jusqu' où est-il nécessaire de remonter dans l'historique pour trouver les transactions incomplètes? On considérera deux cas, suivant que <END CKPT> a été écrit avant la panne ou non.

Exercice E

Voici un fichier undo/redo:

```
<START T>
<T, A, 4, 5>
<START U>
<COMMIT T>
<U, B, 9, 10>
<START CKPT(U)>
<U, C, 14, 15>
<START W>
<W, D, 19, 20>
<END CKPT>
<COMMIT U>
<COMMIT W>
```

Sue ce passe-t-il en cas de panne

1. après <COMMIT W>
2. après <COMMIT U>

Exercice F

Voici les expressions enregistrées sur un fichier log undo/redo par deux transactions T et U:

```
<START T>
<T,A,10,11>
<START U>
<U,B,20,21>
<T,C,30,31>
<U,D,40,41>
<COMMIT U>
<T,E,50,51>
<COMMIT T>
```

Décrire les opérations d'une reprise sur panne, pour chacune des fins suivantes du fichier log.

1. <START U>
2. <COMMIT U>
3. <T,E,50,51>
4. <COMMIT T>

Exercice G

Voici les expressions enregistrées sur un fichier undo log:

```
<START S>
<S,A,60,61>
<COMMIT S>
<START T>
<T,A,61,62>
<START U>
<U,B,20,21>
<T,C,30,31>
<START V>
<U,D,40,41>
<V,F,70,71>
<COMMIT U>
<T,E,50,51>
<COMMIT T>
<V,B,21,22>
<COMMIT V>
```

Supposons qu'on démarre un checkpoint sans accord après l'écriture de:

1. <S,A,60,61>
2. <T,A,61,62>
3. <U,B,20,21>
4. <U,D,40,41>

5. $\langle T, E, 50, 51 \rangle$

Dans chacun des cas,

1. déterminer les places possibles de la fin du checkpoint.
2. pour chaque moment possible de panne jusqu'où doit on remonter dans le fichier log? on considérera deux cas, suivant que le $\langle \text{END CKPT} \rangle$ a été écrit ou non.

II. Gestion des accès concurrents

Notation $o_i(E)$: la transaction i effectue sur l'élément E l'opération x qui peut être:

r lecture (read)

w écriture (write)

Exercice H

Dans un système par verrouillage l'opération x ci-dessus peut aussi être:

l verrouillage (lock)

u déverrouillage (unlock)

On considère les transactions suivantes:

$S_1 : r_1(A); A := A + 2; w_1(A); r_1(B); B := B * 3; w_1(B);$

$S_2 : r_2(B); B := B * 2; w_2(B); r_2(A); A := A + 3; w_2(A);$

$T_1 : l_1(A); r_1(A); A := A + 2; w_1(A); u_1(A); l_1(B); r_1(B); B := B * 3; w_1(B); u_1(B);$

$T_2 : l_2(B); r_2(B); B := B * 2; w_2(B); u_2(B); l_2(A); r_2(A); A := A + 3; w_2(A); u_2(A);$

$U_1 : l_1(A); r_1(A); A := A + 2; w_1(A); l_1(B); r_1(B); B := B * 3; w_1(B); u_1(A); u_1(B);$

$U_2 : l_2(B); r_2(B); B := B * 2; w_2(B); l_2(A); r_2(A); A := A + 3; w_2(A); u_2(B); u_2(A);$

(H.i) Trouver un schéma d'exécution de S_1 et S_2 (c'est-à-dire T_1 et T_2 sans les verrouillages ni déverrouillages) qui soit sérialisable mais pas sérialisable par permutation.

(H.ii) Donner un schéma d'exécution de T_1 et T_2 interdit par les verrous.

(H.iii) Combien y a-t-il de schémas légaux pour T_1 et T_2 ?

(H.iv) Combien y a-t-il de schémas légaux et sérialisables pour T_1 et T_2 ?

(H.v) Combien y a-t-il de schémas légaux et sérialisables par permutation pour T_1 et T_2 ?

(H.vi) T_1 et T_2 sont-elles des transactions à deux phases?

(H.vii) U_1 et U_2 sont-elles des transactions à deux phases?

(H.viii) Combien y a-t-il de schémas légaux d'exécution de U_1 et U_2 (en ne prenant en considération que les opérations effectives, les r et les w).

Exercice I

Dans un système avec des verrous partagés, l'opération de déverrouillage est inchangée, mais l'opération de verrouillage l ci-dessus est remplacée par deux opérations de verrouillage:

xl verrouillage avec un verrou exclusif

sl verrouillage avec un verrou possiblement partagé

On considère deux schémas d'exécution construits sur trois transactions 1,2,3:

$\mathcal{S}_1 : r_1(A); r_2(B); r_3(C); w_1(B); w_2(C); w_3(D);$

$\mathcal{S}_2 : r_1(A); r_2(B); r_3(C); r_1(B); r_2(C); r_3(D); w_1(C); w_2(D); w_3(E);$

(I.i) Pour chacun des deux schémas:

- Insérer des verrous partagés avant chaque lecture si elle n'est pas suivie d'une écriture du même élément par la même transaction.
- Placer un verrou exclusif en face de toutes les autres écritures ou lectures.
- Placer les déverrouillages à la fin de chaque transaction.

(I.ii) Décrire l'exécution de la suite d'opérations ainsi obtenue.

Exercice J

Notation spécifique pour l'estampillage:

- st_i : la transaction i commence.

On considère les séquence d'opération suivantes, ou st_i signifie que la transaction i commence:

$\mathcal{S}_1 : st_1; st_2; st_3; st_4; w_1(A); w_3(A); r_4(A); r_2(A);$

$\mathcal{S}_2 : st_1; st_2; st_3; st_4; w_1(A); w_4(A); r_3(A); w_2(A);$

$\mathcal{S}_3 : st_1; st_2; st_3; st_4; w_1(A); w_2(A); w_3(A); r_2(A); r_4(A);$

(J.i) Décrire l'exécution de chacune des suites d'opérations avec un système d'estampillage simple.

(J.ii) Décrire l'exécution de chacune des suites d'opérations avec un système d'estampillage multiple.

Exercice K

Notation spécifique pour les événements utiles à un système par validation:

- $R_i X$ la transaction i commence, et son ensemble d'éléments lus est X .
- V_i la transaction i essaie de valider.
- $W_i X$ la transaction i finit, et son ensemble d'éléments écrits est X .

On considère les suites suivantes d'événements:

$\mathcal{S}_1 : R_1\{A,B\}; R_2\{B,C\}; V_1R_3\{C,D\}; V_3; W_1\{A\}; V_2; W_2\{A\}; W_3\{B\};$

$\mathcal{S}_1 : R_1\{A,B\}; R_2\{B,C\}; V_1R_3\{C,D\}; V_3; W_1\{C\}; V_2; W_2\{A\}; W_3\{D\};$

$\mathcal{S}_2 : R_1\{A,B\}; R_2\{B,C\}; R_3\{C\}; V_1; V_2; V_3; W_1\{A\}; W_2\{B\}; W_3\{C\};$

$\mathcal{S}_3 : R_1\{A,B\}; R_2\{B,C\}; R_3\{C\}; V_1; V_2; V_3; W_1\{C\}; W_2\{B\}; W_3\{A\};$

$\mathcal{S}_4 : R_1\{A,B\}; R_2\{B,C\}; R_3\{C\}; V_1; V_2; V_3; W_1\{A\}; W_2\{C\}; W_3\{B\};$

Décrire dans chacun des cas l'exécution de la séquence d'opérations dans un système par validation.