Carnegie Mellon University
Department of Computer Science
15-415/615- Database Applications
C. Faloutsos & A. Pavlo, Spring 2014
Prepared by Shen Wang

DUE DATE: Thu, 4/24/2014, 1:30pm

Homework 8

IMPORTANT
- **Deposit hard copy** of your answers in **class at 1:30pm on Thu, 4/24/2014**.
- Separate answers, as usually, i.e., please each question on a separate page, with the usual info (andrewID, etc)

Reminders
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.
- **Typeset** all of your answers whenever possible. Illegible handwriting may get no points, at the discretion of the graders.
- **Late homeworks**: please email late homeworks
  - to all TAs
  - with the subject line exactly `15-415 Homework Submission (HW 8)`
  - and the count of slip-days you are using.

For your information:
- Graded out of **100** points; **4** questions total
- Rough time estimate: ≈4 hours (∼1 hour for each question)

Revision : 2014/05/05 15:52

Question	Points	Score
Serializability and 2PL	20	
Deadlock Detection and Prevention	30	
Hierarchical Locking	30	
B+ tree Locking	20	
Total:	100	

Question 1: Serializability and 2PL . . . . [20 points]

Submit on separate page
Course: 15-415/615; HW: ; Q:
Name: ____________________; andrew-id: ____________________; late days:

(a) Yes/No questions:

i. [**2 points**] All serial transactions are both conflict serializable and view serializable.
■ **Yes** □ No

ii. [**2 points**] For any schedule, if it is view serializable, then it must be conflict serializable.
□ Yes ■ **No**

iii. [**2 points**] Under 2PL protocol, there can be schedules that are not serial.
■ **Yes** □ No

iv. [**2 points**] Any transaction produced by 2PL must be conflict serializable.
■ **Yes** □ No

v. [**2 points**] Strict 2PL guarantees no deadlock.
□ Yes ■ **No**

(b) Serializability:
Consider the schedule given below in Table 1. R(·) and W(·) stand for 'Read' and 'Write', respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
T_1			R(A)	W(A)			R(C)	W(C)				
T_2					R(B)	W(B)						
T_3	R(A)	W(A)							R(C)	W(C)	R(B)	W(B)

Table 1: A schedule with 3 transactions

i. [**1 point**] Is this schedule serial?
□ Yes ■ **No**

ii. [**3 points**] Give the dependency graph of this schedule.

Solution:

- $T_3 \rightarrow T_1$ because of A
- $T_1 \rightarrow T_3$ because of C
- $T_2 \rightarrow T_3$ because of B

iii. [**1 point**] Is this schedule conflict serializable?
□ Yes ■ **No**

iv. [**3 points**] If you answer "yes" to (iii), provide the equivalent serial schedule. If you answer "no", briefly explain why.

Solution: This schedule is not conflict serializable because there exists a cycle ($T_3 \rightarrow T_1 \rightarrow T_3$) in the dependency graph.

v. [**2 points**] Could this schedule have been produced by 2PL?
☐ Yes ■ **No**

Question 2: Deadlock Detection and Prevention . . . . . [30 points]

Submit on separate page
Course: 15-415/615; HW: ; Q:
Name: ______________; andrew-id: ______________; late days:

(a) Deadlock Detection:
Consider the following lock requests in Table 2. And note that

- S(·) and X(·) stand for 'shared lock' and 'exclusive lock', respectively.
- T_1, T_2 and T_3 represent three transactions.
- LM stand for 'lock manager'.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7
T_1	S(D)	S(A)			X(C)		S(B)
T_2			S(A)	X(B)			
T_3						S(C)	
LM	g						

Table 2: Lock requests of 3 transactions

i. **[6 points]** For the lock requests in Table 2, determine which lock will be granted or blocked by the lock manager. Please write 'g' in the LM row to indicate the lock is granted and 'b' to indicate the lock is blocked. For example, in the table, the first lock (S(D) at time t_1) is marked as granted.

Solution:

- S(A) at t_2: g
- S(A) at t_3: g
- X(B) at t_4: g
- X(C) at t_5: g
- S(C) at t_6: b
- S(B) at t_7: b

ii. **[4 points]** Give the wait-for graph for the lock requests in Table 2.

Solution: $T_3 \rightarrow T_1 \rightarrow T_2$

iii. **[3 points]** Determine whether there exists a deadlock in the lock requests in Table 2, and briefly explain why.

Solution: There will be no deadlock because the wait-for graph is acyclic.

(b) Deadlock Prevention:
Consider the following lock requests in Table 3. Again,

- S(·) and X(·) stand for 'shared lock' and 'exclusive lock', respectively.
- T_1, T_2 and T_3 represent three transactions.
- LM_1, LM_2 and LM_3 represent three lock managers with different policies.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7
T_1	S(D)		S(A)		X(C)		
T_2				S(C)		X(B)	
T_3		X(B)					X(A)
LM_1	g						
LM_2	g						
LM_3	g						

Table 3: Lock requests of 3 transactions with multiple lock managers

i. [**6 points**] For the lock requests in Table 3, determine which lock request will be granted, blocked or aborted by the lock manager 1 (LM_1), which has **no** deadlock prevention policy. **Please write 'g' for grant, 'b' for block and 'a' for abort**. Again, example is given in the first column.

Solution:

- X(B) at t_2: g
- S(A) at t_3: g
- S(C) at t_4: g
- X(C) at t_5: b
- X(B) at t_6: b
- X(A) at t_7: b

ii. [**5 points**] Give the wait-for graph for the lock requests in Table 3. Give a one-sentence reason why the lock requests in Table 3 under LM_1 result in a deadlock.

Solution:

- $T_1 \rightarrow T_2$
- $T_2 \rightarrow T_3$

- $T_3 \rightarrow T_1$

The lock requests have a deadlock because there is a cycle in the wait-for graph.

iii. [**3 points**] To prevent deadlock, we use lock manager 2 (LM_2) that adopts the **Wait-Die** policy. We assume that in terms of priority: $T_1 > T_2 > T_3$. Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a') by LM_2. Follow the same format as the previous question.

Solution:

- X(B) at t_2: g
- S(A) at t_3: g
- S(C) at t_4: g
- X(C) at t_5: b
- X(B) at t_6: b
- X(A) at t_7: a

iv. [**3 points**] Now we use lock manager 3 (LM_3) that adopts the **Wound-Wait** policy. Again, we assume that in terms of priority: $T_1 > T_2 > T_3$. Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a') by LM_3. Follow the same format as the previous question.

Solution:

- X(B) at t_2: g
- S(A) at t_3: g
- S(C) at t_4: g
- X(C) at t_5: g, abort t_4
- X(B) at t_6: g, abort t_2 (or doesn't exist because it is already aborted)
- X(A) at t_7: b

Question 3: Hierarchical Locking .... [30 points]

Submit on separate page
Course: 15-415/615; HW: ; Q:
Name: ________________; andrew-id: ________________; late days:

Consider a Database (D) consisting of two tables, Movies (M) and PlayIn (P). In specific:

- Movies(<u>mid</u>, movie_name, movie_rating), spans 300 pages, namely M_1 to M_{300}
- PlayIn(<u>mid, actor_name</u>, actor_rating), spans 600 pages, namely P_1 to P_{600}

Further, **each page contains 100 records**, and we use the notation $P_3 : 20$ to represent the 20^{th} record on the third page of the PlayIn table. Similarly, $M_5 : 10$ represents the 10^{th} record on the fifth page of the Movies table.
We use Multiple-granularity locking, with **S, X, IS, IX** and **SIX** locks, and **four levels of granularity**: (1) *database-level (D)*, (2) *table-level (M, P)*, (3) *page-level* ($M_1 - M_{300}$, $P_1 - P_{600}$), (4) *record-level* ($M_1 : 1 - M_{300} : 100$, $P_1 : 1 - P_{600} : 100$).
For each of the following operations on the database, please determine the sequence of lock requests that should be generated by a transaction that want to carry out these operations efficiently.
Please follow the format of the examples listed bellow:

- write **"IS(D)"** for a request of **database-level IS lock**
- write **"X($P_2 : 30$)"** for a request of **record-level X lock for the 30^{th} record on the second page of the PlayIn table**
- write **"S($P_2 : 30 - P_3 : 100$)"** for a request of **record-level S lock from the 30^{th} record on the second page of the PlayIn table to the 100^{th} record on the third page of the PlayIn table**.

(a) [**5 points**] Read ALL records on ALL pages in the Movies table.

Solution: IS(D), S(M)

(b) [**5 points**] Read ALL records on page M_7 through M_{21}, and modify the record $M_{10} : 10$.

Solution: IX(D), SIX(M), IX(M_{10}), X($M_{10} : 10$); also acceptable: IX(D), IX(M), S($M_7 - M_9$), S($M_{11} - M_{21}$), SIX(M_{10}), X($M_{10} : 10$)

(c) [**5 points**] Modify the first record on EACH and EVERY page of the PlayIn table (these are blind writes that do not depend on the original contents in the pages).

Solution: IX(D), X(P)

(d) [**5 points**] For EACH record in the Movies table, capitalize the English letters in the **movie_name** if it is not capitalized. That is, "The Hobbit: The Desolation of Smaug" will be modified as "THE HOBBIT: THE DESOLATION OF SMAUG" but "THE HOBBIT: AN UNEXPECTED JOURNEY" will be left unchanged.

Question 3 continues. . .

Solution: IX(D), X(M)

(e) [**5 points**] Update the **movie_rating** of EACH movie in the Movies table such that the rating of the movie becomes the sum of the performance ("actor-rating") of all the actors/actresses played in the movie. More specific, we use the following formula:

$$\text{movie\_rating for mid M} = \sum_{rating \in \{\langle r \rangle | \exists m,n,r(\langle m,n,r \rangle \in PlayeIn \;\wedge\; m=M)\}} rating$$

Solution: SIX(D), S(P), X(M)

(f) [**5 points**] Delete ALL the records from ALL tables.

Solution: X(D)

Question 4: B+ tree Locking . . . . [20 points]

Submit on separate page
Course: 15-415/615; HW: ; Q:
Name: ______________; andrew-id: ______________; late days:

Consider the following B+ tree:

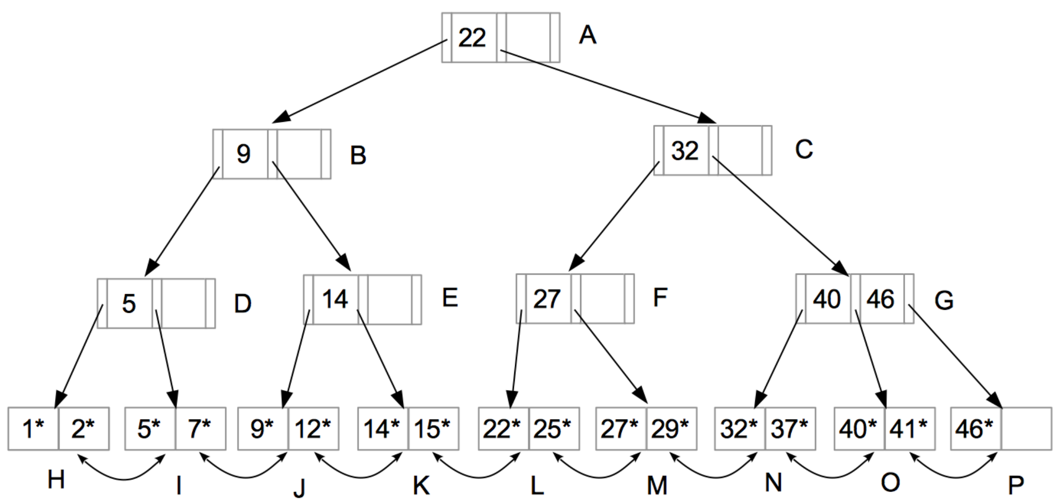

Figure 1: B+ tree locking

To lock this B+ tree, we would like to use the **Bayer-Schkolnick** algorithm (described in lecture notes #22[1], slide 31 - 34). **Important**: we use the version as presented in the lecture, which **does not** use lock upgrade.
For each of the following transactions, give the sequence of lock/unlock requests. For example, please write $S(A)$ for a request of shared lock on node A, $X(B)$ for a request of exclusive lock on node B and $U(C)$ for a request of unlock node C.
Important notes:

- Each of the following transactions is applied on the *original tree*, i.e., please ignore any change to the tree from earlier problems.
- For simplicity, *ignore* the changes on the pointers between leaves.

(a) **[5 points]** Search for data entry "22*"

Solution: S(A), S(C), U(A), S(F), U(C), S(L), U(F), U(L)

Fill in the lock/unlock requests in the corresponding table below (Table 4) - the first request is filled in already, to serve as example: at time $t1$, we should ask for S-lock on 'A'.

(b) **[5 points]** Delete data entry "1*" (Use Table 5)

[1] http://www.cs.cmu.edu/~christos/courses/dbms.S14/slides/22CC2.pdf

time	t1	t2	....
A	S		
C			
F			
L			

Table 4: Template for question (a)

Solution: S(A), S(B), U(A), S(D), U(B), X(H), *note that the greedy algorithm wins because we don't need to merge on deletion.*
U(D), U(H)
Final answer: S(A), S(B), U(A), S(D), U(B), X(H), U(D), U(H)

time	t1	t2	....
A			
.			
.			
.			

Table 5: Template for question (b)

(c) [**5 points**] Insert data entry "33*" (Use Table 6)

Solution: S(A), S(C), U(A), S(G), U(C), X(N), *note that leaf is not safe because we need to split it,*
U(N), U(G), *we need to restart*
X(A), X(C), U(A), X(G), X(N), *note that we need to lock C because G is full*
U(N), U(G), U(C)
Final answer: S(A), S(C), U(A), S(G), U(C), X(N), U(N), U(G), X(A), X(C), U(A), X(G), X(N), U(N), U(G), U(C)

time	t1	t2	....
A			
.			
.			
.			

Table 6: Template for question (c)

(d) [**5 points**] Insert data entry "101*" (Use Table 7)

Solution: S(A), S(C), U(A), S(G), *note that we cannot unlock C here because G is full, meaning that it is not safe,*

X(P), U(C), U(G), U(P), *we can unlock G and C after we lock P because we know G is safe at this point*
Final answer: S(A), S(C), U(A), S(G), X(P), U(C), U(G), U(P)

time	t1	t2	....
A			
.			
.			
.			

Table 7: Template for question (d)