CARNEGIE MELLON UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
15-415/615 - DATABASE APPLICATIONS
C. FALOUTSOS & A. PAVLO, FALL 2015

Homework 8 (by Dana Van Aken) - Solutions

Due: **hard copy, in class** at **3:00pm, on Monday, Dec. 7**

VERY IMPORTANT: Deposit **hard copy** of your answers, in class. For ease of grading, please

1. **Separate** your answers, on different page(s) for each question (staple additional pages, if needed).
2. **Type** the full info on **each** page: your **name**, **Andrew ID**, **course#**, **Homework#**, **Question#** on each of the 4 pages.

Reminders:

- *Plagiarism*: Homework is to be completed *individually*.
- *Typeset* all of your answers whenever possible. Illegible handwriting may get zero points, at the discretion of the graders.
- *Late homeworks*: in that case, please email it
  - 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: *approx. 6 hours* - 1 to 2 hours per question

Revision : 2015/12/15 09:36

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

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

GRADED BY: Anna Etzel

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

(a) Yes/No questions:

i. [**2 points**] Schedules under 2PL could have cascading aborts.
■ **Yes** □ No

ii. [**2 points**] In the shrinking phase of Strict 2PL, locks cannot be released until the end of the transaction.
■ **Yes** □ No

iii. [**2 points**] If a schedule is conflict serializable, then it is also view serializable.
■ **Yes** □ No

iv. [**2 points**] Schedules under both 2PL and Strict 2PL may lead to deadlocks.
■ **Yes** □ No

v. [**2 points**] Every serializable schedule is conflict serializable.
□ Yes ■ **No**

Grading info: -2 for each incorrect answer

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

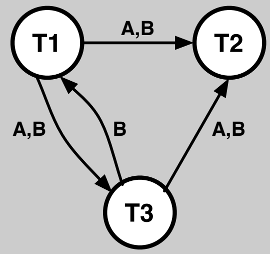
time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1	R(A)	W(A)				R(B)		W(B)		
T_2					R(A)					R(B)
T_3			R(A)	W(A)			R(B)		W(B)	

Table 1: A schedule with 3 transactions

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

Grading info: -1 for incorrect answer

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



Solution:

<u>*Grading info:*</u> *No partial credit awarded unless A, B marked, -1 for each missing/incorrect edge otherwise, -1 for mislabeled nodes in an otherwise correct graph*

iii. [**1 point**] Is this schedule conflict serializable?

☐ Yes ■ **No**

<u>*Grading info:*</u> *-1 for incorrect answer*

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 ($T_1 \rightarrow T_3 \rightarrow T_1$) in the dependency graph.

<u>*Grading info:*</u> *-3 for a justification that does not agree with previous part*

v. [**2 points**] Could this schedule have been produced by 2PL?

☐ Yes ■ **No**

<u>*Grading info:*</u> *-1 for incorrect answer*

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

GRADED BY: Dana Van Aken

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

(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 stands for 'lock manager'.

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

Table 2: Lock requests of 3 transactions

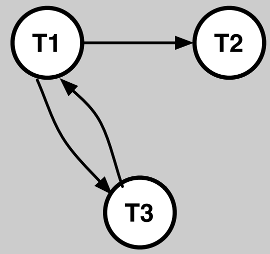
i. **[6 points]** For the lock requests in Table 2, determine which lock will be granted or blocked by the lock manager. Please write '*g*' in the LM row to indicate the lock is granted and '*b*' to indicate the lock is blocked. For example, in the table, the first lock (S(D) at time t_1) is marked as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

Solution:

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

Grading info: Half points for one mistake in the schedule, no points >1 mistake.

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



Solution:
Grading info: Half points for 1 missing directed edge, no points if missing > 1.

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

Solution: Deadlock exists because there is a cycle in the dependency graph.

Grading info: −2 points for not explaining why there is no deadlock

(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, T_3, and T_4 represent four transactions.
- LM represents a 'lock manager'.

time	t_1	t_2	t_3	t_4	t_5	t_6
T_1		X(B)				X(C)
T_2					S(C)	
T_3	S(A)		S(B)			
T_4				X(A)		
LM	g					

Table 3: Lock requests of 4 transactions

i. **[6 points]** For the lock requests in Table 3, determine which lock request will be granted, blocked or aborted by the lock manager (LM), if it 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:

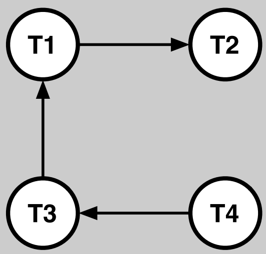
- X(B) at t_2: g
- S(B) at t_3: b
- S(C) at t_4: b

- X(C) at t_5: g
- X(D) at t_6: b

Grading info: Half points for one mistake in the schedule, no points > 1 mistake.

ii. [**5 points**] Give the wait-for graph for the lock requests in Table 3. Determine whether there exists a deadlock in the lock requests in Table 3 under LM, and briefly explain why.

Solution: No deadlock exists because there is not a cycle in the dependency



graph.

Grading info: −2 points for not explaining why there is no deadlock
−3 points for not stating whether deadlock exists

iii. [**3 points**] To prevent deadlock, we use the lock manager (LM) that adopts the Wait-Die policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. *Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a').* Follow the same format as the previous question.

Solution:

- X(B) at t_2: g
- S(B) at t_3: a
- S(C) at t_4: g (there are no locks on A since T_3 was aborted at t_3)
- X(C) at t_5: g
- X(D) at t_6: b

Grading info: −2 points for one mistake in the schedule, no points > 1 mistake.

iv. [**3 points**] Now we use the lock manager (LM) that adopts the Wound-Wait policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. *Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a').* Follow the same format as the previous question.

Solution:

- X(B) at t_2: g

- S(B) at t_3: b
- S(C) at t_4: b
- X(C) at t_5: g
- X(D) at t_6: g (T_2 aborts)

Grading info: -2 *points for one mistake in the schedule, no points* > 1 *mistake.*

Question 3: Hierarchical Locking - A Blogging Website[30 points]

GRADED BY: Yujing Zhang

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Consider a Database (D) consisting of two tables, Users (U) and Posts (P). Specifically,

- Users(uid, first_name, last_name), spans 300 pages, namely U_1 to U_{300}
- Posts(pid, uid, title, body), spans 600 pages, namely P_1 to P_{600}

Further, **each page contains 100 records**, and we use the notation $U_3 : 20$ to represent the 20^{th} record on the third page of the Users table. Similarly, $P_5 : 10$ represents the 10^{th} record on the fifth page of the Posts 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 (U, P)*, (3) *page-level ($U_1 - U_{300}$, $P_1 - P_{600}$)*, (4) *record-level ($U_1 : 1 - U_{300} : 100$, $P_1 : 1 - P_{600} : 100$)*.
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 below:

- 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 Posts 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 Posts table to the 100^{th} record on the third page of the Posts table**.

(a) [**6 points**] Read ALL records on ALL pages in the Users table.

> **Solution:** IS(D), S(U)
>
> *Grading info:* -2 *for each missing/incorrect mistake*

(b) [**6 points**] Read ALL records on page U_{15} through U_{35}, and modify the record $U_{20} : 10$.

> **Solution:** IX(D), SIX(U), IX(U_{20}), X($U_{20} : 10$);
> also acceptable: IX(D), IX(U), S($U_{15} - U_{19}$), S($U_{21} - U_{35}$), SIX(U_{20}), X($U_{20} : 10$)
>
> *Grading info:* -2 *for each missing/incorrect mistake*

(c) [**6 points**] Modify the first record on EACH and EVERY page of the Posts table (these are blind writes that do not depend on the original contents in the pages).

Solution: IX(D), X(P) also acceptable: IX(D), IX(P), IX($P_1 - P_{600}$), X($P_1 : 1 - P_{600} : 1$)

Grading info: -2 *for each missing/incorrect mistake*

(d) **[6 points]** For EACH record in the Posts table, capitalize the English letters in the **title** if it is not capitalized. That is, "My favorite database!" will be modified as "MY FAVORITE DATABASE!" but "CHECK THIS OUT" will be left unchanged.

Solution: IX(D), X(P);
also acceptable: IX(D), SIX(P), then request record-level X locks on any records that need to be updated.

Grading info: -2 *for each missing/incorrect mistake*

(e) **[6 points]** Delete ALL the records from ALL tables.

Solution: X(D)

Grading info: -2 *for each missing/incorrect mistake*

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

GRADED BY: Dana Van Aken

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Consider the following B+ tree:

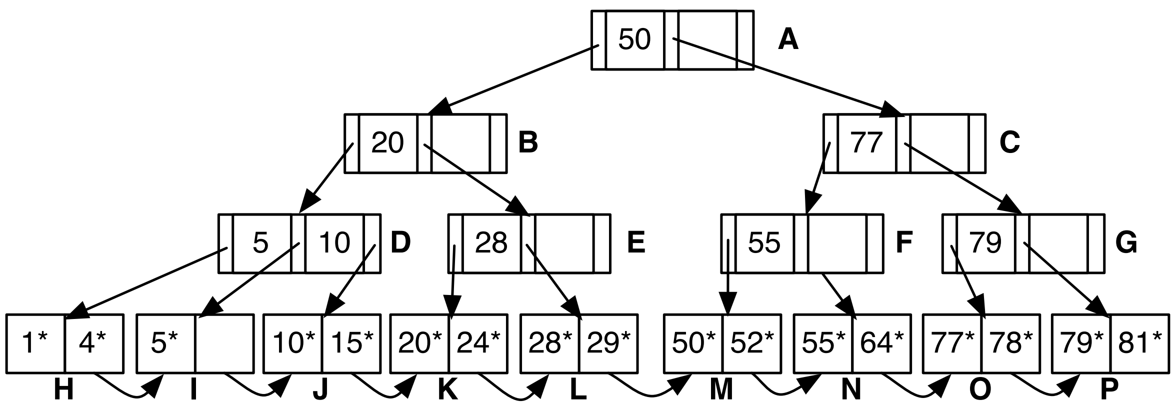


Figure 1: B+ tree locking

To lock this B+ tree, we would like to use the **Bayer-Schkolnick** algorithm (described in lecture notes #22[1], slide 91 - 94). **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.

Solution:

Grading info: -2 *for each missing/incorrect mistake*

(a) [**5 points**] Search for data entry "50*"

[1] http://www.cs.cmu.edu/~christos/courses/dbms.F15/slides/22CC1.pdf

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

(b) [**5 points**] Delete data entry "64*"

Solution: S(A), S(C), U(A), S(F), U(C), X(N), U(F), U(N)

(c) [**5 points**] Insert data entry "7*"

Solution: S(A), S(B), U(A), S(D), U(B), X(I), U(D), U(I)
Also acceptable: S(A), S(B), U(A), S(D), X(I), U(B), U(D), U(I)

(d) [**5 points**] Insert data entry "30*"

Solution: S(A), S(B), U(A), S(E), U(B), X(L), *This leaf is not safe because we need to split. We must restart*, U(E), U(L)
X(A), X(B), U(A), X(E), U(B), X(L), U(E), U(L)
Final answer: S(A), S(B), U(A), S(E), U(B), X(L), U(E), U(L), X(A), X(B), U(A), X(E), U(B), X(L), U(E), U(L)

Grading info: No points deducted for swapping U(E) and U(L). We cannot release X(E) and X(L) until after the insertion (since we need to split) so we end up releasing both locks at roughly the same time.