Tutorial 4

Assignment 1

Remember that the deadline for assignment 1 is Friday at 5pm (i.e. the 29th)!!!

See https://liverpool.instructure.com/courses/46572/discussion_topics/264560

for common issues students are running into

This week's topic

We looked at locks and logs

More precisely, we saw 2PL, simple and more advanced locks and 3 logging schemes: Undo, Redo, Undo/redo and when and how they could be used

2PL: all locks are before every unlock in each transaction

Shared and exclusive locks: Shared is only for read and exclusive can do everything

Shared, exclusive and upgrade locks: Shared and upgrade are only for read, exclusive for everything, but you can't get an exclusive lock on an item if you have shared lock for it

Intention locks: Locks on different levels (like table, block and row level)

Undo is used to directly ensure atomicity (and if some requirements are satisfied also durability)

Redo is used to directly ensure durability (and if some requirements are satisfied also atomicity)

Undo/redo is used to ensure both

Question 1

Consider the following two transactions:

T_1: $read(X)$; $read(Y)$; $Y := X + 2Y$; $write(Y)$;

and

T_2: $read(Y)$; $read(X)$; $X := Y + 2X$; $write(X)$;

There are multiple ways to add in shared lock, exclusive lock and unlock operations to the transactions so that they obey 2PL. Can you find a way to add in locks such that some schedules on them lead to a deadlock and another way so that no schedule on those leads to a deadlock?

Question 2

(Exercise 17.2.4/17.2.5 in Database Systems: The complete book). The following is a sequence of undo-log records

written by two transactions, T_1 and T_2:

<START T_1>

< T_1,X,10>

<START T_2>

< T_2,Y,20>

< T_1,Z,30>

< T_2,U,40>

<COMMIT T_2>

< T_1,V,50>

<COMMIT T_1 >

a) Describe the action of the recovery manager, including changes to both the database and the log on disk, if there is a system failure and the last log record to appear on disk is:

   i. <START T_2>
   ii. <COMMIT T_2>
   iii. < T_1,V,50>
   iv. <COMMIT T_1>

b) For each of the situations i.–iv. in a), describe what values written by T_1 and T_2 must appear on disk? Which values might appear on disk?

Question 3

(Exercise 19.1.2/19.1.3 in Database Systems: the complete book). Consider the following schedules:

S_1: $r_1(X)$; $r_2(Y)$; $w_1(Y)$; $w_2(Z)$; $r_3(Y)$; $r_3(Z)$; $w_3(U)$

S_2: $r_1(X)$; $w_1(Y)$; $r_2(Y)$; $w_2(Z)$; $r_3(Z)$; $w_3(U)$

S_3: $r_2(X)$; $r_3(X)$; $r_1(X)$; $r_1(Y)$; $r_2(Y)$; $r_3(Y)$; $w_2(Z)$; $r_3(Z)$

S_4: $r_2(X)$; $r_3(X)$; $r_1(X)$; $w_1(Y)$; $r_3(Y)$; $w_2(Z)$; $r_3(Z)$

a) Suppose that each of the schedules is followed by an abort operation for transaction T_1.
Tell which transactions need to be rolled back.

b) Now suppose that all three transactions commit and write their commit record on the log immediately after their last operation. However, a crash occurs, and a tail of the log was not written to disk before the crash and is therefore lost. Tell, depending on where the lost tail of the log begins:

i. What transactions could be considered uncommitted?

ii. Are any dirty reads created during the recovery process? If so, what transactions need to be rolled back?

iii. What additional dirty reads could have been created if the portion of the log lost was not a tail, but rather some portions in the middle?