CS127 Homework #5

Due: November 26th, 2019 11:59 P.M.

Warmup #1

List the ACID properties. Explain the usefulness of each.

Warmup #2

Assume there is a database with 2 objects A and B and consider two transactions $T1$ and $T2$.

Give an example an example of schedule composed by reads and writes of $T1$ and $T2$ that result in a read-write conflict.

Warmup #3

Explain why log records for transactions on the undo-list must be processed in reverse order, whereas redo is performed in a forward direction.

Problem 4 Transactions

Consider the following (incomplete) schedule S:

T1	T2	T3
R(X)		
R(Y)		
W(X)		
	R(Y)	
		W(Y)
W(X)		
	R(Y)	

1. Determine the serializability graph for this schedule. Also provide the serializability graph of the order of operations assuming that all three transaction eventually commit. (5 points)
2. For each of the following, modify S to create a complete schedule that satisfies the stated condition. If a modification is not possible, briefly explain why. If it is possible, use the smallest possible number of actions (read, write, commit, or abort). You are free to add commits and aborts to the schedule if that makes it clearer. (10 points each)
   (a) Resulting schedule avoids cascading aborts but is not recoverable
   (b) Resulting schedule is recoverable
   (c) Resulting schedule is conflict-serializable

Problem 5 Concurrency: 2PL

Consider the following two transactions:

T1:
read(A);
read(B);
if A = 0 then B := B +1;
write(B).

T2:
read(B);
read(A);
if B = 0 then A := A +1;
write(A).

Answer the following questions. (5 points each)

1. Add lock and unlock instructions to transactions T31 and T32, so that they observe the two-phase locking protocol.
2. Can the execution of these transactions result in a deadlock? If so, give a sequence of executions that results in a deadlock.

Problem 6 Recovery

Consider a database with the following initial values, and the attached command log:
$A = 41,\ B = 66,\ C = -2,\ D = 104,\ E = 23$

LOG:
$< T_0, \textbf{start} >$
$< T_1, \textbf{start} >$
$< T_1, B, 66, 102 >$
$< T_2, \textbf{start} >$
$< T_2, C, -2, 99 >$
$< T_1, B, 102, 142 >$
$< \textbf{checkpoint} : T_0, T_1, T_2 >$
$< T_0, A, 41, 100 >$
$< T_3, \textbf{start} >$
$< T_2, \textbf{commit} >$
$< T_3, D, 104, -40 >$
$< T_3, \textbf{commit} >$
$< T_4, \textbf{start} >$
$< T_4, E, 23, 24 >$

Assume immediate database modification and the system crashes before the remaining transactions can commit. Use the recovery protocol for concurrent transactions (which persists all in-memory dirty pages and transaction log entries at each checkpoint) to answer the following questions. (5 points each)

1. List any transactions that will need to be undone or redone in the recovery process.
2. List, in order, the set of logged operations to be performed to undo or redo the transactions, using the recovery protocol for concurrent transactions. (*i.e. "Set A to 7", "Set B to 39", etc.*)

3. Give the final values for A, B, C, D, and E after the recovery.

Problem 7 Putting it all together

Cool Concurrency Read: https://bit.ly/2X9EQTp

Read all sections prior to the Evaluation section. You can skip the Related Works section but it's great background information.

Answer the questions below in no more than 3 sentences each. We suggest looking through the questions first, then answer each while reading the paper - this helps you absorb the material more. Make sure you capture the core/key idea in these three sentences. These are 2 points each, 40 points total.

Some helpful reading in case you need a refresher on `fork()` and virtual memory: https://bit.ly/34Wh67Q

Warm-up Question (not graded): Describe the `fork()` system call and how it interacts with virtual memory.

1. From the introduction:
   (a) Where in the memory hierarchy does HyPer run?
   (b) Describe the intuition for how HyPer handles concurrent OLTP transactions? Why can it do so?
   (c) What is the current problem with OLTP databases like VoltDB when handling OLAP workloads?
   (d) What are HyPer's two goals?
   (e) Describe the intuition for how HyPer allows hybrid OLTP and OLAP workloads by working with the operating system?
   (f) Describe the difference between OLAP and OLTP and what is HyPer trying to do
2. OLTP:
   (a) Describe how HyPer handles concurrent OLTP transactions.
   (b) What allows HyPer to use such a simple concurrency control protocol?
3. OLAP:
   (a) Describe the intuition behind how HyPer achieves concurrency in OLAP transactions
   (b) What does fork() do and when can HyPer call it to ensure transactional consistency?
   (c) Why is HyPer able to avoid any locking or latching (latching are lightweight/short-term locks) during multiple OLAP sessions?
   (d) How does HyPer handle multiple OLAP sessions to take advantage of multi-core servers?
4. OLTP + OLAP
   (a) Describe the intuition behind how HyPer handles OLTP transactions in the presence of OLAP snapshots
   (b) When do OLTP transactions require explicit synchronization (concurrency control) and why?
5. Transaction Semantics and Recovery
   (a) What isolation level (e.g. Serializable, Repeatable Read, Snapshot Isolation, etc..) and what concurrency control protocol does HyPer use to achieve this (e.g. 2PL, Strict 2PL, OCC, etc..)?
   (b) How does HyPer make a transaction archive? This is analogous to what concept in recovery that we discussed in class?

(c) How does HyPer handle partition-crossing transactions? Why is this extremely expensive?

(d) How does HyPer use the transaction archive and a redo-log to ensure durability?

(e) Why does Hyper not need to "undo" failed transactions after a system crash (so-called R3 recovery) and when is the undo-log necessary in HyPer?

(f) How does HyPer perform recovery after a system failure?