Midterm (Winter 2013)

We have covered (a) serializability, (b) recoverability, (c) TM-DM model with cache management, and (d) properties of transaction. In addition to these topics, we have discusses some part of concurrency control mechanisms based on two-phase locking. The midterm format and test topics are given below:

There will be about 5 to 6 questions and you will answer any three (3) questions. Points will be deducted (5 points for each extra question) if you answer more than 3 questions.

Topics:

1. Database architecture (TM-DM model): You must be well-familiar with the functionality of each component
2. Types of execution: strict, cascading, recoverable, and serializable
3. Different types of database update (update in place, immediate update, deferred updates, etc.)
4. Limitations of serializability
5. Serializability theory and theorems
6. Partial and total ordering of transaction operations
7. History, schedule, and equivalence of histories
8. NO VIEW SERIALIZABILITY
9. Deadlocks and their management
10. CCM: Only two-phase locking schemes (schemes that we have discussed so far).

CS5570 Sample Questions (Ignore the topics that we have not yet covered)

**Q1**. Define “Undo rule” and “Redo rule”. What actually these rules ensure? Is it necessary that both these rules must be enforced by the recovery manager to recover the database? If your answer is YES, then explain why. If your answer is NO, then explain which one is necessary and which one is not.

**Q2**. Explain Fuzzy checkpointing. Explain how the recovery manager makes sure that all dirty cache slots are flushed when this checkpoint occurs.

**Q3**. Define and explain Thomas Write Rule (TWR). Discuss an integrated scheduler which uses basic TO scheduler for *rw* synchronization and TWR for *ww* synchronization. Show with an example that it correctly serializes transactions and is deadlock free.

**Q4.** Explain a recovery protocol of your choice and establish that it satisfies undo and redo rules.

**Q5**. Log size increases continuously and could overflow the file space if the expansion is not checked. Define a rule to check the expansion and show that you scheme, based on the rule you define, would work.

**Q6**. Suppose we alter the definition of Wait-Die as follows:

*if ts(Ti) > ts(Tj) then Ti waits else abort Ti*

Does this method prevent deadlock? Backup your claims with examples and counterexamples. Compare the dynamic behavior of this method with standard *Wait-Die* and *Wound-Wait*.

**Q7**. Suppose the locking granularity is a page which contains a set of records. A log record of an updated page requires a little over two pages (BFIM, AFIM, log record). This uses too much space. What can you do to reduce the log space requirement? Explain that your mechanism uses much less space and improves roll-back operation.

**Q8**. Do you think that phantom problem only occurs in record level locking (locking granularity-a record of a data page) or it can also occur in a page level (locking granularity-an entire page) locking? Explain your answer.

Q9. Answer the following questions. Provide a simple example to clarify your point.

1. Complete the sentence: The following *problems will occur if some transactions do not follows two-phase locking policy* (a)…
2. Complete the sentence: *An action is atomic if and only if ---*
3. Complete the sentence: *An execution is strict if ---*
4. Complete the sentence: *A history is recoverable if –*

Q10. Consider the following histories. What type (RC, ACA, etc.) of history each one is? Explain your answer.

**H1 = w1(y) w1(x) r2(y) w2(x) r2(z) w2(z) c2 w1(z) c1**

**H2 = w1(x) w1(z) r2(u) w2(x) r2(z) w2(z) w1(b) c1 c2**

**H3 = w1(x) w1(z) r2(u) w1(b) c1 w2(x) r2(z) w2(z) c2**

**H4 = w1(x) w1(z) r2(u) w2(x) w1(b) c1 r2(z) w2(z) c2**

Q11. In a concurrent execution under a two-phase locking policy, in the event of a conflict if the requestor or the holder is immediately rolled-back (no blocking) then deadlock cannot occur. First show with an example and then provide a proof.