Written Exam
Database Transactions and Processes
course code: 211098

10 April 2008 (09:00 - 12:30) - LA 3520

Remarks:

- Motivate yours answers. The motivation / argumentation plays an important role in grading the assignments.
- You may not consult books or notes, but only one page of A4 size, double-sided printed. The page may contain text (typed or hand-written) and (possibly reduced) images (copied from the book, other sources or hand-made).
- For each assignment, the number of points is given. They add up to 90. You get 10 points for showing up at the exam. The grade for the exam is determined by dividing the number of points by 10.
- There is also a practical assignment. The final grade for the course is determined by taking twice the grade of the exam and once the grade of the practical assignment.

Assignment 1 (26 points)

Given this stream of operations:

$$r_1(x)r_2(x)w_1(x)r_3(y)r_3(x)c_1w_3(y)r_2(y)c_3c_2$$

(a) Draw the serialization graph.
(b) Is this a serializable schedule? Explain your answer.
(c) If there were no locking used and the schedule would execute as above, which dirty reads and/or dirty writes would take place?
(d) Under an immediate-update pessimistic concurrency control, what schedule would emerge for the above stream of operations? Explain your answer.
(e) Under a deferred-update optimistic concurrency control, are all commits allowed in the above stream of operations? Explain your answer.

Assignment 2 (8 points)

Isolation level	**Read lock implementation**
READ UNCOMMITTED	No read locks
READ COMMITTED	Short duration read locks on rows returned by one SELECT query
REPEATABLE READ	Long duration read locks on rows returned by one SELECT query
SERIALIZABLE	Long duration read lock on predicate in WHERE clause

(a) On which isolation levels are dirty reads possible? Explain your answer based on the consequences of the read lock implementation.
(b) On which isolation levels are repeatable reads possible? Explain your answer based on the consequences of the read lock implementation.
(c) On which isolation levels are phantoms possible? Explain your answer based on the consequences of the read lock implementation.

Assignment 3 (18 points)

Suppose one single distributed transaction is running using a 2-Phase Commit protocol involving two cohorts A and B under the supervision of coordinator C. The transaction respects all integrity constraints, so both cohorts would commit if nothing bad happens. Unfortunately, a maintenance person accidentally trips over the network cable of server A. The network cable is disconnected, but he immediately plugs it back into the server. Consequently, the network connection between server A and coordinator C has been interrupted for a short while, but just long enough for some time-outs to occur.

(a) Give all possible time-outs that can occur in the 2-Phase Commit protocol in this situation.

(b) Is it possible that the transaction still globally commits? If so, under which conditions would the transaction still globally commit?

(c) Is it possible that the transaction globally aborts? If so, under which conditions would the transaction globally abort?

Assignment 4 (5 points)

An *update lock* is defined to conflict with other update locks and with write locks, but not with read locks. This gives the following grant table:

Requested mode	Granted mode		
	read	write	update
read		X	
write	X	X	X
update		X	X

(a) What anomaly do update locks prevent? Explain your answer.

(b) For which isolation level are update locks important?

Assignment 5 (33 points)

Suppose the database crashed and upon restart we find the following situation. The database uses a forced commit policy.

Log:

	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	T_1	T_1	T_1	T_2	T_1	T_1 T_2	T_2	T_3	T_4	T_4	T_1	T_3	T_2	T_4
	B	U	U	B	U	CK	U	B	B	U	C	U	A	U
		x	y		x		z			p		x		q
		4 5	A B		5 6		0 2			R S		6 7		0 1

Each record from top to bottom:
LSN, transaction, type, variable, before and after image
Type: B:begin transaction, U:update, C:commit, A:abort, CK:sharp check point.

Database pages:

Page 34	Page 35
LSN:20	LSN:22
x=7	p=T
y=B	q=1
z=0	

(a) Describe the *write ahead feature* (also called write ahead policy or write ahead property).

(b) Which transactions were active at the time of the crash? What part of the log needs to be examined for establishing this?

(c) Page 35 has LSN 22, whereas the log only goes upto LSN 21. Explain what apparently happened before the crash that resulted in this situation.

(d) The recovery protocol reconstructs a consistent database state. Give the reconstructed database state (i.e., values of the variables in pages 34 and 35). Explain your answer.

(e) Imagine we would run the same operations on the database and let it crash at exactly the same moment, but now using a *no force commit policy*. Which log records would you necessarily find on disk and what would be the oldest possible contents of the database (i.e., the oldest possible values of the LSN and variables in pages 34 and 35).