Written Exam
Database Transactions and Processes

course code: 211098

16 April 2009 (09:00 - 12:30) - SP 4

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: Recovery (31 points)

Suppose the database crashed and upon restart we find the following situation. The database uses a *no-force commit policy* and pessimistic concurrency control.

	8	9	10	11	12	13	14	15	16	17
	T_1	$T_1; T_3$	T_2	T_3	T_1	T_4	T_2	T_3	T_3	T_2
Log: ···	U	CK	B	U	C	B	U	U	C	U
	x			y			x	z		y
	0 5			0 2			5 6	0 4		2 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 22
LSN:9
x=5
y=0
	Page 25
	LSN:18
	z=1

(a) Describe as precisely as possible
- what is *not* being forced in a no-force commit policy compared to a force commit policy,
- when the log is being forced in a no-force commit policy, and
- when the database pages are being forced in a no-force commit policy.

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

(c) The recovery protocol reconstructs a consistent database state. Present the steps of the protocol for optaining a consistent database state in this situation. Also give the reconstructed database state (i.e., values of the variables in pages 22 and 25). Explain your answer.

(d) Although T_3 changed the value of y from 0 to 2 (LSN 11) and T_3 subsequently committed (LSN 16), the database contents still show that $y = 0$ (Page 22). This looks like a violation of the 'D' of Durability in ACID. Explain how this is possible and why durability is still guaranteed.

(e) Apparently 4 transactions were executing concurrently on the database updating variables x, y, and z. Under a pessimistic concurrency control, transactions may have had to wait for other transactions releasing their locks. Given the activity you see in the log, which transactions are likely to have had to wait for which other transactions and for which locks? Explain your answer.

Assignment 2: Serializability (12 points)

Given this stream of operations:

$$w_1(x)\ w_2(x)\ w_3(y)\ c_1\ w_2(y)\ w_3(z)\ c_3\ w_4(z)\ c_4\ c_2$$

(a) Is this schedule serializable? Explain your answer.

(b) If this stream of operations is fed to a pessimistic concurrency control, in what order are the operations ultimately executed? Explain your answer.

Assignment 3: 2-Phase Commit (16 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 updates, deletes and inserts produced by the transaction respect all integrity constraints, so both cohorts would commit if nothing bad happens.

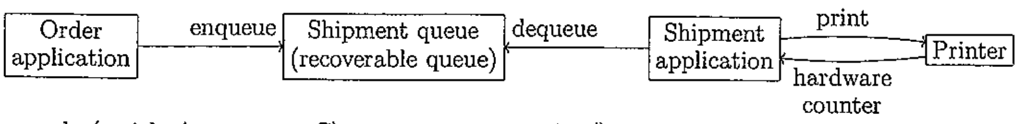
What happens is the following. After phase 1 of the 2-Phase Commit protocol, i.e., after A and B have sent their vote to commit to the coordinator, server A is the target of a 'Denial-of-Service attack (DoS)'. In other words, server A is being flooded with requests. The administrators of server A are very good: they notice the DoS attack quickly and block any incoming messages from outside the organization. The effect is that, although A is temporarily very busy with handling all received outside requests, it does not 'drown' or 'crash', so eventually it will react to the next message of the coordinator. Obviously, this takes some time so the coordinator has a time-out.

(a) Explain what the 2-Phase Commit protocol prescribes regarding what course of action the coordinator needs to take for this time-out *and* what the outcome is: global abort or global commit.

(b) What would have happened if the administrators would have blocked all messages to server A including the messages from the coordinator?

(c) Is it possible for a DoS attack on one server in a distributed transaction to affect other servers participating in the same distributed transaction? Did the DoS attack affect server B in the abovedescribed scenario? Explain your answers.

Assignment 4: Recoverable queues (13 points)



```
order(article A, customer C)
begin transaction
  update stock level of A in Stock table;
  insert order(A,C) in Order table;
  enqueue shipment request for O on queue;
  'some other statements';
commit
```

```
ship()
begin transaction
  dequeue shipment request for an order O from queue;
  HW1 := print shipment labels for O;
  record HW1 in the database;
  HW2 := print invoice for O;
  record HW2 in the database;
commit
```

Above an architecture is depicted for ordering and shipment. An Order application runs the 'order' transaction to mutate the Stock and Order tables and enqueue a shipment request for the order on a recoverable queue. The Shipment application runs a 'ship' transaction to dequeue a shipment request from the recoverable queue and print shipment labels and an invoice. If an employee sees a set of shipment labels or an invoice at the printer, he/she sends out the article or invoice. The printer has a hardware counter to be able to detect whether or not the real-world event of printing has happened or not in case of a crash of the Shipment application. Obviously, they do not want to ship an article twice or send an invoice twice.

(a) After a crash of the shipment application and recovery of the database, is it possible that the hardware counter of the printer is the same as recorded for HW1 in the database? If so, under which conditions can this occur and what action needs to be taken? If not, why not?

(b) After a crash of the shipment application and recovery of the database, is it possible that the hardware counter of the printer is the same as recorded for HW2 in the database? If so, under which conditions can this occur and what action needs to be taken? If not, why not?

(c) After a crash of the shipment application and recovery of the database, is it possible that the hardware counter of the printer has a different value than recorded for HW1 or HW2 in the database? If so, under which conditions can this occur, what different values could it have, and what action needs to be taken? If not, why not?

(d) Suppose the order applications crashes during 'some other statements', i.e., the shipment request was placed on the queue but the 'order' transaction did not commit. Is it possible that the Shipment application in the mean-time has dequeued the shipment request and printed the shipment labels and invoice? If so, under which conditions can this occur? If not, why not?

Assignment 5: Phantoms (18 points)

(a) Describe as precisely as possible what a *phantom* is.
(b) Which SQL-statements can possibly cause a phantom. Explain how.
(c) What kind of locking is especially designed to prevent phantoms? Explain how.
(d) Explain why this kind of locking is so hard to implement.