Written Exam
Database Transactions and Processes

course code: 192110982

17 April 2013 (13:45 - 17:15), CR-2L

Maurice van Keulen

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.

1 Recovery (20 points)

ck = write Page

Consider a DBMS using sharp checkpointing, a pessimistic immediate-update concurrency control, and a no-force commit policy. Suppose the database crashed and upon restart we find the following situation.

no after images in log

Log:	…	6	7	8	9	10	11	12
		T_1	T_2	$T_{1,2}$	T_3	T_3	T_2	T_1
		B	U	CK	B	U	C	C
			x			y		
			0 1			6 5		
		↑						↑

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

Database pages:	Page 54	Page 87
	LSN:14	LSN:8
	x=???	y=???

(a) The information above doesn't give the values of x and y on the database pages 54 and 87.

2 i) Can it be inferred what value of x one would find on disk in this situation? If so, which value?

ii) Can it be inferred what value of y one would find on disk in this situation? If so, which value?

Explain your answer.

(b) The recovery protocol reconstructs a consistent database state. Present the steps of the protocol for obtaining a consistent database state in this situation. Also give the reconstructed database state (i.e., values of the variables x, y in pages 54 and 87). Explain your answer. 2

(c) There are no other records in the log for T_1 other than B and C. Apparently, T_1 is a read-only transaction. It seems superfluous to record anything in the log for read-only transactions: even if there is a crash, there are no updates to roll back or roll forward. Give the main reason why in practice begin and end of transactions that do not update anything, are recorded in the log. 2

2 2-Phase Commit and Persistent Queues (20 points)

Imagine a small on-line bookstore. We focus in this question on placing orders and shipping orders. Customers search/browse the book catalogue on a website. Obviously, the web application has a button "Order". This places the order for one or more books (transaction T_1 in Figure 1(b)). The company has a warehouse with books. Some time after the order has been placed, the warehouse 'processes' the order: the books associated with the order will be shipped to the customer, the inventory administration will be updated, and the status of the order will be changed from 'ordered' to 'shipped' (transaction T_2 in Figure 1(b)). This functionality is realized with an architecture with 5 servers (see Figure 1(a)).

S_1	Webserver running website and all application logic for ordering and shipping.
S_2	Persistent queue server.
S_3	Order DB.
S_4	Warehouse DB (inventory).
S_5	Transaction manager coordinating distributed transactions.

(a) Servers

T_1 **New order**:
    Insert order into S_3;
    Enqueue 'order shipment' request in S_2;
    COMMIT

T_2 **Ship order**:
    Dequeue 'order shipment' request from S_2;
    Update inventory level for the ordered books in S_4;
    Update order status in S_3 to 'shipped';
    COMMIT

(b) Transactions

Figure 1: Involved servers and transactions

(a) The 'COMMIT' of both transactions is executed with the 2-Phase Commit protocol coordinated by S_5.

  i) Which servers are the *cohorts* for T_1?

  ii) Which servers are the *cohorts* for T_2?

Explain you answers.

(b) There is an integrity constraint in the Warehouse DB stating that the inventory of a book should be positive. The constraint is checked at commit time. Suppose for a certain order, the inventory level would become negative, i.e., the constraint would fire causing transaction T_2 to abort.

  i) Specify exactly which messages are exchanged between coordinator and cohorts during the 2-phase commit protocol for this transaction.

  ii) What would happen after the 2-phase commit protocol finished?

Explain you answers.

(c) Suppose during the 2-phase commit protocol of T_1 for a certain order, the 'prepare to commit' message to S_3 gets lost due to some isolated network glitch. Obviously, some time passes before the time-out protocol comes into effect causing the global transaction to abort. Explain exactly what prevents T_2 to be executed for this order (preventing the aborted hence cancelled order to be shipped).

3 Serializability (10 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) Draw the serialization graph. Is it possible to infer from the serialization graph whether or not the schedule is serializable? If so, is it serializable or not?

(b) If this stream of operations is fed to a pessimistic concurrency control,
   i) in what order are the operations ultimately executed?
   ii) with which serial schedule is this execution equivalent?
   Explain your answer.

4 Anomalies (24 points)

Suppose the order DB of question 2 has two tables, 'ORDER' and 'ORDERLINE', and the 'Insert order into S_3' is realized with three insert statements (see Figure 2).

Order DB schema

Table	Attributes
ORDER	OrderID, CustomerName, Status
ORDERLINE	OrderID, BookID, Quantity, Price

SQL statements for 'Insert order into S_3'

```
INSERT INTO ORDER VALUES (4321, 'A', 'ordered');
INSERT INTO ORDERLINE VALUES (4321, 123, 1, 14.95);
INSERT INTO ORDERLINE VALUES (4321, 567, 2, 8.95);
```

Figure 2: More details on Order DB and ordering

(a) INSERT statements can cause phantoms. Construct an example of a scenario (including concrete SQL statements) in which the INSERT statements of Figure 2 cause a phantom to occur. Explain your answer.

(b) INSERT statements can also cause non-repeatable reads. Construct an example of a scenario (including concrete SQL statements) in which the INSERT statements of Figure 2 cause a non-repeatable read to occur. Explain your answer.

Isolation level	Locking implementation
READ UNCOMMITTED	No read locks
READ COMMITTED	Short-duration read locks on rows returned by SELECT
REPEATABLE READ	Long-duration read locks on rows returned by SELECT
SERIALIZABLE	Long-duration read lock on predicate specified in WHERE clause

Figure 3: Isolation levels and their locking implementation

(c) The different isolation levels are specified in terms of which anomalies they do and do not prevent. Their implementation is often realized with different ways of read locking (see Figure 3).

i) Which kind of locking prevents phantoms? Explain in your scenario of question 4(a), which SQL statement obtains this kind of lock that causes the phantom-producing INSERT statement to wait.

ii) Which kind of locking prevents non-repeatable reads? Explain in your scenario of question 4(b), which SQL statement obtains this kind of lock that causes the non-repeatable-read-producing INSERT statement to wait.

5 TRPC (16 points)

Figure 4 presents an example of TRPC where one transaction calls a procedure "p". A Remote Procedure Call (RPC) is a call to a procedure that is executed on another computer. RPC supports programming for client/server with code generators: simply program the call and the procedure itself as if they run on the same computer ('Original' in Figure 4); the code generators attach extra code, called *stubs*, that handles all communication between client and server computers. The arrows in Figure 4 indicate control flow: which line(s) of code calls which procedure.

Transactional RPC (TRPC) is an extension of RPC supporting programming transaction boundaries (tx_begin, tx_commit, tx_abort). The code generators attach even more code in the stubs to handle communication with the transaction manager (not shown).

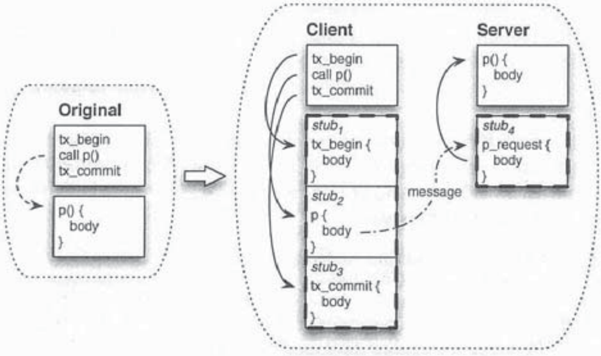

Figure 4: Stubs generated by TRPC (simplified)

(a) Which of the 4 stubs communicate with the transaction manager?

(b) For each of those stubs, what transaction-related information is communicated from client or server to the transaction manager and the other way around?

(c) Suppose the body of "p" appends a line to a particular file. Being part of a transaction which may possibly abort, this write to the file may need to be undone (i.e., rolled back). Explain how this is accomplished, or if you don't know this as a fact, make an educated guess (provide arguments that sketch how this should be accomplished).