Weekly exercises IN3020&4020 (2026)
Week 8: Transaction Management Concepts

1 Transaction Management Introduction: Review

1. What do we mean by transactions (informally) and their concurrent execution of database transactions?

   Solution: Intuitively, transactions are logical units (sequence of operators) of database processing; each transaction should be executed independent on others and in the all-or-nothing manner using its own local memory and CPU but with an ability to access common disk with a database. However, two or more transactions can be run concurrently—that is, at the same time. The actual operations of the transactions can be interleaved.

2. Is transaction management relevant for a single-user DBMS? If so, why?

   Solution: Yes, a single-user DBMS still needs to deal with fail recovery (or even concurrency control if the same user can start another transaction before the previous ends).

3. Why does a transaction need to read a block of data from the disk to local memory when executing `write`(X)?

   Solution: Access to the disk is usually only by block, and the block of data item X may contain many other data items, which may have been updated by other transactions. So, to write just X we need to read the whole block with other items, replace the value of X and write everything back.

4. Why do we need concurrency control?

   Solution: Problems (undesired behaviour) can occur when concurrent transactions execute in an uncontrolled manner.

5. What are the 4 concurrency control problems (phenomena) described in the lecture, describe and discuss each one of them.

Solution:

(a) Dirty write (lost update): Two transactions access the same database item and have their operations interleaved in a way such that the second update begins before the first is complete. This way one update is lost.

(b) Dirty read (temporary update): One transaction updates an item and then the transaction fails. The updated item is accessed before it is rolled back. Thus, the second transaction uses an incorrect value.

(c) Unrepeatable read: One transaction reads the same item twice and this item is changed in between the two reads. Hence the first transaction reads two different values for the same element.

(d) Incorrect summary (phantom phenomena): One transaction calculates an aggregate function while the other transaction are updating some of the same items. The first may use some of values before the update and other after the update. Leading to an inconsistent result.

6. Why do we need recovery management?

   Solution: During a transaction execution, something can go wrong, and the transaction aborts before committing. In this case, all changes made by the transaction should be rolled back.

7. What can cause a transaction abort?

   Solution: Some examples of what can happen: computer failure (system crash): hardware, software, network errors; transaction failure: division by zero, integrity constraint violation, user interrupt, etc.; local transaction errors: no data found, programmed exception, etc.; concurrency control enforcement: serialisability violation, deadlock resolving, etc.; disk failure: errors with disk reads or writes; physical problems: power cut, fire, catastrophe, etc.

8. Discuss the ACID properties of a database transaction, including the subsystems responsible for ensuring them.

   Solution:

   - Atomicity: A transaction is performed in its entirety or not at all (recovery and concurrency subsystems)
   - Consistency: The database should always remain consistent (no transaction subsystem ensures consistency by itself, but a transaction can issue abort command after checking consistency by itself,

and then recovery subsystem handles this command by rolling back)

- Isolation: Transaction should not interfere with other transactions (concurrency subsystem)
- Durability: Changes of committed transactions must persist (recovery)

2 Transactions, schedules, conflicts

1. Describe the structure of committed, aborted, and partial translations.

   Solution: A committed (successful) transaction is a sequence of operations with the following structure:

   1. `begin`
   2. one or several `read`(X), `write`(X), local operations (e.g., var updates), transaction control operations (e.g., locks), etc.
   3. `end`
   4. one or several operations checking consistency, serialisability, etc.
   5. `commit`

   Structure of an aborted (unsuccessful) transaction: same beginning, but ends at any point with `abort`.

   Structure of a partial transaction: a beginning of one above ('waiting' for next operation).

2. Explain what is a schedule and when it is complete.

   Solution:

   A schedule of several transactions is a (total) ordering of the operations this transaction such that:

   – operations of different transitions can interleave

   – operations of of the transactions are in the same order

   It is complete if all the transactions are committed or aborted.

3. Explain what is a conflict in the context of transaction management. What should we do to avoid such conflicts? (Note: this question is provocative)

   Solution:

   A conflict is two operations in a schedule that

T_1
read(X);
$X := X - 1$;
write(X);
read(Y);
$Y := Y + 1$;
write(Y);

T_2
read(X);
$X := X + 1$;
write(X);

Figure 1: Transactions T_1 and T_2

T_1	T_2
read(X);	
$X := X - 1$;	
	read(X);
	$X := X + 1$;
write(X);	
read(Y);	
	write(X);
$Y := Y + 1$;	
write(Y);	

T_1	T_2
read(X);	
$X := X - 1$;	
write(X);	
	read(X);
	$X := X + 1$;
	write(X);
read(Y);	

Figure 2: Schedules A and B

- belong to different transactions,
- access the same data item,
- one of them is a write.

In other words, they are operations from different transactions such that flipping them change the result.

There is no need to avoid conflicts, because there is nothing wrong with them per se. Moreover, they are unavoidable in schedules that has several transactions that access the same data items. If we really wont to avoid conflicts, we can, for example, forbid schedules with more than one transaction or allow only read-only transactions; however, both options severely restricts the essence of databases.

3 Concurrent transactions

Consider the following two transactions in Figure 1 and schedules in Figure 2.

1. Write these transactions in our abbreviated syntax. Which of these

transactions are committed, aborted, and partial?

Solution: We can write T_1 as $\mathtt{r}_1(X), \mathtt{w}_1(X), \mathtt{r}_1(Y), \mathtt{w}_1(Y)$ and T_2 as $\mathtt{r}_2(X), \mathtt{w}_2(X)$. As written, both transactions are partial, because they end neither with $\mathtt{c}_\mathtt{i}$ (commit) nor with $\mathtt{a}_\mathtt{i}$ (abort). However, it is probably the case that the commits are just omitted in both, and it just does not matter in this context that the transactions are committed.

2. Write these schedules in our abbreviated syntax. Which of these schedules are schedules of the two transactions?

   Solution: We can write Schedule A as

   $$\mathtt{r}_1(X), \mathtt{r}_2(X), \mathtt{w}_1(X), \mathtt{r}_1(Y), \mathtt{w}_2(X), \mathtt{w}_1(Y)$$

   and Schedule B as

   $$\mathtt{r}_1(X), \mathtt{w}_1(X), \mathtt{r}_2(X), \mathtt{w}_2(X), \mathtt{r}_1(Y).$$

   Schedule A is a schedule of T_1 and T_2, while schedule B is not, because it misses one operation.

3. Let the value of data item X (in the database) before a schedule starts be 10. Write the value of data item X (in the database) after each operation of Schedule A and of Schedule B.

   Solution: For Schedule A, it is the following sequence: 10, 10, 10, 10, 9, 9, 11, 11, 11. For Schedule B, it is the following sequence: 10, 10, 9, 9, 9, 10, 10. Note here that variable assignment operations are local to transactions and do not change the global value (only write operation do so).

4. Let us assume that Schedule A and of Schedule B both commit after their last operation in the figure. Which of them, you think, ensures Isolation principle?

   Solution: Schedule A does not ensure the principle, because the written value of X by T_1 is overwritten by T_2, which does not take the change by T_1 into account. So the transactions interfere in an essential way. Schedule B ensures the principle, because it overall behaviour is equivalent to executing T_1 in full and then T_2 (the interleaving of operations is not essential).

5. List all the conflicts in Schedule A with their types.

   Solution: There are 3 conflicts: $\mathtt{r}_1(X), \mathtt{w}_2(X)$ (read-write), then $\mathtt{r}_2(X), \mathtt{w}_1(X)$ (read-write), and $\mathtt{w}_1(X), \mathtt{w}_2(X)$ (write-write).

6. (*) Give examples of concurrency control phenomena discussed in the lectures and Question 1.5 written as schedules in our syntax.

Solution: Dirty write: $\mathtt{r}_1(X), \mathtt{w}_2(X), \mathtt{w}_1(X)$.

Dirty read: $\mathtt{w}_1(X), \mathtt{r}_2(X), \mathtt{a}_1$.

Non-repeatable read: $\mathtt{r}_1(X), \mathtt{w}_2(X), \mathtt{r}_1(X)$.

Incorrect summary: $\mathtt{r}_1(X), \mathtt{w}_2(Y), \mathtt{r}_1(Y)$.