Introduction to Database Systems

Exercises: Transactions (Part 1)

A transaction is an indivisible unit of work

Students

StudentID	FirstName	LastName	Graduation
2	Murvin	Drake	2025
3	Kevin	Drumm	2025
4	Claude	Shannon	1936
5	Grace	Hopper	1928
6	Hedy	Lamarr	1942

StudentsArchive

StudentID	FirstName	LastName	Graduation
1	George	Boole	1840

Let us say, that we want to move all the students who already graduated to the archive:

What would happen if someone wanted to change a row just after insertion and before deletion?

```
INSERT INTO StudentArchive (StudentID,
            FirstName, LastName, Graduation)
SELECT * FROM Students
WHERE Graduation < @Graduation
```

```
DELETE FROM Students
WHERE Graduation < @Graduation
```

A transaction is an indivisible unit of work

Students

StudentID	FirstName	LastName	Graduation
2	Murvin	Drake	2025
3	Kevin	Drumm	2025

StudentsArchive

StudentID	FirstName	LastName	Graduation
1	George	Boole	1840
4	Claude	Shannon	1936
5	Grace	Hopper	1928
6	Hedy	Lamarr	1942

```
CREATE PROCEDURE ArchiveStudents(@Graduation INT)
AS
BEGIN TRY
    BEGIN TRANSACTION
        INSERT INTO StudentArchive (StudentID,
                FirstName, LastName, Graduation)
        SELECT * FROM Students
        WHERE Graduation < @Graduation

        DELETE FROM Students
        WHERE Graduation < @Graduation
    COMMIT TRANSACTION
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION
END CATCH
```

Properties (ACID)

Atomicity
The transaction is either executed or not.

Consistency
The database has to be consistent before and after the transaction

Isolation
Concurrently executed transactions do not affect each other.

Durability
Transactions are durable upon confirmation

Types of transaction conflicts (concurrent execution of transactions)

WR – Write-Read conflict
Transaction T2 read an object that has been modified by another transaction T1 (T1 has not been committed yet)

RW – Read-Write conflict
Transaction T2 changed the value of an object that has been read by another transaction T1 (T1 has not been commited yet).

WW – Write-Write conflict
Transaction T2 overwrote the value of an object which has already been modified by another transaction T1 (T1 has not been committed yet)

Strict Two-Phase Locking

1. If a transaction T wants to read (or/and modify) an object, it first requests a shared S(A) or exclusive X(A) lock on the object.

2. All locks held by a transaction are released when the transaction is completed.

Exercises

Task 1

Transaction T1 has the following arrangement over database objects A and B: R(A), W(A), R(B), W(B)

a) Give an example of another transaction T2, that, if executed concurrently (without some form of concurrency control), could interfere with T1.

Task 1

Transaction T1 has the following arrangement over database objects A and B: R(A), W(A), R(B), W(B)

b) Explain how the use of Strict 2PL would prevent interference between the two transactions.

Task 2

Consider the following transactions:

T1: R(A) R(B) W(A)

T2: R(A) R(B) W(A) W(B)

a) Give an example of a schedule with transactions T1 and T2 that results in a WR conflict.

b) Give an example of a schedule with transactions T1 and T2 that results in a RW conflict.

c) Give an example of a schedule with transactions T1 and T2 that results in a WW conflict.

d) For each of the three schedules, demonstrate how Strict 2PL resolves these conflicts.

Task 3

The following transaction schedule is given.

```
T1  R(A) W(A)                    C
-------------------------------------------------------------------------
T2                         W(A) C
-------------------------------------------------------------------------
T3                 R(A) C
```

a) Which conflicts are present in the schedule?

b) Where do we need to place the T3 „commit“ to cause a RW conflict between T2 and T3?

Task 4

Given are two transactions:

T1: R(A), R(B), if A = 0 then B:=B+1, W(B)

T2: R(B), R(A), if B = 0 then A:=A+1, W(A)

The database consistency is guaranteed by ((A = 0) v (B = 0)) with initial values A = B = 0.

a) Show that after both possible serial executions, the database maintains consistency.

b) Provide an example where concurrent execution would result in inconsistency of the database.

c) Is there a case of concurrent implementation that would maintain consistency?

Task 5

What are the outputs of all SELECT statements?

	T1	**T2**
a.	INSERT INTO Squares VALUES (4);	
b.		SELECT * FROM Squares;
c.		INSERT INTO Squares VALUES (9);
d.		INSERT INTO Squares VALUES (16);
e.		SELECT * FROM Squares;
f.	COMMIT;	
g.		SELECT * FROM Squares;
h.	SELECT * FROM Squares;	
i.		COMMIT;
j.	SELECT * FROM Squares;	