Transaction Management

Introduction

- Transaction
  - a logical unit of work
  - a unit of program execution that access and possibly updates various data items.
  - is initiated by user program written is a high-level data-manipulation language
    - SQL, COBOL, C, C++, or Java

2

Introduction

- Transaction
  - delimited by statements such as
    - begin transaction
    - end transaction or COMMIT or ROLLBACK

3

Introduction - Sample Transaction

```
BEGIN TRANSACTION;
    UPDATE ACC 123 { BALANCE := BALANCE – 100 };
    IF (ANY ERROR) THEN
       GO TO UNDO;
    ENDIF;
    UPDATE ACC 456 { BALANCE := BALANCE + 100 };
    IF (ANY ERROR) THEN
       GO TO UNDO;
    ENDIF;
    COMMIT;
    GO TO FINISH;
UNDO:
   ROLLBACK;
FINISH:
   RETURN;
```

COMMIT signals successful end-of-transaction

ROLLBACK signals unsuccessful end-of-transaction

4

Introduction

- ACID properties of transactions:
  - Atomicity: either all operations of the transaction are implemented properly or none are.
  - Consistency: execution of a transaction in isolation preserves the consistency of the database (with no other transaction executing concurrently).

5

Introduction

- ACID properties of transactions:
  - Isolation: each transaction is unaware of other transactions executing concurrently in the system.
  - Durability: after a transaction completes successfully, the changes it has made to the database persist, even if the system fails.

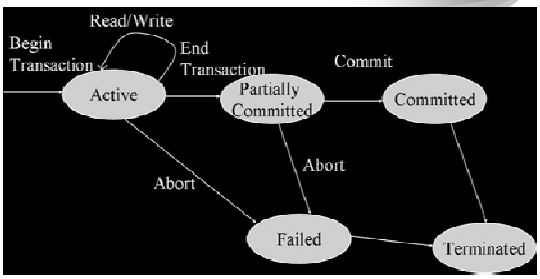
6

Transaction States

- In the absence of any failures, all transactions complete successfully.
- However, there is not always an absence of any failures.
- Thus we have different “states” in which a transaction may reside.

7

Transaction States



8

Transaction States

- Active: the initial state and one in which the transaction stays while it is executing.
- Partially committed: a state after the final statement has been executed.
- Failed: the state after the discovery that normal execution can no longer proceed.

9

Transaction States

- Aborted: the state after the transaction has been rolled back and the database restored to its condition prior to the start of the transaction.
- Committed: the state after successful execution.

10

Committed Transaction

- A transaction is considered committed when it has performed updates that transforms the database into a new consistent state.
- Once a transaction is committed, its effects cannot be undone by a system failure.
- Only way to undo a committed transaction is to execute a compensating transaction.
- However it is not always possible to create a compensating transaction.

11

Failed & Aborted Transaction

- When a transaction cannot be completed (due to some kind of system failure), the transaction must be “rolled back”.
- It also enters the aborted state where the system has two options:
  - Restart the transaction (but only if aborted due to some hardware or software error).
  - Kill the transaction which is usually done due to some internal logical error.

12

Concurrent Executions of Transactions

- Concurrent executions are good:
  - Improved throughput and resource utilization
  - Reduced waiting time
- In transaction processing multiple transactions are allowed to run concurrently.
- Updating within concurrent transactions causes several complications with consistency of the data.

13

Concurrent Executions of Transactions

- Schedules: Execution sequences
  - Represent the chronological order in which instructions are executed in the system
- Serial schedules: consists of a sequence of instructions from various transactions where the instructions belonging to one single transaction appear together in that schedule

14

Concurrent Executions of Transactions

- We are going to look at two accounts in a bank
  - A has initially $1000
  - B has initially $2000
- We have two transactions to apply to the two accounts:
  - T1: transfers $50 from account A to account B
  - T2: transfers 10% of the balance from account A to account B

15

Concurrent Execution of Transactions

- To study the correct concurrent execution, let's look at these two transactions:

```
T1: read (A);
    A:=A – 50;
    write (A);
    read (B);
    B:= B + 50;
    write (B);
```

```
T2: read (A);
    temp := A * 0.1;
    A := A - temp;
    write (A);
    read (B);
    B:= B + temp;
    write (B);
```

16

Concurrent Execution of Transactions

```
T1: read (A);
    A:=A – 50;
    write (A);
    read (B);
    B:= B + 50;
    write (B);
```

If transactions execute one at a time in order T1 followed by T2, the total amount of money in accounts A + B is preserved.

```
T2: read (A);
    temp := A * 0.1;
    A := A - temp;
    write (A);
    read (B);
    B:= B + temp;
    write (B);
```

17

Concurrent Execution of Transactions

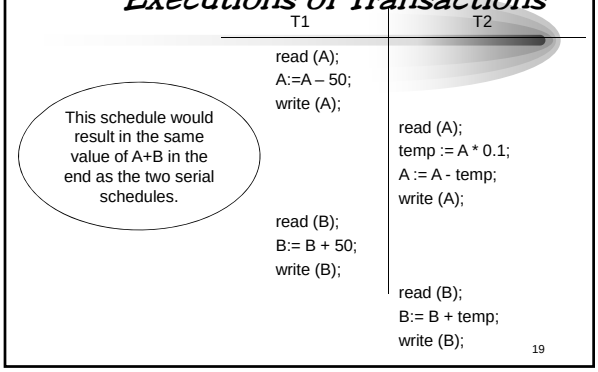
If transactions execute one at a time in order T2 followed by T1, the total amount of money in accounts A + B is preserved.

```
T2: read (A);
    temp := A * 0.1;
    A := A - temp;
    write (A);
    read (B);
    B:= B + temp;
    write (B);
```

```
T1: read (A);
    A:=A – 50;
    write (A);
    read (B);
    B:= B + 50;
    write (B);
```

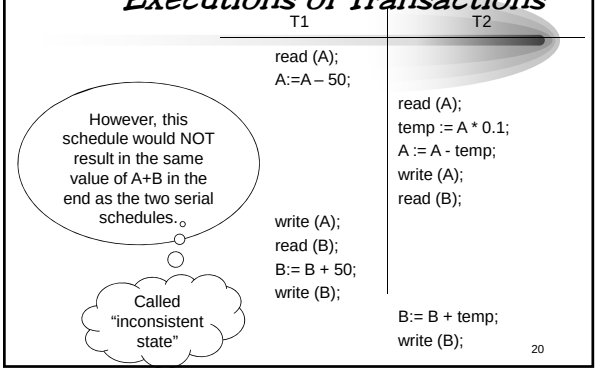
18

Concurrent Executions of Transactions



19

Concurrent Executions of Transactions



20

Concurrency Control

- Concurrency control: the task of ensuring that any schedule that gets executed will leave the database in a consistent state
- The concurrency-control component of the DBMS carries out setting up the schedules to ensure consistency during the execution of multiple transactions

21

Serializability

- Serializability: a schedule that is equivalent to a serial schedule.
- In discussing serializability, only two operations are important
  - read (Q)
  - write (Q)
- We also assume that the transaction may perform an arbitrary sequence of operations on the copy of Q between the read and write operations

22