Lecture 8. Transaction, Concurrency Control (Chapter 20 and 21)

alisa.lincke@lnu.se

Linnæus University

Outline

- Transaction and System Concepts
- Transaction Support in SQL
- Brake 10 min
- Concurrency Control Techniques

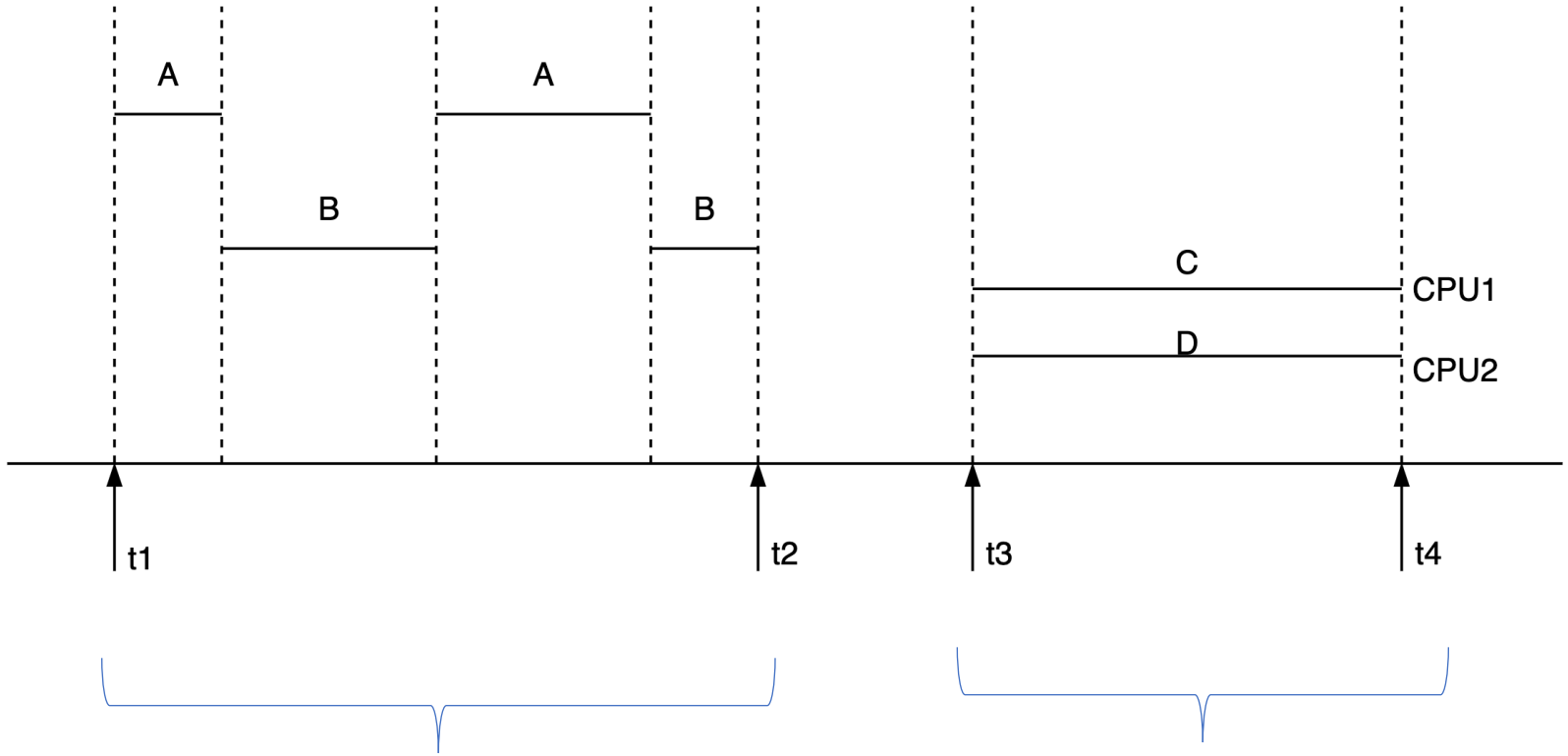
Introduction to Transaction Processing Concepts and Theory

- The concept of transaction provides a mechanism for describing logical units of database processing
- Transaction processing system are systems with large databases and hundreds of concurrent users executing database transaction.
- Examples of such systems are:
  - Airline reservation companies
  - Bank organization
  - Online retail shops
  - Supermarket checkouts
  - And many other applications

Single-User versus Multiuser Systems

- A database system can be characterizes by number of users who can use the system **concurrently**.
- A DBMS is single-user is at most one user at a time can use the system, and it is multiuser if many users can use the system and access the database concurrently.
- In multiuser systems , hundreds and thousands of users submitting transactions concurrently to the system
- Multiple users can access database simultaneously because of the concept of multiprocessing which allows the operation system to execute multiple processes at the same time.
- A single central processing unit (CPU) can only execute at most one process at a time.
- If the computer has multiple CPUs, a parallel processing of transaction are performed

Interleaved Concurrency



Transactions and other Concepts

- A **transaction** is an executing program that form a logical unit of database processing.
  - It may include one or more database access operations such as insertion, deletion, update, or retrieval operations.
  - We can explicitly specify the begging and end of transaction by using **begin and end transaction statement** in the application program. In this case, all database operations between begin and end statement considers as *one transaction.*
  - *Read-only transaction* is a transaction which contains only retrieval operations (SELECT statements only)
  - *Read-write transaction* is a transaction which contains both retrieval operations and modification operations (delete, insert, update).
  - Transactions are submitted by the various users may execute concurrently and may access and update the same database items

Possible problems with uncontrolled concurrence transaction execution (1)

- Example: Airline reservation database where each record includes the number of reserved seats on that flight
- A transaction is a particular execution of a program on a specific date, flight, and number of seats.
- T1 transfers N reservations from flight X to flight Y
- T2 reserves M seats on the flight X.

Two examples of transaction: T1, and T2

T1

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y + N;
write_item(Y);

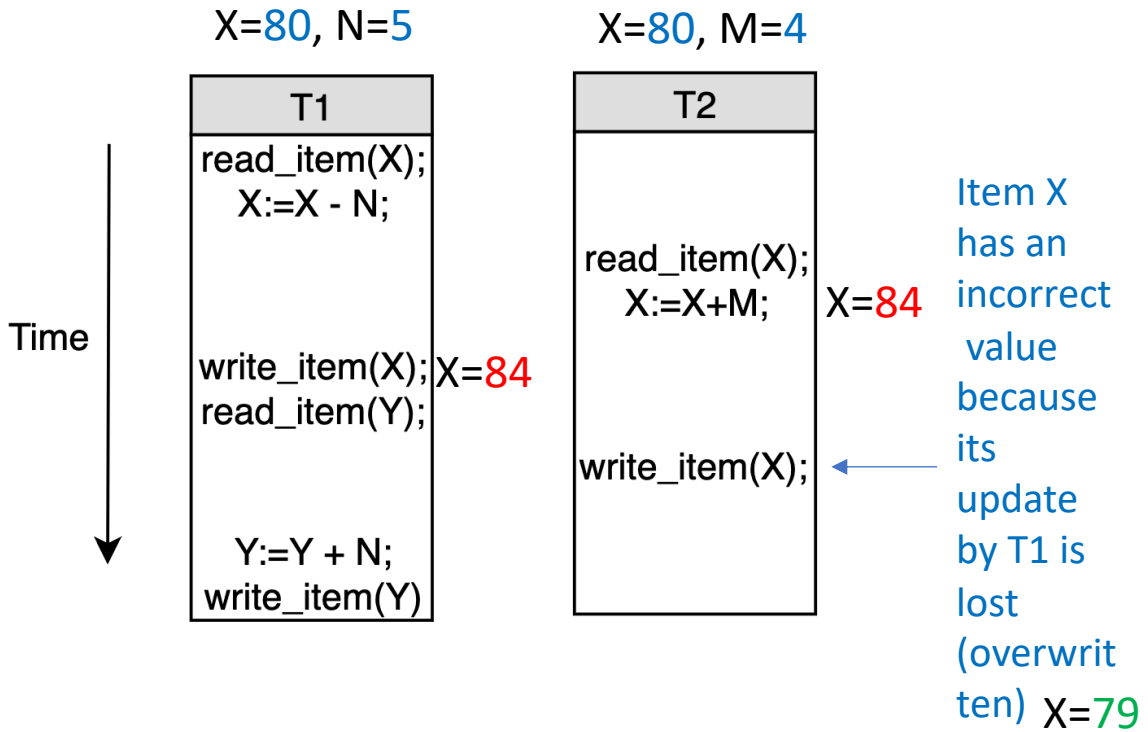
T2

read_item(X);
X:=X+M;
write_item(X);

Possible problems with uncontrolled concurrence transaction execution (2)

Problem 1: Lost Update Problem

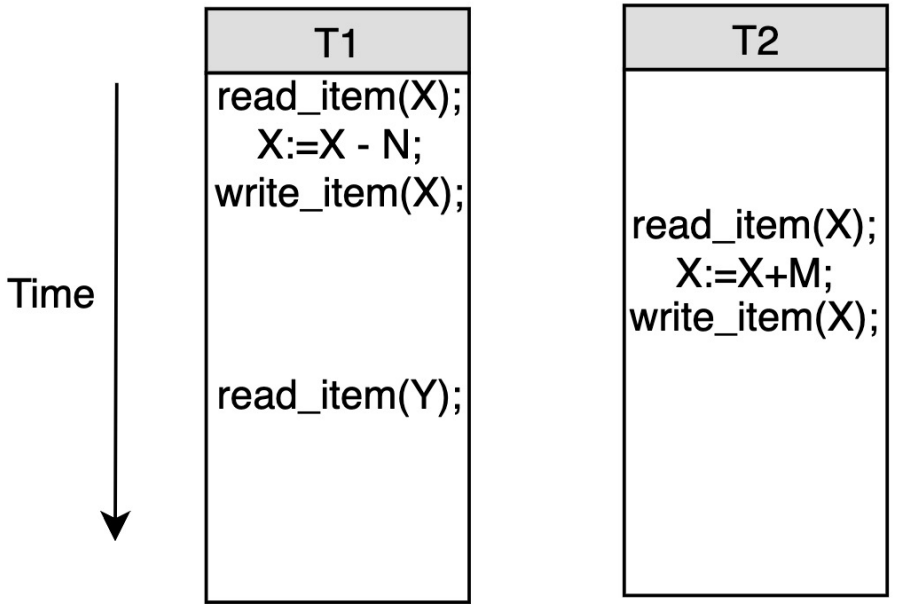
This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect.



Possible problems with uncontrolled concurrence transaction execution (3)

Problem 2 the Temporary Update

Occurs when one transaction updates a database item and then the transaction fails for some reason. Meanwhile, the updated item us accessed (read) by another transaction before it is changed back to its original value.



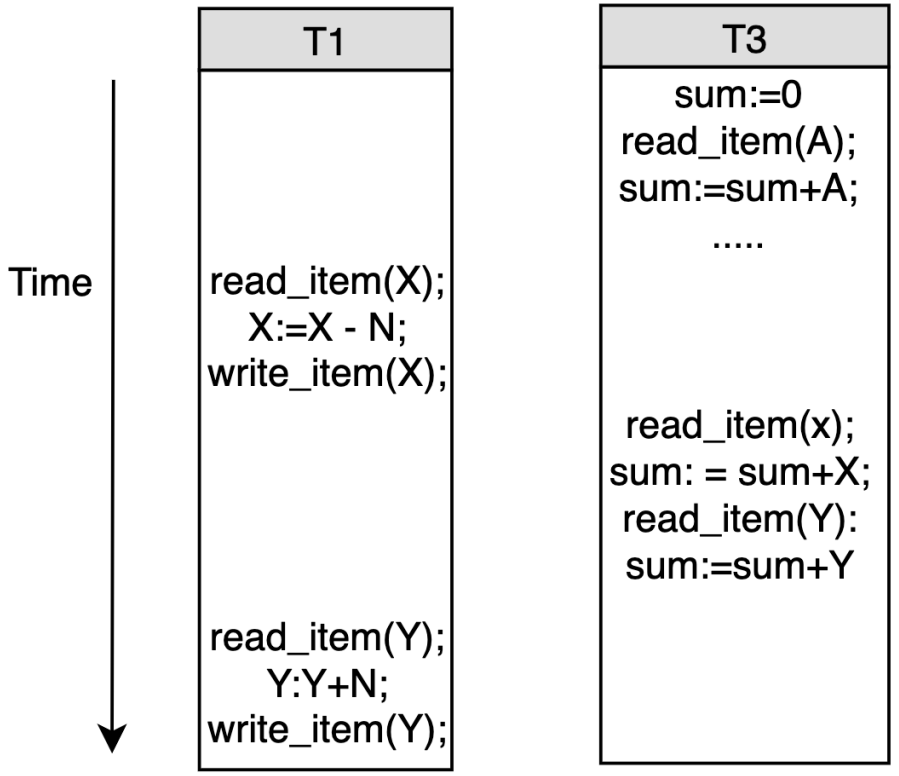
Transaction T1 fails and must
change the value of X back to its old value;
And T2 has read the temporary incorrect value of X

Possible problems with uncontrolled concurrence transaction execution (4)

Problem 3 The Incorrect Summary Problem

Occurs when one transaction is calculating an aggregated summary function (SUM, COUNT, AVG,etc.) on a number of database items while other transactions are updating some of these items, the aggregation function may calculate some values before they are updated and others after they are updated.

T3 calculates total amount of reservations on flights, meanwhile T1 is executing which is modifying number of flights in X.



T3 reads X after N is subtracted and reads Y before N is added

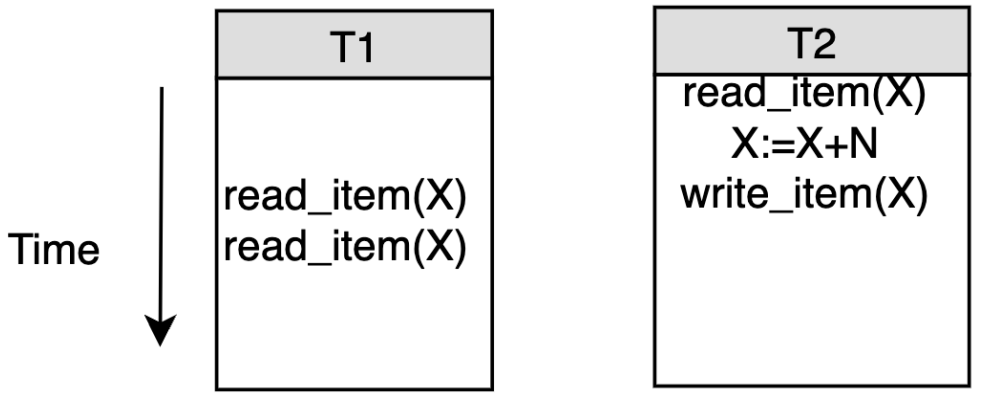
Possible problems with uncontrolled concurrence transaction execution (5)

- **Problem 4 The Unrepeatable Read Problem**

Transaction T1 reads the same item twice

Value is changed by another transaction T2 between the two reads

T receives different values for the two reads of the same item

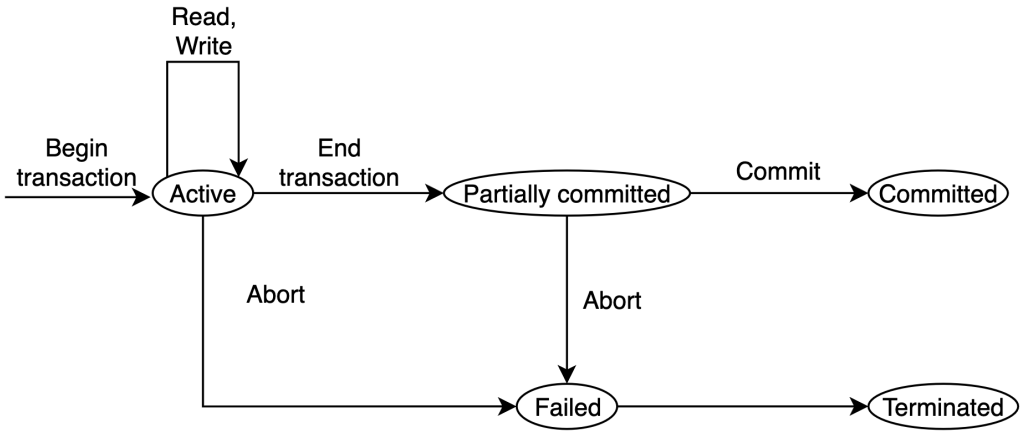


Types of Failures

- **1 A computer failure (system crash),** e.g., a hardware, software, or network error occurs in the computer system during transaction execution.
- **2 A transaction or system error.** Some transaction operation may fail such as integer overflow or division by zero, wrong parameter values (data type), or user may interrupt the transaction during its execution (e.g., pressing button Cancel)
- **3 Local errors or exception condition** detected by the transaction. For example, data for transaction may not be found.
- **4 Concurrency control enforcement**. May abort the transaction because it violates serializability or it may abort one or more transactions to resolve the state of deadlock among several transactions.
- **5 Disk failure**. A disk read/write head crash
- **6 Physical problems and catastrophes**. Endless list of problems that includes power and air conditioning failure, fire, etc.
- Most common failures are 1-4 types,
- What to do when it happened?
  - Run Database Recovery

Transaction States and Additional Operations

- Operations:
  - BEGIN_TRANSACTION
  - READ or WRITE transaction
  - END_TRANSACTION. This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution. Next step, we need to check whether the changes introduced by transaction can be permanently applied to the database (committed) or whether has to be aborted because it violates serializability, etc.
  - COMMIT_TRANSACTION is a successful signal of the end of transaction and the changes can be safely committed to the database and will not be undone.
  - ROLLBACK (or ABORT) is a unsuccessful signal of the end of transaction and any changes that were introduced by the transaction must have been undone.
- States are presented in the diagram
- All of the transaction operation and states are written into the System Log



The System Log

- System log keeps track of transaction operations
- Sequential, append-only file
- Not affected by failure (except disk or catastrophic failure)
- Log buffer
  - Main memory buffer
  - When buffer is full, appended to end of log file on disk
- Log fil is backed up periodically
- Undo and redo operations based on log are possible

Transaction Support in SQL

- Newer SQL standards have more commands for transaction processing
- The transaction processing can be written either in DBMS or application program using DB API/library
- A sample SQL transaction might look like the following:

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRNSACTION
  READ WRITE
   DIAGNOSTIC SIZE 5
   ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname,Lname,….) VALUES('Robert','Smith',…)
EXEC SQL UPDATE EMPLOYEE
  SET Salary = Salary *1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END:…;
```

Desirable Properties of Transaction

- **Atomicity.** A transaction is an atomic unit of processing
- **Consistency preservation.** A transaction should be consistency executed without interference with other transactions, and it should take the database from one consistent state to another.
- **Isolation.** The execution of one transaction should not be interfere with execution of another transaction
- **Durability or permanency.** The changes applied tot eh database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

Break 10 min

Concurrency Control Techniques (CCT)

- Protocol-based:
  - Lock-Based Protocol
  - Two-Phase Locking Protocols
  - Timestamp Based protocols
  - Validation-Based Protcols
- Multiversion CCT:
  - Optimistic protocols and
  - Snapshot isolation based CCT

Two –phase locking protocols CCT (1)

- Applying a lock to the transaction data which blocks other transactions to access the same data simultaneously
- The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:
  - **Growing Phase**: In this phase transaction may obtain locks but may not release any locks.
  - **Shrinking Phase**: In this phase, a transaction may release locks but not obtain any new lock
- Two unwanted situations may occur*: starvation and deadlock*
- Other variations: Strict Two-Phase Locking Method, Centralized 2PL, primary copy 2PL, Distributed 2PL

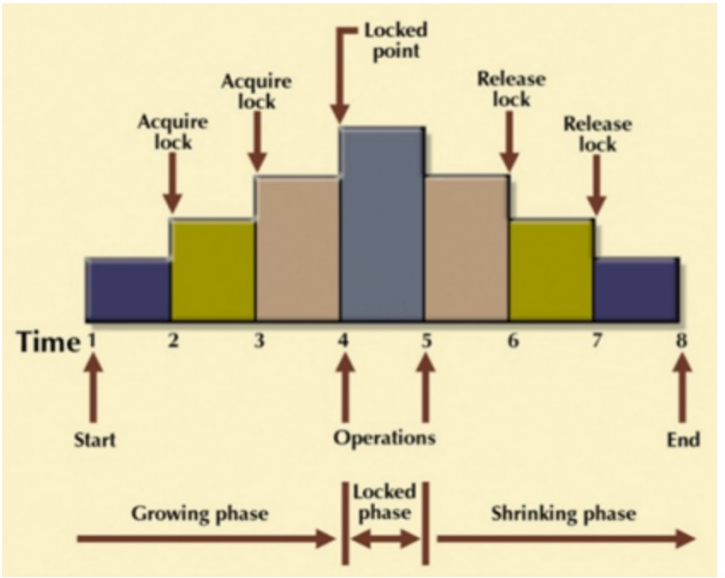


Image taken from : https://www.guru99.com/dbms-concurrency-control.html#4

Two –phase locking protocols CCT (2)

- A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it
  - There are *binary locks* (not used much) and read/write locks (or shared/Exclusive)
  - Shared lock allows to several transactions to access the same item X if they all access X for ***reading purposes only***.
  - Exclusive lock is when the transaction it about to write item into X
- There are three locks: read_lock(X), write_lock(X) and unlock(X).
- Three possible states*: read_locked, write_locked, unlocked*

Two –phase locking protocols CCT(2)

```
read_lock(X):
B:  if LOCK(X) = "unlocked"
        then begin LOCK(X) ← "read-locked";
            no_of_reads(X) ← 1
            end
    else if LOCK(X) = "read-locked"
        then no_of_reads(X) ← no_of_reads(X) + 1
    else begin
            wait (until LOCK(X) = "unlocked"
                and the lock manager wakes up the transaction);
            go to B
            end;
write_lock(X):
B:  if LOCK(X) = "unlocked"
        then LOCK(X) ← "write-locked"
    else begin
            wait (until LOCK(X) = "unlocked"
                and the lock manager wakes up the transaction);
            go to B
            end;
unlock (X):
    if LOCK(X) = "write-locked"
        then begin LOCK(X) ← "unlocked";
                wakeup one of the waiting transactions, if any
                end
    else it LOCK(X) = "read-locked"
        then begin
                no_of_reads(X) ← no_of_reads(X) –1;
                if no_of_reads(X) = 0
                    then begin LOCK(X) = "unlocked";
                            wakeup one of the waiting transactions, if any
                            end
                end;
```

Time

T1
read_lock(Y)
read_item(Y);
unlock(Y);
write_lock(X);
read_item(X);
X:=X+Y;
write_item(X);
unlock(X);

T2
read_lock(X);
read_item(X);
unlock(X);
write_lock(Y);
read_item(Y);
Y:=X+Y;
write_item(Y);
unlock(Y);

Initial values: X=20, Y=30

Result T1 followed by T2 : X=50, Y=80

Result T2 followed by T1: X=70, Y=50

Example of Simple Locking

There are 3 transactions that are all attempting to make changes to a single row in Table A.
U1 obtains an *exclusive lock* on this table when issuing the first update statement. Subsequently, U2 attempts to update the same row and is blocked by U1's lock. U3 also attempts to manipulate this same row, this time with a delete statement, and that is also blocked by U1's lock.
When U1 commits its transaction, it releases the lock and U2's update statement is allowed to complete. In the process, U2 obtains an exclusive lock and U3 continues to block. Only when U2's transaction is rolled back does the U3's delete statement complete.

Time	User 1 Actions	User 2 Actions	User 3 Actions
1	Starts Transaction		
2		Starts Transaction	
3			Starts Transaction
4	Updates row 2 in table A		
5		Attempts to update row 2 in table A	
		U2 Is Blocked by U1	
6			Attempts to delete row 2 in table A
			U3 Is Blocked by U1
7	Commits transaction		
		Update completes	*U3 Is Blocked by U2*
8		Rolls back transaction	
			Delete completes
9			Commits transaction

More examples: https://www.methodsandtools.com/archive/archive.php?id=83

Starvation

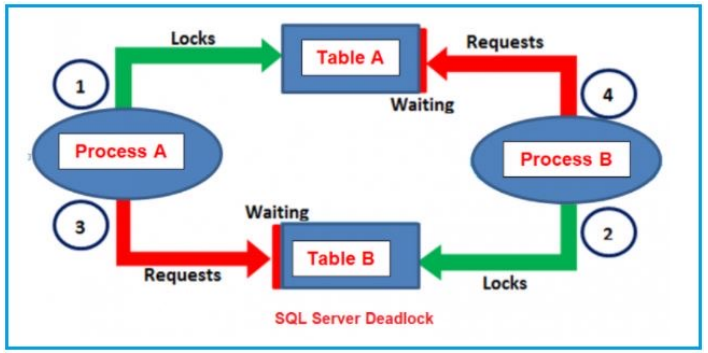
- Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.

Following are the reasons for Starvation:

- When waiting scheme for locked items is not properly managed
- In the case of resource leak
- The same transaction is selected as a victim repeatedly

Deadlock

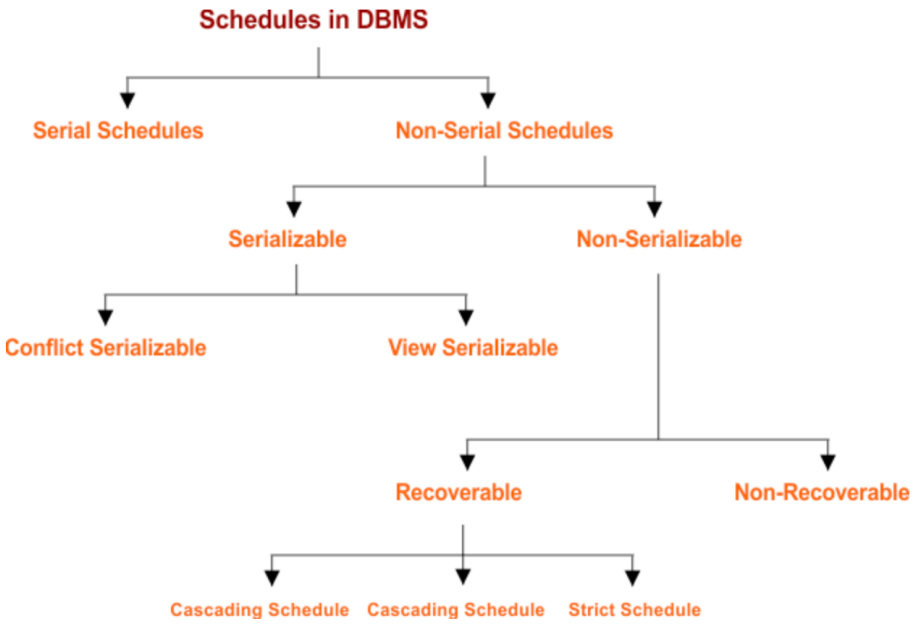
- Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.



Time	User 1 Actions	User 2 Actions
1	Starts Transaction	
2		Starts Transaction
3	Updates row 2 in table A	
4		Updates row 10 in table B
5	Attempts to update row 10 in table B	
	U1 Is Blocked by U2	
6		Attempts to update row 2 in table A
	U1 Is Blocked by U2	*U2 Is Blocked by U1*
	DEADLOCK!!!	

Schedules and Its Types

- Schedules define the order of the operation of the each transaction.



Serial Schedules

- Schedules in which no transaction can starts until a running transaction is ended are called serial schedules.
- Advantage: no concurrency problem, works good for single user application
- Disadvantage: It is very time consuming approach which is not applicable for multiuser application

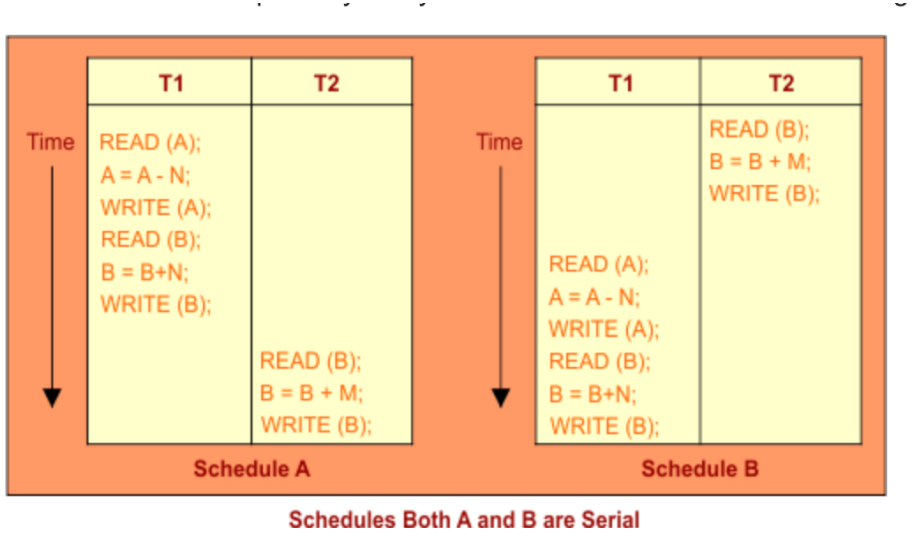
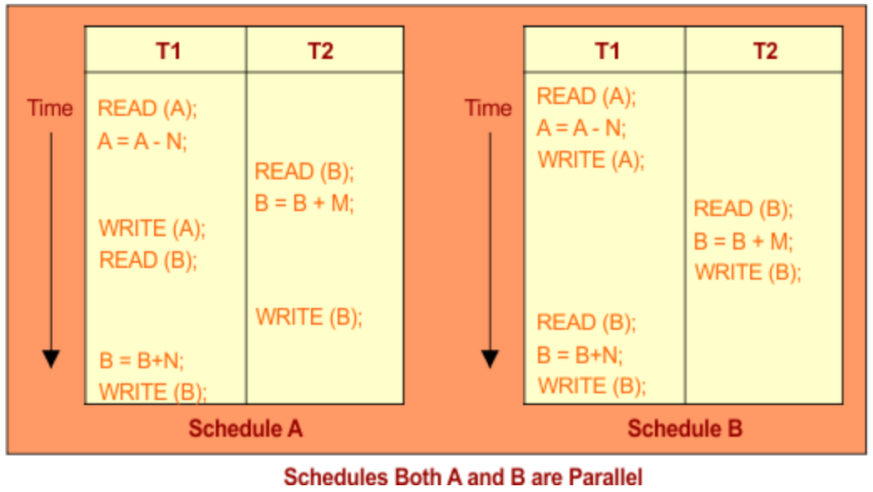


Image and Example is taken from: https://cstaleem.com/schedules-and-its-types

Non-serial (Parallel Schedule)

- In the non-serial schedule, the other transaction proceeds without waiting for the previous transaction to complete.
- Cause the concurrency problem and write conflict problems

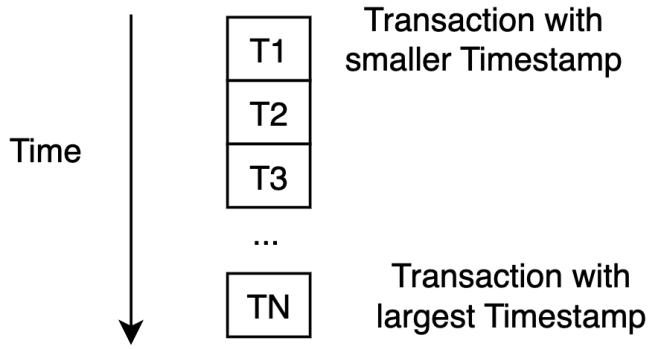


CCT based on Timestamps ordering (TO)

- A timestamp is unique identifies created by DBMS to identify a transaction. Also known as transaction start time (and they are ordered by time). Can be generated in several ways:
  - Using a counter that increments automatically the transaction number like 1,2,3,….to fin9ite maximum value (and reset to 0 in some certain condition)
  - Or use current system date/time value and ensure that no two timestamps values are generated during the same tick of the clock.

- The TO algorithm ensure that for each pair of conflicting operations in the schedule, the order in which item is accessed must follow the timestamp order.
  - It uses two timestamp values : Read_TS(X) and Write_TS(X)
  - Rule-based comparison algorithm of timestamps performed in order to check if the order is not violated. If the violation is occurred the transaction is aborted and resubmitted to the system as a new transaction with a new timestamp.
  - Deadlocks are not occurred with this approach in comparison to previous CCT method
  - Hence, Starvation is possible if the same transaction is restarted and continually aborted

- Examples : https://cstaleem.com/timestamp-ordering-with-examples



Multiversion Concurrency Control Technique

- Several versions (copies) are kept by the system when the data is updated
- A drawback is more storage is needed to maintain multiple version of the database.
- Support different CCT techniques:
  - Two-phase locking protocols
  - Timestamp ordering
  - Validation based technique
  - Snapshot Isolation

Validation based technique

- Uses three phases:
  - 1 Read Phase. A transaction can read values of committed data items from the database. Hence, updates are applied only to local copies (versions_ of the data items kept in the transaction workspace.
  - 2 Validation Phase. Checking is performed to ensure that serializability will not be violated if the transaction updates are applied to the database
  - 3 Write Phase. If the validation phase is successful, the transaction updates are applied to the database, otherwise the updates are discarded and the transaction is restarted.
- This approach is also called optimistic because it assume not much interference between transactions which is not in a real life cases.

Snapshot Isolation

- Snapshot isolation is that a transaction begins in a certain snapshot isolated state (copy) of the database at the time when the transaction is started.
- Some anomalies that violates serializability can occur, hence they are very rare and difficult to detect.
- Uses pointers to the list of items in the tempstore, the temstore items will be removed when they not needed longer.
- Variations of this technique have been used by Oracle and PostGres DBMS.

MySQL

- Locks in MySQL Server:
  - https://www.sqlshack.com/locking-sql-server/
- Optimistic Locking in MySQL: https://stackoverflow.com/questions/17431338/optimistic-locking-in-mysql
- Working out with deadlocks in MySQL: https://stackoverflow.com/questions/62846528/default-concurrency-control-implementation-in-mysql
- Uses Internal and External Locking methods which can be read in more detail at the MySQL documentation:
  - https://dev.mysql.com/doc/refman/8.0/en/internal-locking.html
  - https://dev.mysql.com/doc/refman/8.0/en/external-locking.html

Select Behavior in SQL and Oracle

Select Behavior in Oracle

Time	User 1 Actions	User 2 Actions
1	Starts Transaction	
2		Starts Transaction
3	Updates row 2 in table A	
4		Attempts to read row 2 in table A
		Select statement completes, returning data unchanged by U1
5	Commits transaction	
6		Attempts to read row 2 in table A
		Select statement completes, returning changed data

Select Behavior in SQL Server

Time	User 1 Actions	User 2 Actions
1	Starts Transaction	
2		Starts Transaction
3	Updates row 2 in table A	
4		Attempts to read row 2 in table A
		U2 Is Blocked by U1
5	Commits transaction	
		Select statement completes, returning changed data

Summary

- Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another.
- Lost Updates, dirty read, Non-Repeatable Read, and Incorrect Summary Issue are problems faced due to lack of concurrency control.
- Lock-Based, Two-Phase, Timestamp-Based, Validation-Based, Snapshot Isolation are types of Concurrency handling protocols
- The transaction logic is handled on the application program side, concurrency control manager of database server handles the locks
- Database administrator/developer must take care of the database recovery and backup process/logic.
- Different DBMS have different transaction isolation implementation/logic.

References

- Database Locking: What it is, Why it Matters and What to do About it? https://www.methodsandtools.com/archive/archive.php?id=83