KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

ST. JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CST 204 - Database Management Systems

Prof. Sarju S

30 September 2021

CST 204 – Database Management Systems

Module 5

- Transaction Processing Concepts - overview of concurrency control, Transaction Model, Significance of concurrency Control & Recovery, Transaction States, System Log, Desirable Properties of transactions.
- Serial schedules, Concurrent and Serializable Schedules, Conflict equivalence and conflict serializability, Recoverable and cascade-less schedules, Locking, Two-phase locking and its variations. Log-based recovery, Deferred database modification, check-pointing.
- Introduction to NoSQL Databases, Main characteristics of Key-value DB (examples from: Redis), Document DB (examples from: MongoDB)
- Main characteristics of Column - Family DB (examples from: Cassandra) and Graph DB (examples from : ArangoDB)

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Schedules

Schedules

- A schedule (or history) S of n transactions T1, T2, ..., Tn is an ordering of the operations of the transactions.
- Operations from different transactions can be interleaved in the schedule S.

Recoverable Schedules

- Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules.
  - If some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

T_1	T_2
R(A)	
W(A)	
	W(A)
	R(A)
commit	
	commit

This is a recoverable schedule since T_1 commits before T_2, that makes the value read by T_2 correct.

Irrecoverable Schedule

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		
Failure Point				
Commit;				

- The table shows a schedule with two transactions, T_1 reads and writes A and that value is read and written by T_2.
- T_2 commits. But later on, T_1 fails.
- So we have to rollback T_1. Since T_2 has read the value written by T_1, it should also be rollbacked.
- But we have already committed that.
- So this schedule is irrecoverable schedule.
- When T_j is reading the value updated by T_i and T_j is committed before committing of T_i, the schedule will be irrecoverable.

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Recoverable with Cascading Rollback:

- Cascading Rollback (or cascading abort) to occur in some recoverable schedules, where an *uncommitted transaction has to be rolled back because it read an item from a transaction that failed*.

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
Failure Point				
Commit;				
		Commit;		

- The table shows a schedule with two transactions, T_1 reads and writes A and that value is read and written by T_2.
- But later on, T_1 fails. So we have to rollback T_1.
- Since T_2 has read the value written by T_1, it should also be rollbacked.
- As it has not committed, we can rollback T_2 as well. So it is recoverable with cascading rollback.
- *If T_j is reading value updated by T_i and commit of T_j is delayed till commit of T_i, the schedule is called recoverable with cascading rollback.*

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Cascadeless Recoverable Rollback

- A schedule is said to be cascadeless, or to avoid cascading rollback, *if every transaction in the schedule reads only items that were written by committed transactions.*

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
Commit;				
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		

- The table shows a schedule with two transactions, T1 reads and writes A and commits and that value is read by T2.
- But if T1 fails before commit, no other transaction has read its value, so there is no need to rollback other transaction.
- So this is a Cascadeless recoverable schedule.
- *If Tj reads value updated by Ti only after Ti is committed, the schedule will be cascadeless recoverable.*

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Serial Schedules

- Schedules in which the transactions are executed non-interleaved
  - a serial schedule is one in which no transaction starts until a running transaction has ended are called serial schedules.

T_1	T_2
R(A)	
W(A)	
R(B)	
	W(B)
	R(A)
	R(B)

Example: Consider the schedule involving two transactions T1 and T2. This is a serial schedule since the transactions perform serially in the order T1 —> T2

Non-Serial Schedule

- This is a type of Scheduling where the operations of multiple transactions are interleaved.
  - Unlike the serial schedule where one transaction must wait for another to complete all its operation, in the non-serial schedule, the other transaction proceeds without waiting for the previous transaction to complete.
- This might lead to a rise in the concurrency problem.
- It can be of two types namely, Serializable and Non-Serializable Schedule.

Serializable Schedule

- This is used to maintain the consistency of the database.
- It is mainly used in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not.
- These are of two types:
  - Conflict Serializable
  - View Serializable:

Conflict Serializable Schedule

- A schedule is called conflict serializability if after swapping of non-conflicting operations, it can transform into a serial schedule.
- The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

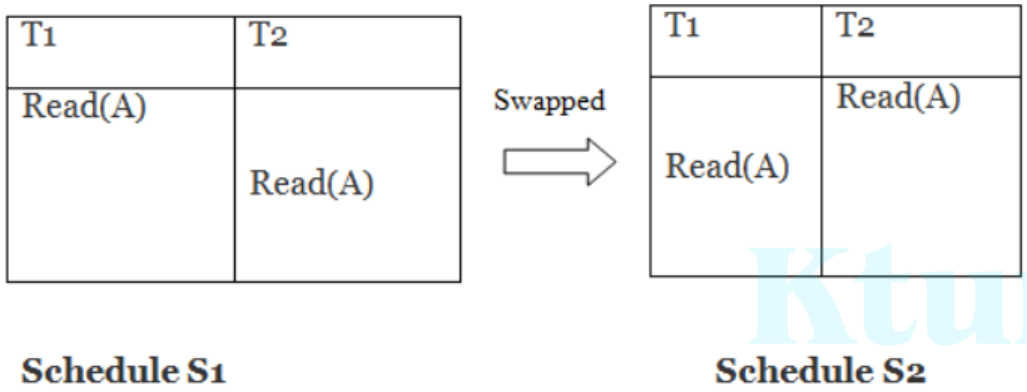
Conflict Serializable Schedule

Conflicting Operations

- The two operations become conflicting if all conditions satisfy:
  - Both belong to separate transactions.
  - They have the same data item.
  - They contain at least one write operation.

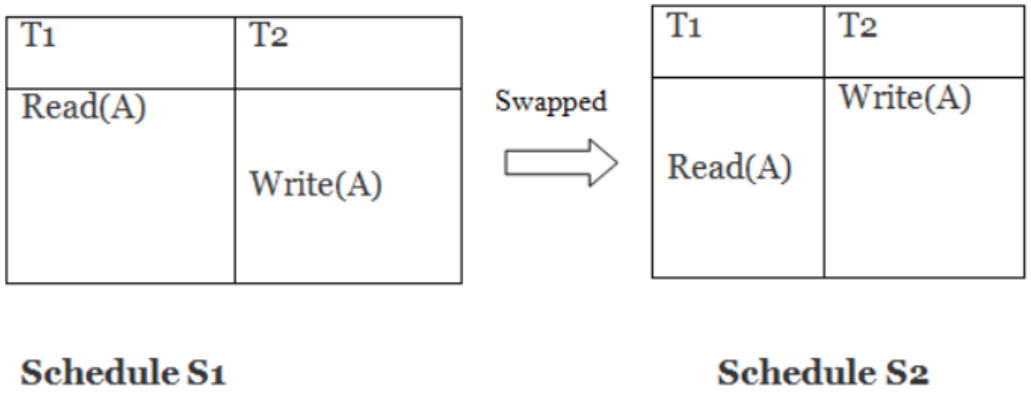
Conflict Serializable Schedule

Conflicting Operations



Schedule S1 **Schedule S2**

Here, S1 = S2. That means it is non-conflict.



Schedule S1 **Schedule S2**

Here, S1 ≠ S2. That means it is conflict

Conflict Serializable Schedule

Conflict Equivalent

- In the conflict equivalent, one can be transformed to another by swapping non-conflicting operations.
- Two schedules are said to be conflict equivalent if and only if:
  - They contain the same set of the transaction.
  - If each pair of conflict operations are ordered in the same way.

Conflict Serializable Schedule

Conflict Equivalent

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Schedule S1

T1	T2
Read(A) Write(A) Read(B) Write(B)	Read(A) Write(A) Read(B) Write(B)

Schedule S1 can be transformed into a serial schedule by swapping non-conflicting operations of S1.

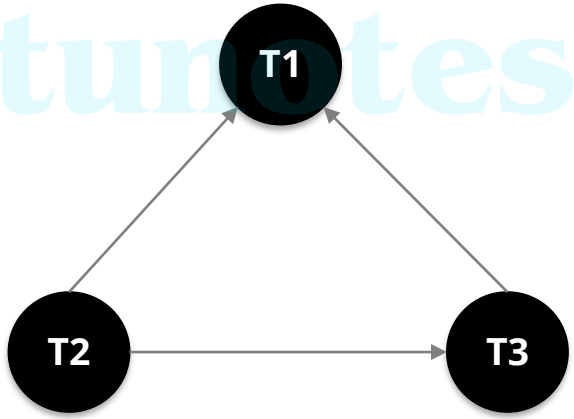
Testing for Conflict Serializability of a Schedule

- There is a simple algorithm for determining whether a particular schedule is conflict serializable or not.
- The Algorithm can be written as:
  1. Create a node T in the graph for each participating transaction in the schedule.
  2. For the conflicting operation read_item(X) and write_item(X) – If a Transaction Tj executes a read_item (X) after Ti executes a write_item (X), draw an edge from Ti to Tj in the graph.
  3. For the conflicting operation write_item(X) and read_item(X) – If a Transaction Tj executes a write_item (X) after Ti executes a read_item (X), draw an edge from Ti to Tj in the graph.
  4. For the conflicting operation write_item(X) and write_item(X) – If a Transaction Tj executes a write_item (X) after Ti executes a write_item (X), draw an edge from Ti to Tj in the graph.
  5. The Schedule S is serializable if there is no cycle in the precedence graph.

Testing for Conflict Serializability of a Schedule - Example

T1	T2	T3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)



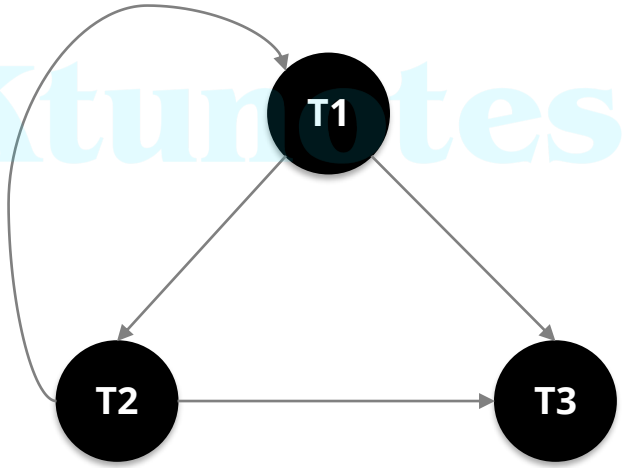
- T1-R(x): no W(x) in T2, T3
- T3-R(y): no W(y) in T2, T1
- T3-R(x): W(x) in T1 draw the edge T3->T1
- T2-R(y): W(y) in T3 draw the edge T2->T3
- T2-R(z): W(z) in T1 draw the edge T2->T1
- T3-W(y): no R(y) or W(y) in T2, T1
- T2-W(z): R(z) and W(z) in T1 draw edge T2->T1(already there)

- As we have no Cycle/Loop in the Precedence Graph these schedule is conflict serializable

Testing for Conflict Serializability of a Schedule – Example 2

T1	T2	T3
R(A)		
	W(A)	
W(A)		
		W(A)

- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)

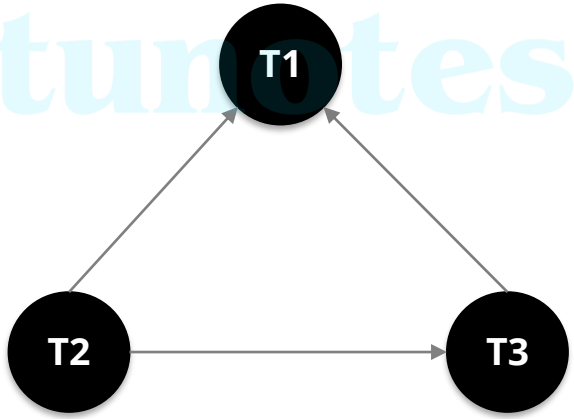


- As we have Cycle/Loop in the Precedence Graph this schedule is Non conflict serializable . Is this serializable?

University Previous Question Paper Question

Check if the following schedules are conflict-serializable using precedence graph. If so, give the equivalent serial schedule(s). r3(X), r2(X), w3(X), r1(X), w1(X). (Note: ri(X)/wi(X) means transaction Ti issues read/write on item X.)

T1	T2	T3
R(x) W(x)	R(x)	R(x) W(x)



- Draw Edge when we have
- read_item(X) and write_item(X)
- write_item(X) and read_item(X)
- write_item(X) and write_item(X)

- As we have no Cycle/Loop in the Precedence Graph this schedule is conflict serializable .

What is View Serializability?

- View Serializability is a process to find out that a given schedule is view serializable or not.
- To check whether a given schedule is view serializable, we need to check whether the given schedule is View Equivalent to its serial schedule.
- Lets take an example to understand what I mean by that.

T1	T2	T3
R(A)		
	W(A)	
W(A)		
		W(A)

View Equivalent →

T1	T2	T3
R(A)		
W(A)		
	W(A)	
		W(A)

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Initial Read
    - An initial read of both schedules must be the same.
    - Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A)	
	Write(A)

Schedule S1

T1	T2
	Write(A)
Read(A)	

Schedule S2

These schedules are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

View Serializability - View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Updated Read
    - In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

T1	T2	T3
Write(A)		
	Write(A)	
		Read(A)

Schedule S1

T1	T2	T3
Write(A)		
	Write(A)	
		Read(A)

Schedule S2

These two schedules are not view equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

View Serializability - View Equivalent

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:
  - Final Write
    - A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

T1	T2	T3
Write(A)		
	Read(A)	
		Write(A)

Schedule S1

T1	T2	T3
	Read(A)	
Write(A)		
		Write(A)

Schedule S2

These two schedules is view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

View Serializability - Example

Non-Serial

S1

T1	T2
R(X)	
W(X)	
	R(X)
	W(X)
R(Y)	
W(Y)	
	R(Y)
	W(Y)

Serial

Beginnersbook.com

S2

T1	T2
R(X)	
W(X)	
R(Y)	
W(Y)	
	R(X)
	W(X)
	R(Y)
	W(Y)

S2 is the serial schedule of S1. If we can prove that they are view equivalent then we we can say that given schedule S1 is view Serializable

View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Initial Read
  - In schedule S1, transaction T1 first reads the data item X.
  - In S2 also transaction T1 first reads the data item X.
  - Lets check for Y. In schedule S1, transaction T1 first reads the data item Y. In S2 also the first read operation on Y is performed by T1.
  - We checked for both data items X & Y and the initial read condition is satisfied in S1 & S2.

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Final Write
  - In schedule S1, the final write operation on X is done by transaction T2. In S2 also transaction T2 performs the final write on X.
  - Lets check for Y. In schedule S1, the final write operation on Y is done by transaction T2. In schedule S2, final write on Y is done by T2.
  - We checked for both data items X & Y and the final write condition is satisfied in S1 & S2.

View Serializability - View Equivalent

- Lets check the three conditions of view serializability:
- Update Read
  - In S1, transaction T2 reads the value of X, written by T1. In S2, the same transaction T2 reads the X after it is written by T1.
  - In S1, transaction T2 reads the value of Y, written by T1. In S2, the same transaction T2 reads the value of Y after it is updated by T1.
  - The update read condition is also satisfied for both the schedules.
- **Result:** Since all the three conditions that checks whether the two schedules are view equivalent are satisfied in this example, which means S1 and S2 are view equivalent.

References

- https://www.geeksforgeeks.org/recoverability-in-dbms/
- https://www.geeksforgeeks.org/types-of-schedules-in-dbms/
- https://www.javatpoint.com/dbms-conflict-serializable-schedule
- https://beginnersbook.com/2018/12/dbms-view-serializability/

Module 5

- Transaction Processing Concepts - overview of concurrency control, Transaction Model, Significance of concurrency Control & Recovery, Transaction States, System Log, Desirable Properties of transactions.
- Serial schedules, Concurrent and Serializable Schedules, Conflict equivalence and conflict serializability, Recoverable and cascade-less schedules, Locking, Two-phase locking and its variations. Log-based recovery, Deferred database modification, check-pointing.
- Introduction to NoSQL Databases, Main characteristics of Key-value DB (examples from: Redis), Document DB (examples from: MongoDB)
- Main characteristics of Column - Family DB (examples from: Cassandra) and Graph DB (examples from : ArangoDB)

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Introduction to Transaction Processing

Transactions

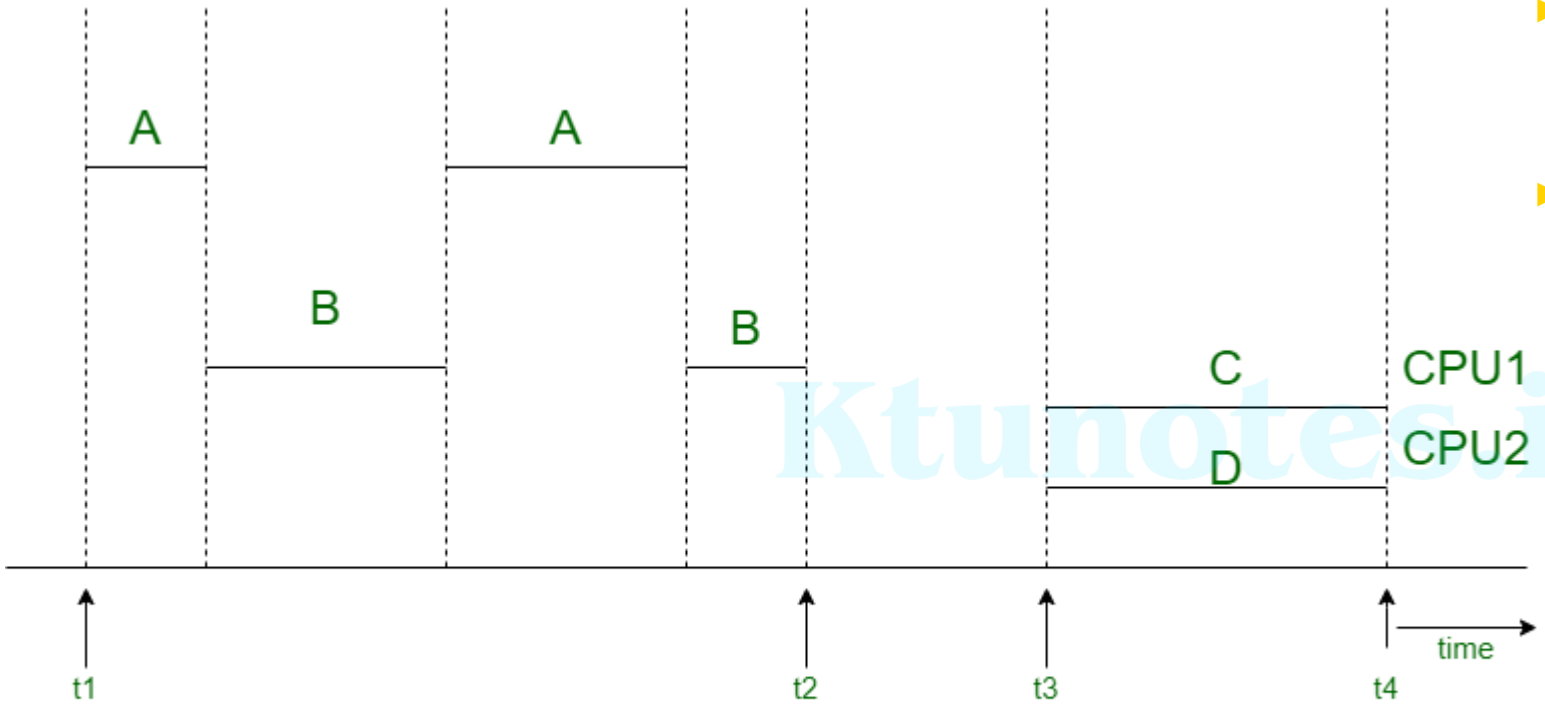
- A Database Transaction is a logical unit of processing in a DBMS which entails one or more database access operation.
- The operations performed in a transaction include one or more of database operations like insert, delete, update or retrieve data.

Single User and Multi User Database Systems

Single User Database Systems	Multi User Database Systems
A DBMS is single-user if at most one user at a time can use the system.	A DBMS is multi-user if many/multi users can use the system and hence access the database concurrently.
Single-User DBMSs are mostly restricted to personal computer systems.	Most DBMSs are multi user, like databases of airline reservation systems, banking databases, etc.
Single user databases do not have multiprogramming thus, single CPU can only execute at most one process at a time.	Multiple users can access databases and use computer systems simultaneously because of the concept of Multiprogramming.
Example: Personal Computers.	Example: Databases of Banks, insurance agencies, stock exchanges, supermarkets, etc.

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Interleaved Processing Vs Parallel processing



Interleaved processing vs Parallel processing of concurrent transactions.

- The figure shows two processes, A and B, executing concurrently in an interleaved fashion.
- if the computer system has multiple hardware processors (CPUs), parallel processing of multiple processes is possible, as illustrated by processes C and D in the figure.

Source: https://www.geeksforgeeks.org/

Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Transaction Operations

- Each high level operation can be divided into a number of low level tasks or operations.
- For example, a data update operation can be divided into three tasks
  - **read_item()** − reads data item from storage to main memory.
  - **modify_item()** − change value of item in the main memory.
  - **write_item()** − write the modified value from main memory to storage.

Transaction Operations

- The low level operations performed in a transaction are −
- **begin_transaction** − A marker that specifies start of transaction execution.
- **read_item** or **write_item** − Database operations that may be interleaved with main memory operations as a part of transaction.
- **end_transaction** − A marker that specifies end of transaction.
- **commit** − A signal to specify that the transaction has been successfully completed in its entirety and will not be undone.
- **rollback** − A signal to specify that the transaction has been unsuccessful and so all temporary changes in the database are undone. A committed transaction cannot be rolled back.

Why Concurrency Control Is Needed?

- When multiple transactions execute concurrently in an uncontrolled or unrestricted manner, then it might lead to several problems.
- These problems are commonly referred to as concurrency problems in database environment.
- The five concurrency problems that can occur in database are:
  - Temporary Update Problem
  - Incorrect Summary Problem
  - Lost Update Problem
  - Unrepeatable Read Problem

Temporary Update Problem

- Temporary update or dirty read problem occurs when one transaction updates an item and fails.
- But the updated item is used by another transaction before the item is changed or reverted back to its last value.

T1	T2
read_item(X) X = X - N write_item(X)	
	read_item(X) X = X + M write_item(X)
read_item(Y)	

- In this example, if transaction 1 fails for some reason then X will revert back to its previous value.
- But transaction 2 has already read the incorrect value of X.

Incorrect Summary Problem

- Consider a situation, where one transaction is applying the aggregate function on some records while another transaction is updating these records.
- The aggregate function may calculate some values before the values have been updated and others after they are updated.

T1	T2
	sum = 0 read_item(A) sum = sum + A
read_item(X) X = X - N write_item(X)	
	read_item(X) sum = sum + X read_item(Y) sum = sum + Y
read_item(Y) Y = Y + N write_item(Y)	

- In this example, transaction 2 is calculating the sum of some records while transaction 1 is updating them.
- Therefore the aggregate function may calculate some values before they have been updated and others after they have been updated.

Unrepeatable Read Problem:

- The unrepeatable problem occurs when two or more read operations of the same transaction read different values of the same variable.

T1	T2
Read(X)	
	Read(X)
Write(X)	
	Read(X)

- In this example, once transaction 2 reads the variable X, a write operation in transaction 1 changes the value of the variable X.
- Thus, when another read operation is performed by transaction 2, it reads the new value of X which was updated by transaction 1.

Lost Update Problem

- In the lost update problem, update done to a data item by a transaction is lost as it is overwritten by the update done by another transaction.

T1	T2
read_item(X) X = X + N	X = X + 10 write_item(X)

- In this example, transaction 1 changes the value of X but it gets overwritten by the update done by transaction 2 on X.
- Therefore, the update done by transaction 1 is lost.

Why Recovery Is Needed

- Whenever a transaction is submitted to a DBMS for execution
  - The system is responsible for making sure that either all the operations in the transaction are completed successfully and their effect is recorded permanently in the database or that the transaction does not have any effect on the database or any other transactions.
- Types of Failures
  - **A computer failure (system crash).** A hardware, software, or network error occurs in the computer system during transaction execution.
  - **A transaction or system error**. Some operation in the transaction may cause it to fail,
    - Integer overflow or division by zero.
    - Erroneous parameter values
    - Logical programming error.
    - User may interrupt the transaction during its execution

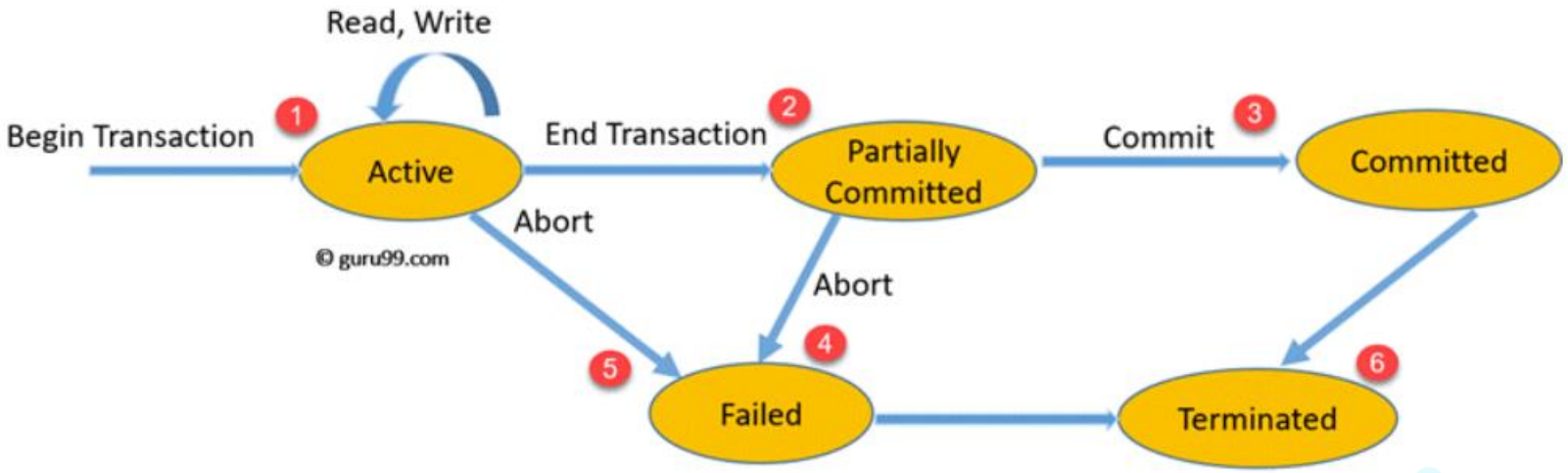
Why Recovery Is Needed

- Types of Failures
  - **Local errors or exception conditions detected by the transaction**
    - During transaction execution, certain conditions may occur that necessitate cancellation of the transaction.
    - Eg: Insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be canceled.
  - **Disk failure**
    - Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash.
    - This may happen during a read or a write operation of the transaction.
  - Physical problems and catastrophes.
    - This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

States of Transactions

State	Transaction types
Active State	A transaction enters into an active state when the execution process begins. During this state read or write operations can be performed.
Partially Committed	A transaction goes into the partially committed state after the end of a transaction.
Committed State	When the transaction is committed to state, it has already completed its execution successfully. Moreover, all of its changes are recorded to the database permanently.
Failed State	A transaction considers failed when any one of the checks fails or if the transaction is aborted while it is in the active state.
Terminated State	State of transaction reaches terminated state when certain transactions which are leaving the system can't be restarted.

States of Transactions



State Transition Diagram for a Database Transaction

1. Once a transaction states execution, it becomes active. It can issue READ or WRITE operation.
2. Once the READ and WRITE operations complete, the transactions becomes partially committed state.
3. Next, some recovery protocols need to ensure that a system failure will not result in an inability to record changes in the transaction permanently. If this check is a success, the transaction commits and enters into the committed state.
4. If the check is a fail, the transaction goes to the Failed state.
5. If the transaction is aborted while it's in the active state, it goes to the failed state. The transaction should be rolled back to undo the effect of its write operations on the database.
6. The terminated state refers to the transaction leaving the system.

The System Log

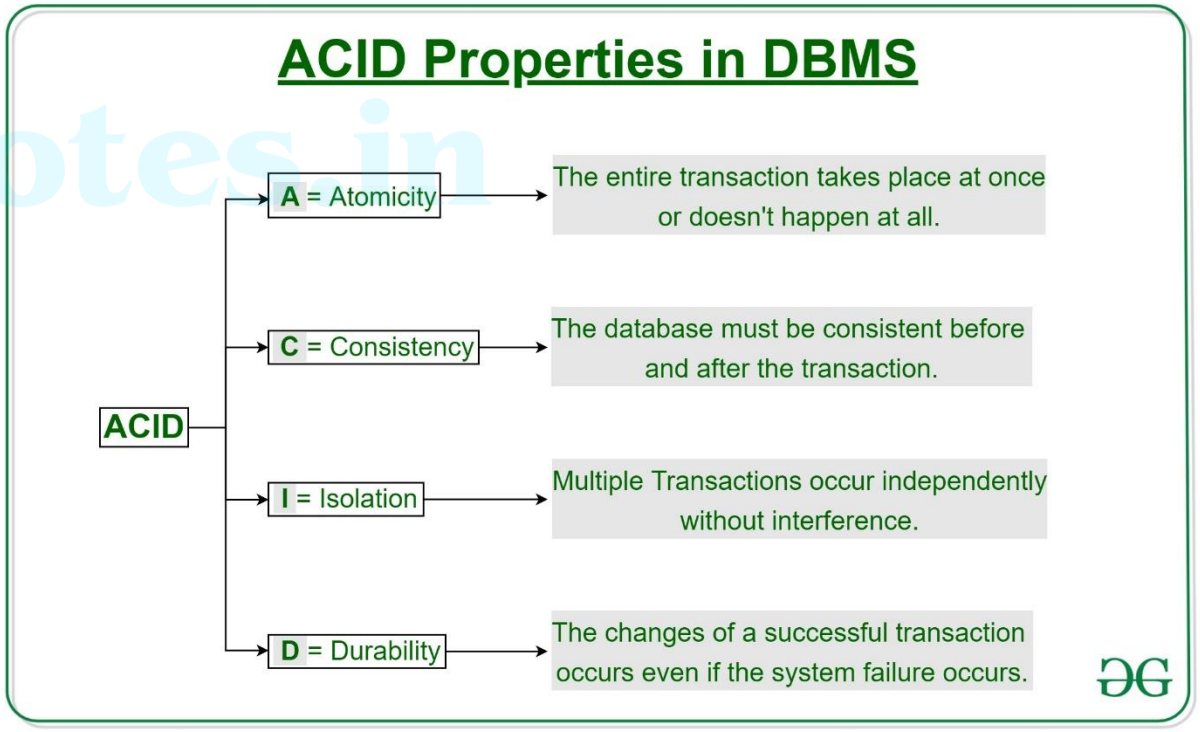
- To be able to recover from failures that affect transactions, the system maintains a log
  - to keep track of all transaction operations that affect the values of database items,
  - as well as other transaction information that may be needed to permit recovery from failures.
- In a stable storage, logs for each transaction are maintained.
- Any operation which is performed on the database is recorded is on the log.
- Prior to performing any modification to database, an update log record is created to reflect that modification.

The System Log - Example

1. [start_transaction, T]. Indicates that transaction T has started execution.
2. [write_item, T, X, old_value, new_value]. Indicates that transaction T has changed the value of database item X from old_value to new_value.
3. [read_item, T, X]. Indicates that transaction T has read the value of database item X.
4. [commit, T]. Indicates that transaction T has completed successfully, and affirms that its effect can be committed (recorded permanently) to the database.
5. [abort, T]. Indicates that transaction T has been aborted.

ACID Properties in DBMS

- ACID Properties are used for maintaining the integrity of database during transaction processing.
- ACID in DBMS stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.



Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

Atomicity

- By this, we mean that either the entire transaction takes place at once or doesn't happen at all.
- There is no midway i.e. transactions do not occur partially.
- Each transaction is considered as one unit and either runs to completion or is not executed at all.
- It involves the following two operations.
  - Abort: If a transaction aborts, changes made to database are not visible.
  - Commit: If a transaction commits, changes made are visible.
- Atomicity is also known as the 'All or nothing rule'.

Atomicity

- Consider the following transaction T consisting of T1 and T2: Transfer of 100 from account X to account Y.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

- If the transaction fails after completion of T1 but before completion of T2.(say, after write(X) but before write(Y)), then amount has been deducted from X but not added to Y.
- This results in an inconsistent database state. Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

Consistency

- This means that integrity constraints must be maintained so that the database is consistent before and after the transaction.
- It refers to the correctness of a database.
- Referring to the example above,
- The total amount before and after the transaction must be maintained.
- Total before T occurs = 500 + 200 = 700.
- Total after T occurs = 400 + 300 = 700.
- Therefore, database is consistent. Inconsistency occurs in case T1 completes but T2 fails. As a result T is incomplete.

Isolation

- In a database system where more than one transaction are being executed simultaneously and in parallel,
  - the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
  - No transaction will affect the existence of any other transaction.

Isolation

- Let X= 500, Y = 500. Consider two transactions T and T”.
-

T	T”
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write (Y)	

- Suppose T has been executed till Read (Y) and then T” starts. As a result , interleaving of operations takes place due to which T” reads correct value of X but incorrect value of Y and sum computed by
- T”: (X+Y = 50, 000+500=50, 500) is thus not consistent with the sum at end of transaction:
- T: (X+Y = 50, 000 + 450 = 50, 450).
- This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Durability

- The database should be durable enough to hold all its latest updates even if the system fails or restarts.
- If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data.
- If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

Why Concurrency Control Is Needed?

- When multiple transactions execute concurrently in an uncontrolled or unrestricted manner, then it might lead to several problems.
- These problems are commonly referred to as concurrency problems in database environment.
- The five concurrency problems that can occur in database are:
  - Temporary Update Problem
  - Incorrect Summary Problem
  - Lost Update Problem
  - Unrepeatable Read Problem
  - Phantom Read Problem

Thank You

Prof. Sarju S
Department of Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Palai
sarju.s@sjcetpalai.ac.in sarju-s