U4. Transaction Management And Query Processing

PAGE No.
DATE / /

Basic concept of Transaction, Describe properties of Transaction, Transaction state, concept of a schedule serializability, concurrenty management

Transaction

- The transaction is a set of logically related operation It contain a group of tasks
- A transaction is an action or series of actions. It is performed by a single user to perform operation for accessing the content of the database

Example:- Suppose an employee of bank transfer Rs 800 from X's account to Y's account, This small transaction contain several low-level tasks:

• Facts about Database Transactions

- A transaction is a program unit whose execution may or may not change the content of a database
- The transaction is executed as a single unit
- If the database operation do not update the database but only retrive data, this type of transaction is called a read-only transaction
- A successful transaction can change the database from one CONSISTENT STATE to another
- DBMS transaction must be atomic, consistent, isolated and durable
- If the database were in an inconsistent state before a transaction, it would remain in the inconsistent state after the transaction.

Operation of Transaction:-

Following are the main operations of Transaction:

Read(X): Read operation is used to read the value of X from the database and stores it in a buffer in main memory

Write(X): Write operation is used to write the value back to the database from the buffer

- Let's take an example to debit transaction from an account which consist of following operation

R(X);

X = X-500;

W(X);

Let's assume the value of X before starting of the transaction is 4000,

The first operation reads X's from database and store it in a buffer

The second operation will decreases the value of X by 500, So buffer will contain 3500

The third operation will write the buffer value to the database. So X's final value will be 3500

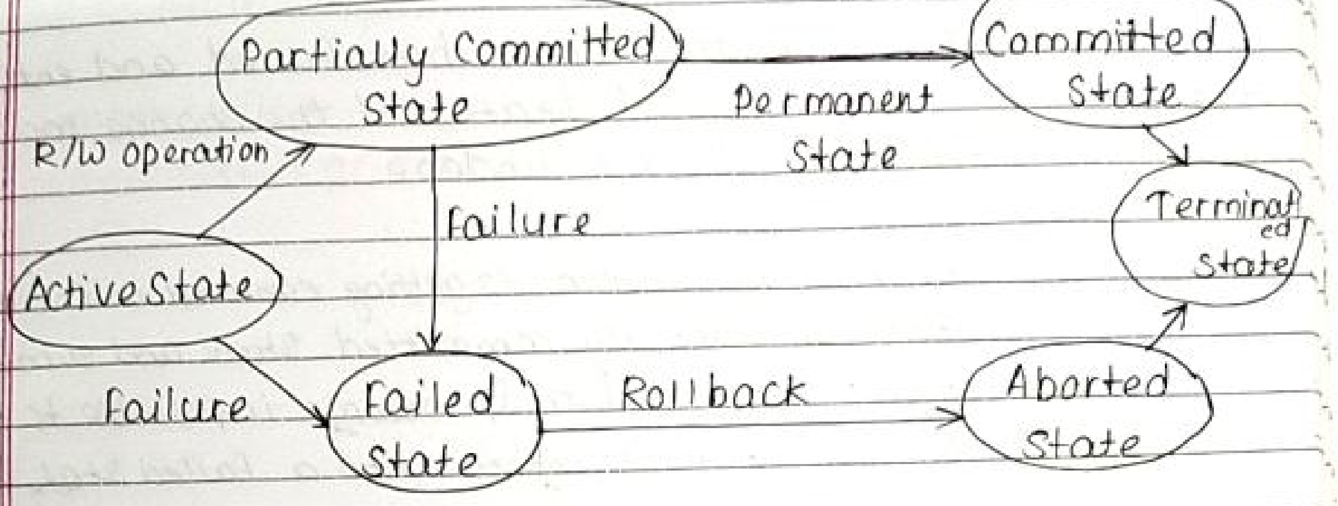
Example: If in a above transaction the debit transaction fail after executing operation 2 then X's value will remain 4000 in the database which is not acceptable by the bank

- To solve this problem, we have two important operations

Commit: It is used to solve the work done permanently

- Rollback: It is used to undo the work done

States Of Transaction



Transaction States in DBMS

The various States of Database Transaction are listed below

State	Transaction types
Active State	A transaction enters into an active state when the execution process begin. During this state read and write operation can be performed.
Partially Committed	After the last instruction Of transaction has executed it enter into a partially committed State After entering this state, the transaction is considered to be partially commited It is not considered fully commited because all the changes made by the transaction are still stored in the buffer in main memory

DATE / /

Committed State	When the transaction is commited to state, it has already completed its execution successfully
aborted State	After the transaction has failed and entered into a failed state, all the change made by it have to be undone
final State	When a transaction is getting executed in the active state or partially commited state and some failure occur due to which it become impossible to continue the execution, it enters into a failed State
Terminated State	After entering the committed state or aborted state, the transaction finally enters into a terminated state which its lifecycle finally come to an end.

* Transaction Property

The transaction has the four properties, These are used to maintain consistency in a database before and after the transaction

- Property of Transaction
- Atomicity
- Consistency
- Isolation
- Durability

PAGE No.

DATE / /

Atomicity

means either all Successful or none

Consistency

ensure bringing the database from the consistent state to another consistent state ensure bringing the database from one consistent state to another consistent state

Isolation

ensure that transaction is isolated from other transaction.

Durability

means once a transaction has been committed it will remain so, even in the event of errors, power loss, etc

DATE / /

- Atomicity
- This property ensure that either the transaction occurs completely or it does not occur at all
- In other words, it ensures that no transaction occurs partially
- That is why, it is also refered as "All or nothing rule"
- It is the responsibility of Transaction Control manager to ensure atomicity of the transactions
- Atomicity involve the following two operation
- Abort:- If a transaction then all the changes made are not visible
- Commit:- If a transaction commit then all the changes made are ~~avai~~ visible

Isolation
- This property ensure that multiple transactions can occur simultanously without causing any inconsistency
- During Execution, each transaction feel as if it getting executed alone in the system
- A transaction does not realize that there are other transaction as well getting executed parallely
- Changes made by a transaction becomes visible to other transaction only after they are written in the memory
- It is the responsibility of concurrency control language control manager to ensure Isolation for all the transactions.

- **Consistency**
- The integrity constraints are maintained so that the database is consistent before and after the transaction
- The Execution of a transaction will leave a database in either its prior stable state or a new stable state
- The transaction is used to transform the database from one consistent state to another consistent state

Example:- The total amount must be maintained before or after the transaction

Total before T occurs = 600 + 300 = 900

Total after T occurs = 500 + 400 = 900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then in consistency will occur.

- **Durability**
- This property ensures that all changes made by a transaction after its successful execution are written successfully to the disk
- It also ensures that these changes exit permanently and are never lost even if there occur a failure of any kind
- It is the responsibility of recovery manager to ensure durability in the database.

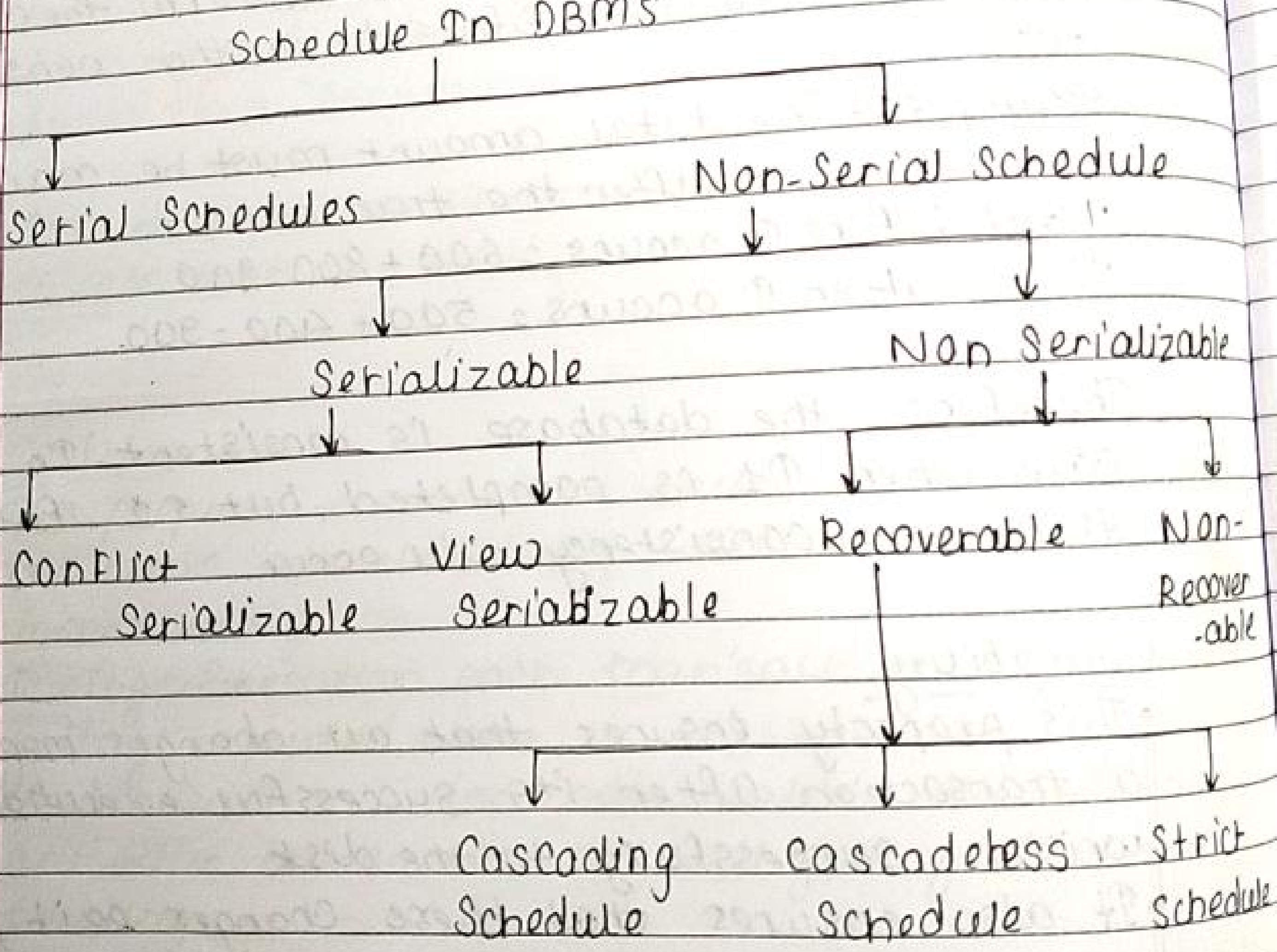
PAGE No.

DATE / /

* Schedule

A series of operation from one transation to another transaction is known as schedule. It is used to preserve the order of the operation in each of the individeial transaction.

• Schedules In DBMS



Serial Schedule

In serial schedules, All the transactions execute serially one after the other
When one transaction executes, no other transaction is allowed to execute

Characteristics -

- Serial Schedules are always -

consistent
Recoverable
cascadeless
strict

(a)

Time

T1	T2
read(A);	
A:=A-N;	
Write(A);	
read(B);	
B:=B+N;	
write(B);	
	read(A);
	A:=A+M
	Write(A);

Schedule A

(b)

Time

T1	T2
	read(A);
	A:=A+M;
	write(A);
read(A);	
A:=A-N;	
write(A);	
read(B);	
B:=B+N	
Write(B);	

Schedule B

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

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

- Non-Serial Schedule

In non serial schedule -

- multiple transaction execute concurrently
- Operations of all the transaction are inter leaved or mixed with each other

Characteristics -

- Non-Serial character schedule are NOT always -

Consistent
Recoverable
Cascadeless
Strict

(c)

Time

T_1	T_2
read(A);	
A:=A-N;	
	read(A);
	A:=A+M;
write(A);	
read(B);	write(A);
B:=B+N;	
write(B);	

Sechedule C

(d)

Time

T_1	T_2
read(A);	
A:=A-N;	
write(A);	
	read(A);
	A:=A+M;
	write(A);
read(B);	
B:=B+N;	
write(B);	

Schedule D

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

PAGE No.
DATE / /

Serial Schedules Vs Serializable Schedules -

Serial Schedules	Serializable Schedules
1. No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other	1. Concurrency is allowed. Thus, multiple transactions can execute concurrently
2. Serial schedule lead to less resource utilization and CPU throughput	2. Serializable schedule improve both resources utilization and CPU throughput
3. Serial schedules are less efficient as compared to serializable schedules (due to above reason)	3. Serializable schedules are always better than serial schedules (due to above reasons

- **Serializability in DBMS**
  - Some non-serial schedule may lead to inconsistency of the database
  - Serializability is a concept that help to indentify which non-serial schedule are correct and will maintain the consistency to the database.

- **Serializable Schedules -**

If a given non-serial schedule of 'n' transactions is equivalent to some serial schedule of 'n' transactions, then, it is called as a serializable schedule

PAGE No.
DATE / /

- Characteristics

Serializable schedules behave exactly same as serial schedules
Thus, serializable schedule are always-
consistent
Recoverable
Casacadeless
Strict

- Conflict Serializable Schedule.

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

Conflicting Operations

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

1. T_1: Read(A) T_2: Read(A)

T_1	T_2
Read(A)	
	Read(A)

Schedule S_1

swapped ⇨

T_1	T_2
	Read(A)
Read(A)	

Schedule S_2

Here, $S_1 = S_2$ That mean it is non conflict op.

2. T_1: Read(A) T_2: Write(A)

T_1	T_2
Read(A)	
	Write(A)

Schedule S_1

swapped ⇨

T_1	T_2
	Write(A)
Read(A)	

Schedule S_2

Here $S_1 \neq S_2$ That means it is conflict operation

*Non Serializable:-
The non serializable schedule is divided into two types: Recoverable and Non Recoverable.
Recoverable Schedule: Schedules in which transaction commit only after all transaction whose changes they read commit are called recoverable schedule. In other words, 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. Example:- Consider the following schedule involving two transaction T_1 and T_2

T_1	T_2
R(A)	
W(A)	
	W(A)
	R(A)
Commit	
	Commit

This is a recoverable Schedule since T_1 commit before T_2, that make the value read by T_2 correct. There are three types of ~~read~~ recoverable schedule

Recoverable Schedule

- Cascading Schedule
- Cascadeless Schedule
- Strict Schedule

-i- Cascading Schedule

If in a schedule failure of one transaction cause several other dependent transaction to rollback or abort, then such a schedule is called as a cascading schedule or cascading Rollback or cascading Abort

It simply leads to the wastage of CPU Time

T1	T2	T3	T4
R(A)			
W(A)			
	R(A)		
	W(A)		
		R(A)	
		W(A)	
			R(A)
			W(A)
Failure			

Cascading ~~Ro~~ Rollback

* Cascadeless Schedule-

If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a cascadeless schedule.

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

* Strict Schedule

If in a schedule, a transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a strict schedule

In other words,
Strict schedule allow only committed read and write operation
clearly, strict schedule implement more restrictions than cascadeless schedule

T1	T2
W(A)	
Commit/Rollback	
	R(A)/W(A)

• Concurrency Control

In the concurrency control, the multiple transaction can be executed simultaneously
It may affect the transaction result. It is highly important to maintain the order of execution of those transactions

• Problem of concurrency Control

Several problem can occur when concurrent transactions are executed in an uncontrolled manner.
Following are the three problem in concurrency control
Lost updates
Dirty read
Unrepeatable read

PAGE No.
DATE / /

- Lost update Problem

This problem occur when multiple transaction execute concurrently and update from one or more transaction get lost

Transaction T1	Transaction T2
R(A)	
W(A)	
	W(A)
	Commit
Commit	

T1 read the value of A (=10 Says)
T1 update the value of A (=15 Says) in the buffer
T2 does blind write A=25 (write without read) in the buffer
T2 commits
When T1 commits, it writes A=25 in the database

In this example
T1 writes the over written value of X in the database
Thus, updated from T1 get lost

- Dirty Read Problem

Reading the data written by an uncommitted transaction is called as dirty read

This read is called as dirty read because-

- There is always a chance that the uncommited transaction might roll back later
- Thus, uncommitted transaction might make other transaction read a value that does not even exit
- This lead to inconsistency of the database

Transaction T1	Transaction T2
R(A)	
W(A)	
	R(A) //Dirty Read
	W(A)
	Commit
failure	

Here

T1 read the value A (A=10)

T1 update the value A in the buffer (A=15)

T2 read the value of A from the buffer (A=15)

T2 writes the updated the value of A (A=25)

T2 commits

T1 fail in later stages and roll backs

In this example

- T2 reads the dirty value of A written by the uncommitted transaction T1

T1 fail in later stage and roll back

Thus, the value that T2 read now stands to the incorrect

Therefore, database become inconsistent

DATE / /

• Unrepeatable Read Problem

This problem occurs when a transaction get to read unrepeated i.e. different value of the same variable in its different read operation even when it has not updated its value

T1 read the value of X (=10 says)

T2 read the value of X (=10)

T1 updates the value of X (From 10 to 15 say) in the buffer.

T2 again read the value of X (but=15)

Transaction T1	Transaction T2
R(X)	
	R(X)
W(X)	
	R(X)

In this example,

T2 get to read a different value of x in its second reading

T2 wonder how the value of X got changed because according to it, it is running in isolation

* Concurrency Control Protocol

Concurrency control Protocol atomicity, isolation, and serializability of concurrent transactions. The concurrency control Protocol can be divided into three categories:

Lock based Protocol

Time-Stamp Protocol

Validation based Protocol.

PAGE No.
DATE / /

• Locking Based Concurrency control Protocols

A lock is a variable associated with a data item that determines wheather read/write operation can be performed on that data item

Lock-Based Protocol:- In this type of protocol, any transaction cannot read or write data ~~limit~~ until it acquires an appropriate lock on it. There are two states:- ~~It either locked~~ type of lock:

Binary Locks:- A lock on a data item can be in two states; it is either locked or unlocked.

Shared/exclusive:- This type of locking mechanism differentiates the lock based on their uses. If a lock is acquired on a data ~~from~~ item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. ~~Read~~ Read locks are shared because no data value is being changed

There are four type of lock protocol available:
1. Simplistic lock protocol
2. Pre-claiming lock protocol
3. Two-phase locking (2PL)
4. Strict Two-phase locking (Strict-2PL)

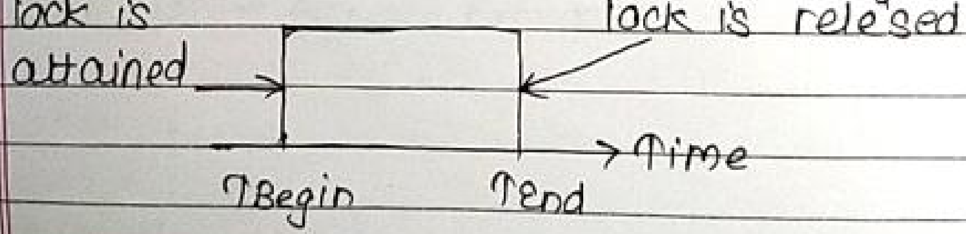
1. Simplistic lock Protocol

It is the simplest way of locking the data while transaction simplistic lock-based protocol allow all the transaction to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.

2. Pre-claiming lock Protocol

- Pre-claiming lock protocol evaluate the transaction to list all the data item on which they need locks. Before initiating an execution of the transaction it requests DBMS for all the lock on all those data items.

If all the locks are granted then this protocol allow the transaction to begin. When the transaction is completed then it releases all the lock.
If all the locks are not granted then this protocol allow the transaction to rolls back and wait until all the lock are granted.



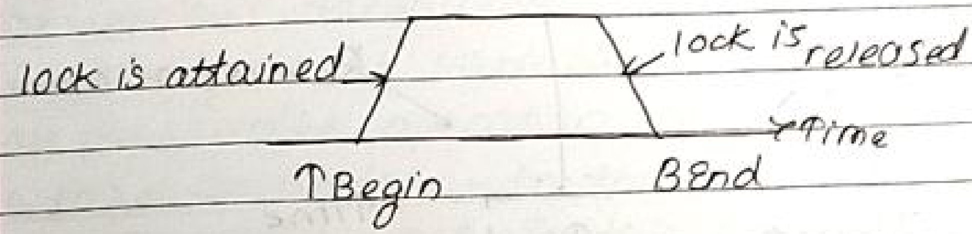
3. Two-phase Locking (2PL)

The two phase locking protocol divides the execution phase of the transaction into three parts

In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires

In the second part, the transaction acquire all the locks. the Third ~~part~~ phase is started as soon as the transaction release its first block

In the third ~~block~~ phase, the transaction cannot demand any new locks, It only release the acquired locks



There are two phases of 2PL

- Growing Phase:- In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released
- Shrinking Phase:- In the shrinking phase, existing lock held by the transaction may be released, but no new lock can be acquired

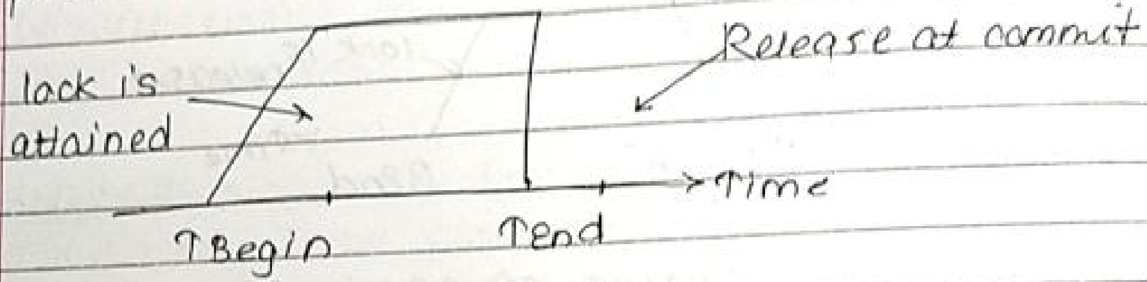
In the below example, if lock conversion is allowed then the following phase can happen:

Upgrading of lock (from S(a) to X(a)) is allowed in growing phase

Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase

* Strict Two-phase locking (Strict-2PL)

- Strict Two-phase locking (Strict-2PL) is similar to 2PL In the first phase of Strict-2PL is similar to 2PL In the
- The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the lock, the transaction continuoues to execute normally
- The only difference betn 2PL and Strict 2PL is that Strict 2PL does not release a lock after using it.
- Strict 2PL wait until the whole transaction to commit, and then it release all the lock at the time
- Strict-2PL protocol does not have shricking phase of lock release.



* Timestamp Ordering Protocol

The timestamp ordering protocol is used in order the transactions based on their Timestamps, The order of transaction is nothing but the ascending order of the transaction creation

The priority the older transaction is higher that's why it execute first. To determine the timestamp of the transaction, this protocal use system time or logical counter

Let's assume there are two transaction T1 and T2 Suppose the transaction T1 has entered the system at 007 time. ~~T1 has the~~ and transaction T2 has entered the system at 009 times. T1 has the higher priority, so ~~th~~ it execute first as it is entered the system first.

The timestamp ordering protocol also maintain the timestamp of last 'read' and 'write' operation on a data.

* Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:-

Read phase: In this phase, the transaction T is read and executed, It is used to read the value of various data item and stores them in temporary local variables. It can perform all the write operation on temporary variable without an update to the actual database.

Validation phase:- In this phase, the temporary variable value will be validated against the ~~temporary~~ actual data to see if it violates the serializability.

Write phase:- If the validation of the transaction is validated then the temporary result are written to the database or system otherwise the transaction is rolled back.

* Data Base Recovery

It is the method of restoring the database to its correct state in the event of a failure at the time of the transaction or after the end of a process

* **Reasons for failure**

- Due to hardware or software errors, the system crashes, which ultemately resulting in loss of main memory
- There can be application software errors, such as logical errors that are accessing the database that can cause one or more transaction to abort or fail
- Natural physical disaster can also occur, such as fires, floods earthquake or power failures
- Carelessness or unintentional destruction of data or directories by operators or users
- Damage or international corruption and hampering of data (using malicious software or files) hardware or software facilities
- failure of main memory, including that database buffers
- failure of the disk copy of that database.

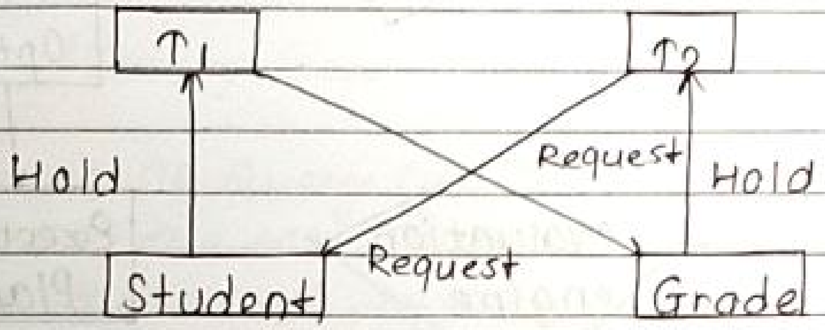
* **Database Recovery in DBMS and its techniques**

Classification of Failure:-

To see whenever the matter has occured, we tend to generalize a failure into numerous classes, as given:-

Deadlock in DBMS

A deadlock is a condition where two or more transaction are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complication in DBMS as no task ever gets finished and is in waiting state forever

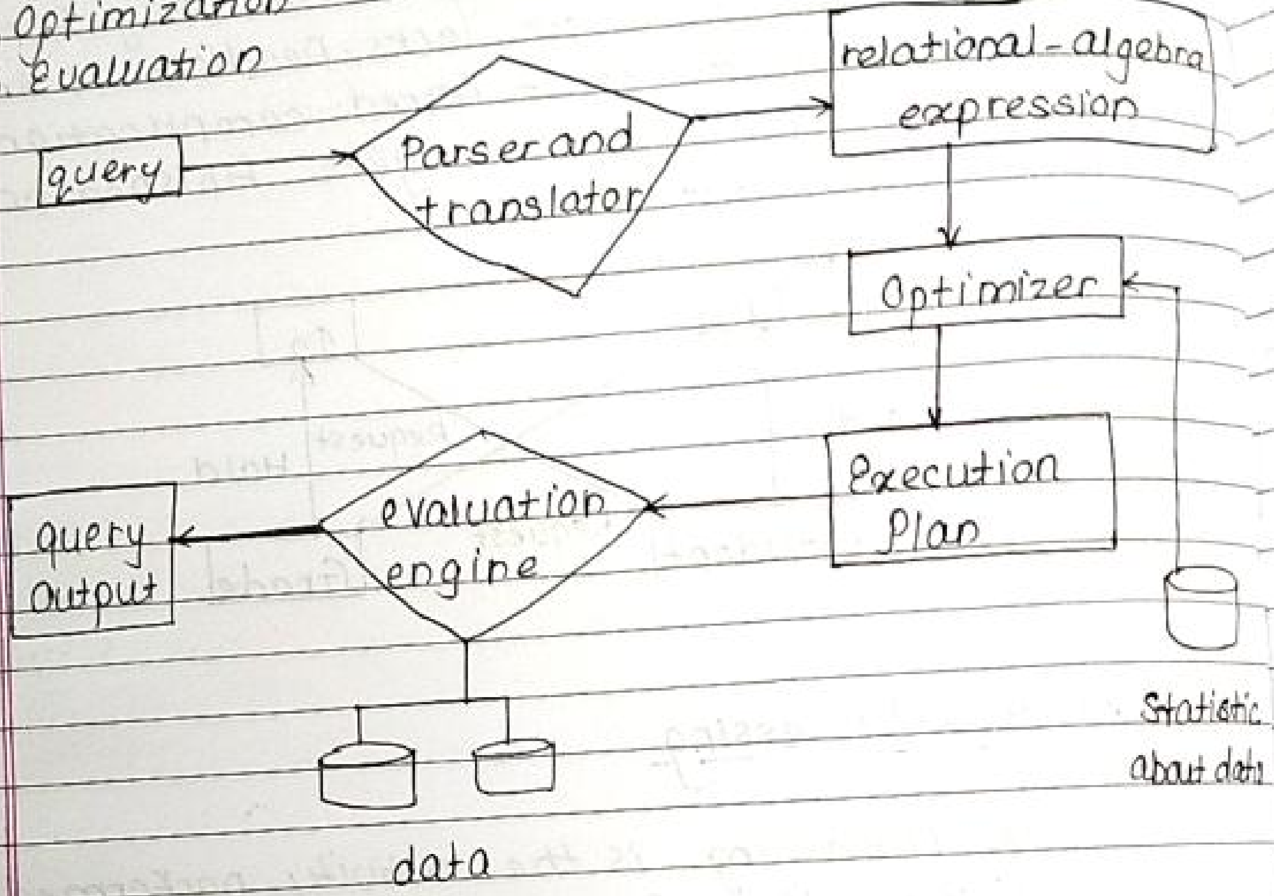


* Query Processing

- Query Processing is the activity performed in extracting data from the database In query processing, it takes various step for fetching the data from the database, The steps involved are:
  - Parsing and Transaction
  - Optimization
  - Evaluation

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Basic Steps in Query Processing (Cont.)

- Parsing and Translation
  - Δ translate the query into its internal form. This is then translate into relational algebra
  - Δ Parser check syntax, verfies relations

- Evaluation
  - Δ The query-execution engine takes a query-evaluation plan, execute that plan, and return the answers to the query

- select emp_name from Employee where salary>1000;

- Thus to make the system understand the user query, it need to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:
  - $\sigma_{salary>10000}(\pi_{salary}(Employee))$
  - $\pi_{salary}(\sigma_{salary>10000}(Employee))$

- Measures Of Query Cost
  - In DBMS, the cost involved in executing a query can be measured by considering the number of different resources that are listed below;
  - The number of disk accesses / the number of disk block transfers / the size of the table
  - Time taken by CPU for executing the query

* Measures Of Query Cost

- Cost is generally measured as total elapsed time for answering query

Query Cost = (number of seek operation
~~Query cost = number~~ X average seek time)
+ (number of blocks read X average transfer time for reading a block) + (number of block written X average transfer time for writing a block)

* Materialization

- In this method, the given expression evaluate one relational operation at a time. Also, each operation is evaluated in an appropriate sequence or order. After evaluating all the operation, the output are materialized in a temporary relation for their subsequent uses

• Pipelining :-

- Pipelining is an alternate method or approach to the materialization method. In pipelining, it enable us to evaluate each relational operation of the expression simultaneously in a pipeline. In this approach, after evaluating one operation, its output is passed on the next operation, and the chain continues till all the ~~next~~ relational operations are evaluated thoroughly

• Query Processing

Query Processing is a feature of many relational database management system and other database such as graph databases. The query optimizer attempt to determine the most ~~effect~~ efficient way to execute a given query by considering the processing query plan.

There are two methods of query optimization
1. cost based optimization (Physical)
2. Heuristic optimization (logical)

PAGE No.
DATE / /

- Cost based Optimization (Physical)

- This is based on the cost of the query. The query can use different path based on indexes, constraints sorting methods, etc. This method mainly uses the statistics like record size, number of record, number of record per block, number of blocks, table size, whether whole table fit in a block, organization of table, uniqueness of column values, size of column, etc.

- Heuristic Optimization (Logical)

- This method is also known as rule based optimization. This is based on the equivalence rule on relational experessions; hence the number of combination of queries get reduces here. Hence the cost of the query too reduces.