UNIT-5

Transaction Management and Concurrency Control: Transaction Concept, A Simple Transaction Model, Storage Structure, ACID Properties, Serializability, Transaction Isolation Levels, Concurrency Control, Lock-Based Protocols, Validation-Based Protocols.

Transaction Concept:

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

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

x's account:

open_account(x)
old_balance = x.balance
new_balance = old_balance - 800
x.balance = new_balance
close_account(x)

y's account:

open_account(y)
old_balance = y.balance
new_balance = old_balance + 800
y.balance = new_balance
close_account(y) .

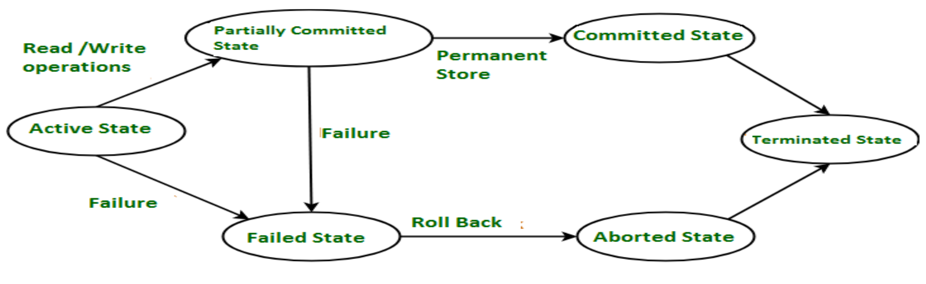
****

Simple Transaction Model:

Simple transaction model is a model of transaction how it must be. It has active, partially committed, failed, aborted, and committed states. Transaction is a several operations that can change the content of the database which is handled by a single program

- **Active**: The initial state where the transaction enters is the active state… i.e, it start the execution of tractions.
- **Partially Committed**: The transaction enter in this state while it is executing read, write or other operations.
- **Committed**: The transaction enters this state after successful completion of the transaction and system checks have issued commit signal. All operations will save per mentally on physical database.
- **Failed**: The transaction goes from partially committed state or active state to failed state when it is discovered that normal execution can no longer proceed or system checks fail.
- **Aborted**: This is the state after the transaction has been rolled back after failure and the database has been restored to its state that was before the transaction begin.

The following state transition diagram depicts the states in the transaction and the low level transaction operations that causes change in states.



Transaction States in DBMS

****

Storage Structure:

1. **Storage devices in DBMS:**

A database system provides an ultimate view of the stored data. However, data in the form of bits, bytes get stored in different storage devices.

Types of Data Storage

For storing the data, there are different types of storage options available. These storage types differ from one another as per the speed and accessibility. There are the following types of storage devices used for storing the data:

- Primary Storage
- Secondary Storage

Primary Storage

It is the primary area that offers quick access to the stored data. We also know the primary storage as volatile storage. It is because this type of memory does not permanently store the data. As soon as the system leads to a power cut or a crash, the data also get lost. Main memory and cache are the types of primary storage.

Main Memory: It is the one that is responsible for operating the data that is available by the storage medium. The main memory handles each instruction of a computer machine.

Cache: It is one of the costly storage media. On the other hand, it is the fastest one. A cache is a tiny storage media which is maintained by the computer hardware usually. While designing the algorithms and query processors for the data structures, the designers keep concern on the cache effects.

Secondary Storage

Secondary storage is also called as Online storage. It is the storage area that allows the user to save and store data permanently. This type of memory does not lose the data due to any power failure or system crash. That's why we also call it non-volatile storage.

There are some commonly described secondary storage media which are available in almost every type of computer system:

Flash Memory: A flash memory stores data in USB (Universal Serial Bus) keys which are further plugged into the USB slots of a computer system. These USB keys help transfer data to a computer system, but it varies in size limits.

Magnetic Disk Storage: This type of storage media is also known as online storage media. A magnetic disk is used for storing the data for a long time. It is capable of storing an entire database. It is the responsibility of the computer system to make availability of the data from a disk to the main memory for further accessing.

2. **File organization in DBMS:**

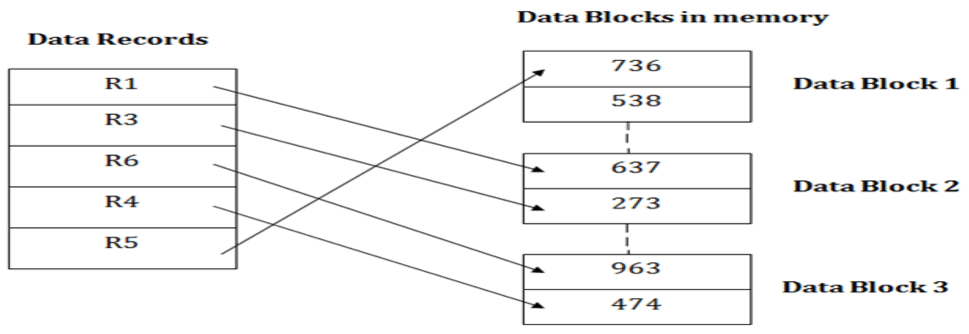
There are several ways to organize files. Based on access or selection, these specific strategies have benefits and drawbacks. In terms of file organization, the programmer selects the approach that best suits his needs.

The following are the types of file organization in DBMS:

1. Heap files organization.
2. Sequential file organization.
3. Indexed file organization.

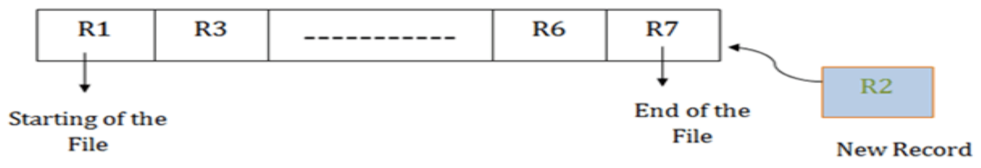
Heap file organization

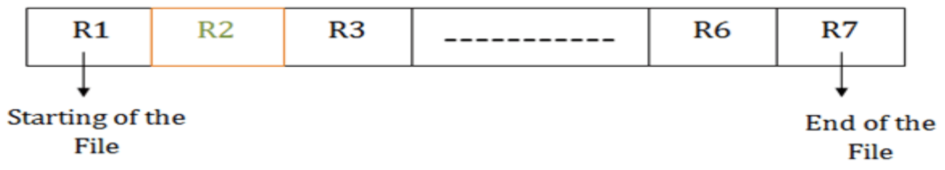
- It is the simplest and most basic type of organization.
- It works with data blocks. In heap file organization, the records are inserted at the file's end.
- When the records are inserted, it doesn't require the sorting and ordering of records. When the data block is full, the new record is stored in some other block.
- This new data block need not to be the very next data block, but it can select any data block in the memory to store new records.
- The heap file is also known as an unordered file. In the file, every record has a unique id, and every page in a file is of the same size.
- It is the DBMS responsibility to store and manage the new records.



<u>Sequential file organization</u>:

- ❖ Every file record contains a data field (attribute) to uniquely identify that record.
- ❖ In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key.
- ❖ Practically, it is not possible to store all the records sequentially in physical form.





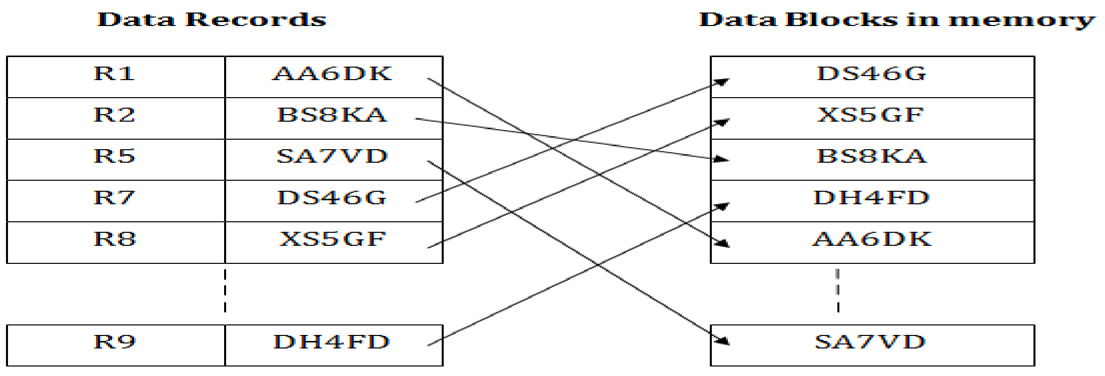
<u>Indexed file organization:</u>

There are two types

1. ISAM.
2. B+ Trees

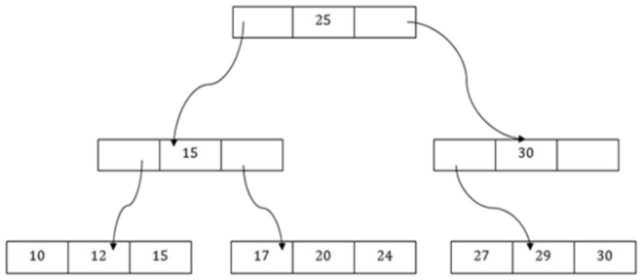
Indexed sequential access method (ISAM):

- ❖ ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key.
- ❖ An index value is generated for each primary key and mapped with the record.
- ❖ This index contains the address of the record in the file.



B+ Trees:

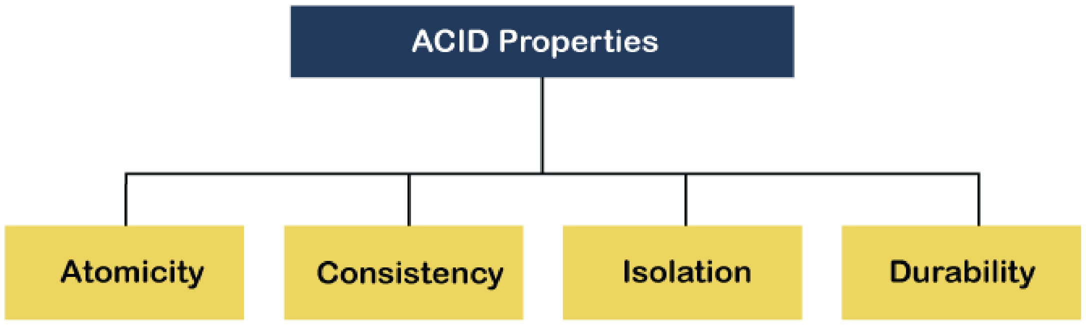
- B+ tree file organization is the advanced method of an indexed sequential access method.
- It uses a tree-like structure to store records in File.
- It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.
- The B+ tree is similar to a binary search tree (BST), but it can have more than two children.
- In this method, all the records are stored only at the leaf node.
- Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.



****

ACID Properties:

A transaction is a logical unit of work that accesses and updates the contents of a database. Read and write operations are used by transactions to access data. To maintain the integrity of the data, there are four properties described in the database management system, which are known as the **ACID** properties. The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties.



1) Atomicity:

The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Tranction-T1

Read(A)	Read(B)
1000	2000
Write(A)	Write(B)
1000-200	2000+200
Read(A)	Read(B)
800	2200

Successful of Transaction.

2) Consistency:

Data is in a consistent state when a transaction starts and when it ends. Before transaction and after transaction data is correct and the structure is stable.

Tranction-T1

Before Transaction A:1000 ,B:2000 Total=1000+2000=3000	

Read(A)
1000
Write(A)
1000-200
Read(A)
800
	Read(B)
	2000
	Write(B)
	2000+200
	Read(B)
	2200

After Transaction A:800 ,B:2200 Total=800+2200=3000	

3) Isolation:

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Tranction-T1

Read(A)
1000
Write(A)
1000-200
Read(A)
800
	Read(B)
	2000
	Write(B)
	2000+200
	Read(B)
	2200

Tranction-T2

Read(B)
2200
Write(B)
2200+500
Read(B)
2700
	Read(C)
	1500
	Write(C)
	1500-500
	Read(C)
	1000

4) Durability

Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.

****

<u>Serializability:</u>

Serializability is related to schedules and transactions. Schedule is a set of transactions, and a transaction is a set of instructions used to perform any logical operations in terms of databases. i.e assign a sequential number to each and every operation in a transaction.

Schedules in DBMS are of two types:

Serial Schedule - A schedule in which only one transaction is executed at a time, i.e., one transaction is executed completely before starting another transaction.

Non-serial Schedule - A schedule in which the transactions are interleaving or interchanging. There are several transactions executing simultaneously as they are being used in performing real-world database operations.

Types of serializability

There are two types of serializability −

View serializability:

A schedule is view-serializability if it is viewed equivalent to a serial schedule.

The rules it follows are as follows −

- T1 is reading the initial value of A, then T2 also reads the initial value of A.
- T1 is the reading value written by T2, then T2 also reads the value written by T1.
- T1 is writing the final value, and then T2 also has the write operation as the final value.

Conflict serializability:

It orders any conflicting operations in the same way as some serial execution. A pair of operations is said to conflict if they operate on the same data item and one of them is a write operation.

That means

- Readi(x) readj(x) - non conflict   read-read operation
- Readi(x) writej(x) - conflict   read-write operation.
- Writei(x) readj(x) - conflict   write-read operation.
- Writei(x) writej(x) - conflict   write-write operation.

****

<u>Transaction Isolation Levels:</u>

Transaction isolation levels specify what data is visible to statements within a transaction. These levels directly impact the level of concurrent access by defining what interaction is possible between transactions against the same target data source.
The different types of database anomalies are described as follows:

1. **Dirty** reads occur when:
   1. Transaction A inserts a row into a table.
   2. Transaction B reads the new row.
   3. Transaction A rolls back..
2. **Non-repeatable** reads occur when:
   1. Transaction A reads a row.
   2. Transaction B changes the row.
   3. Transaction A reads the same row a second time and gets the new results.
3. **Phantom Read Problem**
   1. In the phantom read problem, data is read through two different read operations in the same transaction.
   2. In the first read operation, a value of the data is obtained but in the second operation, an error is obtained saying the data does not exist.

Types of Isolation Level:

Based on the above different phenomena, SQL defines into four isolation levels.

1. **Read Uncommitted:** It is the lowest in the isolation level. At this level, one transaction cannot read the changes made by the other transactions, so it allows dirty reads. At this level, the transaction is not isolated from each other.
2. **Read Committed:** It provides a guarantee to each data that these data gets committed when these are read by any transaction. So that it does not allow dirty Read. The transaction holds the Read or writes action so that it prevents the data from reading or written by any other transaction.
3. **Repeatable Read:** This is the most restrictive isolation level. The transaction holds read locks on all rows it references and writes locks on referenced rows for updates and deletes actions. Since other transactions cannot read, update or delete these rows, consequently, it avoids non-repeatable Read.
4. **Serializable:** This is the highest isolation level. A serializable execution is guaranteed to be serializable. Serializable execution is defined to be an execution of operations in which concurrently executing transactions appears to be serially executing.

****

Concurrency Control:

Concurrency control concept comes under the Transaction in database management system (DBMS). It is a procedure in DBMS which helps us for the management of two simultaneous processes to execute without conflicts between each other, these conflicts occur in multi user systems.Concurrency can simply be said to be executing multiple transactions at a time. It is required to increase time efficiency. If many transactions try to access the same data, then inconsistency arises. Concurrency control required to maintain consistency data.

For example, if we take ATM machines and do not use concurrency, multiple persons cannot draw money at a time in different places. This is where we need concurrency.

Concurrency control techniques:

The concurrency control techniques are as follows −

- **Locking**

  Lock guaranties exclusive use of data items to a current transaction. It first accesses the data items by acquiring a lock, after completion of the transaction it releases the lock.

  Types of Locks

  The types of locks are as follows −

  Shared Lock [Transaction can read only the data item values]
  Exclusive Lock [Used for both read and write data item values]
- **Time Stamping**

  Time stamp is a unique identifier created by DBMS that indicates relative starting time of a transaction. Whatever transaction we are doing it stores the starting time of the transaction and denotes a specific time. This can be generated using a system clock or logical counter. This can be started whenever a transaction is started. Here, the logical counter is incremented after a new timestamp has been assigned.
- **Optimistic**

  It is based on the assumption that conflict is rare and it is more efficient to allow transactions to proceed without imposing delays to ensure serializability.

****

Lock-Based Protocols:

A Lock-based protocol in DBMS as a mechanism that is responsible for preventing a transaction from reading or writing data until the necessary lock is obtained.

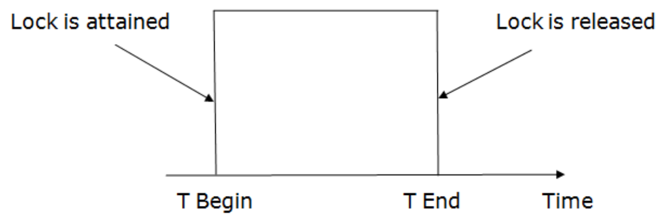
There are four types of lock protocols available:

- ❖ **Simplistic lock protocol:**

  It is the simplest way of locking the data while transaction. Simplistic lock-based protocols allow all the transactions 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.

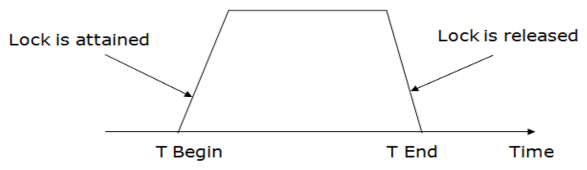
- ❖ **Pre-claiming Lock Protocol:**

  Pre-claiming Lock Protocols evaluate the transaction to list all the data items 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 allows the transaction to begin. When the transaction is completed then it releases all the lock.



- ❖ **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 acquires all the locks. The third phase is started as soon as the transaction releases its first lock.



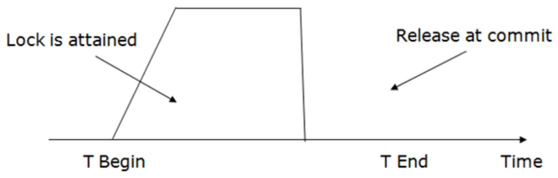
There are two phases of 2PL:

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 locks can be acquired.

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

The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally. The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.Strict-2PL waits until the whole transaction to commit and save all the transactions in physical database and then it releases all the locks at a time.



****

Validation-Based Protocols:

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

Three phases of Validation based Protocol

1. **Read phase:** In this phase, a transaction reads the value of data items from database and store their values into the temporary local variables. Transaction then starts executing but it doesn't update the data items in the database, instead it performs all the operations on temporary local variables.
2. **Validation phase:** In this phase, a validation check is done on the temporary variables to see if it violates the rules of serializability.
3. **Write phase:** This is the final phase of validation based protocol. In this phase, if the validation of the transaction is successful then the values of temporary local variables are written to the database and the transaction is committed. If the validation is failed in second phase then the updates are discarded and transaction is slowed down to be restarted later.

****