ADVANCED DATABASE SYSTEM

Chapter:3 –Transaction Processing Concept

Compiled by: Dureso.M

OUTLINES TO BE COVERED

- In this chapter we are going to cover the following basic concept of transaction processing

✓Introduction to the concept of transaction

✓Transaction and System Concepts

✓Desirable Properties of Transactions

✓Characterizing Schedules based on Recoverability

✓ Characterizing Schedules based on Serializability

✓Transaction Support in SQL(Lab Exercise)

INTRODUCTION TO TRANSACTION PROCESSING

- **Transaction** is a mechanism for applying the desired modifications / operations to a database.
- Any changes made in real time to a database are called **transactions**.
- In database field, a **transaction** is a group of **logical operations** that must all **succeed** or **fail** as a group.
- Systems dedicated for supporting such operations are known as **transaction processing systems(TPS)**.

CONT...

TPS- are systems with large databases and hundreds of **concurrent users** that are executing database transactions.

→**Examples** of such systems may include **reservations system**, **banking system**, **credit card processing system**, **supermarket check generating system**, and other similar systems.

Transaction:

- Logical unit of database processing that includes one or more access operations (read -retrieval, write - insert or update, delete).
- This transaction contains the **steps** involved in the business operation.
- Transactions can be **started, attempted,** then **committed** or **aborted** via **data manipulation commands/ Transaction Control Languages** of SQL.

CONT...

We can have one of the following two outcomes for any transaction:

- **Success -** transaction commits and database reaches a new consistent state

  - Committed transaction cannot be aborted or rolled back.

  - **How do you discard a committed transaction?**

- **Failure** - transaction aborts, and database must be restored to consistent state before it started.

  -Such a transaction is **rolled back** or **undone**.

  -Aborted transaction that is rolled back can be **restarted** later.

CONT...

❖ States of Transactions

A transaction in a database can be in one of the following states:

1. Active state: In this state, the transaction is being executed. This is the initial state of every database transaction.

2. Partially Committed state: When a transaction executes its final operation, it is said to be in a partially committed state.

3. Failed state: A transaction is said to be in a **failed state** if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

CONT…

States of Transactions…

4. Aborted state: If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called **aborted**. The **database recovery module** can select one of the two operations after a transaction aborts:

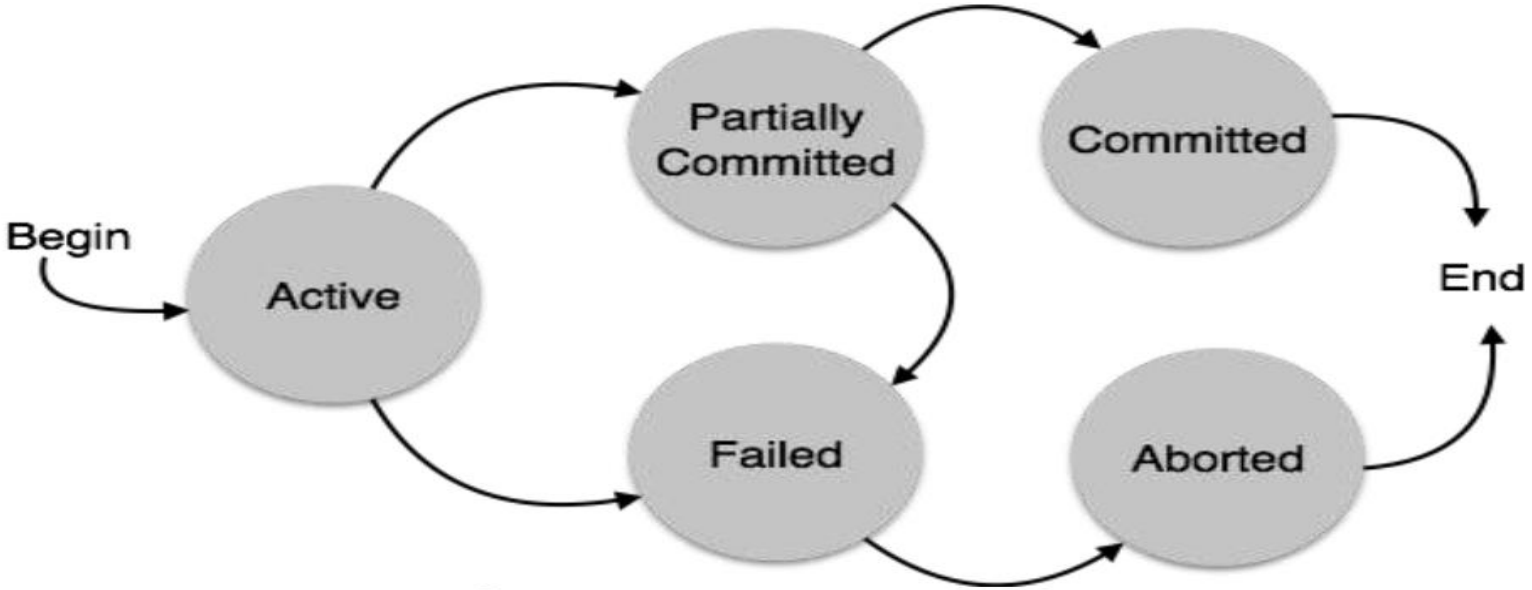
- **Re-start the transaction**
- **Kill the transaction**

INTRODUCTION TO TRANSACTION PROCESSING…

States of Transactions…

5. Committed state: If a transaction executes all its operations successfully, it is said to be **committed**. All its effects are now permanently published on the database system.

❖Here is diagram representing different states in database transaction:



[*Image: Transaction States*]

CONT...

→A transaction can end in three possible ways.

1. **Successful Termination:** when a transaction completes the execution of all operations in it and reaches the COMMITED state.
2. **Suicidal Termination:** when the transaction detects an error during its processing and decide to abrupt itself before the end of the transaction and perform a **ROLL BACK operation(Restart transaction).**
3. **Murderous Termination:** When the DBMS or the system force the execution to abort for any reason. And hence, rolled back(**Kill transaction**)

TRANSACTION AND SYSTEM CONCEPTS.

Single-User vs Multiuser Systems

❖ One criterion for classifying a database system is according to the **number of users** who can use the system **concurrently**—that is, **at the same time.**

DBMS Classifies system in to two based on number of users who are accessing the system concurrently

1. **Single user system :-** if at most one user using the system at a time.
2. **Multi-user system: -** if many users can use the system—and hence access the database concurrently.

Most of today's DBMSs are multiuser. For **example,** an **airline reservations system** is used by hundreds of travel agents and reservation clerks concurrently.

CONT...

Single-User vs Multiuser Systems

- In a multiuser DBMS, the stored data items are the **primary resources** that may be accessed concurrently by **interactive users** or **application programs**, which are constantly **retrieving** information from and **modifying** the database.
- A single transaction might require several queries, each reading and/or writing information in the database. When this happens it is usually important to be sure that the database is not left with only some of the queries carried out.

----> **example,** when doing a **money transfer**, if the money was **debited** from one **account**, it is important that it also be **credited** to the **depositing account**.

-Transactions should not interfere with each other which leads to inconsistent results.

CONT...

- Basic operations done on the database during transactions are **read** and **write**
  - **read_item(X)**: Reads a database item named X into a program variable. To simplify our notation, we assume that the program variable is also named X.
  - **write_item(X)**: Writes the value of program variable X into the database item named X.

Read and Write Operations:

- Basic unit of data transfer from the disk to the computer main memory is one block. In general, a data item (what is read or written) will be the field of some record in the database, although it may be a larger unit such as a record or even a whole block.

CONT...

❖ **read_item(X) command includes the following steps:**

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
3. Copy item X from the buffer to the program variable named X.

❖ **write_item(X) command includes the following steps:**

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
3. Copy item X from the program variable named X into its correct location in the buffer.
4. Store the updated block from the buffer back to disk (either immediately or at some later point in time).

CONT...

→Ways of Executing Transaction

- In a database system many transactions are executed. Basically there are two ways of executing a set of transactions:

(a) Serial Execution: In a serial execution transactions are executed strictly serially.

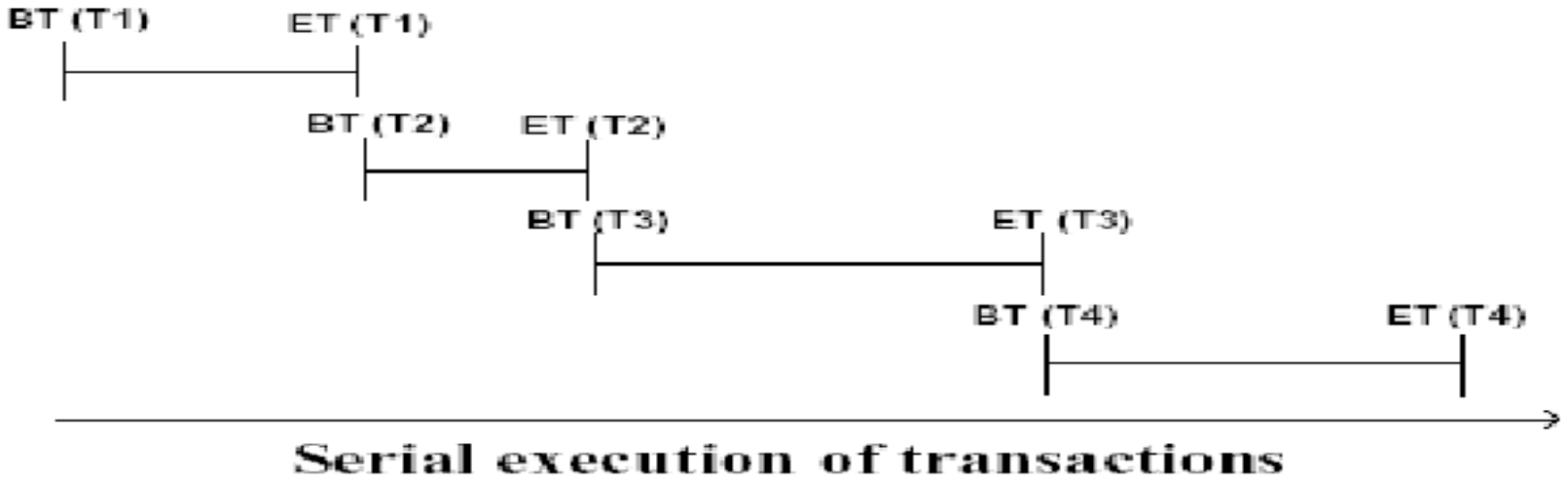
Thus, Transaction Ti completes and writes its results to the database then only the next transaction Tj is scheduled for execution.

This means at one time there is only one transaction is being executed in the system. The data is not shared between transactions at one specific time.

In **Serial** transaction execution, one transaction being executed does not **interfere** the execution of any other transaction.

CONT…

→**Ways of Executing Transaction….**



Serial execution of transactions

❖ **Good things about serial execution**

- ✓ Correct execution, i.e., if the input is correct then output will be correct.
- ✓ Fast execution, since all the resources are available for the active transaction.

❖ The worst thing about serial execution is very inefficient resource utilization. i.e. reduced parallelism

CONT...

→Ways of Executing Transaction

Example of a serial execution

Suppose data items X = 10, Y = 6, and N =1 and T_1 and T_2 are transactions.

T_1	T_2
read (X)	*read (X)*
X := X+N	*X := X+N*
write (X)	*write (X)*
read (Y)	
Y := Y+N	
write (Y)	

We execute this transaction serially as follows:

CONT...

→**When the above transaction is executed serially**

Time	T_1	T_2
↓	*read (X) {X = 10}*	
	X := X+N {X = 11}	
	write (X) {X = 11}	
	read (Y) {Y = 6}	
	Y := Y+N {Y = 7}	
	write (Y) {Y = 7}	
		read (X) {X = 11}
		X := X+N {X = 12}
		write (X)

❖ **Final values of X, Y at the end of T1 and T2: X = 12 and Y = 7**

CONT...

→Ways of Executing Transaction

b) Concurrent execution: is the reverse of serially executable transactions, in this scheme the individual operations of transactions, i.e., reads and writes are interleaved in some order.

The following example illustrate concurrent execution of transaction

Time	T_1	T_2
↓	*read (X) {X = 10}*	
		read (X) {X = 10}
	X := X+N {X = 11}	
		X := X+N {X = 11}
	write (X) {X = 11}	
		write (X) (X=11)
	read (Y) {Y = 6}	
	Y := Y+N {Y = 7}	
	write (Y) {Y = 7}	

CONT...

Final values at the end of T1 and T2: X = 11, and Y = 7.

- ✓This improves resource utilization, unfortunately gives incorrect result.
- ✓The correct value of X is 12 but in concurrent execution X =11, which is incorrect.
- ✓The reason for this error is incorrect sharing of X by T1 and T2.
- ✓In serial execution T2 read the value of X written by T1 (i.e., 11) but in concurrent execution T2 read the same value of X (i.e., 10) as T1 did and the update made by T1 was overwritten by T2's update.

CONT...

→Problem associated with concurrent transaction

- **The Lost Update Problem**
  - This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.
- **The Temporary Update (or Dirty Read) Problem**
  - This occurs when one transaction updates a database item and then the transaction fails for some reason.
  - The updated item is accessed by another transaction before it is changed back to its original value.
- **The Incorrect Summary Problem**
  - If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.

DESIRABLE PROPERTY OF TRANSACTION

→**A transaction** is expected to exhibit some basic features or properties to be considered as a **valid transaction.**

→**A transaction** in a database system must maintain **Atomicity**, **Consistency**, **Isolation**, and **Durability**; commonly known as **ACID** properties in order to ensure accuracy, completeness, and data integrity with the database.

1) **Atomicity-** Is **All** or **None** property

✓Every transaction should be considered as an atomic process which can not be sub divided into small tasks. Due to this property, just like an atom which exists or does not exist, a transaction has only two states. **Done** or **Never Started.**

- **Done -** a transaction must complete successfully and its effect should be visible in the database.
- **Never Started -** If a transaction fails during execution then all its modifications must be undone to bring back the database to the last consistent state, i.e., remove the effect of failed transaction. There is no the state between **done** and **never** started.

CONT...

- ✓ The atomicity property requires that we execute a transaction to **completion.** It is the responsibility of the **transaction recovery subsystem** of a DBMS to ensure atomicity.
- ✓ If a transaction fails to complete for some reason, such as a **system crash** in the midst of transaction execution, the recovery technique must **undo** any effects of the transaction on the database.

2) Consistency/ behaving similarly: The preservation of consistency is generally considered to be the responsibility of the **programmers** who write the database programs or of the DBMS module that enforces integrity constraints.

- ✓ A database program should be written in a way that guarantees if the database is in a consistent state before executing the transaction, it will be in a consistent state after the complete execution of the transaction, assuming that no **interference** with other transactions occurs.

CONT...

3) **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** is enforced by the concurrency control subsystem of the DBMS. If every transaction does not make its updates visible to other transactions until it is committed, one form of isolation is enforced that solves the **temporary update** problem.

4) **Durability:** the durability property of the transaction states that the effect of a completed transaction must persist in the database, i.e., its updates must be available to other transaction immediately after the end of its execution, and is should not be affected due to failures after the completion of the transaction.

✓ **Finally,** the durability property is the responsibility of the recovery subsystem of the DBMS.

WHY RECOVERY IS SO NEEDED?

What causes a Transaction to fail?

1. A computer failure (system crash):

A hardware or software error occurs in the computer system during transaction execution. If the hardware crashes, the contents of the computer's internal memory may be lost.

2. A transaction or system error:

Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error. In addition, the user may interrupt the transaction during its execution.

3. Local errors or exception conditions detected by the transaction:

Certain conditions necessitate cancellation of the transaction. For example, data for the transaction may not be found. A condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal from that account, to be canceled.

CONT...

4. Concurrency control enforcement:

The concurrency control method may decide to abort the transaction, to be **restarted later**, because it violates **serializability** or because several transactions are in a state of **deadlock.**

5. 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.

6. Physical problems and catastrophes:

This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft and etc.

SCHEDULES BASED ON RECOVERABILITY

- ✓In any **transaction processing system,** if concurrent processing is implemented, there will be concept called **schedule** having or determining the **execution sequence of operations** in different transactions.
- ✓**Schedule:** time-ordered sequence of the important actions taken by one or more transitions. Schedule represents the order in which instructions are executed in the system in chronological ordering.
- ✓The scheduler component of a DBMS must ensure that the individual steps of different transactions preserve consistency.

CONT...

- ✓ For some schedules it is easy to **recover** transaction from failures, whereas for other schedules the recovery process can be quite involved. Hence, it is important to characterize the types of schedules for which recovery is possible, as well as those for which recovery is relatively simple.
- ✓ These characterizations do not actually provide the **recovery algorithm** but instead only attempt to theoretically characterize the different types of schedules.
- ✓ First, we would like to ensure that, once a transaction **T** is **committed**, it should never be necessary to **roll back T**. The schedules that theoretically meet this criterion are called **recoverable schedules** and those that do not are called **non-recoverable,** and hence should not be permitted.

CONT...

Schedules classified on the base recoverability as follows:

1. **Recoverable schedule**:
   - One where no transaction needs to be rolled back.
   - A schedule S is recoverable if no transaction T in S commits until all transactions T' that have written an item that T reads have committed.
2. **Cascadeless schedule:**
   - One where every transaction reads only the items that are written by committed transactions.

CONT...

3) Schedules requiring cascaded rollback:

- A schedule in which uncommitted transactions that read an item from a failed transaction must be rolled back.

4) Strict Schedules:

- A schedule in which a transaction can **neither** read nor **write** an item X until the last transaction that wrote X has committed.

SCHEDULE BASED ON SERIALIZABILITY

✓The schedules based on serializability can be of two types:-

- ✓Serial Schedule and
- ✓Non serial schedules

1) **Serial Schedule:** a schedule where the operations of each transaction are executed consecutively without any interleaved operations from other transactions.

2) **Non-serial Schedule:** Schedule where operations from a set of concurrent transactions are interleaved.

CONT…

- ✓ The objective of **serializability** is to find **non-serial** schedules that allow to execute transactions **concurrently** without interfering with one another.
- ✓ In other words, if we find that **non-serial** schedules that are equivalent to some serial schedule. Such a schedule is called **serializable.**
- ✓ If a **non-serial** schedule can come up with a result equivalent to some **serial schedule**, we call the schedule **Serializable**

SUMMARY

To sum-up we have discussed the following basic concept:

- Transaction and System Concepts
- Desirable Properties of Transactions
- Characterizing Schedules based on Recoverability
- Characterizing Schedules based on Serializability

- **Transaction support in SQL(Lab Exercise)**

“IF YOU HAVE ANY QUESTION”



“YOU ARE WEL-COME”


Thank You!