MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 7

Transaction Processing

Table of Contents

1. INTRODUCTION

In the previous unit, you studied the concept of adaptive query processing and query evaluation. You studied query processing mechanism, eddy architecture, properties of query processing algorithms, synchronisation barriers in query processing, etc.

A series of operations comprising database operations that is atomic with regard to concurrency and recovery is called transaction. As a transaction can include various steps, every step in the transaction must be performed well to make the transaction successful. If any step of the transaction does not succeed, then the entire transaction is bound to fail.

Several common techniques and approaches associated with transaction processing are explained here. Our aim is to provide a basic framework for understanding transaction processing. We focus primarily on advantages and disadvantages of transaction processing system, online transaction processing system, serialisability and recoverability, view serialisability, resolving deadlock, distributed locking and transaction management in multi-database system. The unit is concluded with the analysis of long duration transaction and high-performance transaction system.

1.1 Objectives

After studying this unit, you should be able to:

- *Discuss the concept of transaction processing*
- *List various advantages and disadvantages of transaction processing system*
- *Discuss online transaction processing system*
- *Explain the concept of serialisability and recoverability*
- *Recognise and explain different methods used for resolving deadlock*
- *Explain distributed locking*
- *Summarise the concept of transaction management in multi-database system*
- *Explain long duration transaction*
- *Discuss high performance transaction system*

2. TRANSACTION PROCESSING: AN INTRODUCTION

As you know, generally, one or more database operations are grouped into three transactions, which is a unit of work that must be performed atomically and in separation from other transactions. To remind you again, a transaction is an execution of a program that satisfies the four basic properties: Atomicity, Consistency, Isolation, Durability (known as ACID properties). Transactions are meant for: Concurrency control and Recovery of data.

Additionally, a DBMS provides the assurance of stability that the work of a finished transaction will never be vanished. Thus the transaction manager obtains transaction commands from an application, which inform the transaction manager when transactions start and end, in addition to information regarding the expectations of the application. Figure 7.1 demonstrate various features of transaction processing as an example.

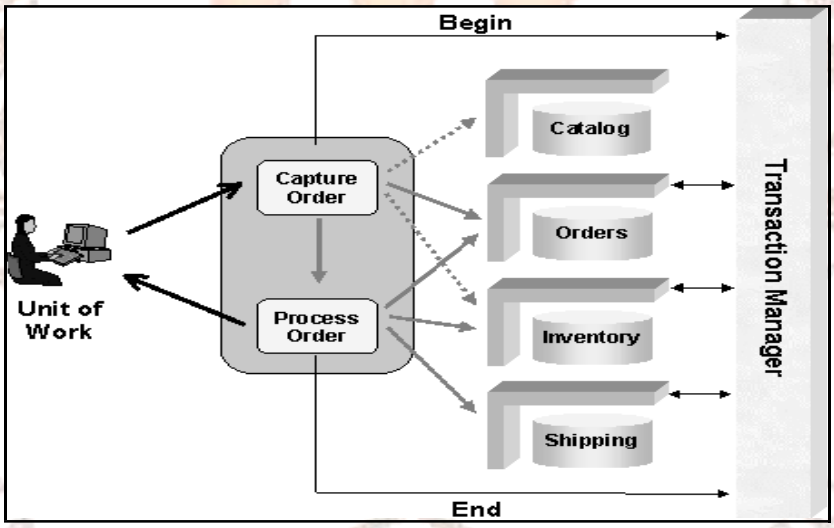

Fig 7.1: An Example of Transaction Processing Architecture

For instance, some may not need atomicity. The following tasks are carried out by the transaction manager.

1. ***Logging:*** To guarantee stability, each change in the database is logged individually on disk. The log manager considers numerous policies designed to guarantee that regardless of when a system failure or ”crash" takes place, a recovery manager can check the log of changes and database is restored to some reliable state.

2. ***Concurrency control:*** Transactions must emerge to perform in isolation. However, in most of the systems, the binary transactions are actually executing at once. Therefore, the scheduler must guarantee that the individual actions of numerous transactions are executed in such a way that the total outcome is the same as if the transactions had actually executed in their entirety, one by one.
3. ***Deadlock resolution:*** As transactions struggle for getting resources via the locks that the scheduler provides, they can be in a condition where none can continue since each requires something another transaction comprises. The transaction manager has the liability to interfere and terminate ("rollback" or "abort") one or more transactions to allow the others to continue.

SELF-ASSESSMENT QUESTIONS – 1

1. To provide which type of assurance, each change in the database is logged individually on disk?
   a) Continuity
   b) Stability
   c) Concurrency
   d) Speed
2. To whom the transaction commands inform about the transactions start and end?
   a) Transaction commander
   b) Transaction coordinator
   c) Transaction manager
   d) Transaction master

3. ADVANTAGES AND DISADVANTAGES OF TRANSACTION PROCESSING SYSTEM

Transaction processing system consists of various advantages and disadvantages which you are going to study now.

3.1 Advantages of Transaction Processing System

The advantages of transaction processing are:

1. Each business firm needs a system for the collection, accumulating and recovering of data and statistics, so that it can function competently. A transaction processing system will fulfil this requirement. Every single transaction is managed and monitored by means of a transaction processing system, in order that the system will identify if entered data is valid. Once the gathered information clears the test, it will then be accumulated and generated in the processing system.
2. The process of monitoring all transactions can be made simpler by means of an organised transaction processing system. It'll enormously save a firm's time, energy, money and attempt. Furthermore, all the entered data will be kept protected and all transactions performed will be monitored and recorded in a correct manner. All records will be maintained in an organised and protected manner. Only approved people have access to its functions.
3. By means of a consistent transaction processing system, any firm will successfully please its consumers and get more satisfied consumers. Consumers rely on the consistency of a company. In future, they will stand by a firm with whom that they can trust their earnings, money and personal information.

3.2 Disadvantages of Transaction Processing System

The main disadvantages of transaction processing systems are:

1. There is the need to manage hundreds and thousands of simultaneous consumers.
2. There is the need to permit several consumers to work on the same set of data, with instant updating.
3. There is the need to manage faults in a protected and consistent way: Transaction Systems generally manage faults in a secure and reliable manner, but there are some

faults that you cannot evade, for example, network faults or deadlocks of databases. Thus, there should be a method which can be used to manage them when they take place. It is not possible to just terminate an existing process.

SELF-ASSESSMENT QUESTIONS – 2

3. _____ transaction processing system can make the process of monitoring each and every transaction simpler.
4. In transaction processing system, it is not required to manage errors in a secured and consistent manner. (True/ False)

4. ONLINE TRANSACTION PROCESSING SYSTEM

Online transaction processing is considered as interactive in which every transaction is processed as it takes place. In Online Transaction Processing (OLTP) system, you can endlessly interact with the system by a computer or a terminal. The Air-line reservation, Railway reservation system, the Banking ATM machine, the Library application, etc. are some of the examples of online transaction processing system.

In these types of systems, you are required to enter pre-defined inputs such as flight number, train number, date of journey, amount to withdraw, etc.

According to these pre-defined inputs, the system generates pre-defined outputs such as the confirmed tickets, or non availability of ticket, etc.

The problem with online transaction processing (OLTP) system is the high costs related with the required security & fault acceptance traits. A data is entered by a person for a system transaction, where it is processed and the output is obtained before entering the next input.

When you use online entry with postponed processing, then data is input as the transaction takes place and is stored online, but files are not updated. Files are updated afterwards in batch. For example, orders received over the phone may be accepted by the system, but the orders are not processed until a slow time, like at night.

Figure 7.2 shows an example of online transaction processing. In it a customer uses an ATM, which demonstrates an easy interface for different functions such as cash withdrawal, account balance query, bill payment, transfer, or cash advance. In the same network, an employee in the branch office executes operations such as fund applications and consulting.

In the bank head office, a business analyst tunes up business deal for better functioning. Other employees execute day-to-day functions such as customer relationship management, budget planning, and stock control. In the Figure, you can see that all requests are addressed to the mainframe for processing.

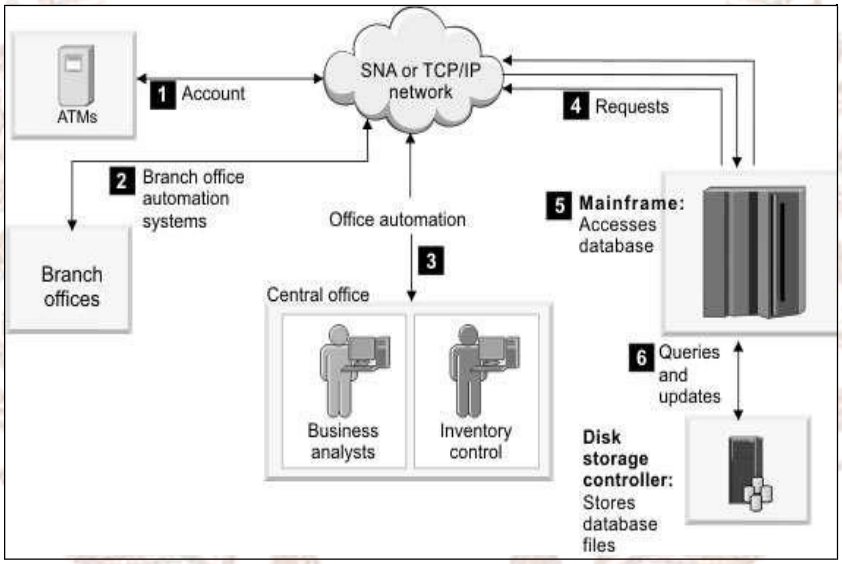

Fig 7.2: Online Transaction Processing Framework

SELF-ASSESSMENT QUESTIONS – 3

5. In online transaction processing systems, which type of input user must provide?
6. In Online transaction processing (OLTP) system, the consumer cannot continuously interact with the system by a computer or a terminal. (True/False)

5. SERIALISABILITY AND RECOVERABILITY

In transaction processing, serialisability is considered as the property of a serialisable schedule. A (possibly concurrent) schedule S is serializable if it is equivalent to a serial schedule S'. That is, S has the same result database state as S'.

Serial Schedule is a schedule where transactions are done in order. Transactions are scheduled one by one. Upon completion of one transaction another transaction is scheduled whereas schedules in which transactions are not executed concurrently is called Non-Serial schedule.

A schedule S of n transactions is serializable if it is equivalent to some serial schedule of the same n transactions. There are n! possible serial schedules of n transactions and many more possible non-serial schedules.

Serialisability is the main criterion for the accuracy of simultaneous transactions executions and a main objective for concurrency control. In the context of data storage and communication, serialisability is the process of converting a data structure into a format that can be stored.

If every transaction is accurate by itself, then any sequential execution of these transactions is accurate.

Notable examples are banking transactions that debit and credit accounts with money. If the associated schedules are not serialisable, then the total sum of money may not be maintained. Money could vanish, or be produced from nowhere. This does not take place if serialisability is sustained.

In systems where transactions can terminate, serialisability by itself is not adequate for accuracy. Schedules are also required to acquire the property of Recoverability. Recoverability signifies that data is not read by the committed transactions where the data is written by terminated transactions.

While serialisability can be negotiated in various applications, negotiating recoverability always infringes the database's integrity.

If a system crash takes place, the actions of the committed transactions can be rebuilt on the disk copy of the database. However, it is to be noted here that only the committed transactions can be re built. No activity is carried out by the logging system to assist serialisability. A database state is simply rebuilt even if it is the consequence of non-serialisable schedule of actions.

Commercial database systems do not always endure on serialisability and serialisability is enforced in some systems, when the user makes a demand.

5.1 Cascading Rollback

On the failing of the transaction, the system is required to return to the situation that it was in before the transaction was initiated. We call this as rollback. That is, on the failing of the transaction, that changes that had been made are returned to their previous values is known as "rolled back".

When a data is written by a non-committed transaction it is stated to be 'dirty'. The dirty data can be viewed either in the buffers, or on disk, or both. They both can lead to problems.

A cascading rollback can be carried out if dirty data can be procured to transactions. . On termination of transaction T, any transaction that has read the data written by it should be found out and terminated. Further other transactions that have read data written by a terminated transaction should also be recursively terminated.

When a transaction T aborts, you must determine which transactions have read data written by T, abort them, and recursively abort any transactions that have read data written by an aborted transaction. In other words, you must find each transaction U that read dirty data written by T, abort U, find any transaction V that read dirty data from U, abort V, and so on.

A log can be used to end the effect of a terminated transaction if it is one of the kinds that offer previous values for e.g. undo or undo/redo. The data can also be fixed from the copy in the disk if the effect of the dirty data has not shifted to disk.

5.2 Recoverable Schedules

If you want any of the logging methods to permit recovery, the set of transactions that are considered as committed after recovery must be reliable.

Specifically, if a transaction T_1 is, after recovery considered as committed, and T_1 utilised a value written by T_2, then T_2 must also remain committed, after recovery. Therefore, we define a recoverable schedule as follows:

"If a transaction commits only after every transaction from which it has read commits, a schedule is said to be recoverable."

5.3 Managing Rollbacks Using Locking

The section now analyses the management of cascading rollbacks in a lock- based scheduler. Strict locking is a simple method used to provide assurance against cascading rollbacks.

Till the time the transaction is either committed or aborted, any write locks or other locks such as increment locks which permit values to be changed should not be discharged by the transaction.

A transaction cannot read dirty data in a recoverable schedule as it remains locked until the transaction commits. The problem of fixing the data in buffers when a transaction terminates however still persists since the changes need to have the effect terminated.

SELF-ASSESSMENT QUESTIONS – 4

7. When a data is written by a non-committed transaction, what is it called?
   a) Dirty
   b) Strict
   c) Rolled back
   d) Serialisable
8. Write locks should not be released by a transaction in case of strict locking, until the transaction is either committed or aborted. (True/ False)

Activity 1

Surf the Internet and find out how serialisability and recoverability is maintained in a banking transaction. Write a short report.

6. VIEW SERIALISABILITY

"View-serialisability" is one of the weaker conditions that assure serialisability. View-serialisability regards all the associations between transactions **T** and **U** in a way that a database element is written by **T**, the value of which is read by **U**. The main dissimilarity among serialisability and view c become apparent when a value **A** is written by transaction **T** and it is not read by any transaction. **A** value for **A** is written thereafter by some other transaction.

In that situation, you can put the **WT (A)** action in some different locations of the schedule that would not be allowed in case of conflict-serialisability. In this section, we will discuss the concept of view-serialisability.

View equivalence: Let us presume two schedules named as $\mathbf{S_1}$ and $\mathbf{S_2}$ which are related to the similar group of transactions. Visualise an imaginary transaction $\mathbf{T_0}$. Preliminary values for every database element read by any transaction in the schedules are written by $\mathbf{T_0}$.

Again visualise some other imaginary transaction $\mathbf{T_f}$. Every element written by one or more transactions after the end of each schedule is read by $\mathbf{T_f}$.

Thereafter we can discover the write action Wj (A) that most closely headed the read in question for every read action rt (A) in one of the schedules. T3 is regarded to be the source of the read action Tt (A). T3 could be the imaginary initial transaction T0, and also Tt could be Tf. If for all read actions in one of the schedules, the source is similar in the other schedule, then $\mathbf{S_1}$and $\mathbf{S_2}$ are said to be view-equivalent

Certainly, view equivalent schedules are factually equivalent. When executed on any one database state, they both perform the same. We can say that **S** is view-seralisable if a schedule **S** is view-equivalent to a sequential schedule.

SELF-ASSESSMENT QUESTIONS – 5

9. In case of View-serialisability, all the associations among transactions T and U are regarded such that T writes a database element, the value of which is read by **U** (True/ False)
10. A schedule S is said to be _____, if S is view-equivalent to a sequential schedule.

7. RESOLVING DEADLOCKS

You may have noticed that simultaneous execution of transactions can lead to fight for resources. A deadlock situation is reached. Deadlock is a situation when two or more transactions are put into wait state simultaneously but in this state each would be waiting for the other transaction to get released so that they can proceed. In this state, no development is made by anyone as all numerous transactions waits for a resource held by one of the others. The deadlock can be effectively dealt with the help of two extensive techniques. The deadlocks can be identified and fixed. The matter could also be dealt with in such a manner that it never occurs. The methods are discussed as below.

7.1 Deadlock Detection by Timeout

On the occurrence of a deadlock, it is usually not possible to fix it in order that the transactions included could continue. Therefore, in such a situation a transaction may need to be rolled back (aborted and restarted).

Timeout is the easiest manner of identifying and resolving deadlocks. A limitation may be maintained on the active period of a transaction, and roll it back if a transaction goes beyond this time.

Example: In a transaction system, where usual transactions are carried out in milliseconds, a timeout of one minute would involve only transactions trapped in a deadlock.

In case of more difficult transactions, you may want the timeout to happen after a much longer interval.

Observe that when any transaction times out which is included in the deadlock, its locks or other resources are discharged. In such a situation, the other transactions included in the deadlock will compete prior to arriving at timeout limits.

7.2 The Waits-For Graph

Waits-for graph can effectively deal with deadlocks that take place due to transactions waiting for locks held by some other. The waits-for graph predicates the transactions waiting for locks held by some other transaction. This graph can be used to identify deadlocks after its formation or to avoid their occurrence at all. It requires us to preserve the waits-for graph always, denying to permit an action which forms a cycle in the graph.

In addition to transactions that at present hold locks on X, a lock table preserves for every database element X, the transactions list waiting for locks on X. The waits-for graph comprises a node for every transaction. Currently, it holds a lock or is awaiting one lock. An arc from node T to node U is present if there is a database element Alike say:

1. a lock on A is held by U,
2. T is awaiting a lock on A, and
3. T can't get a lock on A in the preferred mode unless U discharges its lock on A first.

If the waits-for graph does not include any cycle, then all transactions can finally complete. There will always be a transaction awaiting for no other transaction, which can definitely complete. At the same time, at least some other transaction will be present which is not waiting and can compete.

On the other hand, if a cycle is there, then any transaction in the cycle cannot make progress, owing to which, there is a deadlock. Thus, a deadlock can be avoided by rolling back a transaction making a request; which would lead to a cycle in the waits-for graph.

Example: Let us consider the four transactions. All the transactions read one element and write another:

T_1: $l_1(A)$; $r_1(A)$; $l_1(B)$; $w_1(B)$; $u_1(A)$; $u_1(B)$;
T_2: $l_2(C)$; $r_2(C)$; $l_2(A)$; $w_2(A)$; $u_2(C)$; $u_2(A)$;
T_3: $l_3(B)$; $r_3(B)$; $l_3(C)$; $w_3(C)$; $u_3(B)$; $u_3(C)$;
T_4: $l_4(D)$; $r_4(D)$; $l_4(A)$; $w_4(A)$; $u_4(D)$; $u_4(A)$;

In the above notations l = lock, r = read, w = write, u = unlock

In T_1 transaction, A is firstly locked (l_1), and then read (r_1). B is written (w_1) after that A and B is unlocked (u_1).

In T_2 transaction, C is locked (l_2) and read (r_2), after that A is locked (l_2) and written (w_2). And then both are unlocked (u_2).

In T_3 transaction, B is locked (l_3) and read (r_3), after that C is locked (l_3) and written (w_3). And then both are unlocked (u_3)

In T_4 transaction, D is locked (l_4) and read (r_4), after that A is locked (l_4) and written (w_4). And then both are unlocked (u_4)

Here, we make use of a simple locking system. It has only one lock mode. A similar effect would be observed if we use a shared/exclusive system and took locks in the suitable mode (shared for a read and exclusive for a write).

	T_1	T_2	T_3	T_4
1)	l_1 (A); r_1 (A);			
2)		l_2 (C); r_2 (C);		
3)			l_3 (B); r_3 (B);	
4)				l_4 (D); r_4 (D);
5)		l_2 (A); Denied		
6)			l_3 (C); Denied	
7)				l_4 (A); Denied
8)	l_1 (B); Denied			

Fig 7.3: Starting of a Schedule with a Deadlock

The starting of a schedule of these four transactions is shown in Figure 7.3. In the initial first four steps, every transaction gets a lock on the element sought to be read by it. At step (5),

T2 makes an effort to lock A, but the request is rejected since T1 already comprises a lock on A. Therefore, T2 waits for 1, and an arc is drawn from the node for T2 to the node for T1.

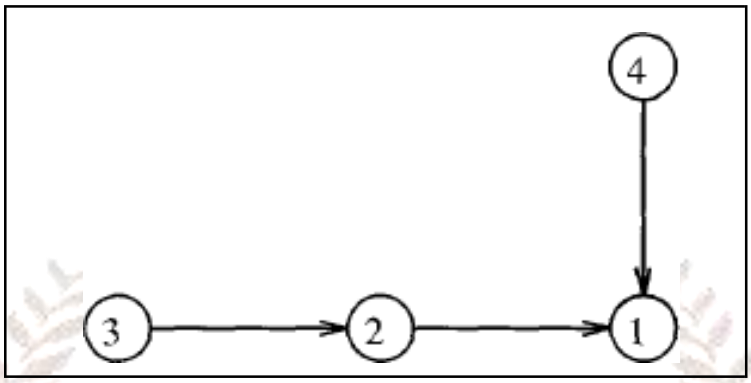

Fig 7.4: Waits-for Graph after Step (7) of Figure 7.3

Likewise, at step (6) T_3 rejected a lock on C due toT_2. At step (7), due to T_1, a lock on A is rejected to T_4. Now the waits-for graph is displayed in Figure 7.4. At step (8), no cycle is included in this graph. T_I must wait for the lock on B, The lock on B is held by T_3. If T_1 is allowed to wait, then this results in a cycle in the waits-for graph including T_I, T_2, and T_3. This is shown in Figure 7.5.

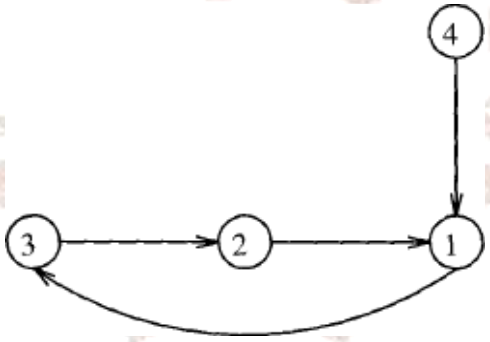

Fig 7.5: Waits-for Graph with a Cycle induced by Step (8) of Figure 7.3

As each of them is waiting for another to finish, there is no growth. This results in a deadlock, which includes these three transactions. Moreover, T_4 cannot come to an end either, even though it is absent in the cycle, as progress of T_2 relies on the progress made by T_1.

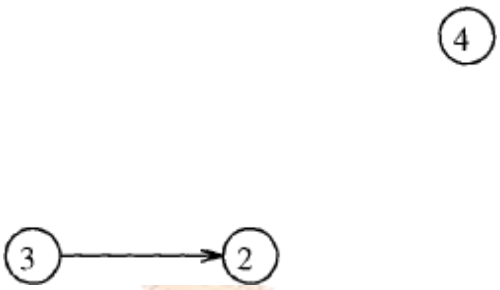

Fig 7.6: Waits-for graph on roll back of T1

If we roll back any transaction which can induce a cycle, then T_1 must be rolled back, deferring the waits-for graph of Figure 7.6. T_1 abandons its lock on *A*. This lock may be provided toT_2.Assuming that the lock is given to T_2, T_2 can now compete. It abandons its locks on *A* and *C*. At some time, T_1 is restarted, but it fails to obtain locks on *A* and *B* tillT_2, T_3,and T_4 finish.

SELF-ASSESSMENT QUESTIONS – 6

11. In case of _____, numerous transactions waits for a resource held by some others, and no one can grow.
12. The timeout method signifies transactions that are awaiting locks held by some other transaction. (True/ False)

8. DISTRIBUTED LOCKING

Now you will analyse extending a locking scheduler to an environment wherein transactions are distributed. The transactions comprise components at numerous sites. It is presumed that individual sites handle lock tables. It is further assumed that component of transaction at a site can at that site just ask for a lock on the data elements. On replication of the data, the copies of a single element X must be organised in such a manner that they are changed in the similar manner by every transaction.

This necessity brings in a difference among locking the logical database element X and also locking one or more copies of X. A clear (and at times sufficient) solution to the issue of maintaining locks in a distributed database is considered as centralised locking. This is discussed below.

8.1 Centralised Lock Systems

The most simple strategy is to assign one site, i.e. the lock site, for preserving a lock table for logical elements, irrespective of whether they comprise copies at that site or not.

A request is sent to the lock site when a transaction requires a lock on logical element X. The lock is either granted or denied. As getting a global lock on X is similar to getting a local lock on X at the lock site, you can be certain that the global locks act properly provided the locks are supervised conventionally by the lock site.

Unless the transaction happens to be running at the lock site the normal cost is three messages per lock, viz. request, grant, and release.

Single lock site would suffice in some circumstances. However, if there are more sites and more concurrent transactions, the lock site can turn out to be a restricted access.

No transaction can get any locks at a site if the lock site crashes. Due to these issues with centralised locking, another method is used for maintaining distributed locks, which is discussed as below.

8.2 Primary-Copy Locking

Primary-copy locking is considered as an enhancement over the centralised locking method. Primary-copy locking method includes distributing the function of the lock site. It however still preserves the rule that each logical element comprises of a single site accountable for the global lock.

This change prevents the chance that the central lock site will turn out to be a restricted access preserving the ease of the centralised method at the same time.

Each logical element in the primary-copy lock method comprises at least one copy which is known as the "primary-copy." The site of the primary copy sustains an entry for X in its lock table. It grants or rejects the request sent by the transaction, as suited.

Global (logical) locks will be managed properly provided the sites manage the locks for the primary copies appropriately.

Similar to centralised lock site, three messages are produced by most of the lock requests, apart from those requests where the transaction and the primary copy are available at the same site. On the other hand, primary copies are selected sensibly, then it can be expected that these sites will mostly be the same.

Example: Consider the chain-of-stores. It is necessary that each store's sales data includes its primary copy. The copies used at the central office, data warehouse or by sales analysts cannot be considered to be primary copies. Only usual transaction is carried out at the store. The updates include sales data for that store. When this type of transaction takes its locks no messages are necessitated. Associated messages will be sent only if the transaction observed or modified data at another store would lock

SELF-ASSESSMENT QUESTIONS – 7

13. Centralised Lock Systems include assigning one site, viz. the lock site, to preserve a lock table for logical elements, irrespective of whether or not they comprise copies at the site. (True/ False)
14. _____ Locking is as an enhancement over the centralised locking method.

9. TRANSACTION MANAGEMENT IN MULTI-DATABASE SYSTEM

A multi-database system is the one that permits the transactions of the customer to invoke retrieval and update commands in opposition to data situated in various hardware and software environments.

There are two types of transaction defined as below:

- ***Local transactions:*** Local transactions control local database system.
- ***Global transactions:*** Global transactions control multiple database system.

Transaction management is considered as complex in multi-database systems due to the supposition of autonomy.

In Global 2 Phase Locking, each local site utilises a strict 2 Phase Locking (locks are discharged at the end). Locks set in consequence of a global transaction are released only when that transaction arrives at the end. Global 2 Phase Locking assures global serialisability.

Because of autonomy necessities, sites cannot collaborate and perform a common concurrency control method. For example, there is no way to make sure that all databases pursue strict 2 Phase Locking.

The solutions to these include:

- provide very low level of simultaneous execution, or
- utilise weaker levels of consistency

Each local DBMS executes local transactions. These are executed outside of the multiple-database system control. We perform global transactions under multi-database control.

Local database managements systems cannot correspond directly to coordinate global transaction execution. The multi-database cannot manage local transaction execution.

To make sure that DBMS's schedule is serialisable, there is a need of local concurrency control technique. In locking, DBMS must be able to protect against local deadlocks.

Additional methods are required to guarantee global serialisability. DBMS guarantees local serialisability between its local transactions ,together with those that are part of a global transaction.

The multi database guarantees serialisability between global transactions. This is done by overlooking the orderings induced by local transactions. Two-level serialisability (2LSR) does not guarantee global serialisability; however, it can fulfil the needs for strong accuracy.

Now let us discuss the Global-read protocol and Local-read protocol.

Global-read protocol: Global transactions are capable to read, however, they cannot perform updation of local data items. Global data cannot be accessed by local transactions. No reliability constraints are there among local and global data items.

Local-read protocol: Local transactions comprise read access to global data. It prohibits all access to local data by global transactions.

A transaction comprises value dependency if the value that it writes to a data item at one site relies on a value that it read for a data on another site.

SELF-ASSESSMENT QUESTIONS – 8

15. Multiple database system is controlled by _____ transactions.
16. 2LSR does not guarantee global serialisability; but can fulfil the needs for _____.

10. LONG-DURATION TRANSACTIONS

Sometimes you will find a set of applications in which a database system is suitable for satisfying data. Yet the model of diverse small transactions for which database concurrency-control techniques are proclaimed, is not suitable. Now you will study the problems that arise during long transactions.

Problems of long transactions: A long transaction is the one that is permitted too much time to hold locks, and that particular lock is needed by another transaction. According to the environment, long time could signify hours, minutes or seconds.

We can presume that at least a number of minutes, and most likely hours, are included in "long" transactions. There are three extensive categories of applications that include long transactions. These are:

1. ***Conventional DBMS applications:*** Although general database applications perform usually short transactions, several applications require sporadic long transactions. For instance, a transaction may inspect all the bank's accounts to verify that total balance is correct. A different application might need one index to be reconstructed infrequently to maintain working at its height.
2. ***Design systems:*** An object designed may be mechanical such as a car, electronics like a microwave, or software system; the universal aspect of the design systems is that the design is separated into a set of different parts or components (for example, codes of some software scheme), and many designers work on these different components simultaneously. We would not like 2 designers working on a copy of one file, doing some edition in designs, and then composing the new file versions. This is for the reason that one group of changes would over- write the other.

Therefore, the system of check-out and check-in permits one designer to "check-out" a file and ‘check-in’ after the final changes, possibly hours or days afterwards. In this case, even if, the first designer is doing some changes a second designer may want to check out the file to discover something regarding its subjects. If the check-out procedure was tantamount to an exclusive lock, then some logical and reasonable actions would be postponed, probably for days.

3. ***Workflow systems:*** These systems include accumulations of processes, some performed by software itself, some including human contact, and probably some by human action alone. For example, let us consider an office paperwork including a bill payment. Such applications may take some time to perform, and throughout this period, some database components might be subject to some changes.

SELF-ASSESSMENT QUESTIONS – 9

17. A long transaction does not take much time to hold locks. (True/ False)
18. Which of the following applications is based on the concept that the design is separated into a set of different parts or components?
    a) Conventional DBMS applications
    b) Design systems
    c) Workflow systems.
    d) None of the above

11. HIGH PERFORMANCE TRANSACTION SYSTEMS

High-performance transaction systems assist in enhancing the rate of transaction processing, but are inadequate to acquire high performance:

- Disk I/O is a restricted access. I/O time does not decrease at a rate analogous to the increase in processor speeds.
- Parallel transactions may try to read or write the similar data item, resulting in data conflicts that decrease effectual parallelism.

We can diminish the extent to which a database system is disk bound by augmenting the size of the database buffer.

Main memory database: Commercial 64-bit systems can sustain main memories of tens of gigabytes. Memory resident data permits quicker processing of transactions.

There are some disk-related limitations which are defined as below:

- Logging is a restricted access when transaction rate is high.
- Use group-commit to decrease various output operations.
- If the update rate for customised buffer blocks is high, the disk data- transfer rate could turn out to be a restricted access.
- If the system crashes, all of main memory is vanished.

SELF-ASSESSMENT QUESTIONS – 10

19. The extent to which a database system is disk bound can be diminished by augmenting the _____ of the database buffer.
20. Commercial 64-bit systems can sustain main memories of tens of gigabytes. (True/False)

Activity 2

Make distinction between centralised locking and primary-copy locking.

12. SUMMARY

Let us recapitulate the important points discussed in this unit:

- A series of operations comprising database operations that is atomic with regard to concurrency and recovery is called transaction.
- The transaction manager obtains transaction commands from an application, which inform the transaction manager when transactions start and end.
- Online transaction processing is considered as interactive in which every transaction is processed as it takes place.
- Serialisability is the main criterion for the accuracy of simultaneous transactions executions and a main objective for concurrency control.
- Recoverability signifies that data is not read by the committed transactions where the data is written by terminated transactions.
- "View-serialisability" is one of the weaker conditions that assure serialisability.
- We can identify stalemates and repair them, or we can handle transactions in such a way that these problems may never occur.
- Transaction management is considered as complex in multi-database systems due to the supposition of autonomy.
- High-performance transaction systems assist in enhancing the rate of transaction processing, but are inadequate to acquire high performance.

13. GLOSSARY

- ***Online transaction processing system:*** Online transaction processing is considered as interactive in which every transaction is processed as it takes place.
- ***Recoverability:*** Recoverability signifies that data is not read by the committed transactions where the data is written by terminated transactions.
- ***Serialisability:*** Serialisability is the main criterion for the accuracy of simultaneous transactions executions and a main objective for concurrency control.
- ***Transaction:*** A series of operations comprising database operations that is atomic with regard to concurrency and recovery is called transaction.

14. TERMINAL QUESTIONS

1. What are the advantages and disadvantages of transaction processing system? Discuss.
2. Explain the concept of serialisability and Recoverability. Illustrate how to manage rollbacks by locking.
3. What do you mean by View-serialisability? Illustrate the concept.
4. Explain the different methods used for resolving deadlocks.
5. What are the problems that occur during long transactions? Illustrate.

15. ANSWERS

Self-Assessment Questions

1. Stability
2. Transaction manager
3. Organised
4. False
5. Pre-defined
6. False
7. Dirty
8. True
9. True
10. View-seralisable
11. Deadlock
12. False
13. True
14. Primary-Copy
15. Global
16. Strong accuracy
17. False
18. b) Design systems
19. Size
20. True

Terminal Questions

1. One advantage is that every business firm needs a system for the collection, accumulating and recovering of data and statistics, so that it can function competently. One of the disadvantages is that there is need to manage hundreds and thousands of simultaneous consumers. Refer Section 3 for more details.
2. Serialisability is the main criterion for the accuracy of simultaneous transactions executions and a main objective for concurrency control. Recoverability signifies that data is not read by the committed transactions where the data is written by terminated

transactions. The difficulty of continuous rollbacks can be handled in a lock-based scheduler. An easy method, known as strict locking, which is used to provide assurance that there are no cascading rollbacks. Refer Section 5 for more details.

3. View-serialisability" is one of the weaker conditions that assure serialisability. View-serialisability considers all the associations among transactions T &U such that T writes a database element whose value U reads. Refer Section 6 for more details.
4. The methods used for resolving deadlock includes Deadlock Detection by Timeout and Waits-For graph. In case f timeout method, keep a limit on the active period of a transaction, and if a transaction goes beyond this time, roll it back. The Waits-For Graph signifies which transactions are waiting for locks held by another transaction. Refer Section 7 for more details.
5. Approximately, a long transaction is one that is allowed too much time to hold locks, and that particular lock is needed by another transaction. According to the environment, long time could signify seconds, minutes, or hours. Refer Section 10 for more details.

16. REFERENCES

References:

- Lewis, P.M. et al. (2002). *Databases and transaction processing: an application-oriented approach,* Addison-Wesley.
- Silberschatz, A. et al. (2006). *Database system concepts,* McGraw-Hill Higher Education.

E-references

- http://ambarwati.dosen.narotama.ac.id/files/2011/05/FIS-2011-w2.pdf, retrieved on 09-04-12
- http://www.scribd.com/doc/44537611/23/Serializability, retrieved on 09-04-12
- http://ambarwati.dosen.narotama.ac.id/files/2011/05/FIS-2011-w2.pdf, retrieved on 09-04-12
- http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zmainframe/zconc_onlinetrans.htm, retrieved on 09-04-12.