MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 8

Concurrency Control

Table of Contents

SL No	Topic		Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction		-	-	3
	1.1	Objectives	-	-	
2	Enforcing Serialisability by Locks		-	1	4-6
	2.1	Locks	-	-	
	2.2	Locking scheduler	-	-	
	2.3	Two phase locking	-	-	
3	Locking Systems with Several Lock Modes		-	2, I	7-9
4	Architecture for a Locking Scheduler		-	3	10-14
	4.1	Two-part scheduler	1	-	
	4.2	The lock table	2	-	
5	Managing Hierarchies of Database Elements		3, 4, 5, 6	4	14-17
6	Concurrency Control by Timestamps		-	5	18-19
	6.1	Timestamp resolution	-	-	
	6.2	Timestamp locking	-	-	
7	Concurrency Control by Validation		-	6, II	20-21
8	Database Recovery Management		-	7	22-23
9	Summary		-	-	24
10	Glossary		-	-	25
11	Terminal Questions		-	-	26
12	Answers		-	-	26-27
13	References		-	-	28

1. INTRODUCTION

In the previous unit, you have learned about transaction processing. We discussed the process in detail and also studied about its advantages and disadvantages, serialisability and recoverability, distributed locking, transaction management in multi-database system, long duration transaction and high-performance transaction system.

In this unit, we will reflect on the concept of concurrency control serialisability which includes methods used for enforcing serialisability by locks, several lock modes used in locking system and architecture of locking scheduler. You will also recognise the process of managing hierarchies of database elements. We will also discuss concurrency control methods like concurrency control by timestamp and validation. Lastly, you will recognise the concept of database recovery management.

1.1 Objectives

After studying this unit, you should be able to:

- ❖ *recognise the methods used for enforcing serialisability by locks*
- ❖ *identify several lock modes used in locking system*
- ❖ *discuss the architecture of locking scheduler*
- ❖ *discuss how to manage hierarchies of database elements*
- ❖ *explain concurrency control by timestamp and validation*
- ❖ *discuss database recovery management*

2. ENFORCING SERIALISABILITY BY LOCKS

Serialisability describes the concurrent execution of several transactions. The objective of serialisability is to find the non-serial schedules that allow transactions to execute concurrently without interfering with one another and thereby producing a database state that could be produced by a serial execution. Serialisability must be guaranteed to prevent inconsistency from transactions interfering with one another. The order of Read and Write operations are important in serialisability.

We use the following methods for enforcing serialisability by locks:

- Locks
- Locking scheduler
- Two phase locking

These are discussed as below.

2.1 Locks

Performing locks on a resource is one of the methods that are used to serialise transactions. Any data that is going to be utilised in support of the transaction will be locked by the process.

A lock is used by the transaction to reject data access to other transactions and thus avoid inaccurate updates. Locks can be of different types such as Read (shared) or Write (exclusive) locks.

A data item's write locks stops other transaction from reading that data item. On the other hand, Read Locks just prevent other transactions from editing (writing to) the data item. You will study these types later in the unit. So, we can say that there can be three states of a lock:

- Exclusive (write) lock
- Shared (read) lock
- Unlocked

You can use locks in the following manner:

1. Firstly, the transaction which is required to access any data item must lock the item. It makes a request to a shared lock for read only access or it makes a request to an exclusive lock for read and write access.
2. If another transaction does not lock the item, then the lock will be provided.
3. If the item is currently in the locked state, then the database management system (DBMS) identifies whether the request is well- matched with the current lock. If an item already having shared lock gets a request of shared lock, then the request will be granted. Or else, transaction must wait until the current lock is released.
4. Transaction maintains to hold a lock until it is clearly released either during execution or when it finishes (that is aborts or commits). The effects of the write operation will become noticeable to another transaction only when the exclusive lock has been released.

2.2 Locking scheduler

The scheduler is used to create the order which implements the operations inside simultaneous transactions. To guarantee serialisability, the implementation of database operations is interleaved by the scheduler. For identifying the proper order, the scheduler establishes its proceedings on algorithms of concurrency control, like time stamping or locking methods, which are discussed later in the unit. It also ensures that the computer's central processing unit (CPU) is used in an efficient manner.

Lock requests are allowed by the locking scheduler provided the condition is that of being in a legal schedule. Lock table accumulates the information regarding existing locks on the elements.

2.3 Two Phase Locking

Two phase locking is another method which is used to ensure serialisability. The "two-phase" lock is followed by a transaction if all lock requests occur prior to the first unlock operation inside the transaction. Two phase locking defines the process of acquiring and giving up the locks. It does not prevent deadlocks. (Deadlocks take place when two or more transactions are in a waiting position and they are waiting for the locks held by each other that are to be released.)

Two main phases are there within the transaction:

- ***Growing phase:*** In case of growing phase, all necessary locks are attained by a transaction without releasing data.
- ***Shrinking phase:*** In case of shrinking phase, all locks are released by a transaction. Also you cannot attain any new lock.

The rules followed in case of two phases locking protocol are given below:

- Two transactions cannot comprise inconsistent locks.
- No unlock operation can be performed before a lock operation in the similar transaction.
- It will not affect any data until the attainment of all the locks.

If, throughout the initial stage, a process is not able to obtain all the locks, then it is necessary to release all of them. They wait, and begin once more. Make note that if two phase locking is utilised by all transactions, then each schedule created by interleaving them are said to be serialisable.

To make sure that data is not accessed by a transaction until another transaction operating on the data has either committed or aborted, locks may be held until the transaction is committed or terminated. We call this as strict two phase locking. This is comparable to two phase locking excluding that here, the shrinking phase occurs throughout the commit or abort. We have one consequence of strict two phase locking. By placing the second phase at the end of a transaction, all lock acquirements and releases can be managed by the system without the knowledge of transaction.

SELF-ASSESSMENT QUESTIONS – 1

1. Which phase of locking defines the process of acquiring and giving up the locks?
   a) Growing Phase
   b) Shrinking Phase
   c) Two Phase
   d) None of the above
2. Locking scheduler permits lock requests only if it is in a legal schedule. (True/ False)

3. LOCKING SYSTEMS WITH SEVERAL LOCK MODES

A lock comprises a mode that identifies its power. It identifies, for example, whether it stops other users from reading, or just from changing, the data.

Based on the type of lock mode, when one user comprises a lock on a record, the lock does not allow other users to change that record. It stops other users even from reading that record.

Let us now discuss the several lock modes as bellow:

1. ***Shared Locks:*** In case of Row-level shared locks, various users are permitted by shared locks to read data, but no user is permitted to change that data. In case of Table-level, various users are permitted by shared locks to carry out read and write operations on the table, but any user is not permitted to carry out Data Definition Language (DDL) operations. Numerous clients can hold shared locks simultaneously.
2. ***Exclusive Locks:*** In this case, only one user is permitted to update a specific part of data (that is insert, update, and delete). When one client holds an exclusive lock on a row or table, it is not allowed to place other lock of any type on it.
3. ***Update Locks:*** Update locks, at all times, are considered as row-level locks. When the client uses the row by means of the SELECT FOR UPDATE statement, the row is locked by using an update mode lock. This signifies that the row cannot be read or updated by any other user and makes sure that the existing user can update the row afterwards.

Update locks are said to be comparable to exclusive locks. The major distinction among the two is that you can obtain an update lock when another user already comprises a shared lock on the same record. This allows the update lock's holder to read data and it does not prohibit other clients. On the other hand, once the data is changed by the update lock holder, the update lock is transformed into an exclusive lock.

In addition, update locks are said to be asymmetric with regard to shared locks. You can obtain an update lock on a record that already comprises of a shared lock, but you cannot obtain a shared lock on a record that already comprises of an update lock. Since an update lock avoids successive read locks, it is simpler to transform the update lock to an exclusive lock.

4. ***Upgrading Locks:*** Let us assume that a transaction is required to read as well as write. In this case, it obtains a shared lock on the element. It carries out the calculations on the element. Also, when it is ready to write, an exclusive lock is granted to it.

Transactions with unexpected read write locks can make use of Upgrading Locks. Use of upgrading locks in a random manner generates a deadlock. For example, both the transactions want to upgrade on the same.

5. ***Increment Locks:*** Increment Locks are utilised for incrementing & decrementing stored values. For example, in case of ticket selling transactions, number of seats is decremented after every transaction.

Read or write locks are not allowed by an increment lock. Various transactions can hold increment lock on element. If an increment lock is granted on element, then shared and exclusive locks cannot be granted.

You cannot merge shared and exclusive locks. If User 1 is having an exclusive lock on a record, then a shared lock or an exclusive lock cannot be obtained on that same record, by User 2.

Within a specific group, all locks are treated as equal.

- All users in spite of the user privileges are equal. Locks that are placed by a Date Base Administrator are considered equivalent to the locks that are placed by some client.
- All methods of executing statements that place locks are equal.

It is not bothered whether you execute lock as portion of a typed statement, which is called from a remote application (compiled), or called from inside the local application, or if you place the lock as a consequence of a statement inside an accumulated procedure or trigger.

You cannot escalate some locks. For example, if you are utilising a scroll cursor and you obtain a shared lock on a record, and then afterward within that same transaction, that record is updated. Obtaining an exclusive lock is only possible if no other locks are there on the table. If you and another client both comprise of shared locks on the similar record, then the server cannot upgrade your shared lock to an exclusive lock until the other client drops her shared lock.

SELF-ASSESSMENT QUESTIONS – 2

3. Which types of locks are used for incrementing & decrementing stored values?
   a) Exclusive locks
   b) Upgrade locks
   c) Decrement locks
   d) Increment lock
4. In shared locks, only one user is permitted to update a specific part of data. (True/False)

Activity 1

How can you compare Shared locks and Exclusive locks? Illustrate.

4. ARCHITECTURE FOR A LOCKING SCHEDULER

To understand the architecture of a locking scheduler, consider a simple scheduler that:

- Inserts locks for transaction
- Releases locks when told

Now we will discuss about two-part scheduler as given below.

4.1 Two-Part Scheduler

Here, the scheduler is divided into two parts as shown in Figure 8.1.

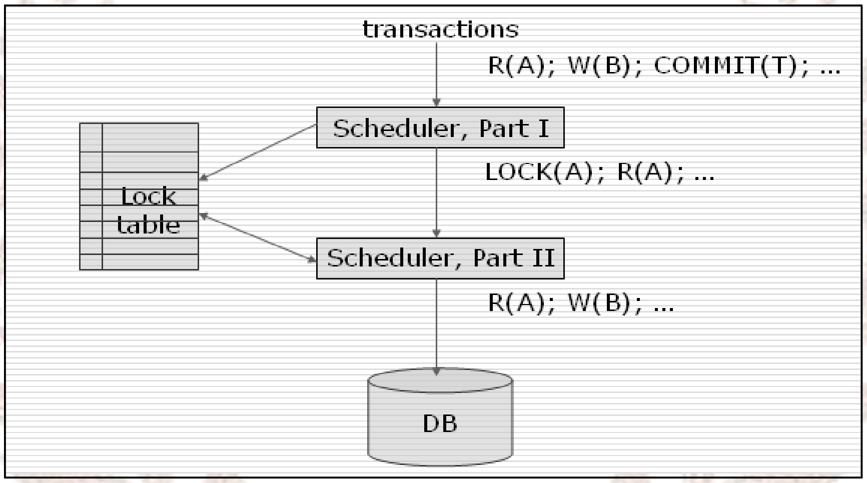


Fig 8.1: Two-part Scheduler

Let us now discuss the concept of two-part scheduler. As you can see in the Figure 8.1 above, it contains two parts, that is, part I and part II.

Part I obtain the number of requests from the transactions and inserts lock actions prior to all database-access operations. It must choose a suitable lock mode. Thus, Part I select & insert suitable lock modes to database (DB) operations such as read, writes, or update.

Actions (such as a lock or db operation) performed by part I are taken by part II. Part II executes the number of actions received by Part I. It finds out if the transaction T delay is required since a lock has not been provided. If this is the case, then add the action to the list of actions that must finally be performed for transaction T.

So, now we identify the transaction (T) to which that action belongs and find out status of T (delayed or not).

- If T is delayed, then action is delayed and it is added to wait list.
- If T is not delayed, then, two cases are possible.
  - ❖ If the action is a db operation, then it is transmitted to the database and executed.
  - ❖ If the action is a lock, check the lock table to observe if the lock can be granted.
    - ♦ If the lock is granted, then modify the lock table to comprise the lock just granted.
    - ♦ If not, make an entry in the lock table to specify that the lock has been requested. More actions for transaction T are postponed, until the lock is provided.
- When T is done (commits or aborts), transaction manager (belongs to T) informs Part I to release all locks, if waiting lock is still there, Part I then informs Part II.
- When Part II is informed that a lock is available on some database element.
  - ❖ It identifies the next transaction or transactions that can now be provided a lock on element.
  - ❖ Those transactions are permitted to execute their postponed actions until they either complete or arrive at another lock request that cannot be granted.

Let us now recognise the concept of lock table.

4.2 The Lock Table

The lock table signifies a relation that relates database elements with information regarding locking. Size, in lock table, is comparative only to the number of lock elements, and not to the size of the whole database.

We have shown this in Figure 8.2 of lock table as below

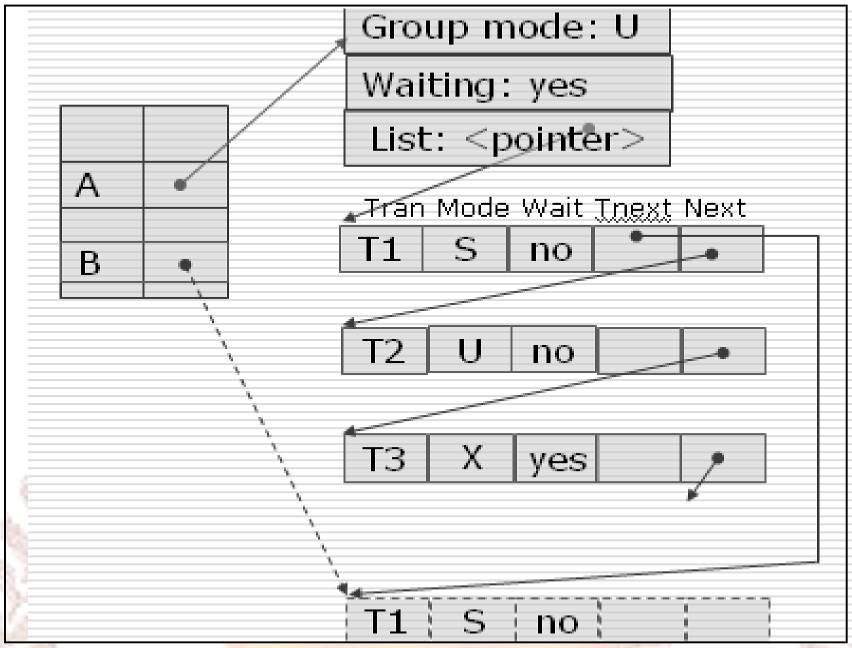


Fig 8.2: The Lock Table

As you can see in the Figure, there are various group modes which are defined as below:

- S: It symbolises only shared locks.
- U: It symbolises one update lock and zero or shared locks.
- X: It symbolises one exclusive lock and no other locks.

Waiting bit notifies that at least one transaction is waiting to have a lock on A and B. Let us illustrate all the transactions which are either holding or waiting for a lock on A and B. Each entry has:

- The name of the transaction
- The mode of this lock
- Whether transaction is holding or waiting for a lock
- Pointer associating entries together
- Pointer associating all entries for a specific transaction (Tnext). It is utilised when a transaction commits or aborts to discover all the locks that must be released.

Handling Lock Requests: Let us assume that a transaction T requests a lock on A

- If there is no lock table entry for A, then there are no locks on A, so create the entry and grant the lock request.
- If the lock table entry for A exists, use the group mode to guide the decision about the lock request.
- If group mode is U (update) or X (exclusive), then
  - ❖ Reject the lock request by T
  - ❖ Place an entry on the list notifying that T requests a lock
  - ❖ And Wait? = 'yes'
- If group mode is S (shared), then
  - ❖ Request I granted for an S or U lock
  - ❖ Entry is created for T on the list with Wait? = 'no'
  - ❖ In the case where the new lock is considered as an update lock, we change the group mode to U.

Handling Unlock Requests: Now let us assume that a transaction T unlocks A. Then:

- Delete entry of T on the list for A
- If T's lock is not matching to that of the group mode, it is not required to change group mode
- Or else if T's lock is:
  - ❖ X, then there are no other locks
  - ❖ U or S, then find out if there are remaining S locks or not.
- If the value of Waiting is 'yes', it is required to grant one or more lock requests

There are various different approaches defined as below:

- ***First-Come-First-Served:*** This approach grants longest waiting request. It does not provide starvation.
- ***Shared Locks Priority:*** This approach grants all S locks waiting, then one U lock. It grants X lock if others are not waiting.
- ***Upgrading Priority:*** If there is a U lock waiting for upgrading an X lock, grant that first.

SELF-ASSESSMENT QUESTIONS – 3

5. The ___________________ represents a relation which is related with the database elements with information regarding locking.
6. Part II selects & inserts suitable lock modes to database (DB) operations such as read, write, or update. (True/False)

5. MANAGING HIERARCHIES OF DATABASE ELEMENTS

To understand this concept, let us first recognise the term "Database Element".

The term "Database Element" points to various elements in the database. Different systems lock different sizes of database elements (such as Relations, Blocks, Tuples). It is based on the Application and the structure of the data for which the Database is utilised.

Managing hierarchies concentrate on two problems that occur when there is a tree structure to our data.

1. The first tree structure that takes place is the hierarchy of lockable elements. It illustrates the process of allowing locks on both large elements, such as Relations and smaller elements included in it such as blocks and tuples of relation, or individual.
2. Another type of hierarchy is data that is itself organised in a tree. One of the major examples would be B-tree index.

Locks with multiple granularity

Locking functions in any situation, however it is important to identify whether to select large objects or small objects. Also it is necessary to recognise the levels of granularity at which we shall lock. It follows the following process of transaction: the lower the level of granularity, the more concurrency, but the more locks and the higher the locking overhead. Best transaction process depends on application, for example, locking blocks or tuples in bank database, and entire documents in document database.

There may be a requirement for locks at numerous levels of granularity, even inside the same application. The database elements such as Relations, Blocks, and Tuples are structured in a hierarchy in the following manner as shown in the Figure 8.3 as below.

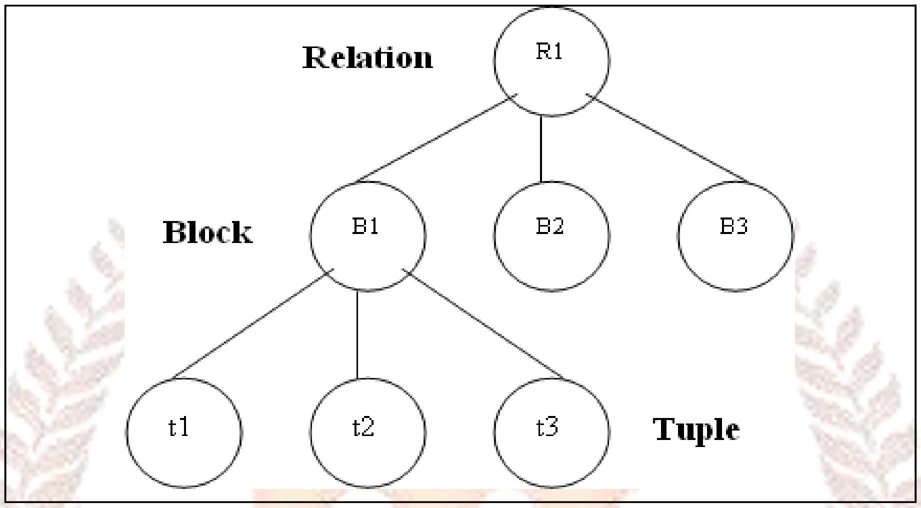


Fig 8.3: Hierarchy of Database Elements

Example: Bank Application (as shown in Figure 8.4)

Here, the relations, block and tuples are considered as follows:

- ***Relation:*** Account Balances
- ***Block:*** Accounts
- ***Tuples:*** Add/Deposit, Remove/Withdraw, Sum

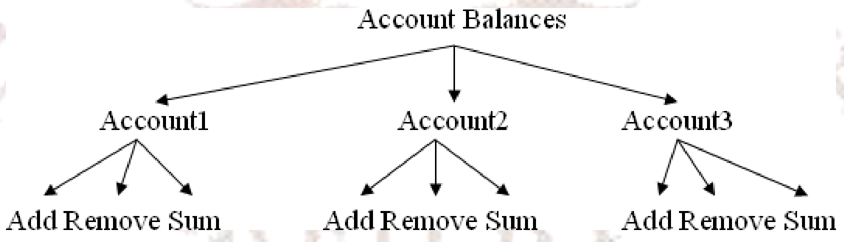


Fig 8.4: Example of Bank Application

- If Locks takes place on the Relation Level, then only one lock would be there for the whole Account Balances Relation.
- The overall Account Balances are changed by most of the transactions. Thus, an exclusive lock is needed by the transactions on the Account Balances Relation.
- Only one deposit or withdrawal could happen at any time

- Thus as a result, the system would permit very little concurrency.
- Just lock separate Account Blocks or Account Tuples
- Offering a Lock for each tuple is too fine-grained and is perhaps does not value the effort.

Warning Protocol

The warning protocol is used to handle locks on a hierarchy of database elements.

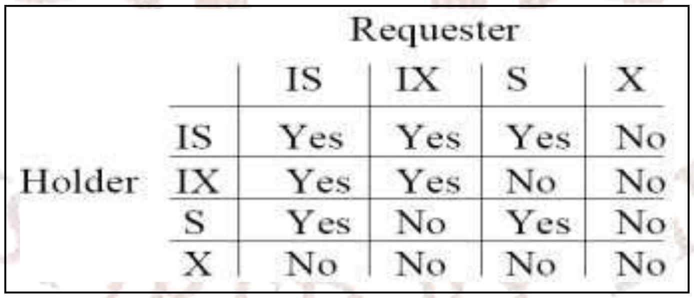
Here two new types of locks are introduced. They are:

- ***IS:*** IS signifies intention to request an S lock.
- ***IX:*** IX signifies intention to request an X lock.

An IS (or IX) lock specifies the intention to request S (or X) lock for a sub element down in the hierarchy. For requesting an S (or X) lock on any element (A) of database, a path is travelled from the root to the element A. If we have arrived at A, then the S (or X) lock is requested.

Or else, an IS (or IX) lock is requested. Once we have acquired the requested lock, we continue to the corresponding child (if required).

Now let us see the Compatibility matrix between locks as shown in Figure 8.5 below:



		Requester			
		IS	IX	S	X
Holder	IS	Yes	Yes	Yes	No
	IX	Yes	Yes	No	No
	S	Yes	No	Yes	No
	X	No	No	No	No

Fig 8.5: Compatibility Matrix between Locks

If two transactions are proposed to read or write a sub element, an I lock is provided to both of them. Thus the potential conflict will be resolved at a lower level.

An I lock used for a super element provides a restriction to the locks that the similar transaction can acquire at a sub element.

- If the parent element P as shown in Figure 8.6, in IS is locked by Ti, then Ti can lock child element C in IS, S.
- If the parent element P in IX is locked by Ti, then Ti can lock child element C in IS, S, IX, X

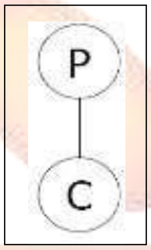


Fig 8.6: Parent Element and Child Element

SELF-ASSESSMENT QUESTIONS – 4

7. _________________ lock specifies the intention to request an S lock.
8. The process of managing locks on a hierarchy of database elements is performed by warning protocol. (True/ False)

6. CONCURRENCY CONTROL BY TIMESTAMPS

Concurrency Control is defined as the coordination of the concurrent execution in a multi-processing database system. The purpose of concurrency control is to guarantee the transactions serialisability in the environment of a multi-user database.

Whenever a transaction begins, a timestamp is provided to it. Through this, we can tell the order in which the transactions are considered to be applied in. Thus, if two transactions are provided that affect the same object, the transaction having the previous timestamp is intended to be applied prior to the other one. But, if we present the wrong transaction first, it is terminated and must be started again.

All the objects in the database comprise of a read timestamp and a write timestamp. Read timestamp is updated when the data of the object is read. A write timestamp is updated when the data of the object is changed. If a transaction is required to read an object, but the transaction initiated before the write timestamp of object, it signifies that an object's data is changed after the initiation of a transaction. In this situation, the transaction is cancelled and must be started again.

If there is a need of a transaction to write to an object, and the transaction initiated prior to the object's read timestamp, it signifies that the object is viewed, and it is presumed that it had taken a copy of the object's data. Thus you cannot write to the object because that would make any copied data invalid. Thus, the transaction is terminated and must be started again.

6.1 Timestamp resolution

Timestamp Resolution is defined as the smallest amount of time passed among two neighbouring timestamps. If the timestamp's resolution is excessively large, the chance of two or more timestamps being equal is augmented. This therefore enables some transactions to give out of correct order.

For example, let us suppose that we are having a system that can generate number of exclusive timestamps per second, and two events are provided that take place 2 milliseconds apart, then they will possibly be provided the same timestamp although they in fact occurred at unusual times.

6.2 Timestamp locking

Although this method is a method of non-locking, that is, the object is not locked from simultaneous access for the period of a transaction. The act of recording every timestamp in opposition to the Object needs a very short duration lock on the object or its substitute.

Concurrency control by means of timestamps is considered dissimilar as compared to locking in one significant manner. When a transaction comes across a later timestamp, it terminates. With locking, it would either wait or continue instantly.

SELF-ASSESSMENT QUESTIONS – 5

9. If we present the wrong transaction first, it is not necessary to abort it. (True/False)
10. What is the minimum time passed among two neighbouring timestamps known?
    a) Timestamp Collection
    b) Timestamp Locking
    c) Timestamp Resolution
    d) Timestamp Revision

7. CONCURRENCY CONTROL BY VALIDATION

Transactions can continue without locking. All database modifications are performed on a local copy. We verify if the schedule for transaction is serialisable or not. If it is serialisable, the local copy variations are affected to the global database. Or else, the local modifications are not needed, and the transaction is started again.

For every transaction T, the scheduler preserves two sets of significant database elements which are defined as below:

- RS(T), the read set of T: It is the set of all database elements read by T.
- WS(T), the write set of T: It is the set of all database elements written by T.

This information is critical to find out whether some schedule that has previously been executed was really serialisable.

We execute Transaction T in three phases:

1. ***Read:*** In this phase, transaction reads every element from database. It performs all its actions in its local address space.
2. ***Validate:*** In this phase, the serialisability of the schedule is verified by contrasting RS(T) and WS(T) to the read / write sets of the simultaneous transactions. If validation is ineffective, skip phase 3.
3. ***Write:*** In this phase, we write the new values of the elements in WS(T) back to the database.

The scheduler preserves three sets of transactions and some significant information at any time.

1. ***START:*** It is a set of transactions that have initiated, but have not still accomplished their validation stage. For every element T of START, keep START(T).
2. ***VAL:*** It is a set of transactions that have completed validation, but not still accomplished their write stage. For elements T of VAL, record VAL(T).
3. ***FIN:*** It is a set of transactions that have accomplished all the phases. For T in FIN, keep FIN(T).

Let the validation order be T1, T2, T3, …. Thus the resultant schedule will be considered as conflict equivalent to sequential schedule S = T1, T2, T3.

You can consider every transaction that productively validates as executing completely at the moment that it validates.

SELF-ASSESSMENT QUESTIONS – 6

11. What do you call a set of transactions that have initiated, but have not still finished their validation phase?
    a) START
    b) VAL
    c) FIN
    d) STOP
12. ***FIN*** is a set of transactions that have finished validation, but still not finished their write phase. (True/False)

Activity 2

Illustrate how concurrency control by validation is performed. Explain with example.

8. DATABASE RECOVERY MANAGEMENT

Database Recovery is concerned with the restoring of the database to an accurate state in the occurrence of a failure. There are numerous storage devices that hold data. They are main memory, magnetic disk, magnetic tape and optical disks. The DBMS needs recovery on the different types of failures defined as below:

1. System crashes (Software & Hardware)
2. Media failures
3. Application software error
4. Natural physical disaster
5. Unintentional distraction
6. Sabotage
7. Programming Exemption

To prevent data from failure different backup methods are used. There are three different levels of backup. These are discussed as below:

- ***Full database backup***: It considers a full backup of the database, or dump of the database.
- ***Differential backup:*** In case of differential backup, only the last variations that are performed on the database are copied.
- ***Transaction log backup:*** A transaction log backup method is used to back up the transaction log operations that are not shown in a preceding backup copy of the database.

The steps generally followed in the database recovery process are given as below:

- First, identify the type and the scope of the required recovery.
- If the whole database is required to be recovered to a reliable state, then the recovery makes use of the latest backup copy of the database in a recognised reliable state.
- The backup copy is then rolled forward to restore all subsequent transactions by means of the transaction log info.

- If the database is required to be recovered but the committed segment of the database is still functional, the recovery procedure makes use of the transaction log to 'undo' every transaction that is not committed.

Transaction recovery includes the following:

- ***Write-ahead protocol:*** This protocol makes sure that transaction logs are always written before any database data that are really updated.
- ***Redundant transaction logs***: Most DBMS maintain numerous copies of the transaction log to make sure that a disk physical failure will not harm the DBMS ability for recovering data.
- ***Database buffers:*** Buffer is considered as a temporary storage part available in primary memory. It is used to accelerate disk operations.
- ***Database checkpoint:*** It is an operation where all the updated buffers are written to the disk by the DBMS.

There are two fundamental characteristics for transaction recovery:

1. ***Deferred-write and Deferred-update:*** When deferred write or deferred update is used by the recovery process, the transaction operations do not update the physical database instantly.
2. ***Write-through:*** When the recovery process utilises the write through or immediate update, the database is instantly updated by transaction operations throughout the transaction's execution, even before the transaction arrives at its commit point.

SELF-ASSESSMENT QUESTIONS – 7

13. Which type of backup of the database is used to copy only the last modifications that are performed on the database?
    a) Full database backup
    b) Differential backup
    c) Transaction log backup
    d) None of the above
14. A temporary storage area in primary memory is known as buffer. (True/ False)

9. SUMMARY

Let us recapitulate the important points discussed in this unit.

- The purpose of concurrency control is to guarantee the transactions' serialisability in a multi-user database environment.
- A lock is used by the transaction to reject data access to other transactions and thus avoid inaccurate updates.
- The scheduler is used to create the order which implements the operations inside simultaneous transactions. To guarantee Serialisability, the implementation of database operations is interleaved by the scheduler
- The "two-phase" lock is followed by a transaction if all lock requests occur before the first unlock operation inside the transaction.
- Based on the type of lock mode, when one user is having a lock on a record, the lock stops other users from changing that record.
- The lock table signifies a relation that relates database elements with information regarding locking.
- The term "Database Element" points to various elements in the database. The database elements such as Relations, Blocks, and Tuples are organised in a hierarchy.
- Whenever a transaction begins, a timestamp is provided to it. Through this, we can tell the order in which the transactions are considered to be applied in.
- Database Recovery is concerned with the restoring of the database to an accurate state in the occurrence of a failure.

10. GLOSSARY

- ***B-Tree Index:*** It is a tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time.
- ***Concurrency Control:*** The coordination of the concurrent execution in a multiprocessing database system is known as Concurrency Control.
- ***Database Element:*** The term "Database Element" points to various elements in the database.
- ***Database Recovery:*** Database Recovery is concerned with the restoring of the database to an accurate state in the occurrence of a failure.
- ***Lock Table:*** The lock table signifies a relation that relates database elements with information regarding locking.
- ***Lock:*** A lock is used by the transaction to reject data access to other transactions and thus avoid inaccurate updates.
- ***Scheduler:*** The scheduler is used to create the order which implements the operations inside simultaneous transactions.
- ***Two-phase locking:*** The "two-phase" lock is followed by a transaction if all lock requests occur before the first unlock operation inside the transaction.

11. TERMINAL QUESTIONS

1. How is two phase locking method used to ensure serialisability? Illustrate.
2. What are the different lock modes used in the locking system? Discuss.
3. Explain the architecture for a Locking Scheduler.
4. Illustrate the concept of locks with multiple granularity with example.
5. Explain the concept of database recovery management. Discuss the different levels of backup used for recovering data.

12. ANSWERS

Self-Assessment Questions

1. c) Two phase
2. True
3. Increment
4. False
5. lock table
6. False
7. IS
8. True
9. False
10. Timestamp Resolution
11. START
12. False
13. b) Differential backup
14. True

Terminal Questions

1. The "two-phase" lock is followed by a transaction if all lock requests occur before the first unlock operation inside the transaction. Two phase locking defines the process of acquiring and giving up the locks. It does not prevent deadlocks. If two phase locking is used by all transactions, then all schedules created by interleaving them are serialisable. Refer Section 2 for more details.

2. Locking system includes various lock modes such as Shared Locks, Exclusive Locks, Update Locks, Upgrading Locks, and Increment Locks. Refer Section 3 for more details.
3. The scheduler is divided into two parts. Part I selects & inserts suitable lock modes to database (DB) operations such as read, write, or update. Actions (such as a lock or db operation) performed by part I are taken by part II. Part II executes the number of actions received by Part I for more details.

   The lock table signifies a relation that relates database elements with information regarding locking. In lock table, size is proportional to the number of lock elements only, not to the size of the whole database. Refer Section 4 for more details.
4. Locking functions in any situation, however it is important to identify whether to select large objects or small objects. Also it is necessary to recognise the level of granularity at which we shall lock. It follows the following process of transaction: the lower the level of granularity, the more concurrency, but the more locks and the higher the locking overhead. Best transaction process depends on application, for example, locking blocks or tuples in bank database, and entire documents in document database. Refer Section 5.1 for more details.
5. Database Recovery is concerned with the restoring of the database to an accurate state in the occurrence of a failure. There are numerous storage devices that holds data. To prevent data from failure different backup methods are used. Refer Section 8 for more details.

13. REFERENCES

References:

- Majumdar, A.K. & Bhattacharya, P. (2006) *Database Management Systems,* 18th edition, Tata McGraw-Hill Education.
- Singh, S.K. (2009) *Database Systems: Concepts, Design and Applications,* 3rd edition, Pearson Education.

E-references

- http://www.eee.metu.edu.tr/~vision/LectureNotes/EE442/Ee442ch7.html, 05-04-12
- http://www.slideshare.net/koolkampus/ch16, 05-04-12