Module - V **Transaction Management, Concurrency Control, Recovery System**

Transaction Management: Transaction processing, Transaction Concept, Transaction States, Implementation of Atomicity and Durability, Concurrent Executions

Concurrency Control: Lock-Based Protocols, Timestamp- Based Protocols, Validation-Based Protocols, Multiple Granularity

Recovery: Failure Classification, Recovery and Atomicity, Log-Based Recovery.

Transaction & Transaction States

A Transaction is a logical unit of work that must be entirely completed or entirely aborted, no intermediate states are acceptable. It consists of a combination of select, update and insert statements.

If any of the SQL statements fail, the entire transaction is rolled back to the original database state, that existed before the transaction started.

Transaction Model in DBMS

Each transaction has following 5 states

- Active
- Partially committed
- Committed
- Failed
- Aborted

Each transaction has following 5 states

Active

This is the first state of transaction and here the transaction is being executed. For example, updating or inserting or deleting a record is done here.

Partially committed

This is also an execution phase of transaction. Here Second step of transaction is executed. But data is still not saved to the database. In our example of calculating total marks, final display the total marks step is executed in this state

Committed

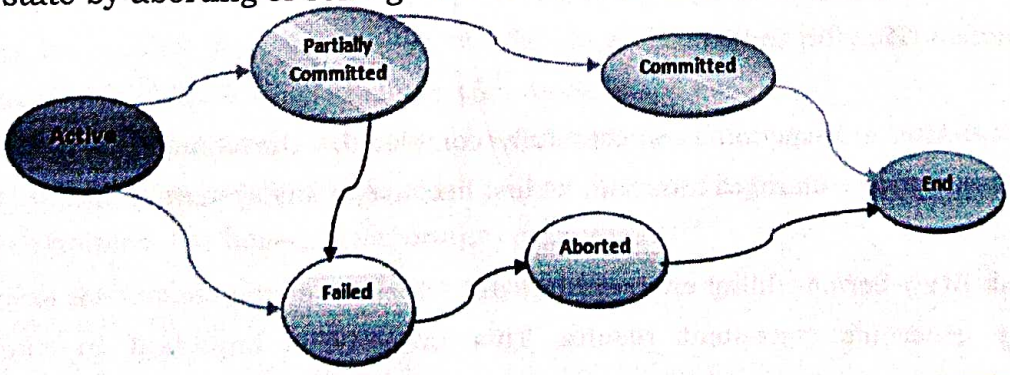
In this state, all the transactions are permanently saved to the database. This step is the last step of a transaction, if it executes without fail.

Failed

When ever the transaction is failure of the system or database, if a transaction cannot proceed to the execution state, then the transaction is said to be in failed state. In the total mark calculation example, if the database is not able fire a query to fetch the marks, i.e.; very first step of transaction, then the transaction will fail to execute.

Aborted

If a transaction is failed to execute, then the database recovery system will make sure that the database is in its previous consistent state. If not, it brings the database to consistent state by aborting or rolling back the transaction.



Transaction Management Control

A Transaction is a logical unit of work that must be entirely completed or entirely aborted, no intermediate states are acceptable. It consists of a combination of statements like select, update and insert statements.

The transaction is usually the programming language. Statements of the transaction is "Begin Transaction...

Transaction Statements....

.....

End Transaction."

The transaction ends with either commit or Rollback statement.

Transaction Properties:-The DBMS should maintain the following transaction properties to ensure data integrity. They are

1) Atomicity
2) Consistency
3) Isolation
4) Durability
5) Serializability. These properties are also called as 'ACIDS' Properties.

➔**Atomicity:-** Atomicity means all operations (SQL requests) of a transaction be completed; if not the transaction is aborted. If a transaction T1 contains four SQL requests(insert, update, delete, alter), all four requests must be successfully completed, other wise, the entire transaction is aborted.

➔**Consistency:-** Consistency means the permanence of the database's consistent state. A transaction must transform the database from one consistent state to another consistent state.

➔**Isolation:-** Isolation means that the data used during the execution of the transaction can not be used by a second transaction until the first one is completed. In other words if a transaction T1 is being executed and is using the data item X, that data item cannot be accessed by any other transaction (T2....Tn) until T1 ends.

➔ **Durability:-**After transaction is successfully completed, the changes it has made to the database persist. These changes must not be lost because of any system failure.

➔ **Serializability :-**Serializability ensures that the schedule for the concurrent execution of the transactions generates consistent results. This property is important in multi-user and distributed databases, when multiple transaction are executed concurrently.

Transaction Concept

A Transaction is a logical unit of work that must be entirely completed or entirely aborted, no intermediate states are acceptable. It consists of a combination of select, update and insert statements.

If any of the SQL statements fail, the entire transaction is rolled back to the original database state, that existed before the transaction started.

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

X's Account

1. Open_Account(X)
2. Old_Balance = X.balance
3. New_Balance = Old_Balance - 800
4. X.balance = New_Balance
5. Close_Account(X)

Y's Account

1. Open_Account(Y)
2. Old_Balance = Y.balance
3. New_Balance = Old_Balance + 800
4. Y.balance = New_Balance
5. Close_Account(Y)

Operations of Transaction:

Following are the main operations of transaction

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

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

Let's take an example to debit transaction from an account which consists of following operations:

1. 1. R(X);
2. 2. X = X - 500;
3. 3. W(X);

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

- The first operation reads X's value from database and stores it in a buffer.
- The second operation will decrease the value of X by 500. So buffer will contain 3500.
- The third operation will write the buffer's value to the database. So X's final value will be 3500.

But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

For example: If in the above transaction, the debit transaction fails after executing operation 2 then X's value will remain 4000 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations:

Commit: It is used to save the work done permanently.
Rollback: It is used to undo the work done.

Transaction Management with SQL

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: **success** or **failure**.

Example of a transaction to transfer $150 from account A to account B:

1. read(A)
2. A := A – 150
3. write(A)
4. read(B)
5. B := B + 150
6. write(B)

Incomplete steps result in the failure of the transaction. A database transaction, by definition, must be atomic, consistent, isolated and durable.

These are popularly known as ACID properties. These properties can ensure the concurrent execution of multiple transactions without conflict.

Transaction Management with SQL

The ANSI has determine SQL database transaction. Transaction supports two types of SQL Statements

i) Commit
ii) Rollback

→ A Commit statement is reached in which case all changes are permanently recorded with in the database. The commit statement automatically ends the SQL Transaction.

→ A Rollback statement is reached in which case all changes are aborted and the database is rolled back to its previous consistent state.

Ex1:- SQL> Update product set prod_price= prod_price +50 where prod_id=108;
SQL> update customer set cust_balance =cust_balance-50 where cust_id=103;
SQL> Commit;
The above transaction is permanently stored

Ex 2:- SQL> Delete from product where prod_id=1510;
1 row deleted [13 rows]
SQL> Rollback;
SQL> Select *from product;
14 rows selected

Transaction Log:

The transaction log contains information of transaction it the transaction failed. The transaction logs includes the transaction code, transaction data, time, used id etc.. By using these transaction log we can recover the data.

Implementation of Atomicity & Durability

Atomicity:- Atomicity means all operations (SQL requests) of a transaction be completed; if not the transaction is aborted. If a transaction T1 contains four SQL requests(insert, update, delete, alter), all four requests must be successfully completed, other wise, the entire transaction is aborted

Techniques to Implement Atomicity in DBMS:

Here are some common techniques used to implement atomicity in DBMS:

- **Undo Log:** An undo log is a mechanism used to keep track of the changes made by a transaction **before it is committed** to the database. If a transaction fails, the undo log is used to undo the changes made by the transaction, effectively rolling back the transaction. By doing this, the database is guaranteed to remain in a consistent condition.
- **Redo Log:** A redo log is a mechanism used to keep track of the changes made by a transaction **after it is committed** to the database. If a system failure occurs after a transaction is committed but before its changes are written to disk, the redo log can be used to redo the changes and ensure that the database is consistent.
- **Two-Phase Commit:** Two-phase commit is a protocol used to ensure that all nodes in a distributed system commit or abort a transaction together. This ensures that the transaction is executed atomically across all nodes and that the database remains consistent across the entire system.
- **Locking:** Locking is a mechanism used to prevent multiple transactions from accessing the same data concurrently. By ensuring that only one transaction can edit a specific piece of data at once, locking helps to avoid conflicts and maintain the consistency of the database.

Durability:

One of the key characteristics of transactions in database management systems (DBMS) is durability, which guarantees that changes made by a transaction once it has been committed are permanently kept in the database and will not be lost even in the case of a system failure

Implementation of Durability in DBMS:

The implementation of durability in DBMS involves several techniques to ensure that committed changes are durable and can be recovered in the event of failure.

Techniques to Implement Durability:

Here are some common techniques used to implement durability in DBMS:

- **Write-Ahead Logging:** Write-ahead logging is a mechanism used to ensure that changes made by a transaction are recorded in the redo log before they are written to the database. This makes sure that the changes are permanent and that they can be restored from the redo log in the event of a system failure.

- **Checkpointing:** Checkpointing is a technique used to periodically write the database state to disk to ensure that changes made by committed transactions are permanently stored. Checkpointing aids in minimizing the amount of work required for database recovery.
- **Redundant storage:** Redundant storage is a technique used to store multiple copies of the database or its parts, such as the redo log, on separate disks or systems. This ensures that even in the event of a disk or system failure, the data can be recovered from the redundant storage.
- **RAID:** In order to increase performance and reliability, a technology called RAID (Redundant Array of Inexpensive Disks) is used to integrate several drives into a single logical unit. RAID can be used to implement redundancy and ensure that data is durable even in the event of a disk failure.

Transaction Isolation Levels in DBMS

Isolation:-

Isolation means that the data used during the execution of the transaction can not be used by a second transaction until the first one is completed. In other words if a transaction T1 is being executed and is using the data item X, that data item cannot be accessed by any other transaction (T2....Tn) until T1 ends.

Isolation Levels in DBMS

1. Dirty Read: A dirty read is a type of situation that occurs when a transaction reads the data and that data has not been committed yet. Let us understand this with an example. Let's say transaction-1 updates a row, and that transaction 1 is not committed yet. Then transaction 2 reads the updated row. If transaction 1 rolls back the changes, then transaction-2 will read all the data, and these data will be considered as never existing.

2. Non-Repeatable Read: The non-repeatable Read occurs when the **transaction reads a single row multiple times and gets a different value** on each Read. Let's take an example. Suppose transaction1 reads the row data. But due to a concurrency issue, transaction 2 updates the same data and is committed. Now transaction 2 reads the same row again and gets a different value this time.

3. Phantom Read: When the two queries are executed, but they retrieve the two rows differently, at that time, phantom Read occurs. Let's take an example. Suppose transaction 1 retrieves a set of rows, and these rows are satisfied by some conditions. Now transaction 2 generates some new rows that match the search criteria of transaction 1. If transaction 1 executes the statement that reads the row, it gets a different row at different times

Concurrency Control

Concurrency Control: Lock-Based Protocols, Timestamp- Based Protocols, Validation-Based Protocols, Multiple Granularity

Concurrency Control with Examples

The coordination of the simultaneous execution of transactions in a multiuser database system is known as "Concurrency Control". The Objective of concurrency control is to ensure the serializability in a multiuser database environment.

Peter	Rob
1. Read Balance (Rs.1000)	1. Read Balance (Rs.1000)
2. Withdraw Rs.300 (Balance Rs.700)	2. Withdraw Rs.200 (Balance Rs.800)
3. Write Balance (Rs. 700)	3. Write Balance ➔Data integrity (Rs.500)

Allowing multiple transactions simultaneously against the database, many problems occurs. The main problems are

1) Lost updates
2) Uncommitted data
3) Inconsistent retrievals

1) Lost updates:- The lost update problem occurs when two concurrent transaction, T1 and T2 are updating the same data element and one of the update is lost.

Assume that Prod_Qoh=35

i) The concurrent transaction to update Prod_Qoh

Transaction	Computation
T1 : Purchase 100 Units	Prod_Qoh = Prod_Qoh + 100
T2 : Sell 30Units	Prod_Qoh = Prod_Qoh - 30

ii) Serial execution of two transactions

Time	Transaction	Step	Stored Values
1	T1	Read Prod_Qoh	35
2	T1	Prod_Qoh = 35+100	
3	T1	Write Prod_Qoh	135
4	T2	Read Prod_Qoh	135
5	T2	Prod_Qoh = 135-30	
6	T2	Write Prod_Qoh	105

iii) Lost Update

Time	Transaction	Step	Stored Values
1	T1	Read Prod_Qoh	35
2	T2	Read Prod_Qoh	35
3	T1	Prod_Qoh = 35+100	
4	T2	Prod_Qoh = 35-30	
5	T1	Write Prod_Qoh	135(Lost update)
6	T2	Write Prod_Qoh	5

2) Uncommitted Data:- The uncommitted data problem occurs when two transactions, T1 and T2 are executed concurrently and the first transaction T1 is rolled back, after the second transaction T2 has already accessed the uncommitted data. Assume that Prod_Qoh = 35

i) The concurrent transaction to uncommitted data problem

Transaction	Computation
T1 : Purchase 100 Units	Prod_Qoh = Prod_Qoh + 100
T2 : Sell 30Units	Prod_Qoh = Prod_Qoh - 30

ii) Serial Execution of two transactions

Time	Transaction	Step	Stored Values
1	T1	Read Prod_Qoh	35
2	T1	Prod_Qoh = 35+100	
3	T1	Write Prod_Qoh	135
4	T1	***ROLLBACK***	35
5	T2	Read Prod_Qoh	35
6	T2	Prod_Qoh = 35-30	
7	T2	Write Prod_Qoh	5

iii) Uncommitted data problem:

Time	Transaction	Step	Stored Values
1	T1	Read Prod_Qoh	35
2	T1	Prod_Qoh = 35+100	
3	T1	Write Prod_Qoh	135
4	T2	Read Prod_Qoh (Read uncommitted data)	135
5	T2	Prod_Qoh = 135-30	
6	T1	***ROLLBACK***	35
7	T2	Write Prod_Qoh	105

3) Inconsistent Retrievals:-

Inconsistent retrievals problem occur when a transaction access data before and after another transaction uses the same data item.

Ex:- There are two transactions T1 and T2, T1 is trying to calculate the sum of Prod_Qoh for all the products and T2 is trying to update the same Prod_Qoh of two records in the table.

Transaction	Computation
T2	Update Prod_Qoh = Prod_Qoh +10 where prod_code="P102" Update prod_Qoh = prod_Qoh -15 where prod_code= "P103"
T1	Select sum(prod_Qoh) from product;

The below table show the quantity of four products and their totals

	BEFORE	AFTER
PROD_CODE	**PROD_QOH**	**PROD_QOH**
P101	8	8
P102	15	(15+10)→25
P103	23	(23-10)→13
P104	6	6
TOTAL	**52**	**52**

Inconsistent Retrievals:

TIME	TRANS-ACTION	ACTION	VALUE	TOTAL
1	T1	Read Prod_Qoh for Prod_code= "p101"	8	8
2	T2	Read Prod_Qoh for Prod_code= "p102"	15	
3	T2	Prod_Qoh= 15+10		
4	T2	Write Prod_Qoh for Prod_code= "p102"	25	
5	T1	Read Prod_Qoh for Prod_code= "p102"	25	33 (After)
6	T1	Read Prod_Qoh for Prod_code= "p103"	23	56 (Before)
7	T2	Read Prod_Qoh for Prod_code= "p103"	23	
8	T2	Prod_Qoh = 23-10		
9	T2	Write Prod_Qoh for Prod_Code="p103"	13	
10	T2	****COMMIT ****		
11	T1	Read Prod_Qoh for Prod_code= "p104"		
			6	62

The above example the transaction T1 calculates the sum of the prod_qoh as 62, but its actual total values is only 52.

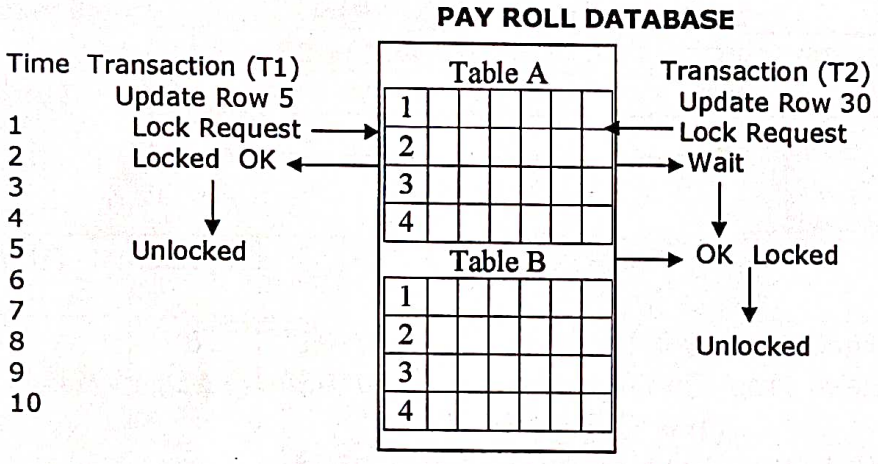
Concurrent Access Control Locking Methods / Lock Granularity/ Lock Types & Two Phase Locking/ Lock-Based Protocols

A lock is mechanism to control concurrent access to a data item. If one transaction is accessing the data item no other transaction are allowed to modify that data item. In other words transaction T2 does not have access to a data item, that is currently being used by transaction T1.

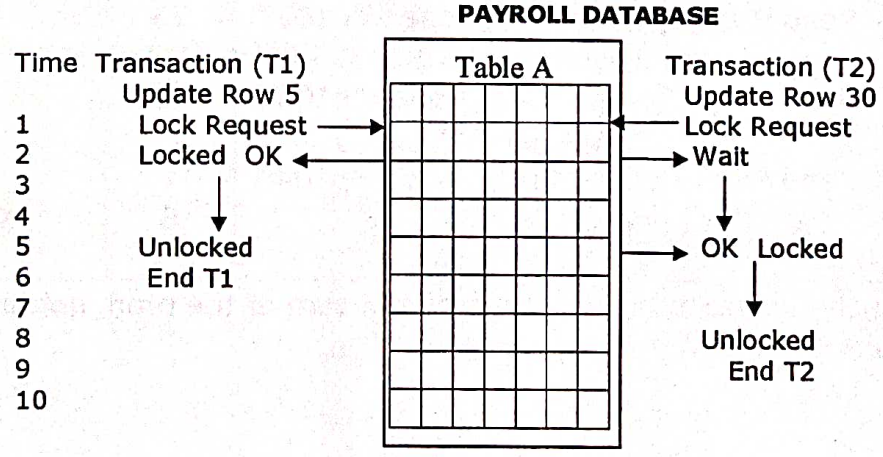
1) **Lock Granularity:-** Lock Granularity indicates the level of lock use. Locking can take place at the following levels

- ➔ Database level lock
- ➔ Table level lock
- ➔ Page level lock
- ➔ Row level lock
- ➔ Field level lock

i) Database level lock:- In a database level lock the entire database is locked, thus preventing the use of any tables in the database by transaction T2, While transaction T1 is being executed.



ii) Table Level Lock:- In a table level lock the entire table is locked thus preventing access to any row by transaction T2, while transaction T1 is using the table.



iii) Page Level Lock:- In a page level lock, the DBMS will lock an entire disk page. A disk page is equivalent of a disk block. A page has a fixed sixe such as 4KB, 8KB, 16KB.....etc.

iv) Row Level Lock: A row level lock is much less restrictions than the previous locking levels. Here only the requested record or row will be locked. Remaining records or rows are available to other transactions

v) Field Level Lock:- In a field level lock only a particular field or column in a requested record will be locked. Remaining all fields are available to other transactions.

2) Lock Types:- The DBMS may use different types of locks

→ Binary Locks

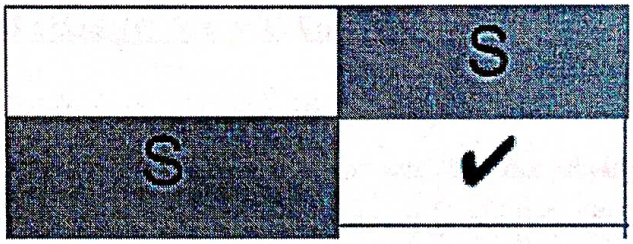
→ Shared / Exclusive Locks

I) Binary Locks:- A Binary lock has only two states i.e locked (1) and unlocked (0). If an object (database, page, table) is locked by a transaction no other transactions can use that object.

ii) Shared / Exclusive Locks:- A lock can have 3 states they are Unlocked, Shared(read), and Exclusive(write)

A shared lock is issued when a transaction wants to read data item. And no exclusive lock is held on that data item. This lock only the **Read** operations, but not provides write operations. It is also called Read Lock and S-Lock

An exclusive lock is issued, when a transaction wants to **Read And Write** operations on data item. It is also called Write Lock and X-Lock



Two Phase Locking

Two phase locking defines how transactions acquire(gain) and release the locks. Any transaction request to new lock, it has two conditions. They are

→ Growing Phase

→ Shrinking Phase

In Growing phase the transaction acquires all required locks, with out unlocking any data.
In Shrinking phase the transaction releases all locks and cannot obtain any new lock

Two-Phase Locking

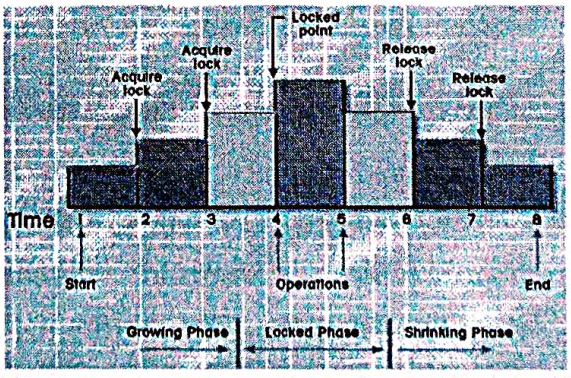


Figure 9.6 Two-Phase Locking Protocol

32

Q. What is dead lock?

A dead lock occurs when two transactions wait indefinitely (no specific time) for each other to unlock data item.

Ex:- consider the two transactions

T1 = Access data items X and Y

T2 = Access data items Y and X

Time	Transaction	Replay	Lock Status	
			Data X	Data Y
1	T1:LOCK(X)	OK	UNLOCKED	UNLOCKED
2	T2:LOCK(Y)	OK	LOCKED	UNLOCKED
3	T1:LOCK(Y)	WAIT	LOCKED	LOCKED
4	T2:LOCK(X)	WAIT	LOCKED	LOCKED
5	T1:LOCK(Y)	WAIT	LOCKED	LOCKED
6	T2:LOCK(X)	WAIT	LOCKED	LOCKED

Dead lock

In the above example If T1 has not unlocked data item X, T2 cannot begin. Similarly if T2 has not unlocked data item Y, T1 cannot continue. As a result T1 & T2 are waiting for each other indefinitely. This is called "Dead Lock".

Concurrency Control Without Locking

Concurrency Control

The coordination of the simultaneous execution of transactions in a multiuser database system is known as "Concurrency Control". To Control the concurrency without locking the following methods are used. They are

1) Optimistic Method
2) Time stamping Method

1) Optimistic Method:-

The execution of transactions simultaneously performed against the Database is called Concurrency.

i) It is most important method for control in the concurrency.

ii) This method is not using locking and time stamping approaches.

iii) In this method the transaction can transfer the following phases.

a. Read phase
b. Validation phase
c. Write phase

a) Read phase: In this phase the transaction reads the Database, executes and updates are stored in private temporary files. These files cannot accessed by remaining transactions.

b) Validation phase: During this phase checks the transactions is validated or not. If the validation test is positive then transaction goes to write phase otherwise the transaction is restarted.

c) Write phase: In this phase by using the private temporary updated files and changes are permanently applied on database.

2) Time stamping Method:

Time stamping method is used to control the concurrent transactions by assigning a unique global time stamping value to each transactions.

The time stamping value produces an order in which transaction are executed. Every time stamp has 2 properties. They are

a) Uniqueness
b) Monotonicity

a) Uniqueness: It ensures no equal time stamp values can exists.

b) Monotonicity: It ensures that the time stamp values always increasable.

If two transactions are conflict, one is stopped, roll backed, rescheduled and assigned a new time stamp value. The disadvantage of timestamp approach is that, it demands lot of memory because many transactions are stopped, rescheduled, roll backed, and re-stamped.

In time stamping method, there are 2 schemes are used to decide which transaction rolled back and which transaction is execute. Those schemes are.

1) Wait/die
2) Wound/wait

1) Wait/die: In this schema

- If the older timestamp transaction requesting the lock, it will wait until other transaction is completed.
- If the younger timestamp transaction requesting the lock, it will die and rescheduled using same time stamp.

2) Wound/Wait: In this schema

- If the older time stamp transaction requesting the lock, it will wound (rolled back) by younger transaction. The younger transaction. Rescheduled using time stamp
- If the younger time transaction requesting the lock, it will wait until other transaction is completed.

Ex: If there are two transactions T1 & T2 are executes simultaneously using time stamping method then the transaction T1 has "1154" time stamp and T2 has "1954" time stamp. We can observe these two transactions T1 is old transaction i.e., it has lower time stamp value and T2 is younger transaction because it has higher time stamp value.

The four possible outcomes are shown below.

Transaction requesting lock	Transaction owning lock	Wait/Die schema	Wound/Wait schema
T1(1154)	T2(1954)	T1 waits until T2 is completed	T1 wound by T2 T2 is rescheduled using same time stamp value.
T2(1954)	T1(1154)	T2 dies and It is rescheduled same timestamp value	T2 waits until T1 is completed.

Now we can observe the above table the both schemas one of the transaction waits for the other transaction to finish and release the lock.

Recovery System

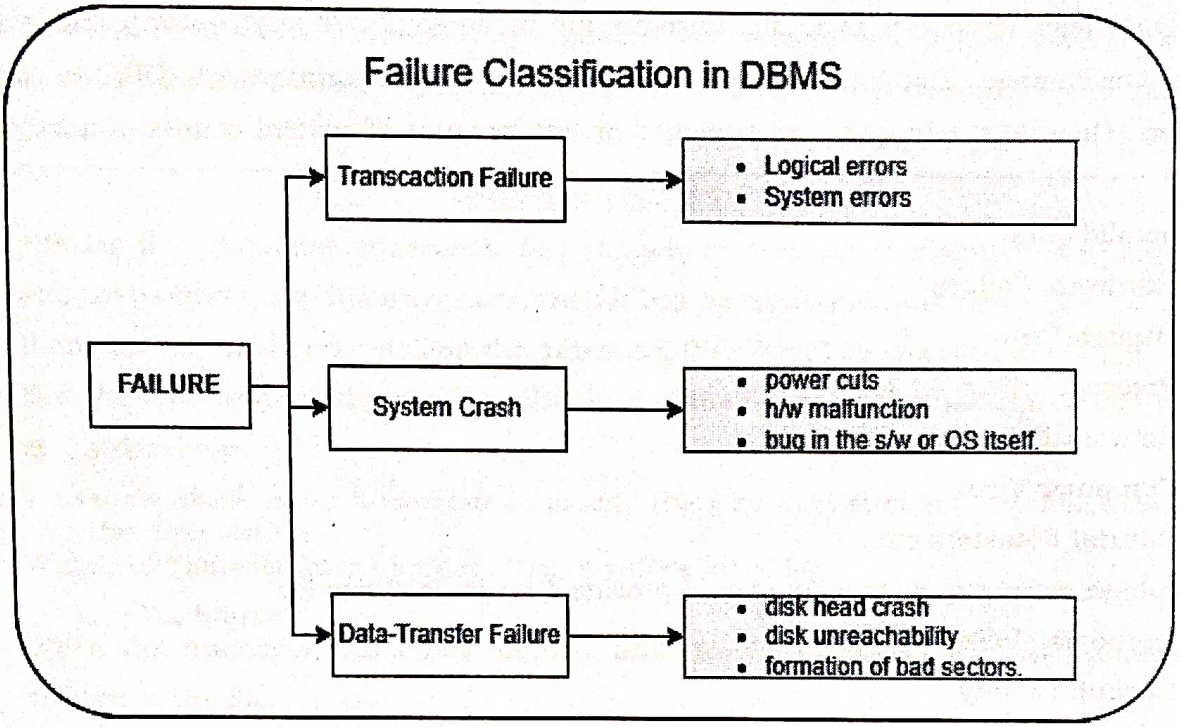
Recovery: Failure Classification, Recovery and Atomicity, Log-Based Recovery.

Failure Classification

Failure in terms of a database can be defined as its inability to execute the specified transaction or loss of data from the database. There are many reasons that can cause database failures such as network failure, system crash, natural disasters, carelessness, sabotage(corrupting the data intentionally), software errors, etc.

Failure Classification in DBMS

A failure in DBMS can be classified as:



1) Transaction Failure: If a transaction is not able to execute or failure due to the following reasons.

a. Logical error: A logical error occurs if a transaction is unable to execute because of **some mistakes in the code or due to the presence of some internal faults.**

b. System error: Where the termination of an active transaction is done by the database system itself due to some system issue or because the database management system is unable to proceed with the transaction. *For example*– The system ends an operating transaction if it reaches a deadlock condition or if there is an unavailability of resources.

2) System Crash:

A system crash usually occurs when there is some sort of hardware or software breakdown. Some other problems which are external to the system and cause the system to abruptly stop or eventually crash include failure of the transaction, operating system errors, power cuts, main memory crash, etc.

3. Data-transfer Failure:

When a disk failure occurs amid data-transfer operation resulting in loss of content from disk storage then such failures are categorized as data-transfer failures. Some other reason for disk failures includes disk head crash, disk unreachability, formation of bad sectors, read-write errors on the disk, etc.

In order to quickly recover from a disk failure caused amid a data-transfer operation, the backup copy of the data stored on other tapes or disks can be used. Thus it's a good practice to backup your data frequently.

Database Recovery Management

Database Recovery is a mechanism for restoring database quickly and accurately after loss or damage. Database recovery is the database administrators job (DBA) to restore the database. Generally databases are damaged or lost because of system causes of failures. They are

i) Invalid data
ii) Hardware Failure
iii) Human Errors
iv) Program Errors
v) Network Failure
vi) Computer Virus
vii) Natural Disasters etc..

The database management system should provide 4 basic facilities for restoring the database. They are

1) Backup Facility
2) Journalizing Facility
3) Checkpoint Facility
4) Recovery Manager Facility

1) Backup Facility:- The DBMS should provide backup facility that produce a backup copy of entire database, generally a backup copy is produced at least once per day, here the backup should be stored in a secured location, where it is protected from loss or damage. The backup copy is used to restore the database in the event of failures or damages.

2) Journalizing Facility:- The DBMS should provide some journalizing facilities to produce backup. In the event of failures, the DBA can reestablish the database using the information stored in the journals (files or logs). This process contains two basic type of journals they are

a) Transaction Log
b) Database change log

a) Transaction log:- The transaction log contains the information of a transaction. The transaction log includes the transaction code, transaction date, time, user id, and so on.

b) Database change log:- The data base change log contains before and after information of records, that have been modified by a transaction.

3) Check point facility:- Check points are also called save points. It contains necessary information to restore the database from update buffers. The DBA creates different check points in application programs.

4) Recovery Manager:- A recovery manager is a program of DBMS. Which restores the database when a failure or damage occurs. There are 2 types

1) Deferred update

2) Immediate update

i) Deferred update:- This technique **do not immediately update the physical database until transaction reaches its commit point.** Before reaching the commit point all transaction updates are recorded in the buffers. During commit those buffers are written in the database.

ii) immediate update:- This technique is also known as the undo/redo algorithm. **In this technique the database is immediately updated by transaction,** before reaches its commit point. If the transaction fails before reaches its commit point, then rollback operation restores the database.

Log-Based Recovery

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Let's assume there is a transaction to modify the City of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.

  1. <Tn, Start>

- When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.

  1. <Tn, City, 'Noida', 'Bangalore' >

- When the transaction is finished, then it writes another log to indicate the end of the transaction.

  1. <Tn, Commit>

There are two approaches to modify the database:

1. Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

2. Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

Recovery using Log records

When the system is crashed, then the system consults the log to find which transactions need to be undone and which need to be redone.

1. If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.
2. If log contains record<T_n, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

Recovery Algorithm (ARIES)

Algorithms for Recovery and Isolation Exploiting Semantics

ARIES is a recovery algorithm. It is more simple and flexible than other algorithms.

ARIES algorithm is used by the recovery manager which is invoked after a crash. Recovery manager will perform restart operations.

Main Key Terms Used in ARIES

Log. Log contains the information about a failed transaction, The transaction logs includes the transaction code, transaction data, time, used id etc.. By using these transaction log we can recover the data. Normally copy of the log file placed in the different parts of the disk for safety.

LSN. The LSN means Log Sequence Number. It is the ID given to each record in the log file of database. It is used for recovery purpose. Every page in the database contains the LSN. The LSN is also called page LSN.

CLR. The CLR means Compensation Log Record. It is used to store the before changed record details. i.e the action taking during a rollback. These data is used to recovery process to repeat history (redo) first

WAL. The WAL means Write-Ahead Log. Before updating a page to disk, every update log record is stored into WAL.

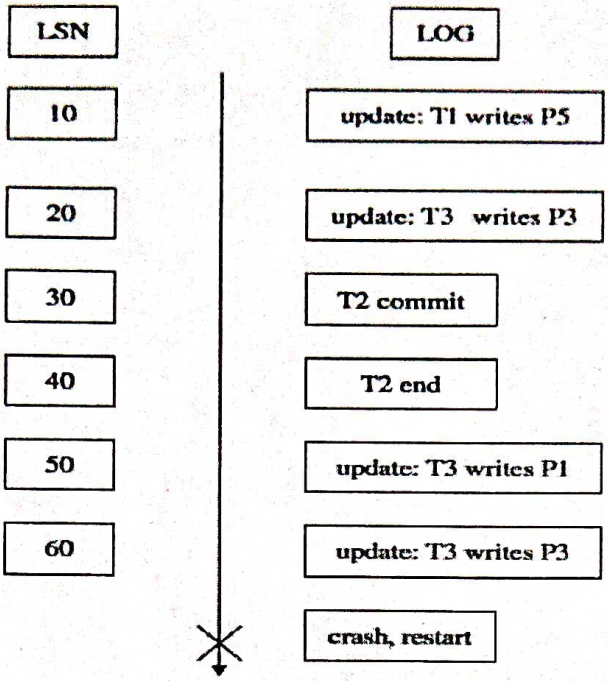
There are three phases in restarting process, they are:

1. Analysis
2. Redo
3. Undo

1. Analysis. In this phase, it will identify whether any page present in the buffer pool is not written into disk and activate the transactions which are in active at the time of crash.

2. Redo. In this phase, all the operations are restarted and the state of database at the time of crash is obtained. This is done by the help of log files.

3. Undo. In this phase, the actions of uncommitted transactions are undone. So only committed are taken into account.



Example:-

In the above diagram, when the system is restarted, the ***Analysis*** phase identifies T1 and T3 transactions are undone. T2 as a committed transaction, So, T2 is written to disk.

P1, P3, and P5 as potentially dirty pages (not yet written into disk). All the updates are reapplied during the ***Redo*** phase. Finally the actions of T1 and T3 are undone in reverse order during the ***Undo*** phase; that is, T3's write of P3 is undone, T3's write of P1 is undone, and then T1's write of P5 is undone.

→ END ←

If you want success every time in your life, then ***Fight With Time Until You Die***!! ***That's It***! Otherwise if you want to be with every one, then ***Travel With Time!***

Prepared by

DVH. Venu Kumar

M.Sc.,M.Ed.,M.Tech(CSE)

Asst.Professor., Dept.of CSE., Geethanjali Institute of Science and Technology : Gangavaram : Nellore