Computer

Chapter (9): DBMS

Er.Dhiredra k. Yadav

DBMS CH-9

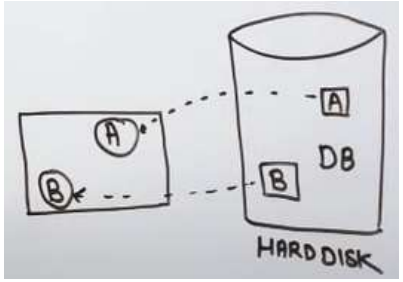
OUTLINE

- Introduction

DBMS_CH_9

❑ Introduction to crash recovery

- Crash recovery in database systems refers to the processes and mechanisms implemented to r estore the database to a consistent state after a failure.
- The goal of crash recovery is to ensure the database remains reliable and that all transactions are either fully completed or fully undone, thereby preserving the integrity of the data.



▪ Importance of Crash Recovery

- Data Integrity: it ensures that the data remains accurate and consistent after a failure.
- Transaction Atomicity: Guarantees that transactions are either completely processed or not proc essed at all.
- System Reliability: Database system ability to recover from failures.
- Minimized Downtime: Reduces the downtime of the database system after a crash, allowing for quick recovery and resumption of operations.
- User Trust: Builds trust among users and stakeholders by ensuring that their data is safe and reli able, even in the event of system failures.

DBMS_CH_9

❑ Failures

– Types

1. Transaction Failures:

   A transaction cannot complete successfully due to logical errors, constraints violations, or deadlocks.
2. System Failures: software bugs, other internal issues.
3. Media Failures: hard drives or SSDs) fail
4. Natural Disasters: fires, floods, or earthquake
5. Logical Error: Bad instruction, Data Notified, Overflow
6. System error: Deadlock.

▪ Transaction Failures

– Cause: Logical errors, integrity constraint violations, deadlocks.

– Recovery Mechanism:

- Rollback: The DBMS undoes all changes made by the failed transaction using the log recor ds.
- Redo: In some cases, the system may reapply the transaction from the beginning if it was int errupted.

DBMS_CH_9

- **System Failures**

– Cause: Operating system crashes, power failures, software bugs.

– Recovery Mechanism:

  - Checkpointing: Periodically saving the state of the database. After a crash, the system can roll forward from the last checkpoint using the transaction log.
  - Logging: Maintaining a log of all transaction operations. After a crash, the system uses the log to redo committed transactions and undo uncommitted transactions.

- **Media Failures**

– Cause: Disk failures, data corruption.

– Recovery Mechanism:

  - Backup and Restore: Regular backups of the database are restored, followed by applyi ng the transaction log to bring the database up to date.
  - RAID Technology: Using redundant arrays of independent disks (RAID) to protect aga inst disk failures by mirroring data across multiple disks.

- **Natural Disasters**

– Cause: Fires, floods, earthquakes.

– Recovery Mechanism:

  - Disaster Recovery Plans: Having off-site backups, cloud storage, and restore database operations.
  - Redundant Data Centers: Using geographically dispersed data centers to ensure data availability even if one site is affected by a disaster.

DBMS_CH_9

- **Backup:**

– This involves creating copies of the database that can be used to restore the original after a data loss event. There are different types of backups:

1. Full Backup: Copies the entire database.
2. Incremental Backup: Copies only the data that has changed since the last backup.
3. Differential Backup: Copies all the data that has changed since the last full backup.

- **Recovery:**

– This involves restoring the database from a backup and applying any necessary transactions to bring the database up to date. The recovery process may include:

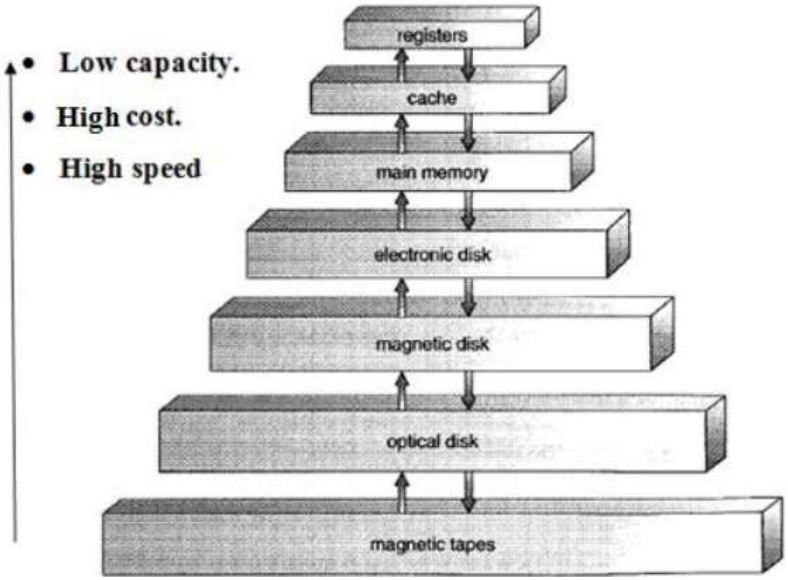
1. Restoring from Backup: Using the backup copies to restore the database.
2. Rolling Forward: Applying changes from the log files to bring the database up to date.
3. Rolling Backward: Undoing uncommitted transactions using the log files to ensure data integrity.

DBMS_CH_9

❑ Storage Hierarchy

– The storage hierarchy is the organization of storage media in a hierarchical manner based on speed, cost, and volatility.

1. Registers: Small, extremely fast storage locations within the CPU.
2. Cache: Fast but small memory used to temporarily hold data that is frequently accessed.
3. Main Memory (RAM): it used by the CPU to hold data and instructions that are currently being processed.
4. Secondary Storage (Hard Drives, SSDs): Persistent storage used to hold data permanently.
5. Tertiary Storage (Optical Discs, Magnetic Tapes): Used for backups and archival purposes, slower and cheaper
6. Off-line Storage (Cloud Storage, External Drives): Storage that is not directly connected to the system and is used for backup and data transfer.



DBMS_CH_9

❑ Transaction Model

– A transaction in database management is a sequence of operations performed as a single logical unit of work. A transaction must exhibit the ACID properties:

1. Atomicity: All operations within the transaction are completed; if not, the transaction is aborted, and all operations are rolled back.
2. Consistency: The database transitions from one valid state to another valid state.
3. Isolation: The operations of a transaction are isolated from those of other transactions.
4. Durability: once a transaction is committed, its changes are permanent, even in the case of a system failure.

DBMS_CH_9

❑ Log-Based Recovery

- Log-based recovery uses a log file to keep a records of all transactions and their operations.
- it is a sequence of records i.e it contain complete records of DB activates
- Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- In the event of a failure, the log file is used to restore the database to a consistent state.
- 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.

- The following logs are written for this transaction.

1. When the transaction is initiated, then it writes 'start' log.
   - <Tn, Start>
2. When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.
   - <Tn, City, 'Noida', 'Bangalore' >
3. When the transaction is finished, then it writes another log to indicate the end of the transaction.
   - <Tn, Commit>
4. When a transaction is aborted due to error
   - <Tn, ABORTED)

- i.e

  <Ti START> When Transaction Ti Start

  <Ti, x, v1,v2 > Transaction Ti has Performed where Operation On xj. Xj have old value V1 bef ore update & v2 Updated value

  <Ti COMMIT> Ti Has commited

  <Ti ABORT> Ti has Aborted.

eg.

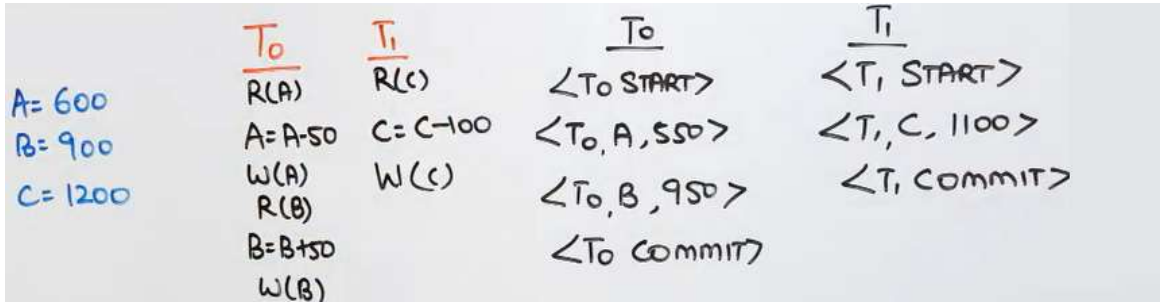
- for following instruction A =5, B= 7, Read(A),Read(B) , A = A+B =12; and write A

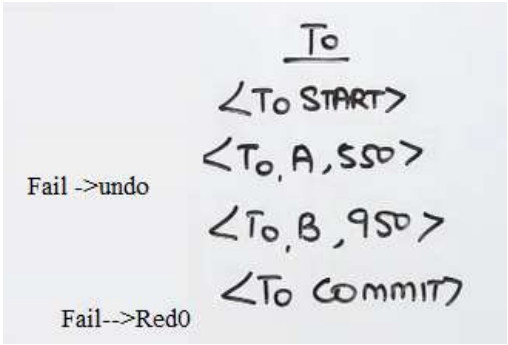
  <T1 START>

  <T1, A, 5, 12>

  <T1, COMMIT>

- DEFEREED DATABASE MODIFICATION
- This Technique ensures Transaction atomicity by Recording all DB modification in LOG, but D EFERRING the execution of write operation until Transaction partially COMMIT.
- Eg.

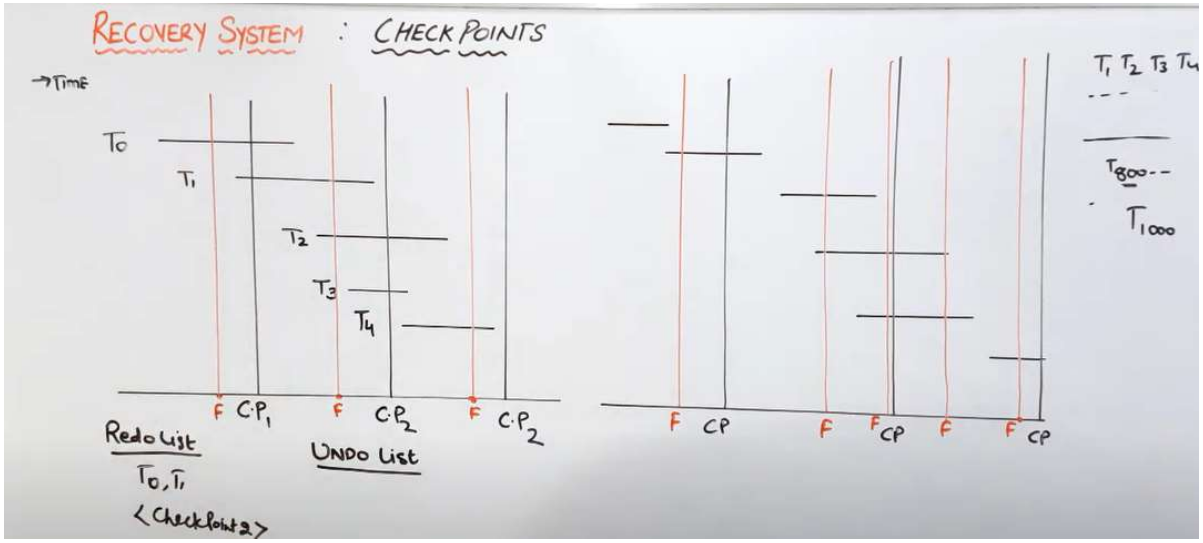




DBMS_CH_9

❑ Checkpoint

- The checkpoint is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.
- The transaction is executed using checke point then the log files will be created.
- When transaction reaches to the checkpoint, then the transaction will be updated into the databa se, The entire log file will be removed from the file.
- The checkpoint is used to declare a point before which the DBMS was in the consistent state, an d all transactions were committed.
- The recovery system reads log files from the end to start.
- Recovery system maintains two lists, a redo-list, and an undo-list.
- The transaction is put into redo state if the recovery system sees a log with <Tn, Start> and <Tn, Commit> or just <Tn, Commit>. In the redo-list and their previous list, all the transactions are r emoved



DBMS_CH_9

❑ Shadow Paging

- Shadow paging is a recovery technique that avoids the use of logs. It involves maintaining two c opies of the database: the current page table and the shadow page table.

- Shadow Page Table: Represents the state of the database before the transaction began.
- Current Page Table: Represents the state of the database after the transaction.
- Commit Operation: When a transaction commits, the current page table is written to disk, replac ing the shadow page table.
- Abort Operation: If a transaction aborts, the shadow page table is used to restore the database to its previous state.

Chapter 10:

❑ Concurrency

- Concurrency control is a database management technique used to ensure that transactions are executed concurrently without violating the consistency of the database. It prevents conflicts that can arise when multiple transactions access the same data simultaneously.

DBMS_CH_10

❑ Transaction

- The transaction is a set of logically related operation. It contains a group of tasks(Single logical unit of work)
- A transaction is an action or series of actions. It is performed by a single user to perform operati ons for accessing the contents of the database.
- A transaction must exhibit the ACID properties.

X's Account	Y's Account
Open_Account(X)	Open_Account(Y)
Old_Balance = X.balance	Old_Balance = Y.balance
New_Balance = Old_Balance - 800	New_Balance = Old_Balance + 800
X.balance = New_Balance	Y.balance = New_Balance
Close_Account(X)	Close_Account(Y)

- Operations of Transaction:

- Read(X): Read operation is used to read the value of X from the database and stores it in a buffe r in main memory.
- Write(X): Write operation is used to write the value back to the database from the buffer.
- Commit: It is used to save the work done permanently.
- Rollback: It is used to undo the work done.

DBMS_CH_10

- Example of Transaction.

```
1. R(X);
2. X = X - 500;
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.

- Transaction may fail because failure of hardware, software or power before finished all the operations in the set.

- **Transaction property**

- The transaction has the four properties
    1. Atomicity
    2. Consistency
    3. Isolation
    4. Durability

- Atomicity : Ensures that all operations within the transaction are completed; if not, the transaction is aborted, and all operations are rolled back. i.e means either all successful or not
- Consistency: Ensures that the database transitions from one valid state to another valid state.
- Isolation: Ensures that the operations of a transaction are isolated from those of other transactions.
- Durability: Ensures that once a transaction is committed, its changes are permanent, even in the case of a system failure.

- Example 2. : Let's assume that following transaction T consisting of T1 and T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

T1	T2
Read(A)	Read(B)
A:= A-100	Y:= Y+100
Write(A)	Write(B)

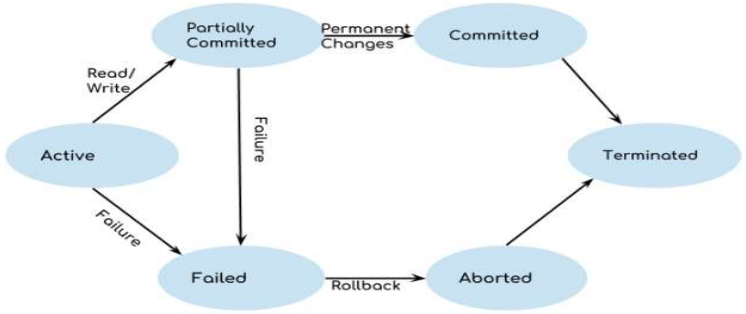
- After completion of the transaction, A consists of Rs 500 and B consists of Rs 400.
- If the transaction T fails after the completion of transaction T1 but before completion of transact ion T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state need atomicity.
- The total amount must be maintained before or after the transaction need consistency
-

Total before T occurs = 600+300=900

Total after T occurs= 500+400=900

DBMS_CH_10

- States of Transaction
- In a database, the transaction can be in one of the following states



1. Active state

- The active state is the first state of every transaction. In this state, the transaction is being executed.
- Data to e called in primary memory i.e data reading
- For example: Insertion or deletion or updating a record is done

2. Partially committed

- In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
- In the total mark calculation example, a final display of the total marks step is executed in this state.

3. Committed

- A transaction is said to be in a committed state if it executes all its operations successfully. In this stat e, all the effects are now permanently saved on the database system.

4. Failed state

- If any of the checks made by the database recovery system fails, then the transaction is said to be in th e failed state.
- In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, t hen the transaction will fail to execute.

5. Aborted

- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state.
- If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two
- operations:
  1. Re-start the transaction
  2. Kill the transaction

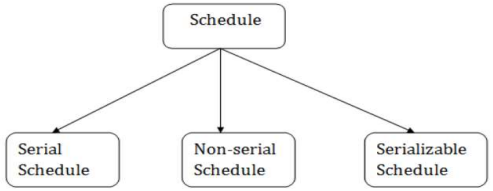
6. Aborted :

- Transaction is completed and resources are freed or deallocated.

DBMS_CH_10

❑ Schedule

- A series of operation from one transaction to another transaction is known as schedule.
- It is used to preserve the order of the operation in each of the individual transaction.



1. Serial Schedule

- The serial schedule is a type of schedule where one transaction is executed completely before st arting another transaction.
- In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

2. Non-serial Schedule

- If interleaving of operations is allowed, then there will be non-serial schedul
- It contains many possible orders in which the system can execute the individual operations of th e transactions.

3. Serializable schedule

- The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.
- It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
- A non-serial schedule will be serializable if its result is equal to the result of its transactions exe cuted serially.

- **Eg.**

(a)

Time

T1	T2

read(A);
A := A – N;
write(A);
read(B);
B := B + N;
write(B);	

	read(A);
	A := A + M;
	write(A);

Schedule A

Serial

(b)

Time

T1	T2

	read(A);
	A := A + M;
	write(A);

read(A);
A:= A - N;
write(A);
read(B);
B:= B +N;
write(B);	

Schedule B

(c)

Time

T1	T2
read(A); A := A – N;	
	read(A); A := A + M;
write(A); read(B);	
	write(A);
B := B + N; write(B);	

Schedule C

Non-serial schedules

(d)

Time

T1	T2

read(A);
A := A – N;
write(A);	

	read(A);
	A := A + M;
	write(A);

read(B);
B := B + N;
write(B);	

Schedule D

- **Eg.**

(a)

Time

T1	T2

read(A);
A := A – N;
write(A);
read(B);
B := B + N;
write(B);	

	read(A);
	A := A + M;
	write(A);

Schedule A

Serial

(b)

Time

T1	T2

	read(A);
	A := A + M;
	write(A);

read(A);
A:= A - N;
write(A);
read(B);
B:= B +N;
write(B);	

Schedule B

(c)

Time

T1	T2
read(A); A := A – N;	
	read(A); A := A + M;
write(A); read(B);	
	write(A);
B := B + N; write(B);	

Schedule C

Non-serial schedules

(d)

Time

T1	T2

read(A);
A := A – N;
write(A);	

	read(A);
	A := A + M;
	write(A);

read(B);
B := B + N;
write(B);	

Schedule D

DBMS_CH_10

- **Locking and Lock-Based Protocols**

– Locking is a mechanism used to control concurrent access to data in a database.

– Locks can be shared (read) or exclusive (write). Lock-based protocols are rules governing how transactions acquire and release locks.

– Two-Phase Locking (2PL) is a common lock-based protocol that ensures serializability:

1. Growing Phase: A transaction may acquire locks but not release any.
2. Shrinking Phase: A transaction may release locks but not acquire any.

- Strict Two-Phase Locking is a variation where transactions hold all their exclusive locks until th ey commit or abort, ensuring that no other transaction can access the modified data before the tr ansaction completes.

- Time-Stamp-Based Protocols

– Time-stamping-based protocols assign a unique timestamp to each transaction, determining the order of execution.

– Read Timestamp (RTS): The largest timestamp of any transaction that has successfully read the item.

– Write Timestamp (WTS): The largest timestamp of any transaction that has successfully written the item.

Basic Time-Stamp Ordering:

- If a transaction T_i wants to read a data item:
  - If $WTS(x) > T_i$'s timestamp, T_i is rolled back.
  - Otherwise, T_i reads the data item, and $RTS(x)$ is set to the maximum of $RTS(x)$ and T_i 's timestamp.
- If a transaction T_i wants to write a data item:
  - If $RTS(x) > T_i$'s timestamp or $WTS(x) > T_i$'s timestamp, T_i is rolled back.
  - Otherwise, T_i writes the data item, and $WTS(x)$ is set to T_i's timestamp.

❑ Deadlock Handling

– Deadlock occurs when two or more transactions are waiting indefinitely for each other to release locks. There are several techniques to handle deadlocks:

– Deadlock Prevention: Ensures that the system will never enter a deadlock state by imposing restrictions (e.g., no transaction can hold a lock and request another).

– Deadlock Detection: Periodically checks for deadlocks and takes action when one is detected (e.g., aborting one of the transactions).

– Deadlock Recovery: Involves rolling back one or more transactions to break the deadlock, often choosing the transaction that has done the least amount of work (victim selection).

Multiple Granularity

- Multiple Granularity allows transactions to lock data items at various levels of granularity, such as tuples, pages, tables, or the entire database. This approach improves the concurrency and flex ibility of locking mechanisms.
  - Intention Locks: Used to indicate that a higher-level node in the hierarchy will be lock ed in a particular mode.
  - Intention Shared (IS): Indicates that the transaction intends to acquire shared locks on some lower-level nodes.
  - Intention Exclusive (IX): Indicates that the transaction intends to acquire exclusive locks on some lower-level nodes.
  - Shared Intention Exclusive (SIX): Indicates that the transaction holds a shared lock on the higher-level node and intends to acquire exclusive locks on some lower-level node s.

Thank you