CHAPTER 4

Crash Recovery

Contents...

Objectives...

- To understand Concepts in Crash Recovery
- To learn Recovery Concepts in Database
- To study Database Backup and Recovery Techniques
- To learn Buffer Manager and Buffer Cache with their Relationship
- To study Recovery from Catastrophic Failure

4.0 INTRODUCTION

- A crash is the sudden failure of a software application or operating system or of a hardware device such as a hard disk.
- An important feature of conventional database systems is the ability to recover from system crashes. Recovery means restoring data into its previous state before the incident.
- Database recovery is the process of restoring the database to a correct (consistent) state in the event of a failure.
- In other words, it is the process of restoring the database to the most recent consistent state that existed shortly before the time of system failure.

- A failure of a DBMS occurs when the database system does not meet its specification, i.e., the database is in an inconsistent state. Recovery is required to protect the database from data inconsistencies and data loss.
- Transactions (or units of work) against a database can be interrupted unexpectedly. If a failure occurs before al of the changes that are part of the unit of work are completed, committed, and written to disk, the database is left in an inconsistent and unable state.
- Crash recovery is the process by which the database is moved back to a consistent and unable state. This is done by rolling back incomplete transactions and completing committed transactions that were still in memory when the crash occurred.
- The recovery process in database is often closely intertwined with operating system functions - in particular, the buffering and catching of disk pages in main memory.
- Typically, one or more disk pages that include the data items to be updated are cached into main memory buffers and then updated in memory before being written back to the disk.
- The caching of disk pages is traditionally an operating system function, but because of its importance to the efficiency of recovery procedures, it is handled by the DBMS by calling low-level operating systems routines.
- Typically a collection of in-memory buffers, called the DBMS cache, is kept under the control of the DBMS for the purpose of holding these buffers.
- Database backup and recovery techniques are designed to protect the database against data loss and to reconstruct the database after data loss.

4.1 FAILURE CLASSIFICATION

- Failure refers to the state when system can no longer continue with its normal execution and that results in loss of information. There are different types of failure that may occur in a system, each of failure needs to be dealt with in a different manner.
- The simplest type of failure to deal with is one that does not result in the loss of information or data in the system.
- The failures that are more difficult or critical to deal with are those, that result in loss of information or data (content).

1. **Transaction Failure:**

There are two types of errors that may lead to transaction failure. They are:

(i) System Error: The system has entered an undesirable state as a result of which a transaction cannot continue with its normal execution. The transaction, however, can be re-executed at a later time.

(ii) Logical Error: The transaction can no longer continue with its normal execution, owing the some internal condition, such as data not found, bad input or resource limit overflow exceeded.

System Crash:

There is a hardware bug or error in the database software or the operating system that causes the loss of the content of volatile storage and leads transaction processing to a halt.

The content of non-volatile storage remains undamaged and is not corrupted.

Disk Failure:

In disk failures disk block losses its content as a result of either a head failure during a data transfer operation. Copies of the data on other disks on tertiary media, such as tapes, disks etc., are used to recover from the failure.

- To find out how the system should recover from failure or crash, we need to recognize the failure modes of those devices used for storing data.
- Next, we must consider how these failure modes or crash modes affect the contents or information of the database. We can then use algorithms to ensure database consistency and transaction atomicity despite failures known as recovery algorithms.
- Although they have following two parts:
  1. actions taken following a failure to recover the database contents or data to a state that ensures database transaction atomicity, durability and consistency
  2. actions taken during normal transaction processing to ensure that enough information exists to allow recovery from failures.

4.2 RECOVERY CONCEPTS

- The process of restoring the database to a correct state in the event of a failure is known as database recovery. It is a service to be provided by DBMS to ensure the database reliability.
- Reliability here means both, the resilience of DBMS to various types of failure and its capability to recover from them.
- When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items.
- Transactions are made of various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained i.e., either all the operations are executed or none.
- When a DBMS recovers from a crash, it should maintain the following –
  1. It should check the states of all the transactions, which were being executed.
  2. A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
  3. It should check whether the transaction can be completed now or it needs to be rolled back.
  4. No transactions would be allowed to leave the DBMS in an inconsistent stat
- Consider banking system and transaction T_i that transfers ₹ 1000 from account X to account Y, with initial values of X and Y being ₹ 5000 and ₹ 10000 respectively.
- Suppose that a system crash has occurred during the execution of transaction T_i, after output (Y_A) has taken place, but before output (Y_B) was executed, where Y_A and Y_B denote the buffer blocks on which X and Y are stored. Since, the memory contents were lost, user do not know the fate of the transaction; thus, we could use one of two possible recovery procedures.
  1. **Re-execute T_i:** This procedure will result in the value of X becoming ₹ 3000, rather than ₹ 4000. Thus, the system enters an inconsistent state.
  2. **Do Not Re-execute T_i:** The current system state is value of ₹ 4000 and ₹ 10000 for X and Y respectively. Thus, the system enters an inconsistent state.
- To achieve our goal of atomicity, we must first output information describing the updations of stable storage, without modifying the database itself. As we shall see, this procedure will allow us to output all the updations made by a committed transaction, despite failures.
- There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction:
  1. Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
  2. Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

4.3 LOG-BASED RECOVERY (DEFERRED AND IMMEDIATE UPDATE) (April 13, 16)

- The log-based recovery techniques maintain transaction logs to keep track of all update operations of the transaction. A log consists of log records. A log record maintains a record of all operations of the database.
- It assumes that transactions are executed serially, i.e. only one transaction is active at a time. It uses a structure called log to store the database modifications.
- There are two techniques for using log to achieve the recovery and ensure atomicity in case of failures:
  1. The **deferred database modification technique** ensures transaction atomicity by recording all database modifications in the log, but deferring (delaying) the execution of all write operations of transaction until the transaction partially commits.
  2. The **immediate database modification technique** allows database modifications to be output to the database while the transaction is still in the active state. Data modifications written by active transactions are called uncommitted modifications. If a failure occurs during execution, the system must use the old value field of log records.

Concept of Log:

- Log is a structure used to store the database modifications. It is a sequence of log records and maintains a record of all the update activities in the database.
- There are several types of log records to record significant events during transaction processing.
  1. **Start of Transaction (<start> Log Record):** It contains information about the start of each transaction. It has a transaction identifier, which is always unique. It is denoted as: $<T_i$ start$>$.
  2. **Update Log (<update> Log Record):** It describes a single database write and it is denoted as follows:

     $<T_i, X_j, V_1, V_2>$

     where, $T_i \rightarrow$ Transaction identifier

     $X_j \rightarrow$ Data item identifier

     $V_1 \rightarrow$ Old value of X_j

     $V_2 \rightarrow$ New value of X_j after the write operation.
  3. **Transaction commit (<commit> Log Record):** It records to log when T_i successfully commits. It is denoted as: $<T_i$ commit$>$.
  4. **Transaction abort (<abort> Log Record):** It records to log if T_i aborts. It is denoted as: $<T_i$ abort$>$.
- To recover from a failure in database, basically two operations namely, undo and redo are applied/used with the help of the log on the last consistent state of the database.
- The **undo operation** reverses (rolls back) the changes made to the database by an uncommitted transaction and restores the database to the consistent state that existed before the start of transaction.
- The **redo operation** reapplies the changes of a committed transaction and restores the database to the consistent state it would be at the end of the transaction. The redo operation is required when the changes of a committed transaction are not or partially reflected to the database on disk. The redo modifies the database on disk to the new values for the committed transaction.

4.3.1 Deferred Database Modification (April 17)

- Deferred database modification technique stores the database modifications in the log. Execution of write operation is done when transaction is in partially committed state.

- Deferred database modification ensures transaction atomicity by recording all database modification in the log, by deferring the execution of all write operations of a transaction until the transaction partially commits.
- In deferred database modification, deferred (stops) all the write operations of any Transaction T_i until it partially commits, means modify real database after T_i partially commits. All the activities are recorded in log and the log records are used to modify actual database.
- If a transaction fails in a database before reaching its commit point, it will not have changed the database in any way, so UNDO is not needed. It may be necessary to REDO the effect of the operations of a committed transaction from the log, because their effect may not yet have been recorded in the database. Hence, deferred update is also known as the NO UNDO/REDO algorithm.
- According to transaction state diagram, a transaction is in partially committed state when transaction completes executing the last instruction. Execution of transaction T_i proceed as follows:
  - Before T_i starts its execution record, a record $<T_i$ start$>$ is written to log. Write (X) operation of T_i results in writing a new record $<T_i, X, V_2>$ to the log.
  - When T_i partially commits a record $<T_i$ commit$>$ is written to log. When T_i partially commits, the records associated with it in the log are used in executing the deferred writes. If system crashes before the transaction completes its execution or if the transaction aborts, then the information on the log is ignored.
  - If some failure occurs while updating the database using log records, the log record is written in some stable storage, hence, the transaction can resume its database modification.
  - Using log the system can handle any failure that results in loss of information on volatile storage. The recovery scheme uses the following recovery procedure.
    - **redo (T_i)** sets the value of all data items updated by transaction T_i to the new values. The set of data items and their respective new values can be found in the log. The redo operation must be idempotent i.e. executing it several times must be equivalent to executing it once.
    - A transaction can execute redo (T_i) if the log contains both the record $<T_i$ start$>$ and the record $<T_i$ commit$>$.
- **Example:** Consider the transaction T_0, it transfers ₹ 50 from account A to account B. Original values of account A and B are ₹ 1000 and ₹ 2000 respectively. This transaction is defined as:

  T_0 : read (A);
  A := A – 50;
  write (A);
  read (B);
  B := B + 50;
  write (B).
- Consider the second transaction T_1 that withdraws ₹ 100 from account C. This transaction is defined as: (Assume that original values of account C is 700).

  T_1 : read (C);
  C := C – 100;
  write (C).
- These two transactions are executed serially $<T_0, T_1>$.
- The log containing relevant information on these two transactions is given below:

  $<T_0$ start$>$
  $<T_0, A, 950>$

<T_0, B, 2050>
<T_0 commit>
<T_1 start>
<T_1, C, 600>
<T_1 commit>

- It shows the log that result from the complete execution of T_0 and T_1.
- The actual output can take place to database system in various orders. One such order is given below:

Log	Database
<T_0 start>	
<T_0, A, 950>	
<T_0, B, 2050>	
<T_0 commit>	
	A = 900
	B = 2050
<T_1 start>	
<T_1, C, 600>	
<T_1 commit>	C = 600

- If system crashes before the completion of transactions:

Case 1: Crash occurs just after the log record for write (B) operation.

Log contents after the crash are:

<T_0 start>
<T_0, A, 950>
<T_0, B, 2050>

<T_0 commit> is not written; hence no redo operation is possible.

The values of account remain unchanged i.e. account balance of A is ₹ 1000 and B is ₹ 2000.

Case 2: Crash occurs just after log record for write (C) operation.

The log contents after the crash are:

<T_0 start>
<T_0, A, 950>
<T_0, B, 2050>
<T_0 commit>
<T_1, start>
<T_1, C, 600>

When the system comes back it finds <T_0 start> and <T_0 commit>. But there is no <T_1 commit> for <T_0 start>. Hence, system can execute redo (T_0) but not redo <T_1>. Hence, the value of account C remains unchanged.

Case 3: Crash occurs just after the log record <T_1 commit>.

The log contents after the crash are:

<T_0 start>
<T_0, A, 950>
<T_0, B, 2050>
<T_0, commit>
<T_1 start>
<T, C, 600>
<T_1 commit>

- When system comes back it can execute both redo (T_0) and redo (T_1) operations. For each commit record, the redo (T_i) operation is performed. redo (T_i) writes the values to the database independent of the values currently in the database. Hence, the redo (T_i) is idempotent.

4.3.2 Immediate Database Modification

(April 15, 16, 18)

- It allows the database modifications to be output to the database while transaction is still in active state. Database modifications written by active transactions are called uncommitted modifications.
- In immediate database modification, database is modified by any transaction T_i during its active state means; real database is modified just after the write operation but after log record is written to stable storage. This is because log records are used during recovery.
- Use both Undo and Redo operations in immediate database modification method.
- If a transaction fails after recording some changes in the database but before reaching its commit point, the effect of its operations on the database must be undone i.e., the transaction must be rolled back. Here both undo and redo operations may be required during recovery. This technique is known as UNDO/REDO algorithm.
- Execution of transaction proceeds as follows:
  1. Before T_i starts its execution, the record <T_i start> is written to the log.
  2. Before executing any write (X) i.e. before modifying the database for write operation, it writes an update record <T_i, X, V_1, V_2> to the log.
  3. When T_i partially commits, the record <T_i commit> is written to log.
- As an illustration consider the same example of bank accounts and transactions T_0 and T_1. Transactions T_0 and T_1 are executed serially.
- The log corresponding to this execution is given below:

<T_0 start>
<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>
<T_0 commit>
<T_1 start>
<T_1, C, 700, 600>
<T_1 commit>

- The order in which output took place to both database system and log as a result of execution of T_0 and T_1 is:

Data	Database
<T_0 start>	
<T_0, A, 1000, 950>	
<T_0, B, 2000, 2050>	
<T_0 commit>	A = 950, B = 2050
<T_1 start>	
<T_1, C, 700, 600>	C = 600
<T_1 commit >	

- Using the log, the system can handle any failure. Two recovery procedures are there to recover:
  1. **Undo (T_i):** It restores the value of all data items updated by transaction T_i to the old values.
  2. **Redo (T_i):** It sets the value of all data items updated by transaction T_i to the new values.
- The undo (T_i) and redo (T_i) operations must be idempotent.
- Depending on the log, the recovery scheme determine which transaction need to be redone and which need to be undone.
- If the log record contains the record <T_i start> but does not contain the record <T_i commit> then T_i needs to be undone.
- If the log record contains both the records <T_i start> and <T_i commit> then T_i needs to be redone i.e. If a failure occurs before <T_i commit> the database modifications are rolled back by undo (T_i) operation. If a failure occurs after <T_i commit> then the transaction is reexecuted by redo (T_i) operation.
- Consider the following conditions of failure for transactions T_0 and T_1.

Case 1: Failure occurs just after the log record for write (B) operation.

Log contents and database are:

Log		**Database**
<T_0 start>		
<T_0 ,A, 1000, 950>		
<T_0, B, 2000, 2050>	⟶	A = 950
		B = 2050

Log contains <T_0 start> but does not contain <T_0, commit>. Hence T_0 must be undone hence undo (T_0) is executed, the values of account A and B are restored to 1000 and 2000 respectively.

Case 2: If some failure occurs just after log record for write (C) T_i has written to log.

The log and database contents are:

Log		**Database**
<T_0 start>		
<T_0 A, 1000, 950>		
<T_0 B, 2000, 2050>	⟶	A = 950
		B = 2050
<T_0 commit>		
<T_1 start>		
<T_1 C, 700, 600>	⟶	C = 600
		A = 950

Log contains <T_0 start> and <T_0 commit>. Hence, the redo (T_0) is executed and values of accounts A and B are restored to the same (or new values) 950 and 2050 respectively. But the log doesn't contain <T_1 commit> for <T_1 start>. Hence, the value of account C to restored to old value by undo (T_1) operation. Hence value of C is ₹ 700.

Case 3: If system crashes just after the log record <T_1 commit> has been written to log.

The log and database contents are:

Log		**Database**
<T_0 start>		
<T_0 A, 1000, 950>		
<T_0 B, 2000, 2050>	⟶	A = 950
		B = 2050

<T_0 commit >		
<T_1 start >		
<T_1 C, 700, 600 >		
<T_1 commit>	⟶	C = 600

- Log contains <T_0 start>, <T_0 commit> and <T_1 start> <T_1 commit> operations. Hence recovery system executes redo (T_0) and redo (T_1) operations. The values of accounts A, B and C are ₹ 950, ₹ 2050 and ₹ 600 respectively.

4.4 CHECKPOINTS, RELATIONSHIP BETWEEN DATABASE MANAGER AND BUFFER CACHE, ARIES RECOVERY ALGORITHM

- Checkpoints, relationship between database manager and buffer cache, ARIES recovery algorithm are explained in this section.

4.4.1 Checkpoints

- When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.
- The checkpoint (or syncpoint) is defined as, the point of synchronization between the database and the transaction log file.
- As we know that the general method of database recovery is using information in the transaction log. So, we need to search the entire log. This is time consuming. So, we use checkpoints.
- A checkpoint is like a snapshot of the DBMS state. By taking checkpoints periodically, DBMS can reduce the amount of work to be done during restart in the event of a system crash.
- The approach is to find a point that is sufficiently far back to ensure that any item written before that point has been done correctly and stored safely. This method is called as checkpointing.

4.4.2 Relationship between Database Manager and Buffer Cache

- A Database Manager (DB Manager) is a computer program or a set of computer programs, that provide basic database management functionalities including creation and maintenance of databases.
- Database managers have several capabilities including the ability to back up and restore, attach and detach, create, clone, delete and rename the databases.
- They provide a handful of administrative functionalities such as managing tables, views and stored procedures, as well as run ad hoc queries.
- Each time a database is started ON the server (instance startup), a portion of the computer's main memory (RAM) is allocated, the so called as SGA which contains Buffer cache and Shared pool. Database Buffer cache is one of the most important components of System Global Area (SGA).
- Database buffer cache is the place where data blocks are copied from data files to perform SQL operations. Buffer cache is shared memory structure and it is concurrently accessed by all server processes.
- Buffer cache is a shared and reserved memory area that stores the most recently accessed data blocks from hard disk in RAM. It is also called as data cache.
- It may contain "dirty" (modified data blocks) and "clean" (unused) blocks. Buffer cache is mainly used to take advantage of a computer's fast primary memory compared to the slower secondary memory, thereby minimizing the number of Input/Output (I/O) operations between the primary and secondary memories.

Relationship between Database Manager and Buffer Cache:

- The buffer cache has a significant impact on the performance of OnLine Transaction Processing (OLTP) applications. Hence, it is crucial to monitor the effectiveness of the buffer cache which is an important priority of database manager.
- The buffer cache hit rate is one of the main database metrics that needs to be monitored by database manager.
- Monitoring the buffer cache hit rate measures how many database blocks were found in the buffer cache rather than read from disk. Blocks read from disk are referred to as physical reads.
- Blocks retrieved either from disk or the buffer cache are referred to as logical reads (so physical disk reads are included in both metrics).
- Database manager plays vital role in managing the buffer cache so as to use it effectively in terms of memory as well as response time.
- Database manager has to make appropriate choice among data that should be retained in the buffer cache and the one that should be swapped out of the memory. This makes the query processing faster as the required data set is ready in the buffer cache.
- To conclude, database manager has a very close relationship with buffer cache that leads to well managed memory and effective execution of queries.

4.4.3 ARIES Recovery Algorithm

- ARIES algorithm is used by the recovery manager which is invoked after a crash. Recovery manager will perform restart operations. ARIES algorithm is more simple and flexible algorithms.
- ARIES (Algorithms for Recovery and Isolation Exploiting Semantics) uses a steal/no-force approach for writing and it is based on following three concepts:
  1. **Write-Ahead Logging:** Any change to an object is first recorded in the log, and the log must be written to stable storage before changes to the objects are written to disk.
  2. **Repeating History During Redo:** ARIES will retrace all actions of the database system prior to the crash to reconstruct the database state when the crash occurred. Transactions that were uncommitted at the time of the crash (active transactions) are undone.
  3. **Logging Changes During Undo:** Will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.
- The ARIES recovery procedure consists of following three main steps (phases):
  1. **Analysis:** The analysis phase identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of the crash. The appropriate point in the log where the REDO operation should start is also determined.
  2. **REDO:** The REDO phase actually reapplies updates from the log to the database. Generally, the REDO operation is applied to only committed transactions. However, in ARIES, this is not the case. Certain information in the ARIES log will provide the start point for REDO, from which REDO operations are applied until the end of the log is reached. In addition, information stored by ARIES and in the data pages will allow ARIES to determine whether the operation to be redone has actually been applied to the database and hence need not be reapplied. Thus, only the necessary REDO operations are applied during recovery.
  3. **UNDO:** During the UNDO phase, the log is scanned backwards and the operations of transactions that were active at the time of the crash are undone in reverse order. The information needed for ARIES to accomplish its recovery procedure includes the log, the Transaction Table, and the Dirty Page Table. In addition, check pointing is used. These two tables are maintained by the transaction manager and written to the log during check pointing.

- **Example:** Consider the recovery example shown in Fig. 4.1. There are three transactions namely T_1, T_2 and T_3. T_1 updates page C, T_2 updates pages B and C, and T_3 updates page A.
- Fig. 4.1 (a) shows the partial contents of the log and Fig. 4.1 (b) shows the contents of the Transaction Table and Dirty Page Table.
- Now, suppose that a crash occurs at this point. Since a checkpoint has occurred, the address of the associated begin_checkpoint record is retrieved, which is location 4. The analysis phase starts from location 4 until it reaches the end.
- The end_checkpoint record would contain the Transaction Table and Dirty Page Table in Fig. 4.1 (b), and the analysis phase will further reconstruct these tables.
- When the analysis phase encounters log record 6, a new entry for transaction T_3 is made in the Transaction Table and a new entry for page A is made in the Dirty Page Table.
- After log record 8 is analyzed, the status of transaction T_2 is changed to committed in the Transaction Table. Fig. 4.1 (c) shows the two tables after the analysis phase.
- For the REDO phase, the smallest LSN in the Dirty Page Table is 1. Hence the REDO will start at log record 1 and proceed with the REDO of updates.
- The LSNs {1, 2, 6, 7} corresponding to the updates for pages C, B, A and C respectively, are not less than the LSNs (Log Sequence Records) of those pages (as shown in the Dirty Page Table).
- So those data pages will be read again and the updates reapplied from the log (assuming the actual LSNs stored on those data pages are less than the corresponding log entry).
- At this point, the REDO phase is finished and the UNDO phase starts. From the Transaction Table (Fig. 4.1 (c)), UNDO is applied only to the active transaction T_3.
- The UNDO phase starts at log entry 6 (the last update for T_3) and proceeds backward in the log. The backward chain of updates for transaction T_3 (only log record 6 in this example) is followed and undone.

LSN	Last_LSN	Transaction_id	Type	Page_ID	Other_Information
1	0	T_1	update	C	...
2	0	T_2	update	B	...
3	1	T_1	commit		...
4	begin checkpoint				
5	end checkpoint				
6	0	T_3	update	A	...
7	2	T_2	update	C	...
8	7	T_2	commit		...

(a)

Transaction Table

Transaction_ID	Last_LSN	Status
T_1	3	commit
T_2	2	in progress

(b)

Dirty Page Table

Page_ID	LSN
C	1
B	2

Transaction Table

Transaction_ID	Last_LSN	Status
T_1	3	commit
T_2	8	commit
T_3	6	in progress

Dirty Page Table

Page_ID	LSN
C	1
B	2
A	6

(c)

Fig. 4.1: An example of recovery in ARIES (a) The Log at Point of Crash (b) The Transaction and Dirty Page Tables at Time of Checkpoint (c) The Transaction and Dirty Page Tables after the Analysis Phase

4.5 RECOVERY WITH CONCURRENT TRANSACTIONS (ROLLBACK, COMMIT, CHECKPOINT)

- Now we will discuss how user can modify and extend the log-based recovery scheme to deal with multiple concurrent transactions.
- The number of concurrent transactions, computer system has a single disk buffer and a single log. The buffer blocks are shared by all transactions.
- User allows immediate updates and the permit a buffer block to have data items updated by one or more transactions.
- There are four ways/schemes of recovery from concurrent transactions as explained below:
  1. In **Interaction with Concurrency Control** scheme, recovery depends upon the concurrency control scheme being used. So, to rollback a failed transact-ion, we must undo the updates performed by the transaction.
  2. In **Transaction Rollback** scheme, we rollback a failed transaction by using the log. The system scans the log backward, for every log record found in the log the system restores the data item.
  3. We use **Checkpoints** scheme to reduce the number of log records that the system must scan when it recovers from a crash. In a concurrent transaction processing system, we require that the checkpoint log record be of the form <checkpoint Q> where, 'Q' is a list of transactions active at the time of the checkpoint. A fuzzy checkpoint is a checkpoint where transactions are allowed to perform updates even while buffer blocks are being written out.
  4. In **Restart Recovery** scheme, when the system recovers from a crash, it constructs two lists namely, Undo-list and Redo-list. The undo-list consists of transactions to be undone. The redo-list consists of transaction to be redone.

4.5.1 Rollback

- User rollback a failed transaction, T_i, using the log.
- The log is scanned backward; for every log record of the form $<T_i, X_j, V_1, V_2>$ found in the log, the data item X_j is restored to its old value V_1.

The scanning of the log terminates when the log record $<T_i$, start> is found.

Scanning the log backward is more important, since a transaction may have updated a data item more than once. Consider the pair of log records as given below:

<T, A, 10, 20>

<T, A, 20, 30>

They represent a modification of data item A by T_i, followed by another modification of A by T_i. Scanning the log backward set A correctly to 10.

- If the log were scanned in the forward direction, A would be set to 20, which value is incorrect. If strict two-phase locking is used for concurrency control, locks held by a transaction T may be released only after the transaction has been rolled back as described.
- Once, transaction T has updated a data item, no other transaction could have updated the same data item, due to the concurrency-control requirements.

Interaction with Concurrency Control :

- The recovery scheme depends on the concurrency-control technique that is used. To roll back a failed transaction, user must undo the updates or modification performed by the transaction.
- Suppose that a transaction T_0 has to be rolled back and a data item Q that was updated by T_0 has to be restored to its old value. Using the log-based schemes for recovery, we restore the value using the undo information in a log record.
- Suppose a second transaction T_1 has performed yet another update on Q before T_0 is rolled back. Then, the updation performed by T_1 will be lost if T_0 is rolled back.
- If a transaction T has updated a data item Q, no other transaction may update the same data item until T has committed. We can ensure this requirement easily by using strict two-phase locking.

4.5.2 Commit

- If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.
- Commit command is used to permanently save any transaction into database.
- Following is commit command's **syntax**:

```
commit;
```

For example:

```
INSERT into class values(5,'Rahul');
commit;
```

- After commit point the transaction then writes an entry [COMMIT, T_i] into the log.
- If a system failure occurs, the log is searched back for all transactions T that have written a [START_TRANSACTION, T_i] entry in the log but do not make [COMMIT, T_i]. Then these transactions may have to be rolled back to undo their effect on the database.
- Transactions that are committed must also have recorded all their write entries in the log. So, their effect on the database can be redone from the log entries.

4.5.3 Recovery Using Checkpoint

- Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.
- Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.
- When a system with concurrent transactions crashes and recovers, it behaves in the manner as shown in Fig. 4.2.

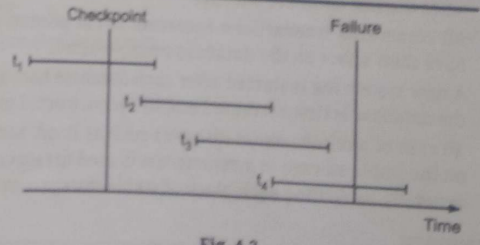

Fig. 4.2

- The recovery process using checkpoints contains following:
  1. The recovery system reads the logs backwards from the end to the last checkpoint.
  2. It maintains two lists, an undo-list and a redo-list.

3. If the recovery system sees a log with <T_n, Start> and <T_n, Commit> or just <T_n, Commit>, it puts the transaction in the redo-list.
4. If the recovery system sees a log with <T_n, Start> but no commit or abort log found, it puts the transaction in undo-list.

- All the transactions in the undo-list are then undone and their logs are removed. All the transactions in the redo-list and their previous logs are removed and then redone before saving their logs.

Advantages of Checkpoints:

1. No need to redo successfully completed transactions before most recent checkpoints.
2. In checkpointing less searching required.
3. In checkpointing old records can be deleted,
4. Checkpointing is done with log-based recovery schemes to reduce the time required for recovery after a crash.

4.6 DATABASE BACKUP AND RECOVERY FROM CATASTROPHIC FAILURE (April 17; Oct. 17)

- The recovery techniques in previous sections applied in case of non-catastrophic failure. The recovery manager of a database system must also be equipped to handle more catastrophic failures such as disk crashes. The main technique used to handle such crashes is a database backup.
- In database backup the whole database and the log are periodically copied onto a storage medium such as magnetic tapes or disks. In case of a catastrophic system failure, the latest backup copy can be reloaded from the tape to the disk, and the system can be restarted.
- Catastrophic recovery is required if the simple database recovery fails, which happens when the data on the disk gets corrupted. In case of a catastrophic system failure, the backup copy is restored and the system can be restarted.
- The main technique used to handle catastrophic failures including disk crash is database backup (also known as dump).
- The whole database and the log are periodically copied onto a storage medium such as magnetic tapes. In case of a catastrophic system failure, the latest backup copy can be reloaded from the tape to the disk, and the system can be restarted.
- To avoid losing all the effects of transactions that have been executed since the last backup, it is customary to back up the system log by periodically copying it to magnetic tape.
- The system log is usually smaller than the database itself and hence can be backed up more frequently. When the system log is backed up, users do not lose all transactions they have performed since the last database backup.
- All committed transactions recorded in the portion of the system log that has been backed up can have their effect on the database reconstructed.
- A new system log is started after each database backup operation. Hence, to recover from disk failure, the database is first recreated and/or reconstructed on disk from its latest backup copy on tape.

In case of environmental disasters such as flood, earthquakes, etc., the backup of the database taken on the stable storage at a remote site is used to recover the lost data. The copy of the database is made generally by writing each block of stable storage over a computer network called remote backup.

Examples:

Example 1: Following are the log entries at the time of system crash:

[start - transaction, T_1]
[write - item, T_1, A, 10]

[commit T_1]
[start - transaction, T_3]
[write - item T_3, B, 15]
[checkpoint]
[commit T_3]
[start - transaction, T_2]
[write - item T_2, B, 20]
[start - transaction, T_4]
[write - item, T_4, D, 25]
[write – item, T_2, C, 30] ⟵ system crash

If deferred update technique with checkpoint is used, what will be the recovery procedure?

Sol.:

Transactions T_2 and T_4 are ignored, because they did not reach their commit points. i.e. T_2 and T_4 do not have commit instruction before the system crash.

T_1 is committed before the last system checkpoint, hence it need not be redone. T_3 is redone because its commit point is after the last system checkpoint.

Example 2: Following are the log entries at the time of system crash:

[start - transaction, T_1)
[write - item, T_1, B, 100]
[commit, T_1]
[checkpoint]
[start - transaction, T_2]
[write - item, T_2, D, 100]
[commit, T_2]
[start - transaction, T_3]
[write - item, T_3, D, 200]
[write - item, T_3, B, 200] ⟵ system crash.

If deferred update technique with checkpoint is used, What will be the recovery procedure?

Sol.:

Transaction T_3 is ignored, because it did not reach its commit point. i.e. T_3 does not have commit instruction before the system crash.

T_1 is committed before the last system checkpoint, hence it need not be redone. T_2 is redone because its commit point is after the last system checkpoint.

Example 3: Following are the log entries at the time of system crash:

[start - transaction, T_1]
[write - item, T_1, A, 10, 100]
[commit, T_1]
[checkpoint]
[start - transaction, T_2]
[write - item, T_2, B, 20, 200]
[commit, T_2]
[start - transaction, T_3]
[write - item, T_3, C, 30, 300] ⟵ system crash

If immediate update technique with checkpoint is used, what will be the recovery procedure?

Sol.:

Log contains start instruction for T_3, but does not contain commit instructions for T_3 before the system crash. Hence, T_3 need to be undone.

A log contains start and commits instruction for T_2 before the system crash. Hence, T_2 need to be redone.

T_1 is committed before the last system checkpoint. Hence, it need not be redone.

Example 4: Consider the log given below corresponding to a schedule S. Specify which transactions are rolled back, which operations in the log are redone and which (if any) are undone.

[start - transaction, T_1]
[read - item, T_1, A]
[read - item, T_1, D]
[write - item, T_1, D, 20]
[commit, T_1]
[check point]
[start - transaction, T_2]
[read - item, T_2, B]
[write - item T_2, B, 12]
[start - transaction, T_4]
[read - item,T_4, B]
[write - item, T_4, B, 15]
[start - transaction, T_3]
[write - item T_3, A, 30]
[read - item, T_4, A]
[write - item, T_4, A, 20]
[commit, T_4]
[read - item, T_2, D]
[write - item, T_2, D, 25] ⟵ System Crash

Sol.:

We now describe the immediate update protocol:

Since, there is a checkpoint after {commit, T_1}, all redo and undo operations are done for the entry after this checkpoint.

In case of immediate method, the following transactions are redone → T_4, and follow transactions are undone → T_2, T_2.

In case of deferred update modification method, the following transactions are redone → T_4.

No transactions are undone, since in deferred update method, the actual updation to the database takes place only when transaction issues commit.

Example 5: Consider the following log image, that is obtained during recovery after a crash:

<T_1, start>
<T_1, X, 10, 10>
<T_1, Y, 20, 5>
<T_2, start>
<T_2, X, 20, 200>
<T_1, commit>
<T_3, start>

<T_3, Z, 10, 20>
<Checkpoint>
<T_3, K, 20, 200>
<T_2, commit>
<T_4, start>
<T_4, X, 200, 100> ⟵ System Crash

(i) List the contents in the list L.
(ii) List the contents in:
  (a) Undo-list, and
  (b) Redo-list.

Sol.:

(i) The contents in the list L are: {T_2, T_3, T_4}
(ii) (a) Undo List = {T_4}
  (b) Redo List = {T_2}.

Example 6: Consider the log in Example 1. State how the recovery procedure takes place using the immediate update algorithm and using the deferred update algorithm.

Sol.: The list L, the undo list and the redo list are given in the previous example. L = {T_2, T_3, T_4}, Undo list = {T_4}, Redo List = {T_2}

Next the recovery manager updates undo and redo lists as per the contents in the list L.

List contains T_2, T_3 out of which T_2 is in redo list, but T_3 is not. Hence, it adds T_3 to undo-list, so now the undo list is = {T_3, T_4}.

Immediate Update Method: As per this method, actions of active transactions have to be undone. Hence, the recovery procedure is as follows:

(i) The recovery manager scans the log backwards and for each record of each transaction in the undo list, it undoes the action.

  Here, Undoing <T_4, X, 200, 100>, leads

  X = 200.

  T_4 undo completes, since we reached <T_4, start>

  Undoing <T_3, K, 20, 200> leads

  K = 20.

  Undoing <T_3, Z, 10, 20> leads

  Z = 10.

  At this point T_3's undos are completed, since we have reached <T_3, start> records.

(ii) Next recovery manager scans log forward from most recent checkpoint record and redoes the actions of T_2, since only T_2 is in the redo list.

Deferred Update Method: In this method only redo of committed transactions takes place. There is no undo of active transactions, since their actions don't affect the database at all. Hence here, the log is scanned forward from the most recent checkpoint, and actions of the transaction T_2 are redone.

PRACTICE QUESTIONS

Q.I Multiple Choice Questions:

1. The process of restoring the database to a consistent state when a failure occurred called as database?
  (a) recovery (b) integrity
  (c) Security (d) All of these

2. Which must be written on the non-volatile (stable) storage?
   (a) roles (b) Logs
   (c) Both (a) and (b) (d) All of these
3. Types of failures include?
   (a) System crash (b) Disk failure
   (c) Transaction failure (d) All of these
4. Which operations for log-based recovery are required?
   (a) Redo (b) Undo
   (c) Both (a) and (b) (d) None of these
5. Which manager component of database is responsible for performing the recovery operations?
   (a) Recovery (b) Transaction
   (c) Both (a) and (b) (d) None of these
6. Which techniques maintain transaction logs to keep track of all update operations of the transactions?
   (a) lock-based recovery (b) shadow paging
   (c) log-based recovery (d) All of these
7. In which technique, the transaction is not allowed to update the database on disk until the transaction enters into the partially committed state?
   (a) Immediate update (b) deferred update
   (c) Both (a) and (b) (d) None of these
8. A collection of in-memory buffers, called the database?
   (a) buffer (b) cache
   (c) log (d) All of these
9. Deferred update is also known as the _____ algorithm.
   (a) UNDO/REDO (b) NO UNDO/REDO
   (c) Both (a) and (b) (d) None of these
10. The method/techniques for recovery from concurrent transactions include?
   (a) Transaction rollback (b) Checkpoints
   (c) Restart recovery (d) All of these
11. Which management provides a temporary copy of a database page?
   (a) Buffer (b) Log
   (c) Lock (d) All of these
12. When a system restarts after a crash, the ARIES recovers from the crash in three phases namely _____.
   (a) analysis (b) redo
   (c) undo (d) All of these

Answers

1. (a)	2. (b)	3. (d)	4. (c)	5. (a)	6. (c)	7. (b)	8. (b)	9. (b)	10. (d)
11. (a)	12. (d)								

Q.II Fill in the Blanks:

1. _____ recovery is the process by which the database is moved back to a consistent state.
2. In _____ update technique, as soon as a data item is modified in cache, the disk copy is immediately updated. That is, the transaction is allowed to update the database in its active state.
3. To recover from catastrophic failures which result in loss of data in non-volatile storage, the _____ of the database is used.
4. The _____ technique does not require the use of a log as both the after image and the before image of the data item to be modified are maintained on the disk.

5. Database _____ is the process of restoring the database to a correct (consistent) state in the event of a failure.
6. The _____ operation is required when the changes of a committed transaction are not or partially reflected to the database on disk.
7. A _____ is a sequence of log records that contains essential data for each transaction which has updated the database.
8. The _____ is the point of synchronisation between the database and the transaction log file.
9. A log of log _____.
10. A _____ is responsible for the efficient management of the database buffers that are used to flush pages between buffer and secondary storage.

Answers

1. Crash	2. immediate	3. backup	4. shadow paging	5. recovery
6. redo	7. log	8. checkpoint	9. records	10. buffer manager

Q.III State True or False:

1. To recover from a failure, basically undo and redo operations are applied with the help of the log on the last consistent state of the database.
2. The process of restoring the database to a correct state in the event of a failure is known as database recovery.
3. Commit command is used to permanently save any transaction into database.
4. In immediate database modification technique, deferred (stops) all the write operations of any Transaction T_i until it partially commits.
5. In transaction rollback, we rollback a failed transaction by using the log.
6. To find a point that is sufficiently far back to ensure that any item written before that point has been done correctly and stored safely and this method is called as rollbacking.
7. The assignment and management of memory blocks is called buffer management.
8. ARIES (Algorithm for Recovery and Isolation Exploiting Semantics) is a recovery algorithm designed to work with a steal, no-force approach.
9. If the transaction does not modify the database until it has partially committed, it is said to use deferred modification technique.
10. The main technique used to handle catastrophic failures including disk crash is database backup.
11. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.

Answers

1. (T)	2. (T)	3. (T)	4. (F)	5. (T)	6. (F)	7. (T)	8. (T)	9. (T)	10. (T)	11. (T)

Q.IV Answer the following Questions:

(A) Short Answer Questions:

1. What is crash recovery? The process by which the database is moved back to a consistent and usable state
2. Define the term failure.
3. What is log?
4. Enlist purpose of checkpoint. saves the current state of the database to disk
5. Define recovery.
6. What is deferred database modification? occurs if the transaction does not modify the database until it has committed
7. What is immediate database modification? allows database modifications to occur while the transaction is still active

8. What is meant by buffer management?
9. What is ARIES?
10. What is the use of commit in concurrent transaction?
11. Enlist recovery techniques for concurrent transaction.
12. Define the term checkpoint.
13. What is the use of rollback in concurrent transaction?
14. What is meant by log-based recovery?
15. List ant two responsibilities of log manager.

(B) Long Answer Questions:

1. Explain failure classification in detail.
2. Enlist various types of transaction failures.
3. Explain the term log-based recovery in detail.
4. With neat diagram explain shadow paging.
5. Explain the following terms:
   (i) Crash recovery
   (ii) Failure.
6. Describe the term Buffer management in detail.
7. Explain ARIES algorithm with example.
8. Write short notes on:
   (i) Checkpoint.
   (ii) Buffer management.
   (iii) Deferred update modification
   (iv) Immediate update modification.
9. Enlist various types of errors.
10. What do you meant by recovery system?
11. What is log and log record? Explain in detail.
12. Explain the following terms in detail:
    (i) Commit
    (ii) Rollback
    (iii) Disk failure
    (iv) Log
    (v) Log-based buffering.
13. Describe immediate database modification with example.
14. What is system crash? Explain in detail.
15. With the help of diagram explain relation between recovery management and buffer management.
16. Explain the term database backup and recovery from catastrophic failure.

UNIVERSITY QUESTIONS AND ANSWERS

April 2015

1. State and explain any two types of log records.
   Ans. Refer to Section 4.3. **(1 M)**
2. The following are log entries at the time of system crash:
   [start- transaction, T1]
   [write-item, T1, B, 10]
   [commit, T1]
   [start- transaction, T2]
   [write-item, T2, D, 20]
   [write-item, T2, A, 30]
   [commit, T2]
   [checkpoint]
   [start-transaction, T3]
   [write-item, T3, B, 20]
   [start-transaction, T4]
   [write-item, T4, C, 10] → System Crash
   If immediate update with checkpoint is used, what will be the recovery procedure?
   Ans. Refer to Examples and Section 4.3.2. **(5 M)**

April 2016

1. Define a system log.
   Ans. Refer to Section 4.3. **(1 M)**
2. Consider the following log at the time of system crash:
   <T1, start>
   <T1, A, 30, 50>
   <T2, start>
   <T2, B, 90, 40>
   <T1, commit>
   <T3, start>
   <T2, commit>
   <checkpoint>
   <T3, C, 100, 150>
   <T4, start>
   <T3, commit>
   <T4, D, 70, 50> ← system crash
   If the immediate update method is used what will be the recovery procedure?
   Ans. Refer to Examples and Section 4.3.2. **(5 M)**

April 2017

1. What is lost update problem?
   Ans. Refer to Section 4.3. **(1 M)**
2. Discuss how the recovery from catastrophic failure is handled.
   Ans. Refer to Section 4.6. **(3 M)**
3. The following are log entries at the time of system crash:
   [start-transaction, T1]
   [read-item, T1, D]
   [write-item, T1, D, B]
   [commit, T1]
   [checkpoint]
   [start-transaction, T2]
   [read-item, T2, B]
   [write-item, T2, B, 12]

[start-transaction, T3]
[write-item, T3, A, 20]
[read-item, T3, D]
[write-item, T3, D, 20] ← system crash
If defferred update with checkpoint is used, what will be the recovery procedure? (5 M)

Ans. Refer to Examples and Section 4.3.1.

October 2017

1. The following are log entries at the time of system crash:
[start-transaction, T1]
[write-item, T1, A, 5]
[commit, T1]
[start-transaction, T2]
[write-item, T2, B, 10]
[write-item, T2, D, 6]
[commit, T2]
[checkpoint]
[start-transaction, T3]
[write-item, T3, B, 20]
[start- transaction, T4]
[write-item, T4, C, 10] ← system crash
If immediate update with checkpoint is used, what will be the recovery procedure? (5 M)

Ans. Refer to Examples and Section 4.3.2.

2. Explain catastropic database security. (5 M)

Ans. Refer to Section 4.6.

April 2018

1. The following is log entries at time of system crash:
[start-transaction, T1]
[write, T1, A, 40]
[start-transaction, T2]
[write, T2, B, 80]
[start-transaction, T3]
[write, T3, C, 100]
[commit, T2]
[commit, T1]
[checkpoint]
[start- transaction, T4]
[write, T4, C, 200]
[write, T3, E, 10] ← system crash
If immediate update with checkpoint is used, what will be the recovery procedure? (5 M)

Ans. Refer to Section 4.3.2.

❖❖❖

Syllabus ...

1. Relational Database Design Using PLSQL **(8 Hrs.)**

1.1 Introduction to PLSQL.
1.2 PL/pgSqL: Datatypes, Language Structure.
1.3 Controlling the Program Flow, Conditional Statements, Loops.
1.4 Stored Procedures.
1.5 Stored Functions.
1.6 Handling Errors and Exceptions.
1.7 Cursors.
1.8 Triggers.

2. Transaction Concepts and Concurrency Control **(10 Hrs.)**

2.1 Describe a Transaction, Properties of Transaction, State of the Transaction.
2.2 Executing Transactions Concurrently Associated Problem in Concurrent Execution.
2.3 Schedules, Types of Schedules, Concept of Serializability, Precedence Graph for Serializability.
2.4 Ensuring Serializability by Locks, Different Lock Modes, 2PL and its Variations.
2.5 Basic Timestamp Method for Concurrency, Thomas Write Rule.
2.6 Locks with Multiple Granularity, Dynamic Database Concurrency (Phantom Problem).
2.7 Timestamps versus Locking.
2.8 Deadlock and Deadlock Handling - Deadlock Avoidance (Wait-Die, Wound-Wait), Deadlock Detection and Recovery (Wait for Graph).

3. Database Integrity and Security Concepts **(6 Hrs.)**

3.1 Domain Constraints.
3.2 Referential Integrity.
3.3 Introduction to Database Security Concepts.
3.4 Methods for Database Security.
  3.4.1 Discretionary Access Control Method.
  3.4.2 Mandatory Access Control.
  3.4.3 Role Base Access Control for Multilevel Security.
3.5 Use of Views in Security Enforcement.
3.6 Overview of Encryption Technique for Security.
3.7 Statistical Database Security.

4. Crash Recovery **(4 Hrs.)**

4.1 Failure Classification.
4.2 Recovery Concepts.
4.3 Log base recovery Techniques (Deferred and Immediate Update).
4.4 Checkpoints, Relationship between Database Manager and Buffer Cache. ARIES Recovery Algorithm.
4.5 Recovery with Concurrent Transactions (Rollback, Checkpoints, Commit).
4.6 Database Backup and Recovery from Catastrophic Failure.

5. Other Databases **(2 Hrs.)**

5.1 Introduction to Parallel and Distributed Databases.
5.2 Introduction to Object Based Databases.
5.3 XML Databases.
5.4 NoSQL Database.
5.5 Multimedia Databases.
5.6 Big Data Databases.