Q1: Explain log-based recovery technique.

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.
* Deferred Update is a technique used for the maintenance of transaction log files in a Database Management System (DBMS).
* It is also referred to as the NO-UNDO/REDO technique.
* This technique is employed for the recovery of transaction failures resulting from power outages, memory issues, or operating system failures.
* When a transaction is executed, the updates are not immediately applied to the database.
* Instead, these updates are first recorded in the log file.
* Subsequently, these changes are applied to the database once the transaction is committed; this process is known as the "Re-doing" process.
* In case of a rollback, none of the changes made during the transaction are applied to the database, and the changes recorded in the log file are discarded as well.
* However, if the system crashes after the commit is performed, upon system restart, the changes recorded in the log file are applied to the database.

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.
* Immediate Update is a technique used for maintaining transaction log files in a Database Management System (DBMS).
* It is alternatively known as the UNDO/REDO technique.
* This method serves for the recovery of transaction failures stemming from power outages, memory issues, or operating system failures.
* Upon execution of any transaction, updates are directly applied to the database, and a log file is maintained containing both old and new values.
* Once the transaction is committed, all changes are permanently stored in the database, and records in the log file are discarded accordingly.
* In the event of a rollback, old values are restored in the database, and all changes made to the database are discarded; this process is termed the "Un-doing" process.
* If the system crashes after the commit is executed, upon system restart, changes are permanently stored in the database.

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<Tn, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

Q2: Compare Deferred Database Modification and Immediate Database Modification.

|  |  |  |
| --- | --- | --- |
| Feature | Deffered | Immediate |
| Update Timing | Updates occur after instruction execution | Updates occur during instruction execution |
| Processor Speed | May be faster: update occurs after instruction execution | May be slower: update occurs during instruction execution |
| Complexity | More complex: requires additional mechanisms for updates | Less complex: updates occur immediately during execution |
| Consistency | May result in temporary inconsistency between data | Data consistency is maintained between registers and memory |
| Flexibility | May be more flexible: allows for complex data manipulations | Less flexible: immediate updates can limit data manipulation |
| Log File Contents | Contains all changes to be applied to the database | Contains both old and new values |
| Rollback Procedure | Discards all records in log file | Restores old values using log file records |
| Recovery Time in System Failure | Requires more time for recovery | Frequent I/O operations during active transactions |
| Method of Recovery | Changes recorded in log file, applied to database on commit | Database updated directly, log file maintains old and new values |
| Use of Buffering and Caching | Utilizes buffering and caching | Uses shadow paging for efficient update operations |
| Handling of Updates | Deferred until commit, allowing for batch processing | Immediate, potentially causing processor to stall |
|  |  |  |

**Q3: Explain the purpose of Checkpoint mechanism. Explain the steps for performing a checkpoint.761**

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 checkpoint is like a bookmark. While the execution of the transaction, such checkpoints are marked, and the transaction is executed then using the steps of the transaction, the log files will be created.
* When it reaches to the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file. Then the log file is updated with the new step of transaction till next checkpoint and so on.
* The checkpoint is used to declare a point before which the DBMS was in the consistent state, and all transactions were committed.

**Purpose of checkpoint mechanism:**

1. Efficient Recovery Preparation:
   * When a system crash occurs, recovery involves redoing or undoing transactions based on the log records.
   * Without checkpoints, the system would need to search the entire log, which is time-consuming.
   * Checkpoints streamline this process by marking a point in the log where certain actions have been completed, reducing the search space during recovery.
2. Reducing Recovery Overhead:

* Transactions that need to be redone may have already written their updates into the database by the time of a crash.
* Checkpoints help reduce recovery overhead by ensuring that updates are written to disk before the checkpoint, minimizing the need for redo operations after a crash.

Steps for Performing a Checkpoint:

* **Output Log Records to Stable Storage:**
  + All log records currently residing in main memory are output to stable storage.
* **Output Modified Buffer Blocks to Disk:**
  + All modified buffer blocks are output to the disk.
* **Record Active Transactions:**
  + A log record of the form <checkpoint L> is output onto stable storage, where L is a list of transactions active at the time of the checkpoint.

During a checkpoint, transactions are not allowed to perform any update actions. This ensures data consistency during the checkpointing process.

After a system crash, the system examines the log to find the last <checkpoint L> record, which marks the transactions active at the time of the crash. Recovery actions are then applied based on this information, including undoing incomplete transactions and redoing committed transactions.

Checkpoints play a crucial role in recovery procedures, facilitating efficient recovery and minimizing overhead by ensuring that database modifications are appropriately logged and synchronized with disk storage.

**Advantages of Checkpoints**

* Checkpoints help us in recovering the transaction of the database in case of a random shutdown of the database.
* It enhancing the consistency of the database in case when multiple transactions are executing in the database simultaneously.
* It increasing the data recovery process.
* Checkpoints work as a synchronization point between the database and the transaction log file in the database.
* Checkpoint records in the log file are used to prevent unnecessary redo operations.
* Since dirty pages are flushed out continuously in the background, it has a very low overhead and can be done frequently.
* Checkpoints provide the baseline information needed for the restoration of the lost state in the event of a system failure.
* A database checkpoint keeps track of change information and enables incremental database backup.
* A database storage checkpoint can be mounted, allowing regular file system operations to be performed.
* Database checkpoints can be used for application solutions which include backup, recovery or database modifications.

**Disadvantages of Checkpoints**

1. Database storage checkpoints can only be used to restore from logical errors (E.g. a human error).

2. Because all the data blocks are on the same physical device, database storage checkpoints cannot be used to restore files due to a media failure.

**Real-Time Applications of Checkpoints**

* Backup and Recovery
* Performance Optimization
* Auditing

**3.State and explain various classes of failure in database system.**

**Classes of Failure in Database Systems:**

1. **Transaction Failure:**
   * Transaction failure occurs when a transaction cannot complete its execution successfully due to errors.
   * Transaction failure can result from logical errors, such as bad input or data not found, which prevent the transaction from proceeding.
   * System errors, such as deadlocks, occur due to undesirable system states, halting the transaction's execution temporarily.
   * Recovery from transaction failure involves identifying the cause of the failure and taking appropriate corrective actions, such as rolling back the transaction or re-executing it later.
   * There are two types of errors that can cause transaction failure:
     + **Logical Error:** The transaction encounters an internal condition (e.g., bad input, data not found, overflow) that prevents it from continuing with normal execution.
     + **System Error:** The system enters an undesirable state (e.g., deadlock) that prevents the transaction from continuing. However, the transaction can be re-executed at a later time.
2. **System Crash:**
   * A system crash occurs due to hardware malfunctions or bugs in the software, causing the loss of volatile storage contents.
   * Despite the fail-stop assumption, where the system halts in case of errors without corrupting nonvolatile storage, recovery mechanisms are required to restore the system to a consistent state.
   * Recovery from a system crash involves restarting the system and performing recovery actions to restore database consistency and transaction atomicity.
   * A system crash occurs due to hardware malfunctions or bugs in the database software or operating system.
   * It leads to the loss of the content of volatile storage (e.g., RAM), bringing transaction processing to a halt.
   * Nonvolatile storage (e.g., disk) contents remain intact and are not corrupted due to the fail-stop assumption.
3. **Disk Failure:**
   * Disk failures can occur due to various reasons, leading to the loss of data stored on disk blocks.
   * Recovery from disk failure involves using redundant copies of data on other disks or archival backups to restore the lost data.
   * Algorithms for recovery from disk failure ensure database consistency and transaction atomicity despite the failure by reconstructing lost data from redundant copies or backups.
   * Disk failure happens when a disk block loses its content due to an head crash or failure during a data-transfer operation.
   * Recovery from disk failure involves using copies of the data on other disks or archival backups on tertiary media (e.g., DVD, tapes).

**Elaborate the Immediate Database Modification with its Recovery mechanism.**

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.
* Immediate Update is a technique used for maintaining transaction log files in a Database Management System (DBMS).
* It is alternatively known as the UNDO/REDO technique.
* This method serves for the recovery of transaction failures stemming from power outages, memory issues, or operating system failures.
* Upon execution of any transaction, updates are directly applied to the database, and a log file is maintained containing both old and new values.
* Once the transaction is committed, all changes are permanently stored in the database, and records in the log file are discarded accordingly.
* In the event of a rollback, old values are restored in the database, and all changes made to the database are discarded; this process is termed the "Un-doing" process.
* If the system crashes after the commit is executed, upon system restart, changes are permanently stored in the database.

4. Explain Shadow paging in detail.

**Shadow Paging:**

1. **Overview:**
   * Shadow paging is a technique used for implementing atomicity and durability in database systems.
   * It ensures that transactions are either fully committed or fully aborted, without any intermediate states.
2. **Concept:**
   * In shadow paging, two identical copies of the database are maintained: the current database and a shadow database.
   * Transactions operate on the shadow database while the current database remains unchanged.
   * Once a transaction is committed, the shadow database becomes the new current database, ensuring atomicity.
3. **Process:**
   * Initially, both the current and shadow databases point to the same physical storage.
   * When a transaction begins, it operates on the shadow database, making all modifications there.
   * If the transaction commits, the shadow database becomes the new current database by updating the pointer.
   * If the transaction aborts, the shadow database is discarded, and no changes are made to the current database.
4. **Atomicity and Durability:**
   * Atomicity is achieved because transactions are applied entirely to the shadow database before committing.
   * Durability is ensured because changes are made to the current database only after a transaction commits.
5. **Advantages:**
   * Simple to implement and understand.
   * Provides strong atomicity and durability guarantees.
   * Well-suited for systems with limited resources or where speed is critical.
6. **Disadvantages:**
   * Requires additional storage space for maintaining the shadow database.
   * Overhead involved in copying the entire database for each transaction.
7. **Example:**
   * Suppose a transaction modifies several records in the database.
   * These modifications are applied to the shadow database, keeping the current database intact.
   * If the transaction commits, the pointer is updated to make the shadow database the new current database.

5. List and elaborate the Drawbacks of Shadow Paging.

**Drawbacks of Shadow Paging:**

1. **Storage Overhead:**
   * Maintaining two identical copies of the database (current and shadow) requires additional storage space.
   * This overhead can be significant, especially for large databases, leading to increased costs.
2. **Performance Impact:**
   * Shadow paging involves copying the entire database for each transaction, which can impact performance.
   * Copying large databases frequently can result in slower transaction processing times.
3. **Disk I/O Overhead:**
   * Writing changes to both the shadow database and the log file increases disk I/O operations.
   * This can lead to increased disk latency and reduced system performance, particularly in high-transaction environments.
4. **Fragmentation Issues:**
   * Shadow paging can cause fragmentation of the database file due to frequent copying.
   * Fragmentation can lead to inefficient disk usage and slower access times, especially during read operations.
5. **Complexity in Recovery:**
   * Recovering from a system crash with shadow paging can be complex.
   * Rollback procedures may require restoring the entire shadow database to its previous state, which can be time-consuming.
6. **Limited Scalability:**
   * Shadow paging may not scale well for very large databases or systems with a high volume of concurrent transactions.
   * The overhead of maintaining multiple copies of the database can become prohibitive as the system grows.
7. **Concurrency Issues:**
   * Shadow paging may introduce concurrency issues, especially in multi-user environments.
   * Concurrent transactions operating on the same data may lead to conflicts or inconsistencies if not properly managed.