Unit-10 Database Recovery Techniques

- Ankit Pargeni

DATE

(Recovery Concepts)

* Recovery outline

** Database recovery is the process of restoring the database to a correct (consistent) state in the event of a failure. A computer can fail due to disk crash, software failure, fire in a machine or any natural disaster. In case of failure, transaction may be lost.

So, every database system must have a recovery scheme that can restore the database to the consistent state that existed before failure. To do this, the system must keep information about the changes that were applied to the data items by various transactions.

A typical strategy for recovery are:

- If there is an extensive damage to the wide portion of database due to catastrophic failure, such as disk crash the recovery method restores a past copy of the database that was backed up to archival storage like tape.

- When the database on disk is not physically damaged a non catastrophic failure has occured, then the recovery strategy is to identify the changes that may cause an inconsistency in the database. We don't need a complete archivial copy of the database. Rather, entries kept in online system log on disk are analyzed for recovery.

Note: In log based recovery system, a log is maintained, in which all the modifications of database are kept. Log consists of log records & for each activity of database, separate log record is made.

Categorization of Recovery algorithms

1. Caching (Buffering) of disk blocks

→ In this, one or more disk pages that include data items to be updated are cached into main memory buffers & then updated in memory before being written back to disk.

→ A collection of in-memory buffers called the DBMS cache is kept under control of DBMS for holding these buffers.

→ A directory is used to keep track of which database items are in the buffer.

→ A dirty bit is associated with each buffer, which is 0 if the buffer is not modified else 1 if modified

2. Write-ahead Logging

If the failure occurs, the RAM buffer that stores database information as well as the log may be lost which will cause loosing of information. Write-ahead logging protocol is derived to protect the system in such case. It states that

→ The old value cannot be replaced by its new value until undo type logging record has been permanently stored on disk.

→ Prior to commit operation of a transaction, the redo operation & undo operation of the log have been written permanently in the disk. DBMS recovery subsystem may maintain a list of active transactions not committed yet & all the commited & aborted transactions until the last checkpoints for efficient recovery.

DATE

3. Steal/no-steal and force/no-force

These specify the rules that govern when a page from database cache can be written to disk.

→ If a cache buffer page updated by transaction can't be written to disk before the transaction commits, the recovery method is called a no-steal approach. On the other hand, if the recovery protocol allows writing an updated buffer before the transaction commits, it is called steal.

→ If all pages updated by a transaction are immediately written to disk before the transaction commits, the recovery approach is called a force approach. Otherwise, no-force approach.

4) Checkpoints and Fuzzy Checkpointing.

Another type of entry in the log is called a Check point. It is a mechanism where all the previous logs are removed from the system & permanently stored in the storage device (disk). The checkpoint is like a bookmark. It is used to declare a point before which the DBMS was in the consistent state, and all transactions were committed. Taking a checkpoint consists of following actions:

→ Suspend execution of transactions temporarily.
→ Force write all main memory buffers that have been modified to disk.
→ Write a [checkpoint] record to the log & force-write the log to disk.
→ Resume executing transactions.

CLASSMATE

PAGE

DATE

To overcome delay transaction processing during the time needed to force-write the memory buffers, a technique is used called fuzzy checkpointing.

5) Transaction Rollback and Cascading Rollback

If a transaction fails for whatever reason after updating the database, but before the transaction commits, it may be necessary to roll back the transaction. If any data item values have been changed by the transaction & written to the database on disk, they must be restored to their previous values. The undo-type log entries are used to restore the old values of data items that must be rolled back.

If a transaction T is rolled back, any transaction S that has, in the temporary, read the value of some data item X written by T must also be rolled back. Similarly, once S is rolled back, any transaction R that has read the value of some data item Y written by S must also be rolled back, and so on. This phenomenon is called cascading rollback, and it can occur when the recovery protocol ensures recoverable schedules but does not ensure strict or cascadeless schedules.

DATE

Concept of no-undo/redo recovery based on deferred update.

The idea behind deferred update is to defer or postpone any actual updates to the database on disk until the transaction completes its execution successfully and reaches its commit point. After the transaction reaches its commit point & the log is force-written to disk, the updates are recorded in the database

If a transaction fails before reaching its commit point, there is no need to undo any operations because the transaction has not affected the database on disk in any way. Therefore, only REDO-type log entries are needed in the log, which include the new value (AFIM) of the item written by a write operation. The UNDO-type log entries are not needed since no undoing of operations will be required during recovery.

A drawback of this method is it limits the concurrent execution of transaction because all write-locked items remain locked until the transaction reaches its commit point. Additionally, it may require excessive buffer space to hold all updated items until the transactions commit.

Its main benefit is that transaction operations never need to be undone for two reasons

DATE

-> A transaction does not record any changes in the database on disk until after it reaches its commit point.

-> A transaction will never read the value of an item that is written by an uncommitted transaction

* Concept of recovery technique based on immediate update

Unlike the deferred database modification, the immediate database modification immediately modifies the data to database whenever any modification in data takes place. This method does not wait for the commit to make changes to the database. However, the operations are typically recorded in the log on the disk before they are applied to database, making recovery still possible.

If the transaction fails after recording some changes in the database but before reaching its commit point, the effect of its operation must be undone (UNDO) ie. the transaction must be Rolled Back.

If the transaction fails after reaching its commit point, the effects of its operation must be redone (REDO) because their effect may not have been recorded in the database. So, immediate update requires both UNDO & REDO during recovery. Therefore, it is also known as UNDO/REDO algorithm.

DATE

* Concept of shadow paging

→ Shadow paging is an alternative to log-based cash recovery which is useful if transaction executes serially

→ In this technique, database is partitioned into some number of fixed length block called as pages

→ The key idea behind it is to maintain two page tables during the lifetime of a transaction. These tables are called as current page table & shadow page table.

→ These two pages are created in the main memory which contains block number & pointers to each block. When transaction starts, both tables are identical

→ The shadow page table is never changed during the life of the transaction. The current page table is updated with each write operation. Before starting transaction, the shadow page table is copied to the hard disk

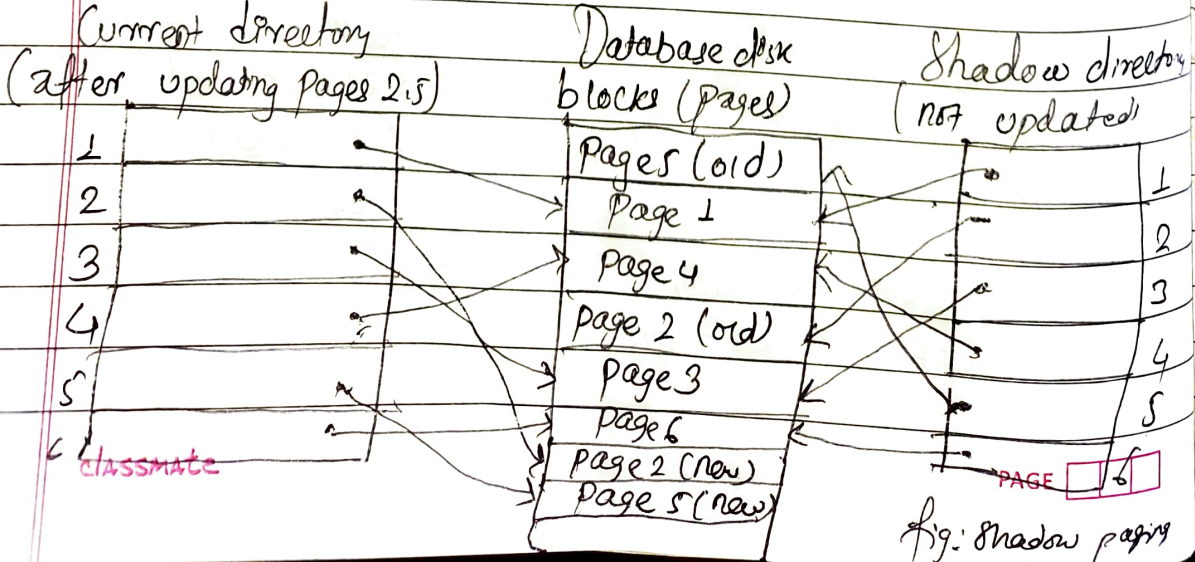

fig: Shadow paging

DATE

* Concept of database backup & recovery from catastrophic failures.

→ All the techniques we have discussed apply to non catastrophic failures.

→ A key assumption has been that the system log is maintained on the disk and is not lost as a result of the failure. Similarly, the shadow directory must be stored on disk to allow recovery when shadow paging is used.

→ The recovery manager of a DBMS 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 & the logs are periodically copied onto a cheap storage medium such as magnetic tapes or other large capacity offline storage devices

→ Databases from banks, insurance, stock market, etc are periodically backed up & moved to physically separate safe locations

→ Subterranean storage vaults have been used to protect such data from flood, storm, earthquake or fire damage

Questions asked from this chapter

Q. What is checkpoints in data recovery? How does it help in database recovery? Explain. (2078 - 5 marks) (2075 - short note)

Q. What are different approaches of Database recovery? What should log file maintain in log-based recovery? (2076 - 5 marks)

Q. What is meant by transaction rollback? What is meant by cascading rollback? Why do practical recovery methods use protocols that do not permit cascading rollback? (2072 - 5 marks)

Q. Why do we need database recovery? Discuss shadow paging technique. (2074 - 5 marks (2071 - 5 marks)

Q. Why do you need recovery? Discuss different types of failures. (2075 - 5 marks)