# UNIT-10

## Database Recovery Techniques

### ⊛. Recovery Outline and Categorization of Recovery Algorithms:

**Recovery Outline:** Recovery from transaction failures usually means that the database is restored to the most recent consistent state before the time of failure. To do this, the system must keep information about the changes that were applied to data items by the various transactions. This information is typically kept in the system log. A typical strategy for recovery may be ~~suzzo~~ summerized as follows:

i) If there is extensive damage to a wide portion of a database due to catastrophic failure, such as disk crash, the recovery method restores a past copy of database that was backed up.

ii) When the database on disk is not physically damaged, and a non catastrophic failure has occurred, the recovery strategy is to identify any changes that may cause an inconsistency in the database.

**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 and 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 lost which will cause loosing of information. Write-Ahead logging protocol is derived to protect the system in such case. Write-Ahead logging states that;

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

ii) Prior to commit operation of a transaction, the redo portion and undo portion of the log have been written permanently on the disk. To make the recovery process more efficient DBMS recovery subsystem may maintain a list of transaction details, which include the list of active transactions that are not committed yet as well as the list of all the committed and aborted transactions until the last checkpoints.

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 cannot 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, it is called no-force.

4. Checkpoints and Fuzzy Checkpointing → Another type of entry in the log is called a checkpoint. It is a mechanism where all the previous logs are removed from the system and permanently stored in the storage disk. The checkpoint is like a bookmark. The checkpoint 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:

i) Suspend execution of transactions temporarily.
ii) Force-write all main memory buffers that have been modified to disk.
iii) Write a [checkpoint] record to the log, and force-write the log to disk.
iv) Resume executing transactions.

To overcome delay transaction processing during the time needed to force-write all 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 and written to the database on disk, they must be restored to their previous values (BFIMs). 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 temprory, 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.

## ⊛ 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 and 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 transactions because all write-locked items remain locked until the transaction reaches its commit point. Additionaly, it may require excessive buffer space to hold all updated items until the transactions commit. This method's main benefit is that transaction operations never need to be undone, for two reasons:

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

ii) A transaction will never read the value of an item that is written by an uncommitted transaction.

## ⊛ Recovery Technique Based on Immediate Update:

In these techniques, when a transaction issues an update command, the database on disk can be updated immediately without any need to wait for the transaction to reach its commit point. It is not requirement that every update be applied immediately to disk; it is just possible that some updates are applied to disk before the transaction commits.

Provisions must be made for undoing the effect of update operations that have been applied to the database by a failed transaction. Theoretically, we can distinguish two main categories of immediate update algorithms:

i) If the recovery technique ensures that all updates of a transaction are recorded in the database on disk before the transaction commits, there is never a need to REDO any operations of committed transactions. This is called the UNDO/NO-REDO recovery algorithm. In this method, all

updates by a transaction must be recorded on disk before the transaction commits, so that REDO is never needed. Hence this method must utilize the steal/force strategy for deciding when updated main memory buffers are written back to disk.

ii) If the transaction is allowed to commit before all its changes are written to the database, we have the most general case, known as the UNDO/REDO recovery algorithm. In this case, steal/no-force strategy is applied.

## ⊛. Shadow Paging:

Shadow paging considers the database to be made up of a number of fixed sized disk blocks (say n) for recovery purposes. A directory with n entries is constructed, where the ith entry points to the ith database page on disk. The directory is kept in main memory if it is too large.

When a transaction begins executing, the current directory—whose entries point to the most recent or current database pages on disk—is copied into a shadow directory. The shadow directory is then saved on disk while the current directory is used by the transaction.

During the transaction execution, the shadow directory is never modified. When a write_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is not overwritten. Instead, the new page is written elsewhere on some previously unused disk block. figure below illustrates the concepts of shadow and current directories. For pages updated by the transaction two versions are kept. The old version is refrenced by the shadow directory and new version by the currend directory.

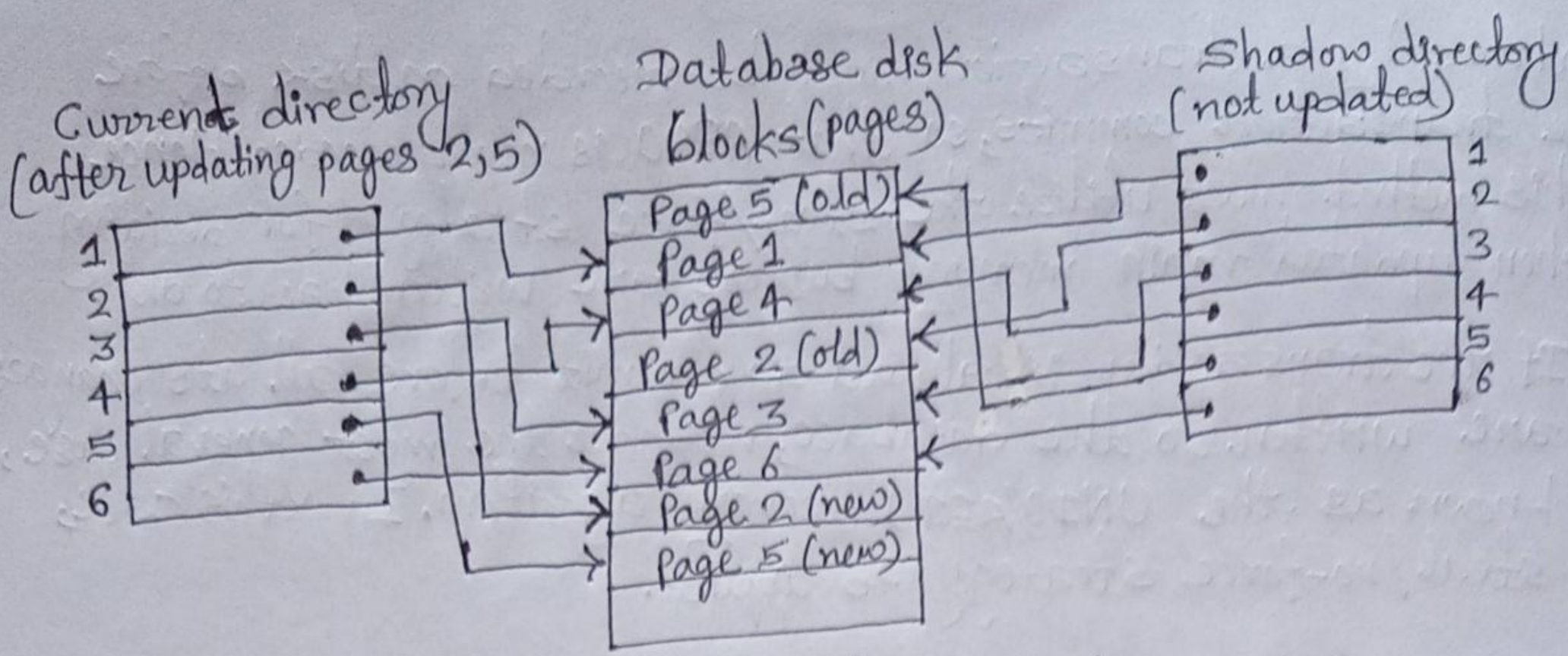


Fig. An example of shadow paging.

## ⊛ Database Backup and Recovery from Catastrophic Failures:

The recovery manager of a DBMS must be able to handle catastrophic failures such as disk crashes. The main technique to handle such crashes is a database backup, in which the whole database and the log are periodically copied onto a cheap storage medium such as magnetic tapes or other large capacity offline storage devices. 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.

Data from critical applications such as banking, insurance, stock market and other databases is periodically backed up in its entirety and moved to physically seperate safe locations. 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 at more frequent intervals than full database backup by periodically copying it to magnetic tape.