With in-place updating, it is necessary to use a log for recovery. In this case, the recovery mechanism must ensure that the BFIM of the data item is recorded in the appropriate log entry and that the log entry is flushed to disk before the BFIM is overwritten with the AFIM in the database on disk. This process is known as write-ahead logging.

A Redo-type log entry includes the new value (AFIM) of the item written by the operation since it is needed to redo the effect of the operation from the log.

The Undo-type log entries include the old value (BFIM) of the item since this is needed to undo the effect of the operation from the log.

If a cache buffer page updated by a transaction cannot be written to disk before the transaction commits, the recovery method is called a no-steal approach. (pin-unpin bit) 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, it is called a force approach. otherwise, it is called no-force. With the force rule, REDO will never be needed during recovery.

The deffered update (NO-UNDO) recovery scheme follows a no-steal approach. Typical database systems employ a steal/no-force strategy. The advantage of steal is that it avoids the need for a very large buffer space to store all updated pages in memory.

The advantage of no-force is that an updated page of a committed transaction may still be in the buffer when another transaction needs to update it, thus eliminating the I/O cost to

write that page multiple times to disk, and possibly to have to find it again from disk.

A checkpoint record is written into the log periodically at that point when the system writes out to the database on disk all DBMS buffers that have been modified. All transactions that have their [Commit, T] entries in the log before a [Checkpoint] entry do not need to have their WRITE operations redone in case of a system crash, since all their updates will be recorded in the database on disk during checkpointing.

The following actions happens during Checkpointing:

1. Suspend execution of transaction temporarily.
2. Force-write all main memory buffers that have been modified to disk.
3. write a [checkpoint] record to the log, and force-write the log to disk.
4. Resume executing transactions.

Fuzzy checkpointing technique is used to reduce the delay of transactions processing due to step 1. ~~To reduce the~~ In this technique, the system can resume transaction processing after a [begin_checkpoint] record is written to the log without having to wait for step 2 to finish.

NO-UNDO/REDO Recovery based on deffered update:-

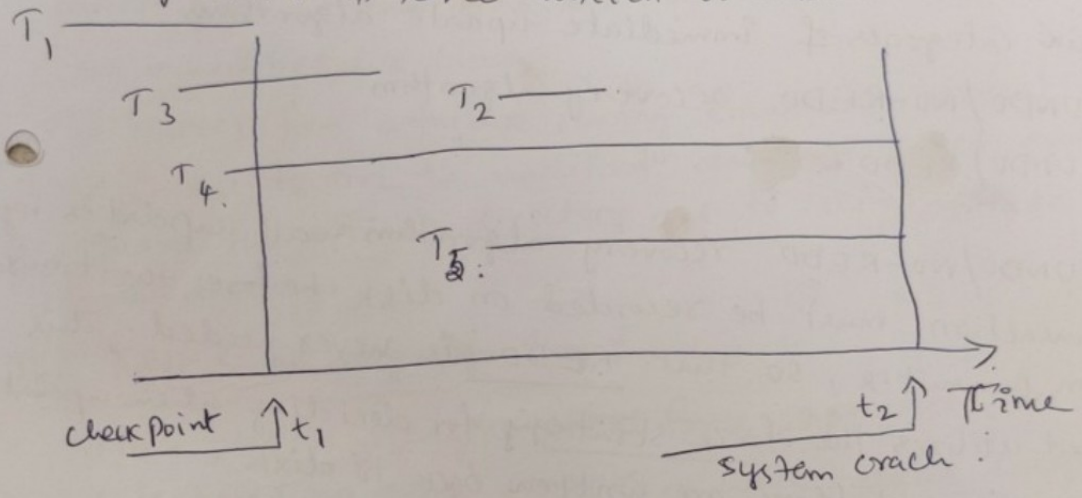
The technique of deffered update to defer or postpone any actual updates to the database on disk until the transaction completes its execution successfully and reaches its commit point.

During transaction execution, the updates are recorded only in the log and in the cache buffers. After the transaction reaches its commit point and the log is force written to disk, the updates are recorded in the database. The UNDO-type log entries are not needed since no undoing of operations will be required during recovery.

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 deffered update protocol is as follows:

1. A transaction cannot change the database on disk until it reaches its commit point.
2. A transaction does not reach its commit point until its REDO-type log entries are recorded in the log and the log buffer is force-written to disk.



The above fig illustrates a timeline for a possible schedule of executing transactions.

At time t_1 the checkpoint was taken and transaction T_1 had committed. T_3 and T_4 are not committed.

Before the system crash at time t_2, T_3 and T_2 were committed but not T_4 and T_5.

There is no need to redo the write_item operations of transaction T_1, as it is commited before the last checkpoint time t_1. The write_item operations of T_2 and T_3 must be redone; because they reached their commit points after the last checkpoint. Transactions T_4 and T_5 are ignored.

Adv:- 1. A transaction operation never need to be undone.

disadv:- 1. Limited concurrency.
2. Excessive buffer space may be required

Recovery Techniques 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.

The UNDO-type log entries, which include the old value (BFIM) of the item, must be stored in the log. Because UNDO can be needed during recovery of a failed transaction. These methods follow a steal strategy for deciding when updated main memory buffers can be written back to disk.

Two main categories of immediate update algorithms:

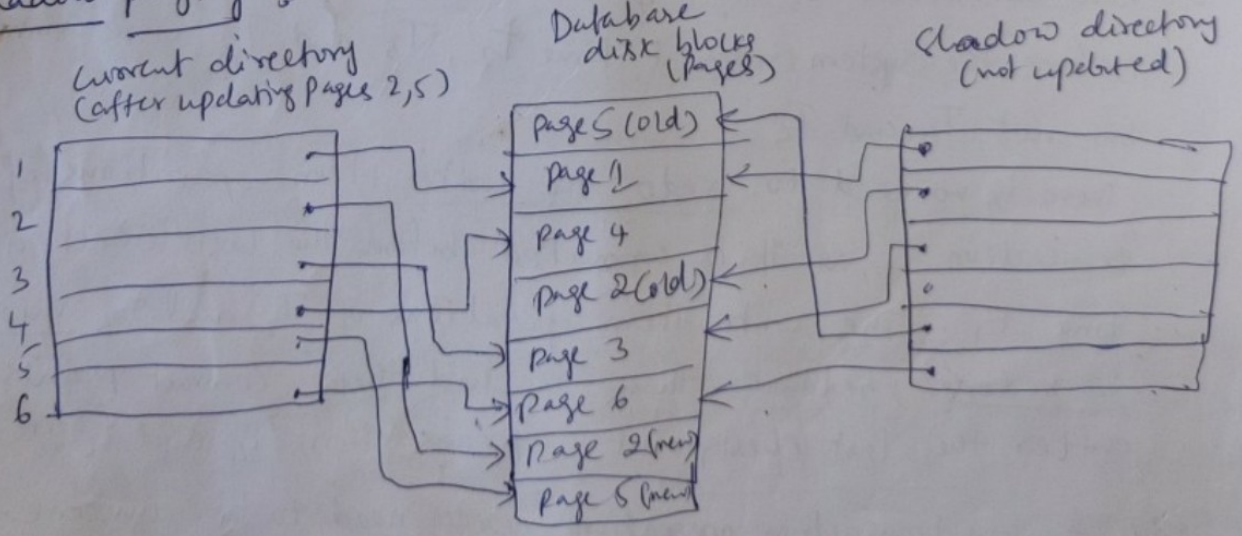
UNDO/NO-REDO recovery algorithm

UNDO/REDO " "

In UNDO/NO-REDO recovery algorithm, all updates by a transaction must be recorded on disk before the transaction commits, so that REDO is never needed. This method utilizes the force strategy for deciding when updated main memory buffers are written back to disk.

In the UNDO/REDO recovery algorithm, the transaction is allowed to commit before all its changes are written to the database. In this case, the steal/no-force strategy is applied.

Shadow Paging :-



A directory with n entries is constructed, where the ith entry points to the ith database page on disk, which is kept in main memory.

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.

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. The current directory entry points to the new disk block. The shadow directory is not modified and continues to point to the old unmodified disk block.

To recover from a failure during transaction execution, it is sufficient to free the modified database pages and to discard the current directory and to reinstantiate the shadow directory.

The ARIES Recovery algorithm :-

ARIES uses a steal/no-steal force approach for writing, and is based on three concepts : write-ahead logging (WAL)
repeating history during REDO
logging changes during UNDO

The ARIES recovery procedure consists of three main steps : analysis, REDO and UNDO.

The analysis step identifies the dirty pages in the buffer and the set of transactions active at the time of the crash.

The REDO phase actually reapplies updates from the log to the database. Generally, the REDO operation is applied only to committed transactions, but that is not the case with ARIES. Thus, only the necessary REDO operations are applied during recovery.

During the UNDO phase, the log is scanned backward and the operations of transactions that were active at the time of the crash are undone in reverse order.

In ARIES, every log record has an associated log sequence number (LSN) that is increasing and indicates the address of the log record on disk.

Each data page will store the LSN of the latest log record corresponding to a change for that page.

A log record is written for any of the following actions; write, commit, abort, undo and end.

Along with the log, two other tables are needed for effecient recovery: The transaction table and the dirty page table.

when a crash occurs, these tables are rebuilt in the analysis phase of recovery.

The transaction table contains transaction ID, status, and for each active transaction. the LSN of the most recent log record for the transaction.

The dirty page table contains an entry for each dirty page in the buffer, which includes the page ID and the LSN corresponding to the earliest update of that page.

A begin_checkpoint and end_checkpoint record is written to the log. LSN of the begin_checkpoint record is written to a special file. The special file is accessed during recovery to locate the last checkpoint information.

After a crash, the ARIES recovery manager accesses the checkpoint ~~is first accessed~~ through the special file. The analysis phase starts at the begin_checkpoint record and proceeds to the end of the log. when the end_checkpoint record is encountered, the transaction table and dirty page table are accessed. During analysis, the log records being analyzed may cause modifications to these two tables.

The REDO phase starts redoing at a point in the log where ~~it knows~~ the previous changes to dirty pages have already been applied to the database on disk. The smallest LSN, M of all the dirty pages in the dirty page Table is determined, which indicates the log position where ARIES needs to start the REDO phase.

REDO starts at the log record with LSN = M and scans forward to the end of the log.

once the REDO phase is finished, the database is in the exact state that it was in when the crash occured.

The Undo_set has been identified in the transaction table during the analysis Phase. Now, the UNDO phase proceeds by scanning backward from the end of the log and undoing the appropriate actions.

An example of Three transactions: T_1, T_2 & T_3 is shown below.

The log at point of Crash:-

Lsn	Last_lsn	Trans_id.	Type	Page_id	other_info
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.		

The transaction and Dirty page tables at time of checkpoint:

Transaction table.

Trans_id	Last_lsn	Status
T_1	3	Commit
T_2	2	in progress

Dirty page Table

Page_id	Lsn.
C	1
B	2

The transaction and Dirty page tables after the analysis phase:-

Transaction Table.

Trans_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