Chapter 4

Database Recovery Techniques

Outline

- Recovery Concepts
- Recovery Concepts Based on Deferred Update
- Recovery Concepts Based on Immediate Update
- Shadow Paging
- Recovery in Multi database Systems

Recovery Concepts

- Recovery from transaction failures usually means that the database is restored to the most recent consistent state before the time of failure.
- the system must keep information about the changes that were applied to data items by the various transactions.

Database Recovery

- Purpose of Database Recovery
  - To bring the database into the last consistent state, which existed prior to the failure.
  - To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).
- Example:
  - If the system crashes before a fund transfer transaction completes its execution, then either one or both accounts may have incorrect value. Thus, the database must be restored to the state before the transaction modified any of the accounts

Transaction Log

- For recovery from any type of failure data values prior to modification (BFIM - BeFore Image) and the new value after modification (AFIM – AFter Image) are required.
- These values and other information is stored in a sequential file (appended file) called *Transaction log*
- These log files becomes very useful in brining back the system to a stable state after a system crash.
- A sample log is given below. Back P and Next P point to the previous and next log records of the same transaction.

Cont’d

T ID	Back P	Next P	Operation	Data item	BFIM	AFIM
T1	0	1	Begin			
T1	1	4	Write	X	X = 100	X = 200
T2	0	8	Begin			
T1	2	5	W	Y	Y = 50	Y = 100
T1	4	7	R	M	M = 200	M = 200
T3	0	9	R	N	N = 400	N = 400
T1	5	nil	End			

Cont’d

- To recover from system failure , the system keeps information about the change in the system log
- Strategy for recovery may be summarized as :
- **Recovery from catastrophic**
  - If there is extensive damage to the wide portion of the database
  - This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure

Cont’d

- Recovery from non-catastrophic failure
  - When the database is not physically damaged but has be come in consistent
  - The strategy uses *undoing and redoing* some operations in order to restore to a consistent state

Data Update

Immediate Update:

As soon as a data item is modified in cache, the disk copy is updated.

- These update are first recorded *on the log and on the disk* by force writing, before the database is updated
- If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Cont’d

Deferred Update:

- All transaction updates are recorded in the local workspace (cache)
- During commit the updates are first recorded on the log and then on the database
- If a transaction fails before reaching its commit point undo is not needed because it didn’t change the database yet
- If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log

Cont’d

- **Shadow update**
  - The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.
- **In-place update:**
  - The disk version of the data item is overwritten by the cache version.

Transaction **Roll-back (Undo)** and **Roll-Forward (Redo)**

- If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
- A transaction’s operations are redone or undone.
  - ✓ **Undo**: Restore all BFIMs on to disk (Remove all AFIMs).
  - ✓ **Redo**: Restore all AFIMs on to disk.
- Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two.

Recovery Concepts Based on Deferred Update

(No Undo/Redo)

- A set of transactions records their updates in the log.
- At commit point under WAL scheme these updates are saved on database disk.
- After reboot from a failure the log is used to redo all the transactions affected by this failure.
- No undo is required because no AFIM is flushed to the disk before a transaction commits

Cont'd

- Two tables are required for implementing this protocol:
  - ✓ **Active table**: All active transactions are entered in this table.
  - ✓ **Commit table:** Transactions to be committed are entered in this table.
- During recovery, all transactions of the **commit table are** redone and all transactions of **active tables are ignored** since none of their AFIMs reached the database.

Recovery Concepts Based on Immediate Update

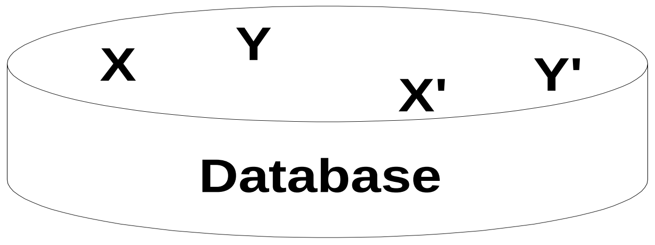
- ✓ For this reason the recovery manager undoes all transactions during recovery.
- ✓ No transaction is redone.
- ✓ It is possible that a transaction might have completed execution and ready to commit but this transaction is also undone.

Shadow Paging

- Maintain two page tables during life of a transaction: *current* page and *shadow* page table.
- When transaction starts, two pages are the same.
- Shadow page table is never changed there after and is used to restore database in event of failure.
- During transaction, current page table records all updates to database.
- When transaction completes, current page table becomes shadow page table.

Cont’d

- the shadow directory is not modified and continues to point to the old unmodified disk block.
- For pages updated by the transaction, two versions are kept.



X and Y: Shadow copies of data items
X' and Y': Current copies of data items

Cont’d

Transaction table and the Dirty Page table

- For efficient recovery following tables are also stored in the log during check pointing:
- **Transaction table:** Contains an entry for each active transaction, with information such as transaction ID, transaction status and the LSN of the most recent log record for the transaction.
- **Dirty Page table:** Contains an entry for each dirty page in the buffer.

Recovery in Multi-database Systems

- A multi-database transaction, may require access to multiple databases.
- These databases may even be stored on different types of DBMSs;
- For example, some DBMSs may be relational, whereas others are object oriented, hierarchical, or network DBMSs.
- Each DBMS involved in the multi-database transaction may have its own recovery technique and transaction manager separate from those of the other DBMSs.

Cont’d

- To maintain the atomicity of a multi-database transaction, it is necessary to have a two-level recovery mechanism.
- A **global recovery manager, or coordinator,** is needed to maintain information needed for recovery,
- **the local recovery managers** and the information they maintain (log, tables).
- The coordinator usually follows a protocol called the **two-phase commit protocol**