

# 07. Database Recovery

# Database Recovery

- **Focus:**

- Recovery from transaction failure
  - Various causes:
    - Logical Error e.g. bad input data, unavailable data, overflow, exceeding resource limits
    - System Errors e.g. deadlocks
  - Restoring the database to its most recent consistent state (before the failure)
  - Use 'recovery algorithms/strategies'
- Recovery from catastrophic failure (e.g. System Crash: hardware malfunction, DBMS software bug, OS bug etc). Database is physically damaged.
  - Restore a past copy of the database (backup copy) then reconstruct it to a more current state by reapplying/redoing operations

- **Implication:**

- The system must keep record of changes applied: Usually kept in system log

# Database Recovery: System Log

TRL_ID	TRX_NUM	PREV_PTR	NEXT_PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				



**TRL\_ID** = Transaction log record ID      **PTR** = Pointer to a transaction log record ID  
**TRX\_NUM** = Transaction number  
(Note: The transaction number is automatically assigned by the DBMS.)

# DR Concepts: Caching/Buffering of Disk Blocks

- Database Recovery process is closely linked to Operating Systems functions
- DB disk pages are buffered in the DBMS main memory cache
- Typically multiple disk pages are cached in MM buffers and then updated in memory before being written back to disk

**Caching is traditionally an OS function.¶**

**Due to its importance in this context, it is handled by the DBMS calling low-level OS routines¶**

# DR Concepts: Caching/Buffering of Disk Blocks

- The collection of in-memory buffers is under the control of the DBMS – e.g. keeping track of which database item is in which buffer
- When the DBMS requests action on some item, it first checks the cache directory to determine whether the disk page containing the item is in the DBMS cache
- If it isn't, the item must be located on disk and the appropriate disk page copied into the cache
- It may be necessary at this point to replace (flush) some of the cache buffers to make space for the new item

# DR Concepts: Dirty Bit, Pin-Unpin-Bit

- Associated with each page in the cache is a '**dirty bit**'
  - Set to **0** when the page is new in the cache
  - Set to **1** as soon as the buffer is modified
- Additional info also kept: the Transaction ID of the transaction that modified the buffer
- When buffer contents are flushed from the cache the contents must be written back to disk only if the dirty bit is **1**
- Also required for each page is the '**pin-unpin bit**'
  - A page in the cache is pinned if it cannot be written back to disk yet e.g. if the protocols in use restrict buffer pages from being written back to the disk until the transaction that has made changes to the buffer has committed

# DR Concepts: Buffer Flushing Strategies

- In-Place Updating:

- Writes buffer page to the same original position, thereby overwriting the old values

- Shadowing:

- Writes an updated buffer to a different location from the original copy
- Multiple versions of the data item could then be maintained
- Not a common/practical method
  
- Before Image (BFIM) – Old value of data item before updating
- After Image (AFIM) – New value of data item after updating

# DR Concepts: Write-Ahead Logging (WAL)

- When in-place updating is used, it is necessary to use a log for recovery
- The recovery mechanism must ensure that the BFIM of the data item is recorded in the log and that the log is flushed to disk before the BFIM is overwritten.
- This enables a potential UNDO, if it is required during recovery
  - A REDO-type log entry includes the new value (AFIM) of the item written by the operation
  - The UNDO-type log entries include the old value of the item, since it is needed to undo the effect of the operation
  - And UNDO/REDO algorithm would record both the BFIM and the AFIM in a single log entry



# DR Concepts: Write-Ahead Logging (WAL)

- Take Note: The DBMS Cache includes
  - Data file blocks
  - Index file blocks
  - Log file blocks
- When an update to a data block stored in the DBMS Cache happens, an associated log record is written to the last log buffer in the DBMS Cache
- With Write-Ahead Logging, log buffers containing the log records of a particular data block update must first be written to disk before the data block itself can be written from its main memory buffer

# Rules for Flushing Pages to Disk: Steal/No Steal

- If a cache buffer page updated by a transaction cannot be written to disk before transaction commits, the recovery method is called a **‘no-steal’** approach
  - The pin-unpin bit is set to 1 to indicate that the cache buffer cannot be written back
  - Steal is used when the DBMS cache manager needs a buffer frame for another transaction ... and it goes ahead to replace an existing page that had been updated but whose transaction has not committed
- If the recovery protocol allows writing an updated buffer before the transaction commits it is called a **‘steal’** approach.
  - The no-steal rule means that UNDO will never be needed during recovery, since a committed transaction will not have any of its updates on disk before it commits

# Rules for Flushing Pages to Disk: Force/No Force

- 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 a **no-force** approach

- The force rule means that REDO will never be needed during recovery, since any committed transaction will have all its updates on disk before it is committed.

## DR Concepts: Checkpoints in System Log

- Checkpoint = a type of log entry written periodically at that point when the system writes back to the database on disk all DBMS buffers that have been modified
- Consequently, all transactions that have entries in the log before a [checkpoint] entry don't have to have their WRITE operations REDONE (in case of system failure)
- The recovery manager must decide the interval in which to take a checkpoint (e.g. based on time intervals or number of transactions since the last checkpoint)

# DR Concepts: Policies for Recovery – Deferred Update

- Updating of the database on disk is postponed until after a transaction completes its execution successfully and reaches commit point
- Before the commit stage, all transaction updates are held in the main memory buffers
- Also, before commit, the updates are recorded persistently in the log file on disk and then after commit, the updates are written to the database on disk from the memory buffers
- At commit point, the log is force-written to disk and the updates recorded to the database on disk
- If the transaction fails before committing, it will not have changed the database in any way ... so no UNDO is needed

# DR Concepts: Policies for Recovery – Deferred Update

- It may be necessary to REDO the operations of the committed transaction, hence only REDO-type entries are needed in the system log which include the AFIM of the item to be written
- The approach is hence called No-UNDO/REDO algorithm
- In case of a large transaction, there is the risk of running out of buffer space (with many pages pinned)
- The Deferred update (No UNDO) recovery scheme follows the no-steal approach

# DR Concepts: Policies for Recovery – Immediate Update

- The database may be updated by some operation before the transaction reaches commit point
- Provision must be provided for undoing the effects of update operations applied by a failed transaction
- These operations must also be recorded in the log on disk by force-writing before they are applied to the database on disk ... making recovery possible
- If the transaction fails after recording some changes in the database but before reaching commit point, the effect of its operations on the database must be undone (Transaction must be rolled back).
- The log requires UNDO-type log entries which include the old values (BFIM). REDO entries may also be necessary
- The most common in practice

# DR Concepts: Policies for Recovery – Immediate Update

- If the recovery algorithm ensures that all updates of a transaction are recorded in the database on disk before the transaction commits, there's never a need to redo any of the operations: UNDO/No REDO Algorithm
- The method must utilize the steal/force strategy for deciding when updated memory buffers are written back to disk
- If the transaction is allowed to commit before all its changes are written to the database, the recovery technique would be the UNDO/REDO recovery algorithm
- The steal/no-force strategy is applied
- This is a most complex technique and the most commonly used in practice



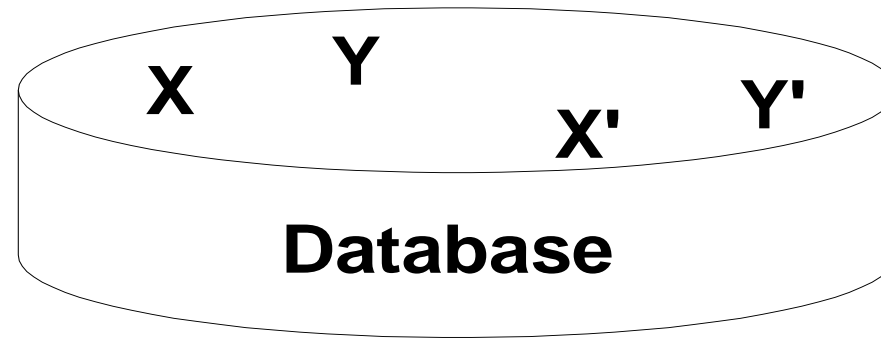
# DR Concepts: Policies for Recovery – Shadow Paging

- The recovery scheme doesn't require the use of a log (in a single user environment)
- The scheme assumes the database as consisting of a number of fixed sized disk pages
- A directory with  $n$  entries (same number as the disk pages) is constructed. It is intended that an entry will correspond to a page
- The directory is kept in main memory and all references to the pages on disk go through it
- When a transaction begins executing, the current directory (whose entries point to the current database pages on disk) is copied into a shadow directory. The shadow directory is then saved on disk

# DR Concepts: Policies for Recovery – Shadow Paging

- During execution the shadow directory is never modified. When an update operation is performed, a new copy of the modified database page is created, but the old copy of that page is not overwritten - it is written on some previously unused disk block
- The current directory is updated to point to the new disk block ... as the shadow directory (still unmodified) continues to point to the old unmodified disk block
- Recovery from failed transaction:
  - Simply free the modified database pages and discard the current directory.
  - Revert to the state of the database before the transaction by reinstating the shadow directory
- The technique is categorized as No UNDO/No REDO technique
- Committing a transaction is equivalent to discarding the shadow directory

# DR Concepts: Policies for Recovery – Shadow Paging



X and Y: Shadow copies of data items

X' and Y': Current copies of data items

# DB Backup & Recovery from Catastrophic Failure

- The DBMS Recovery Manager must also be equipped to handle more drastic failures
- Main Technique: Use of a database backup, where both the database and the log are periodically copied onto a cheap storage medium or other offline storage device
  - Backup is ideally moved and kept at a physically separate and safe location
- In case of failure, a recent (the latest) backup copy is reloaded and the system restarted
- It is convenient to (more frequently) make backups of the system log ... which is usually much smaller than the database
- A recent log would be invaluable at the point of reconstructing the best possible version of the database