DATABASE TECHNOLOGY - 1DL116
Spring 2007

An introductury course on database systems

http://user.it.uu.se/~udbl/dbt-vt2007/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/vt07/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

UPPSALA
UNIVERSITET

Introduction to Recovery Techniques

Elmasri/Navathe ch 19
Padron-McCarthy/Risch ch 23 and 24

Kjell Orsborn

Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

UPPSALA
UNIVERSITET

Recovery

- Recovery is needed after aborted transactions.
- The goal is to restore the database to an earlier and consistent state.
- Is possible by saving a log file.
- The recovery manager is a subsystem of the (DBMS that handles these problems.
- Strategy:
- a) If the disc crasches, fetch the latest backup copy of the database and use the log file to reproduce the latest updates.
- b) If some other type of failure has caused the inconsistency, eleminate (undo) the updates that led to the inconsistency.

Recovery . . .

- To be able to keep the atomicty principle for the transactions the system must handle different typesof failure that can cause that the execution of a transaction is aborted.
- The system must reassure that either:
  - all operations in a transaction succeed completely and their intended effect is registered in the database, or
  - the transaction is aborted without any side effect.
- The following operations are required:
  - Rollback: eleminate side effects of a failured transaction.
  - Undo: eleminate a single operation.
  - Redo: redo one/several operations (transactions).

Logging

- Logging updates in a log(file) required for recovery through in-place updating
- Write-ahead logging - flush log to disk before updating the database
- Page management:
  - No-steal/steal approach - in a *no-steal* approach updated pages *cannot* be written to disk before transaction commits.
  - No-force/force approach - in a *force* approach all updated pages are immediately written to disk when transactions commit.

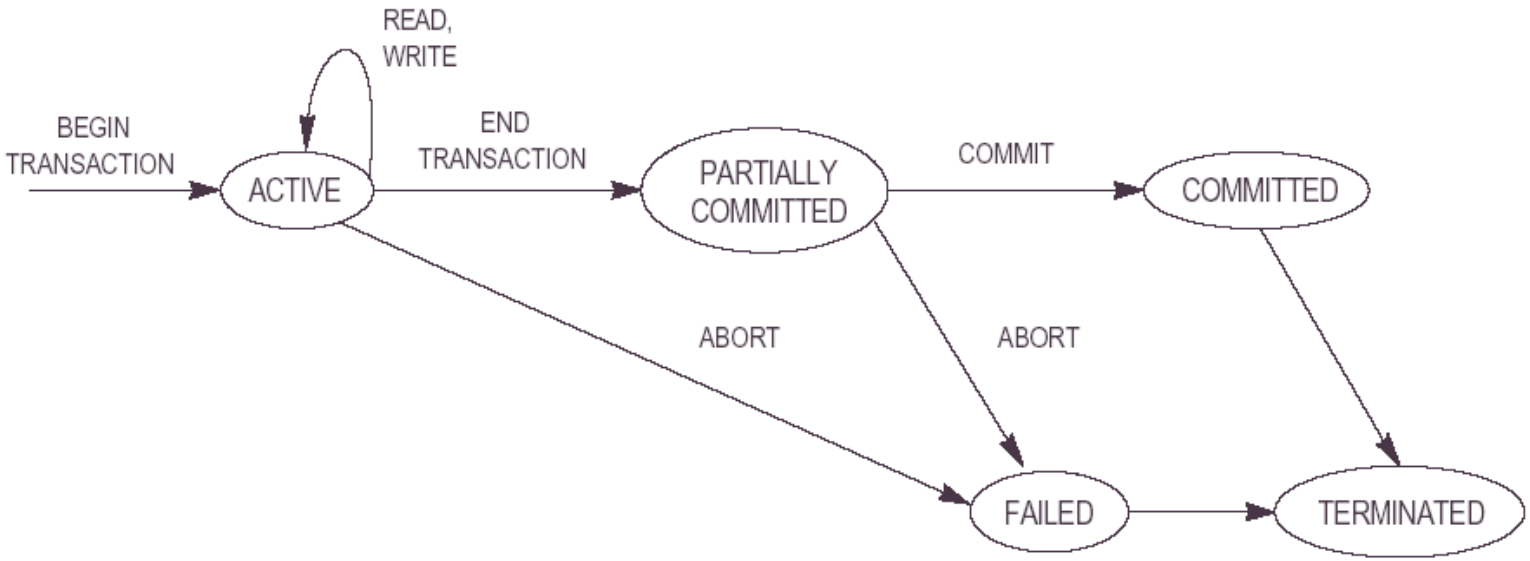
System log - the log file

- During the execution of transactions the following följande information is stored on the log (file):

(Start, T)	Marks the start for transaction T.
(Write,T,X,old_val,new_val)	Marks that T changes the value of X from old_val to new_val.
(Read, T, X)	Marks that T reads the value of X.
(Commit, T)	Marks that T is finished with all accesses and its effect can be introduced in the database.
(Check point)	A feature described later.

Commit point for a transaction

- When a transaction is finished with all its operations (and no errors have ocurred) it reaches its commit point.
- Failured transactions do not reach their commit point.



Rollback - cascading rollback

- **Rollback**: when a transaction fails to reach its commit point, its effects must be eleminated, i.e. all values that have been changed by the operations in the transaction must be restored.
- **Cascading rollback**: when rollback is applied to a transaction T, we must apply rollback to all transactions S that have read item values that has been updated by T. We must then do the same for transactions that have read values that each such S has updated and so forth.
- (Read, T, ...) records in the log file are used to decide if cascading (recursion) is required or not.

Deferred update
(or no-undo/redo)

- A recovery method that defers actual database updates until a transaction reaches its commit point.
- Under the execution of the operations, updates are registered in the log file. When the commit point is reached, first the log file is updated on secondary memory and thereafter the actual updates is written to the database.
- If a transaction fails before it reaches the commit point, no undo operations are required since the database has not been effected.

Recovery using the deferred update model

- Recovery according to the deferred update model:
  1. Start from the last record in the log file and traverse backwards. Create two lists:
     - C transactions that have reached their commit points
     - NC transactions that have not reached their commit points.
  2. Start from the beginning of the log file and redo all (Write,T,...) for all transactions T in the list C.
  3. Restart all transactions in the list NC.
- If the log file is long, step 2 will take long time. An improvement of this method is accomplished by introducing what's called check points.

Check points

- Check points are special records stored in the log file to mark that all write operations (for committed transactions) to this point have been introduced in the database.
- This means that it is not necessary to redo operations before this point when a crasch ocurres.
- The recovery manager decides when a check point should be created.

Creating checkpoints

The creation of a checkpoint usually include:

1. Suspend execution of transactions temporarily.
2. Force-write all main-memory buffers that have been modified to disk.
4. Write a checkpoint record to the log and force-write the log to disk.
5. Continue with the transactions.

Recovery using deferred updates with checkpointing
(multiuser version and assuming strict schedules)

- Recovery according to the deferred update model with check points:
  1. Start from the last record in the log file and traverse backwards until a check point is reached. Create two lists:
     - C transactions that have reached their commit points
     - NC transactions that have not reached their commit points.
  2. Start from the position after the check point in the log file and redo all (Write,T,...) for all transactions T in the list C.
  3. Restart all transactions in the list NC.
- Step 2 is much cheaper now since the set C is much smaller.

Immediate updates
(or undo/redo)

- In this update model, the effect of update operations is introduced in the database even before there commit point has been reached. Operations are registered in the log file (on disc) before they are applied to the database.
- If a transaction is aborted before the commit point, its side effects must be eliminated (rollback).
- To UNDO, i.e. eleminate, an operation means that the value of item X is reset to old_value.
  - (Write,T,X,old_value,new_value)

Recovery using immediate updates

(multiuser version and assuming strict schedules)

- Recovery according to the immediate update model:
  1. Start from the last record in the log file and traverse backwards until a check point is reached. Create two lists:
     - C transactions that have reached their commit points
     - NC transactions that have not reached their commit points.
  2. Start from the last record in the log file and apply the UNDO procedure to all (Write,T,...) where T ∈ NC.
  3. Start from the checkpoint and REDO all transactions (Write,T,...) such that T ∈ C.
  4. Restart all transactions in NC.

Shadow paging

- Alternative to log-based recovery
- Idea: maintain two page tables during the lifetime of a trans-action - the *current* page table, and the *shadow* page table
- Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered. Shadow page table is never modified during execution
- To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.
- Whenever any page is about to be written for the first time, a copy of this page is made onto an unused page. The current page table is then made to point to the copy, and the update is performed on the copy

Shadow paging cont'd

- To commit a transaction:
  1. Flush all modified pages in main memory to disk
  2. Output current page table to disk
  3. Make the current page the new shadow page table
     - keep a pointer to the shadow page table at a fixed (known) location on disk.
     - to make the current page table the new shadow page table, simply update the pointer to point to current page table on disk
- Once pointer to shadow page table has been written, transaction is committed.
- No recovery is needed after a crash — new transactions can start right away, using the shadow page table.
- Pages not pointed to from current/shadow page table should be freed (garbage collected).

Shadow paging cont'd

- Advantages of shadow-paging over log-based schemes – no overhead of writing log records; recovery is trivial
- Disadvantages:
  - Commit overhead is high (many pages need to be flushed)
  - Data gets fragmented (related pages get separated)
  - After every transaction completion, the database pages containing old versions of modified data need to be garbage collected and put into the list of unused pages
  - Hard to extend algorithm to allow transactions to run concurrently