Database Systems
- Recovery Techniques -

Agnès Voisard
Institute of Computer Science,
Databases and Information Systems Group
and Fraunhofer FOKUS

2025
v2

Notes

Notes

Introduction

Recovery techniques:

- Used to recover from **transaction failures** (e.g., system crashes and transaction errors)
- Database *restored* to some state from the past so that a correct state can be reconstructed from that past state. (correct state close to the time of failure)
- Concepts used: **System log, check points, commit points**.

Catastrophic vs. non catastrophic failures

◇ **Catastrophic failure**:
=> Recovery method **restores** a past copy of the DB that was dumped to archival storage, and
reconstructs a more current state by **redoing** committed transaction operations from the log up to the time of failure.

Database backup: whole DB + log periodically copied onto a storage medium.

◇ **Non catastrophic failure but DB inconsistent**:
(cf. slide Trans. Proc. 7).
reverse the changes that caused inconsistency by **undoing** some operations (+ maybe redo some operations).

Update types and recovery

2 main techniques:

- **Immediate updates**
  DB updated before the transaction reaches its commit point.
- **Deferred updates**
  DB not updated until after a transaction reaches its commit point.

System concepts

Relationship with OS functions

E.g., Buffering and cashing disk pages in MM.

Disk pages that include the data item to be updated are cached into MM

and updated in MM before being written back to disk.

Cashing: OS function but sometimes handled by the DBMS.

Cascading rollback

Transaction T1 is rolled back,
any transaction T2 that has read the values of some X written by T1 must be rolled back
any transaction T3 that has read the value of some Y written by T2 ...
Time consuming!!
Example.

System log

Log (or **journal**): to recover from failure.
Keeps track of transactions that affect the value of DB items.
Types of entries
(T is a unique transaction *id* generated by the system):

- `[start-transaction, T]`: records that T has started execution
- `[write-item, T, X, oldvalue, newvalue]`
- `[read-item, T, X]`
- `[commit, T]`: records that T has completed successfully
- `[abort, T]` records that T has been aborted

System log (cont'd)

Assumptions:

Transactions are not nested.
All permanent changes to the DB occur within transactions.

When system crashes (*recovery techniques*):
log contains a record for every *Write* operation.
Undone by tracing them back.

Or redo operations by tracing forward through the log:
Set all items changed by a *Write* to their new value.

Commit point of a transaction

T reaches its **commit point** when
- all its operations that access the DB have been executed successfully, and
- the effect of the operations on the DB are recorded in the log.

Beyond the commit point it is committed and the effect are permanently recorded in the DB.
Transaction writes an entry `[commit,T]` into the log.

Commit point of a transaction (cont'd)

If system failure:

Search for the transactions that have started and not committed (have not written their `[commit, T]` entry yet), and roll them back to undo their effect on the database during the recovery process.

Transactions that have written their commit in the log must have recorded all their write operations in the log so their effect on the database can be redone from the log entry.

Commit point of a transaction (cont'd)

Log file is kept on disk:

Usually keep one block in MM until is filled and write it back to disk once.
When system crash, only entries written back to disk are considered in the recovery process.

Force-writing the log file before committing a transaction.

Check points in the system log

Other type of entry in the log: `[checkpoint]` record.

Written periodically (log) when the system writes to the DB on disk the effect of all *Write* operations of committed transactions.

All transactions that have their `[commit,T]` before a checkpoint do not need to have their *Write* redone when system crashes.

Recovery manager decides at which intervals to take a checkpoint.

Check points in the system log (cont'd)

Taking a checkpoint:

- Suspend execution of transactions
- Force-write all update operations of committed transactions from MM buffer to disk
- Write a `[checkpoint]` record to the log and force-write the log to disk
- Resume executing transactions

Check point: "When it is written on disk"

When? Decision of the *recovery manager*.
Interval measured in time: Every *m* minutes,
or f (number of transactions after last checkpoint).

Summary

- Basic ideas of recovery techniques
- Immediate, deffered update
- Log, commit, check point

What will come next?

1. Welcome to Database Systems
2. Introduction to Database Systems
3. Entity Relationship Design Diagram (ERM)
4. Relational Model
5. Relational Algebra
6. Structured Query Language (SQL)
7. Relational Database Design - Functional Dependencies
8. Relational Database Design - Normalization
9. Online Analytical Processing + Embedded SQL
10. Data Mining
11. Physical Representation - Storage and File Structure
12. Physical Representation - Indexing and Hashing
13. Transactions
14. Concurrency Control Techniques
15. Recovery Techniques
16. Query Processing and Optimization