# Logging and Recovery

Erhard Rahm, University of Leipzig, Germany

## SYNONYMS

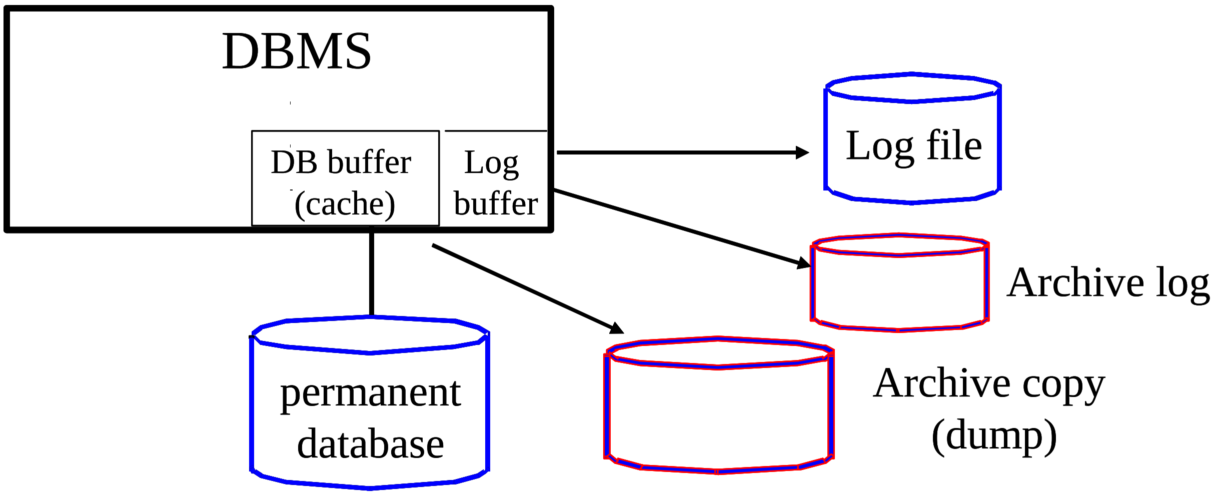
Failure handling; rollback, undo, redo; checkpoint; backup; dump

## DEFINITION

Logging and recovery ensure that failures are masked to the users of transaction-based data management systems by providing automatic treatment for different kinds of failures, such as transaction failures, system failures (crashes), media failures and disasters. The main goal is to guarantee the atomicity (A) and durability (D) properties of ACID transactions by providing undo recovery for failed transactions and redo recovery for committed transactions. Logging is the task of collecting redundant data needed for recovery.

## MAIN TEXT

The ACID concept requires that no data changes of failed transactions remain in the database. Failed transactions thus have to be rolled back by undoing all their changes (undo recovery). On the other hand, data changes of successfully ended (committed) transactions must not be lost but have to survive possible failures. Failure treatment thus implies a redo recovery for committed transactions. Recovery support is typically provided for transaction failures during normal processing (transaction recovery), for system failures (crash recovery), and media failures (media recovery). In addition, disaster recovery can deal with the complete destruction of a computer center, e.g. due to an earthquake or terror attack. Recovery is typically based on logging, i.e. the collection of protocol data recording which transactions have been executed and which changes have been performed by them.



The Figure shows components of a central database management system (DBMS) involved in logging and recovery. The database objects (e.g. tables, records) are persistently stored in the *permanent database,* typically on one or several disks. All database operations including updates are performed in main memory. For this reason, pages of the database are cached in a main memory buffer (database buffer). Log records are persistently stored in a sequential *log file* on dedicated disks. Log records are written for the start, rollback and commit of transactions as well for every database change. For performance reasons log records are first collected in a log buffer in main memory, which is written to the log file when it becomes full or when a transaction commits. A transaction is committed when its commit record is logged on the log file.

Numerous approaches have been proposed and current database management systems provide efficient implementations for logging and recovery. The major tasks to be solved for dealing with the mentioned types of failures are:

- *Transaction recovery* (rollback) is performed when a transaction fails during normal processing, e.g. due to a program error or invalid input data. The log records in the log buffer and in the log file are used to undo the changes of the failed transaction in reverse order.
- *Crash recovery* is needed when the whole database (transaction) system fails, e.g. due to a hardware or software error. All transactions which were active and not yet committed at crash time have failed so that their changes must be undone. The changes for transactions that have committed before the crash must survive. A redo recovery is needed for all changes of committed transactions that have been lost by the crash because the changed pages resided only in main memory but were not yet written out to the permanent database. Periodically writing out modified pages, e.g. within so-called *checkpoints*, help to reduce the amount of redo work during crash recovery. Furthermore, the number of relevant log records and thus the size of the log file can be reduced by checkpoints.
- *Media recovery* deals with failures of the storage media holding the permanent database, in particular disk failures. The traditional database approach for media recovery uses archive copies (dumps) of the database as well as archive logs (see Figure). Archive copies represent snapshots of the database and are periodically taken. The archive log contains the log records for all committed changes which are not yet reflected in the archive copy. In the event of a media failure, the current database can be reconstructed by using the latest archive copy and redoing all changes in chronological order from the archive log. A faster recovery from disk failures is supported by disk organizations like RAID (redundant arrays of independent disks) which store data redundantly on several disks. However, they do not eliminate the need for archive-based media recovery since they cannot completely rule out the possibility of data loss, e.g. when multiple disks fail.
- *Disaster recovery* can be achieved by maintaining a backup copy of the database at a geographically remote location. By continuously transferring log data from the primary database to the backup and applying the changes there, the backup can be kept (almost) up-to-date.

## CROSS REFERENCES

ACID, Crash recovery, Multi-level recovery and ARIES, Application recovery, RAID, Buffer management

## REFERENCES

1. Gray, J., A. Reuter (1993): *Transaction Processing: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann.
2. Haerder, T.; A. Reuter (1983): *Principles of Transaction-Oriented Database Recovery*. ACM Comput. Surv. 15(4): 287-317