Log Records

The most widely used structure for recording database modifications is the **log**.

The log is a sequence of **log records**, recording all the update activities in the database.

There are several types of log records.

An **update log record** describes a single database write. It has these fields:

- **Transaction identifier:** which is the unique identifier of the transaction that performed the write operation.
- **Data-item identifier:** which is the unique identifier of the data item written.

{Typically, it is the location on disk of the data item, consisting of the block identifier of the block on which the data item resides, and an offset within the block.}

- **Old value:** which is the value of the data item prior to the write.
- **New value:** which is the value that the data item will have after the write.

Represent an update log record as<Ti , Xj , V1, V2>,
indicating that transaction T_i has performed a write on data item X_j . X_j had value V_1 before the write, and has value V_2 after the write.

Other special log records exist to record significant events during transaction processing, such as the start of a transaction and the commit or abort of a transaction. Among the types of log records are:

- <T_i start>. Transaction T_i has started.
- <T_i commit>. Transaction T_i has committed.
- <T_i abort>. Transaction T_i has aborted.

Whenever a transaction performs a write, it is essential that the log record for that write be created and added to the log, before the database is modified.

Once a log record exists, we can output the modification to the database if that is desirable.

Also, we have the ability to *undo* a modification that has already been output to the database.

We undo it by using the old-value field in log records.

log records to be useful for recovery from system and disk failures, the log must reside in stable storage.

For now, we assume that every log record is written to the end of the log on stable storage as soon as it is created.

Database Modification

a transaction creates a log record prior to modifying the database.

The log records allow the system to undo changes made by a transaction in the event that the transaction must be aborted;

they allow the system also to redo changes made by a transaction if the transaction has committed but the system crashed before those changes could be stored in the database on disk.

To understand the role of these log records in recovery, we need to consider the steps a transaction takes in modifying a data item:

1. The transaction performs some computations in its own private part of main memory.
2. The transaction modifies the data block in the disk buffer in main memory holding the data item.
3. The database system executes the output operation that writes the data block to disk.

We say a transaction *modifies the database* if it performs an update on a disk buffer, or on the disk itself; updates to the private part of main memory do not count as database modifications.

If a transaction does not modify the database until it has committed, it is said to use the **deferred-modification** technique.

If database modifications occurwhile the transaction is still active, the transaction is said to use the **immediate-modification** technique.

Deferred modification has the overhead that transactions need to make local copies of all updated data items; further, if a transaction reads a data item that it has updated, it must read the value from its local copy.

The recovery algorithms we describe in this chapter support immediate modification

A recovery algorithm must take into account a variety of factors, including:

- The possibility that a transaction may have committed although some of its database modifications exist only in the disk buffer in main memory and not in the database on disk.

- The possibility that a transaction may have modified the database while in the active state and, as a result of a subsequent failure, may need to abort.

Because all database modifications must be preceded by the creation of a log record, the system has available both the old value prior to the modification of the data item and the new value that is to be written for the data item. This allows the system to perform *undo* and *redo* operations as appropriate.

- **Undo** using a log record sets the data item specified in the log record to the old value.
- **Redo** using a log record sets the data item specified in the log record to the new value.