Transactions

Joseph Kawamura

What is a Transaction?

- Unit of program execution that consists of all operations between the begin transaction and end transaction
- **Atomicity**: A transaction executes entirely or not at all
- **Consistency**: Preserves the consistency of the database (job of programmer)
- **Isolation**: Must operate independently of concurrent database operations
- **Durability**: Transaction’s changes must persist even if system crashes

Example

- Transaction where $50 is transferred from account A to account B

```
read(A);
A := A – 50;
write(A);
read(B);
B := B + 50;
write(B);
```

Example ACID properties

- **Atomicity**: If the write(B) operation were to fail, A would lose $50 and B would remain the same.
    - The transaction would be a failure and the changes would be rolled back from a log.
- **Consistency**: total amount of money in accounts A and B must remain the same (can’t create or destroy money).
- **Isolation**: If another transaction were to read the sum of A and B mid-transaction, it would receive read the wrong value.
    - Solution: Serial transactions, or better yet, Concurrent transactions.
- **Durability**: Any system failure must not cause a loss of data on the transfer

Storage

- Transaction **atomicity** and **durability** are dependent upon storage
- **Volatile Storage**: Lost when system crashes (e.g. RAM)
- **Non-volatile Storage**: Survives crashes (e.g. SSD, Magnetic Tape)
- **Stable Storage**: Theoretically never lost, requires replicating information across multiple non-volatile storage mediums (e.g. RAID Storage)
- Writing to stable storage allows for **durability**
- Logging old data to stable storage allows for **atomicity**

Atomicity

- A transaction that does not complete successfully is termed **aborted**
  - Any changes made are **rolled back** using a **log**
  - Can be restarted (if system failure caused abortion) or killed (if logical error)
- A successful transaction is termed **committed**
  - Cannot be undone
- Transaction is only **partially committed** until enough information is written from memory to the disk to where the transaction could be recreated in the event of failure.

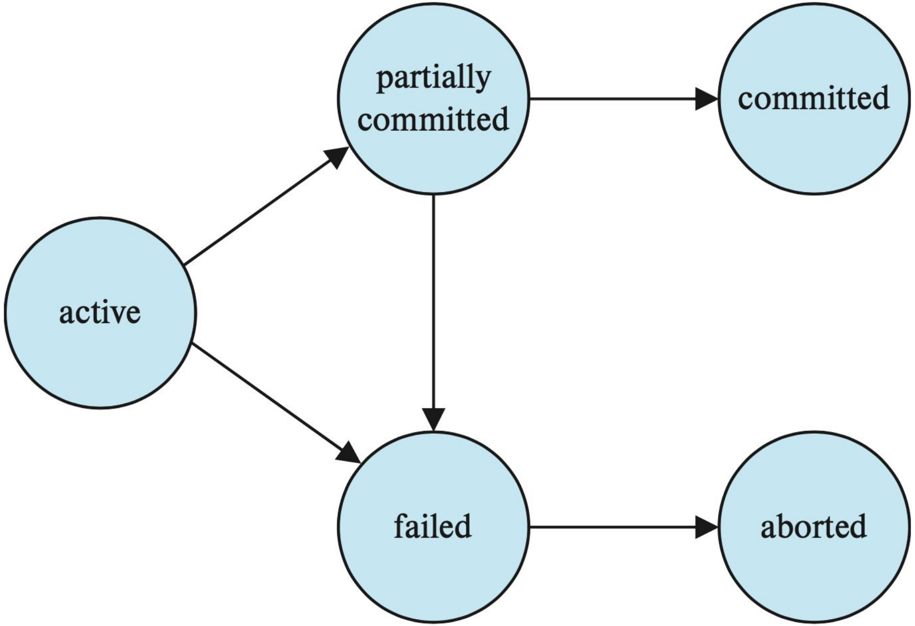

Figure 17.1 State diagram of a transaction.