1.3.2 Transaction Processing

What is a transaction?

A transaction is **a sequence of operations that is treated as a single unit of work.**

For example, transferring £100 from one bank account to another involves:

1. Deduct £100 from Account A.
2. Add £100 to Account B.

Both operations need to be completed successfully for the transaction to be considered successful.

If one operation fails, the database should be returned to its original state.

ACID

Database transactions should follow four properties known as **ACID**:

- **A — Atomicity**
- **C — Consistency**
- **I — Isolation**
- **D — Durability**

These properties help ensure that transactions are processed reliably

Atomicity

Atomicity means that a transaction is treated as a single unit.

Either:

- **all operations are completed**, or
- **none of the operations are completed**.

Example

A customer transfers £100 from Account A to Account B.

Account A: -£100

Account B: +£100

If the money is deducted from Account A but the operation adding it to Account B fails, the transaction must be **rolled back**.

The database should return to its state before the transaction began.

Consistency

Consistency means that **a transaction must follow the rules of the database**.

Any rules and constraints defined for the database must still be satisfied after the transaction.

Example

A database has a rule that an account balance cannot be negative.

If a transaction would cause the balance to become negative, the transaction should not be completed.

Isolation

Isolation means that **transactions should not interfere with each other.**

A transaction should appear to operate independently of other transactions.

Example

Two people attempt to purchase the last available concert ticket at the same time.

The database must prevent both transactions from successfully purchasing the same ticket.

One transaction should complete before the other can access the relevant data.

Record Locking

When a transaction is using a record, the database can lock the record so that another transaction cannot modify it until the first transaction has finished.

Durability

Durability means that **once a transaction has been successfully completed, the changes are permanent.**

The changes should not be lost if there is a:

- power failure
- system crash
- hardware failure

Example

A customer transfers £100 and the transaction is successfully completed.

If the database server immediately loses power, the transfer should still be recorded when the system restarts.

This can be achieved using methods such as **persistent storage and backups/logs**.

Redundancy

Having additional copies of data or system resources so that a failure does not result in data being lost or the system becoming unavailable.

For example:

- maintaining backup copies of a database
- storing data on multiple storage devices
- having redundant servers