Relational Databases and Normalisation

Core Info and Definitions

A relational database contains tables, making use of relations between them to store data with minimal data redundancy.

Normalisation is a process used to come up with the best possible database design by following a set of formal rules. It ensures:

- No data is unnecessarily duplicated.
- Data is consistent throughout the database

Transaction Processing and ACID

ACID

Atomicity	A rule that requires a transaction to be fully completed or not started at all. E.g. Rolling back changes if a system crashes mid-payment.
Consistency	Any changes to the database must retain its overall state/integrity. E.g. if an attempt is made to add students beyond maximum capacity, it fails.
Isolation	Each transaction is executed and committed to the database in isolation. E.g. record locking.
Durability	Once a transaction has been committed, it must be processed until completion. If it requires multiple actions, all of them must be completed before committing. I.e. saving changes.

Transaction Processing

Measures need to be taken in order to ensure that multiple users using a database does not harm referential integrity.

Record locking prevents simultaneous access to a record to prevent updates from being lost or inconsistencies in data arising.
However, deadlock can occur. This is when a committed process depends on data from a locked record, which is locked because of a running process that depends on another record locked by the former process. This can be solved with one of the following:

Serialisation	Ensures transactions do not overlap in time, so they cannot interfere with each other or lead to lost updates. A transaction cannot start until the previous one is complete. This is implemented with timestamp ordering.
Timestamp ordering	When a transaction starts, it is assigned a timestamp. If two processes are simultaneously affecting the same record/table, the process with the earlier timestamp is executed first. Each object in a database has a read timestamp and a write timestamp to mark when it was last read from or written to. If on write the object's read timestamp does not match the process', it knows that another process is operating on the object.
Commitment ordering	Ensures that transactions are not lost when two or more users are trying to access the same object. Transactions are ordered in terms of their dependencies on each other as well as the time they were initiated. Helps prevent deadlock.
Redundancy	Having multiple identical systems all manipulated at the same time. If one system goes down, another can instantly start being used to replace it until it comes back online.

First Normal Form

1. A table contains unique records.
2. Data in fields is atomic - i.e. no grouped data in fields, e.g. grades for three separate subject should be in three separate records, not one.

Second Normal Form

1. The table is in first normal form.
2. There are no partial dependencies - i.e. a field depends on only part of the primary key (seen in composite keys).

Third Normal Form

1. The table is in second normal form.
2. There are no transitive dependencies - i.e. fields **ONLY** depend on the primary key, not **any** other field.

Revision #6
Created 2026-01-19 12:55:50 UTC by Asad
Updated 2026-01-23 12:45:58 UTC by Asad