A simple example

A simple design for DB for managing a company's orders for spare parts:

- Keeps track of suppliers we use and items we require
- Also track details of outstanding orders

Three tables:

- suppliers (snum, sname, status, city)
- parts (pnum, pname, colour, weight)
- ordered_from (snum, pnum, quantity)

We assume status is city-dependent, e.g. a delivery charge.

A worse design

A worse design (because of more redundancy), would be to move (status) column into the ordered_from table.

Now the same city-dependent numbers are being repeated many times.

Design principle: avoid redundancy

We want to avoid redundancy wherever possible.

We try to represent one fact in one place, rather than in multiple places.

Issues arising from redundancy

- Duplicating info in several places wastes space.
- If we change one copy of a piece of info, we need to make sure we change them all, which is inconvenient.

Normalisation

Normal forms

Normal forms capture desirable traits that reduce risk of certain DB problems.

They are ordered by number, with higher numbers denoting stricter conditions.

First normal form

R is in First Normal Form (1NF) if and only if it contains atomic values only (no multi-valued attributes). We took this as given when we started looking at DBs.

A poor design

first (snum, status, city, pnum, quantity) and parts (pnum, pname, colour, weight)

Here we've combined the info on suppliers and orders together.

Problems:

- The city value for suppliers is replicated many times.
- Can't record suppliers details until that supplier supplies at least one part.
- If we delete the last tuple for a supplier, we lose all the information about that supplier.

Functional Dependencies

Dependencies capture how different attributes in table relate to one another.

Definition: Given relation R, attribute Y is functionally dependent on attribute X if and only if, whenever two tuples of R agree on their X-value, they are also guaranteed to agree on their Y-value. Notated (X → Y).

By analysing these dependencies with normalisation, we can reduce redundancy problems.

Functional dependencies in our poor design

parts:

- pnum → pname
- pnum → weight
- pnum → colour

first:

- pnum, snum → status, city, quantity

- snum → status, city
- city → status

In this example, pnum and snum together form the key for the 'first' table, but some attributes are determined only by part of this key (snum). This is called a non-full dependency.

Second normal form

A DB without any non-full dependencies is in Second Normal Form (2NF).

A second (now better) design

If we split first(…) into second(snum, status, city) and ordered_from(snum, pnum, quantity), then we have a DB in 2NF.

There are still some problems:

- The status value for each city is replicated many times.
- Can't enter status values for a city until we have a supplier located in that city.
- If we delete the only tuple for a particular city, we lose the status info for that city.

Dependencies in the second design

- snum → city
- city → status
- snum → status

Status has a transitive dependence on snum via city.

We can fix this by splitting the table into suppliers(snum, city) and cities(snum, status).

Third normal form

DBs without any transitive dependencies are in Third Normal Form.