COURSEWARE

Data Normalisation

Contents

Overview

Normalisation is the process of modelling the way a database might look by designing its schema.

Every database itself is built from a well-structured schema, which describes the structure of the database - tables, fields and their relationships.

We've seen the use of Entity Relationship Diagrams for describing this, but we may need to further refine the design; one way in which we should do this is through normalisation.

Normalisation is needed to reduce redundant data within the database.

Normal Forms

Database modelling strives to adhere to one of the following paradigms:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

Each form iterates over the other - a 2NF table will always also conform to 1NF, and a 3NF table will always conform to 2NF *and* 1NF.

First Normal Form

For a table to conform to *1NF*, the following 4 rules need to be followed:

- each column should only contain a single value
- values stored in a column should all be of the same type
- all columns in a table should have unique names
- the data should not be stored in any particular order

As an example, we'll fix the following table:

student_id	forename	subject
1	Jeff	Biology, Chemistry
2	Cyrus	Physics
3	John	Music, Latin

Here, our table doesn't conform to all four rules - there are multiple values in the `subject` columns.

We can fix the table to display multiple records for these students, which adheres to 1NF:

student_id	forename	subject
1	Jeff	Biology
1	Jeff	Chemistry
2	Cyrus	Physics
3	John	Music
3	John	Latin

Second Normal Form

The second level of normalisation, 2NF, asserts that tables should not contain **partial dependencies**.

Generally, data within a table will **depend** on a certain column.

For our student example earlier, this is the `student_id` column - because each student, even if they have the same `name` and take the same `subject`, will always have a *unique* `student_id`.

Thus, data within a table is usually **dependent** on *primary keys*.

Partial dependency is slightly different.

For a simple table like our student example, the `student_id` can uniquely identify all the records within the table - but this is, of course, not always true.

For example, let's say that, in our games shop database, we had the `forename` field in a both our `customers` and `orders` tables:

customer_id	forename	surname
1	Jeff	Darnielle
2	Cyrus	Douglas

order_id	fk_customer_id	forename	shipped
1	2	Cyrus	true
2	1	Jeff	true
3	1	Jeff	false
4	2	Cyrus	true

This does not conform to 2NF, because the `forename` column in our `orders` table relies on the `fk_customer_id` column, which is a foreign key linking to `customer_id` in our `customers` table.

This is a **partial dependency** - all the information in our `orders` table should depend on its primary key `order_id`.

Here, we'd fix the problem by dropping the `forename` column from `orders`:

customer_id	forename	surname

customer_id	forename	surname
1	Jeff	Darnielle
2	Cyrus	Douglas

order_id	fk_customer_id	shipped
1	2	true
2	1	true
3	1	false
4	2	true

Third Normal Form

The final level of normalisation, 3NF, requires that our tables do not have any **transitive dependencies**.

A **transitive dependency** is a non-key dependency - if one column depends on another, and neither of those columns are primary keys, then it is transitive.

Let's keep using our games shop database as an example - here we have a `products` table:

product_id	product_name	developer_id	developer	price
1	Sitar Hero	103	StockRAR	5.99
2	Paint Drying Simulator	101	Gr9 Games	3.95
3	Extreme ChessBoxing 2021	102	MashButton Ltd	1.19

In this table, we see a transitive relationship between `product_id`, `developer_id`, and `developer_name`:

- `developer_name` relies on `developer_id` because it is unique
- `developer_id` relies on `product_id` because the latter is the primary key

Therefore, `developer_name` transitively relies on `product_id`.

We can fix this issue by moving the relationship between a developer and its id to a different table:

product_id	product_name	fk_developer_id	price
1	Sitar Hero	103	StockRAR
2	Paint Drying Simulator	101	Gr9 Games
3	Extreme ChessBoxing 2021	102	MashButton Ltd

developer_id	developer_name
101	Gr9 Games

developer_id	developer_name
102	MashButton Ltd
103	StockRAR

Having eliminated the transitive dependency, we now have two tables which conform to 3NF.

Further normalisation

There exist further ways to normalise data, such as Boyce-Codd Normal Form, but they are not covered here as 1) 3NF is generally sufficient for most cases, and 2) normalisation becomes increasingly more abstracted at higher levels.

Tutorial

There is no tutorial for this module.

Exercises

Try to refine your ERD for the games shop database to at least **2NF**.

(*note: if you have not created an ERD already, refer back to the Entity Relationship Diagrams module for context*)