Third Normal Form

Relational Database Design

Session Outline

- The definition of third normal form
- How we can apply it to our sample database

Third Normal Form

- What is third normal form?

Fulfils the requirements of second normal form
Has no transitive functional dependency

- What is a transitive functional dependency?

Transitive Functional Dependency

- It means that every **non-key attribute** must be dependent on the **primary key and the primary key only**
- Example:
  - Column A determines B
  - Column B determines C
  - Therefore, column A determines C
  - A > B > C
  - This is a transitive functional dependency
  - It should be removed
  - Column C should be in a separate table
- Let's apply it to our example

Our Example

- Student: Student ID, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Student_Subject: *student ID*, *subject ID*
- Category: category ID, category name
- Subject: subject ID, *category ID, university ID, teacher ID,* subject name
- Teacher: teacher ID, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- University: university ID, university name, unit number, street number, street name, suburb, city, state, code, country

Student Table

- Student: Student ID, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Are any of these fields dependent on something that isn't the primary key?
- Part of the address is (depending on your region)
- In this example, the code is used to determine the suburb, city, and state
- But not the country (a code could be the same thing in more than one country and represent two different things)
- It should be moved to another table

Student Table

- Student: Student ID, *code,* first name, last name, date of birth, unit number, street number, street name, country
- Code: code, suburb, city, state
- As the code was used to determine suburb, city, and state, it is a unique identifier and has been made the primary key
- It also means it is the foreign key in the Student table

Student Subject Table

- Student_Subject: *student ID*, *subject ID*
- This table is OK

Category Table

- Category: category ID, category name
- This table is also OK

Subject Table

- Subject: subject ID, *category ID, university ID, teacher ID,* subject name
- This table is also OK

Teacher Table

- Teacher: teacher ID, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Same situation as student
- Code can be split from the address
- But we already created a code table. Do we need another one?
- No, we can use the same table, as it refers to the same thing
- Teacher: teacher ID, *code,* first name, last name, date of birth, unit number, street number, street name, country

University Table

- University: university ID, university name, unit number, street number, street name, suburb, city, state, code, country
- Same as Teacher and Student
- We can split the code into another table, using the same table
- This means the university table has changed:
- University: university ID, *code*, university name, unit number, street number, street name, country

Our Example So Far

- Student: Student ID, *code,* first name, last name, date of birth, unit number, street number, street name, country
- Code: code, suburb, city, state
- Student_Subject: *student ID*, *subject ID*
- Category: category ID, category name
- Subject: subject ID, *category ID, university ID, teacher ID,* subject name
- Teacher: teacher ID, *code,* first name, last name, date of birth, unit number, street number, street name, country
- University: university ID, *code*, university name, unit number, street number, street name, country

MySQL Workbench

- Let’s see what this looks like in MySQL Workbench

Our Database

- Third normal form has been completed
- Our database is now normalised!
- It should:
  - meet all of the requirements we have
  - meet all conditions of the normal forms
  - be easy to insert, update, delete, and create records in
  - achieve all of the advantages of a relational database
- There are more design considerations to remember
- We'll learn about them in the next few lessons

Summary

- Third normal form is where a database:
  - Fulfils the requirements of second normal form
  - Has no transitive functional dependency
- Once a database is in third normal form, it is normalised
- It should then be more efficient and achieve all of the benefits of a relational database

Action

Apply third normal form to your database by:

1. Looking at each table
2. Determine if there are any transitive functional dependencies (any situations where A > B > C
3. Move these into a separate table

What's Next?

- Learn some ways we can improve the design of our database and things to consider when creating tables