Second Normal Form – Part 2

Relational Database Design

Session Outline

- Continue applying second normal form to our database

Subject

- We need to do the same thing for student name
- Subject: <u>subject ID</u>, *category ID,* subject name, student name
- Student name is not a way to **uniquely identify** the student that has enrolled in the subject
- Student ID is the right field
- But how do we know which way the relationship goes? Do we add student ID to the Subject table, or subject ID to the Student table?

Subject Table

- “Does a **subject** have many **students**, or does a **student** have many **subjects**?”
- In this example – it’s both
- This is a many to many relationship
- What do we do?
- Relational databases should not have a **many to many** relationship
- Create a **joining table** as mentioned earlier

Joining Tables

- Joining tables are created to support a many to many relationship
- Commonly named after the two tables they are joining
- Our example:
- Student: Student ID, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Subject: subject ID, *category ID*, subject name, student name
- Joining table would be:
- Student_Subject: student ID, subject ID
- Combination of students and subjects
- It is always unique
- Allows us to answer that “both” question

Student and Subject

- Our examples now are:
- Student: <u>Student ID</u>, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Subject: <u>subject ID</u>, *category ID,* subject name
- Student_Subject: *student ID*, *subject ID*

MySQL Workbench

- Let’s see how the joining table is represented in MySQL Workbench

Teacher Table

- Teacher: teacher ID, first name, last name, date of birth, subject taught, unit number, street number, street name, suburb, city, state, code, country
- "Each non-key attribute must be functionally dependent on the primary key"
- Does this satisfy this statement?
- Are all attributes specific to this teacher?
- All are OK except for "subject taught"

Teacher Table

- What does subject taught refer to? The name of the subject in the subject table
- However, it’s not the primary key in that table
- It doesn’t uniquely identify the record
- Same as the other tables we just looked at

Teacher Table

- Let’s confirm the relationship
- Does a **teacher** have many **subjects**, or does a **subject** have many **teachers**?
- First one – a teacher has many subjects
- Add the primary key of the first table (**teacher ID)** to the second table (**subject**)
- If we’re trying to represent the relationship, we should use a foreign key, based on the primary key in that table
- Remove “subject taught”

Teacher Table

- Teacher: <u>teacher ID</u>, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- Subject: <u>subject ID</u>, *category ID, teacher ID,* subject name

University Table

- University: <u>university ID</u>, university name, unit number, street number, street name, suburb, city, state, code, country
- All fields look OK
- How is the university table related to the others?
- Subjects are taught at a university

University Table

- How is this relationship defined?
- Does a **subject** have many **universities**, or does a **university** have many **subjects**?
- Sure, you could have the same subject across many universities
- But for our purposes, a subject is where students enrol in it and it is taught by one teacher
- Therefore, second statement is true – **university has many subjects**
- So, add the primary key of the **second table** (university, university ID) to the **first table** (subject)

Subject and University Table

- Subject: <u>subject ID</u>, *category ID, university ID, teacher ID,* subject name
- University: <u>university ID</u>, university name, unit number, street number, street name, suburb, city, state, code, country

Our Example So Far

- Student: <u>Student ID</u>, 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: <u>category ID</u>, category name
- Subject: <u>subject ID</u>, *category ID, university ID, teacher ID,* subject name
- Teacher: <u>teacher ID</u>, first name, last name, date of birth, unit number, street number, street name, suburb, city, state, code, country
- University: <u>university ID</u>, university name, unit number, street number, street name, suburb, city, state, code, country

MySQL Workbench

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

Summary

- Second normal form is where:
  - The database fulfils the requirements of first normal form
  - Each non-key attribute must be functionally dependent on the primary key
- A foreign key is a primary key from one table added into another table, to link the records together

Action

Apply second normal form to your database by:

1. Look at each of your tables
2. See if there are any attributes that don’t rely on the primary key, and if not, move them to a new table
3. Determine how these tables are related
4. Add foreign keys to these tables

What's Next?

- The definition of third normal form and how to apply it to our database