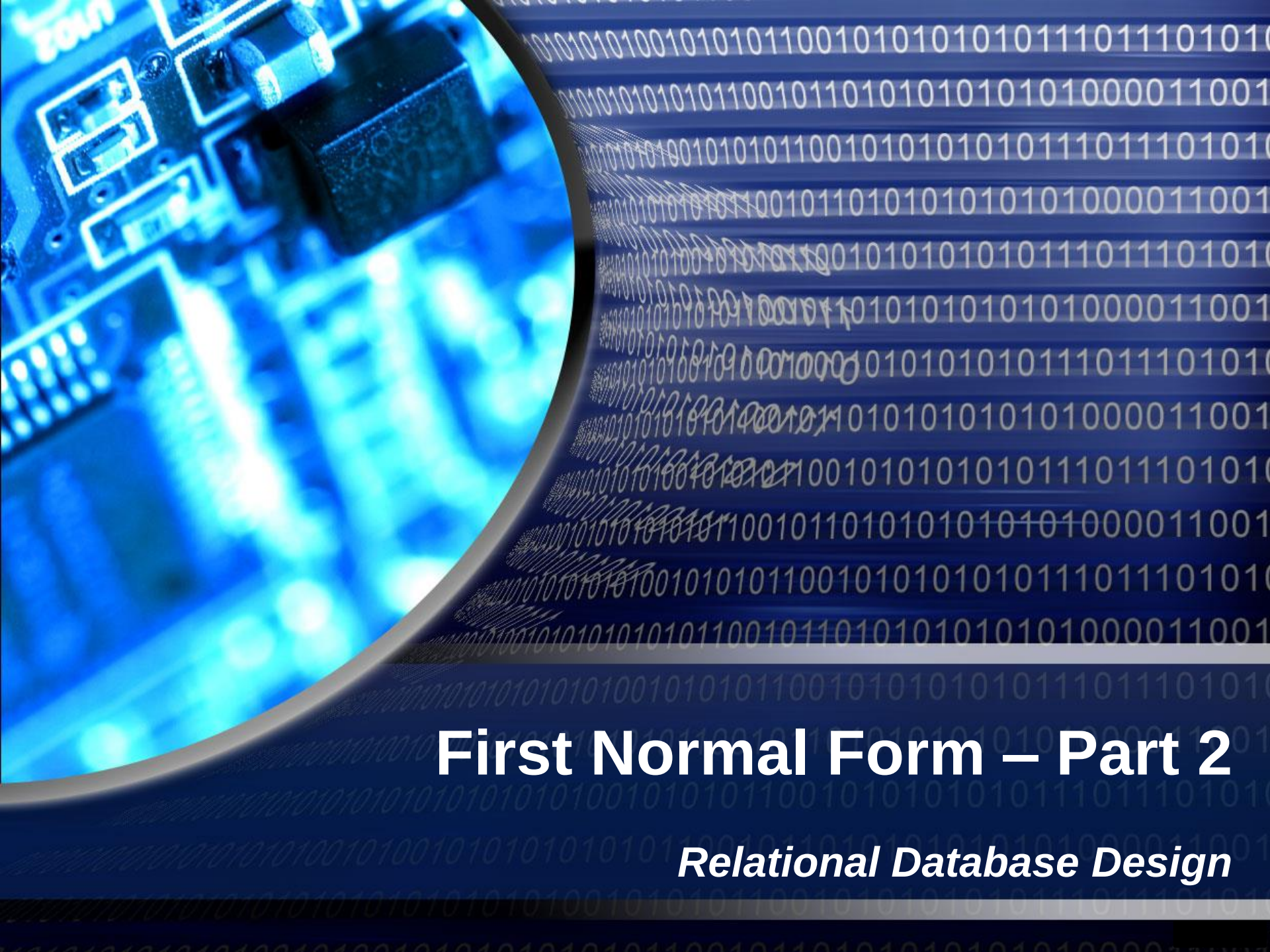

First Normal Form – Part 2

Relational Database Design

Session Outline

- Continue applying first normal form to the rest of our tables

Subject Table

- Subject: subject name, subject category, student name
- Does this combination uniquely identify the row *every time*?
- No
  - The same subject name and category could have two students with the same name
- Why is student name there?
  - To identify the student in the subject
  - But it isn't unique!
  - Let's come back to this

Subject Table

- Subject: subject name, subject category, student name
- What can be used to uniquely identify a row?
- Subject name? Maybe
  - What if the subject name changes
  - It's not good to have a primary key for something that may change
  - E.g. "Introduction to Biology" may change to "Biology 101" which could mess up the database
- Category?
  - No, many subjects can be in the one category
- Student name?
  - No

Subject Table

- Let’s create a new field for this table as well
- Keep it consistent with the other table
- **Subject ID**

Subject Table Updated

- Our new Subject table:
- Subject: subject ID, subject name, subject category, student name

MySQL Workbench

- Let’s update the **subject** table in our MySQL Workbench file

Teacher Table

- Teacher: first name, last name, date of birth, address, subject taught
- Does this combination uniquely identify the row *every time*?
- No
  - Two teachers with the same name, date of birth, and address, could teach the same subject
  - Very unlikely! But also possible
- Remember, we should be allowing for exceptions, no matter how rare they are
- We don't want the system to break because of a bad database
- Similar situation to the student table

Teacher Table

- Teacher: first name, last name, date of birth, address, subject taught
- What can be used to uniquely identify a row?
- No combination of these fields
- We need to create a new field again
- Let’s call it **Teacher ID**
- Also, let’s split up the Address field like we did with student: unit number, street number, street name, suburb, city, state, code, country

Teacher Table Updated

- Our new 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

MySQL Workbench

- Let’s update the **teacher** table in our MySQL Workbench file

University Table

- University: university name, university address
- Does this combination uniquely identify the row *every time*?
- Yes
  - Even if there are two universities with the same name, they won't be at the same address

University Table

- University: university name, university address
- What can be used to uniquely identify a row?
- University name and address

University Table

- We’ve added new columns to all other tables for the primary key
- This is because the columns did not uniquely identify a row
- Do we need a new column for the university table?

University Table

- In this case, we don't need one, but it comes down to personal opinion
- We also don't want to have a primary key on a value or combination of values that might change
- What if the university name changes? Or the address?
- **I usually create a new field**
- Remember to be consistent with the name
- Let's create one here – **university ID**
- Even though we are creating a database for one university, using this table means it can be used by many

University Table

- Also, we should split up the address: unit number, street number, street name, suburb, city, state, code, country

University Table Updated

- Our new University table:
- University: university ID, university name, unit number, street number, street name, suburb, city, state, code, country

MySQL Workbench

- Let’s update the **university** table in our MySQL Workbench file

Summary

- First normal form is the first stage of the normalisation process
- It means that each set of columns must uniquely identify a row
- We've applied these rules to our sample database

Action

1. For each of your tables, determine if any combination of the columns can be used to uniquely identify a record (including any exceptions, even if they are rare)
2. If so, determine what column or columns can be used as the primary key?
3. If not, then create a new column for the primary key (and make it consistent for each table)

What's Next?

- Relationships between your tables