Let's Create a Schema Walkthrough

Follow along as three different database schemas are designed, described by an ER Diagram, and implemented as SQL databases. This document will give a basic walkthrough of the process. There will be additional handouts for the final ER Diagrams and SQL Database Creation Scripts for the three examples. In our next classes, we will be using these databases to write SQL queries against.

Choose a Topic

Let's think of a topic with a lot of data, that has relationships we can discuss

- DnD Character Creation
- Meal Planning
- Family schedule planning

What Relations are Needed?

- A relation is like an object, it should be a distinct thing that can be described.
    - For DnD Character Creation, we could have relations such as:
        - Class
        - Spell
        - Weapon
    - For Meal Planning, we could have relations such as:
        - Ingredient
        - Recipe
    - For a Family Schedule, we could have relations such as:
        - Family Member
        - Chore
        - Location

Do all Relations have a unique name?

This one seems obvious, but if the relations don't have unique names that clearly describe their exact purpose, then the Databases Schema is going to get confusing, and ultimately, MySql is going to force us to have unique names.

What Attributes Does Each Relation Need?

- Brainstorming what attributes are needed is an important design step. Sometimes, as we brainstorm attributes, it will become clear we need more relations. Or fewer.
- We should give each attribute within a relation a distinct name, but sometimes it helps to be clear what is needed, as we may end up with more relations to accommodate.
    - Note that its is perfectly fine for different relations to have attributes of the same name
    - For the DnD Character Creation relations:
        - Class
            1. Name
            2. Alignment
            3. Type
        - Spell
            1. Name
            2. Cost
            3. Effect
            4. Duration
            5. Effective Distance
        - Weapon
            1. Name
            2. Cost
            3. Deals X Damage
            4. Weight
            5. How is it wielded?
    - For Meal Planning relations
        - Ingredient
            1. Name
            2. Size
            3. Type
        - Recipe
            1. Name
            2. Serving Size
            3. List of Ingredients
            4. Cook Time
            5. Difficulty Rating
    - For Family Planning relations:
        - Family Member
            1. First Name
            2. Middle Name
            3. Last Name
            4. Birthday
        - Chore
            1. Name

2. Difficultly
3. Duration
4. Tools Needed
5. Frequency

Let's add some tuples

- Adding data to your relations is a great way to ensure you are actually storing the data you intend to store.
    - In the DnD Character Creation database schema,
        - A spell's duration should be a unit of Time, so we should make sure that our attribute is of type Time
        - A Spell costs mana and a weapon costs coin, do we want to use the name "Cost" in both scenarios? The same type?
    - In the Recipe database Schema
        - Cook Time is definitely a unit of time
        - A recipe calls for a list of ingredients, but each attribute in a relation stores a distinct value. We cannot store a list of ingredients
            - This is a tricky fix, and something we will revisit in a future example
                - Suffice to say, our relation cannot represent the data we need to store, and we need to add additional relations.
    - In the Family Planning database schema:
        - Do we have enough relations to know if a family member can complete a chore?
            - A child cannot drive
            - A parent cannot attend soccer practice
            - How can we add more relations to better capture these restrictions?

What are the domains of each attribute?

- Now that we have all our relations, or are at least pretty sure we've captured everything that is needed, let's identify the domain of each attribute.
    - We want to decide this before we think about what type to assign, because we are designing, and we want to make sure we think about the data itself before trying to force it into a nice little defined box
- Are our domains atomic? Do we need to massage our existing attributes or domains to make sure this is true?
- Can any of our attributes be null?
    - Think, can this attribute have an unknown value?
    - Can the attribute's value simply not exist?
    - How would we handle these two unique cases?

What are the relation schemas for our relations

- RelationName(Attribute1Name Attribute1Domain, Attribute2Name Attribute2Domain, etc…)
- What are the super keys for each relation?
  - What values of the tuple are needed to uniquely identify the tuple?
  - What are the redundant super keys, as remember those are potential options for a primary key that we might find out later we need, based on our data or requirements!
  - What are the candidate keys? Remember these are the minimal super keys, meaning the super keys containing the fewest attributes, and there could be more than one!
  - What is the primary key? It should be the candidate key we think is going to best suit our needs, and remember this could change as we design and develop, which is why we have gone through this process to narrow down our super keys.
- Let's underline our primary keys in our relation schema definitions

What are the foreign key constraints for our relations?

- In the referencing table, we are looking at our primary keys, and then drawing an arrow to the attribute we are referencing. We need to ensure referential integrity constraints are understood and applied
- We do not have to use the primary key in the referencing, but most languages are going to require this.