Database design

The purpose of this exercise is to practice designing and implementing a database from scratch based on an initial set of requirements.

Learning objectives

After completing this exercise, you'll understand:

- How to create a normalized database based on a set of requirements.
- The differences between One-to-Many and Many-to-Many relationships and how to define them.
- How to implement associative entities to appropriately resolve Many-to-Many relationships.
- How to use Data Definition Language (DDL) to create tables and relationships.

Evaluation criteria and functional requirements

- All of the DDL and DML statements run as expected.
- All tables and relationships necessary to satisfy the requirements are defined.
- Design is normalized appropriately.
- The correct data types are used.
- Necessary constraints have been defined.
- Code is clean, concise, and readable.
- Your database script runs correctly, and your database is created as a result of running the script.
- The correct number of rows are represented in each of the tables based on the requirements.

Implement a database to represent Meetups

The Meetups database keeps track of members and events for various interest groups.

Each Member has the following attributes:

- Member Number
- Last Name
- First Name
- Email Address
- Phone Number (some members may not provide one)
- Date of Birth
- Flag indicating if they want reminder emails

Each Interest Group has the following attributes:

- Group Number
- Name (no two groups can have the same name)
- Has zero-to-many Members

Each Event has the following attributes:

- Event Number
- Name

- Description
- Start Date and Time
- Duration (minimum of 30 minutes)
- Group running this event
- Has zero-to-many Members

Requirements

1. All tables must have a primary key.
2. Populate the tables with data for at least four events, three groups, and eight members.
3. Make sure each event has at least one member assigned to it, and each group has at least two members.

Design tips

Think about these questions as you work through the exercise:

- Is a natural key available for the primary key, or will you need a surrogate?
- What data type should you use for each column?
- Which attributes are required? Which are optional?
- Are there additional constraints such as length, data format, or restricted values for any of the columns?
- What table columns will the user likely search?

Don't forget to normalize.

Getting started

1. Get a pen and paper or a whiteboard and a whiteboard marker.
2. Write down the list of entities that must be represented in the database.
3. Draw lines between the entities to represent the relationships between them.
4. Create a file called `meetups.sql`.
5. Inside of the file, take the database you designed in the first two steps, and begin adding the DDL statements to implement the design.
6. Add DML statements to populate the tables with the required data.
7. Create your database by running the script you created.

Tips and tricks

- While you could start by writing your DDL statements directly, it's typically better to create a diagram first. Database diagrams allow you to visualize relationships between tables, and typically, using a whiteboard or pen and paper is cheaper and more efficient than writing code first.
- There are many varying levels of normalization a database design can achieve. How might you know if your database is normalized enough?
- Remember: Many-to-Many relationships require an additional entity, often called an associative entity, to correctly define the relationship. If you didn't do this, which table needs to carry the foreign key reference?
- How do you know which table needs to hold the reference to the other table? The "crow's foot" is often a good indication as to which table the reference column belongs in.

    - For instance, consider an artist and a song. For simplicity's sake, assume that a song belongs to one artist. That said, you might say that "A Song must belong to one Artist, and an Artist may have many songs." The talons of the crow's foot would point at the Song table.
    - This means that a column should be created on the Song table that would allow you to specify the Artist the Song belongs to.
- In the previous example, how might the design be impacted if you decide to allow a Song to belong to multiple Artists?
