SceneSense: A database for theatrical production companies

Conceptual and Logical Modeling

Prof. Rachlin

Friends, Romans, countrymen, lend me your ears!

The Huntington Theater Company in Boston Massachusetts needs a database to manage their theatrical productions. This season, the theater plans on putting on a production of Shakespeare's *Julius Caesar* as well as a production of Tom Stoppard's tragicomedy *Rosencrantz and Guildenstern are Dead*. The theater director wants to keep track of a rehearsal schedule so that she can generate (via a SQL query) a *call list* identifying which actors need to show up for rehearsal on which dates. You have been hired as a co-op student to design and build their database. Your first step is to develop a conceptual and logical model. You will also build the table definitions using CREATE TABLE statements and then populate your tables with appropriate INSERT statements.

Design Constraints

1. You don't have to model multiple production companies - your database will be for **one** production company only, namely the Huntington Theater Company. But you do need to be able to keep track of the various productions that the HTC will put on both this season and in years to come.

2. Each production is for a particular play. A play might be produced many times or not at all. A production includes a name, description, and a billboard image (used for marketing purposes) and the premier date. For example, the play might be Shakespeare's Julius Caesar, but the *production* could be: "Caesar! The family musical!" based on the original play.

3. Actors work for the production company and have a name, contact email, and optional cell-phone number.

4. Each play is an *ordered* sequence of scenes, and a set of character roles. Plays have a title and author. Scenes have a title and a sequence number (1,2,3…). You do not need to model the Shakespearian distinction between Acts and Scenes. For example, one scene may simply be called: "Act 3 Scene 1" (which might have a sequence number of 13), followed by "Act 3 Scene 2" (which would have a sequence number of 14). The sequence number is not the same as the scene_id primary key. (Sequence numbers are not unique – they could repeat from play to play.)

5. Characters refer to characters in the play. They have a name and optional description. Characters appear in one or more scenes. In each production, each actor that works for the company portrays zero or one character in a given play. An actor could portray a different character in a different production of that same play or be assigned no part whatsoever. The key thing to note is that an acting role defines the actor, the production, and the character being portrayed.

6. Each production has a set of rehearsals with a date and a start and end time. The rehearsal schedule identifies which scenes are specifically rehearsed during each rehearsal period and for how many minutes. (Multiple scenes might be rehearsed in one rehearsal period.) You need not be concerned about the order in which the scenes are rehearsed within a given rehearsal. That is up to the discretion of the director. Your database should eventually enable the generation of call lists that identify which actors need to show up for which rehearsals, but call lists themselves are not stored in the database.

Phase I: Database Design

Create a diagram of your database using the MySQL modeling tool. Note that the modeling tool incorporates some aspects of both Conceptual and Logical modeling by converting many-to-many relationships to multiple many-to-one relationships and by having you define both the attributes, their data types, and the foreign key constraints. Depict all cardinality and participation constraints. Assign appropriate attribute names and data types. Specify all constraints including foreign key constraints where appropriate. Label all relationships.

Phase II: Build the tables and populate with data

Use CREATE TABLE statements to manually define each table in your database. Do NOT use the MySQL forward-engineering or reverse-engineering tool. The CREATE TABLE statements that it creates are unnecessarily verbose. Once all your tables are created, populate your database with the information below. (For this exercise, we are going to model only some of the characters and scenes in Julius Caesar. In addition, you only need to model *three* rehearsals.

1. Store two productions in a **production** table: "Julius Caesar the Musical" (based on the play Julius Caesar) and "*Rosencrantz and Guildenstern are Dead"* (based on the play by the same name). Store their corresponding plays in a **play** table. We will model the Julius Caesar production in greater detail. You do not need to store the billboard image.

2. For Julius Caesar, model the following five characters only:
    a. Caesar (played by Peter O'Toole)
    b. Brutus (played by Will Smith)
    c. Cassius (played by Brad Pitt)
    d. Antony (played by Russell Crowe)
    e. Portia (played by Angelina Jolie)

    Also insert a sixth actor, Scarlett Johansson, who will not be appearing in this production and has been assigned no part. (She must be doing a movie!)

3. Model the following two scenes only:
    a. Act 3 Scene 1: (Caesar, Brutus, Cassius, Antony)
    b. Act 3 Scene 2: (Brutus, Cassius, Antony)

4. Create a rehearsal schedule with three rehearsals for Act 3, each four hours long. Rehearsals are 2pm to 6pm each day. Don't be late!

    a. Rehearsal 1: March 15th, 2021: Scene 1 will be rehearsed for two hours and Scene 2 will be rehearsed for two hours.

    b. Rehearsal 2: March 16th, 2021: Scene 2 will be rehearsed for all four hours.

    c. Rehearsal 3: March 17th, 2021: Scene 1 will be rehearsed for one hour and Scene 2 will be rehearsed for three hours.

What to submit:

1. Your MySQL model as a native MySQL model (scenesense_yourname.mwb) along with a .pdf image (scenesense_yourname.pdf) of your model diagram. Be sure that the column attributes and datatypes are clearly shown.

2. A .SQL script (scenesense_yourname.sql) with all CREATE TABLE and INSERT statements. The TAs should be able to execute your script and run various test queries to validate the contents of each table.