CSCE 156 – Computer Science II

Lab 06 - SQL+JDBC - Worksheet

Sarah Roscoe[*]
Summer 2019

Names ____________________

For each question, write an SQL query to get the specified result. You are highly encouraged to use a GUI SQL tool such as MySQL Workbench and keep track of your queries in an SQL script so that lab instructors can verify your work. If you do, write your queries in the script file provided rather than hand-writing your queries here.

1 Pre Lab Activity - Querying a Database

1.1 Simple Queries

1. List all albums in the database.
2. List all albums in the database from newest to oldest.
3. List all bands in the database that begin with “The”.
4. List all songs in the database in alphabetic order.
5. Write a query that gives just the `albumId` of the album “Nevermind”.

1.2 Simple Aggregate Queries

6. Write a query to determine how many musicians are in the database.
7. Write a (nested) query to list the old(est) albums in the database.

[*]Developed by Dr. Chris Bourke. Revamped by Yi Xia

8. Write a query to find the total running time (in seconds) of all tracks on the album *Rain Dogs* by Tom Waits

1.3 Join Queries

9. Write a query list all albums in the database along with the album's band, but only include the album title, year and band name.
10. Write a query that lists all albums and all tracks on the albums for the band Nirvana.
11. Write a query that list all bands along with all their albums in the database *even if they do not have any.*

1.4 Grouped Join Queries

12. Write a query list all bands along with a *count* of how many albums they have in the database (as you saw in the previous query, some should have zero).
13. Write a query that lists all albums in the database along with the number of tracks on them.
14. Write the same query, but limit it to albums which have 12 or more tracks on them.
15. Write a query to find all musicians that are not in any bands.
16. Write a query to find all musicians that are in more than one band.

2 Pre Lab Activity – Database Design

Design Entity Relations (ER)

In this section you will design the ER by completing the diagram in Figure 1 as instructed in the following steps. All 6 information items in List 1 of the handout should be supported by the resulting design.

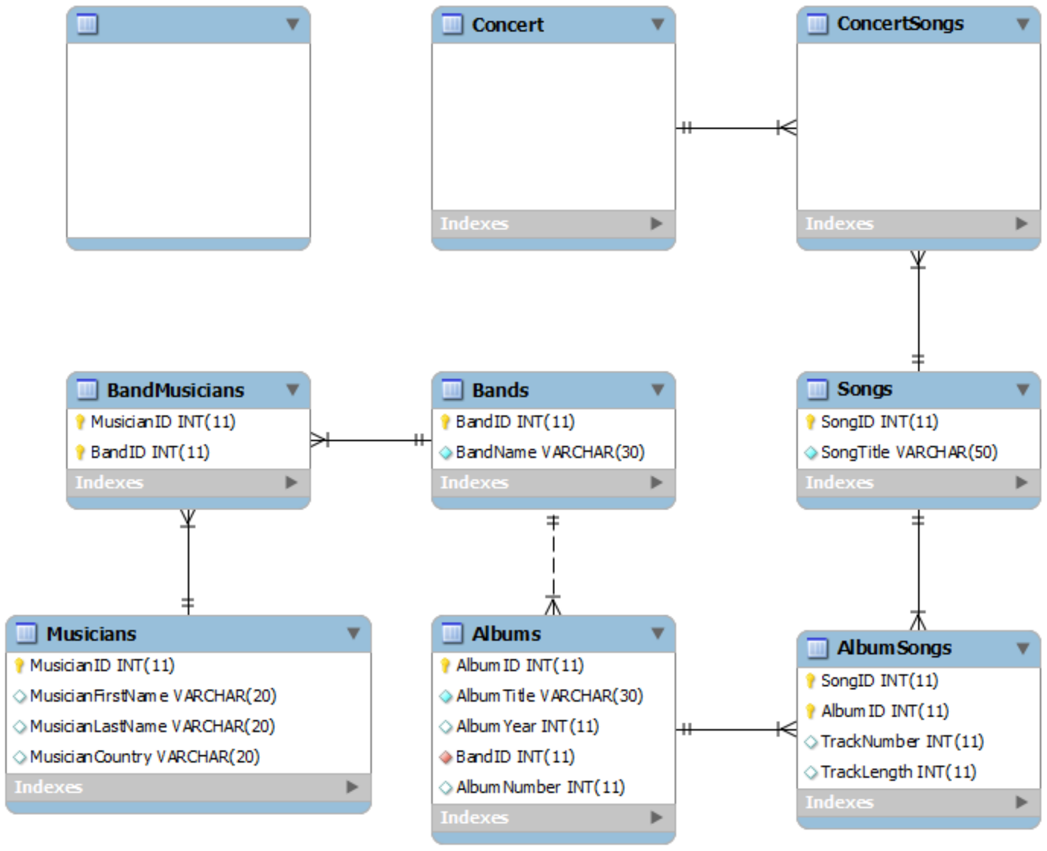

Figure 1 An incomplete ER diagram for the new database being designed.

1. Use the tables from the original *Albums* database as an example to list the fields and field types associated with each of the *Concerts* and *ConcertSongs* entities in Figure 1. If your design needs an additional entity then use the blank item.
   Note: Pay attention to how the primary/foreign keys must be designed to support the relations between *Concert*, *Concert Songs*, and *Songs*; use *BandMusicians* and *AlbumSongs* as examples to complete *ConcertSongs*.
2. How does *Bands* from the original Albums database relate to the *Concert* and/or *ConcertSongs* entities? Indicate the relation(s) by drawing the appropriate line(s) onto Figure 1 and make sure any primary/foreign keys are updated to reflect this/these relation(s).
3. Every concert takes place at a concert hall; it is possible that the name of a concert hall or its seating capacity changes. In such an event your current design should not require that any previously stored concerts be updated to reflect the modifications to the concert halls; if so add an entity to your design to solve the problem by filling the blank item in Figure 1.

4. Specify the relations between the entity you described in the blank item and the other entities in Figure 1. Make sure the primary/foreign keys of the entities match their relations.

Show your design to a lab instructor and get it signed off by a lab instructor before proceeding.

Signature____________________________________

Create the New Database

Write and run a new SQL script (or simply just modify `albums.sql`) to generate the new tables you designed in Activity 1 using the *CREATE TABLE* statement (alternatively, you may write a new script that *modifies* the existing database). Use Figure 1 as a blueprint for your script making sure the following items are satisfied:

- **Naming:** Use a uniform naming conventions for the tables and their fields
- **Field Types:** Make sure to use appropriate types and a uniform typing conventions for each field
- **Primary keys:** Make sure to specify which fields are primary keys
- **Foreign keys:** Enforce appropriate foreign key restrictions to reflect each relation.
- **Note:** Some of the new relations may require you to use the *ALTER TABLE* statement since you may need to modify the table from the original Albums database design to enforce the constraints.
- **Null/Default values:** Some entity fields are so essential that if they are not provided a value the entity itself is unable to server its purpose (i.e. a primary key should never be *nullable*). Make sure all fields are able to store valid values.

Write SQL statements to do the following and demonstrate your working script to a lab instructor to have them sign off on this lab.

1. Write a statement to create the table ConcertSongs
2. Write a statement to create the table Concert
3. Write statement(s) to create any other tables/entities that your design requires
4. Write statement(s) to alter the original tables that you made if needed.

Signature____________________________________

Use the new Database

You will now make sure that your design makes sense by writing several queries to insert and query data out of it.

1. Write queries to insert *at least* two Concert records.
2. Write queries to associate at least 3 songs with each of the two concerts
3. Write a select-join query to retrieve these new results and produce a playlist for each concert
4. Modify the query to include the *name of the band* playing the concert. If such a query is not possible, explain why and sketch an alternative design in which it would be possible.

Signature____________________________________

3 Lab Activity – JDBC in a Web Application

1. Demonstrate your working `main` method in the `Test` class before you deploy to the server.

____Lab Instructor Signature

2. Think about how you made your JDBC queries and processed the results. What would happen if there were no results?

3. What values in the database are allowed to be null? What potential problems does that pose to the end user?

4. What other values in the database could pose a problem with your code? How (or where) did/should you handle these contingencies? At the database level? During a query? In code? Which solution is ideal?

5. Demonstrate your working application to a lab instructor and have them sign this worksheet.

____Lab Instructor Signature