CSCE 156 – Computer Science II

Lab 8.0 - SQL II

Prior to Lab

1. Review this laboratory handout prior to lab.
2. Review the following materials:
   - Creating Tables:
     http://www.w3schools.com/sql/sql_create_table.asp
     http://www.w3schools.com/sql/sql_autoincrement.asp
   - Types:
     http://www.w3schools.com/sql/sql_datatypes.asp
   - Primary Keys:
     http://www.w3schools.com/sql/sql_primarykey.asp
   - Foreign Keys:
     http://www.w3schools.com/sql/sql_foreignkey.asp
   - Altering Tables:
     http://www.w3schools.com/sql/sql_alter.asp
   - Constraints:
     http://www.w3schools.com/sql/sql_notnull.asp
     http://www.w3schools.com/sql/sql_unique.asp
     http://www.w3schools.com/sql/sql_default.asp

Lab Objectives & Topics

Following the lab, you should be able to:

- Gain experience with simple database design strategy

- Understand the motivation for entity relations and concepts
- Learn how to create a database from your design

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

Getting Started

Material for this lab is available for download on the course website from the previous lab.

Manipulating Data

In this lab we will continue to work with the Album database. For reference, the database schema is presented in Figure 1 as an Entity-Relation diagram.

Data can be added to a database using `insert` statements and existing data in a database can be manipulated using `update` and `delete` queries. However, records in a table that may be referenced by records in other tables via a foreign key cannot be deleted unless the referencing records are handled first. Likewise, data that requires a reference to data in other tables cannot be inserted prior to inserting the referenced records.

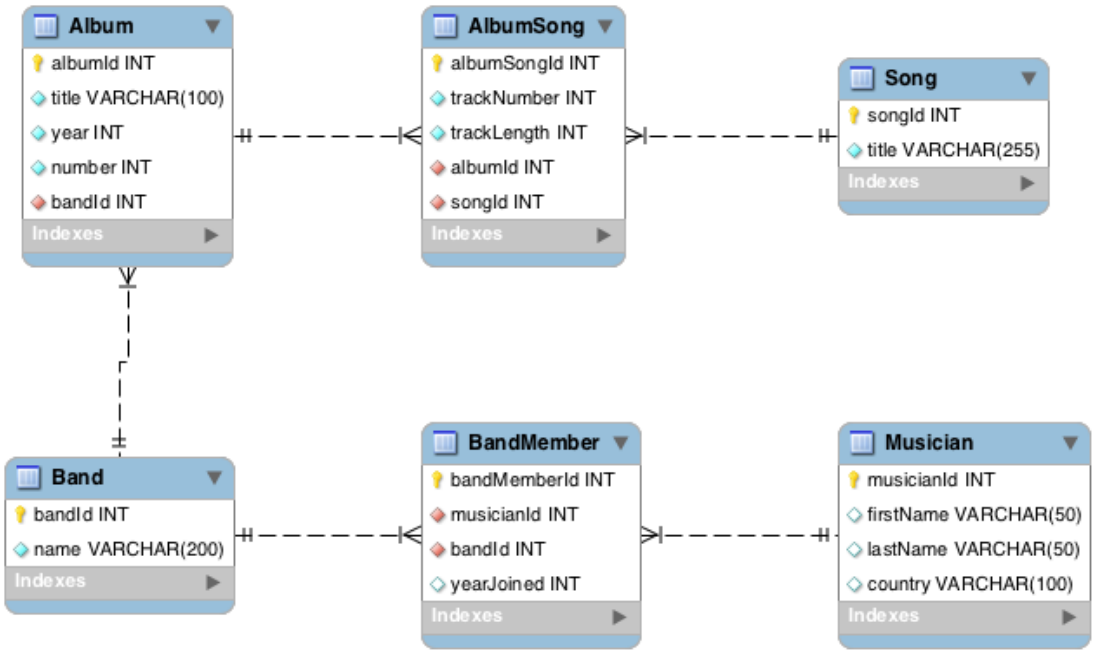

Figure 1: Albums Database

Complete the first section of your worksheet.

Altering a Database

A careful examination of the DDL file provided (`albums.sql`) indicates how these tables were built and related to each other. New requirements may mean that the underlying data model must be modified to support new pieces of data. For example, if we wanted to keep track of the emails of each Musician we could modify the Musicians table to include an email address. SQL allows us to *alter* existing tables using the following syntax.

```
alter table Musician add emailAddress varchar(50);
```

A better solution would add support for multiple emails in which case we would need to add an entirely new table.

```
1  create table Email (
2    emailID int not null primary key auto_increment
3    musicianId int not null,
4    address varchar(100) not null,
5    foreign key `fk_email_to_musician` (musicianId)
6      references Musician(musicianId)
```

```
7 );
```

In this lab, you will build on the Albums database to add support for modeling venues (concert halls) at which bands are under contract to play select album songs. You will add tables and keys to this database to support this functionality.

Developing a Solution

Design entities and relation(s) to extend the Albums database such that it supports the following concert information:

- The band playing at the concert
- The band's select album songs played at the concert
- The date the concert was held (use an appropriately formatted `VARCHAR`, date time types are not cross-compatible)
- The name of the hall where the concert was held
- The number of seats in the concert hall
- The number of concert tickets sold

The entities and relations illustrated in Figure 2 serve as a basis for the solution. You will add fields and another entity by following the first section of the worksheet.

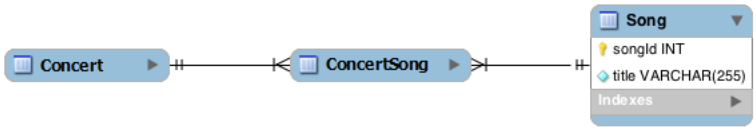

Figure 2: The Song table is part of the original database, relations between the concert and concert songs are suggested.

The `Song` table is part of the original Albums database. Relationships between two entities are indicated by a line between the two entities and in general are either a one-to-one relationship or a one-to-many relationship. Figure 3 shows the standard way to draw the two types of relations.

Complete the first section of the worksheet.

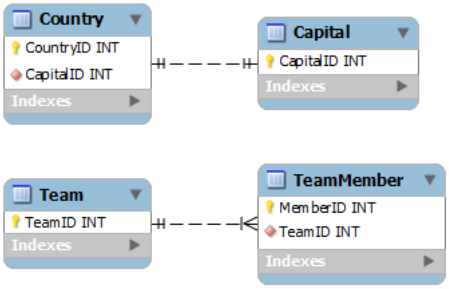

Figure 3: The line connecting the Country entity and the Capital entity describes a one-to-one relation. The line connecting the Team entity with the TeamMember entity describes a one-to-many relation.

SQL Script Modification

Now that you have properly designed the database modifications, you will realize them by writing SQL scripts (or modifying the original DDL file, `albums.sql`). Follow the directions in the worksheet in order to add the entities and relations you have designed in the previous activity to the tables and relations in the original Albums database.

Advanced Activity (Optional)

Consider the venues listed in Table 1.

Venue Name	Capacity
The Mega Dome	12,000
The Gorge	32,000
Hotel Concert Hall	5,000
Cruise Concert Hall	2,000

Table 1: The concert halls

Say that one concert was held at each of the concert halls according to the following rules listed bellow. Write an SQL script to insert data into the newly designed tables using the rules bellow.

1. In descending order of concert hall capacity, bands are assigned a concert hall in descending order of the number of their album songs and ascending order of their band name. That means the band with the most number of album songs and the smallest (lexicographically) band name gets the highest capacity concert hall and the band with the second highest number of album songs get the second highest capacity concert hall.

2. The songs played at each concert are those which have at least one album song of 5 minutes or longer.
3. Each concert is sold out.