**CSE 3241 Project Checkpoint 03**

**Functional Dependencies and Normalization**

Names Date

In a **NEATLY TYPED** document, provide the following:

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. **If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models.** You must highlight and indicate the corrections/modifications.
2. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer’s credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).
3. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.
4. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.
5. The purpose of this part of the checkpoint is to refresh Java programming. You will re-use this program in a future checkpoint. Write a Java program (no database are needed at this point). The program will present the library user with a list of options provided below to choose from, and then each option will ask the user for the input needed. You will need to store that data into any Java data structure that you consider more appropriate (classes, ArrayList, Sets, ..) and then be able to retrieve it as an output when other option requires it:

The Java app will present a staff user with a main menu to select one of the following options:

* 1. **Search**
  2. **Add new records**
  3. **Order items**
  4. **Edit records**
  5. **Useful reports** (not to be implemented yet)
* Each option in the main menu will provide the following options:
  1. **Search**
     1. Artist
     2. Tracks
  2. **Add new records**
     1. Add an Artist
     2. Add an Track
  3. **Order items**
     1. Order a Movie
     2. Activate item received (not to be implemented yet)
  4. **Edit records**
     1. Edit an Artist
  5. **Useful reports** (show options but not to be implemented yet)
     1. Tracks by ARTIST released before YEAR
     2. Number of albums checked out by a single patron
     3. Most popular actor in the database
     4. Most listened to artist in the database
     5. Patron who has checked out the most videos

Notes about the options:

**Search:**

* 1a. The user provides an artist, the program retrieves the information available
* 1b. The user provides a track name to search information about it.

**Add new records:**

* 2a and 2b. The user provides all the info needed to enter a new artist or a new Song (track). Use the attributes that you defined in your relational schema. Store it in your program (define an artist and track class).

**Order items:**

* 3a. The user enter the information to order a new movie, with number of copies purchase, price and an estimated date of arrival. Store it in your program (define an item or movie class).

**Edit records**

* 4a. The staff user selects an artist (provide the name), edit any field of the artist and then save it, updating your internal structure that storage the artist. Display in the screen.

Notes:

* All the options indicated must be implemented and tested.
* The program must compile and execute.
* The test output for each option must be included
* The program must run on eclipse so we can test it
* You don’t need user graphical interface. Input and output could be from the console
* DO NOT create databases at this time, this will be done on another checkpoint. Use classes and ArrayLists if needed.

Zip the Java project and include it in the submission.

1. Each team member, individually, needs to fill out the Peer-evaluation form provided and submit it to Carmen.

Please DO NOT zip the report file when you submit so that the grader can give you detailed feedback in Carmen.

**Note**: Your submission should include your ER-diagram, the relation schema before this checkpoint (before normalizing), the relational schema after this checkpoint (after normalizing) as well as the Java project. Give them clear file name that grader can distinguish 2 versions. Please DO NOT zip your files when submitting, so that the grader can give detailed feedback on Carmen.