CS 327E Lab 1: Relational Database Design

Learning Objectives:

1. Design a database for IMDB
2. Create a data dictionary with mappings to the source data
3. Collaborate with your partner and learn to work as one team
4. Get some exposure to Lucidchart and Github
5. Create an AWS account with an IAM user

Prerequisites:

1. Lab partner
2. Personal Github account
3. Private repo for your team
4. Lucidchart account with education upgrade [4]
5. Git client on your machine [4]

Steps Outlined:

1. Familiarize yourself with imdb.com if you're not already. Run some basic searches and get a feel for the type of data that's available from the web site.

2. Download the IMDB dataset in csv format from [1] and [2]. Note that [2] is just a trimmed down version of [1] and is being provided to allow opening the files in Excel. The full dataset is 1.1G uncompressed and the small dataset is 248M uncompressed.

3. Explore the dataset in Excel or using your favorite editor. Pay attention to both the content and the structure of the data. You will see that the first line of each file contains the column headings. Try to infer the relationships between the files based on the columns headings and cell contents. Feel free to run some searches on the IMDB.com to check your understanding of the data. You can also look at the raw IMDB dataset [3] from which the csv files were derived. Note that the raw dataset is a superset of the csv dataset that we are working with and it contains many more files.

4. Think of 12-15 different **queries** that are interesting to you and you'd like to be able to answer once you build the database. For example, how many actors are in the dataset? What time period does the dataset cover? Who are the actors that have been in the most movies? Make sure that these queries can be answered from the dataset we are working with. Write your queries in plain English (no SQL yet!) and name this file queries.txt. We will return to these queries in lab 3 and actually translate them into SQL.

5. Draw a **conceptual diagram** for your IMDB database based on your queries of interest and the actual data that's available. The diagram should represent all the entities and relationships between them, but it does not need to contain every attribute from the dataset if you are not using it. For example, if you have no queries that make use of the billing_position field in the cast.csv file, then you do not need to include it in your diagram. Use Lucidchart to create the conceptual diagram. Refer to the SXSW diagram

example we worked on in class if that's helpful [7]. Download the diagram from Lucid as a pdf file and name it conceptual_diagram.pdf.

6. Draw a **physical diagram** from your conceptual diagram. Recall that the physical diagram provides a more granular view of the database. It should specify the primary key for each table and datatypes for all the fields. It should also specify the foreign keys between entities as well as resolve all many-to-many relationships through the use of junction tables. Again, refer to the SXSW example diagram if that's helpful. Use Lucidchart to create this diagram. Download the diagram as a pdf file and name it physical_diagram.pdf.

7. Create a **data dictionary** that describes the layout of each table in the database. For each table, provide the following details:

- Column names
- Column data type
- Primary key column(s)
- Foreign key column(s) if applicable
- Columns with not null and unique constraints if applicable
- Source filename(s) (e.g. movies.csv, etc.)
- Source column heading(s) (e.g. title, year, location, etc.)
- Explanations for any columns that are not immediately obvious (e.g. "year" refers to the release year of the movie, etc.)

The data dictionary should be done in Excel with a separate sheet for each table. Name the file data_dictionary.xlsx.

8. Create a new folder in your local git repository called **lab1**. Add your conceptual schema, physical schema and data dictionary to this folder and commit the files to your local repo. Push the commit to your team's remote private repo on Github.

9. Create an **AWS account** by following the steps in the setup guide [4]. Each team member should have his/her own AWS account. If you have already done this, skip to the next step.

10. Create an **IAM user** with administrative access by following the steps in the setup guide [4]. This IAM user will be used by the TAs for grading future assignments. You can also choose to use the same IAM user to log into the AWS console (instead of using the root credentials which is not recommended). Note that since you and your partner will be receiving the same grade on all labs and final project, you only need to share with us an IAM user from the account that you'll be using for submissions. Take a screenshot of the IAM dashboard as proof that you completed this task and commit it to your repo.

11. Create a **Stache** entry with the IAM credentials by following the steps in the setup guide [4]. Remember that the name for the Stache entry should be your actual team name. Add the professor and both TAs as readers of your Stache entry. Our EIDs are cohens5, am65355, and syw279.

12. Locate the **commit id** that you will be using for your submission. This is a long 40-character that shows up on your main Github repo page next to the heading "Latest commit" (e.g. commit 6ca6f695bca36f7fc2c33485d1080ae30f8b9928). Locate the link to your team's repo. This is the URL to

your private repo on Github (e.g. https://github.com/cs327e-spring2017/xyz.git where xyz is your repo name). Replace the commit id and repo link in the json string below with your own:

```
{
  "repository-link": "https://github.com/cs327e-spring2017/xyz.git",
  "commit-id": "6ca6f695bca36f7fc2c33485d1080ae30f8b9928"
}
```

Create a file called **submission.json** that contains your modified json string.

Click on the Lab 1 Assignment in Canvas and upload submission.json. This submission is due by **Friday, 02/03 at 11:59pm**. If it's late, there will be a **10% grade reduction per late day**. This late policy is also documented in the syllabus. Note: Only one person per team should send the submission email.

Naming Conventions:

1. The first letter of an entity/table should be capitalized. If an entity/table contains more than one word, use an underscore between the words and capitalize the first letter of each word. For example: Movie_Companies.

2. All fields in an entity/table should be in lower cased. For example: movie_id.

Diagram Conventions:

1. Use the single-column Entity shape in Lucid for the conceptual diagram, which has only a textfield named Field.
2. Use the three-column Entity shape Lucid for the physical schema which has 3 textfields named Key, Field, and Type.

Teamwork:

1. We will use 2 class meetings (01/30 and 02/01) to work on this lab.
2. We expect each team-member to contribute equally to the assignment and we will be checking the commit history to ensure that this is happening as expected.
3. We want you to use the Github Issue Tracker to assign and track the status of tasks.

Resources:

[1] IMDB small csv dataset: https://s3.amazonaws.com/cs327e-imdb/csv_dataset_small.zip
[2] IMDB large csv dataset: https://s3.amazonaws.com/cs327e-imdb/csv_dataset_full.zip
[3] IMDB plain-text files: ftp://ftp.fu-berlin.de/pub/misc/movies/database/
[4] Lab 1 Setup Guide: https://github.com/wolfier/CS327E/wiki/Setting-up-Lab-1
[5] Lab 1 Grading Rubric: http://www.cs.utexas.edu/~scohen/assignments/rubric1.pdf
[6] Team Sign-up Sheet (In-Class Lab Sessions Only): http://tinyurl.com/hezev3m
[7] SXSW Diagrams: https://github.com/cs327e-spring2017/snippets