# Lab Report 3 – Video Database

## Databases - CSC 3320 121

### Hudson Arney

### 5 February 2024

Documentation:

Part 1.1 – Create a Video Database

1. Review videodb-readme.txt which contains field names for the Video\_Categories.txt, Video\_Actors.txt, and Video\_Recordings.txt data files.

A screenshot of a computer

Description automatically generated

I glanced over all files to get an idea of them. This helped me set up the queries in SQL.

1. Create a Video database (called schema in MySQL workbench).

A screenshot of a computer program

Description automatically generated

I later had to change ratings to varchar(10) because it is not an integer

A screenshot of a computer

Description automatically generated

1. Create an SQL script file (import.sql) that "load data" tables for importing the data from the tab-delimited text files.
   * ***Question to answer as an observation***: There are differences between importing csv- and tab-delimited files. Why would you select one format over another?
2. Run your script to load the tables from the data files (best done in command line by running individual commands).

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

I was confused for the actors because not all rows were being allowed. Once I made the show warnings command it made it easier to understand that the text files contained duplicate entries and that’s where all of the errors came from.

1. Use select commands on each table to verify the data has been successfully imported. See the load table primer at the end of this assignment if you need a little help with this step.

All the select commands looked proper. I was able to remove the quotes and for the actors, José worked successfully with the latin command.

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Part 1.2 – Design *Your* Database Schema

1. Create a relational data model for your Lab3 database (an ERD and a relational schema or equivalent). Use the MySQL Workbench Modeling Tool or erdplus (both ERD and relational schema) to show the recording, category, actor, and any other entities and their relationships for your final design. Your goal is to minimize/eliminate repeating data in multiple places, so for example the name of a person should only be saved in one place in the database.

ERD:

A diagram of a company

Description automatically generated

Summary:

* Recording can have one category, but a category could have many Recordings.
* This also goes for ratings where one recording (or movie) will have one rating, but a rating will be applied to many different recordings.
* An actor could be in many recordings, and a recording can have many actors.
* A director could be a part of many recordings (direct more than one movie), but a recording only has one director (although in the real world this isn’t always true, as there are some sibling group directors like the Russo or Cohen Brothers, the data within the dataset follows this rule).
* With this in mind I think it would be helpful if Recording had an attribute or column for starring actor/actress.

7. From MySQL Workbench or ERD plus to generate an SQL script to define and create the database tables for your DB schema design (or write the sql for this from scratch). Once your diagram is complete, use File → Export → Forward Engineer SQL CREATE Script to generate your SQL create tables script from MySQL Workbench, or choose Export SQL from the Diagram gear menu in the upper left from ERD Plus. Save this SQL in a .sql script file.

A lot of the variable types had to be changed in order to get this to work. Here is what my final SQL query looks like:

DROP TABLE IF EXISTS actor;

DROP TABLE IF EXISTS recording;

DROP TABLE IF EXISTS director;

DROP TABLE IF EXISTS category;

DROP TABLE IF EXISTS directed\_with;

DROP TABLE IF EXISTS associated\_with;

DROP TABLE IF EXISTS rating;

CREATE TABLE Actor

(

Actor\_ID INT NOT NULL,

Name VARCHAR(50) NOT NULL,

PRIMARY KEY (Actor\_ID)

)

ENGINE=INNODB;

CREATE TABLE Category

(

Name VARCHAR(50) NOT NULL,

Category\_ID INT NOT NULL,

PRIMARY KEY (Category\_ID)

)

ENGINE=INNODB;

CREATE TABLE Director

(

Name VARCHAR(50) NOT NULL,

Director\_ID INT AUTO\_INCREMENT,

PRIMARY KEY (Director\_ID)

)

ENGINE=INNODB;

CREATE TABLE Rating

(

Name VARCHAR(50) NOT NULL,

Rating\_ID INT AUTO\_INCREMENT,

PRIMARY KEY (Rating\_ID)

)

ENGINE=INNODB;

CREATE TABLE Recording

(

Recording\_ID INT NOT NULL,

Director VARCHAR(50) NOT NULL,

Title VARCHAR(50) NOT NULL,

Category VARCHAR(50) NOT NULL,

Image\_Name VARCHAR(50) NOT NULL,

Duration INT NOT NULL,

Rating VARCHAR(10) NOT NULL,

Year\_Released INT NOT NULL,

Price DOUBLE NOT NULL,

Stock\_Count INT NOT NULL,

PRIMARY KEY (Recording\_ID)

)

ENGINE=INNODB;

-- ------------------------------------------------------------------------------------------------

CREATE TABLE Associated\_With

(

Recording\_ID INT NOT NULL,

Actor\_ID INT NOT NULL,

PRIMARY KEY (Recording\_ID, Actor\_ID),

FOREIGN KEY (Recording\_ID) REFERENCES Recording(Recording\_ID),

FOREIGN KEY (Actor\_ID) REFERENCES Actor(Actor\_ID)

)

ENGINE=INNODB;

CREATE TABLE Directed\_With

(

Director\_ID INT NOT NULL,

Recording\_ID INT NOT NULL,

PRIMARY KEY (Director\_ID, Recording\_ID),

FOREIGN KEY (Director\_ID) REFERENCES Director(Director\_ID),

FOREIGN KEY (Recording\_ID) REFERENCES Recording(Recording\_ID)

)

ENGINE=INNODB;

-- ------------------------------------------------------------------------------------------------

-- Populate Actor table

INSERT INTO Actor (Actor\_ID, Name)

SELECT DISTINCT Actor\_ID, Name

FROM video\_actors;

-- Populate Recording table

INSERT INTO Recording (Recording\_ID, Director, Title, Category, Image\_Name, Duration, Rating, Year\_Released, Price, Stock\_Count)

SELECT DISTINCT Recording\_ID, Director, Title, Category, Image\_Name, Duration, Rating, Year\_Released, Price, Stock\_Count

FROM video\_recordings;

INSERT INTO Category (Category\_ID, Name)

SELECT DISTINCT Category\_ID, Name

FROM video\_categories;

INSERT INTO Director (Name)

SELECT DISTINCT TRIM(Director)

FROM video\_recordings;

INSERT INTO Rating (Name)

SELECT DISTINCT TRIM(Rating)

FROM video\_recordings;

INSERT INTO Associated\_With (Recording\_ID, Actor\_ID)

SELECT Recording\_ID, Actor\_ID

FROM Recording, Actor;

INSERT INTO Directed\_With (Director\_ID, Recording\_ID)

SELECT d.Director\_ID, r.Recording\_ID

FROM Director d

JOIN Recording r ON d.Name = r.Director;

8. Use ENGINE=INNODB at the end of each create table statement, but before the semi-colon. If you are interested, here is some more information about that: [https://stackoverflow.com/questions/1082194/when-should-you-choose-to-use-innodb-in-mysqlLinks to an external site.](https://stackoverflow.com/questions/1082194/when-should-you-choose-to-use-innodb-in-mysql)

The code above will show this in the created tables.

A screenshot of a computer

Description automatically generated

Stretch Goals for Lab 3 - part1 (work on these can carry over into part 2, so are optional for the submission for part 1):

9. In your table creation script (previous step 7), **add the SQL to load your final schema tables from your imported tables**. This can be a very involved and time consuming step. We will talk about some strategies in class. Some of this work  (steps 9, 10 and 11) can spill over into the second week of the lab if needed. See [this example](https://msoe.instructure.com/courses/18290/files/2350478?wrap=1)[Download this example](https://msoe.instructure.com/courses/18290/files/2350478/download?download_frd=1)for how this works in the context of the world\_db.

10. Verify you have primary key and foreign key constraints (when needed) for each entity and relationship. If not present, create them.

* ***Question for Observation:***Why would I create the primary key index after the table has been created and the data imported versus defining the primary key in the table definition?

11.   Run select commands on each table to verify the data has been successfully imported. Note that you will have 3 "load data" tables (based on the 3 text files you load data from), and then an additional 4 to 8 final design tables based on your ER design. Once we copy data over from the load tables into the final design tables, we no longer use the load tables (but don't delete them in case you mess up and need to reload your final tables).

Final Schema from Workbench

A screenshot of a computer

Description automatically generated