Francisco Delger

CSC423

Professor Tarek Sayed

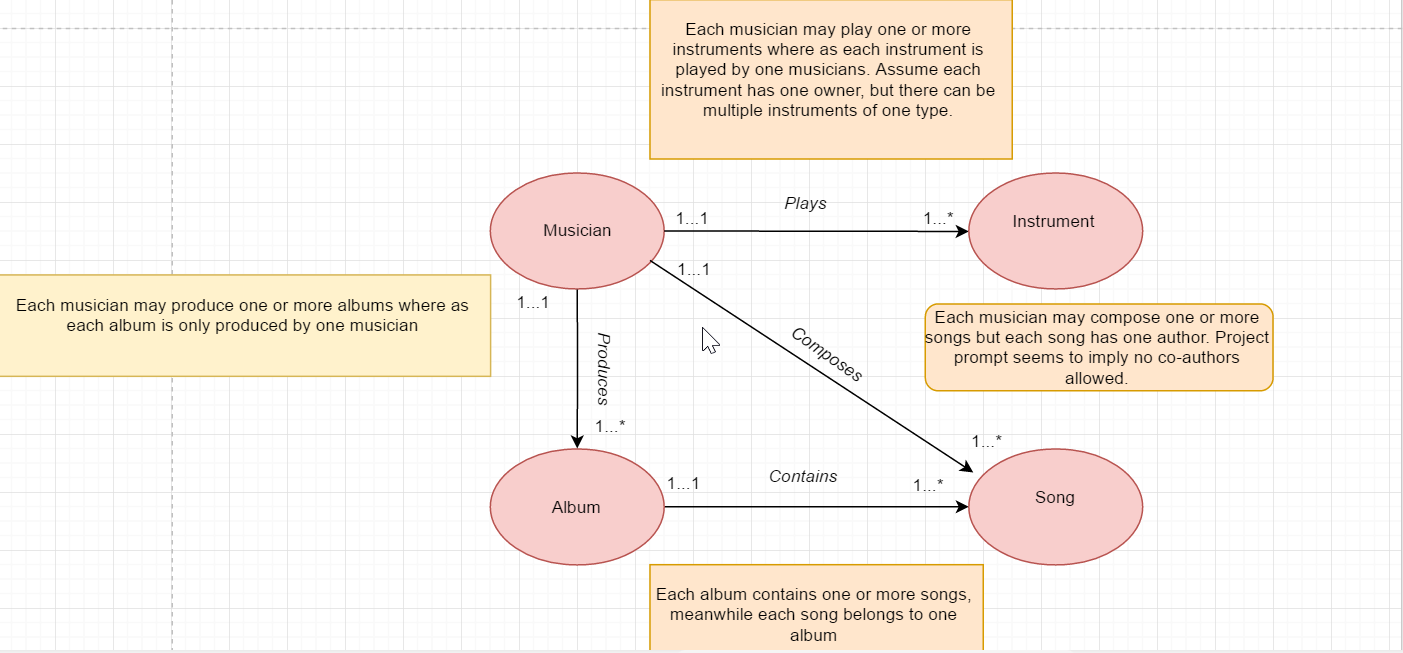
December 13th

Final Project Report

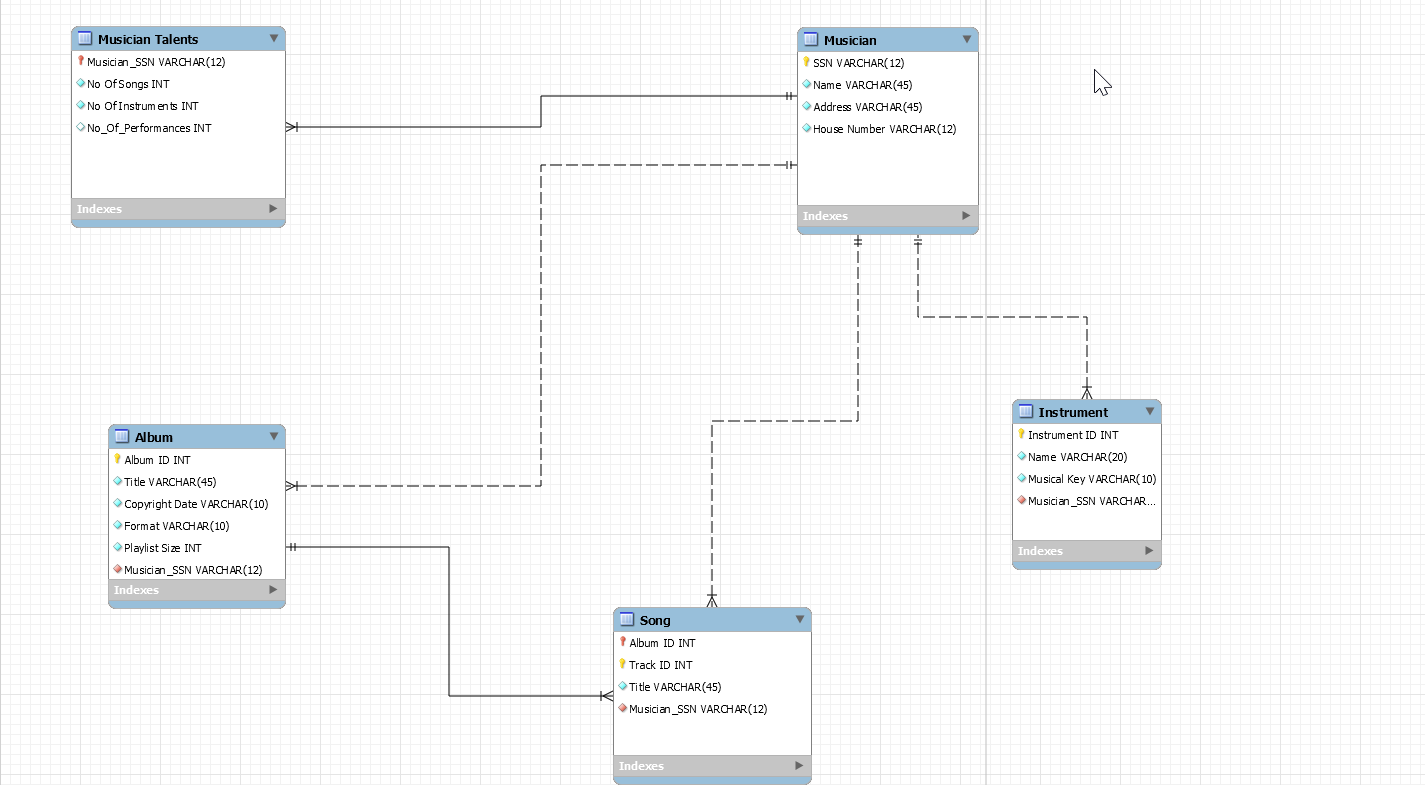
The objective of this report is to give a step by step walkthrough of the stages of development my Musician Database went through. From the identification of the relations all the way to the successful execution of queries. Therefore, I intend to type up this report by following the stage outline the professor established when assigning the project.

*Stage 1 and 2 (from rough ER to refined EER BCNF normalized)*

The conception of the data model. I began by identifying the general relationships between all the attributes in the project: Musician, album, song, etc. I wanted to have a general idea of the structure my model needed to have before sitting down to code the tables. I wanted to identify the relationships and their cardinality (which I went ahead of time before writing this report and addressed it in the model below with the orange signs, so you’ll have to forgive me for skipping over rewriting that). The result was the following:



Here I was able to establish the general skeleton of the database, and also organize the hierarchy of my data. I quickly saw that the musician table needed to have a key that worked everywhere, superkey for literally the entire database, but that is also easy to refer when I want to extract the data of the individual related to the key. The conclusion was very simple: use the Social Security Number. It is unique for each individual, and the personal information in the Musician Table satisfies the BCNF normalization if I chose it, so it was a no brainer.

From here on I had to start refining the model, this ER model wasn’t enough to start building the data. I had to identify the attributes needed in each relation, their constraints like unique, not null, PK, FK.... This is where the EER model comes in. I needed it always ready in case I had to get back to the drawing board and make changes (like I actually had to because of BCNF normalization), so I decided to use the MySQL one:

Here we have my final conception of the data model, identifying all the attributes and relations needed:

* Musician (SSN, Name, Address, Number) – Contains basic contact information. Primary key is SSN.
* Musician\_Talents (SSN, Songs produced, instruments played, performances done) – Contains facts related to each musicians career. It couldn’t go in the Musician relation because the dependencies SSN -> Songs, instruments made it ineligible for BCNF in the context of the Musician relation. SSN is the PK but it works this time because it is its own relation.
* Album (ID, title, copyright date, playlist size, format, Musician\_SSN) – Contains facts related to the album, which is why album ID was chosen as the primary key. Musician\_SSN was brought in as a foreign key because a producer id was needed. It had to be easily trackable to the musician’s info and the other relations in case subqueries were needed, so SSN was brought in.
* Instrument (ID, name, musical key, owner)– Contains detailed information for each instrument a musician knows how to play. Each instrument has its own ID in order to track it back to the musician that plays that particular instrument. This was originally designed like this foreseeing instrument names such as “fender Stratocaster guitar” and not just “guitar”, since each instrument is unique to each musician. However, since in one of the subqueries assigned this turned out very complicated if made this way, I simply settled for names like “guitar”, though I left the ID of the instrument feature unchanged because I think it’s a positive. PK is instrument ID.
* Song – (Track ID, Album ID, Title, SSN\_Author) Contains information pertaining the songs in an album. Very simple. PK is the combined track ID and album ID, which in turn is a FK from album. I felt like it was the most unique identifier to each song. Song 2 in album 1, song 3 in album 1… This line of thought made me think a 1-2, 1-3 and so on set up was the best setup, hence my choice.

A feature I decided to implement for this database was not to allow null values on any fields. I cannot conceive of a recording company who wouldn’t have any of this information on their files, assuming they’re a serious company. So, I implemented this database wide constraint. Please see below (or the Tables.sql file attached on the email you will be receiving) for the implemention of each relation into MySQL connected to Google Cloud:

CREATE TABLE Musician (

SSN VARCHAR(11) NOT NULL,

Full\_Name VARCHAR(45) NOT NULL,

Address VARCHAR(45) NOT NULL,

House\_Number VARCHAR(12) NOT NULL UNIQUE,

PRIMARY KEY(SSN)

);

CREATE TABLE Musician\_Talents (

No\_Of\_Songs INT NOT NULL,

No\_Of\_Instruments INT NOT NULL,

SSN VARCHAR(11) NOT NULL UNIQUE,

PRIMARY KEY(SSN) ,

FOREIGN KEY(SSN)

REFERENCES Musician(SSN) ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE Instrument (

InstrumentID INT NOT NULL UNIQUE,

Full\_Name VARCHAR(45) NOT NULL,

Musical\_Key VARCHAR(10) NOT NULL,

SSN VARCHAR(11) NOT NULL,

PRIMARY KEY(InstrumentID),

FOREIGN KEY(SSN)

REFERENCES Musician(SSN) ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE Album (

AlbumID INT NOT NULL UNIQUE,

Title VARCHAR(45) NOT NULL,

Copyright\_Date VARCHAR(10) NOT NULL,

Format VARCHAR(10) NOT NULL,

Playlist\_Size INT NOT NULL,

SSN VARCHAR(11) NOT NULL ,

PRIMARY KEY (AlbumID),

FOREIGN KEY(SSN)

REFERENCES Musician(SSN) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE Song (

AlbumID INT NOT NULL,

TrackID INT NOT NULL,

Title VARCHAR(45) NOT NULL,

SSN\_Author VARCHAR(11) NOT NULL,

PRIMARY KEY(AlbumID, TrackID),

FOREIGN KEY(SSN\_Author)

REFERENCES Musician(SSN) ON UPDATE CASCADE ON DELETE CASCADE

);

You will notice an update/delete cascade option was enabled for the FKs. This was so that in case a musician was deleted from the records, so was all the information related to them, in order to maximize storage capacity. Next, the implementation of tuples into the relation.

*Stage 3 and 4 (implementation on MySQL/Google Cloud)*

Getting the tuples in the relations was the first thing to do. I needed data in order to verify the validity of my queries. See below (or, again, the separate file attached called Insertions.sql to see in better format) the SQL code implemented for the tuples:

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("477-13-1811","Mick Jagger","777 cat st",3815940121);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("477-13-1811",100,1);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (1,"Harmonica","C","477-13-1811");

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (1,"Exile on Main St.","05-12-1972","CD",18,"477-13-1811");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (1,1,"Rocks Off","477-13-1811");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("456-76-1811","John Frusciante","738 turtle st",7893820121);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("456-76-1811",80,1);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (2,"Fender Standard Guitar","G Major","456-76-1811");

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (2,"Stadium Arcadium","04-03-2006","CD",28,"456-76-1811");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (2,1,"Dani California","456-76-1811");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (2,2,"Snow","456-76-1811");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("179-13-1241","Keith Moon","1270 whiskey st",7283645152);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("179-13-1241",54,1);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (3,"Tama Drums","D","179-13-1241");

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (3,"Who's Next","8-14-1971","CD",9,"179-13-1241");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (3,9,"Won't Get Fooled Again","179-13-1241");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("079-13-1001","Roger Waters","1270 whiskey st",7283645152);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("079-13-1001",54,1);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (4,"Piano","E-Flat","079-13-1001");

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (4,"Dark Side of the Moon","3-1-1973","CD",10,"079-13-1001");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (4,6,"Speak To Me","079-13-1001");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("619-92-2901","Chris Wolstenholme","1872 Black River st",9683194152);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("619-92-2901",34,1);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (5,"Bass","G-Flat","619-92-2901");

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (5,"Absolution","9-22-2003","CD",14,"619-92-2901");

INSERT INTO Song (AlbumID,TrackID,Title,SSN\_Author)

VALUES (5,14,"Fury","619-92-2901");

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("063-21-2001",24,3);

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (7,"Bass","G-Flat","063-21-2001");

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (6,"Guitar","C-Flat","063-21-2001");

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (8,"Drums","Any","063-21-2001");

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (9,"Guitar","Any","619-92-2901");

INSERT INTO Instrument (InstrumentID,Full\_Name,Musical\_Key,SSN)

VALUES (10,"Synthesizer","Any","619-92-2901");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("231-22-2903","Lady Gaga","123 Riverwalk st",2288994852);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("231-22-2903",24,0);

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (6,"Cheek to Cheek","1-20-2015","CD",10,"231-22-2903");

INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number)

VALUES ("971-22-2253","Katy Perry","103 NoMusicallSkill st",7123994842);

INSERT INTO Musician\_Talents (SSN,No\_Of\_Songs,No\_Of\_Instruments)

VALUES ("971-22-2253",36,0);

INSERT INTO Album (AlbumID,Title,Copyright\_Date,Format,Playlist\_Size,SSN)

VALUES (7,"Imaginary Album","11-30-2015","CD",12,"971-22-2253");

I decreased the font in order for it to not occupy so much space and look nicer. It’s essentially the same queries over and over again so the database had enough data to work with. (There is a bonus Katy Perry dig from yours truly).

Once the data is ready, we are set to start working on some actual code and get some visual results on the information we want to consult. The implementation of the code was done through Java with Eclipse. The code consists of a Main.java and QueryConnection.java. Main calls the other and then the magic happens. The code is attached on the email you will be receiving along with this report, it takes too much space in this file. The queries had the following menu were as follows:

Menu:

Please select one of the following options, 0 to exit:

1: Insert new Musician into the Database

2: Delete a song from the Database

3: Change the address of a musician

4: List albums by a certain musician

5: List Musicians who know how to play a given instrument ordered alphabetically

6: List Songs of Desired Album

7: Number of Albums in 2015

8: List Musicians who have produced more than the average number of songs in the company

9: Musicians who play more than two instruments

10: List Number of Performances by Each Musician

1. Insert a new musician into the Database.

I used a ps.updatestatement and scanners, as you will see on my code, in order to get this query working. I filled out the following query:

Query = "INSERT INTO Musician (SSN,Full\_Name,Address,House\_Number) VALUES (?,?,?,?)";

The output of the program when running this query would be:

Input Option here, once you hit enter, scroll up for results:

1

Input the SSN of the musician you wish to register:

923-92-2192

Input the full name of the musician:

Jim Morrison

Input the home address of the musician:

9204 Allendale Rd

Input the home phone number:

1292920309

Submitted information successfully

1. Deleting a song from the database. Works similar to the first one:

query = "Delete from Song where Title=?";

There is no output for this query other than the successful completion, but I invite you to try it out and delete the song Hysteria from the Absolution album by muse. If you then try to list the songs that belong to album id =5 (absolution) you will see hysteria missing.

1. Changing the address where a Musician lives:

query = "Update Musician set Address=? where Full\_Name=?";

Output:

Input the Musician you desire to alter:

Jim Morrison

Input the new address:

3919 Changing St

Submitted information successfully

1. List albums by a certain musician (given by the user).

query = "Select AlbumID, Copyright\_Date, Format from Album where SSN = (Select SSN from Musician where Full\_Name=?)";

Output:

Input Option here, once you hit enter, scroll up for results:

4

Input the Musician you desire to list:

Chris Wolstenholme

The Query result is:

AlbumID Copyright Format

5 9-22-2003 CD

Submitted information successfully

1. List musicians who know how to play a given instrument, ordered alphabetically:

query = "Select m.SSN, m.Full\_Name, m.Address, m.House\_Number from Musician m, Instrument i where m.SSN = i.SSN and i.Full\_Name=? order by m.Full\_Name DESC";

Output:

Input Option here, once you hit enter, scroll up for results:

5

Input the instrument you desire to list:

Guitar

The Query result is:

SSN Name Address Number

063-21-2001 Kurt Cobain 2809 Leaf Road 2020092212

619-92-2901 Chris Wolstenholme 1872 Black River st 9683194152

1. List all songs in a given album:

query = "Select s.TrackID, s.Title from Song s, Album a where a.AlbumID=(Select AlbumID from Album where Title=?) and a.AlbumID = s.AlbumID";

Output:

Input Option here, once you hit enter, scroll up for results:

6

Input the Album you desire to list:

Absolution

The Query result is:

TrackID Title

1 Apocalypse Please

2 Time is Running Out

3 Sing for Absolution

4 Stockholm Syndrome

5 Falling Away With You

6 Interlude

7 Hysteria

8 Blackout

9 Butterflies and Hurricanes

10 The Small Print

11 Endlessly

12 Thoughts of a Dying Atheist

13 Ruled by Secrecy

14 Fury

Submitted information successfully

1. Number of Albums with a Copyright date in 2015:

query = "Select Count(\*) as NoOfAlbums from Album where Copyright\_Date between \"1-1-2015\" AND \"12-31-2015\"";

Input Option here, once you hit enter, scroll up for results:

7

The Query result is:

NoOfAlbums

2

1. List musicians who have produced more than the average number of songs in the company. (I know you requested albums, but it’s a matter of semantics really. I could change the field in my relation from “no of songs” to “no of albums” and it would be the same. It’s just that the numbers are big, so it made more sense to keep it as songs.

query = "Select m.Full\_Name, n.No\_Of\_Songs from Musician m, Musician\_Talents n where No\_Of\_Songs >=(Select AVG(No\_Of\_Songs) from Musician\_Talents) AND m.SSN = n.SSN";

Input Option here, once you hit enter, scroll up for results:

8

The Query result is:

Name Amount Produced

Roger Waters 54

Keith Moon 54

John Frusciante 80

Mick Jagger 100

1. Musicians who play more than two instruments

query = "Select m.Full\_Name from Musician m, Musician\_Talents n where n.No\_Of\_Instruments>=2 and m.SSN = n.SSN;";

Input Option here, once you hit enter, scroll up for results:

9

The Query result is:

Name

Kurt Cobain

Chris Wolstenholme

1. List number of performances by each musician

query = "Select n.Full\_Name, m.No\_Of\_Performances from Musician\_Talents m, Musician n where m.SSN = n.SSN";

Input Option here, once you hit enter, scroll up for results:

10

The Query result is:

Name Performances

Kurt Cobain 195

Roger Waters 1726

Keith Moon 495

Lady Gaga 620

John Frusciante 876

Mick Jagger 1450

Chris Wolstenholme 920

Katy Perry 30

And with this last query, we have reached the end of this report. As for a conclusion, I don’t have much to say in regards to this project. I would say the most challenging part is definitely the data model, conception, normalization, etc. and not so much the coding. It does seem like at times normalization is almost counterproductive for efficiency, I think I even read there is a lossless decomposition principle which seems to support my feelings toward it, but I can still see its advantages.

The class overall has been a great learning experience and I feel like I have acquired an actual set of skills to use in real life, unlike with other classes. What I appreciate the most is having been taught to implement google cloud, VM, along with mysql. We could have just ran a few queries and such, but you pushed us further, I greatly appreciate it. Please feel free to run any queries you wish to test my data whatsoever. I had to rush it given my circumstances, so I apologize for any sloppiness, but I also greatly appreciate that you gave me this extension to get it done, I honestly really do. My sister is doing better, she left the ICU on Sunday and looks like everything is going as expected. I feel very relieved, from person to person, thank you for helping me out.