UNIVERSITY OF WASHINGTON, SUMMER QUARTER 2017

PROJECT ITERATION 2

DESIGN DOCUMENT

Designed and Developed by Team ProjectX

Reshma Maduri Sivakumar *Dawit Y Abera*

Haimanot Zerkel *Lok Him Tam*

AUGUST 10, 2017

TABLE OF CONTENTS

PART I: Design Documentation 3

1. Introduction 3
2. Database Application 3
3. User requirement 3
4. System description and functionalities 5
5. Entities 6
6. Constraints 7
7. Relationships 8
8. Entity Relationship Diagram 8
9. Relational Data Model 10

PART III: Database Creation with SQL Scripts 11

1. Create SQL Statements: 11
2. Insert SQL statements to populate data 12

PART IV : Feedback / Changes from Iteration 1 17

PART I: Design Documentation

1. Introduction

This document is an overview of the the 2nd iteration for the Online Movie Store data model created and developed by team ProjectX for Database Systems class at University of Washington. The following sections describes the application through the ER Schema and Relational Model and also lists all the relationships, constraints, limitations and assumptions. The document also gives an overview of the method that the team developed to create the database, including some sql statements in order to create and populate the database along with the test cases adopted.

2. Database Application

The application area of the database is going to be an online movie store that has a collection of different kinds of movies and customer information. The database allows the video store to manage the rental of videos, keep track of inventory and provide enhanced service to their customers. Some of the things we would like to do once the database is up and running are

1. Search for customers by last name or phone number
2. Keep track of which movies customers rented, on what dates, and how much they spent on each date and in total
3. Search for movies by movie name or type of movie
4. Search for movies that have a certain actor

3. User requirement

CUSTOMER:

- Each customer’s will have a first name, middle initial and last name
- Each customer has unique ID number provided by the store. The unique ID is 6 digits total.
- Each customer’s will have an address comprising of street, city, state and zip code.
- Each customer’s subscription date will be stored
- Each customer’s phone and email address is stored
- Each customer has a date of birth
- Each customer must have a credit card that can be used to pay for the subscription

MOVIE:

- Each movie has a unique ID
- Each movie has a title,genre,actors name,director's name, release date,movie type(HD/Non-HD)
- Each movie has a copy (1-20). A movie copy does not exist if the corresponding movie does not exist

CHECKOUT:

- A checkout is a unique transaction occurring any particular time a customer checks out a movie
- A customer makes a rental, the customer he/she is at least one copy, any may rent up to 5 copies.
- Each rental is made by exactly one customer
- A customer can renew their rental and submit their rental

PAYMENT:

- Each transaction/payment is uniquely identified by the payment ID
- Only the credit card type is stored in the database ** due project limitation
- Each payments credit card may belong to exactly one customer

- The payment due date is stored in database. Payments will be due after 29 days of subscription

SUBSCRIPTION:

- Each subscription is uniquely identified by the subscription ID
- Each subscription offers three types of subscription plan: HD, Non-HD and Tv-shows
- Each subscription package has it's own price.

4. System description and functionalities

The DBMS will be a database with a user interface that will allow the Movie Store's data to be viewed, stored and updated using sql queries to the the database. These will allow the store to maintain and have easy and efficient access to accurate and uptodate records relevant to the Movie Store information.
The system will facilitate the rental of movie copies by updating database tables. It will also also allow the store to easily look for a customer and all sort of queries by using complex queries.

System functionality
The system is mainly a DBMS build around a SQL database and a Java GUI. The system has several main functions and those will be delivered.

- The system will let the a user connect to the database with the mean of a username and password
- The system will let a user search or see a movie information
- The system will allow a user to subscribe to our HD, Non-HD or Tv-show offers and also allow the user to rent a movie
- The system will allow a user to update their personal customer information
- The system will allow the admin/employee to add and update a movie to the database
- The system will allow the admin/employee to search and update the record for a specific customer using the customer's ID.
- The system will allow the admin/employee to add credit card type to the customer record
- The system will allow a customer to rent only 5 copies at one time

Due to time constraints and project limitation:

- the system assumes only people(employees) with administrative access can delete customer information
- The system only assumes only one admin has access to the DBMS
- The system does not perform any form of transactions. The credit card informations are only saved on the DB.

5. Entities

Customer

Primary key: Customer_ID

Foreign key: Subscription_id references SUBSCRIPTION record.

Attributes: Customer_id, Customer_fname, Customer_minit, Customer_lname, Customer_phone, Customer_email, Address, City, State, Zip, DOB, Subscription_date

Movie

Primary key: Movie_id

Foreign key: Movie_type , Movie_genre

Attributes: Movie_id, Movie_title, Movie_year, Movie_genre, Moive_director, Movie_actor, Movie_copy, Movie_type

Sub_type

Primary key: sub_ID

Foreign key: None

Attributes: sub_id, type, price

Genre

Primary key: Genre_ID

Foreign key: None

Attributes: Genre_ID, Genre_type

Payment

Primary key: Payment_id

Foreign key: Customer_id references CUSTOMER record, Subscription_id references SUBSCRIPTION record.

Attributes: Payment_id, Payment_method, Payment_date, Customer_id, (Customer_id), Subscription_id

Subscription

Primary key: Subscription_id

Foreign key: Customer_id

Attributes: Subscription_id, Customer_id, Sub_plan, sub_type

Checkout

Foreign key: Customer_id, Movie_id

Attributes: Customer_id, Movie_id, renew_count, checkout_time

6. Constraints

The three main types of integrity constraints are: Key, Entity integrity and Referential integrity constraints.

CUSTOMER TABLE:

Key: customer_ID is the primary key and must hold "unique" values. Values cannot be repeated for customer_ID.

Entity Integrity: customer_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: None. No foreign keys in Customer table.

MOVIE TABLE

Key: Movie_ID is the primary key and must hold "unique" values. Values cannot be repeated for Movie_ID.

Entity Integrity: Movie_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: Movie_Id references MOVIE table and

SUBSCRIPTION TABLE:

Key: subscription_ID is the primary key and must hold "unique" values. Values cannot be repeated for subscription_ID.

Entity Integrity: subscription_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: Customer_id references CUSTOMER table, Sub_id references SUB_TYPE record. Customer_id and Sub_id cannot be NULL and must be a value that already exists in the parent table.

SUB_TYPE TABLE

Key: sub_ID is the primary key and must hold "unique" values. Values cannot be repeated for sub_ID.

Entity Integrity: sub_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: None. No foreign keys in Sub_type table.

GENRE TABLE

Key: genre_ID is the primary key and must hold "unique" values. Values cannot be repeated for genre_ID.

Entity Integrity: genre_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: None. No foreign keys in Genre table.

PAYMENT TABLE

Key: Payment_ID is the primary key and must hold "unique" values. Values cannot be repeated for Payment_ID.

Entity Integrity: Payment_ID is the primary key and hence cannot be a NULL value.
Referential Integrity: Customer_id references CUSTOMER table, Subscription_id references SUBSCRIPTION table. Customer_id and Subscription_id cannot be NULL and must be a value that already exists in the parent table.

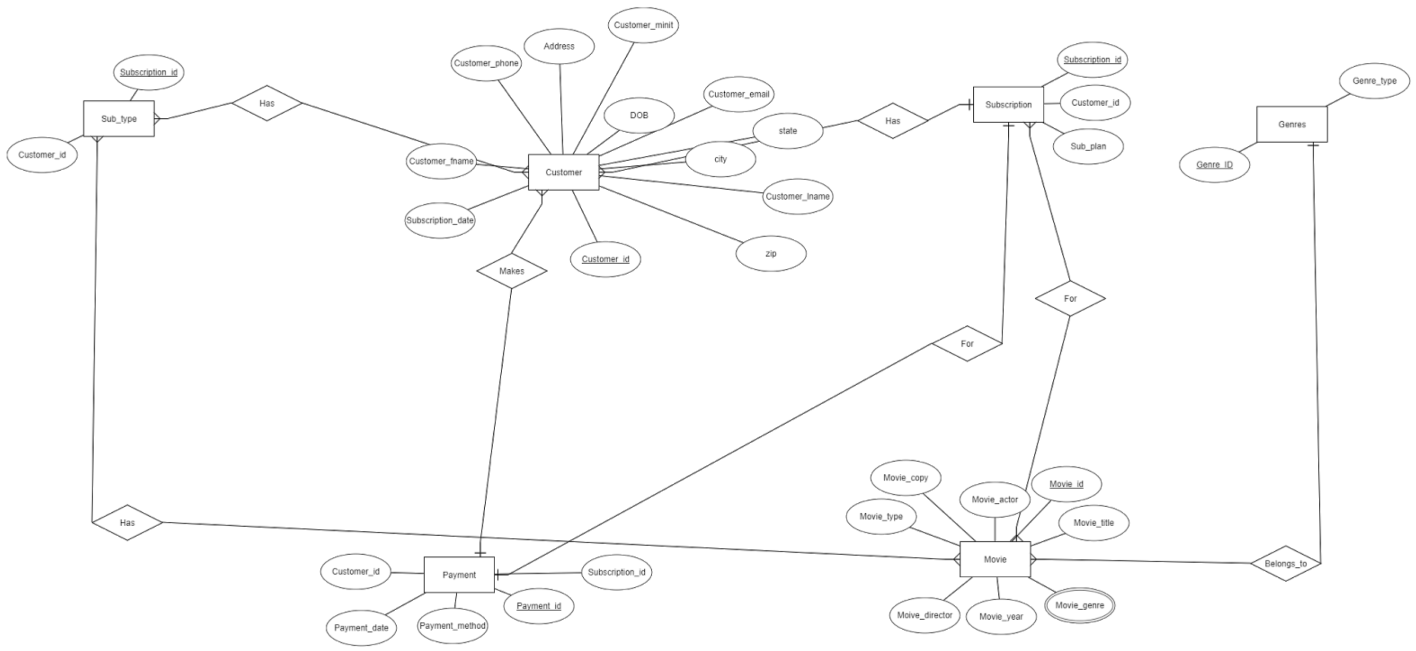
7. Relationships

One CUSTOMER record has one and only one SUBSCRIPTION record.
One CUSTOMER makes one PAYMENT record.

One Movie only have one subscription type.
One PAYMENT record for one SUBSCRIPTION record.
One subscription have more than one movie.
One MOVIE record belongs to one and only one GENRE record.

8. Entity Relationship Diagram

Entity-Relationship Diagram: Use standard ER notations as discussed in class



-
-

9. Relational Data Model

- Relational data model: Include all relations, attributes, primary key, foreign key & any other constraints (if applicable) **(Himson/Haimanot)**

PART III: Database Creation with SQL Scripts

The database is created using MySQL and Cloud Shell on Google Cloud Platform.

1. Create SQL Statements:

```
CREATE TABLE Customer (
Customer_id INTEGER PRIMARY KEY NOT NULL UNIQUE,
Customer_fname VARCHAR(50) NOT NULL,
Customer_minit CHAR(1),
Customer_lname VARCHAR(50) NOT NULL,
Customer_phone VARCHAR(30) NOT NULL,
Customer_email VARCHAR(100),
Address VARCHAR(100) NOT NULL,
City VARCHAR(30) NOT NULL,
State CHAR(2) NOT NULL,
Zip INTEGER NOT NULL,
DOB DATE NOT NULL,
Subscription_date DATE NOT NULL
);

CREATE TABLE Sub_type (
sub_id INTEGER PRIMARY KEY,
type VARCHAR (8),
price INTEGER
);

CREATE TABLE Genre (
Genre_ID INTEGER PRIMARY KEY UNIQUE NOT NULL,
Genre_type VARCHAR(15) NOT NULL
);

CREATE TABLE Subscription (
Subscription_id  INTEGER  PRIMARY KEY not null,
Customer_id INTEGER REFERENCES Customer (Customer_id),
Sub_plan DECIMAL(5,2) REFERENCES sub_type (sub_id)
);

CREATE TABLE Movie (
Movie_id INTEGER PRIMARY KEY UNIQUE,
Movie_title VARCHAR (20) NOT NULL,
Movie_year DATE,
Movie_genre INTEGER REFERENCES Genre (Genre_ID),
Moive_director VARCHAR (20) NOT NULL,
Movie_actor VARCHAR (20) NOT NULL,
Movie_copy INTEGER,
Movie_type INTEGER REFERENCES sub_type (sub_id)
);
CREATE TABLE Payment (
Payment_id INTEGER PRIMARY KEY UNIQUE,
Payment_method CHAR (5) NOT NULL,
Payment_date DATE,
Customer_id INTEGER REFERENCES CUSTOMER (Customer_id),
Subscription_id INTEGER REFERENCES Subscription (Subscription_id)
);
```

2. Insert SQL statements to populate data

POPULATING DATA INTO *ProjectXDB* DATABASE:

The method that the group adopted in order to import data onto the ProjectXDB database

SQL INSERT SCRIPTS:

Inserting into Customer table:

```
insert into Customer values (11110, 'Helen', '', 'James', '2061223221',
'helen@gmail.com', '145th Pine St', 'Seattle', 'WA', 98012, '1978-02-13',
'2017-02-01');

insert into Customer values (11111, 'Erik', 'M', 'Flint', '323-646-63',
'ErikBNolen@armyspy.com', '1111 Evergreen Lane', 'San Francisco', 'CA',
90040, '1964-12-06', '2017-05-25');

insert into Customer values (11112, 'Gary', 'A', 'Freeman', '910-236-7112',
'GaryAMcCarthy@superrito.com', '3803 Clarence Court', 'Appleton', 'WI',
28540, '1943-06-26', '2016-11-14');

insert into Customer values (11113, 'Jack', '', 'Hahn', '618-921-4848,
'JackCFlint@teleworm.us', '3770 Davis Court', 'Appleton', 'WI', 98012,
'1978-02-13', '2017-02-01');

insert into Customer values (11114, Thomas, '', 'Parker',  '908-526-5025',
'ThomasMFreeman@armyspy.com', '1357 Caynor Circle', 'San Francisco', 'CA',
8876, '1974-01-24', '2017-02-05' );


insert into Customer values (11115, 'Rita', 'R', 'Foy', '321-396-0863',
'RitaMHahn@superrito.com',  '3870 Terry Lane', 'San Francisco', 'CA', 32789,
'1966-12-10', '2017-02-06');
```

```
insert into Customer values (11120, ‘Janice’, ‘M’, ‘Toothaker’,
‘956-948-4861’, ‘JaniceJHughes@fleckens.hu’, ‘2812 Carolina Avenue’,
‘Seattle’, ‘WA’, 28540, ‘1967-03-10’, ‘2016-02-15’);

insert into Customer values (11121, 'Shannon', 'A', 'Wexler',
'773-213-8530', 'ShannonKHicks@dayrep.com', '4290 Oakmound Road', 'Seattle',
'WA', 60605, ‘1998-01-16’, '2016-06-20');


insert into Customer values (11122, 'Roland', 'F', 'Miles', '407-814-3440',
'RolandTToothaker@dayrep.com', '1441 Grand Avenue', 'San Francisco', 'CA',
32703, ‘1966-05-05’, '2016-02-23');


insert into Customer values (11123, 'Mary', 'R', 'Goodwin', '702-327-2609',
'MaryCWexler@jourrapide.com', '1366 Mesa Drive', 'Las Vegas', 'NV', 89101,
‘1962-01-13’, '2017-07-13');
```

Inserting into Sub_type table

```
insert into Sub_type values (3455, 'Non-HD', 20);

insert into Sub_type values (3456, ‘HD’, 40);

insert into Sub_type values (3457, ‘TV-Shows’, 10);
```

Inserting into Subscription table

```
insert into Subscription values ( 45300, 11110, ‘2/1/17’, 3455);

insert into Subscription values (45301, 11111, ‘5/25/17’, 3455);

insert into Subscription values (45302, 11112, ‘3/29/16’, 3456);

insert into Subscription values (45303, 11113, ‘11/14/16’, 3456);

insert into Subscription values (45304, 11114, ‘2/5/17’, 3457);

insert into Subscription values (45305, 11115, ‘2/6/17’, 3457);

insert into Subscription values (45306, 11116, ‘2/7/17’, 3457);

insert into Subscription values (45307, 11117, ‘5/12/17’, 3457);

insert into Subscription values (45308, 11118, ‘2/24/17’, 3455);

insert into Subscription values (45309, 11119, ‘2/28/17’, 3455);

insert into Subscription values (45310, 11120, ‘2/15/16’ 3455);

insert into Subscription values (45311, 11121, ‘6/20/16’, 3455);
insert into Subscription values (45312, 11122, ‘2/23/16’, 3456);
insert into Subscription values (45313, 11123, ‘7/13/17’, 3457);
insert into Subscription values (45314, 11124, ‘5/25/17’, 3457);
insert into Subscription values (45315,  11125, ‘6/14/16’, 3457);
insert into Subscription values (45316, 11126, ‘9/20/16’, 3456);
insert into Subscription values (45317, 11127, ‘11/22/16’, 3455);
insert into Subscription values (45318, 11128, ‘2/19/17’, 3456);
```

Inserting into Payment table

```
insert into Payment values (21200, 11110, 45300);
insert into Payment values (21201, 11111, 45301);
insert into Payment values (21202, 11112, 45302);
insert into Payment values (21203, 11113, 45303);
insert into Payment values (21204, 11114, 45304);
insert into Payment values (21205, 11115, 45305);
insert into Payment values (21206, 11116, 45306);
insert into Payment values (21207, 11117, 45307);
insert into Payment values (21208, 11118, 45308);
insert into Payment values (21209, 11119, 45309);
insert into Payment values (21210, 11120, 45310);
insert into Payment values (21211, 11121, 45311);
insert into Payment values (21212, 11122, 45312);
insert into Payment values (21213, 11123, 45313);
insert into Payment values (21214, 11124, 45314);
insert into Payment values (21215, 11125, 45315);
insert into Payment values (21216, 11126, 45316);
insert into Payment values (21217, 11127, 45317);
insert into Payment values (21218, 11128, 45318);
```

Inserting into Movie table

```
INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1001,'007 - 00 James Bond Story',1999,'James
Bond','Chris Hunt','Daniel Craig',10,3455);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1002,'Air',2015,'Sci-Fi','Christian
Cantamessa','Lewis Gilbert',10,3455);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1003,'Abbitte (Atonement)',2007,'Drama','Joe
Wright','Guy Hamilton',10,3455);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1004,'Aladdin',1992,'Animation','Walt
Disney','John Glen',10,3456);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1005,'Anna Karenina',1997,'Drama','Bernard
Rose','Guy Hamilton',10,3456);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1006,'Baader Meinhof Komplex:
Der',2008,'Crime','Uli Edel','Guy Hamilton',10,3456);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1007,Bones: Die Knochenjägerin  (Season
1)',2005,'TV-Serie','Jesús Salvador Treviño','John Glen',10,3457);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1008,'Bones: Die Knochenjägerin  (Season
2)',2008,'TV-Serie','Jesús Salvador Treviño','John Glen',10,3457);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1009,'Bones: Die Knochenjägerin  (Season
3)',2009,'TV-Serie','Jesús Salvador Treviño','John Glen',10,3457);

INSERT INTO
Movie(Movie_id,Movie_title,Movie_year,Movie_genre,Moive_director,Movie_actor,M
ovie_copy,Movie_type) VALUES (1010,'Bones: Die Knochenjägerin (Season
4)',2010,'TV-Serie','Jesús Salvador Treviño','John Glen',10,3457);
```

Inserting into Genre table

```
insert into Genre values (400, ‘James Bond’);

insert into Genre values (401, ‘Sci-Fi’);

insert into Genre values (402, ‘Drama’);

insert into Genre values (403, ‘Crime’);

insert into Genre values (404, ‘TV-serie’);

insert into Genre values (405, ‘Animation’);

insert into Genre values (406, ‘Fantasy’);
```

Inserting into Checkout table

4. Populate your database with sample data. [5pt] **(Reshma)**

- ○ Don’t spend an inordinate amount of time populating the database. This is important for the sample queries, but will not be a major part of the grade.
    - ■ Your SQL file should include ALL the inserts you have used so that we can regenerate and populate your database.
    - ■ Your document may contain only a few of the inserts, consider including the most relevant and the ones that show **constraint violation testing.**
- ○ "Upload" your database file and list in the report the name of the tables used by your project.
- ○ **You are encouraged to use a cloud server (or similar) to host your database, and give the professor and the grader access to your database**

- If you are not hosting the database on a server and you are using MySQL or SQLite, you need to submit all the SQL queries to recreate and populate your database.

(Reshma, Haimanot, Dawit, Himson)

- **Explain in detail your methodology to create test data:**
  - **Reference where you obtained from (e.g., online sources, actual application)**
  - **Explain how you processed it (e.g., if you used a program or script)**

- Include the code together with your submission, or discuss it in the design document (e.g., you can use code snippets as figures)

Methodology

(dawit- working on it--)

Testing

SPIRAL AND AGILE

NOTE: You are not expected to completely populate the database for this iteration, however, you should have essential data for testing, and/or you should device a strategy for generating data automatically for testing purposes. (Himson/dawit/Haimanot/Reshma)

PART IV : Feedback / Changes from Iteration 1

1. Feedback from Iteration 1:

1. Introduce ER Diagram and RDM figures and not leave it as "floating figures".
2. Incorporate cardinality or min max notation on the diagram.
3. Include introductions, descriptions, assumptions, concerns, limitations and or explanations for diagrams.
4. Give use case more context as well, as opposed to just having a simple diagram.

5. Good documentation that is understood by all.

2. Changes/Improvements:

1. Each entities on the ER Diagram are described along with their attributes, primary keys and foreign keys stated.
2. The ER Diagram now incorporates the min max notation for more clarity on the relationships.
3. The document