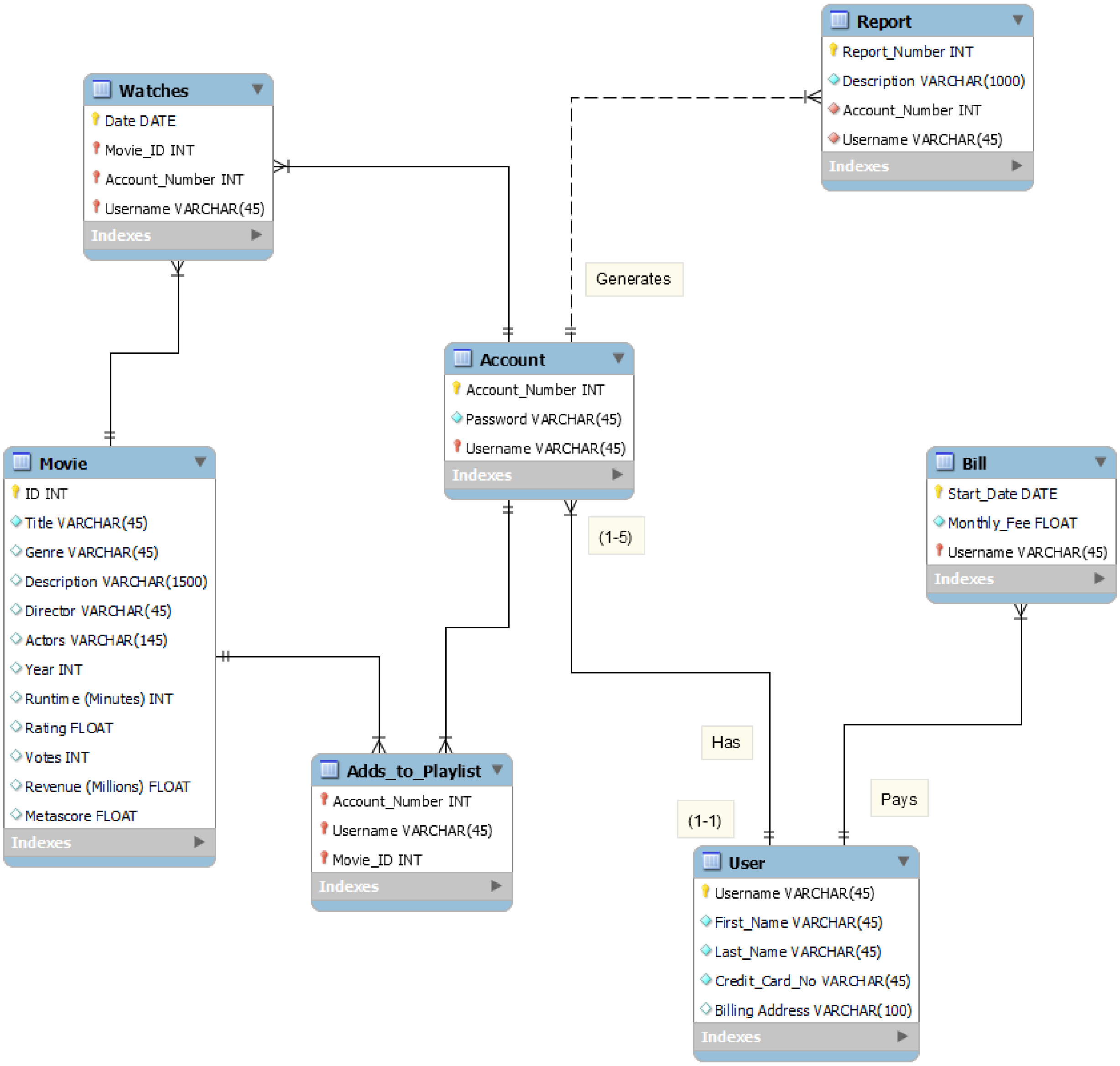
Ramin Mousivand_CMPE226_Lab1

Project Description

In this project a database system similar to Netflix is designed. Users can create accounts, pay their monthly bills and enjoy watching movies.

System Description

Customers open an account by giving their info, and then can sign into their accounts and paying the for the service. As soon as they pay, an account will be created for them. They can build up to 5 accounts. Account 1 is always considered for the customer himself or herself. Each account has its own number and encrypted password. Accounts will have a watch history and only one playlist. Movies can be searched through stored procedures for enhanced security by title or actors, or filter ratings more than a given value. Also, each account can generate multiple report in case of a problem. Movies must have title, and id. Others fields might be null. The movies may be added to account playlist. Watching history is recorded for all accounts. If a user is, deleted, its history will be kept for data analysis purposes.


Report
Report_Number INT
Description VARCHAR(1000)
Account_Number INT
Username VARCHAR(45)
Indexes
Watches
Date DATE
Movie_ID INT
Account_Number INT
Username VARCHAR(45)
Indexes
Generates
Account
Account_Number INT
Password VARCHAR(45)
Username VARCHAR(45)
Indexes
Movie
ID INT
Title VARCHAR(45)
Genre VARCHAR(45)
Description VARCHAR(1500)
Director VARCHAR(45)
Actors VARCHAR(145)
Year INT
Runtime (Minutes) INT
Rating FLOAT
Votes INT
Revenue (Millions) FLOAT
Metascore FLOAT
Indexes
(1-5)
Bill
Start_Date DATE
Monthly_Fee FLOAT
Username VARCHAR(45)
Indexes
Has
Pays
(1-1)
Adds_to_Playlist
Account_Number INT
Username VARCHAR(45)
Movie_ID INT
Indexes
User
Username VARCHAR(45)
First_Name VARCHAR(45)
Last_Name VARCHAR(45)
Credit_Card_No VARCHAR(45)
Billing Address VARCHAR(100)
Indexes

Tables

Table Queries and ER diagrams are accessible on the github link. All the tables are normalized up to 3NF. There is no multi-valued properties (1NF), No partial dependency (2NF) and no Transitional dependency (3NF). All the datatypes, constraints, triggers, indexes and many other details can be found in the file named. “Entire Code_Forward Engineering.docx”. Here just very briefly the structures of the tables are discussed.

Tables include: User, Account, Bill, Watches, Adds_to_Playlist, Report and last but not least, Movie.

User: First Name, Last Name, Username, and credit_card_no, Billing_Address

PK: Username

First_Name Not NULL

Last_Name Not NULL

Credit Card Not NULL, Unique

Account (weak entity): Account Number, Username, Password

Account Number is partial primary key
primary key = (Account Number, Username)

Foreign Key (Username) refers to User table (Username)

It is On Delete and Update Cascade

Constraint -> Check (Account_Number<=5)
Passwrod Not NULL, there must be a password

Bill: ID, start date, monthly_fee, Username (who pays)

This bill is monthly, so it does not have end_date to avoid transitional dependency.

Primary Key (ID)

Foreign Key (Username) refers to Account table (Username)

Monthly_Fee Not NULL

Start_Date Not NULL

This table is to keep track of watch history

Watches: Date, Account_Number, Username, Movie_ID

Primary Key (Date, Account_Number, Username, Movie_ID)

Foreign Key (Username) refers to Account table (Username)

Foreign Key (Account_Number) refers to Account table (Account_Number)

Foreign Key (Movie_Id) refers to Movie table (ID)

This table is the playlist

Adds_to_Playlist: Account_Number, Username, Movie_ID

Primary Key (Account_Number, Username, Movie_ID)

Foreign Key (Username) refers to Account table (Username)

Foreign Key (Account_Number) refers to Account table (Account_Number)

Foreign Key (Movie_Id) refers to Movie table (ID)

Report table is for reports in case of a problem, it can be sent by any account. Accounts are not required to generate reports and it is optional.

Report: Report_ID, Username, Description, Account_Number

Primary Key (Report_ID)

Foreign Key (Username) refers to Account table (Username)

Description Not NULL, there must be a description

Table Movie holds movie information, Table Deleted Movie has this same structure, used for logging.

Primary Key (ID)

Title Not NULL,

Check Rating<=10

Table Deleted_Credit_Card is used for logging and keeping track of credit cards used for the system in the past

Primary Key (Credit_Card)

First_Name Not NULL

Last_Name Not NULL

No foreign key

Stored Procedures

5 Stored Procedures are implemented. They are explained with codes and screenshots

Triggers and Logging

5 triggers are created. They can be found with great detail in a file named trigger.

The first 2 Triggers are used for logging. They move deleted movies into deleted_movie table.

Also they move credit card history of deleted users or updated credit cards.

SQL Codes

11 SQL codes are provided in a file with the same name. They include select, alter, adding constrants, transactions, creating view, creating index, copying table structure and many other things.

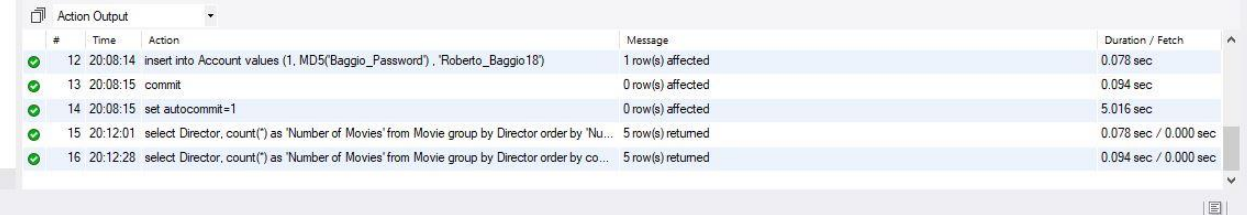
DB Connection

The DB was connected to python though mysql.connector and performed a select query that lists movies played by Brad Pitt. Python code and Connection screenshot are included.

Performance Measurement

Regarding Performance, screen shot of runtime for different queries is presented bellow.

The average time is around 1.07 s.



	#	Time	Action	Message	Duration / Fetch
✓	12	20:08:14	insert into Account values (1, MD5('Baggio_Password') , 'Roberto_Baggio18')	1 row(s) affected	0.078 sec
✓	13	20:08:15	commit	0 row(s) affected	0.094 sec
✓	14	20:08:15	set autocommit=1	0 row(s) affected	5.016 sec
✓	15	20:12:01	select Director, count(*) as 'Number of Movies' from Movie group by Director order by 'Nu...	5 row(s) returned	0.078 sec / 0.000 sec
✓	16	20:12:28	select Director, count(*) as 'Number of Movies' from Movie group by Director order by co...	5 row(s) returned	0.094 sec / 0.000 sec

At the end I'd like to thank our TA for helping a lot with this assignment.