Database Design Proposal

John Boyle

- Executive Summary 3
- ER Diagram 4
- Create Table 5-12
- Insert Into 13
- Views 14-15
- Sample Queries 16-17
- Triggers 18
- Security 19
- Implementation, problems, enhancements 20

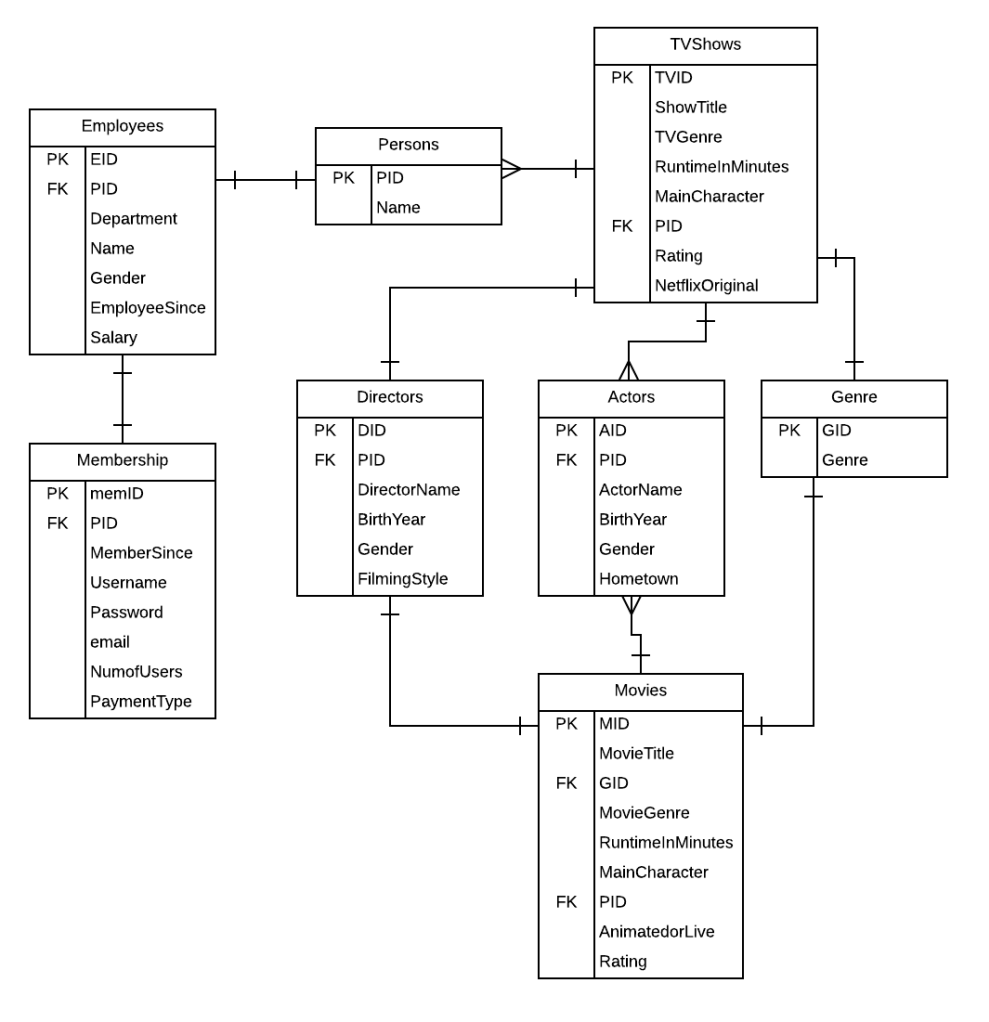
Table of contents

Executive Summary

This document is a representation of all of the data stored within the Netflix database. Netflix is a corporation that started off renting out movie DVD's and VHS tapes by sending them to the customers home to be returned by a certain date. However, by 2007 Netflix started to move into the direction of streaming movies from the convenience of the customers own home. In 2016, there was an estimated 98.75 million subscribers to Netflix, and this is continuously growing.

This database holds the information for all Movies, TV Shows, Actors, Membership and Employee's that are affiliated with the corporation. This database is specifically designed to hold large scale data, and to make it as simple as possible to navigate throughout the database.

There are some records within this database that are fictional, such as a persons name or Email, and there are some legitimate names in there as well.



ER Diagram

Create Table Statements: Persons

pid character	name character varying
P1	Charlie Cox
P2	Bryan Cranston
P3	Wentworth Miller
P4	Seth MacFarlane

The Persons table is a table that assigns a certain PID to every person within the database.

```
CREATE TABLE IF NOT EXISTS PERSONS
(
 PID CHAR(9) NOT NULL UNIQUE,
 NAME VARCHAR NOT NULL,
PRIMARY KEY(PID)
);
```

Functional Dependencies:
PID > Name

Create Table Statements: TVShows

tvid character	showtitle characte...	tvgenre characte...	runtimei... integer	maincharacter character varying	pid character	rating characte...	netflixori... boolean
t01	Daredevil	Thriller	60	Charlie Cox	P1	TV-MA	true
t02	Breaking ...	Thriller	50	Bryan Cranston	P2	TV-MA	false
t03	Prison Br...	Thriller	50	Wentworth Miller	P3	TV-14	false

The TVShows table has all of the generic parameters that a TV show pertains, such as Title, genre, rating, runtime, the main character and their PID, added is the true or false value of being a Netflix Original (a show that was created by Netflix). It also has a unique TID.

```
CREATE TABLE IF NOT EXISTS TVShows (
 TVID char(50) NOT NULL unique,
 ShowTitle VARCHAR (50) NOT NULL,
 TVGenre VARCHAR (10) NOT NULL,
 RuntimeInMinutes INTEGER NOT NULL,
 MainCharacter VARCHAR NOT NULL,
 PID CHAR(9) NOT NULL,
 Rating VARCHAR NOT NULL,
 NetflixOriginal BOOLEAN,
primary key(TVID),
Foreign key (PID) References Persons(PID)
);
```

Functional Dependencies:
TID > ShowTitle, Genre, Rating

Create Table Statements: Genre

gid character	genre characte...
g01	Comedy
g02	Thriller
g03	Cartoon

The Genre table is a simple table that has a unique GID that specifies what Genre the certain TV Show or movie is.

```
CREATE TABLE IF NOT EXISTS Genre (
 GID char (10) NOT NULL unique,
 Genre VARCHAR NOT NULL,
primary key(GID)
);
```

Functional Dependencies:
GID > Genre

Create Table Statements: Actors

aid character	pid character	actorname characte...	birthyear integer	gender characte...	hometown characte...
a01	P1	Charlie C...	1982	Male	London
a02	P2	Bryan Cr...	1956	Male	Hollywood
a03	P3	Wentwor...	1972	Male	Chipping ...

The Actors table contains information about actors that appear on Netflix shows such as: aID, Name, Birth Year, Gender, and their hometown.

```
CREATE TABLE IF NOT EXISTS Actors (
 AID char (5) NOT NULL unique,
 PID CHAR(9) NOT NULL,
 ActorName VARCHAR NOT NULL,
 BirthYear integer NOT NULL,
 Gender VARCHAR NOT NULL,
 Hometown VARCHAR NOT NULL,
primary key(AID),
foreign key (PID) References PERSONS(PID)
);
```

Functional Dependencies:
AID >ActorName, Gender

Create Table Statements: Directors

did character	pid character	directorname character varying	birthyear integer	gender characte...	filmingstyle character v...
d01	P21	Steve Hickner	1972	Male	Animation
d02	P22	Robert Zemeckis	1952	Male	Drama
d03	P23	Dennis Dugan	1946	Male	Comedy

The Directors table is similar to the Actors table with the information it presents, the DID, PID, Name, Birth Year, and Gender. This one includes the Director filming style.

```
CREATE TABLE IF NOT EXISTS Directors (
 DID char (5) NOT NULL UNIQUE,
 PID CHAR(9) NOT NULL,
 DirectorName VARCHAR NOT NULL,
 BirthYear integer NOT NULL,
 Gender VARCHAR NOT NULL,
 FilmingStyle VARCHAR NOT NULL,
primary key (DID),
foreign key (PID) References PERSONS(PID)     );
```

Functional Dependencies:
DID > DirectorName, Gender, BirthYear

Create Table Statements: Movies

mid character	movietitle characte...	gid character	moviege... characte...	runtimei... integer	maincharacter character varying	pid character	animate... characte...	rating characte...
m01	Bee Movie	g01	Comedy	95	Jerry Seinfeld	P11	Animated	PG
m02	Forrest G...	g06	Drama	144	Tom Hanks	P12	LiveAction	R
m03	The Benc...	g01	Comedy	98	Rob Schneider	P13	Live	PG-13

The Movies table containt information pertaining to certain Netflix movies, such as MID, Movie Title, genreID, genre, run time, the main character, etc…

```
CREATE TABLE IF NOT EXISTS Movies (
 MID char (5) NOT NULL unique,
 MovieTitle VARCHAR (50) NOT NULL,
 GID char (10) NOT NULL,
 MovieGenre VARCHAR (10) NOT NULL,
 RuntimeInMinutes INTEGER NOT NULL,
 MainCharacter VARCHAR NOT NULL,
 PID CHAR(9) NOT NULL,
 AnimatedOrLive VARCHAR NOT NULL,
 Rating VARCHAR NOT NULL,
primary key(MID),
foreign key (GID) References Genre(GID),
FOREIGN KEY(PID) REFERENCES PERSONS(PID)
);
```

Functional Dependencies:
MID > MovieTitle, RuntimeInMinutes, Rating

Create Table Statements: Membership

memid character	member... integer	username characte...	password characte...	email character varying	numofus... integer	payment... character
mem01	2009	database	alpaca	movie@gmail.com	3	Debit Card
mem02	2008	systems	bear	show@gmail.com	1	Check
mem03	2014	sql	fish	video@gmail.com	4	Cash

The Membership table has the information from a users account, such as their membership start year, username, password, email, how many users are on the account, and their payment type.

```
CREATE TABLE IF NOT EXISTS Membership (
 memID char (10) NOT NULL unique,
 MemberSince integer Not Null,
 Username VARCHAR NOT NULL unique,
 password varchar not null,
 email varchar not null unique,
 NumOfUsers integer not null,
 PaymentType char (15) not null,
primary key(memID) ,
Foreign key (PID) references Persons(PID)        );
```

Functional Dependencies:
memID > Username, Password, Email

Create Table Statements: Employees

eid character	pid character	department character ...	name character varying	gender characte...	employeesi... date	salary integer
e01	P31	CEO	Alan Labouseur	Male	2017-05-01	200000
e02	P32	Minion	Piradon Liengtiraphan	Male	2017-05-01	2
e03	P33	IT Admin	Casimer Decusatis	Male	2017-05-01	10000
e04	P34	Sales Rep	John Smith	Male	2007-06-12	14000

The employees table contains very basic information of the employees, like their department, name, and how long they've been working there.

```
CREATE TABLE IF NOT EXISTS Employees (
 EID char (10) not null unique,
 PID CHAR(9) NOT NULL,
 Department varchar not null,
 Name varchar not null,
 Gender varchar not null,
 EmployeeSince date not null,
 Salary integer not null,
primary key(EID),
FOREIGN KEY(PID) REFERENCES PERSONS(PID)
);
```

Functional Dependencies:
EID > Name, Department, EmployeeSince

Insert Into: TVShows table:

This is a sample of the **insert into** statements used for all of the tables.
You specify the columns in the first half of the statement, then state the field vlaues in the **values** section of the statement.

```
INSERT INTO TVShows
  ( TVID, ShowTitle, TVGenre, RuntimeInMinutes, MainCharacter, PID, Rating, NetflixOriginal )

VALUES
  ('t01', 'Daredevil', 'Thriller', 60, 'Charlie Cox', 'P1', 'TV-MA', 'True'),
 ……etc
```

Insert Statement

MovieInfo View

movietitle character varying	moviege... characte...	runtimei... integer	directorname character varyi...
The Benchwarmers	Comedy	98	Dennis Dugan

This view will show the selected movie information from any movie Directed by Dennis Dugan.

```
CREATE VIEW MovieInfo AS
Select M.MovieTitle, m.movieGenre,
m.RuntimeInMinutes, m.DirectorName
From Movies m, persons p
where m.pid = p.pid
and p.name = 'Dennis Dugan'
```

This will display the view, named MovieInfo:

```
select*
from MovieInfo
```

Views

DirectorName View:

directorn... character...	pid character	filmingst... character...
Ben Stiller	P29	Action

This view shows the director information who direct action movies that are 100+ minutes in length.

```
CREATE VIEW DirectorName AS
select d.directorname, d.pid, d.filmingstyle
from directors d, movies m
where d.directorname = m.directorname
and m.gid = 'g07'
and m.runtimeinminutes > 100;
```

```
select*
from directorname
```

Views

movietitle character varying	moviege... characte...	runtimei... integer
Forrest Gump	Drama	144
Goon	Comedy	101
Hannah Montana ...	Comedy	102
Tropic Thunder	Action	107
The Do-Over	Comedy	108

This is a simple query that displays the movies that are more than 100 mintes in length.

```
select movietitle, moviegenre,
runtimeinminutes
from movies
where runtimeinminutes > 100
```

Sample Queries

directorn... characte...	did character	birthyear integer
Steve Hic...	d01	1972
Dennis D...	d03	1946
Michael D...	d04	1973
David Zu...	d05	1947
Jim Fall	d06	1962
Peter Ch...	d07	1956
Steven Brill	d10	1962

This query shows the information about directors who have directed comedies.

```
select d.directorname, d.did, d.birthyear
from directors d inner join movies m on
d.directorname = m.directorname
where m.moviegenre = 'Comedy'
```

Sample Queries

```
CREATE OR REPLACE FUNCTION newUser() RETURNS trigger AS $$
BEGIN
IF NEW.memID is null THEN
                raise exception 'No pid';
END IF;
IF NEW.userName IS NULL THEN
                raise exception 'Please make a Username';
 END IF;
IF NEW.password IS NUL THEN
                raise exception 'Please set a password';
 END IF;
INSER INTO Membership(pID, MemberSince, UserName, password, email, NumofUsers, PaymentType)
values (NEW.memID, NEW.username, NEW.password, NEW.email ,NEW.numofusers, NEW.PaymentType);
RETURN new;   END;
$$ language plpgsql;
```

This trigger is for a new user is registered into the database.

Triggers

```
CREATE role CEOAlan
GRANT SELECT, INSERT, UPDATE,
DELETE
On all tables in schema public to CEOAlan
```

This security measure insures that only the CEO, Alan Labouseur, has permission to make any changes to the database.

Security

- This database was implemented with little to no problems.
- This database can be used to easily analyzed the data within the Netflix corporation, as it is constantly evolving and expanding.
- The actors, movies, and Tvshows tables can all be expanded, with numerous different parameters from the Tvshow can be added, as there are countless different operations and information utilized within.

Implementation, Problems, Enhancements