Homework 3, William Berg and Gustav Henningsson, group 104

1. Constraining the attributes.

In addition to the necessary constraints defined in homework 1 and 2, add the following constraints to your database.

b. Add support for age restricted movies and series.

To add constraints for age restricted movies and series we would first need a new *rated* attribute for the Media and TVSeries relations that signifies its age rating. We would also have to add an age or date of birth attribute to the UserProfiles relation so that the age can be checked against the new rated attribute of the movie or series that the user wants to watch. This feature will not be implemented in SQL, instead the frontend will handle this type of interaction.

1c. Add passwords that have to be at least 8 characters long and a maximum of 32 characters long.

We will create a password attribute for Administrators and Customers (both admins and customers will have a password for their account) with a check to make sure the length constraint is met. This will be done with the Postgres `char_length()` function:

```
ALTER TABLE Customers
ADD COLUMN password TEXT CHECK(char_length(password) >= 8 AND (char_length(password) <= 32));
```

2. NULL-values.

2a. You have identified and introduced constraints i.e. key constrains, unique constraints, check constraints on at least one relation in your database schema. What restrictions do each of these constraints enforce if the value of an attribute is NULL?

All primary keys are NOT NULL attributes since primary key attributes are NOT NULL and UNIQUE attributes by definition - this means that a NULL value for a primary key will throw an error. A foreign key is allowed to be NULL however. A column that enforces the UNIQUE constraint allows NULL values since NULL signifies the absence of a value and is therefore not considered to be equal to other NULL values. This is the case for PostgreSQL, but not the case for all other SQL dialects. A check constraint on its own always allows NULL values, since a check constraint is satisfied if the check expression evaluates to either true or NULL. A NOT NULL constraint would have to be added in order to prevent this.

2b. Identify at least one attribute in each of your database schemas that requires a NOT NULL constraint and explain why.

It is good design to make sure that some attributes can't be NULL. It would be reasonable for the name attribute in the Actors relation to have a NOT NULL constraint for example. Below is a table that shows each relation and a corresponding attribute in that relation that requires a NOT NULL constraint:

Relation	NOT NULL attribute
Actors	name
Administrators	Position
Card	ccn
Customers	email
Directors	name
Episodes	tvseries
Media	name
MediaActors	ALL
Series	franchise
SubscriptionDuration	ALL
SubscriptionPayment	ALL
SubscriptionPlan	customer
TVSeries	name

Relation	NOT NULL attribute
TVSeriesActors	ALL
UserProfile	customer
Watchlist	userprofile

The relations where ALL attributes must be NOT NULL are either relations that contain only two attributes that make up a composite primary key (primary keys are NOT NULL by default) or relations that contain one attribute which is a primary key and another that must be NOT NULL. Examples of the first kind of relation are: MediaActors and TVSeriesActors. Examples of the second kind of relation are: SubscriptionDuration and SubscriptionPayment (the other attribute in both of these relations are *derived* from the primary key).

3. Adding Views

Views in databases are predetermined SQL queries that display specific data in temporary tables. Your task is to create a few different views. Make sure that you depict your domains and describe what assumptions or choices you make for your view. The views you need to create are specified in question a and b:

3a. A movie-view that a user will use that provides information about one specified movie.

We decided that the information to be displayed along with the movie should be its year of release (INT), name (TEXT), director (TEXT), genre (TEXT), country (TEXT), length in minutes (NUMERIC), and rating (NUMERIC). These are taken from the Media relation and the Directors relation.

```
/* Creating the view that provides information about the specified movie */
CREATE VIEW get_movie_info AS (
    SELECT media.name, directors.name AS director, genre, year, country, length, rating
    FROM Media
    JOIN Directors ON director = directorID
    /* Specified movie */
    WHERE media.name = 'Lord of the Rings: The Fellowship of the Ring'
);
```

3b. A view that displays all the movies released within the current week.

To make this possible we added a new column to the Media relation called dateAdded that tells us when a movie was added to the MSS database. The code can be seen below:

```
/* Adding dateAdded attribute */
ALTER TABLE Media
ADD COLUMN dateAdded DATE;

/* Creating the view to display movies that were added since the first day of this week */
CREATE VIEW current_week AS (
    SELECT name, year, dateAdded
    FROM Media
    WHERE dateAdded >= date_trunc('week', CURRENT_DATE)
);
```

The `current_week` view displays the name (TEXT) of the movie, the year of release (INT), and the date it was added to the database - dateAdded (DATE). The year of release it to distinguish between the original movie and its remake. The view works by only displaying movies where the dateAdded is larger than or equal to the first day of the current week. This functionality is achieved with the built in `date_trunc('datepart', field)` function where the datepart specifies the interval for truncation and the field specifies what to truncate.