КУРСОВОЙ ПРОЕКТ

Основы реляционных баз данных. MySQL

Аннотация

Скрипты создания и наполнения базы данных, ER диаграмма, скрипты выборок, представления, хранимые процедуры, триггеры.

Екатерина Грищенкова

https://github.com/helloworld755

ВВЕДЕНИЕ

Для успешного функционирования любых организаций и сервисов требуется наличие развитой информационной системы, которая реализует автоматизированный сбор, обработку и манипулирование данными. Современной формой информационных систем являются банки данных, включающие в свой состав вычислительную систему, систему управления базами данных (СУБД), одну или несколько баз данных (БД), набор прикладных программ (приложений БД). БД обеспечивает хранение информации, а также удобный и быстрый доступ к данным. БД представляет собой совокупность данных различного характера, организованных по определенным правилам.

В данной работе представлена разработка базы данных для сервиса Netflix. Это позволяет управлять данными о фильмах, сериалах, актерах, пользователях, лайках и списках к просмотру, а также выполнять анализ данных на основе формирования выборок и представлений.

1 ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Проектирование баз данных представляет собой процесс создания схемы базы данных и определения необходимых ограничений целостности. Основными его задачами являются: обеспечение хранения в БД всей необходимой информации, обеспечение возможности получения данных по всем необходимым запросам, сокращение избыточности и дублирования данных, обеспечение целостности данных (исключение противоречий в содержании данных, исключение их потери и т.д.).

Этапы проектирования базы данных можно представить следующим образом:

1. Определение цели создания базы данных.
2. Определение таблиц, которые должна содержать база данных.
3. Определение необходимых в таблице полей.
4. Задание первичного ключа для каждой таблицы.
5. Определение связей между таблицами.
6. Обновление структуры базы данных.
7. Добавление данных и создание других объектов базы данных.

На первом этапе проектирования базы данных необходимо определить цель создания базы данных, основные ее функции и информацию, которую она должна содержать, т.е. нужно определить основные темы таблиц базы данных и информацию, которую будут содержать поля таблиц.

База данных должна отвечать требованиям тех, кто будет непосредственно с ней работать. Для этого нужно определить темы, которые должна покрывать база данных, отчеты, которые она должна выдавать, проанализировать формы, которые в настоящий момент используются для записи данных, сравнить создаваемую базу данных с хорошо спроектированной, подобной ей базой.

Одним из наиболее сложных этапов в процессе проектирования базы данных является разработка таблиц, так как результаты, которые должна выдавать база данных (отчеты, выходные формы и др.) не всегда дают полное представление о структуре таблицы.

При проектировании таблиц необходимо руководствоваться следующими основными принципами:

- Информация в таблице не должна дублироваться (не должно быть повторений и между таблицами). Это делает работу более эффективной, а также исключает возможность несовпадения информации в разных таблицах.
- Сведения на каждую тему обрабатываются намного легче, если содержатся в независимых друг от друга таблицах. Каждая таблица содержит информацию на отдельную тему, а каждое поле в таблице содержит отдельные сведения по теме таблицы.

При разработке полей для каждой таблицы необходимо учитывать:

- Каждое поле должно быть связано с темой таблицы.
- Не рекомендуется включать в таблицу данные, являющиеся результатом выражения.
- В таблице должна присутствовать вся необходимая информация.
- Информацию следует разбивать на наименьшие логические единицы.

С тем чтобы СУБД могла связать данные из разных таблиц каждая таблица должна содержать поле или набор полей, которые будут однозначно идентифицировать каждую запись в таблице. Такое поле или набор полей называют первичным ключом.

После распределения данных по таблицам и определения ключевых полей необходимо определить связи между таблицами. Связи нужны для того, чтобы обеспечить синхронное изменение одноименных полей в разных таблицах. Самый распространенный вид связи – «один ко многим».

После проектирования таблиц, полей и связей необходимо еще раз просмотреть структуру базы данных и выявить возможные недочеты. Желательно это сделать на данном этапе, пока таблицы не заполнены данными.

Для проверки необходимо ввести несколько записей в каждую таблицу и посмотреть, отвечает ли база данных поставленным требованиям. Рекомендуется также создать черновые выходные формы и отчеты и проверить, выдают ли они

требуемую информацию. Кроме того, необходимо исключить из таблиц все возможные повторения данных.

Если структуры таблиц отвечают поставленным требованиям, то можно вводить все данные. Затем можно создавать любые запросы, выборки, представления и т.д.

2 СОЗДАНИЕ БАЗЫ ДАННЫХ

В рамках данного проекта были поставлены следующие требования:

1. Составить общее текстовое описание БД и решаемых ею задач;
2. минимальное количество таблиц - 10;
3. скрипты создания структуры БД (с первичными ключами, индексами, внешними ключами);
4. создать ERDiagram для БД;
5. скрипты наполнения БД данными;
6. скрипты характерных выборок (включающие группировки, JOIN'ы, вложенные таблицы);
7. представления (минимум 2);
8. хранимые процедуры / триггеры.

Далее в соответствующих разделах будет приведено решение и описание каждого пункта.

2.1 Текстовое описание базы данных и решаемых ею задач

В данном проекте разрабатывается база данных для сервиса Netflix. База данных предназначена для управления информацией о фильмах и сериалах, актерах, пользователях, лайках и списках просмотра.

Netflix – это один из ведущих в мире развлекательных сервисов с более чем 209 миллионами платных подписок (пользователей) в более чем 190 странах, где можно смотреть сериалы, документальные и художественные фильмы в разных жанрах и на разных языках. Пользователи могут смотреть сколько угодно, в любое время, в любом месте, на любом подключенном к Интернету устройстве.

Логически база данных разделена на три части:

1. Информация о фильмах и сериалах.
2. Информация об актерах и их ролях.
3. Информация о пользователях, их лайках и списках к просмотру.

Связи между разделами и таблицами:

1. Есть общий список всех видеоматериалов сервиса Netflix. В нем информация по каждому фильму и сериалу (в целом, без конкретики по сезонам и сериям).
2. Запись в общей таблице с видеоматериалами типа «сериал» ведет к записям в таблице с сезонами, а оттуда – к записям в таблице с сериями.
3. Есть таблица с ролями, которая ссылается на id фильма или сериала, а также на id актера.
4. Есть таблица с пользователями Netflix. Она имеет связь с лайками этих пользователей и списками к дальнейшему просмотру.

2.2 Таблицы базы данных

В базе данных Netflix находятся следующие таблицы:

1.	Genres (id, genre_name)	Таблица с жанрами (комедия, драма, боевик, исторический и т.д.)
2.	All (id, type, name, description, genre_id, rating)	Таблица с перечнем всех фильмов и сериалов на Netflix. Она включает id, тип («фильм» или «сериал»), название, описание, id жанра (ссылка на таблицу с жанрами) и рейтинг на IMDb.
3.	Movies (id, all_id, videofile)	Таблица с перечнем всех фильмов, их id и видеофайлами. Она включает поле all_id, которое ссылается на запись в общей таблице All.
4.	TV_seasons (id, all_id, season_name, season_description)	Таблица с перечнем всех сезонов сериалов, их id, названиями и описанием. Она включает поле all_id, которое ссылается на запись в общей таблице All.
5.	TV_episodes (id, season_id, episode_name, episode_description, videofile)	Таблица с перечнем всех эпизодов сериалов, их id, названиями, описанием и самим видеофайлом серии. Она включает поле season_id, которое ссылается на запись в таблице TV_seasons.
6.	Actors (id, firstname, lastname, country, picture)	Таблица с перечнем всех актеров, их id, именами, страной и фотографией.
7.	Cast (id, all_id, actor_id, role, type_of_role)	Таблица с перечнем ролей актеров в фильмах и сериалах. Она включает id роли, ссылку на фильм или сериал (в таблице All), ссылку на id актера, название роли и ее тип (постоянная или временная).
8.	Users (id, firstname, lastname, country, email, password, phone)	Таблица с перечнем пользователей Netflix. Она включает их id, имена, страны, email, пароль и телефон.
9.	Likes (id, user_id, all_id, like_type)	Таблица со всеми лайками или дизлайками пользователей. Она включает id оценки, id пользователя, id фильма или сериала и тип оценки (лайк или дизлайк).
10.	Watch_later (id, user_id, all_id)	Таблица с перечнем к дальнейшему просмотру. Содержит id записи, id пользователя и id видеоматериала.

2.3 Скрипты создания структуры БД

Data Definition Language (DDL) – это группа операторов определения данных. Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы, т.е. создаем, изменяем и удаляем их.

В эту группу входят следующие операторы:

- CREATE – используется для создания объектов базы данных;
- ALTER – используется для изменения объектов базы данных;
- DROP – используется для удаления объектов базы данных.

Далее приведен код с комментариями.

```
DROP DATABASE IF EXISTS netflix;
CREATE DATABASE netflix;
USE netflix;
```

Создание базы данных

```
DROP TABLE IF EXISTS genres;
CREATE TABLE genres(
      id SERIAL,
    genre_name VARCHAR(50)
) COMMENT 'Жанр – комедия, драма, боевик,
документальный и т.д.';
```

Создание таблицы Genres
SERIAL – это
BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE;
Для имени жанра выбран тип VARCHAR, 50 символов.

```
DROP TABLE IF EXISTS `all`;
CREATE TABLE `all` (
      id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    `type` VARCHAR(30) COMMENT ‘Фильм или сериал’,
    name VARCHAR(255),
    description TEXT,
    genre_id BIGINT UNSIGNED NOT NULL,
    rating FLOAT,

    FOREIGN KEY (genre_id) REFERENCES genres(id)
) COMMENT 'ID всех фильмов и сериалов';
```

Создание таблицы All:
id фильма или сериала;
тип;
название фильма или сериала;
описание (тип данных TEXT, так как описание может быть длинным);
id жанра;
рейтинг.

Внешний ключ – связь поля genre_id с полем id из таблицы Genres

```
DROP TABLE IF EXISTS movies;
CREATE TABLE movies(
      id SERIAL,
      all_id BIGINT UNSIGNED NOT NULL,
      videofile TEXT
) COMMENT ‘Отдельный фильм’;
```

Создание таблицы Movies:
id фильма в таблице Movies;
id фильма в таблице All;
видеофайл (тип Text, потому что здесь будет ссылка).

```
ALTER TABLE movies ADD CONSTRAINT video_movie
    FOREIGN KEY (all_id) REFERENCES `all`(id);
```

Команда ALTER TABLE применяется в SQL при добавлении, удалении либо модификации колонки в существующей таблице.
ADD CONSTRAINT – создание ограничения. Здесь связь поля all_id с полем id из таблицы All.

```sql
DROP TABLE IF EXISTS TV_seasons;
CREATE TABLE TV_seasons(
	id SERIAL,
	all_id BIGINT UNSIGNED NOT null,
	season_name VARCHAR(100),
	season_description TEXT
) COMMENT 'Отдельный сезон, views – среднее по всем
эпизодам';
```

Создание таблицы TV_seasons:
id сезона;
id сериала из таблицы All;
название сезона;
описание сезона.

```sql
ALTER TABLE TV_seasons ADD CONSTRAINT show_season
    FOREIGN KEY (all_id) REFERENCES `all`(id);
```

Связь поля all_id с полем id из таблицы All.

```sql
DROP TABLE IF EXISTS TV_episodes;
CREATE TABLE TV_episodes(
	id SERIAL,
	season_id BIGINT UNSIGNED NOT NULL,
	episode_name VARCHAR(100),
	episode_description TEXT,
	videofile TEXT
) COMMENT ‘Отдельный эпизод’;
```

Создание таблицы TV_episodes:
id серии;
id сезона из таблицы TV_seasons;
название серии;
описание серии;
видеофайл серии.

```sql
ALTER TABLE TV_episodes ADD CONSTRAINT season_episode
    FOREIGN KEY (season_id) REFERENCES
TV_seasons(id);
```

Связь поля season_id с полем id из таблицы TV_seasons.

```sql
DROP TABLE IF EXISTS actors;
CREATE TABLE actors (
	id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    firstname VARCHAR(70),
    lastname VARCHAR(70),
    country VARCHAR(70),
	picture TEXT,

    INDEX actor_firstname_lastname_idx(firstname,
lastname)
) COMMENT ‘Актеры’;
```

Создание таблицы Actors:
id актера;
имя актера;
фамилия актера;
страна актера;
фото актера.

Формирование индекса по имени и фамилии актера.

```sql
DROP TABLE IF EXISTS `cast`;
CREATE TABLE `cast` (
	id SERIAL,
	all_id BIGINT UNSIGNED NOT NULL,
	actor_id BIGINT UNSIGNED NOT NULL,
	`role` VARCHAR(70),
	type_of_role VARCHAR(255)
) COMMENT 'Список ролей актеров в фильмах';
```

Создание таблицы Cast:
id роли;
id фильма или сериала;
id актера;
роль;
тип роли (постоянная, временная, камео и т.д.).

```sql
ALTER TABLE `cast` ADD CONSTRAINT cast_video
    FOREIGN KEY (all_id) REFERENCES `all`(id);
```

Связь поля all_id с полем id таблицы All.

```sql
ALTER TABLE `cast` ADD CONSTRAINT cast_actor
    FOREIGN KEY (actor_id) REFERENCES actors(id);
```

Связь поля actor_id с полем id таблицы Actors.

```sql
DROP TABLE IF EXISTS users;
CREATE TABLE users (
	id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    firstname VARCHAR(70),
    lastname VARCHAR(70),
    country VARCHAR(70),
    email VARCHAR(120) UNIQUE,
	`password` VARCHAR(100),
	phone BIGINT UNSIGNED UNIQUE,

    INDEX user_firstname_lastname_idx(firstname,
lastname)
) COMMENT ‘Пользователи Netflix’;
```

Создание таблицы Users:
id пользователя;
имя пользователя;
фамилия пользователя;
страна пользователя;
email;
пароль;
телефон.

Формирование индекса по имени и фамилии пользователя.

```sql
DROP TABLE IF EXISTS likes;
CREATE TABLE likes (
	id SERIAL,
	user_id BIGINT UNSIGNED NOT NULL,
	all_id BIGINT UNSIGNED NOT NULL,
	like_type CHAR(1) comment ‘l/d’,

	PRIMARY KEY (user_id, all_id)
) COMMENT ‘Пользователи Netflix’;
```

Создание таблицы Likes:
id лайка/дизлайка;
id пользователя;
id фильма или сериала;
вид оценки – лайк или дизлайк (l или d).

Первичный ключ по id пользователя и фильма/сериала.

```sql
ALTER TABLE likes ADD CONSTRAINT like_user
FOREIGN KEY (user_id) REFERENCES users(id);
```

Связь поля user_id с полем id таблицы users.

```sql
ALTER TABLE likes ADD CONSTRAINT like_video
FOREIGN KEY (all_id) REFERENCES `all`(id);
```

Связь поля all_id с полем id таблицы All.

```sql
DROP TABLE IF EXISTS watch_later;
CREATE TABLE watch_later (
	id SERIAL,
	user_id BIGINT UNSIGNED NOT NULL,
	all_id BIGINT UNSIGNED NOT NULL,

	PRIMARY KEY (user_id, all_id)
) COMMENT 'Отмеченное для просмотра позже
пользователем';
```

Создание таблицы Watch_later:
id добавления элемента в список;
id пользователя;
id фильма/сериала.

Первичный ключ по id пользователя и фильм/сериала.

```sql
ALTER TABLE watch_later ADD CONSTRAINT watch_user
FOREIGN KEY (user_id) REFERENCES users(id);
```

Связь поля user_id с полем id таблицы users.

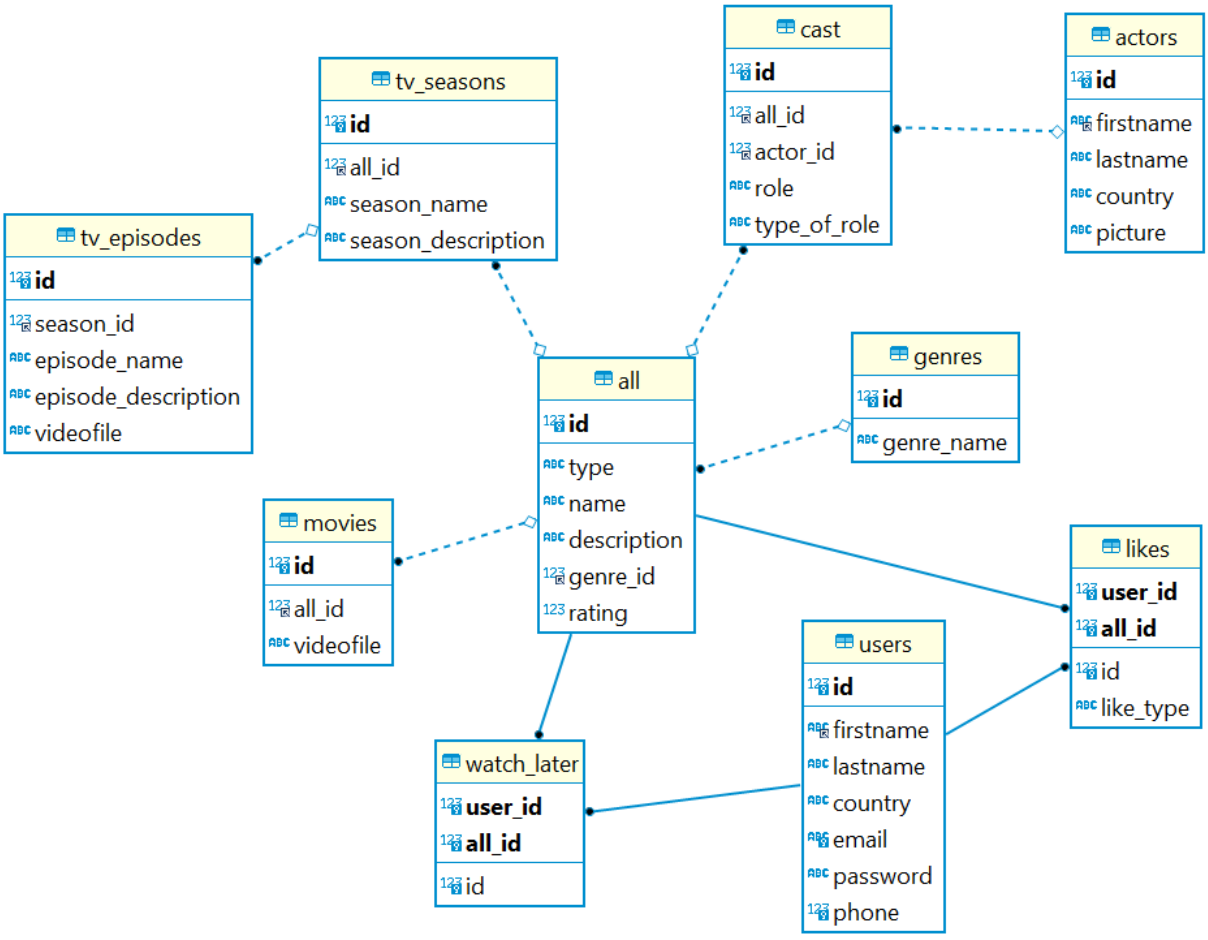
```sql
ALTER TABLE watch_later ADD CONSTRAINT watch_video
FOREIGN KEY (all_id) REFERENCES `all`(id);
```

Связь поля all_id с полем id таблицы All.

2.4 ERDiagram

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

В данном разделе приведена ER диаграмма базы данных Netflix.



2.5 Скрипты наполнения БД

Data Manipulation Language (DML) – это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы, т.е. манипулировать ими.

В эту группу входят самые распространённые операторы языка SQL:

- SELECT – осуществляет выборку данных;
- INSERT – добавляет новые данные;
- UPDATE – изменяет существующие данные;
- DELETE – удаляет данные.

Далее приведен код с комментариями.

```
INSERT INTO genres(genre_name)
VALUES ('Comedy'), ('History'), ('Drama'),
('Action'), ('Fantasy');
```

Ввод данных в таблицу Genres.

```
INSERT INTO `all`(`type`, name, description,
genre_id, rating)
VALUES ('TV Show', 'How I Met Your Mother', 'Some
text', '1', '8.3'),
            ('Movie', 'Pride and Prejudice', 'Some
text', '2', '7.8'),
            ('Movie', 'Harry Potter', 'Some text',
'5', '7.6'),
            ('TV Show', 'The Crown', 'Some text',
'2', '8.6'),
            ('Movie', 'Quantum of Solace', 'Some
text', '4', '6.6'),
            ('Movie', 'The Mask', 'Some text', '1',
'6.9'),
            ('Movie', 'Pirates of Caribbean', 'Some
text', '5', '8.0'),
            ('TV Show', 'Black Mirror', 'Some text',
'3', '8.8'),
            ('TV Show', 'Sherlock', 'Some text',
'4', '9.1'),
            ('Movie', 'Me Before You', 'Some text',
'3', '7.4');
```

Ввод данных в таблицу All.

```
INSERT INTO movies(all_id, videofile)
VALUES ('2', 'link'),
            ('3', 'link'),
            ('5', 'link'),
            ('6', 'link'),
            ('7', 'link'),
            ('10', 'link');
```

Ввод данных в таблицу Movies.

```
INSERT INTO TV_seasons(all_id, season_name,
season_description)
VALUES ('1', 'Season 1', 'Some text'),
            ('8', 'Season 2', 'Some text'),
            ('4', 'Season 2', 'Some text'),
            ('1', 'Season 2', 'Some text'),
            ('1', 'Season 3', 'Some text'),
            ('8', 'Season 1', 'Some text'),
            ('4', 'Season 1', 'Some text'),
            ('9', 'Season 1', 'Some text'),
            ('4', 'Season 3', 'Some text'),
            ('8', 'Season 3', 'Some text');
```

Ввод данных в таблицу TV_seasons.

```
INSERT INTO tv_episodes(season_id, episode_name,
episode_description, videofile)
VALUES ('1', 'Episode 6', 'Text', 'link'),
            ('2', 'Episode 3', 'Text', 'link'),
            ('4', 'Episode 10', 'Text', 'link'),
            ('6', 'Episode 15', 'Text', 'link'),
            ('9', 'Episode 2', 'Text', 'link'),
            ('3', 'Episode 8', 'Text', 'link'),
            ('3', 'Episode 1', 'Text', 'link'),
            ('5', 'Episode 4', 'Text', 'link'),
            ('6', 'Episode 9', 'Text', 'link'),
            ('2', 'Episode 5', 'Text', 'link');
```

Ввод данных в таблицу TV_episodes.

```
insert into actors(firstname, lastname, country,
picture)
values ('Daniel', 'Radcliffe', 'UK', 'link'),
            ('Jim', 'Carrey', 'USA', 'link'),
            ('Keira', 'Knightley', 'UK', 'link'),
            ('Benedict', 'Cumberbatch', 'UK',
'link'),
            ('Neil Patrick', 'Harris', 'USA',
'link');
```

Ввод данных в таблицу Actors.

```
INSERT INTO `cast`(all_id, actor_id, `role`,
type_of_role)
VALUES ('2', '3', 'Elizabeth Benneth', 'Main'),
            ('9', '4', 'Sherlock Holmes', 'Main'),
            ('3', '1', 'Harry Potter', 'Main'),
            ('6', '2', 'Stanley Ipkiss', 'Main'),
            ('1', '5', 'Barney Stinson', 'Main');
```

Ввод данных в таблицу Cast.

```
INSERT INTO users(firstname, lastname, country,
email, `password`, phone)
VALUES ('Andy', 'Johnson', 'USA', 'andy@gmail.com',
'jhbdjfghdb', '12345'),
            ('Mary', 'Denver', 'UK',
'mary@gmail.com', 'dfsdhfskdhf', '23456'),
            ('Cody', 'Clinton', 'Germany',
'cody@gmail.com', 'jdygfvidyf', '34567'),
            ('Tracy', 'McConnell', 'USA',
'tracy@gmail.com', 'djyvgudyfgs', '45678'),
            ('Linda', 'Yesman', 'France',
'linda@gmail.com', 'jwgfjygdfjg', '56789');
```

Ввод данных в таблицу Users.

```
INSERT INTO likes(user_id, all_id, like_type)
VALUES ('3', '2', 'l'),
            ('4', '6', 'l'),
            ('1', '3', 'd'),
            ('3', '9', 'd'),
            ('2', '1', 'l'),
            ('5', '4', 'l'),
            ('1', '6', 'd'),
            ('3', '5', 'd');
```

Ввод данных в таблицу Likes.

```
INSERT INTO watch_later(user_id, all_id)
VALUES ('2', '3'),
            ('4', '1'),
            ('2', '8'),
            ('1', '4');
```

Ввод данных в таблицу Watch_later.

2.6 Скрипты характерных выборок

В данном разделе будут продемонстрированы различные решения для формирования характерных выборок с комментариями.

```
select `type`, count(*) as `number` from `all`
group by `type`;
```

Подсчет количества фильмов и сериалов в таблице All.
Count(*) – функция подсчета
Group by – группировка.

```
select * from `all`
order by rating desc
limit 3;
```

Выбор трех объектов из таблицы All с наивысшим рейтингом. Для этого выполняется упорядочение по рейтингу и ограничение 3 элементов.

```
select avg(rating) from `all`
where `type` = 'TV Show';
```

Вычисление среднего значения рейтинга только среди сериалов в таблице All.

```
select country as list_of_countries from actors
group by country;
```

Определение перечня стран, актеры из которых занесены в таблицу Actors.

```
select `type`, name, max(rating) from `all`
where `type` = 'Movie'
union
select `type`, name, max(rating) from `all`
where `type` = 'TV Show';
```

Выбор лидирующего фильма и лидирующего сериала по рейтингу из таблицы All.
Используется union, чтобы объединить запросы.

```
select a.firstname, a.lastname, c.`role`, al.name from
actors a
join
(select `role`, actor_id, all_id from `cast`) c
on a.id = c.actor_id
join
(select name, id from `all`) al
on al.id = c.all_id;
```

Таблица с именами актеров, ролями и сериалами/фильмами, в которых они эти роли сыграли.
Используется join.

```
select u.country as 'gave likes' from users u
join
(select like_type, user_id from likes where like_type
= 'l') l
on l.user_id = u.id;
```

Таблица, показывающая то, пользователи каких стран поставили лайки.

```
select a.name, ts.season_name, te.episode_name from
`all` a
join
(select id, all_id, season_name from tv_seasons) ts
on ts.all_id = a.id
join
(select season_id, episode_name from tv_episodes) te
on te.season_id = ts.id
order by a.name asc;
```

Вывод всех фильмов и сериалов с обозначением жанра.

Вывод названий сериалов, сезонов и эпизодов, которые хранятся в базе данных Netflix.

2.7 Представления

Представление – это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из ряда именованных столбцов и строк данных. Пока представление не будет проиндексировано, оно не существует в базе данных как хранимая совокупность значений. Строки и столбцы данных извлекаются из таблиц, указанных в определяющем представление запросе и динамически создаваемых при обращениях к представлению.

Далее приведены примеры представлений для базы данных Netflix.

```
create view view_1 as
select u.firstname, u.lastname, a.name from users u
join watch_later wl
on wl.user_id = u.id
join `all` a
on a.id = wl.all_id;

select * from view_1;
```

Создание представления, содержащего имена пользователей и фильмы или сериалы, которые они отметили, чтобы позже посмотреть.

```
create view view_2 as
select g.genre_name, a.name, a.rating from `all` a
join genres g
on a.genre_id = g.id
order by g.genre_name asc;


select * from view_2;
```

Создание представления, содержащего название жанра, название фильма/сериала и рейтинг.

2.8 Хранимые процедуры и триггеры

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Триггер – хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определённом столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Далее приведены примеры хранимой процедуры и триггера для базы данных Netflix.

```
drop procedure if exists TV_Characters;
delimiter //
create procedure TV_Characters()
begin
	select a.name as 'Movie or TV Show', c.`role` as
'Character', ac.firstname as 'Actor Name', ac.lastname
as 'Actor Lastname' from `all` a
	join `cast` c on c.all_id = a.id
	join actors ac on ac.id = c.actor_id
	where a.`type` = 'TV Show';
end //
delimiter ;

call TV_Characters;
```

Процедура, которая находит героев сериалов и актеров, которые их сыграли.

```
drop trigger if exists new_user_null;
delimiter //
create trigger new_user_null
before insert on users
for each row
begin
	if(isnull(new.firstname)
	or isnull(new.lastname)
	or isnull(new.country)
	or isnull(new.email)
	or isnull(new.password)
	or isnull(new.phone))
	then
	signal sqlstate '45000' set message_text = 'Null
is not allowed!';
end if;
end //
delimiter ;

INSERT INTO netflix.users(firstname, lastname,
country, email, `password`, phone)
VALUES  (null,  'Davidson',  'USA',  'morty@gmail.com',
'123', '123456');
```

Триггер, который не позволяет ввести значение null в любое из полей при добавлении записи.

ЗАКЛЮЧЕНИЕ

В результате работы была создана и наполнена база данных для сервиса Netflix, позволяющая осуществлять работу с данными о фильмах, сериалах, актерах, пользователях, лайках и т.д. Пользователь имеет возможность организовать БД, просмотреть ее содержание, изменить ее состав, добавить, удалить записи БД, создать собственные таблицы и представления, сгруппировать записи, создать хранимые процедуры и триггеры.