САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

---

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе № 2

Дисциплина: «Базы данных»

Тема: «Язык SQL-DDL»

Выполнил студент гр. 43501/3 _____ А.Ю. Ламтев
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

«___» _____ 2018 г.

Санкт-Петербург
2018

# Содержание

# 1. Цели работы

Познакомиться с основами проектирования схемы БД, языком описания сущностей и ограничений БД SQL-DDL.

# 2. Программа работы

1. Самостоятельное изучение SQL-DDL.
2. Создание скрипта БД в соответствии с согласованной схемой. Должны присутствовать первичные и внешние ключи, ограничения на диапазоны значений. Демонстрация скрипта преподавателю.
3. Создание скрипта, заполняющего все таблицы БД данными.
4. Выполнение SQL-запросов, изменяющих схему созданной БД по заданию преподавателя. Демонстрация их работы преподавателю.

# 3. Создание скрипта БД в соответствии с согласованной схемой

На рис. 3.1 изображена согласованная схема БД.

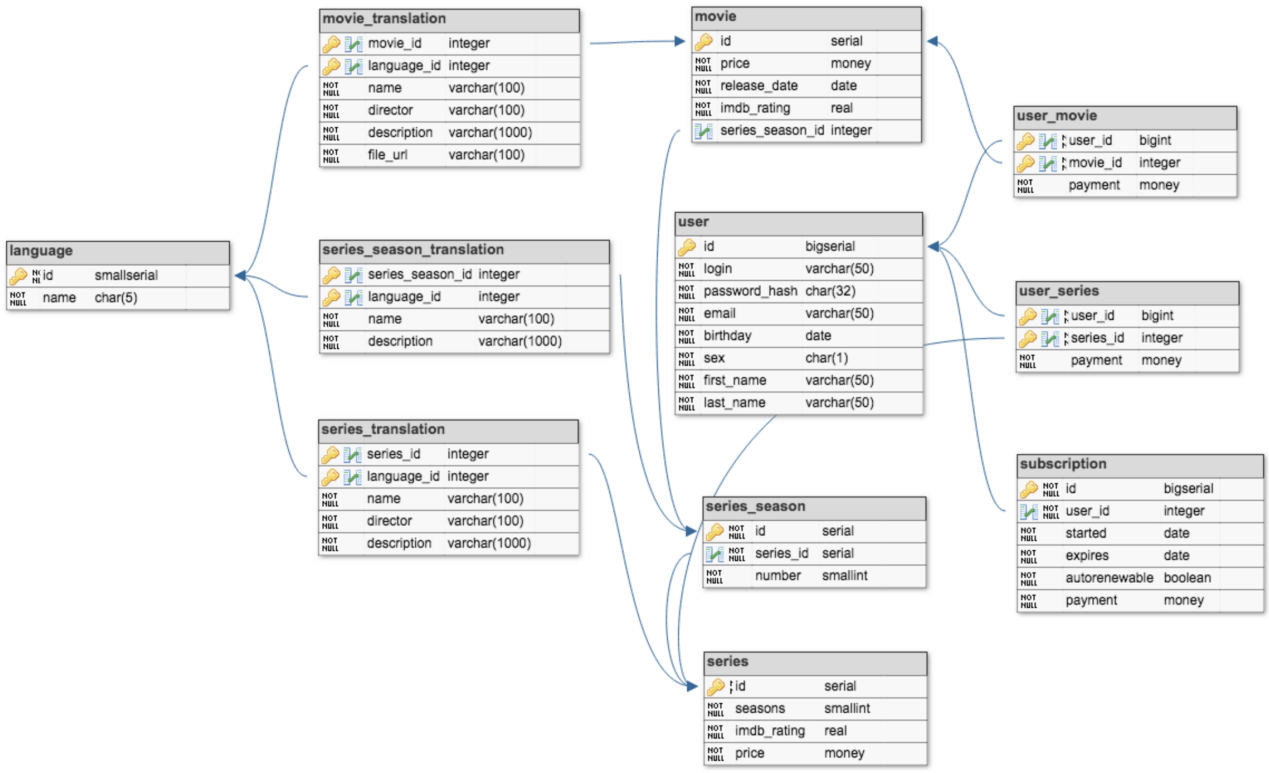


Рис. 3.1: Согласованная схема БД

В листинге 1 представлен скрипт, создающий БД в соответствии с согласованной схемой, изображённой на рис. 4.1.

```sql
CREATE TABLE IF NOT EXISTS language
(
  id    SERIAL PRIMARY KEY,
  name CHAR(5) NOT NULL UNIQUE
);


CREATE TABLE IF NOT EXISTS series
(
  id       SERIAL PRIMARY KEY,
  seasons SMALLINT NOT NULL,
  price    MONEY    NOT NULL
);


CREATE TABLE IF NOT EXISTS series_season
(
  id         SERIAL PRIMARY KEY,
  series_id INTEGER  NOT NULL,
  number     SMALLINT NOT NULL,

  FOREIGN KEY (series_id) REFERENCES series (id)
);


CREATE TABLE IF NOT EXISTS series_translation
(
  series_id    INTEGER        NOT NULL,
  language_id INTEGER        NOT NULL,
  name         VARCHAR(100)  NOT NULL,
  director     VARCHAR(100)  NOT NULL,
  description VARCHAR(1000) NOT NULL,

  CONSTRAINT series_translation_pk PRIMARY KEY (series_id,
   language_id),
  FOREIGN KEY (series_id) REFERENCES series (id),
  FOREIGN KEY (language_id) REFERENCES language (id)
);


CREATE TABLE IF NOT EXISTS series_season_translation
(
  series_season_id INTEGER        NOT NULL,
  language_id      INTEGER        NOT NULL,
  name             VARCHAR(100)  NOT NULL,
  description      VARCHAR(1000) NOT NULL,

  CONSTRAINT series_season_translation_pk PRIMARY KEY (
   series_season_id, language_id),
  FOREIGN KEY (series_season_id) REFERENCES series_season (id),
  FOREIGN KEY (language_id) REFERENCES language (id)
```

```
);


CREATE TABLE IF NOT EXISTS movie
(
  id                SERIAL PRIMARY KEY,
  price             MONEY NOT NULL,
  release_date      DATE  NOT NULL,
  imdb_rating       REAL  NOT NULL,
  series_season_id INTEGER,

  FOREIGN KEY (series_season_id) REFERENCES series_season (id)
);


CREATE TABLE IF NOT EXISTS movie_translation
(
  movie_id     INTEGER        NOT NULL,
  language_id INTEGER        NOT NULL,
  name         VARCHAR(100)  NOT NULL,
  director     VARCHAR(100)  NOT NULL,
  description VARCHAR(1000) NOT NULL,
  file_url     VARCHAR(100)  NOT NULL,

  CONSTRAINT movie_translation_pk PRIMARY KEY (movie_id, language_id
    ),
  FOREIGN KEY (movie_id) REFERENCES movie (id),
  FOREIGN KEY (language_id) REFERENCES language (id)
);

CREATE TABLE IF NOT EXISTS "user"
(
  id              BIGSERIAL PRIMARY KEY,
  login           VARCHAR(50) NOT NULL UNIQUE,
  password_hash CHAR(32)     NOT NULL,
  email           VARCHAR(50) NOT NULL UNIQUE,
  birthday        DATE        NOT NULL,
  sex             CHAR(1)     NOT NULL,
  first_name      VARCHAR(50) NOT NULL,
  last_name       VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS user_movie
(
  user_id  BIGINT  NOT NULL,
  movie_id INTEGER NOT NULL,
  payment  MONEY   NOT NULL,

  CONSTRAINT user_movie_pk PRIMARY KEY (user_id, movie_id),
  FOREIGN KEY (user_id) REFERENCES "user" (id),
  FOREIGN KEY (movie_id) REFERENCES movie (id)
);
```

```sql

CREATE TABLE IF NOT EXISTS user_series
(
  user_id    BIGINT  NOT NULL,
  series_id  INTEGER NOT NULL,
  payment    MONEY   NOT NULL,

  CONSTRAINT user_series_pk PRIMARY KEY (user_id, series_id),
  FOREIGN KEY (user_id) REFERENCES "user" (id),
  FOREIGN KEY (series_id) REFERENCES series (id)
);


CREATE TABLE IF NOT EXISTS subscription
(
  id             SERIAL PRIMARY KEY,
  user_id        BIGINT  NOT NULL,
  started        DATE    NOT NULL,
  expires        DATE    NOT NULL,
  autorenewable BOOLEAN NOT NULL,
  payment        MONEY   NOT NULL
);
```

Листинг 1: Скрипт, создающий БД в соответствии с согласованной схемой

Затем заполним созданную базу данными. Скрипт заполнения приведён в листинге 2.

```sql
INSERT INTO language (name)
VALUES ('en-US');

INSERT INTO "user" (login, password_hash, email, birthday, sex,
    first_name, last_name)
VALUES ('login', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ123456', 'email@email.
    email', '1984-11-12', '0', 'Firstname', 'Lastname');

INSERT INTO series (seasons, price)
VALUES (2, 25);

INSERT INTO series_season (series_id, number)
VALUES (1, 1),
       (1, 2);

INSERT INTO series_translation (series_id, language_id, name,
    director, description)
VALUES (1, 1, 'Silicon Valley', 'Some Director', 'Cool series');

INSERT INTO series_season_translation (series_season_id, language_id
    , name, description)
VALUES (1, 1, 'First season', 'Very cool season'),
       (2, 1, 'Second season', 'Cooler season');
```

```
20
21 INSERT INTO movie (price, release_date, imdb_rating,
       series_season_id)
22 VALUES (5, '2018-03-18', 9.0, 1),
23        (5, '2018-03-18', 8.0, 1),
24        (5, '2018-03-18', 9.5, 2),
25        (5, '2018-03-18', 8.5, 2),
26        (7, '2012-03-04', 8.0, NULL);
27
28 INSERT INTO movie_translation (movie_id, language_id, name, director
       , description, file_url)
29 VALUES (1, 1, 'Episode 1', 'Some Director', 'First episode', 'https
       ://url.to.video.com/file1'),
30        (2, 1, 'Episode 2', 'Some Director', 'First episode', 'https
       ://url.to.video.com/file2'),
31        (3, 1, 'Episode 1', 'Some Director', 'First episode', 'https
       ://url.to.video.com/file3'),
32        (4, 1, 'Episode 1', 'Some Director', 'First episode', 'https
       ://url.to.video.com/file4'),
33        (5, 1, 'Departures', 'Martin Scorsese', 'Cool movie', 'https
       ://url.to.video.com/file5');
34
35 INSERT INTO user_movie (user_id, movie_id, payment)
36 VALUES (1, 5, 7);
37
38 INSERT INTO subscription (user_id, started, expires, autorenewable,
       payment)
39 VALUES (1, '2018-12-22', '2019-12-22', TRUE, 15),
40        (2, '2018-12-19', '2019-06-19', TRUE, 15);
```

Листинг 2: Скрипт, заполняющий базу данными

## 4. Изменение схемы БД

Заданием было внести изменения в схему БД для удовлетворения следующим требованиям:

1. Ввести возможность подписок на часть фильмов/сериалов.
2. Реализовать каталог фильмов и сериалов по различным категориям: жанры, новинки и пр. Каждый фильм может быть отнесен к нескольким категориям. Категории могут иметь иерархию, но при этом фильмы могут относиться к узлу любого уровня иерархии.

В соответствии с заданием схема БД была изменена, и она приняла следующий вид:

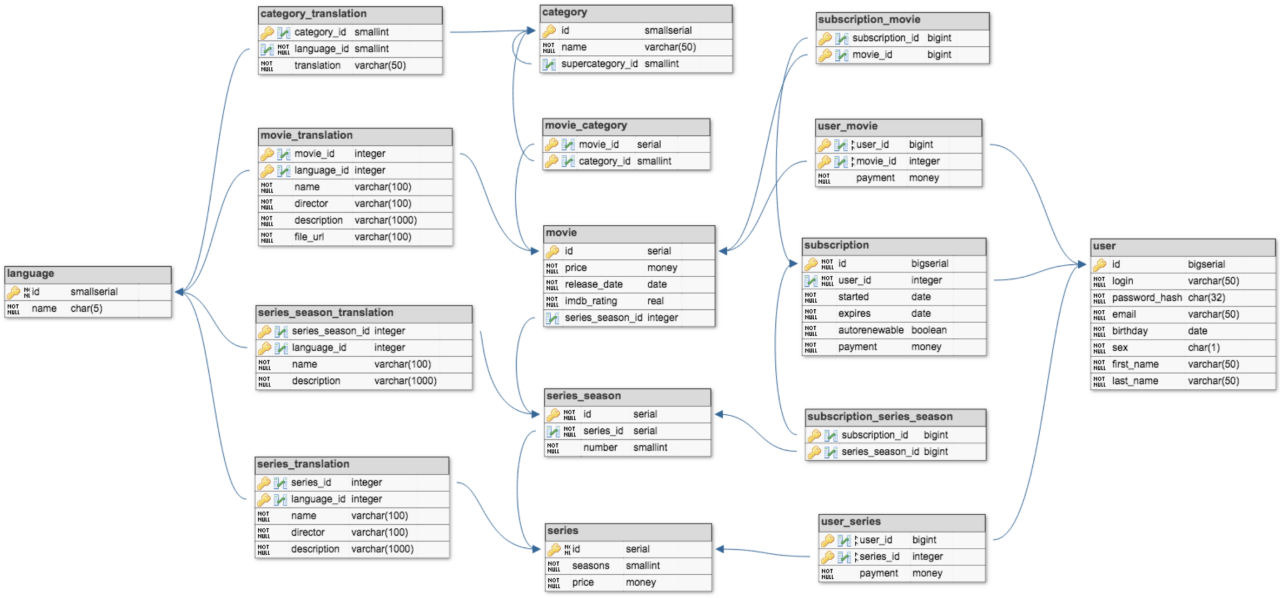


Рис. 4.1: Изменённая схема БД

В сравнении с предыдущей версией были добавлены следующие таблицы: `subscription_movie`, `subscription_series_season`, `category`, `movie_category` и `category_translation`.

В листинге 3 представлен скрипт, вносящий изменения в БД в соответствии с новой схемой, изображённой на рис. 4.1, а также заполняющий базу новыми данными.

```
CREATE TABLE IF NOT EXISTS subscription_movie
(
  subscription_id BIGINT  NOT NULL,
  movie_id        INTEGER NOT NULL,

  CONSTRAINT subscription_movie_pk PRIMARY KEY (subscription_id,
   movie_id),
  FOREIGN KEY (subscription_id) REFERENCES subscription (id),
  FOREIGN KEY (movie_id) REFERENCES movie (id)
);


INSERT INTO subscription_movie (subscription_id, movie_id)
VALUES (1, 5);


CREATE TABLE IF NOT EXISTS subscription_series_season
(
  subscription_id  BIGINT  NOT NULL,
  series_season_id INTEGER NOT NULL,

  CONSTRAINT subscription_series_season_pk PRIMARY KEY (
   subscription_id, series_season_id),
  FOREIGN KEY (subscription_id) REFERENCES subscription (id),
  FOREIGN KEY (series_season_id) REFERENCES series_season (id)
```

```
);


INSERT INTO subscription_series_season (subscription_id,
    series_season_id)
VALUES (2, 1);


CREATE TABLE IF NOT EXISTS category
(
  id                SERIAL PRIMARY KEY,
  name              VARCHAR(50) NOT NULL,
  supercategory_id SMALLINT,

  FOREIGN KEY (supercategory_id) REFERENCES category (id)
);


INSERT INTO category (name, supercategory_id)
VALUES ('genre', NULL),
       ('comedy', 1),
       ('thriller', 1),
       ('drama', 1),
       ('criminal', 1);


CREATE TABLE IF NOT EXISTS movie_category
(
  movie_id    INTEGER  NOT NULL,
  category_id SMALLINT NOT NULL,

  CONSTRAINT movie_category_pk PRIMARY KEY (movie_id, category_id),
  FOREIGN KEY (movie_id) REFERENCES movie (id),
  FOREIGN KEY (category_id) REFERENCES category (id)
);


INSERT INTO movie_category (movie_id, category_id)
VALUES (1, 2),
       (2, 2),
       (3, 2),
       (4, 2),
       (5, 3),
       (5, 4),
       (5, 5);


CREATE TABLE IF NOT EXISTS category_translation
(
  category_id SMALLINT    NOT NULL,
  language_id SMALLINT    NOT NULL,
  translation VARCHAR(50) NOT NULL,
```

```
75
76    CONSTRAINT category_translation_pk PRIMARY KEY (category_id,
       language_id),
77    FOREIGN KEY (category_id) REFERENCES category (id),
78    FOREIGN KEY (language_id) REFERENCES language (id)
79 );
80
81 INSERT INTO category_translation (category_id, language_id,
       translation)
82 VALUES (1, 1, 'genre'),
83        (2, 1, 'comedy'),
84        (3, 1, 'thriller'),
85        (4, 1, 'drama'),
86        (5, 1, 'criminal');
```

Листинг 3: Скрипт, вносящий изменения в БД

## 5. Выводы

В результате работы было проведено ознакомление с языком SQL-DDL, был разработан скрипт, создающий базу данных в соответствии с выбранной ранее моделью данных. Затем было осуществлено изменение модели данных и был разработан скрипт, обновляющий базу данных в соответствии с новой схемой.