

# Образовательная экосистема GeekBrains

## КУРСОВАЯ РАБОТА (КР)

**Направление:** Geek University Python-разработка  
(шифр и название направления подготовки/специальности)

**Курс:** Основы реляционных баз данных. MySQL  
(наименование образовательной программы)

**Тема КР:** БД «Кинопоиск». Superlite version

**Выполнил:** \_\_\_\_\_ Колокольчиков В.Д.  
(подпись) (ФИО выпускника)

Москва, 2021 г.

## ОГЛАВЛЕНИЕ

Оглавление .....	2
Ведение .....	3
Работы с сервером. Подключение к БД .....	4
Сбор данных для БД.....	5
Собственно БД .....	7
Представления. Процедуры и запросы .....	9
Представления .....	9
Процедуры .....	9
Запросы .....	11
Заключение .....	13
Список литературы .....	14

## ВЕДЕНИЕ

Темой курсовой работы в рамках курса основ по изучению реляционных баз данных (MySQL) была выбрана эко-система кино-портала «Кинопоиск». Курсовая писалась не правах рекламы, а для учебных целей. Данные о фильмах брались из открытых источников.

Для написания данной КР использовалось следующее ПО:

1. MySQL – version 8.0
2. Dbeaver
3. Python – version 3.9.0
4. PyCharm
5. Oracle VM
6. Ubuntu Desktop (для сервера)

Библиотеки, задействованные в ходе работы:

1. Для парсинга сайтов – Requests и BS4 (BeautifulSoup)
2. Для работы с таблицами Excel – xlrd, xlwt
3. Для удаленной работы с БД посредством Python – pymysql и sshunnel

Шаблоны кодов, а также основные запросы, триггеры и т.п. документы связанные с MySQL будут находиться в репозитории.

## РАБОТЫ С СЕРВЕРОМ. ПОДКЛЮЧЕНИЕ К БД

Подключение к базе данных производилось из двух программ DBeaver, как основа для редакции структуры БД и создания запросов, и PyCharm для парсинга данных с последующей их загрузкой на сервер (Рис. 1). Подключался через SSH-подключение. Предварительно пришлось отредактировать конфигурационный файл – .my.cnf .

```
ssh_host='192.168.1.7'
server = SSHTunnelForwarder(
    (ssh_host, 22),
    ssh_username=ssh_user,
    ssh_password=ssh_pass,
    remote_bind_address=("127.0.0.1", 3306))

server.start()

with connect(user='root',
            password=password,
            host="127.0.0.1",
            port=server.local_bind_port,
            database='kinopoisk') as connection:
    # insert_reviewers_query = """
    # INSERT INTO films
    # (name, description, country, genres_type_id, creators_id)
    # VALUES (%s, %s, %s, %s, %s) """
```

Рисунок 1. Подключение к БД с помощью python

## СБОР ДАННЫХ ДЛЯ БД

Данные собирались с двух ресурсов: генератора данных (<http://www.generatedata.com/#generator>) и сайта Википедия ([https://ru.wikipedia.org/wiki/250\\_лучших\\_фильмов\\_по\\_версии\\_IMDb](https://ru.wikipedia.org/wiki/250_лучших_фильмов_по_версии_IMDb)) для основной информации по кино и его создателям. Для основы БД взял таблицу топ-250 фильмов. Таблицу из википедии забрал с помощью специальных функций в Excel (Рис. 2). Оставил для работы только 100 строк.

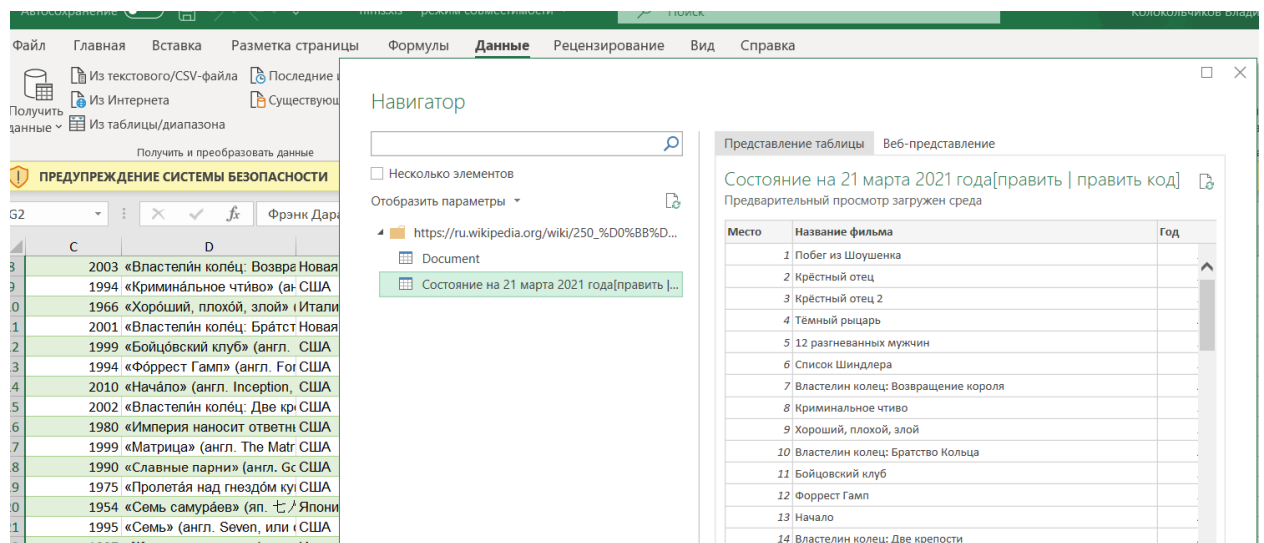


Рисунок 2. Основа для личной БД

Те данные, которых не было в полученной таблице, пришлось собирать с помощью парсинга страниц в Википедии. Настроил программу так, чтобы она переходила самостоятельно по нужным мне ссылкам и забирала нужную для меня информацию (имена режиссеров, сценаристов, актеров, страна выпуска, описание к фильмам). Пример кода для сбора описаний будет доступен в репозитории.

Для увязки данных была написана также программа, которая сначала преобразовала массив данных во множество, а после в словарь (ключ – значение). На основе последнего выстраивалась связь таблиц между собой.

После данных подгружались также с помощью python и библиотеки pymysql (Рис. 3).

```

with connect(user='root',
             password=password,
             host="127.0.0.1",
             port=server.local_bind_port,
             database='kinopoisk') as connection:
    # insert_reviewers_query = """
    # INSERT INTO films
    # (name, description, country, genres_type_id, creators_id)
    # VALUES (%s, %s, %s, %s, %s) """
    for inx, el in enumerate(sheet.col_values(5)):
        print(inx+1, el)
        update_query = """
        UPDATE
        |     films
        SET
        |     release_date = "%s"
        WHERE
        |     id = "%s"
        """
        val_tup = (int(el), inx+1)












        with connection.cursor() as cursor:
            cursor.execute(update_query, val_tup)
            connection.commit()

```

Рисунок 3. Закомментированная часть добавляла данные для таблицы *films*. Часть вне комментария исправляла дату релиза (точнее изначальное его отсутствие)

## СОБСТВЕННО БД

В ходе работы над БД kinopoisk было создано 11 таблиц (Рис. 4). Все таблицы созданы на движке InnoDB. БД в настоящее время имеет следующую структуру:

Название	Движок	Авто-увеличение	Размер	О
 actors	<a href="#">InnoDB</a>	169	16K	
 creators	<a href="#">InnoDB</a>	101	16K	
 directors	<a href="#">InnoDB</a>	66	16K	
 films	<a href="#">InnoDB</a>	101	112K	
 genres_types	<a href="#">InnoDB</a>	17	16K	
 likes	<a href="#">InnoDB</a>	101	16K	
 media	<a href="#">InnoDB</a>	101	16K	
 plans	<a href="#">InnoDB</a>	101	16K	
 profiles	<a href="#">InnoDB</a>	0	16K	
 scriptwriters	<a href="#">InnoDB</a>	89	16K	
 users	<a href="#">InnoDB</a>	101	16K	

После добавления всех данных таблицу была выстроена связь между ними, а также проиндексированные столбцы, которые не попали в число уникальных или не стали ключами. Для таких таблиц как profiles, directors, actors, scriptwriter и media были созданы триггеры либо редакции данных (как в первых 4), либо для автозаполнения (как это было с таблицей медиа). Все триггеры и индексы будут представлены в отдельном файле с соответствующими им названиями.

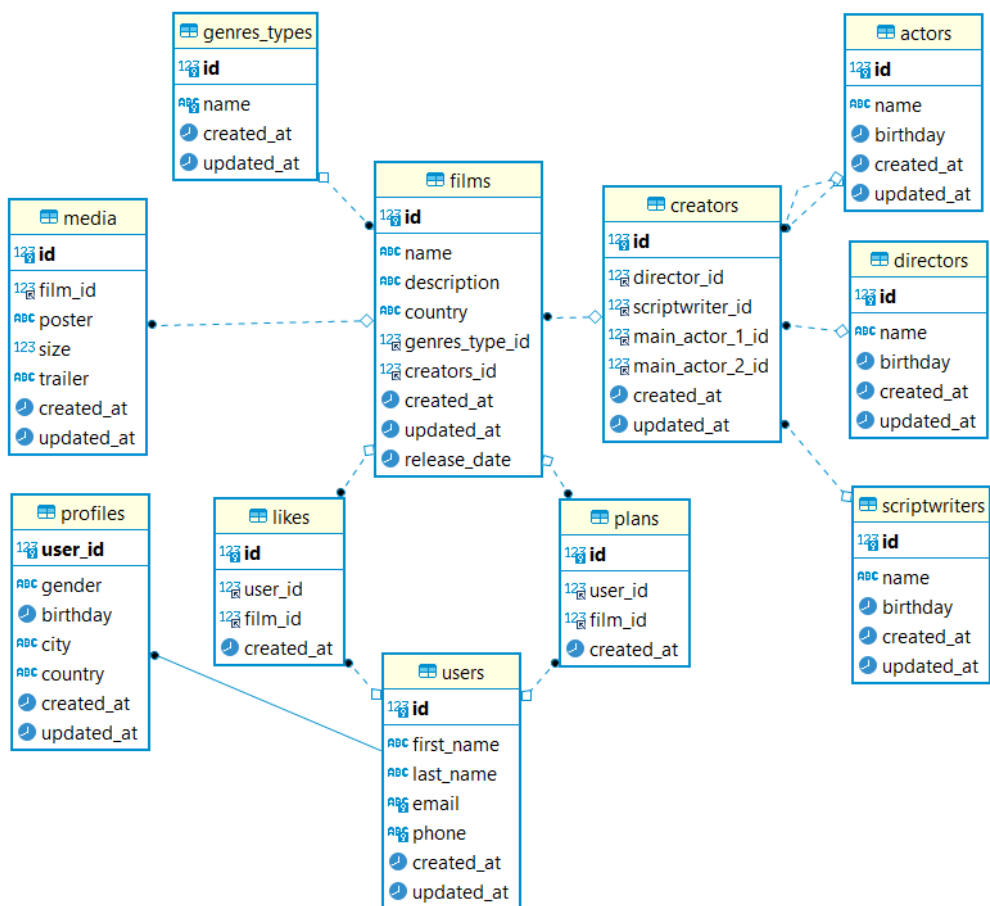


Рисунок 4. ERDiagrama для БД



## ПРЕДСТАВЛЕНИЯ. ПРОЦЕДУРЫ И ЗАПРОСЫ

### Представления

Для БД kinopoisk были созданы два представления, которые предоставляли пользователям информацию о последних фильмах (новинках кино) и информацию о создателях. Ознакомиться с ними можно в отдельном файле с соответствующим названием. Пример такого запроса с новинками кино представлен на рисунке 5.

<pre>-- Новинки кино в БД  CREATE OR REPLACE VIEW new_films AS SELECT f.name as 'Фильм', f.description as 'Описание', d.name as 'Режиссер', f.release_date FROM films f , creators c JOIN directors d ON d.id = c.director_id WHERE f.creators_id = c.id AND f.release_date &gt; 2018 ORDER BY f.release_date DESC WITH CHECK OPTION;</pre>			
abc Фильм	abc Описание	abc Режиссер	release_date
1	Гамильтон	История первого министра финансов США Александра Гамильтона на фоне реальных исторических событий: Войны за независимость, основан	2020
2	Мстители: Финал	«Мстители: Финал» (англ. Avengers: Endgame) — американский супергеройский фильм 2019 года киностудии Marvel Studios, срежиссированный Брат	2019
3	Джокер	«Джокер» (англ. Joker) — американский психологический триллер режиссёра Тодда Филлипса по сценарию, написанному Филлипсом совместно	2019
4	Паразиты	«Паразиты» (кор. 기생충) — южнокорейский комедийно-драматический фильм с элементами триллера режиссёра Пон Чжун Хо, получивший	2019

Рисунок 5. Новинки кино (2021 год пока еще не в топе =)

### Процедуры

В процедурах были реализованы рекомендации по фильмам для пользователей. Первая была основана на жанрах тех, фильмов, которые были «лайкнуты», исключая просмотренные из списка вывода. Вторая была построена аналогичным образом, но уже по режиссерам. Ознакомиться с ними подробно можно в отдельном файле. Примеры хранимой процедуры представлен на рисунке 6.

```
DROP PROCEDURE IF EXISTS recomend_films_by_genre;
```

```
DELIMITER //
CREATE PROCEDURE recomend_films_by_genre (pid VARCHAR(10))
BEGIN
    SELECT f2.id, f2.name FROM films f2 WHERE genres_type_id IN (SELECT DISTINCT f.genres_type_id FROM users u
    JOIN likes l
    ON l.user_id = u.id
    JOIN films f
    ON f.id = l.film_id
    WHERE u.id = pid) AND id NOT IN (SELECT DISTINCT f.id FROM users u
    JOIN likes l
    ON l.user_id = u.id
    JOIN films f
    ON f.id = l.film_id
    WHERE u.id = pid);
END//
DELIMITER ;
```

```
CALL recomend_films_by_genre(60);
```

```
CALL recomend_films_by_director(88);
```

ms 1

CALL recomend\_films\_by\_genre(60) | Введите SQL выражение чтобы отфильтровать результаты

id	name
2	Крёстный отец (фильм)
3	Крёстный отец 2
17	Славные парни
20	Семь
22	Город Бога
23	Молчание ягнят
27	Зелёная миля
31	Леон
33	Подозрительные лица
43	Отступники
51	Окно во двор
54	Помни
60	Джокер
66	Свидетель обвинения
83	Рай и ад
89	Бешеные псы
92	Головокружение
93	М

Рисунок 6. Вывод хранимой процедуры. Рекомендации для пользователя с ид 60.

## Запросы

Были сформированы наиболее интересные запросы (на мой взгляд и что пришло в голову) разных типов: вложенные, с JOIN и оконные. Со всеми запросами можно ознакомиться в соответствующем файле. Примеры некоторых будут представлены на рисунках 7 и 8.

-- Любимые фильмы пользователей

```
SELECT DISTINCT COUNT(l2.user_id) OVER w as top_likes,  
f.name as 'Название фильма', f.release_date as 'Дата релиза',  
a2.name as 'Главный актер 1', a3.name as 'Главный актер 2'  
FROM (films f  
JOIN likes l2  
ON l2.film_id = f.id  
JOIN creators c2  
ON c2.id = f.creators_id  
JOIN actors a2  
on a2.id = c2.main_actor_1_id  
JOIN actors a3  
ON a3.id = c2.main_actor_2_id)  
WINDOW w AS (PARTITION BY l2.film_id) ORDER BY top_likes DESC LIMIT 10;
```

	top_likes	Название фильма	Дата релиза	Главный актер 1	Главный актер 2
1	3	Новые времена	1936	Чарли Чаплин	Полетт Годдар
2	3	Рай и ад	1963	Тосиро Мифунэ	Тацуя Накадай
3	3	Храброе сердце	1995	Мел Гибсон	Софи Марсо
4	3	Унесённые призраками	2001	Мию Ирино	Руми Хиираги
5	3	Огни большого города	1931	Чарльз Чаплин	Вирджиния Черрилл
6	3	Доктор Стрейнджлав, или Как я перестал бояться и полюбил бомбу	1964	Питер Селлерс	Джордж К. Скотт
7	3	Крёстный отец 2	1974	Аль Пачино	Роберт Де Ниро
8	3	Тропы славы	1957	Кирк Дуглас	Джордж Макреди
9	2	Чужой	1979	Сигурни Уивер	Том Скерритт
10	2	Престиж	2006	Хью Джекман	Кристиан Бейл

Рисунок 7. Любимые фильмы пользователей

```
-- Главные любители КИНО
SELECT u.id as id , CONCAT(u.first_name, ' ', u.last_name) as name,
COUNT(film_id) as likes FROM likes l
RIGHT JOIN users u
ON u.id = l.user_id
GROUP BY name, id ORDER BY likes DESC LIMIT 5;
```

	id	name	likes
1	88	Brendan Douglas	6
2	79	Colin Huff	4
3	89	Cole Mills	4
4	30	Patrick Cole	3
5	6	Pamela Joyner	3

Рисунок 8. Главные любители кино

## ЗАКЛЮЧЕНИЕ

Была проделана большая работа по сбору данных и по её дальнейшей обработке. Лично для меня данная работа принесла огромное количество опыта, который, как мне кажется, может пригодиться для дальнейшей работы. Возможно, таблица получилось не такой хорошей и корректной, однако во всяком случае я доволен ей. Прежде всего тем, что смог применить знания Python для обработки данных что заняло у меня много дней и ночей. В общем сделал все возможное, чтобы сделать курсовую работу более интересное и менее шаблонной.

Ждем оценки, результатов и мнений. Спасибо за знания!

## **СПИСОК ЛИТЕРАТУРЫ**

1. Портал GeekBrains
2. Википедия
3. CodeRoad
4. Proglib
5. Habr
6. Tproger