**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«РУТ (МИИТ)»**

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

КУРСОВОЙ ПРОЕКТ

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«Фильмы»**

Выполнил: ст. гр. ТКИ-441  
Лукьянов Д.В.  
Проверил: Васильева М.А.

Москва 2023

Оглавление

[ЦЕЛЬ КУРСОВОГО ПРОЕКТА 5](#_Toc136252795)

[ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ 5](#_Toc136252796)

[ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 6](#_Toc136252797)

[1 Инфологическое проектирование 6](#_Toc136252798)

[1.1 Анализ предметной области 6](#_Toc136252799)

[1.2 Анализ информационных задач и круга пользователей системы 7](#_Toc136252800)

[2 Логическое проектирование реляционной БД 8](#_Toc136252801)

[2.1 Составление реляционных отношений 8](#_Toc136252802)

[2.2 Нормализация полученных отношений 11](#_Toc136252803)

[3 Физическое проектирование БД 12](#_Toc136252804)

[3.1 Разработка скриптов на создание базы данных и таблиц 12](#_Toc136252805)

[3.2 Разработка скриптов на добавление данных в таблицы 15](#_Toc136252806)

[3.3 Разработка необходимых запросов, представлений (view), процедур, функций и триггеров 17](#_Toc136252807)

[Список литературы 21](#_Toc136252808)

# ЦЕЛЬ КУРСОВОГО ПРОЕКТА

Целью курсового проекта является изучение методов и закрепление знаний в проектировании реляционных баз данных (РБД) в системе управления базами данных (СУБД) Postgres Pro.

# ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ

В данном курсовом проекте ставится задача разработать РБД в СУБД Postgres Pro по заданной теме «Фильмы». Проектирование РБД проводится с помощью метода «Сущность-связь». Проверка построенной модели РБД осуществляется с помощью метода нормализации отношений.

Описание предметной области.

БД создается для информационного обслуживания сайта с фильмами и обслуживания посетителей.

Ниже приведено описание основных сущностей предметной области:

1. Таблица "movies", которая содержит информацию о фильмах, включая их id, название, описание, дату выхода, продолжительность и рейтинг.
2. Таблица "genres" содержит информацию о жанрах, включая их id и название.
3. Таблица "countries" содержит информацию о странах, их id и названии.
4. Таблица "directors" содержит информацию о режиссерах, их id, имени и фамилии.
5. Таблица "actors" содержит информацию об актерах, включая id, имя и фамилию.
6. Таблица "movie\_genres" содержит информацию об фильмах и жанрах.
7. Таблица "movie\_countries" содержит информацию об фильмах и странах.
8. Таблица "movie\_directors" содержит информацию об фильмах и режиссерах.
9. Таблица "movie\_actors" содержит информацию об фильмах и актерах.

# ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

## Инфологическое проектирование

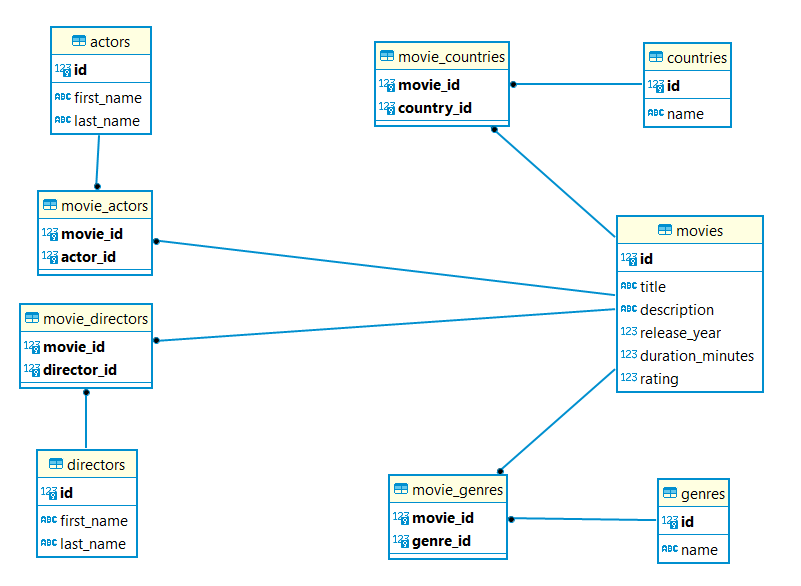
### Анализ предметной области

База данных создаётся для сайта с музыкой.

Выделим базовые сущности этой предметной области:

* movies – фильмы.
* genres – жанры фильмов.
* countries – страны производства.
* directors - режиссеры.
* actors - актеры.

ER-диаграмма БД, приведена на (Рисунке 1). На данной ER-диаграмме присутствуют связи многие - ко - многим и один - ко - многим.



*Рисунок 1 - Уточненная ER-диаграмма*

В полученной ER-диаграмме присутствуют только связи многие к многим. Такие связи легко реализуются средствами СУБД.

### Анализ информационных задач и круга пользователей системы

Система создаётся для обслуживания следующих групп пользователей:

* Администраторы
* Посетители

Определим границы информационной поддержки пользователей:

**1) Функциональные возможности:**

* ведение БД (запись, чтение, модификация, удаление в архив);
* обеспечение логической непротиворечивости БД;
* обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
* реализация наиболее часто встречающихся запросов в готовом виде.

**2) Готовые запросы:**

* выбрать названия всех фильмов;
* выбрать фильм с id 2;
* выбрать фильмы, которые длиннее 120 минут;
* выбрать жанры, которые присутствуют в более чем двух фильмах.

## Логическое проектирование реляционной БД

### Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту ПрО), и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи (если они есть). В том случае, если базовое отношение не имеет потенциальных ключей, вводится *суррогатный первичный ключ*, который не несёт смысловой нагрузки и служит только для идентификации записей.

movie\_genres: отношение "многие ко многим" между фильмами и жанрами. Один фильм может быть связан с несколькими жанрами, и один жанр может быть связан с несколькими фильмами.

movie\_countries: отношение "многие ко многим" между фильмами и странами. Один фильм может быть связан с несколькими странами, и одна страна может быть связана с несколькими фильмами.

movie\_directors: это отношение "многие ко многим".

movie\_actors: отношение "многие ко многим" между фильмами и актёрами. Один фильм может иметь много актёров, и один актёр может играть во многих фильмах.

Таким образом, были определены связи между сущностями, которые позволят хранить информацию и связывать данные между собой.

**Таблица 1 – Схема отношения** **movies**

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | title | Строковый тип | Обязательное поле |
| Описание | description | Текст | Обязательное поле |
| Год выхода | release\_year | Целое число | Обязательное поле |
| Продолжительность | duration\_minutes | Целое число | Обязательное поле |
| Рейтинг | rating | Число с плав. точ. | Обязательное поле |

Таблица 2 – Схема отношения genres

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | name | Строковый тип | Обязательное поле |

**Таблица 3 – Схема отношения** **countries**

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | name | Строковый тип | Обязательное поле |

**Таблица 4 – Схема отношения** **directors**

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Имя | first\_name | Строковый тип | Обязательное поле |
| Фамилия | last\_name | Строковый тип | Обязательное поле |

Таблица 5 – Схема отношения actors

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Имя | first\_name | Строковый тип | Обязательное поле |
| Фамилия | last\_name | Строковый тип | Обязательное поле |

Таблица 6 – Схема отношения movie\_genres

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | movie\_id | Целое число | Первичный ключ |
| Первичный ключ | genre\_id | Целое число | Первичный ключ |

Таблица 7 – Схема отношения movie\_countries

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | movie\_id | Целое число | Первичный ключ |
| Первичный ключ | country\_id | Целое число | Первичный ключ |

Таблица 8 – Схема отношения movie\_directors

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | movie\_id | Целое число | Первичный ключ |
| Первичный ключ | director\_id | Целое число | Первичный ключ |

Таблица 9 – Схема отношения movie\_actors

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | movie\_id | Целое число | Первичный ключ |
| Первичный ключ | actor\_id | Целое число | Первичный ключ |

### Нормализация полученных отношений

**1НФ.** Для приведения таблиц к 1НФ требуется составить таблицы (один атрибут – один столбец) и разбить сложные атрибуты на простые, а многозначные атрибуты вынести в отдельные отношения.  
В наших таблицах все атрибуты простые.

**2НФ.** Все отношения находятся во 2НФ, так как для каждого из них введен уникальный ключ.

**3НФ**. Все неключевые атрибуты зависят только от первичного ключа и не зависят друг от друга. Поэтому спроектированная **база данных находится в третьей нормальной форме**.

## Физическое проектирование БД

### Разработка скриптов на создание базы данных и таблиц

БД «Фильмы» проектируется в среде Postgres Pro

*Создание базы данных «*Фильмы*»:*

*Создание таблицы movies:*

*CREATE TABLE movies (*

*id INT PRIMARY KEY,*

*title VARCHAR(255) NOT NULL,*

*description TEXT NOT NULL,*

*release\_year INTEGER NOT NULL,*

*duration\_minutes INTEGER NOT NULL,*

*rating DECIMAL(3,1) NOT NULL*

*);*

*Создание таблицы genres:*

*Создание таблицы genres*

*CREATE TABLE genres (*

*id INT PRIMARY KEY,*

*name VARCHAR(255) NOT NULL*

*);*

*Создание таблицы countries:*

*CREATE TABLE countries (*

*id INT PRIMARY KEY,*

*name VARCHAR(255) NOT NULL*

*);*

*Создание таблицы directors:*

*CREATE TABLE directors (*

*id INT PRIMARY KEY,*

*first\_name VARCHAR(255) NOT NULL,*

*last\_name VARCHAR(255) NOT NULL*

*);*

*Создание таблицы actors:*

*CREATE TABLE actors (*

*id INT PRIMARY KEY,*

*first\_name VARCHAR(255) NOT NULL,*

*last\_name VARCHAR(255) NOT NULL*

*);*

*Создание таблицы movie\_genres:*

*CREATE TABLE movie\_genres (*

*PRIMARY KEY(movie\_id, genre\_id),*

*movie\_id INT,*

*genre\_id INT,*

*FOREIGN KEY (movie\_id) REFERENCES movies(id),*

*FOREIGN KEY (genre\_id) REFERENCES genres(id)*

*);*

*Создание таблицы movie\_countries:*

*CREATE TABLE movie\_countries (*

*PRIMARY KEY(movie\_id, country\_id),*

*movie\_id INT,*

*country\_id INT,*

*FOREIGN KEY (movie\_id) REFERENCES movies(id),*

*FOREIGN KEY (country\_id) REFERENCES countries(id)*

*);*

*Создание таблицы movie\_directors:*

*CREATE TABLE movie\_directors (*

*PRIMARY KEY(movie\_id, director\_id),*

*movie\_id INT,*

*director\_id INT,*

*FOREIGN KEY (movie\_id) REFERENCES movies(id),*

*FOREIGN KEY (director\_id) REFERENCES directors(id)*

*);*

*Создание таблицы movie\_actors:*

*CREATE TABLE movie\_actors (*

*PRIMARY KEY(movie\_id, actor\_id),*

*movie\_id INT,*

*actor\_id INT,*

*FOREIGN KEY (movie\_id) REFERENCES movies(id),*

*FOREIGN KEY (actor\_id) REFERENCES actors(id)*

*);*

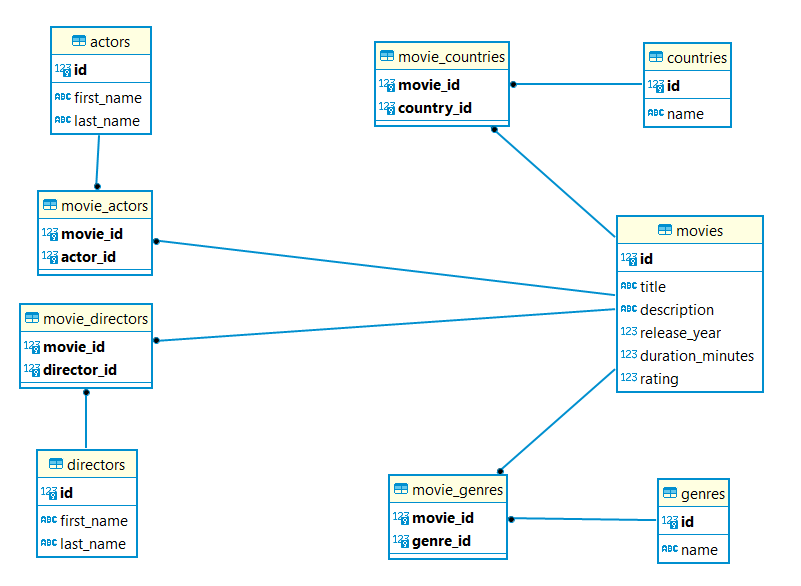


Рисунок 2 -Уточненная ER-диаграмма

### Разработка скриптов на добавление данных в таблицы

*Заполнение таблицы* movies*:*

INSERT INTO movies (id, title, description, release\_year, duration\_minutes, rating)

VALUES

(1, 'Movie 1', 'Description 1', 2020, 120, 8.5),

(2, 'Movie 2', 'Description 2', 2019, 105, 7.9),

(3, 'Movie 3', 'Description 3', 2021, 110, 6.7),

(4, 'Movie 4', 'Description 4', 2018, 95, 9.2),

(5, 'Movie 5', 'Description 5', 2022, 130, 7.1);

*Заполнение таблицы* genres*:*

INSERT INTO genres (id, name)

VALUES

(1, 'Genre 1'),

(2, 'Genre 2'),

(3, 'Genre 3'),

(4, 'Genre 4'),

(5, 'Genre 5');

*Заполнение таблицы* countries*:*

INSERT INTO countries (id, name)

VALUES

(1, 'Country 1'),

(2, 'Country 2'),

(3, 'Country 3'),

(4, 'Country 4'),

(5, 'Country 5');

*Заполнение таблицы* directors*:*

INSERT INTO directors (id, first\_name, last\_name)

VALUES

(1, 'Director 1', 'Lastname 1'),

(2, 'Director 2', 'Lastname 2'),

(3, 'Director 3', 'Lastname 3'),

(4, 'Director 4', 'Lastname 4'),

(5, 'Director 5', 'Lastname 5');

*Заполнение таблицы* actors*:*

INSERT INTO actors (id, first\_name, last\_name)

VALUES

(1, 'Actor 1', 'Lastname 1'),

(2, 'Actor 2', 'Lastname 2'),

(3, 'Actor 3', 'Lastname 3'),

(4, 'Actor 4', 'Lastname 4'),

(5, 'Actor 5', 'Lastname 5');

*Заполнение таблицы* movie\_genres*:*

INSERT INTO movie\_genres (movie\_id, genre\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

*Заполнение таблицы* movie\_countries*:*

INSERT INTO movie\_countries (movie\_id, country\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

*Заполнение таблицы* movie\_directors*:*

INSERT INTO movie\_directors (movie\_id, director\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

*Заполнение таблицы* movie\_actors*:*

INSERT INTO movie\_actors (movie\_id, actor\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

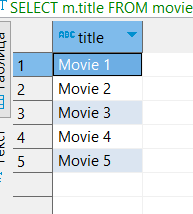
### Разработка необходимых запросов, представлений (view), процедур, функций и триггеров

1. Выбрать названия всех фильмов:

**SELECT**

m.title

**FROM** movies **AS** m;

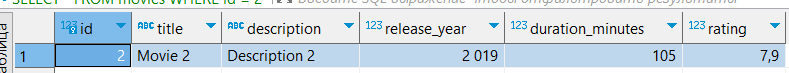
**

1. Выбрать фильм с id 2:

**SELECT** \*

**FROM** movies

**WHERE** id = 2;

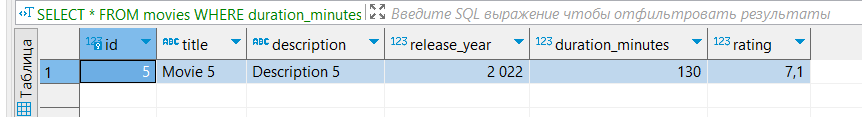
**

1. Выбрать фильмы, которые длиннее 120 минут:

**SELECT** \*

**FROM** movies

**WHERE** duration\_minutes > 120;

**

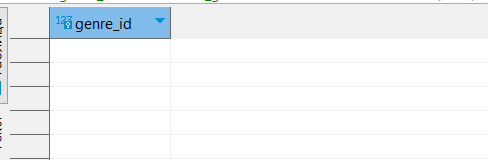
1. Выбрать жанры, которые присутствуют в более чем двух фильмах:

**SELECT** genre\_id

**FROM** movie\_genres

**GROUP** **BY** genre\_id

**HAVING** **COUNT**(movie\_id) > 2;



1. *Пример триггера на DROP TABLE для таблицы movies:*

CREATE OR REPLACE FUNCTION drop\_movies() RETURNS TRIGGER AS $$

BEGIN

DROP TABLE movies;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER drop\_movies\_trigger

BEFORE DELETE ON movies

FOR EACH ROW EXECUTE PROCEDURE drop\_movies();

1. *Пример процедуры добавляющей новый фильм в таблицу:*

CREATE OR REPLACE PROCEDURE add\_movie(

p\_id INT,

p\_title VARCHAR,

p\_description TEXT,

p\_release\_year INT,

p\_duration\_minutes INT,

p\_rating DECIMAL

)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO movies(id, title, description, release\_year, duration\_minutes, rating)

VALUES (p\_id, p\_title, p\_description, p\_release\_year, p\_duration\_minutes, p\_rating);

END;

$$;

Вызов процедуры:

CALL add\_movie(1, 'The Matrix', 'A computer hacker learns..', 1999, 136, 8.7);

1. *Пример функции для подсчета фильмов, вышедших в указанный год:*

CREATE OR REPLACE FUNCTION count\_movies\_by\_year(p\_release\_year INT)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE

v\_count INT;

BEGIN

SELECT COUNT(\*) INTO v\_count

FROM movies

WHERE release\_year = p\_release\_year;

RETURN v\_count;

END;

$$;

Вызов функции:

SELECT count\_movies\_by\_year(1999);

1. *Пример view, которая показывает название фильма и количество связанных с ним жанров:*

CREATE OR REPLACE VIEW movie\_genre\_count AS

SELECT

m.title,

COUNT(mg.genre\_id) AS genre\_count

FROM

movies m

JOIN

movie\_genres mg ON m.id = mg.movie\_id

GROUP BY

m.title;

# Список литературы

* + - 1. Васильева М.А., Хобта Д.О., Фильтрация набора данных. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Издание второе, исправленное и дополненное–М.:РУТ(МИИТ). 2023.–105с.
      2. Васильева М.А., Меркулов Д.А. Группировка и обобщение данных. Рекомендации по выполнению работы и перечень типовых заданий. Учебно-методическое пособие. М.:РУТ(МИИТ), 2023. 46–с.
      3. Васильева М.А., Ракинцев Н.А. Соединение данных из множества таблиц. Рекомендации по выполнению работы и перечень типовых заданий. Учебно-методическое пособие. М.:РУТ(МИИТ), 2023. 63–с.
      4. Балакина Е.П., Васильева М.А., Филипченко К.М. Информационное обеспечение систем управления. Методические указания к курсовому проектированию. Учебно-методическое пособие. Издание второе, исправленное и дополненное, 2023.102–с.