Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОМ ЗАДАНИИ**

Практическое задание №10.

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Е.К. Моргунов

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc18956750)

[1 Цель работы 3](#_Toc18956751)

[2 Задача работы 3](#_Toc18956752)

[3 Ход работы 4](#_Toc18956753)

[4 Вывод 12](#_Toc18956754)

# Цель работы

Проектирование базы данных и ее реализация в среде СУБД PostgreSQL.

# Задачи работы

1. Изучить материал лекций:
   * Лекция 6. Модель данных «сущность–связь»
   * Лекция 7. Нормализация
   * Лекция 8. Методология проектирования баз данных
2. Выбрать предметную область, которая вам интересна и в которой вы разбираетесь.
3. Спроектировать базу данных для выбранной предметной области. При этом нужно следовать общей методологии проектирования баз данных: сначала необходимо создать концептуальную модель данных с использованием ER-диаграмм, затем построить логическую модель, выполнив отображение сущностей и связей в отношения, в завершение нужно выполнить физическое проектирование, создав реляционные таблицы в среде целевой СУБД PostgreSQL.

Число таблиц должно быть равно 4-5. Обязательно должен быть создан хотя бы один триггер (и триггерная функция к нему).

1. Ввести небольшое количество записей в таблицы базы данных, чтобы можно было продемонстрировать типичные запросы к базе данных.
2. Создавать приложение (интерфейс пользователя) не обязательно. В том случае, если приложение не разрабатывалось, нужно заранее подготовить несколько типичных запросов к базе данных и сохранить их в отдельных текстовых файлах. Для демонстрации этих запросов их можно вызывать как извне утилиты psql, так и изнутри нее.

Например, для выполнения запроса, содержащегося в файле file\_with\_request.sql, нужно сделать так :

psql -d your\_database -f file\_with\_request.sql -U postgres

1. Подготовить отчет.

* Нужно поместить в него краткое описание предметной области (не более половины страницы), концептуальную, логическую и физическую схемы базы данных. Физическую схему базы данных нужно представить в отчете в виде SQL-команд для создания таблиц, представлений (если они используются), триггеров и триггерных функций.
* Для каждого отношения (таблицы) необходимо указать номер нормальной формы, в которой это отношение находится, и кратко обосновать, из чего это следует. Если какое-либо отношение не находится хотя бы в 3НФ, необходимо обосновать, почему принято такое проектное решение.

# Ход работы

1. Концептуальная модель:

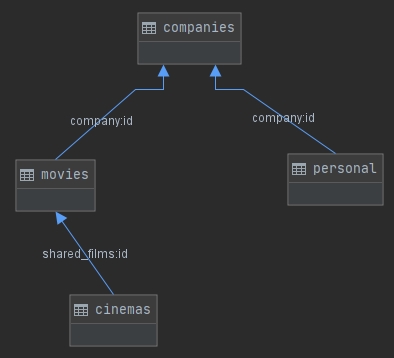


Рисунок 1 – Концептуальная модель

1. Логическая модель:

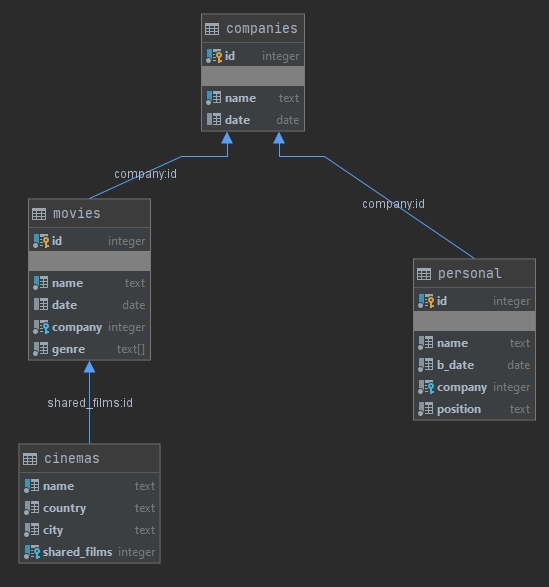


Рисунок 2 – Логическая модель

1. Физическое проектирование:

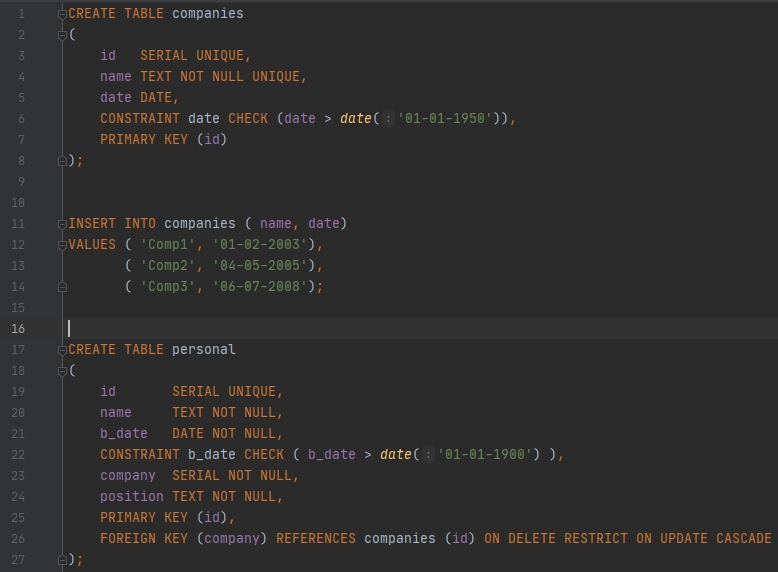


Рисунок 3 – Физическое проектирование (часть кода 1)

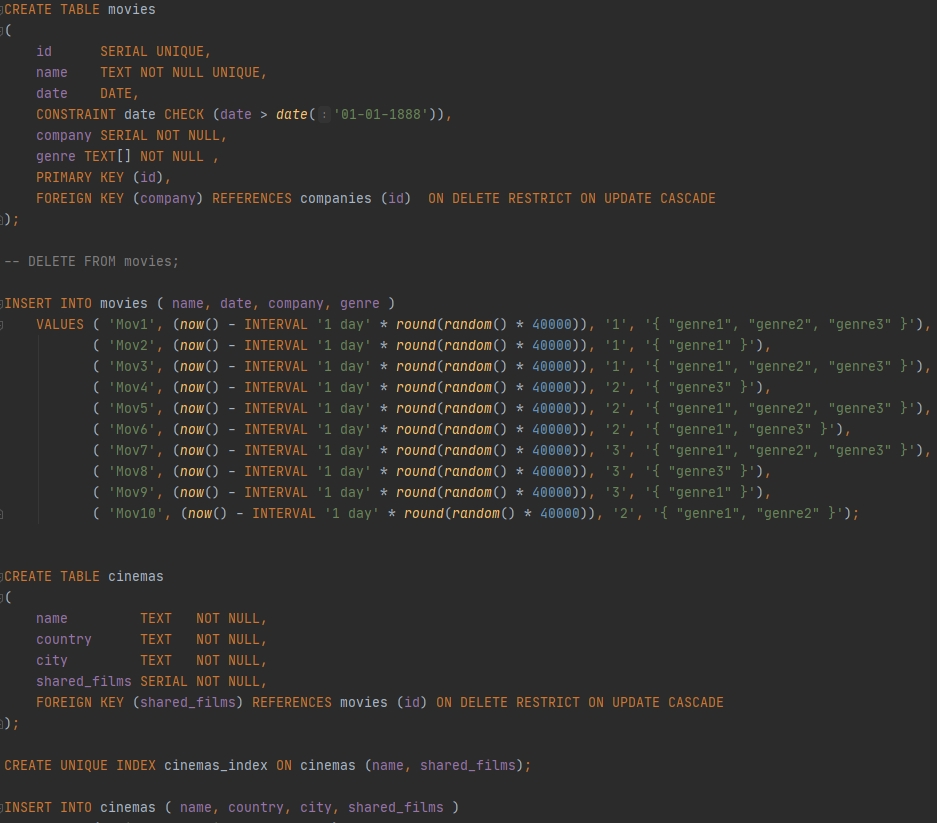


Рисунок 4 – Физическое проектирование (часть кода 2)

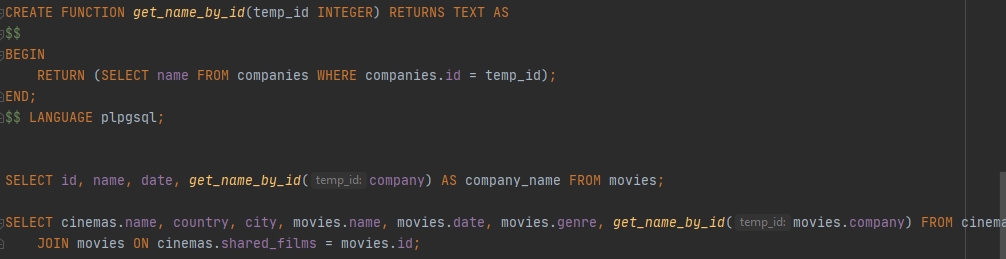


Рисунок 5 – Физическое проектирование (часть кода 3)

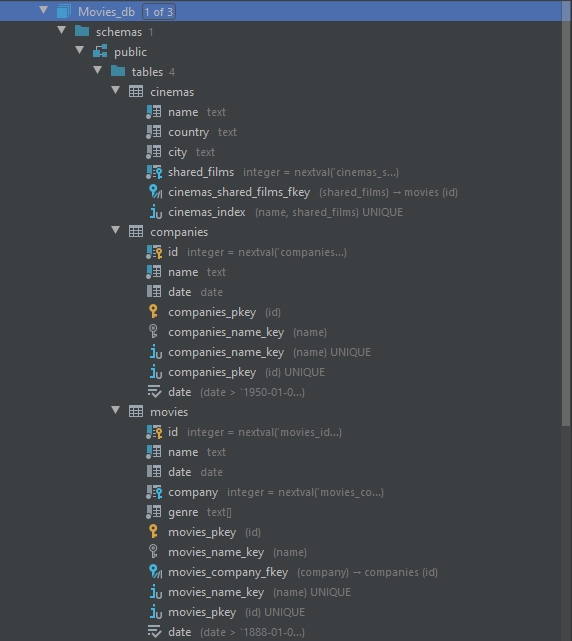


Рисунок 6 – Физическое проектирование (структура базы данных 1)

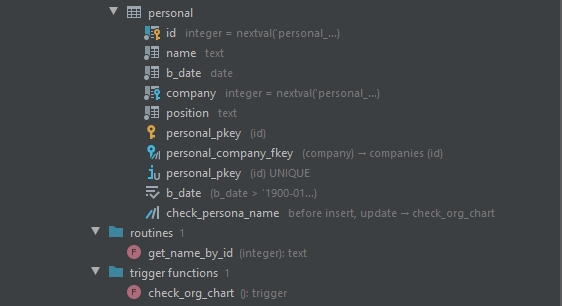


Рисунок 7 – Физическое проектирование (структура базы данных 1)

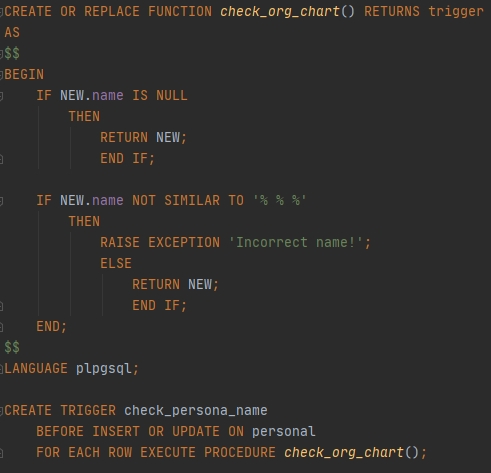


Рисунок 8 – Физическое проектирование (триггерная функция и сам триггер)

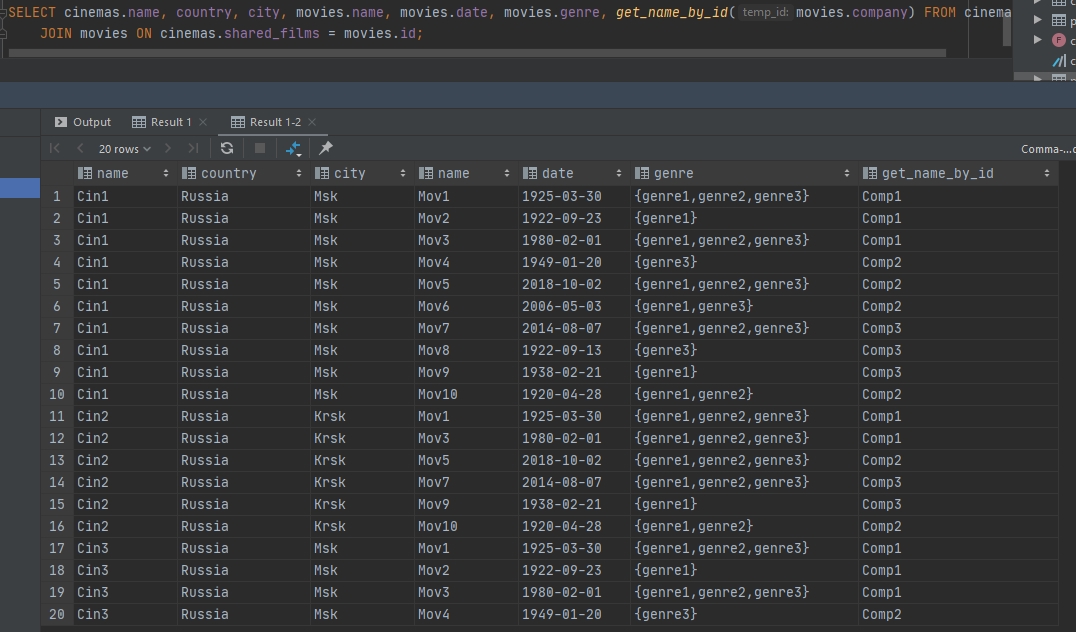


Рисунок 9 – Физическое проектирование (результат составного запроса, с использованием функции)

1. Таблица «cinemas» находится во второй нормальной форме, так как столбец «country» зависит не только он названия кинотеатра («name», ключевой атрибут), но и от города «city», в котором находится кинотеатр. Транзитивная зависимость атрибутов «name» => «city» => «country».

Таблица «companies» находится в третьей нормальной форме.

Таблица «movies» находится в третьей нормальной форме.

Таблица «personal» находится в третьей нормальной форме.

1. Листинг 1 – код для реализации данной БД:

CREATE TABLE companies  
(  
 id SERIAL UNIQUE,  
 name TEXT NOT NULL UNIQUE,  
 date DATE,  
 CONSTRAINT date CHECK (date > *date*('01-01-1950')),  
 PRIMARY KEY (id)  
);  
  
  
CREATE TABLE personal  
(  
 id SERIAL UNIQUE,  
 name TEXT NOT NULL,  
 b\_date DATE NOT NULL,  
 CONSTRAINT b\_date CHECK ( b\_date > *date*('01-01-1900') ),  
 company SERIAL NOT NULL,  
 position TEXT NOT NULL,  
 PRIMARY KEY (id),  
 FOREIGN KEY (company) REFERENCES companies (id) ON DELETE RESTRICT ON UPDATE CASCADE  
);  
  
  
CREATE OR REPLACE FUNCTION *check\_org\_chart*() RETURNS trigger  
AS  
$$  
BEGIN  
 IF NEW.name IS NULL  
 THEN  
 RETURN NEW;  
 END IF;  
  
 IF NEW.name NOT SIMILAR TO '% % %'  
 THEN  
 RAISE EXCEPTION 'Incorrect name!';  
 ELSE  
 RETURN NEW;  
 END IF;  
 END;  
$$  
LANGUAGE plpgsql;  
  
CREATE TRIGGER check\_persona\_name  
 BEFORE INSERT OR UPDATE ON personal  
 FOR EACH ROW EXECUTE PROCEDURE *check\_org\_chart*();  
CREATE TABLE movies  
(  
 id SERIAL UNIQUE,  
 name TEXT NOT NULL UNIQUE,  
 date DATE,  
 CONSTRAINT date CHECK (date > *date*('01-01-1888')),  
 company SERIAL NOT NULL,  
 genre TEXT[] NOT NULL ,  
 PRIMARY KEY (id),  
 FOREIGN KEY (company) REFERENCES companies (id) ON DELETE RESTRICT ON UPDATE CASCADE  
);  
  
CREATE TABLE cinemas  
(  
 name TEXT NOT NULL,  
 country TEXT NOT NULL,  
 city TEXT NOT NULL,  
 shared\_films SERIAL NOT NULL,  
 FOREIGN KEY (shared\_films) REFERENCES movies (id) ON DELETE RESTRICT ON UPDATE CASCADE  
);  
  
CREATE UNIQUE INDEX cinemas\_index ON cinemas (name, shared\_films);

# Вывод

В ходе данной лабораторной работы была спроектирована и реализована база данных в среде СУБД PostgreSQL.