**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  
«База данных музыка»**

Выполнил: ст. гр. ТКИ-441  
Мясумов Р.М.  
Проверил: Васильева М.А.

Москва 2023

Оглавление

[ЦЕЛЬ КУРСОВОГО ПРОЕКТА 5](#_Toc136214271)

[ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ 5](#_Toc136214272)

[ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 6](#_Toc136214273)

[1 Инфологическое проектирование 6](#_Toc136214274)

[1.1 Анализ предметной области 6](#_Toc136214275)

[1.2 Анализ информационных задач и круга пользователей системы 7](#_Toc136214276)

[2 Логическое проектирование реляционной БД 8](#_Toc136214277)

[2.1 Составление реляционных отношений 8](#_Toc136214278)

[2.2 Нормализация полученных отношений 11](#_Toc136214279)

[3 Физическое проектирование БД 11](#_Toc136214280)

[3.1 Разработка скриптов на создание базы данных и таблиц 11](#_Toc136214281)

[3.2 Разработка скриптов на добавление данных в таблицы 14](#_Toc136214282)

[3.3 Разработка необходимых запросов, представлений (view), процедур, функций и триггеров 15](#_Toc136214283)

[Список литературы 19](#_Toc136214284)

# 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 «База данных музыка». 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 ресурса с музыкой.

Ниже приведено описание основных сущностей предметной области:

1. Таблица "genres", которая содержит информацию о жанрах, включая их id жанра и название.
2. Таблица "styles" содержит информацию о стилях, включая их id, название и id жанра к ним относящегося.
3. Таблица "album\_styles" содержит информацию об альбомах и стилях.
4. Таблица "albums" содержит информацию об альбомах, их id, название, дату выхода и id исполнителя.
5. Таблица "tracks" содержит информацию о песнях, включая ID песни, название, продолжительность и id альбома.
6. Таблица "artists" содержит информацию о исполнителях, включая ID исполнителя, имя и страну.

# ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

## Инфологическое проектирование

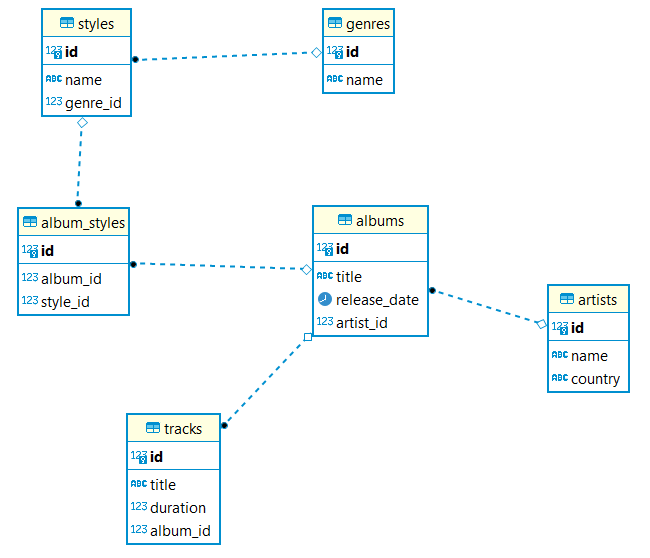
### Анализ предметной области

База данных создаётся для сайта с музыкой.

Выделим базовые сущности этой предметной области:

* genres – жанры музыки.
* styles – стили музыки.
* album\_styles – вспомогательная таблица с альбомами и стилями.
* albums - альбомы.
* tracks - песни.
* artists – исполнители.

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 один-ко-многим. Такие связи легко реализуются средствами СУБД.

### Анализ информационных задач и круга пользователей системы

Система создаётся для обслуживания следующих групп пользователей:

* Администратор
* Модератор
* 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 из таблицы "albums";
* выбрать артистах, которые из "United Kingdom";
* выбрать песни, продолжительность которых больше 300;
* выбрать жанры, которые имеют один и более стилей.

## 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.

Предметная область включает в себя несколько базовых сущностей: жанры, стили, альбомы, песни(треки) и исполнители.

1. Отношение один-ко-многим между artists и albums: один исполнитель (artist) может иметь много альбомов (albums), но каждый альбом связан только с одним исполнителем. Это отношение устанавливается через внешний ключ artist\_id в таблице albums, который ссылается на первичный ключ id в таблице artists.
2. Отношение один-ко-многим между albums и tracks: один альбом может иметь много треков, но каждый трек связан только с одним альбомом. Это отношение устанавливается через внешний ключ album\_id в таблице tracks, который ссылается на первичный ключ id в таблице albums.
3. Отношение один-ко-многим между genres и styles: один жанр может иметь много стилей, но каждый стиль связан только с одним жанром. Это отношение устанавливается через внешний ключ genre\_id в таблице styles, который ссылается на первичный ключ id в таблице genres.
4. Отношение многие-ко-многим между albums и styles: один альбом может быть связан с несколькими стилями, и каждый стиль может быть связан с несколькими альбомами. Это отношение реализуется через вспомогательную таблицу album\_styles, в которой внешние ключи album\_id и style\_id соответственно ссылаются на albums(id) и styles(id).

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

**Таблица 1 – Схема отношения** **Исполнители** (artists)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | name | Текст | Обязательное поле |
| Страна | country | Дата | Необязательное поле |

Таблица 2 – Схема отношения Альбомы (albums)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | title | Текст | Обязательное поле |
| Дата выхода | release\_date | Дата | Обязательное поле |
| Исполнитель | artist\_id | Целое число | Необязательное поле |

**Таблица 3 – Схема отношения** **Треки** (tracks)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | title | Текст | Обязательное поле |
| Продолжительность | duration | Целое число | Обязательное поле |
| Альбом | album\_id | Целое число | Необязательное поле |

**Таблица 4 – Схема отношения** **Жанров** (genres)

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

Таблица 5 – Схема отношения Стилей (styles)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Название | name | Текст | Обязательное поле |
| Жанр | genre\_id | Целое число | Необязательное поле |

Таблица 6 – Схема отношения связей между альбомами и стилями (album\_styles)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | id | Целое число | Первичный ключ |
| Альбом | album\_id | Целое число | Необязательное поле |
| Стиль | style\_id | Целое число | Необязательное поле |

### 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 «База данных музыки» проектируется в среде Postgres Pro

*Создание базы данных «*База данных музыки*»:*

*Создание таблицы исполнителей:*

*CREATE TABLE artists (*

*id SERIAL PRIMARY KEY,*

*name VARCHAR(255) NOT NULL,*

*country VARCHAR(100) NULL*

*);*

*Создание таблицы альбомов:*

*CREATE TABLE albums (*

*id SERIAL PRIMARY KEY,*

*title VARCHAR(255) NOT NULL,*

*release\_date DATE NOT NULL,*

*artist\_id INTEGER,*

*FOREIGN KEY (artist\_id) REFERENCES artists(id)*

*);*

*Создание таблицы треков:*

*CREATE TABLE tracks (*

*id SERIAL PRIMARY KEY,*

*title VARCHAR(255) NOT NULL,*

*duration INTEGER NOT NULL,*

*album\_id INTEGER,*

*FOREIGN KEY (album\_id) REFERENCES albums(id)*

*);*

*Создание таблицы жанров:*

*CREATE TABLE genres (*

*id SERIAL PRIMARY KEY,*

*name VARCHAR(255) NOT NULL UNIQUE*

*);*

*Создание таблицы стилей:*

*CREATE TABLE styles (*

*id SERIAL PRIMARY KEY,*

*name VARCHAR(255) NOT NULL UNIQUE,*

*genre\_id INTEGER,*

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

*);*

*Создание таблицы связей между альбомами и стилями:*

*CREATE TABLE album\_styles (*

*id SERIAL PRIMARY KEY,*

*album\_id INTEGER,*

*style\_id INTEGER,*

*FOREIGN KEY (album\_id) REFERENCES albums(id),*

*FOREIGN KEY (style\_id) REFERENCES styles(id)*

*);*

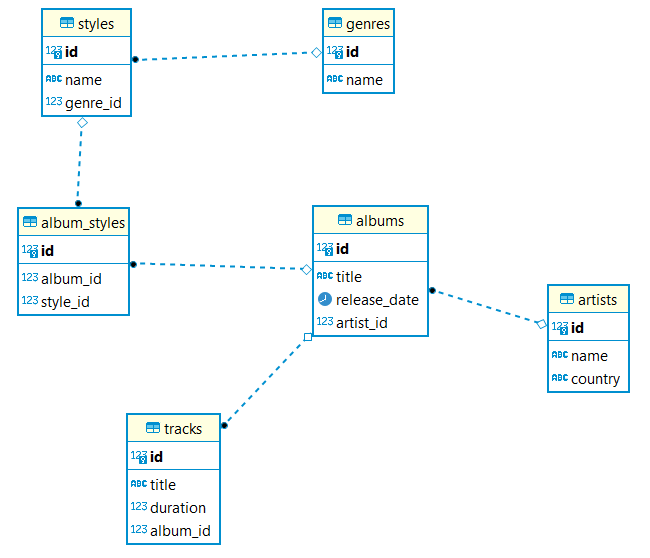


Рисунок 2 -Уточненная ER-диаграмма

### Разработка скриптов на добавление данных в таблицы

*Заполнение таблицы* исполнителей*:*

INSERT INTO artists (name, country)

VALUES ('The Beatles', 'United Kingdom'),

('Led Zeppelin', 'United Kingdom'),

('Pink Floyd', 'United Kingdom'),

('Queen', 'United Kingdom'),

('AC/DC', 'Australia');

*Заполнение таблицы* альбомов*:*

INSERT INTO albums (title, release\_date, artist\_id)

VALUES ('Abbey Road', '1969-09-26', 1),

('IV', '1971-01-01', 2),

('The Wall', '1979-11-30', 3),

('A Night at the Opera', '1975-11-21', 4),

('Back in Black', '1980-07-25', 5);

*Заполнение таблицы* треков*:*

INSERT INTO tracks (title, duration, album\_id)

VALUES ('Come Together', 260, 1),

('Stairway to Heaven', 482, 2),

('Another Brick in the Wall', 376, 3),

('Bohemian Rhapsody', 354, 4),

('Hells Bells', 312, 5);

*Заполнение таблицы* жанров*:*

INSERT INTO genres (name)

VALUES ('Rock'),

('Pop'),

('Jazz'),

('Blues'),

('Electronic');

*Заполнение таблицы* стилей*:*

INSERT INTO styles (name, genre\_id)

VALUES ('Classic Rock', 1),

('Pop Rock', 2),

('Smooth Jazz', 3),

('Electric Blues', 4),

('House', 5);

*Заполнение таблицы* связей между альбомами и стилями*:*

INSERT INTO album\_styles (album\_id, style\_id)

VALUES (1, 1),

(2, 1),

(3, 1),

(4, 2),

(5, 1);

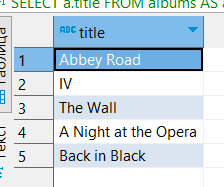
### Разработка необходимых запросов, представлений (view), процедур, функций и триггеров

1. Выбрать названия из таблицы "albums":

**SELECT**

a.title

**FROM** albums **AS** a;

**

1. Выбрать артистах, которые из "United Kingdom":

**SELECT** \*

**FROM** artists

**WHERE** country = 'United Kingdom';

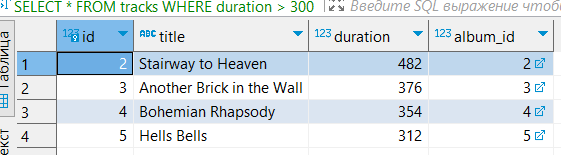
**

1. Выбрать песни, продолжительность которых больше 300:

**SELECT** \*

**FROM** tracks

**WHERE** duration > 300;

**

1. Выбрать жанры, которые имеют один и более стилей:

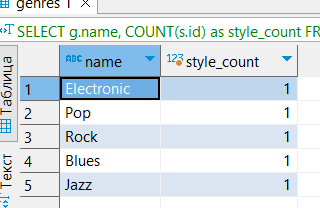
**SELECT** g.**name**, **COUNT**(s.id) **as** style\_count

**FROM** genres **AS** g

**JOIN** styles **AS** s **ON** g.id = s.genre\_id

**GROUP** **BY** g.**name**

**HAVING** **COUNT**(s.id) >= 1;



1. *Пример триггера на DROP TABLE, на удаление таблицы albums:*

CREATE OR REPLACE FUNCTION delete\_album\_relations() RETURNS TRIGGER AS $$

BEGIN

DELETE FROM tracks WHERE album\_id = OLD.id;

DELETE FROM album\_styles WHERE album\_id = OLD.id;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_album\_relations\_trigger

BEFORE DELETE ON albums

FOR EACH ROW

EXECUTE FUNCTION delete\_album\_relations();

1. *Пример процедуры добавляющей новый альбом:*

CREATE OR REPLACE PROCEDURE create\_album(IN p\_title VARCHAR(255), IN p\_release\_date DATE, IN p\_artist\_id INTEGER)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO albums (title, release\_date, artist\_id) VALUES (p\_title, p\_release\_date, p\_artist\_id);

END;

$$;

Вызов процедуры:

CALL create\_album('Название альбома', '2023-06-01', 1);

1. *Пример функции для подсчета альбомов исполнителя:*

CREATE OR REPLACE FUNCTION count\_artist\_albums(p\_artist\_id INTEGER)

RETURNS INTEGER

LANGUAGE plpgsql

AS $$

DECLARE

album\_count INTEGER;

BEGIN

SELECT COUNT(\*) INTO album\_count FROM albums WHERE artist\_id = p\_artist\_id;

RETURN album\_count;

END;

$$;

Вызов функции:

SELECT count\_artist\_albums(1);

1. *Пример view, которая показывает название каждого альбома вместе с именем соответствующего исполнителя:*

CREATE OR REPLACE VIEW artist\_albums AS

SELECT

artists.name AS artist\_name,

albums.title AS album\_title

FROM

albums

JOIN

artists ON albums.artist\_id = artists.id;

# 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.