

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА** - **Российский технологический университет"**  **РТУ МИРЭА** |

**Институт** Искусственного Интеллекта (ИИИ)

**Кафедра** Промышленной Информатики

**ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

**по дисциплине**

**«Разработка баз данных»**

Студент группы: ИКБО-04-22 Егоров Л.А. *(Ф. И.О.студента)*

Преподаватель Зайцев И.Ю. *(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2024

# Содержание

[1 Практическая работа №1 5](#_Toc181037007)

[1.1 Создание базы данных 5](#_Toc181037008)

[1.2 Таблица Musician 6](#_Toc181037009)

[1.3 Таблица Music\_group 7](#_Toc181037010)

[1.4 Таблица group\_musician 8](#_Toc181037011)

[1.5 Таблица performer 9](#_Toc181037012)

[1.6 Таблица release\_type 10](#_Toc181037013)

[1.7 Таблица album 11](#_Toc181037014)

[1.8 Таблица album\_performer 11](#_Toc181037015)

[1.9 Таблица song 12](#_Toc181037016)

[1.10 Таблица song\_performer 14](#_Toc181037017)

[1.11 Таблица single 14](#_Toc181037018)

[1.12 Таблица single\_song 15](#_Toc181037019)

[1.13 Таблица producer 16](#_Toc181037020)

[1.14 Таблица producer\_album 17](#_Toc181037021)

[1.15 Таблица producer\_song 18](#_Toc181037022)

[1.16 Результат создания базы данных 19](#_Toc181037023)

[2 Практическая работа №2 23](#_Toc181037024)

[2.1 Выборка данных 23](#_Toc181037025)

[2.2 Выборка данных с сортировкой 28](#_Toc181037026)

[2.3 Операторы изменения данных 29](#_Toc181037027)

[2.4 Результат выполнения работы 31](#_Toc181037028)

[3 Практическая работа №3 32](#_Toc181037029)

[3.1 Бэкап базы данных 32](#_Toc181037030)

[3.2 Выборка данных согласно операциям реляционной алгебры 32](#_Toc181037031)

[3.2.1 Операция проекции 32](#_Toc181037032)

[3.2.2 Операция селекции 33](#_Toc181037033)

[3.2.3 Операция соединения 33](#_Toc181037034)

[3.2.4 Операция объединения 34](#_Toc181037035)

[3.2.5 Операция пересечения 35](#_Toc181037036)

[3.2.6 Операция разности 35](#_Toc181037037)

[3.2.7 Операция группировки 36](#_Toc181037038)

[3.2.8 Операция сортировки 37](#_Toc181037039)

[3.2.9 Операция деления 37](#_Toc181037040)

[3.2.10 Создание представления 38](#_Toc181037041)

[3.3 Хранимые процедуры, функции и триггеры 39](#_Toc181037042)

[3.3.1 Процедура 39](#_Toc181037043)

[3.3.2 Функция 40](#_Toc181037044)

[3.3.3 Триггер, отрабатывающий после добавления записи 41](#_Toc181037045)

[3.3.4 Триггер, отрабатывающий до добавления записи 42](#_Toc181037046)

[4 Практическая работа №4 44](#_Toc181037047)

[4.1 Агрегирующие функции 44](#_Toc181037048)

[4.2 Ранжирующие функции 44](#_Toc181037049)

[4.3 Функции смещения 47](#_Toc181037050)

[Вывод 48](#_Toc181037051)

[Приложения 49](#_Toc181037052)

[Приложение А 50](#_Toc181037053)

[Приложение Б 54](#_Toc181037054)

[Приложение В 55](#_Toc181037055)

[Приложение Г 75](#_Toc181037056)

# 1 Практическая работа №1

**Цель:** создать базу данных согласно выбранной теме, используя PostgreSQL.

**Выбранная тема:** Музыкальный лейбл

На Рисунке 1.1 продемонстрирована физическая модель базы данных, отражающая ключи в каждой таблице, а также связи между ними.

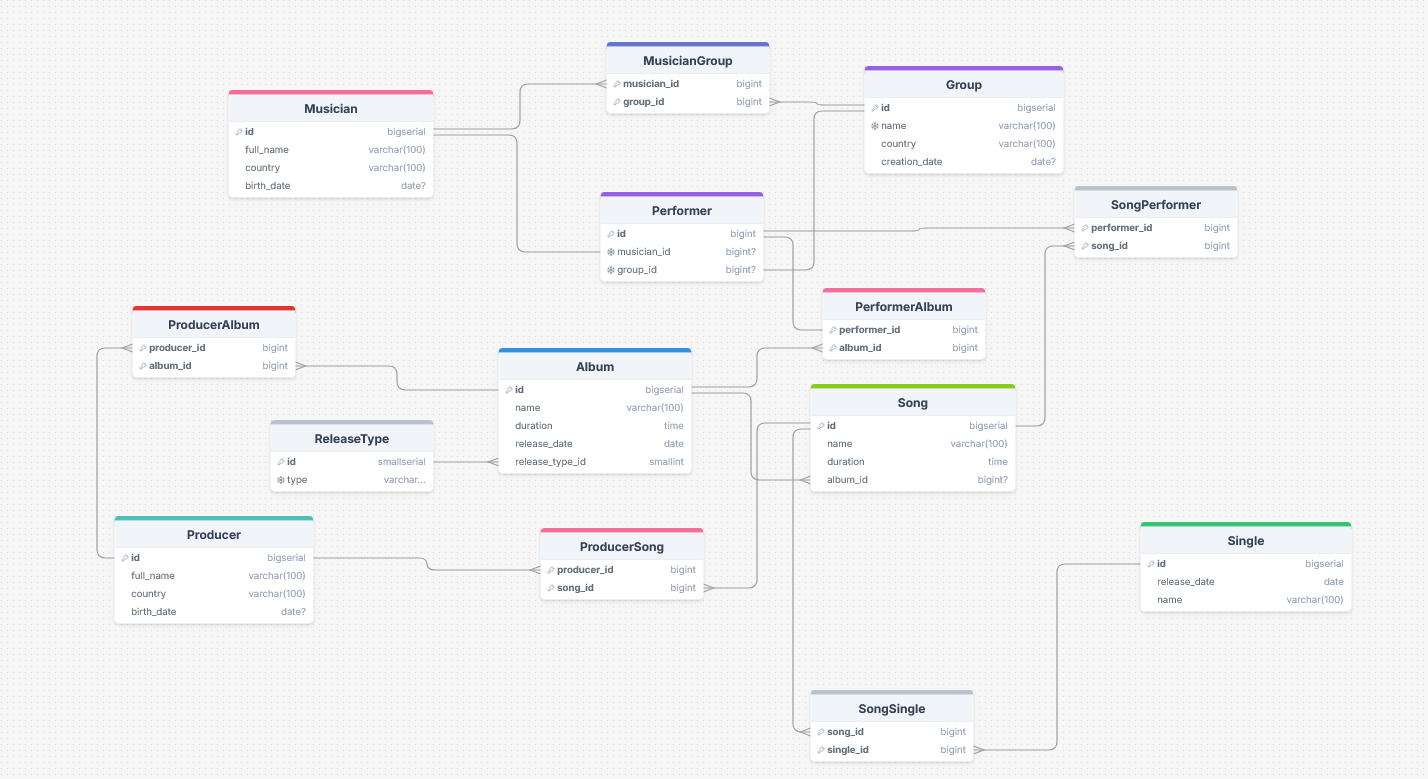


Рисунок 1.1 – Физическая модель базы данных

## 1.1 Создание базы данных

На Рисунке 1.1.1 представлен запрос на создание базы данных с названием music\_label, а также последующий переход в неё.

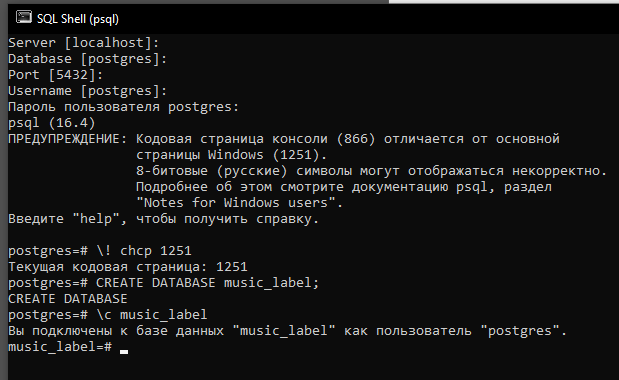


Рисунок 1.1.1 – Создание базы данных

## 1.2 Таблица Musician

На Рисунке 1.2.1 представлен запрос на создание таблицы Musician, которая затем была заполнена данными (Рисунок 1.2.2). Её содержимое представлено на Рисунке 1.2.3.

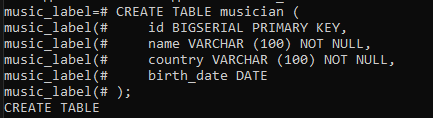


Рисунок 1.2.1 – Создание таблицы Musician



Рисунок 1.2.2 – Заполнение таблицы Musician



Рисунок 1.2.3 – Содержимое таблицы Musician

## 1.3 Таблица Music\_group

На Рисунке 1.3.1 представлен запрос на создание таблицы Music\_group, которая затем была заполнена данными (Рисунок 1.3.2). Её содержимое представлено на Рисунке 1.3.3.

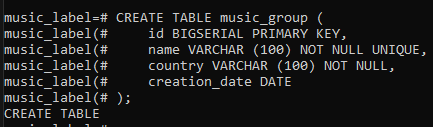


Рисунок 1.3.1 – Создание таблицы Music\_group

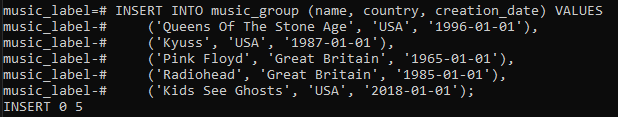


Рисунок 1.3.2 – Заполнение таблицы Music\_group

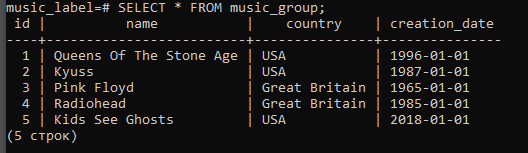


Рисунок 1.3.3 – Содержимое таблицы Music\_group

## 1.4 Таблица group\_musician

На Рисунке 1.4.1 представлен запрос на создание таблицы group\_musician, которая затем была заполнена данными (Рисунок 1.4.2). Её содержимое представлено на Рисунке 1.4.3.

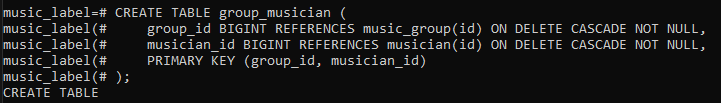


Рисунок 1.4.1 – Создание таблицы group\_musician

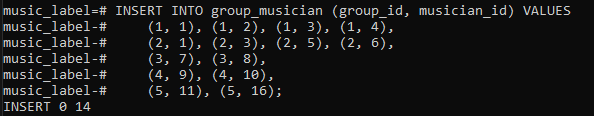


Рисунок 1.4.2 – Заполнение таблицы group\_musician

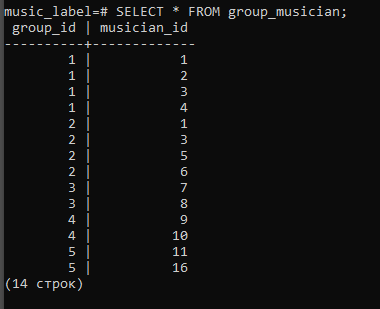


Рисунок 1.4.3 – Содержимое таблицы group\_musician

## 1.5 Таблица performer

На Рисунке 1.5.1 представлен запрос на создание таблицы performer, которая затем была заполнена данными (Рисунок 1.5.2). Её содержимое представлено на Рисунке 1.5.3.

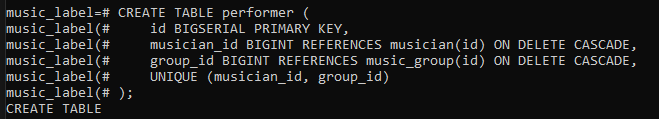


Рисунок 1.5.1 – Создание таблицы performer

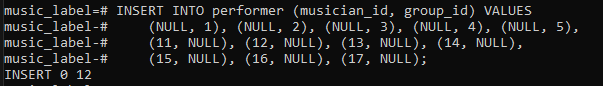


Рисунок 1.5.2 – Заполнение таблицы performer

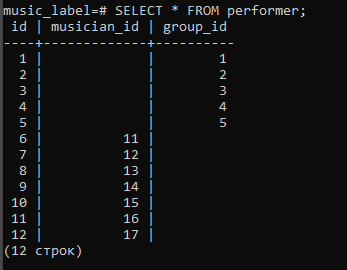


Рисунок 1.5.3 – Содержимое таблицы performer

## 1.6 Таблица release\_type

На Рисунке 1.6.1 представлен запрос на создание таблицы release\_type, которая затем была заполнена данными (Рисунок 1.6.2). Её содержимое представлено на Рисунке 1.6.3.

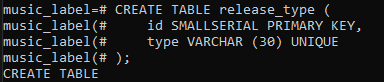


Рисунок 1.6.1 – Создание таблицы release\_type



Рисунок 1.6.2 – Заполнение таблицы release\_type

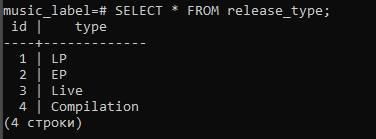


Рисунок 1.6.3 – Содержимое таблицы release\_type

## 1.7 Таблица album

На Рисунке 1.7.1 представлен запрос на создание таблицы album, которая затем была заполнена данными (Рисунок 1.7.2). Её содержимое представлено на Рисунке 1.7.3.

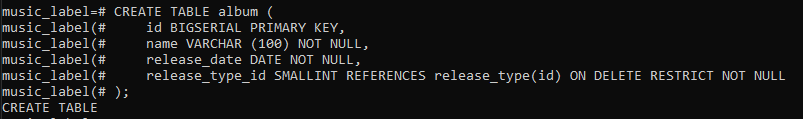


Рисунок 1.7.1 – Создание таблицы album

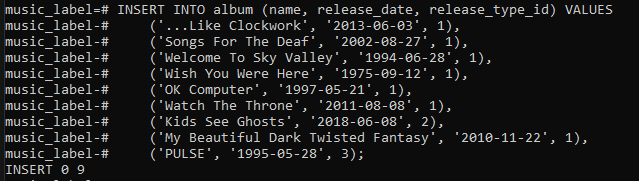


Рисунок 1.7.2 – Заполнение таблицы album



Рисунок 1.7.3 – Содержимое таблицы album

## 1.8 Таблица album\_performer

На Рисунке 1.8.1 представлен запрос на создание таблицы album\_performer, которая затем была заполнена данными (Рисунок 1.8.2). Её содержимое представлено на Рисунке 1.8.3.

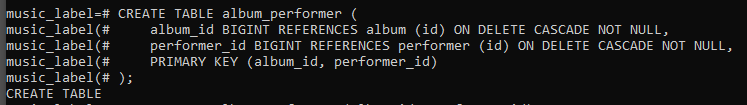


Рисунок 1.8.1 – Создание таблицы album\_performer

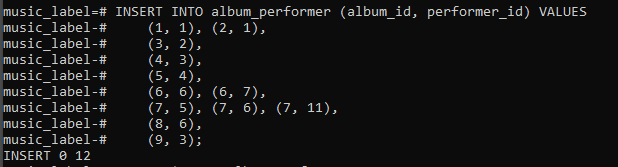


Рисунок 1.8.2 – Заполнение таблицы album\_performer

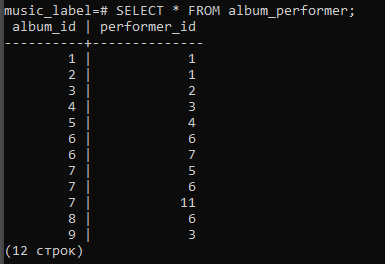


Рисунок 1.8.3 – Содержимое таблицы album\_performer

## 1.9 Таблица song

На Рисунке 1.9.1 представлен запрос на создание таблицы song, которая затем была заполнена данными (Рисунок 1.9.2). Её содержимое представлено на Рисунке 1.9.3.

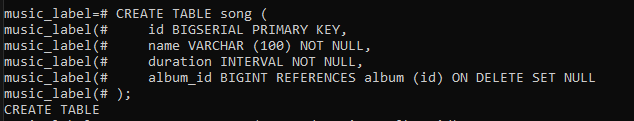


Рисунок 1.9.1 – Создание таблицы song

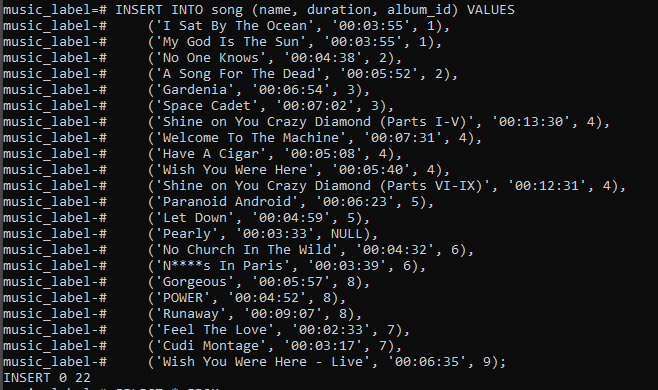


Рисунок 1.9.2 – Заполнение таблицы song

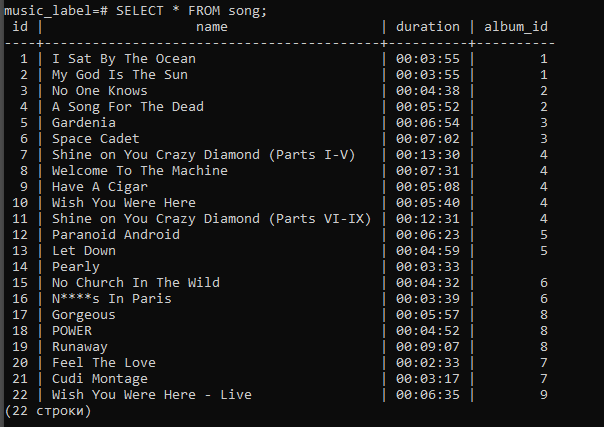


Рисунок 1.9.3 – Содержимое таблицы song

## 1.10 Таблица song\_performer

На Рисунке 1.10.1 представлен запрос на создание таблицы song\_performer, которая затем была заполнена данными (Рисунок 1.10.2). Её содержимое представлено на Рисунке 1.10.3.

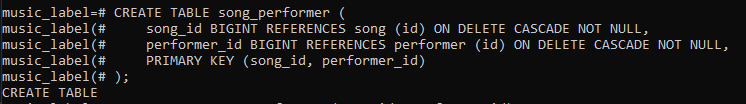


Рисунок 1.10.1 – Создание таблицы song\_performer

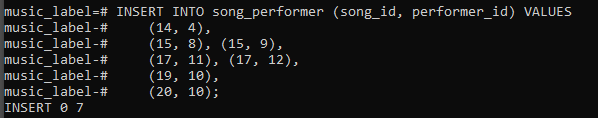


Рисунок 1.10.2 – Заполнение таблицы song\_performer

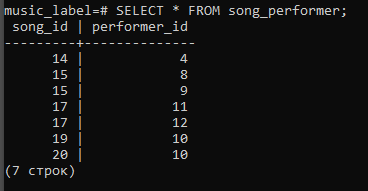


Рисунок 1.10.3 – Содержимое таблицы song\_performer

## 1.11 Таблица single

На Рисунке 1.11.1 представлен запрос на создание таблицы single, которая затем была заполнена данными (Рисунок 1.11.2). Её содержимое представлено на Рисунке 1.11.3.

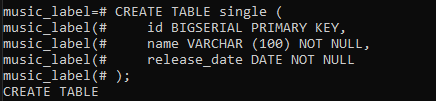


Рисунок 1.11.1 – Создание таблицы single

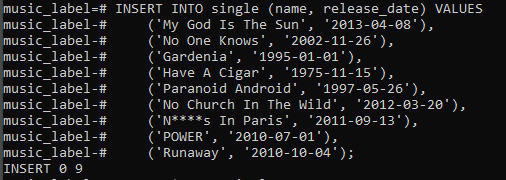


Рисунок 1.11.2 – Заполнение таблицы single

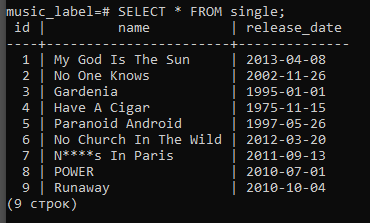


Рисунок 1.11.3 – Содержимое таблицы single

## 1.12 Таблица single\_song

На Рисунке 1.12.1 представлен запрос на создание таблицы single\_song, которая затем была заполнена данными (Рисунок 1.12.2). Её содержимое представлено на Рисунке 1.12.3.

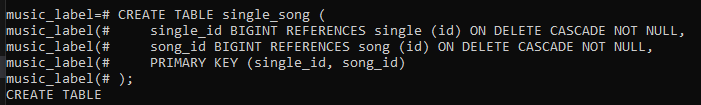


Рисунок 1.12.1 – Создание таблицы single\_song

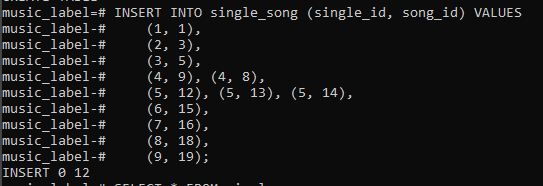


Рисунок 1.12.2 – Заполнение таблицы single\_song

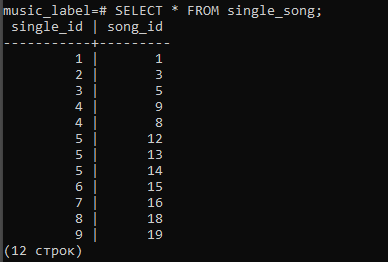


Рисунок 1.12.3 – Содержимое таблицы single\_song

## 1.13 Таблица producer

На Рисунке 1.13.1 представлен запрос на создание таблицы producer, которая затем была заполнена данными (Рисунок 1.13.2). Её содержимое представлено на Рисунке 1.13.3.

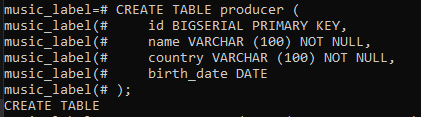


Рисунок 1.13.1 – Создание таблицы producer



Рисунок 1.13.2 – Заполнение таблицы producer



Рисунок 1.13.3 – Содержимое таблицы producer

## 1.14 Таблица producer\_album

На Рисунке 1.14.1 представлен запрос на создание таблицы producer\_album, которая затем была заполнена данными (Рисунок 1.14.2). Её содержимое представлено на Рисунке 1.14.3.

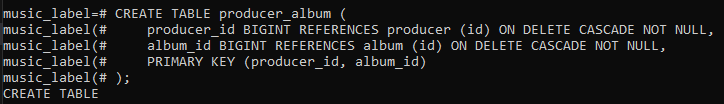


Рисунок 1.14.1 – Создание таблицы producer\_album

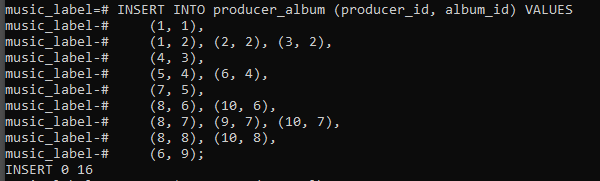


Рисунок 1.14.2 – Заполнение таблицы producer\_album

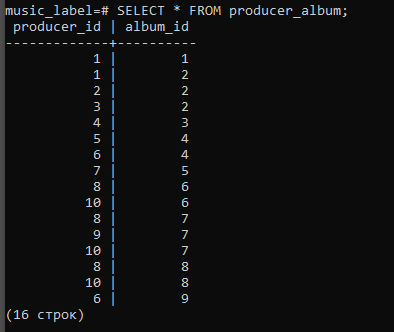


Рисунок 1.14.3 – Содержимое таблицы producer\_album

## 1.15 Таблица producer\_song

На Рисунке 1.15.1 представлен запрос на создание таблицы producer\_song, которая затем была заполнена данными (Рисунок 1.15.2). Её содержимое представлено на Рисунке 1.15.3.

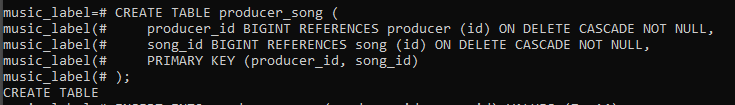


Рисунок 1.15.1 – Создание таблицы producer\_song



Рисунок 1.15.2 – Заполнение таблицы producer\_song

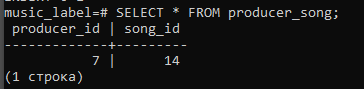


Рисунок 1.15.3 – Содержимое таблицы producer\_song

## 1.16 Результат создания базы данных

После проделанных запросов создана база данных music\_label с 14 таблицами, каждая из которых была заполнена несколькими записями. Структура базы данных представлена на Рисунке 1.16.1, структуры таблиц представлены на Рисунках 1.16.2 - 1.16.5. В Листинге А.1 представлен скрипт, содержащий все запросы на создание таблиц и их заполнение.

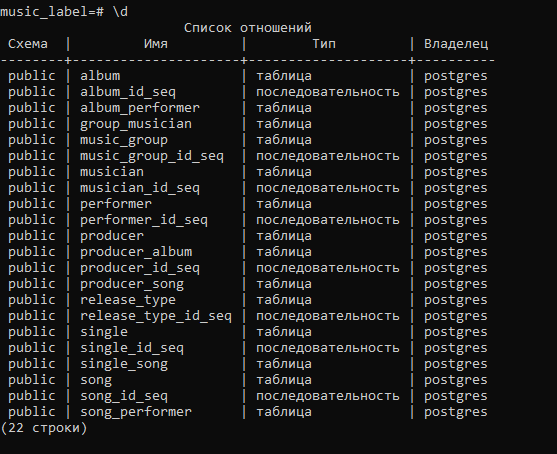


Рисунок 1.16.1 – Структура базы данных

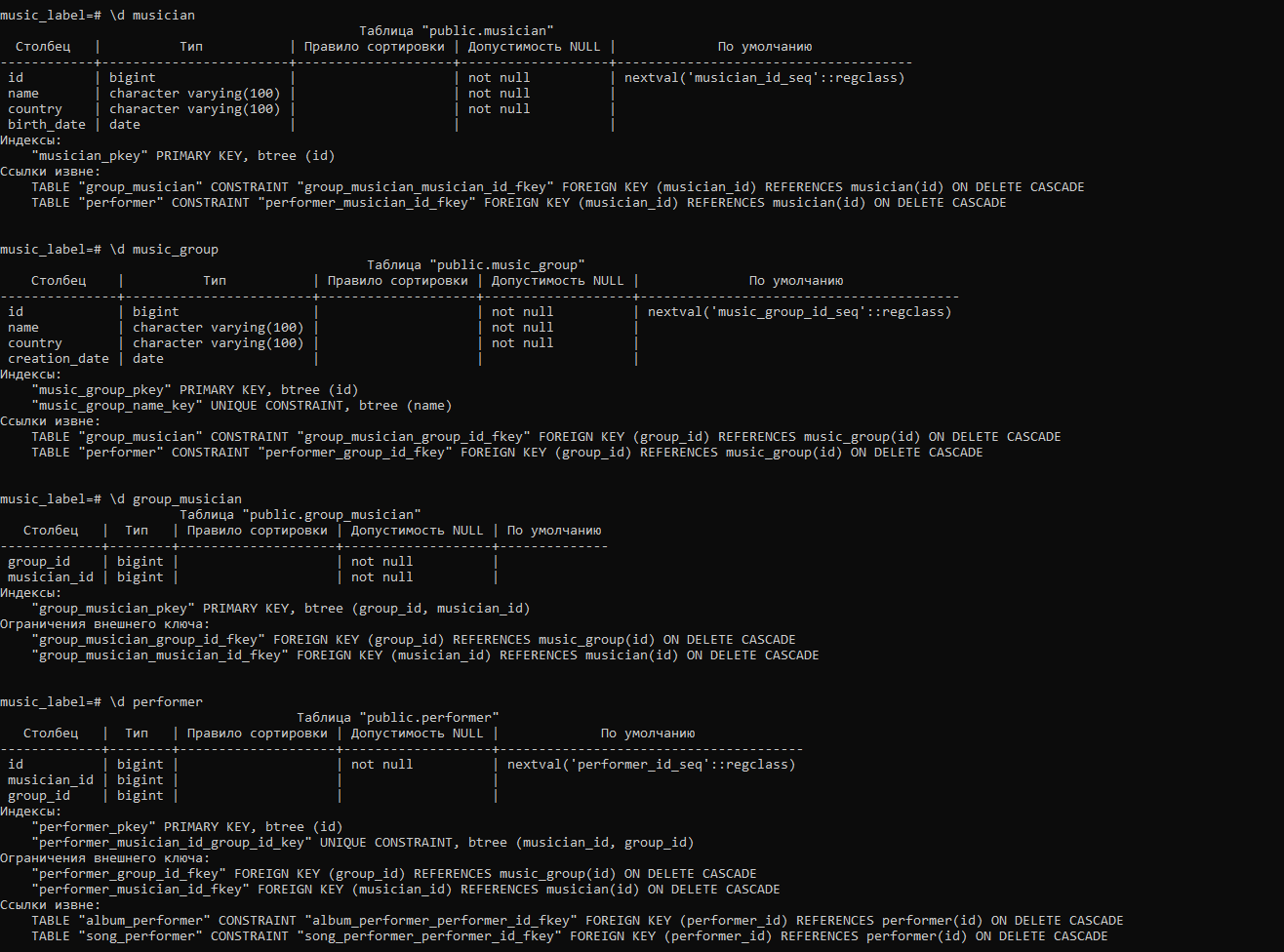


Рисунок 1.16.2 – Структура таблиц в базе данных

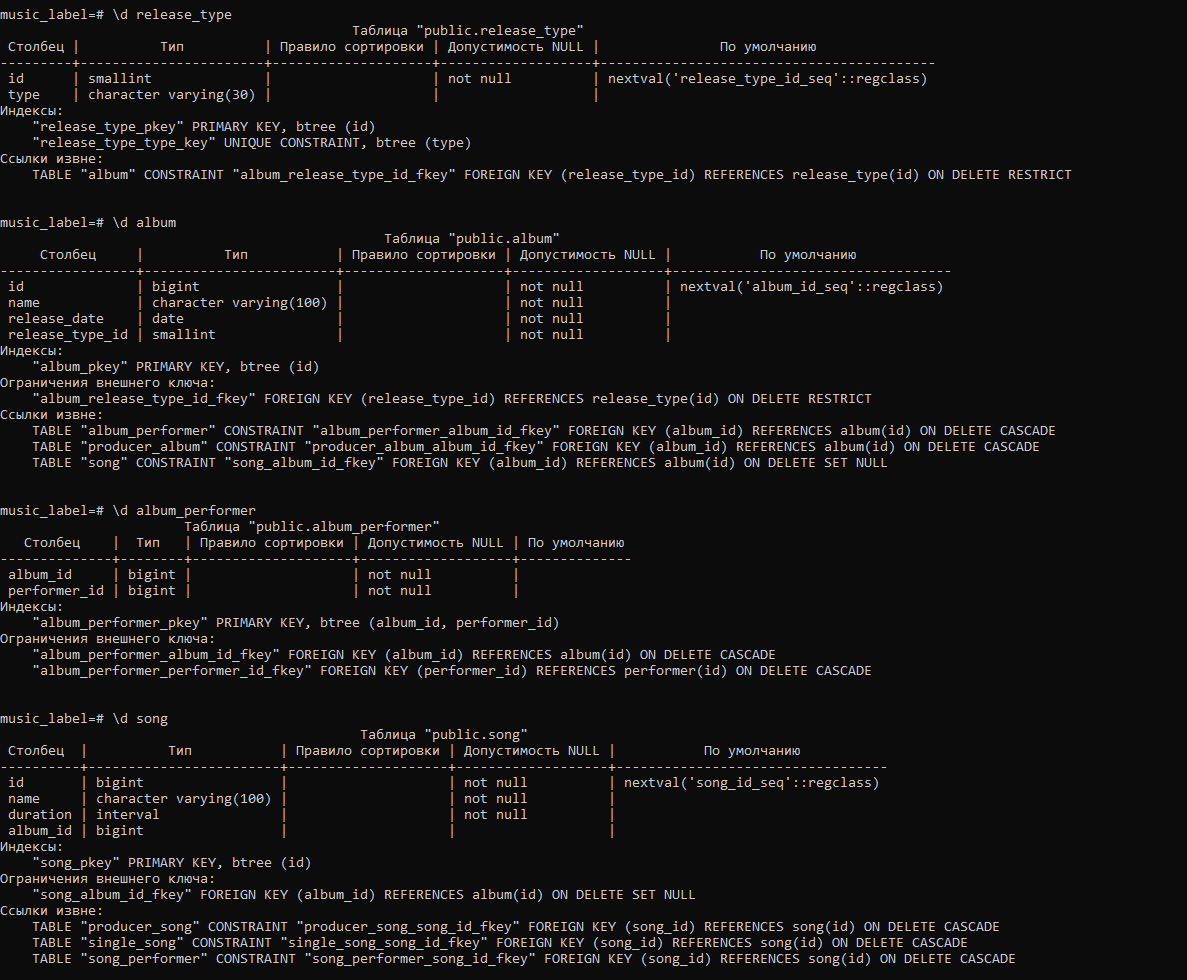


Рисунок 1.16.3 – Структура таблиц в базе данных

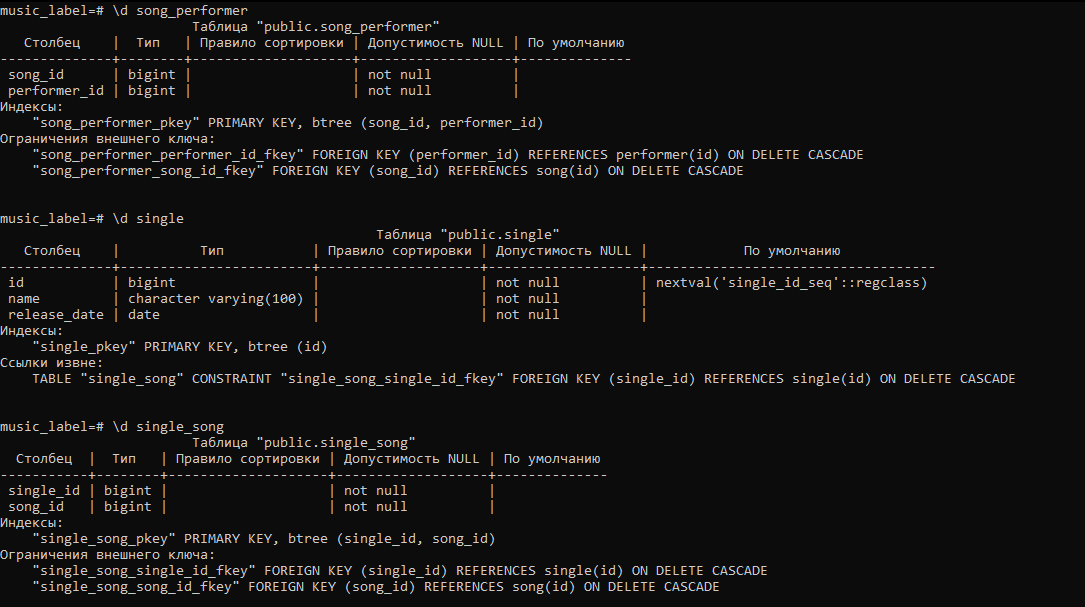


Рисунок 1.16.4 – Структура таблиц в базе данных

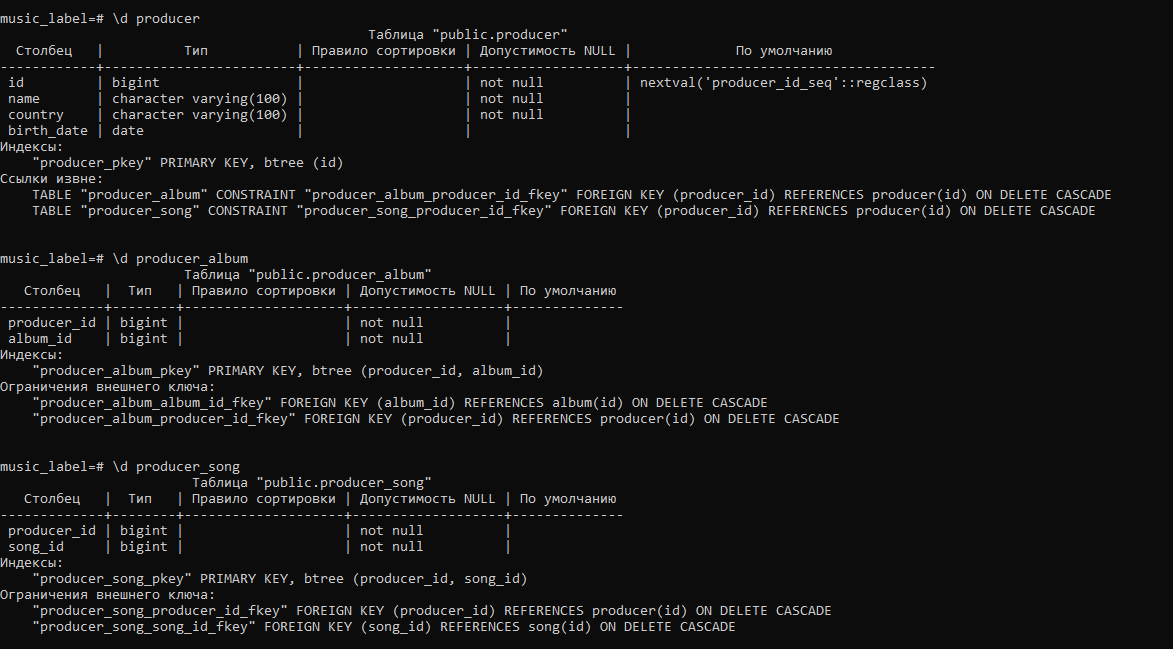


Рисунок 1.16.5 – Структура таблиц в базе данных

# 2 Практическая работа №2

**Цель:** Изучить и создать выборку и сортировку данных. Изучить и применить операторы для изменения данных в таблицах.

## 2.1 Выборка данных

* Оператор «=»

На Рисунке 2.1.1 представлен запрос на выборку записей из таблицы album, у которых значение поля id равно 2.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.1.1 – Выборка с оператором «=»

* Оператор «!=»

На Рисунке 2.1.2 представлен запрос на выборку записей из таблицы album, у которых значение поля release\_type\_id не равно 1.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.1.2 – Выборка с оператором «!=»

* Оператор «>»

На Рисунке 2.1.3 представлен запрос на выборку записей из таблицы song, у которых значение поля id больше 3.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 2.1.3 – Выборка с оператором «>»

* Оператор «>=»

На Рисунке 2.1.4 представлен запрос на выборку записей из таблицы song, у которых значение длительность больше или равна 6 минутам.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 2.1.4 – Выборка с оператором «>=»

* Оператор «<»

На Рисунке 2.1.5 представлен запрос на выборку записей из таблицы album, которые выпущены до 2000 года не включительно.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.1.5 – Выборка с оператором «<»

* Оператор «<=»

На Рисунке 2.1.6 представлен запрос на выборку записей из таблицы album, у которых значение поля release\_type\_id меньше или равно 2.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 2.1.6 – Выборка с оператором «<=»

* Оператор IS NOT NULL

На Рисунке 2.1.7 представлен запрос на выборку записей из таблицы song, у которых значение поля album\_id не пустое.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2.1.7 – Выборка с оператором IS NOT NULL

* Оператор IS NULL

На Рисунке 2.1.8 представлен запрос на выборку записей из таблицы song, у которых пустое значение album\_id.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.1.8 – Выборка с оператором IS NULL

* Оператор BETWEEN

На Рисунке 2.1.9 представлен запрос на выборку записей из таблицы song, у которых длительность от 4 до 6 минут.

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок 2.1.9 – Выборка с оператором BETWEEN

* Оператор IN

На Рисунке 2.1.10 представлен запрос на выборку записей из таблицы album, у которых значение поля id равно 2, 5, 7, 9 или 10.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.1.10 – Выборка с оператором IN

* Оператор NOT IN

На Рисунке 2.1.11 представлен запрос на выборку записей из таблицы music\_group, у которых значение поля id не равно 2, 3 или 5.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.1.11 – Выборка с оператором NOT IN

* Оператор LIKE

На Рисунке 2.1.12 представлен запрос на выборку записей из таблицы musician, где имя музыканта начинается на “Jo”.

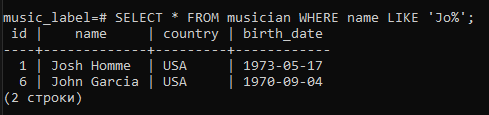


Рисунок 2.1.12 – Выборка с оператором LIKE

* Оператор NOT LIKE

На Рисунке 2.1.13 представлен запрос на выборку записей из таблицы album, в которых нет словосочетания “The”.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.1.13 – Выборка с оператором NOT LIKE

## 2.2 Выборка данных с сортировкой

На Рисунке 2.2.1 представлен запрос на выборку записей из таблицы song, отсортированную по возрастанию длительности, а на Рисунке 2.2.2 представлен запрос на выборку записей из таблицы album, отсортированную по полю name по убыванию.



Рисунок 2.2.1 – Выборка данных с сортировкой по возрастанию

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2.2.2 – Выборка данных с сортировкой по убыванию

## 2.3 Операторы изменения данных

* Изменение структуры таблицы

В таблицу album добавлено дополнительное поле duration, отвечающее за длительность альбома (тип – INTERVAL). Результат изменения представлен на Рисунке 2.3.1.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.3.1 – Изменение структуры таблицы

* Обновление данных в таблице

На Рисунке 2.3.2 представлен запрос на изменение записи в таблице album, у которой поле name равно ‘PULSE’ – полю duration присвоено значение ’00:06:35’.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2.3.2 – Обновление данных в таблице

* Удаление записей из таблицы

Для безопасного удаления записи сначала применён оператор BEGIN, открывающий транзакцию. Далее из таблицы musician удалена запись, где значение поля name начинается с “Kanye” (Рисунок 2.3.3). После этого сделана проверка, что удалена соответствующая запись в таблице musician, а также связанная с ней запись в таблице performer (Рисунок 2.3.4), а затем сделан откат транзакции (Рисунок 2.3.5).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.3.3 – Удаление записи внутри транзакции

Изображение выглядит как текст, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 2.3.4 – Проверка удаления

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Рисунок 2.3.5 – Откат транзакции

## 2.4 Результат выполнения работы

Все запросы, сделанные в ходе выполнения данной работы, представлены в Листинге Б.1.

# 3 Практическая работа №3

**Цель:** ознакомиться с тем, как выполняется перенос базы данных на другой сервер и её бэкап; выполнить запросы выборки, реализующие различные операции реляционной алгебры; создать хранимые процедуры, функции и триггеры.

## 3.1 Бэкап базы данных

Бэкап выполняется c помощью команды pg\_dump, в параметры командной строки передаются название базы данных, имя пользователя и название файла, в который будет сохранён бэкап (Рисунок 3.1.1).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.1.1 – Выполнение бэкапа базы данных

Содержимое файла backup.sql представлено в Листинге В.1.

## 3.2 Выборка данных согласно операциям реляционной алгебры

Все выполненные ниже запросы представлены в Листинге В.2.

### 3.2.1 Операция проекции

На Рисунке 3.2.1 представлен запрос на выборку названий всех альбомов.

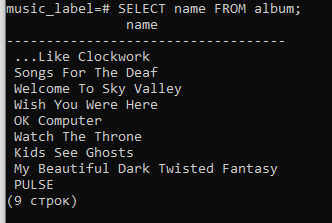


Рисунок 3.2.1 – Операция проекции

### 3.2.2 Операция селекции

На Рисунке 3.2.2 представлен запрос на выборку названий и длительностей песен, у которых длительность не меньше 6 минут.

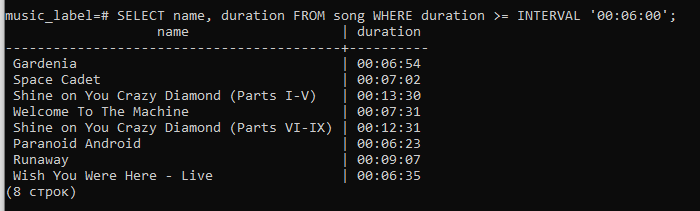


Рисунок 3.2.2 – Операция селекции

### 3.2.3 Операция соединения

На Рисунке 3.2.3 представлен запрос на выборку всех песен с выводом названий их соответствующих альбомов. На Рисунке 2.3.4 представлен этот же запрос, но с использованием JOIN.

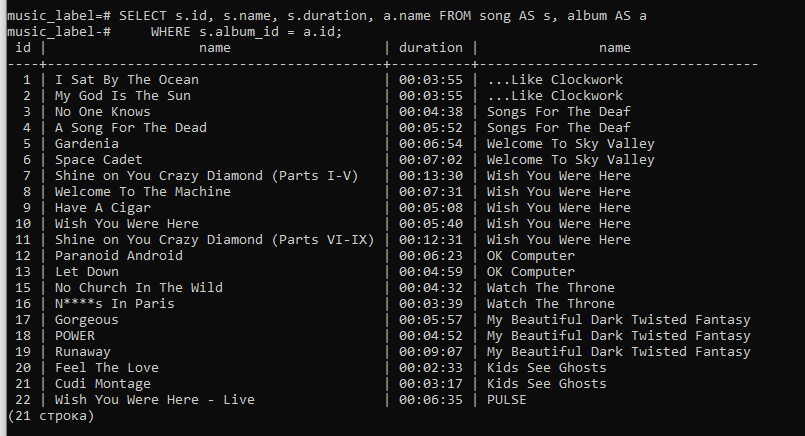


Рисунок 3.2.3 – Операция соединения

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3.2.4 – Операция соединения

### 3.2.4 Операция объединения

На Рисунке 3.2.5 представлен запрос на выборку музыкантов, которые состоят в группе Queens Of The Stone Age или Kyuss.

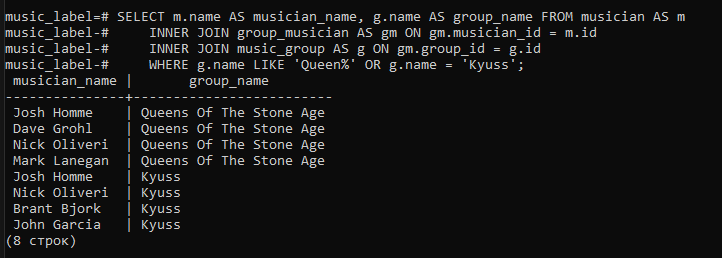


Рисунок 3.2.5 – Операция объединения

### 3.2.5 Операция пересечения

На Рисунке 3.2.6 представлен запрос на выборку музыкантов, состоящих одновременно в группе Queens Of The Stone Age и Kyuss.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3.2.6 – Операция пересечения

### 3.2.6 Операция разности

На Рисунке 3.2.7 представлен запрос на выборку песен, которые исполнены Kanye West строго без участия Kid Cudi.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.7 – Операция разности

### 3.2.7 Операция группировки

На Рисунке 3.2.8 представлен запрос на выборку альбомов с выводом длительности каждого из них, используя длительности связанных песен. На Рисунке 3.2.9 представлен этот же запрос, но с ограничением, что суммарная длительность должна быть больше 10 минут.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.2.8 – Операция группировки

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.2.9 – Операция группировки с условием

### 3.2.8 Операция сортировки

На Рисунке 3.2.10 представлен запрос на выборку альбомов в порядке убывания их длительности.

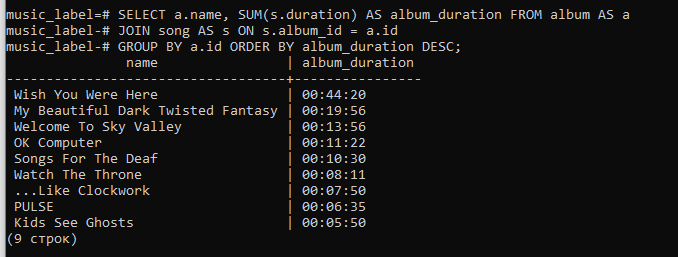


Рисунок 3.2.10 – Операция сортировки

### 3.2.9 Операция деления

На Рисунке 3.2.11 представлен запрос на получение продюсеров, которые связаны со всеми альбомами группы Queens Of The Stone Age. Чтобы уменьшить размер получившегося запроса, сделан подзапрос на получение альбомов, исполненных указанной группой, с помощью оператора WITH.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.11 – Операция деления

### 3.2.10 Создание представления

На Рисунке 3.2.12 представлено создание представления, содержащего в себе музыкантов-участников группы Kyuss, и на Рисунке 3.2.13 представлен запрос на выборку музыкантов из этого представления, имена которых начинаются на Jo.

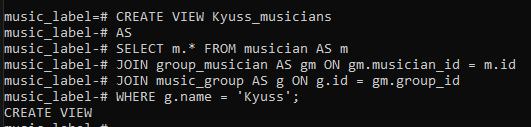


Рисунок 3.2.12 – Создание представления

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.13 – Выполнение выборки из представления

## 3.3 Хранимые процедуры, функции и триггеры

Коды всех процедур, функций и триггеров представлены в Листинге В.3.

### 3.3.1 Процедура

На Рисунке 3.3.1 представлена процедура, предназначенная для заполнения таблицы song\_performer, если песня входит в какой-то альбом. Записи в таблице song\_performer представлены на Рисунке 3.3.2, а результаты после вызова процедуры для песни “Cudi Montage” (id = 21) и появившиеся записи в таблице представлены на Рисунке 3.3.3.

В этой процедуре также есть курсор, позволяющий проходиться по записям из выборки.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.1 – Код процедуры

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.2 – Содержимое таблицы до вызова процедуры

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3.3.3 – Вызов процедуры и добавленные записи в таблицу

### 3.3.2 Функция

На Рисунке 3.3.4 представлена функция, вычисляющая возраст музыканта, принимая на вход id интересующего музыканта и возвращающая возраст музыканта, вычисленный на основе указанной даты рождения. На Рисунке 3.3.5 представлен запрос на выборку всех музыкантов с вызовом этой функции.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.4 – Код функции

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3.3.5 – Результат вызова функции

### 3.3.3 Триггер, отрабатывающий после добавления записи

На Рисунке 3.3.6 представлена процедура для триггера, которая будет обновлять длительность альбома при добавлении новой песни, связанной с этим альбомом, а на Рисунке 3.3.7 представлен сам триггер. Пример вызова триггера представлен на Рисунке 3.3.8.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.6 – Код процедуры для триггера

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.7 – Код триггера

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.8 – Результат вызова триггера

### 3.3.4 Триггер, отрабатывающий до добавления записи

На Рисунке 3.3.9 представлена процедура для триггера, которая будет проверять количество песен в альбоме при добавлении новой песни – если в альбоме уже есть 20 песен, то добавление новой записи станет невозможным, а на Рисунке 3.3.10 представлен сам триггер. Пример вызова триггера представлен на Рисунке 3.3.11 – добавление части записей проходит успешно, но как только количество песен в альбоме превысило 20, то запрос заблокировался.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.9 – Код процедуры для триггера

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.3.10 – Код триггера

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3.3.11 – Результат вызова триггера

# 4 Практическая работа №4

**Цель:** изучить синтаксис оконных функций и научиться их применять.

Все приведённые ниже запросы представлены в Листинге Г.1.

## 4.1 Агрегирующие функции

На Рисунке 4.1.1 представлен запрос на выборку альбомов с информацией о количестве песен, суммарной, средней, минимальной и максимальной продолжительностью песен.

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок 4.1.1 – Выборка с агрегирующими оконными функциями

## 4.2 Ранжирующие функции

На Рисунке 4.2.1 представлен запрос на выборку альбомов и песен в них, пронумерованных по порядку в альбоме.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.2.1 – Запрос с использованием ROW\_NUMBER

На Рисунке 4.2.2 представлен запрос с использованием RANK и DENSE\_RANK, где исполнители отсортированы по количеству исполненных песен в порядке невозрастания.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.2.2 – Запрос с использованием RANK и DENSE\_RANK

На Рисунке 4.2.3 представлен запрос с использованием NTILE, разделяющий песни в альбоме на две равные группы.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.2.3 – Запрос с использованием NTILE

## 4.3 Функции смещения

На Рисунке 4.3.1 представлен запрос, где рядом с каждой песней на альбоме выводится предыдущая и следующая за ней песня.

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 4.3.1 – Запрос с использованием LAG и LEAD

На Рисунке 4.3.2 представлен запрос, где для каждого альбома выводится первая, последняя и вторая песня в альбоме.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.3.2 – Запрос с использованием FIRST\_VALUE, LAST\_VALUE и NTH\_VALUE

# Вывод

В ходе выполнения практических работ освоены навыки работы с СУБД PostgreSQL, создана база данных со структурой, описанной в физической модели, сделаны запросы на её заполнение. Изучены операторы выборки и модификации данных, применение операторов выборки для выполнения операций реляционной алгебры. В базе данных созданы хранимые процедуры, функции и триггеры для упрощения работы. Также изучены оконные функции.

# Приложения

Приложение А – Скрипты из первой практической работы

Приложение Б – Скрипты из второй практической работы

Приложение В – Скрипты из третьей практической работы

Приложение Г – Скрипты из четвёртой практической работы

### Приложение А

Скрипты из первой практической работы

Листинг А.1 – Скрипт создания и заполнения таблиц в базе данных

CREATE DATABASE music\_label;

\c music\_label

CREATE TABLE musician (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL,

    country VARCHAR (100) NOT NULL,

    birth\_date DATE

);

CREATE TABLE music\_group (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL UNIQUE,

    country VARCHAR (100) NOT NULL,

    creation\_date DATE

);

CREATE TABLE performer (

    id BIGSERIAL PRIMARY KEY,

    musician\_id BIGINT REFERENCES musician(id) ON DELETE CASCADE,

    group\_id BIGINT REFERENCES music\_group(id) ON DELETE CASCADE,

    UNIQUE (musician\_id, group\_id)

);

CREATE TABLE group\_musician (

    group\_id BIGINT REFERENCES music\_group(id) ON DELETE CASCADE NOT NULL,

    musician\_id BIGINT REFERENCES musician(id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (group\_id, musician\_id)

);

CREATE TABLE producer (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL,

    country VARCHAR (100) NOT NULL,

    birth\_date DATE

);

CREATE TABLE release\_type (

    id SMALLSERIAL PRIMARY KEY,

    type VARCHAR (30) UNIQUE

);

CREATE TABLE album (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL,

    release\_date DATE NOT NULL,

    release\_type\_id SMALLINT REFERENCES release\_type(id) ON DELETE RESTRICT NOT NULL

);

CREATE TABLE album\_performer (

    album\_id BIGINT REFERENCES album (id) ON DELETE CASCADE NOT NULL,

    performer\_id BIGINT REFERENCES performer (id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (album\_id, performer\_id)

);

CREATE TABLE producer\_album (

    producer\_id BIGINT REFERENCES producer (id) ON DELETE CASCADE NOT NULL,

    album\_id BIGINT REFERENCES album (id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (producer\_id, album\_id)

);

CREATE TABLE song (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL,

Продолжение Листинга А.1

    duration INTERVAL NOT NULL,

    album\_id BIGINT REFERENCES album (id) ON DELETE SET NULL

);

CREATE TABLE song\_performer (

    song\_id BIGINT REFERENCES song (id) ON DELETE CASCADE NOT NULL,

    performer\_id BIGINT REFERENCES performer (id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (song\_id, performer\_id)

);

CREATE TABLE single (

    id BIGSERIAL PRIMARY KEY,

    name VARCHAR (100) NOT NULL,

    release\_date DATE NOT NULL

);

CREATE TABLE single\_song (

    single\_id BIGINT REFERENCES single (id) ON DELETE CASCADE NOT NULL,

    song\_id BIGINT REFERENCES song (id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (single\_id, song\_id)

);

CREATE TABLE producer\_song (

    producer\_id BIGINT REFERENCES producer (id) ON DELETE CASCADE NOT NULL,

    song\_id BIGINT REFERENCES song (id) ON DELETE CASCADE NOT NULL,

    PRIMARY KEY (producer\_id, song\_id)

);

INSERT INTO music\_group (name, country, creation\_date) VALUES

    ('Queens Of The Stone Age', 'USA', '1996-01-01'),

    ('Kyuss', 'USA', '1987-01-01'),

    ('Pink Floyd', 'Great Britain', '1965-01-01'),

    ('Radiohead', 'Great Britain', '1985-01-01'),

    ('Kids See Ghosts', 'USA', '2018-01-01');

INSERT INTO musician (name, country, birth\_date) VALUES

    ('Josh Homme', 'USA', '1973-05-17'),

    ('Dave Grohl', 'USA', '1969-01-14'),

    ('Nick Oliveri', 'USA', '1971-10-21'),

    ('Mark Lanegan', 'USA', '1964-11-25'),

    ('Brant Bjork', 'USA', '1973-03-19'),

    ('John Garcia', 'USA', '1970-09-04'),

    ('David Gilmour', 'Great Britain', '1946-03-06'),

    ('Roger Waters', 'Great Britain', '1943-09-06'),

    ('Thom Yorke', 'Great Britain', '1968-10-07'),

    ('Colin Greenwood', 'Great Britain', '1969-06-26'),

    ('Kanye West', 'USA', '1977-06-08'),

    ('JAY-Z', 'USA', '1969-12-04'),

    ('Frank Ocean', 'USA', '1987-10-28'),

    ('The-Dream', 'USA', '1977-09-20'),

    ('Pusha T', 'USA', '1977-05-13'),

    ('Kid Cudi', 'USA', '1984-01-30'),

    ('Raekwon', 'USA', '1970-01-12');

INSERT INTO performer (musician\_id, group\_id) VALUES

    (NULL, 1), (NULL, 2), (NULL, 3), (NULL, 4), (NULL, 5),

    (11, NULL), (12, NULL), (13, NULL), (14, NULL),

    (15, NULL), (16, NULL), (17, NULL);

INSERT INTO group\_musician (group\_id, musician\_id) VALUES

    (1, 1), (1, 2), (1, 3), (1, 4),

    (2, 1), (2, 3), (2, 5), (2, 6),

    (3, 7), (3, 8),

    (4, 9), (4, 10),

    (5, 11), (5, 16);

Продолжение Листинга А.1

INSERT INTO release\_type (type) VALUES ('LP'), ('EP'), ('Live'), ('Compilation');

INSERT INTO album (name, release\_date, release\_type\_id) VALUES

    ('...Like Clockwork', '2013-06-03', 1),

    ('Songs For The Deaf', '2002-08-27', 1),

    ('Welcome To Sky Valley', '1994-06-28', 1),

    ('Wish You Were Here', '1975-09-12', 1),

    ('OK Computer', '1997-05-21', 1),

    ('Watch The Throne', '2011-08-08', 1),

    ('Kids See Ghosts', '2018-06-08', 2),

    ('My Beautiful Dark Twisted Fantasy', '2010-11-22', 1),

    ('PULSE', '1995-05-28', 3);

INSERT INTO album\_performer (album\_id, performer\_id) VALUES

    (1, 1), (2, 1),

    (3, 2),

    (4, 3),

    (5, 4),

    (6, 6), (6, 7),

    (7, 5), (7, 6), (7, 11),

    (8, 6),

    (9, 3);

INSERT INTO producer (name, country, birth\_date) VALUES

    ('Josh Homme', 'USA', '1973-05-17'),

    ('Eric Valentine', 'USA', '1968-03-07'),

    ('Adam Kasper', 'USA', '1962-12-01'),

    ('Chris Goss', 'USA', '1959-08-17'),

    ('Roger Waters', 'Great Britain', '1943-09-06'),

    ('David Gilmour', 'Great Britain', '1946-03-06'),

    ('Nigel Godrich', 'Great Britain', '1971-02-28'),

    ('Kanye West', 'USA', '1977-06-08'),

    ('Kid Cudi', 'USA', '1984-01-30'),

    ('Mike Dean', 'USA', '1965-03-01');

INSERT INTO producer\_album (producer\_id, album\_id) VALUES

    (1, 1),

    (1, 2), (2, 2), (3, 2),

    (4, 3),

    (5, 4), (6, 4),

    (7, 5),

    (8, 6), (10, 6),

    (8, 7), (9, 7), (10, 7),

    (8, 8), (10, 8),

    (6, 9);

INSERT INTO song (name, duration, album\_id) VALUES

    ('I Sat By The Ocean', '00:03:55', 1),

    ('My God Is The Sun', '00:03:55', 1),

    ('No One Knows', '00:04:38', 2),

    ('A Song For The Dead', '00:05:52', 2),

    ('Gardenia', '00:06:54', 3),

    ('Space Cadet', '00:07:02', 3),

    ('Shine on You Crazy Diamond (Parts I-V)', '00:13:30', 4),

    ('Welcome To The Machine', '00:07:31', 4),

    ('Have A Cigar', '00:05:08', 4),

    ('Wish You Were Here', '00:05:40', 4),

    ('Shine on You Crazy Diamond (Parts VI-IX)', '00:12:31', 4),

    ('Paranoid Android', '00:06:23', 5),

Окончание Листинга А.1

    ('Let Down', '00:04:59', 5),

    ('Pearly', '00:03:33', NULL),

    ('No Church In The Wild', '00:04:32', 6),

    ('N\*\*\*\*s In Paris', '00:03:39', 6),

    ('Gorgeous', '00:05:57', 8),

    ('POWER', '00:04:52', 8),

    ('Runaway', '00:09:07', 8),

    ('Feel The Love', '00:02:33', 7),

    ('Cudi Montage', '00:03:17', 7),

    ('Wish You Were Here - Live', '00:06:35', 9);

INSERT INTO song\_performer (song\_id, performer\_id) VALUES

    (14, 4),

    (15, 8), (15, 9),

    (17, 11), (17, 12),

    (19, 10),

    (20, 10);

INSERT INTO producer\_song (producer\_id, song\_id) VALUES (7, 14);

INSERT INTO single (name, release\_date) VALUES

    ('My God Is The Sun', '2013-04-08'),

    ('No One Knows', '2002-11-26'),

    ('Gardenia', '1995-01-01'),

    ('Have A Cigar', '1975-11-15'),

    ('Paranoid Android', '1997-05-26'),

    ('No Church In The Wild', '2012-03-20'),

    ('N\*\*\*\*s In Paris', '2011-09-13'),

    ('POWER', '2010-07-01'),

    ('Runaway', '2010-10-04');

INSERT INTO single\_song (single\_id, song\_id) VALUES

    (1, 1),

    (2, 3),

    (3, 5),

    (4, 9), (4, 8),

    (5, 12), (5, 13), (5, 14),

    (6, 15),

    (7, 16),

    (8, 18),

    (9, 19);

### Приложение Б

Скрипты из второй практической работы

Листинг Б.1 – Скрипты выборки и модификации

SELECT \* FROM album WHERE id = 2;

SELECT \* FROM song WHERE id > 3;

SELECT \* FROM album WHERE release\_date < '2000-01-01';

SELECT \* FROM song WHERE duration >= INTERVAL '6 MINUTES';

SELECT \* FROM album WHERE release\_type\_id <= 2;

SELECT \* FROM album WHERE release\_type\_id != 1;

SELECT \* FROM song WHERE album\_id IS NOT NULL;

SELECT \* FROM song WHERE album\_id IS NULL;

SELECT \* FROM song WHERE duration BETWEEN INTERVAL '4 MINUTES' AND INTERVAL '6 MINUTES';

SELECT \* FROM album WHERE id IN (2, 5, 7, 9, 10);

SELECT \* FROM music\_group WHERE id NOT IN (2, 3, 5);

SELECT \* FROM musician WHERE name LIKE 'Jo%';

SELECT \* FROM album WHERE name NOT LIKE '%The%';

SELECT \* FROM song ORDER BY duration;

SELECT \* FROM album ORDER BY name DESC;

ALTER TABLE album ADD COLUMN duration INTERVAL;

UPDATE album SET duration = INTERVAL '00:06:35' WHERE name = 'PULSE';

BEGIN;

DELETE FROM musician WHERE name LIKE 'Kanye%';

ROLLBACK;

### Приложение В

Скрипты из третьей практической работы

Листинг В.1 – Скрипт backup.sql

--

-- PostgreSQL database dump

--

-- Dumped from database version 16.4

-- Dumped by pg\_dump version 16.4

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- Name: album; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.album (

    id bigint NOT NULL,

    name character varying(100) NOT NULL,

    release\_date date NOT NULL,

    release\_type\_id smallint NOT NULL,

    duration interval

);

ALTER TABLE public.album OWNER TO postgres;

--

-- Name: album\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.album\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.album\_id\_seq OWNER TO postgres;

--

-- Name: album\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.album\_id\_seq OWNED BY public.album.id;

Продолжение Листинга В.1

--

-- Name: album\_performer; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.album\_performer (

    album\_id bigint NOT NULL,

    performer\_id bigint NOT NULL

);

ALTER TABLE public.album\_performer OWNER TO postgres;

--

-- Name: group\_musician; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.group\_musician (

    group\_id bigint NOT NULL,

    musician\_id bigint NOT NULL

);

ALTER TABLE public.group\_musician OWNER TO postgres;

--

-- Name: music\_group; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.music\_group (

    id bigint NOT NULL,

    name character varying(100) NOT NULL,

    country character varying(100) NOT NULL,

    creation\_date date

);

ALTER TABLE public.music\_group OWNER TO postgres;

--

-- Name: music\_group\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.music\_group\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.music\_group\_id\_seq OWNER TO postgres;

--

-- Name: music\_group\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.music\_group\_id\_seq OWNED BY public.music\_group.id;

--

-- Name: musician; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.musician (

    id bigint NOT NULL,

Продолжение Листинга В.1

    name character varying(100) NOT NULL,

    country character varying(100) NOT NULL,

    birth\_date date

);

ALTER TABLE public.musician OWNER TO postgres;

--

-- Name: musician\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.musician\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.musician\_id\_seq OWNER TO postgres;

--

-- Name: musician\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.musician\_id\_seq OWNED BY public.musician.id;

--

-- Name: performer; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.performer (

    id bigint NOT NULL,

    musician\_id bigint,

    group\_id bigint

);

ALTER TABLE public.performer OWNER TO postgres;

--

-- Name: performer\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.performer\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.performer\_id\_seq OWNER TO postgres;

--

-- Name: performer\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.performer\_id\_seq OWNED BY public.performer.id;

--

-- Name: producer; Type: TABLE; Schema: public; Owner: postgres

Продолжение Листинга В.1

--

CREATE TABLE public.producer (

    id bigint NOT NULL,

    name character varying(100) NOT NULL,

    country character varying(100) NOT NULL,

    birth\_date date

);

ALTER TABLE public.producer OWNER TO postgres;

--

-- Name: producer\_album; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.producer\_album (

    producer\_id bigint NOT NULL,

    album\_id bigint NOT NULL

);

ALTER TABLE public.producer\_album OWNER TO postgres;

--

-- Name: producer\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.producer\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.producer\_id\_seq OWNER TO postgres;

--

-- Name: producer\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.producer\_id\_seq OWNED BY public.producer.id;

--

-- Name: producer\_song; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.producer\_song (

    producer\_id bigint NOT NULL,

    song\_id bigint NOT NULL

);

ALTER TABLE public.producer\_song OWNER TO postgres;

--

-- Name: release\_type; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.release\_type (

    id smallint NOT NULL,

    type character varying(30)

);

Продолжение Листинга В.1

ALTER TABLE public.release\_type OWNER TO postgres;

--

-- Name: release\_type\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.release\_type\_id\_seq

    AS smallint

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.release\_type\_id\_seq OWNER TO postgres;

--

-- Name: release\_type\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.release\_type\_id\_seq OWNED BY public.release\_type.id;

--

-- Name: single; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.single (

    id bigint NOT NULL,

    name character varying(100) NOT NULL,

    release\_date date NOT NULL

);

ALTER TABLE public.single OWNER TO postgres;

--

-- Name: single\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.single\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.single\_id\_seq OWNER TO postgres;

--

-- Name: single\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.single\_id\_seq OWNED BY public.single.id;

--

-- Name: single\_song; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.single\_song (

Продолжение Листинга В.1

    single\_id bigint NOT NULL,

    song\_id bigint NOT NULL

);

ALTER TABLE public.single\_song OWNER TO postgres;

--

-- Name: song; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.song (

    id bigint NOT NULL,

    name character varying(100) NOT NULL,

    duration interval NOT NULL,

    album\_id bigint

);

ALTER TABLE public.song OWNER TO postgres;

--

-- Name: song\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.song\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public.song\_id\_seq OWNER TO postgres;

--

-- Name: song\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.song\_id\_seq OWNED BY public.song.id;

--

-- Name: song\_performer; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.song\_performer (

    song\_id bigint NOT NULL,

    performer\_id bigint NOT NULL

);

ALTER TABLE public.song\_performer OWNER TO postgres;

--

-- Name: album id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.album ALTER COLUMN id SET DEFAULT nextval('public.album\_id\_seq'::regclass);

--

-- Name: music\_group id; Type: DEFAULT; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

ALTER TABLE ONLY public.music\_group ALTER COLUMN id SET DEFAULT nextval('public.music\_group\_id\_seq'::regclass);

--

-- Name: musician id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.musician ALTER COLUMN id SET DEFAULT nextval('public.musician\_id\_seq'::regclass);

--

-- Name: performer id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.performer ALTER COLUMN id SET DEFAULT nextval('public.performer\_id\_seq'::regclass);

--

-- Name: producer id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer ALTER COLUMN id SET DEFAULT nextval('public.producer\_id\_seq'::regclass);

--

-- Name: release\_type id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.release\_type ALTER COLUMN id SET DEFAULT nextval('public.release\_type\_id\_seq'::regclass);

--

-- Name: single id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.single ALTER COLUMN id SET DEFAULT nextval('public.single\_id\_seq'::regclass);

--

-- Name: song id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.song ALTER COLUMN id SET DEFAULT nextval('public.song\_id\_seq'::regclass);

--

-- Data for Name: album; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.album (id, name, release\_date, release\_type\_id, duration) FROM stdin;

1   ...Like Clockwork   2013-06-03  1   \N

2   Songs For The Deaf  2002-08-27  1   \N

3   Welcome To Sky Valley   1994-06-28  1   \N

4   Wish You Were Here  1975-09-12  1   \N

5   OK Computer 1997-05-21  1   \N

6   Watch The Throne    2011-08-08  1   \N

7   Kids See Ghosts 2018-06-08  2   \N

8   My Beautiful Dark Twisted Fantasy   2010-11-22  1   \N

9   PULSE   1995-05-28  3   00:06:35

\.

Продолжение Листинга В.1

--

-- Data for Name: album\_performer; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.album\_performer (album\_id, performer\_id) FROM stdin;

1   1

2   1

3   2

4   3

5   4

6   6

6   7

7   5

7   6

7   11

8   6

9   3

\.

--

-- Data for Name: group\_musician; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.group\_musician (group\_id, musician\_id) FROM stdin;

1   1

1   2

1   3

1   4

2   1

2   3

2   5

2   6

3   7

3   8

4   9

4   10

5   11

5   16

\.

--

-- Data for Name: music\_group; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.music\_group (id, name, country, creation\_date) FROM stdin;

1   Queens Of The Stone Age USA 1996-01-01

2   Kyuss   USA 1987-01-01

3   Pink Floyd  Great Britain   1965-01-01

4   Radiohead   Great Britain   1985-01-01

5   Kids See Ghosts USA 2018-01-01

\.

--

-- Data for Name: musician; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.musician (id, name, country, birth\_date) FROM stdin;

Продолжение Листинга В.1

1   Josh Homme  USA 1973-05-17

2   Dave Grohl  USA 1969-01-14

3   Nick Oliveri    USA 1971-10-21

4   Mark Lanegan    USA 1964-11-25

5   Brant Bjork USA 1973-03-19

6   John Garcia USA 1970-09-04

7   David Gilmour   Great Britain   1946-03-06

8   Roger Waters    Great Britain   1943-09-06

9   Thom Yorke  Great Britain   1968-10-07

10  Colin Greenwood Great Britain   1969-06-26

11  Kanye West  USA 1977-06-08

12  JAY-Z   USA 1969-12-04

13  Frank Ocean USA 1987-10-28

14  The-Dream   USA 1977-09-20

15  Pusha T USA 1977-05-13

16  Kid Cudi    USA 1984-01-30

17  Raekwon USA 1970-01-12

\.

--

-- Data for Name: performer; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.performer (id, musician\_id, group\_id) FROM stdin;

1   \N  1

2   \N  2

3   \N  3

4   \N  4

5   \N  5

6   11  \N

7   12  \N

8   13  \N

9   14  \N

10  15  \N

11  16  \N

12  17  \N

\.

--

-- Data for Name: producer; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.producer (id, name, country, birth\_date) FROM stdin;

1   Josh Homme  USA 1973-05-17

2   Eric Valentine  USA 1968-03-07

3   Adam Kasper USA 1962-12-01

4   Chris Goss  USA 1959-08-17

5   Roger Waters    Great Britain   1943-09-06

6   David Gilmour   Great Britain   1946-03-06

7   Nigel Godrich   Great Britain   1971-02-28

8   Kanye West  USA 1977-06-08

9   Kid Cudi    USA 1984-01-30

10  Mike Dean   USA 1965-03-01

\.

--

-- Data for Name: producer\_album; Type: TABLE DATA; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

COPY public.producer\_album (producer\_id, album\_id) FROM stdin;

1   1

1   2

2   2

3   2

4   3

5   4

6   4

7   5

8   6

10  6

8   7

9   7

10  7

8   8

10  8

6   9

\.

--

-- Data for Name: producer\_song; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.producer\_song (producer\_id, song\_id) FROM stdin;

7   14

\.

--

-- Data for Name: release\_type; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.release\_type (id, type) FROM stdin;

1   LP

2   EP

3   Live

4   Compilation

\.

--

-- Data for Name: single; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.single (id, name, release\_date) FROM stdin;

1   My God Is The Sun   2013-04-08

2   No One Knows    2002-11-26

3   Gardenia    1995-01-01

4   Have A Cigar    1975-11-15

5   Paranoid Android    1997-05-26

6   No Church In The Wild   2012-03-20

7   N\*\*\*\*s In Paris 2011-09-13

8   POWER   2010-07-01

9   Runaway 2010-10-04

\.

--

-- Data for Name: single\_song; Type: TABLE DATA; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

COPY public.single\_song (single\_id, song\_id) FROM stdin;

1   1

2   3

3   5

4   9

4   8

5   12

5   13

5   14

6   15

7   16

8   18

9   19

\.

--

-- Data for Name: song; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.song (id, name, duration, album\_id) FROM stdin;

1   I Sat By The Ocean  00:03:55    1

2   My God Is The Sun   00:03:55    1

3   No One Knows    00:04:38    2

4   A Song For The Dead 00:05:52    2

5   Gardenia    00:06:54    3

6   Space Cadet 00:07:02    3

7   Shine on You Crazy Diamond (Parts I-V)  00:13:30    4

8   Welcome To The Machine  00:07:31    4

9   Have A Cigar    00:05:08    4

10  Wish You Were Here  00:05:40    4

11  Shine on You Crazy Diamond (Parts VI-IX)    00:12:31    4

12  Paranoid Android    00:06:23    5

13  Let Down    00:04:59    5

14  Pearly  00:03:33    \N

15  No Church In The Wild   00:04:32    6

16  N\*\*\*\*s In Paris 00:03:39    6

17  Gorgeous    00:05:57    8

18  POWER   00:04:52    8

19  Runaway 00:09:07    8

20  Feel The Love   00:02:33    7

21  Cudi Montage    00:03:17    7

22  Wish You Were Here - Live   00:06:35    9

\.

--

-- Data for Name: song\_performer; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.song\_performer (song\_id, performer\_id) FROM stdin;

14  4

15  8

15  9

17  11

17  12

19  10

20  10

\.

--

-- Name: album\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

Продолжение Листинга В.1

--

SELECT pg\_catalog.setval('public.album\_id\_seq', 9, true);

--

-- Name: music\_group\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.music\_group\_id\_seq', 5, true);

--

-- Name: musician\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.musician\_id\_seq', 17, true);

--

-- Name: performer\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.performer\_id\_seq', 12, true);

--

-- Name: producer\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.producer\_id\_seq', 10, true);

--

-- Name: release\_type\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.release\_type\_id\_seq', 4, true);

--

-- Name: single\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.single\_id\_seq', 9, true);

--

-- Name: song\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.song\_id\_seq', 22, true);

--

-- Name: album\_performer album\_performer\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.album\_performer

    ADD CONSTRAINT album\_performer\_pkey PRIMARY KEY (album\_id, performer\_id);

--

-- Name: album album\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

ALTER TABLE ONLY public.album

    ADD CONSTRAINT album\_pkey PRIMARY KEY (id);

--

-- Name: group\_musician group\_musician\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.group\_musician

    ADD CONSTRAINT group\_musician\_pkey PRIMARY KEY (group\_id, musician\_id);

--

-- Name: music\_group music\_group\_name\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.music\_group

    ADD CONSTRAINT music\_group\_name\_key UNIQUE (name);

--

-- Name: music\_group music\_group\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.music\_group

    ADD CONSTRAINT music\_group\_pkey PRIMARY KEY (id);

--

-- Name: musician musician\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.musician

    ADD CONSTRAINT musician\_pkey PRIMARY KEY (id);

--

-- Name: performer performer\_musician\_id\_group\_id\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.performer

    ADD CONSTRAINT performer\_musician\_id\_group\_id\_key UNIQUE (musician\_id, group\_id);

--

-- Name: performer performer\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.performer

    ADD CONSTRAINT performer\_pkey PRIMARY KEY (id);

--

-- Name: producer\_album producer\_album\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_album

    ADD CONSTRAINT producer\_album\_pkey PRIMARY KEY (producer\_id, album\_id);

--

Продолжение Листинга В.1

-- Name: producer producer\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer

    ADD CONSTRAINT producer\_pkey PRIMARY KEY (id);

--

-- Name: producer\_song producer\_song\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_song

    ADD CONSTRAINT producer\_song\_pkey PRIMARY KEY (producer\_id, song\_id);

--

-- Name: release\_type release\_type\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.release\_type

    ADD CONSTRAINT release\_type\_pkey PRIMARY KEY (id);

--

-- Name: release\_type release\_type\_type\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.release\_type

    ADD CONSTRAINT release\_type\_type\_key UNIQUE (type);

--

-- Name: single single\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.single

    ADD CONSTRAINT single\_pkey PRIMARY KEY (id);

--

-- Name: single\_song single\_song\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.single\_song

    ADD CONSTRAINT single\_song\_pkey PRIMARY KEY (single\_id, song\_id);

--

-- Name: song\_performer song\_performer\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.song\_performer

    ADD CONSTRAINT song\_performer\_pkey PRIMARY KEY (song\_id, performer\_id);

--

-- Name: song song\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.song

    ADD CONSTRAINT song\_pkey PRIMARY KEY (id);

Продолжение Листинга В.1

--

-- Name: album\_performer album\_performer\_album\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.album\_performer

    ADD CONSTRAINT album\_performer\_album\_id\_fkey FOREIGN KEY (album\_id) REFERENCES public.album(id) ON DELETE CASCADE;

--

-- Name: album\_performer album\_performer\_performer\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.album\_performer

    ADD CONSTRAINT album\_performer\_performer\_id\_fkey FOREIGN KEY (performer\_id) REFERENCES public.performer(id) ON DELETE CASCADE;

--

-- Name: album album\_release\_type\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.album

    ADD CONSTRAINT album\_release\_type\_id\_fkey FOREIGN KEY (release\_type\_id) REFERENCES public.release\_type(id) ON DELETE RESTRICT;

--

-- Name: group\_musician group\_musician\_group\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.group\_musician

    ADD CONSTRAINT group\_musician\_group\_id\_fkey FOREIGN KEY (group\_id) REFERENCES public.music\_group(id) ON DELETE CASCADE;

--

-- Name: group\_musician group\_musician\_musician\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.group\_musician

    ADD CONSTRAINT group\_musician\_musician\_id\_fkey FOREIGN KEY (musician\_id) REFERENCES public.musician(id) ON DELETE CASCADE;

--

-- Name: performer performer\_group\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.performer

    ADD CONSTRAINT performer\_group\_id\_fkey FOREIGN KEY (group\_id) REFERENCES public.music\_group(id) ON DELETE CASCADE;

--

-- Name: performer performer\_musician\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.performer

Продолжение Листинга В.1

    ADD CONSTRAINT performer\_musician\_id\_fkey FOREIGN KEY (musician\_id) REFERENCES public.musician(id) ON DELETE CASCADE;

--

-- Name: producer\_album producer\_album\_album\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_album

    ADD CONSTRAINT producer\_album\_album\_id\_fkey FOREIGN KEY (album\_id) REFERENCES public.album(id) ON DELETE CASCADE;

--

-- Name: producer\_album producer\_album\_producer\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_album

    ADD CONSTRAINT producer\_album\_producer\_id\_fkey FOREIGN KEY (producer\_id) REFERENCES public.producer(id) ON DELETE CASCADE;

--

-- Name: producer\_song producer\_song\_producer\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_song

    ADD CONSTRAINT producer\_song\_producer\_id\_fkey FOREIGN KEY (producer\_id) REFERENCES public.producer(id) ON DELETE CASCADE;

--

-- Name: producer\_song producer\_song\_song\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.producer\_song

    ADD CONSTRAINT producer\_song\_song\_id\_fkey FOREIGN KEY (song\_id) REFERENCES public.song(id) ON DELETE CASCADE;

--

-- Name: single\_song single\_song\_single\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.single\_song

    ADD CONSTRAINT single\_song\_single\_id\_fkey FOREIGN KEY (single\_id) REFERENCES public.single(id) ON DELETE CASCADE;

--

-- Name: single\_song single\_song\_song\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.single\_song

    ADD CONSTRAINT single\_song\_song\_id\_fkey FOREIGN KEY (song\_id) REFERENCES public.song(id) ON DELETE CASCADE;

--

-- Name: song song\_album\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

Окончание Листинга В.1

ALTER TABLE ONLY public.song

    ADD CONSTRAINT song\_album\_id\_fkey FOREIGN KEY (album\_id) REFERENCES public.album(id) ON DELETE SET NULL;

--

-- Name: song\_performer song\_performer\_performer\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.song\_performer

    ADD CONSTRAINT song\_performer\_performer\_id\_fkey FOREIGN KEY (performer\_id) REFERENCES public.performer(id) ON DELETE CASCADE;

--

-- Name: song\_performer song\_performer\_song\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.song\_performer

    ADD CONSTRAINT song\_performer\_song\_id\_fkey FOREIGN KEY (song\_id) REFERENCES public.song(id) ON DELETE CASCADE;

--

-- PostgreSQL database dump complete

--

Листинг В.2 – Операции реляционной алгебры

-- Операции выборки

-- #1 Операция проекции - вывод нахваний всех альбомов

SELECT name FROM album;

-- #2 Операция селекции - вывод песен, которые не короче 6 минут

SELECT name, duration FROM song WHERE duration >= INTERVAL '00:06:00';

-- #3 Операция соединения - вывод названия альбома у песен

SELECT s.id, s.name, s.duration, a.name FROM song AS s, album AS a

    WHERE s.album\_id = a.id;

-- #3.1 То же самое, но с JOIN

SELECT s.id, s.name, s.duration, a.name FROM song AS s

    INNER JOIN album AS a

    ON s.album\_id = a.id;

-- #4 Операция объединения - вывод музыкантов, состоящих в группе QOTSA или Kyuss

SELECT m.name AS musician\_name, g.name AS group\_name FROM musician AS m

    INNER JOIN group\_musician AS gm ON gm.musician\_id = m.id

    INNER JOIN music\_group AS g ON gm.group\_id = g.id

    WHERE g.name LIKE 'Queen%' OR g.name = 'Kyuss';

-- #5 Операция пересечения - вывод музыкантов, состоящих одновременно в группе QOTSA и Kyuss

SELECT m.name AS musician\_name

    FROM musician AS m

    INNER JOIN group\_musician AS gm ON gm.musician\_id = m.id

    INNER JOIN music\_group AS g ON gm.group\_id = g.id

    WHERE g.name LIKE 'Queen%' AND EXISTS (

        SELECT \* FROM group\_musician AS gm1

Продолжение Листинга В.2

        INNER JOIN music\_group AS g1 ON g1.id = gm1.group\_id

        WHERE gm1.musician\_id = m.id AND g1.name = 'Kyuss'

    );

-- #6 Гигантский пример на операцию разности

-- Вывод песен Kanye West без участия Kid Cudi

SELECT s.name AS song\_name,

        CASE WHEN (m.name, g.name) IS NULL THEN NULL

             ELSE concat(m.name, g.name) END AS performer\_name

FROM song AS s

JOIN song\_performer AS sp ON sp.song\_id = s.id

JOIN performer AS p ON p.id = sp.performer\_id

LEFT OUTER JOIN musician AS m ON m.id = p.musician\_id

LEFT OUTER JOIN music\_group AS g ON g.id = p.group\_id

WHERE m.name LIKE 'Kanye%' AND NOT EXISTS (

    SELECT \* FROM song AS s1

    JOIN song\_performer AS sp ON sp.song\_id = s.id

    JOIN performer AS p ON p.id = sp.performer\_id

    LEFT OUTER JOIN musician AS m ON m.id = p.musician\_id

    LEFT OUTER JOIN music\_group AS g ON g.id = p.group\_id

    WHERE s1.id = s.id AND m.name LIKE 'Kid%'

)

ORDER BY s.id, g.id, m.id;

-- #7 Операция группировки - вывод длины каждого альбома

SELECT a.name, SUM(s.duration) AS album\_duration FROM album AS a

JOIN song AS s ON s.album\_id = a.id

GROUP BY a.id;

-- #7.1 Операция группировки c HAVING - вывод альбомов с длиной не менее 10 минут

SELECT a.name, SUM(s.duration) AS album\_duration FROM album AS a

JOIN song AS s ON s.album\_id = a.id

GROUP BY a.id HAVING SUM(s.duration) >= INTERVAL '00:10:00';

-- #8 Операция сортировки - вывод альбомов в порядке убывания их длины

SELECT a.name, SUM(s.duration) AS album\_duration FROM album AS a

JOIN song AS s ON s.album\_id = a.id

GROUP BY a.id ORDER BY album\_duration DESC;

-- #9 Чуть менее ужасаюащий пример операции деления (по сравнению с предыдущей вариацией)

-- вывести продюсера(-ов), который(-е) продюссировал(-и) все альбомы Queens Of The Stone Age

-- Представление, в котором сведения только о тех альбомах, которые исполнены QOTSA

WITH qotsa\_albums

AS

(SELECT a.\* FROM album AS a

JOIN album\_performer AS ap ON ap.album\_id = a.id

JOIN performer AS p ON p.id = ap.performer\_id

JOIN music\_group AS g ON g.id = p.group\_id

WHERE g.name LIKE 'Queen%')

-- Сам запрос

SELECT pr.name, a.name FROM producer AS pr

JOIN producer\_album AS pra ON pra.producer\_id = pr.id

JOIN qotsa\_albums AS a ON a.id = pra.album\_id

AND NOT EXISTS (

    SELECT pr1.name, a1.name FROM producer AS pr1, qotsa\_albums AS a1

    WHERE NOT EXISTS (

        SELECT 1 FROM producer AS pr2

Окончание Листинга В.2

        JOIN producer\_album AS pra1 ON pra1.producer\_id = pr2.id

        JOIN album AS a2 ON a2.id = pra1.album\_id

        WHERE pr2.id = pr1.id AND a2.id = a1.id

    )

    AND pr1.id = pr.id

);

-- #10 Создание представлений - запрос на участников Kyuss

CREATE VIEW Kyuss\_musicians

AS

SELECT m.\* FROM musician AS m

JOIN group\_musician AS gm ON gm.musician\_id = m.id

JOIN music\_group AS g ON g.id = gm.group\_id

WHERE g.name = 'Kyuss';

-- Выбор музыкантов с именами на Jo.. из этого представления

SELECT name FROM Kyuss\_musicians WHERE name LIKE 'Jo%';

Листинг В.3 – Хранимые процедуры, функции и триггеры

-- #1 Процедура добавления песни к исполнителю

CREATE PROCEDURE add\_song\_to\_performer (song\_id INTEGER)

AS $$

DECLARE performer\_id INTEGER;

DECLARE performer\_cursor CURSOR FOR

    SELECT ap.performer\_id FROM song AS s

    JOIN album AS a ON a.id = s.album\_id

    JOIN album\_performer AS ap ON ap.album\_id = a.id

    WHERE s.id = song\_id;

BEGIN

    OPEN performer\_cursor;

    LOOP

        FETCH performer\_cursor INTO performer\_id;

        IF NOT FOUND THEN EXIT; END IF;

        INSERT INTO song\_performer (song\_id, performer\_id)

        VALUES ($1, performer\_id);

    END LOOP;

    CLOSE performer\_cursor;

END;

$$ LANGUAGE plpgsql;

-- #2 Функция, вычисляющая возраст музыканта

CREATE FUNCTION musician\_age (musician\_id BIGINT)

RETURNS SMALLINT AS $$

DECLARE age SMALLINT;

BEGIN

    SELECT EXTRACT(years FROM AGE(

        NOW(),

        (SELECT birth\_date FROM musician

            WHERE id = musician\_id)

    )) INTO age;

    RETURN age;

END;

$$ LANGUAGE plpgsql;

-- #3 Процедура для триггера

CREATE FUNCTION calculate\_duration ()

RETURNS trigger AS $$

DECLARE album\_duration INTERVAL;

BEGIN

    SELECT SUM(duration) FROM song

Окончание Листинга В.3

    WHERE album\_id = NEW.album\_id

    GROUP BY album\_id

    INTO album\_duration;

    UPDATE album SET duration = album\_duration WHERE id = NEW.album\_id;

    RAISE NOTICE 'Триггер отработал';

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- #3.1 Сам триггер

CREATE TRIGGER tr\_calc\_dur

AFTER INSERT ON song FOR EACH ROW

EXECUTE FUNCTION calculate\_duration();

-- #4 Процедура для триггера BEFORE

CREATE FUNCTION check\_album\_length ()

RETURNS trigger AS $$

DECLARE album\_song\_count INTEGER;

BEGIN

    SELECT COUNT(\*) FROM song

    WHERE album\_id = NEW.album\_id

    GROUP BY album\_id

    INTO album\_song\_count;

    IF album\_song\_count >= 20 THEN

        RAISE EXCEPTION 'Превышен лимит на количество песен в альбоме';

    ELSE

        RAISE NOTICE 'Запись добавлена, количество песен в альбоме: %', album\_song\_count + 1;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER tr\_check\_album\_length

BEFORE INSERT ON song FOR EACH ROW

EXECUTE PROCEDURE check\_album\_length();

### Приложение Г

Скрипты к четвёртой практической работе

Листинг Г.1 – Скрипты для оконных функций

-- Агрегирующие функции

SELECT

    a.name,

    s.name,

    COUNT(\*) OVER (PARTITION BY a.id) AS song\_count,

    SUM(s.duration) OVER (PARTITION BY a.id) AS album\_duration,

    AVG(s.duration) OVER (PARTITION BY a.id) AS avg\_song\_duration,

    MIN(s.duration) OVER (PARTITION BY a.id) AS min\_song\_duration,

    MAX(s.duration) OVER (PARTITION BY a.id) AS max\_song\_duration

FROM song AS s

JOIN album AS a ON a.id = s.album\_id;

-- Ранжирующие функции

SELECT

    a.name,

    s.name,

    ROW\_NUMBER() OVER (PARTITION BY a.id ORDER BY s.id) AS song\_number

FROM song AS s

JOIN album AS a ON a.id = s.album\_id;

WITH song\_performer\_info

AS

(SELECT CASE WHEN (m.name, g.name) IS NULL THEN NULL

             ELSE concat(m.name, g.name) END AS performer\_name,

        COUNT(\*) AS song\_count

FROM song AS s

JOIN song\_performer AS sp ON sp.song\_id = s.id

JOIN performer AS p ON p.id = sp.performer\_id

LEFT OUTER JOIN musician AS m ON m.id = p.musician\_id

LEFT OUTER JOIN music\_group AS g ON g.id = p.group\_id

GROUP BY performer\_name)

SELECT

    performer\_name,

    song\_count,

    RANK() OVER (ORDER BY song\_count DESC) AS song\_cnt\_rank,

    DENSE\_RANK() OVER (ORDER BY song\_count DESC) AS song\_cnt\_dense\_rank

FROM song\_performer\_info;

SELECT

    a.name,

    s.name,

    NTILE(2) OVER (PARTITION BY a.id ORDER BY s.id) AS vinyl\_side

FROM song AS s

JOIN album AS a ON a.id = s.album\_id;

-- Функции смещения

SELECT

    a.name AS album\_name,

    s.name AS song\_name,

    LAG(s.name) OVER (PARTITION BY a.name ORDER BY s.id) AS previous\_song,

    LEAD(s.name) OVER (PARTITION BY a.name ORDER BY s.id) AS next\_song

FROM song AS s

JOIN album AS a ON a.id = s.album\_id;

SELECT DISTINCT ON (a.name)

    a.name AS album\_name,

Окончание Листинга Г.1

    FIRST\_VALUE(s.name) OVER (PARTITION BY a.name ORDER BY s.id) AS first\_song,

    LAST\_VALUE(s.name) OVER value\_window AS last\_song,

    NTH\_VALUE(s.name, 2) OVER value\_window AS second\_song

FROM song AS s

JOIN album AS a ON a.id = s.album\_id

WINDOW value\_window AS (PARTITION BY a.name ORDER BY s.id

                         ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING);