Санкт-Петербургский Государственный Университет

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

ДАНИЛОВА МАРИНА ЮРЬЕВНА

КУРСОВАЯ РАБОТА

МЕРТВЫЕ МУЗЫКАНТЫ

(DEAD MUSICIANS)

Направление 010302

Прикладная математика, фундаментальная информатика и программирование

Преподаватель Филиппов Р. О.

Санкт-Петербург

2017

Содержание

Ссылка на базу данных «Dead musicians» на сервисе GitHub:
https://github.com/danilova-inspiration/Database-dead_musicians

Глава 1 Схема

Здесь описывается структура базы данных «Dead musicians»

Сущность Musicians

- Id
- Name surname
- Birthday
- Music_education
- Id_country
- Data_death
- Cause_death

Сущность Music groups

- Id
- Genre
- Name_group
- Data_begin
- Data_end
- Country_id

Сущность Music awards

- id
- name
- begin
- country_m

Countries

- Id
- Country

Genres

- Id
- genre

Musical instruments

- Id
- instrument

Musicians musical instruments relation

- Id
- Id_instrument
- Id_musician

Musicians music groups relation

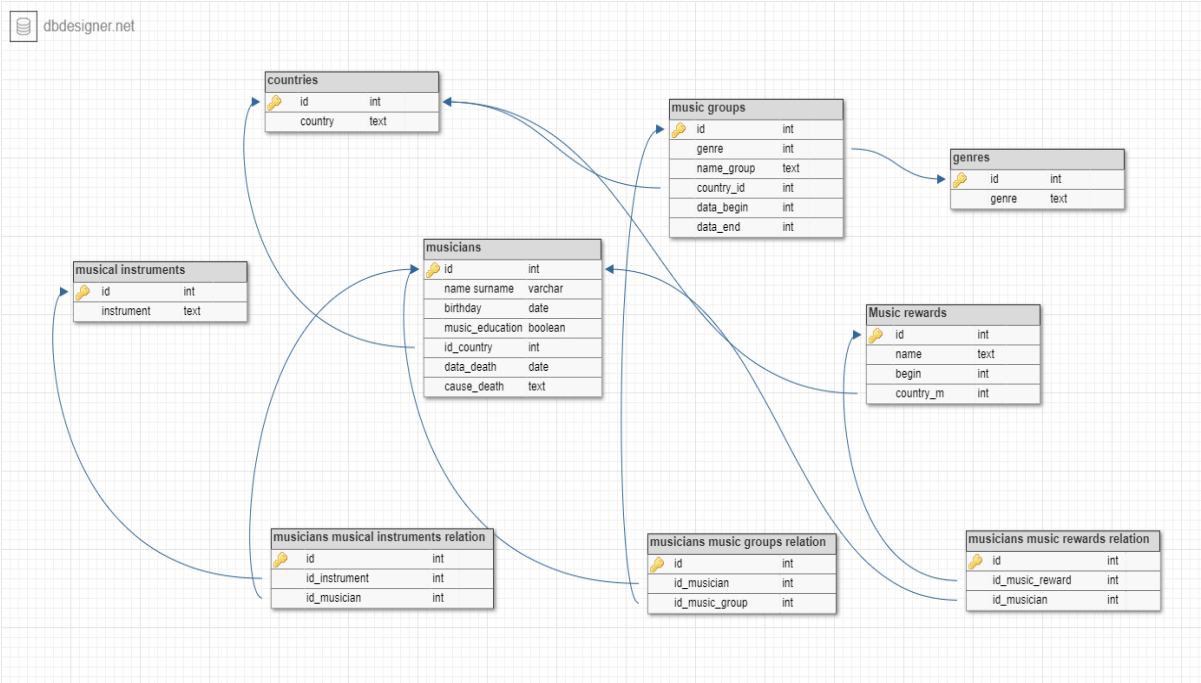
- Id
- Id_music_group
- Id_musician

Musicians music rewards relation

- Id

- Id_music_reward
- Id_musician

Схема базы данных «Dead musicians» смоделированная при помощи ресурса dbdesigner.net



Глава 2 Описание базы данных

База данных «Dead musicians» описывает основную информацию о карьере и жизни музыкантов. Помимо этого, в ней содержится информация о самых известных музыкальных наградах мира.

База данных предназначена для людей, которые хотят узнать главные факты из жизни музыканта и не желают сильно углубляться в подробности.

Описание взаимоотношений объектов

Musicians

В этой таблице описывается основные факты из биографии музыкантов

- Id – primary key данной таблицы (тип данных: serial)
- Name surname – фамилия и имя музыканта (тип данных: varchar)
- Birthday – день рождения (тип данных: date)
- Music_education – информация о наличии музыкального образования (да/нет) (тип данных: boolean)
- Id_country – страна жительства (тип данных: int)
- Data_death – дата смерти (тип данных: date)
- Cause_death – причина смерти (тип данных: text)

Music groups

В таблице находится основная информация о музыкальных группах

- Id - primary key данной таблицы (тип данных: serial)
- Genre – музыкальный жанр группы (тип данных: int)
- Name_group – название музыкальной группы (тип данных: text)
- Data_begin – дата основания группы (тип данных: int)
- Data_end – дата окончания существования группы (тип данных: int)

- Country_id – страна, в которой дислоцировалась группа (тип данных: int)

Music awards

В этой таблице представлена информация о музыкальных наградах

- id – primary key данной таблицы (тип данных: serial)
- name – название музыкальной награды (тип данных: text)
- begin – в каком году впервые была присуждена (тип данных: int)
- country_m – страна, которая учреждает награду (тип данных: int)

Countries

В таблице перечислены названия стран

- Id – primary key данной таблицы (тип данных: serial)
- Country – название страны (тип данных: text)

Genres

В таблице перечислены названия музыкальных жанров

- Id – primary key данной таблицы (тип данных: serial)
- genre– название жанра (тип данных: text)

Musical instruments

Здесь содержится список музыкальных инструментов

- Id - primary key данной таблицы (тип данных: serial)

- instrument – название инструмента (вокал, барабаны, гитара и т.п.) (тип данных: text)

Musicians musical instruments relation

В данной таблице реализуется связь вида m:m между таблицами «musical instruments» и «musicians»

- Id - primary key данной таблицы (тип данных: serial)
- Id_instrument - ссылка на id инструмента из таблицы «musical instruments» (тип данных: int)
- Id_musician - ссылка на id музыканта из таблицы «musicians» (тип данных: int)

Musicians music groups relation

В данной таблице реализуется связь вида m:m между таблицами «music groups» и «musicians»

- Id - primary key данной таблицы (тип данных: serial)
- Id_music_group - ссылка на id инструмента из таблицы «music groups» (тип данных: int)
- Id_musician - ссылка на id музыканта из таблицы «musicians» (тип данных: int)

Musicians music rewards relation

В данной таблице реализуется связь вида m:m между таблицами «music rewards» и «musicians»

- Id - primary key данной таблицы (тип данных: serial)
- Id_music_reward - ссылка на id инструмента из таблицы «music rewards» (тип данных: int)
- Id_musician - ссылка на id музыканта из таблицы «musicians» (тип данных: int)

Глава 3 Легкие запросы и их оптимизация

1. Выбрать всех музыкантов, не имеющих музыкальное образование и скончавшихся в 21 веке

SELECT name_surname, cause_death FROM musicians WHERE (data_death > '2000-01-01'AND musicians.music_education='No');

До оптимизации:

QUERY PLAN

Seq Scan on musicians (cost=0.00..1.20 rows=3 width=512) (actual time=0.056..0.067 rows=8 loops=1)

Filter: ((NOT music_education) AND (data_death > '2000-01-01'::date))

Rows Removed by Filter: 8

Planning time: 0.874 ms

Execution time: 0.141 ms

(5 строк)

Оптимизация: добавлен индекс по дате смерти музыканта

CREATE INDEX ON musicians(cause_death);

EXPLAIN(ANALYZE) SELECT name_surname, cause_death FROM musicians WHERE (data_death > '2000-01-01'AND musicians.music_education='No');

После оптимизации:

QUERY PLAN

```
 Seq Scan on musicians  (cost=0.00..1.20 rows=3 width=512) (actual
time=0.031..0.039 rows=8 loops=1)
   Filter: ((NOT music_education) AND (data_death > '2000-01-01'::date))
   Rows Removed by Filter: 8
 Planning time: 1.136 ms
 Execution time: 0.071 ms
(5 строк)
```

2. Выбрать все награды в названии которых присутствует слово "Awards" и которые были учреждены после 1962 года

```
SELECT name,begin FROM Music_rewards WHERE name LIKE '%Awards%'
AND Music_rewards.begin>1962;
```

До оптимизации:

```
                                QUERY PLAN
-----------------------------------------------------------------------------------------------
 Seq Scan on music_rewards  (cost=0.00..1.23 rows=1 width=160) (actual
time=0.563..0.575 rows=8 loops=1)
   Filter: (((name)::text ~~ '%Awards%'::text) AND (begin > 1962))
   Rows Removed by Filter: 7
 Planning time: 232.351 ms
 Execution time: 0.629 ms
(5 строк)
```

Оптимизация: добавлен индекс по дате учреждения награды

```
CREATE INDEX ON Music_rewards(begin);
```

EXPLAIN(ANALYZE)SELECT name,begin FROM Music_rewards WHERE name LIKE '%Awards%' AND Music_rewards.begin>1962;

После оптимизации:

QUERY PLAN

Seq Scan on music_rewards (cost=0.00..1.23 rows=1 width=160) (actual time=0.038..0.048 rows=8 loops=1)

Filter: (((name)::text ~~ '%Awards%'::text) AND (begin > 1962))

Rows Removed by Filter: 7

Planning time: 1.291 ms

Execution time: 0.083 ms

(5 строк)

3. Вывести 3 самых старых музыкантов, у которых в причине смерти содержится слово ‘cancer’ и которые родились после 1900 года.

SELECT name_surname, cause_death, age(data_death,birthday) FROM musicians WHERE (cause_death LIKE '%cancer%' AND musicians.birthday > '1900-01-01') ORDER BY age DESC LIMIT 3;

До оптимизации:

QUERY PLAN

Limit (cost=1.26..1.26 rows=1 width=528) (actual time=0.132..0.133 rows=3 loops=1)

-> Sort (cost=1.26..1.26 rows=1 width=528) (actual time=0.130..0.130 rows=3 loops=1)

Sort Key: (age((data_death)::timestamp with time zone, (birthday)::timestamp with time zone)) DESC

Sort Method: quicksort Memory: 25kB

-> Seq Scan on musicians (cost=0.00..1.25 rows=1 width=528) (actual time=0.076..0.096 rows=4 loops=1)

Filter: (((cause_death)::text ~~ '%cancer%'::text) AND (birthday > '1900-01-01'::date))

Rows Removed by Filter: 12

Planning time: 0.358 ms

Execution time: 0.205 ms

(9 строк)

Оптимизация: добавлен индекс по дате рождения музыканта

```
CREATE INDEX ON musicians(birthday);

EXPLAIN(ANALYZE)SELECT name_surname, cause_death, age(data_death,birthday) FROM musicians WHERE (cause_death LIKE '%cancer%' AND musicians.birthday > '1900-01-01' ) ORDER BY age DESC LIMIT 3;
```

После оптимизации:

QUERY PLAN

Limit (cost=1.26..1.26 rows=1 width=528) (actual time=0.095..0.096 rows=3 loops=1)

-> Sort (cost=1.26..1.26 rows=1 width=528) (actual time=0.093..0.094 rows=3 loops=1)

Sort Key: (age((data_death)::timestamp with time zone, (birthday)::timestamp with time zone)) DESC

Sort Method: quicksort Memory: 25kB

-> Seq Scan on musicians (cost=0.00..1.25 rows=1 width=528) (actual time=0.053..0.071 rows=4 loops=1)

Filter: (((cause_death)::text ~~ '%cancer%'::text) AND (birthday > '1900-01-01'::date))

Rows Removed by Filter: 12

Planning time: 1.582 ms

Execution time: 0.146 ms

(9 строк)

4. Вывести значение среднего возраста музыкантов, которые скончались от сердечной недостаточности

SELECT AVG(age(data_death,birthday)) FROM musicians WHERE (musicians.cause_death LIKE '%Heart failure%');

До оптимизации:

QUERY PLAN

Aggregate (cost=1.21..1.22 rows=1 width=16) (actual time=0.094..0.094 rows=1 loops=1)

-> Seq Scan on musicians (cost=0.00..1.20 rows=1 width=8) (actual time=0.034..0.044 rows=3 loops=1)

Filter: ((cause_death)::text ~~ '%Heart failure%'::text)

Rows Removed by Filter: 13

Planning time: 0.309 ms

Execution time: 0.209 ms

(6 строк)

Оптимизация: добавлен индекс по причине смерти

```
CREATE INDEX ON musicians(cause_death);

EXPLAIN(ANALYZE)SELECT AVG(age(data_death,birthday)) FROM
musicians WHERE (musicians.cause_death LIKE '%Heart failure%');
```

После оптимизации:

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=1.21..1.22 rows=1 width=16) (actual time=0.089..0.089 rows=1 loops=1)
   ->  Seq Scan on musicians  (cost=0.00..1.20 rows=1 width=8) (actual time=0.038..0.047 rows=3 loops=1)
         Filter: ((cause_death)::text ~~ '%Heart failure%'::text)
         Rows Removed by Filter: 13
 Planning time: 1.300 ms
 Execution time: 0.185 ms
(6 строк)
```

Глава 4 Средние запросы и их оптимизация

1. Выбрать всех музыкантов из Великобритании и сортировать их от страшего к младшему.

```
SELECT name_surname, country, age(data_death,birthday) FROM (musicians AS m INNER JOIN countries AS c ON (m.id_country = c.id)) WHERE (c.country = 'Great Britain') ORDER BY age DESC;
```

До оптимизации:

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=2.39..2.40 rows=2 width=328) (actual time=0.260..0.262 rows=6 loops=1)
   Sort Key: (age((musicians.data_death)::timestamp with time zone, (musicians.birthday)::timestamp with time zone)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  Hash Join  (cost=1.14..2.38 rows=2 width=328) (actual time=0.098..0.135 rows=6 loops=1)
         Hash Cond: (musicians.id_country = countries.id)
         ->  Seq Scan on musicians  (cost=0.00..1.16 rows=16 width=168) (actual time=0.027..0.031 rows=16 loops=1)
         ->  Hash  (cost=1.13..1.13 rows=1 width=160) (actual time=0.028..0.028 rows=1 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on countries  (cost=0.00..1.13 rows=1 width=160) (actual time=0.019..0.022 rows=1 loops=1)
                     Filter: ((country)::text = 'Great Britain'::text)
                     Rows Removed by Filter: 9
 Planning time: 0.492 ms
 Execution time: 0.359 ms
```

(13 строк)

Оптимизация: добавлен индекс по названию страны

```
CREATE INDEX ON countries(country);

EXPLAIN (ANALYZE) SELECT name_surname, country,
age(data_death,birthday) FROM (musicians CROSS JOIN countries) WHERE
(musicians.id_country = countries.id AND countries.country = 'Great Britain')
ORDER BY age DESC;
```

После оптимизации:

```
                                                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=2.39..2.40 rows=2 width=328) (actual time=0.261..0.263 rows=6 loops=1)
   Sort Key: (age((musicians.data_death)::timestamp with time zone, (musicians.birthday)::timestamp with time zone)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  Hash Join  (cost=1.14..2.38 rows=2 width=328) (actual time=0.168..0.201 rows=6 loops=1)
         Hash Cond: (musicians.id_country = countries.id)
         ->  Seq Scan on musicians  (cost=0.00..1.16 rows=16 width=168) (actual time=0.039..0.045 rows=16 loops=1)
         ->  Hash  (cost=1.13..1.13 rows=1 width=160) (actual time=0.071..0.071 rows=1 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on countries  (cost=0.00..1.13 rows=1 width=160) (actual time=0.033..0.038 rows=1 loops=1)
                     Filter: ((country)::text = 'Great Britain'::text)
                     Rows Removed by Filter: 9
 Planning time: 1.673 ms
```

Execution time: 0.346 ms

(13 строк)

2. Выбрать музыкантов, которые получили британскую премию, учрежденную после 1955 года

SELECT name_surname, name, begin FROM (countries AS c INNER JOIN (music_rewards AS mr INNER JOIN (musicians_music_rewards_relation AS mmrr INNER JOIN musicians AS m ON (mmrr.id_musician = m.id)) ON (mmrr. id_music_reward = mr.id)) ON (mr.country_m=c.id)) WHERE (c.country = 'Great Britain' AND mr.begin>'1955');

До оптимизации:

QUERY PLAN

Hash Join (cost=3.75..42.52 rows=136 width=316) (actual time=0.301..0.320 rows=6 loops=1)

Hash Cond: (mmrr.id_musician = m.id)

-> Hash Join (cost=2.39..40.54 rows=136 width=164) (actual time=0.224..0.239 rows=6 loops=1)

Hash Cond: (mmrr.id_music_reward = mr.id)

-> Seq Scan on musicians_music_rewards_relation mmrr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.099..0.102 rows=24 loops=1)

-> Hash (cost=2.38..2.38 rows=1 width=164) (actual time=0.083..0.083 rows=5 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=0.00..2.38 rows=1 width=164) (actual time=0.055..0.065 rows=5 loops=1)

Join Filter: (c.id = mr.country_m)

Rows Removed by Join Filter: 9

-> Seq Scan on countries c (cost=0.00..1.13 rows=1 width=4) (actual time=0.031..0.033 rows=1 loops=1)

Filter: ((country)::text = 'Great Britain'::text)

Rows Removed by Filter: 9

-> Seq Scan on music_rewards mr (cost=0.00..1.19 rows=5 width=168) (actual time=0.017..0.024 rows=14 loops=1)

Filter: (begin > 1955)

Rows Removed by Filter: 1

-> Hash (cost=1.16..1.16 rows=16 width=160) (actual time=0.058..0.058 rows=16 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on musicians m (cost=0.00..1.16 rows=16 width=160) (actual time=0.036..0.043 rows=16 loops=1)

Planning time: 40.422 ms

Execution time: 0.438 ms

(21 строка)

Оптимизация: добавлен индекс по названию страны и году учреждения музыкальной награды

CREATE INDEX ON countries(country);

CREATE INDEX ON Music_rewards(begin);

EXPLAIN (ANALYZE) SELECT name_surname, name, begin FROM (countries AS c INNER JOIN (music_rewards AS mr INNER JOIN (musicians_music_rewards_relation AS mmrr INNER JOIN musicians AS m ON (mmrr.id_musician = m.id)) ON (mmrr. id_music_reward = mr.id)) ON (mr.country_m=c.id)) WHERE (c.country = 'Great Britain' AND mr.begin>'1955');

После оптимизации:

QUERY PLAN

```
---------------------------------------------------------------------------------------------

 Hash Join  (cost=3.75..42.52 rows=136 width=316) (actual time=0.168..0.186
rows=6 loops=1)

   Hash Cond: (mmrr.id_musician = m.id)

   ->  Hash Join  (cost=2.39..40.54 rows=136 width=164) (actual time=0.094..0.107
rows=6 loops=1)

         Hash Cond: (mmrr.id_music_reward = mr.id)

         ->  Seq Scan on musicians_music_rewards_relation mmrr  (cost=0.00..30.40
rows=2040 width=8) (actual time=0.012..0.015 rows=24 loops=1)

         ->  Hash  (cost=2.38..2.38 rows=1 width=164) (actual time=0.061..0.061
rows=5 loops=1)

               Buckets: 1024  Batches: 1  Memory Usage: 9kB

               ->  Nested Loop  (cost=0.00..2.38 rows=1 width=164) (actual
time=0.041..0.053 rows=5 loops=1)

                     Join Filter: (c.id = mr.country_m)

                     Rows Removed by Join Filter: 9

                     ->  Seq Scan on countries c  (cost=0.00..1.13 rows=1 width=4) (actual
time=0.021..0.023 rows=1 loops=1)

                           Filter: ((country)::text = 'Great Britain'::text)

                           Rows Removed by Filter: 9

                     ->  Seq Scan on music_rewards mr  (cost=0.00..1.19 rows=5
width=168) (actual time=0.014..0.019 rows=14 loops=1)

                           Filter: (begin > 1955)

                           Rows Removed by Filter: 1

   ->  Hash  (cost=1.16..1.16 rows=16 width=160) (actual time=0.052..0.052
rows=16 loops=1)

         Buckets: 1024  Batches: 1  Memory Usage: 9kB

         ->  Seq Scan on musicians m  (cost=0.00..1.16 rows=16 width=160) (actual
time=0.027..0.035 rows=16 loops=1)

 Planning time: 2.043 ms

 Execution time: 0.293 ms
```

(21 строка)

3. Выбрать вокалистов из Америки, которые скончались после 1990 года, вывести причину и дату их смерти

SELECT name_surname, cause_death, data_death FROM (countries AS c INNER JOIN (musical_instruments AS mi INNER JOIN (musicians AS m INNER JOIN musicians_musical_instruments_relation AS mmir ON (mmir.id_musician=m.id)) ON (mmir. id_instrument = mi.id)) ON (m.id_country = c.id)) WHERE (c.country = 'USA' AND mi.instrument = 'vocal' AND m.data_death>='1990-01-01');

До оптимизации:

QUERY PLAN

Nested Loop (cost=2.55..7.41 rows=1 width=516) (actual time=0.347..0.408 rows=2 loops=1)

-> Hash Join (cost=2.40..4.16 rows=3 width=520) (actual time=0.233..0.263 rows=6 loops=1)

Hash Cond: (mmir.id_musician = m.id)

-> Seq Scan on musicians_musical_instruments_relation mmir (cost=0.00..1.53 rows=53 width=8) (actual time=0.121..0.129 rows=53 loops=1)

-> Hash (cost=2.39..2.39 rows=1 width=520) (actual time=0.084..0.084 rows=3 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=0.00..2.39 rows=1 width=520) (actual time=0.059..0.073 rows=3 loops=1)

Join Filter: (c.id = m.id_country)

Rows Removed by Join Filter: 8

-> Seq Scan on countries c (cost=0.00..1.13 rows=1 width=4) (actual time=0.031..0.034 rows=1 loops=1)

Filter: ((country)::text = 'USA'::text)

Rows Removed by Filter: 9

-> Seq Scan on musicians m (cost=0.00..1.20 rows=5 width=524) (actual time=0.023..0.029 rows=11 loops=1)

Filter: (data_death >= '1990-01-01'::date)

Rows Removed by Filter: 5

-> Index Scan using musical_instruments_pk on musical_instruments mi (cost=0.15..1.07 rows=1 width=4) (actual time=0.021..0.022 rows=0 loops=6)

Index Cond: (id = mmir.id_instrument)

Filter: ((instrument)::text = 'vocal'::text)

Rows Removed by Filter: 1

Planning time: 85.019 ms

Execution time: 0.550 ms

(21 строка)

Оптимизация: добавлен индекс по дате смерти музыканта

```
CREATE INDEX ON musicians(data_death);

EXPLAIN (ANALYZE) SELECT name_surname, cause_death, data_death
FROM (countries AS c INNER JOIN (musical_instruments AS mi INNER JOIN
(musicians AS m INNER JOIN musicians_musical_instruments_relation AS mmir
ON (mmir.id_musician=m.id)) ON (mmir. id_instrument = mi.id)) ON
(m.id_country = c.id)) WHERE (c.country = 'USA' AND mi.instrument = 'vocal'
AND m.data_death>='1990-01-01');
```

После оптимизации:

QUERY PLAN

Nested Loop (cost=2.55..7.41 rows=1 width=516) (actual time=0.136..0.185 rows=2 loops=1)

-> Hash Join (cost=2.40..4.16 rows=3 width=520) (actual time=0.122..0.151 rows=6 loops=1)

Hash Cond: (mmir.id_musician = m.id)

-> Seq Scan on musicians_musical_instruments_relation mmir (cost=0.00..1.53 rows=53 width=8) (actual time=0.028..0.033 rows=53 loops=1)

-> Hash (cost=2.39..2.39 rows=1 width=520) (actual time=0.070..0.070 rows=3 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=0.00..2.39 rows=1 width=520) (actual time=0.045..0.059 rows=3 loops=1)

Join Filter: (c.id = m.id_country)

Rows Removed by Join Filter: 8

-> Seq Scan on countries c (cost=0.00..1.13 rows=1 width=4) (actual time=0.024..0.027 rows=1 loops=1)

Filter: ((country)::text = 'USA'::text)

Rows Removed by Filter: 9

-> Seq Scan on musicians m (cost=0.00..1.20 rows=5 width=524) (actual time=0.017..0.024 rows=11 loops=1)

Filter: (data_death >= '1990-01-01'::date)

Rows Removed by Filter: 5

-> Index Scan using musical_instruments_pk on musical_instruments mi (cost=0.15..1.07 rows=1 width=4) (actual time=0.004..0.004 rows=0 loops=6)

Index Cond: (id = mmir.id_instrument)

Filter: ((instrument)::text = 'vocal'::text)

Rows Removed by Filter: 1

Planning time: 2.099 ms

Execution time: 0.363 ms

(21 строка)

Глава 3 Сложные запросы

1. Вывести всех участников музыкальной группы самого многочисленного жанра, которая была основана раньше других групп из этого жанра

```
SELECT name_surname, name_group FROM (musicians INNER JOIN
(music_groups INNER JOIN musicians_music_groups_relation ON
(musicians_music_groups_relation.id_music_group=music_groups.id)) ON
(musicians_music_groups_relation.id_musician=musicians.id)) WHERE
name_group=( SELECT N.name_group FROM (SELECT genres.genre,
name_group, data_begin FROM (music_groups INNER JOIN genres ON
(music_groups.genre=genres.id))) AS N WHERE genre=(SELECT g.genre FROM
genres AS g LEFT JOIN music_groups AS m ON (g.id=m.genre) GROUP BY g.id
ORDER BY count(g.id) DESC LIMIT 1) ORDER BY data_begin LIMIT 1);
```

2. Посчитать количество музыкантов, которые умели играть на гитаре и скончались после 1990 года

```
WITH t1 AS (SELECT instrument, name_surname FROM (musical_instruments
AS mi INNER JOIN(musicians_musical_instruments_relation AS mmir INNER
JOIN musicians AS m ON (mmir.id_musician=m.id)) ON
(mmir.id_instrument=mi.id AND mi.instrument=’guitar’))), t2 AS (SELECT
name_surname FROM (countries INNER JOIN (musical_instruments INNER
JOIN (musicians INNER JOIN musicians_musical_instruments_relation ON
(musicians_musical_instruments_relation.id_musician=musicians.id)) ON
(musicians_musical_instruments_relation. id_instrument =
musical_instruments.id)) ON (musicians.id_country = countries.id)) WHERE
(countries.country = 'Great Britain' AND musical_instruments.instrument =
'guitar') GROUP BY (name_surname, data_death) HAVING
musicians.data_death>='1990-01-01') SELECT count(instrument) FROM t1
WHERE name_surname in (SELECT name_surname FROM t2);
```

3. Вывести музыкантов, получивших американские награды, и игравших в жанрах, в которых присутствует слово «Rock», вывести полные названия этих жанров.

```
WITH r1 AS(SELECT name_surname, g.genre FROM(musicians AS m INNER
JOIN(musicians_music_groups_relation AS mmgr INNER JOIN(music_groups
AS mg INNER JOIN genres AS g ON (mg.genre=g.id AND g.genre LIKE
'%Rock%' )) ON (mmgr.id_music_group=mg.id)) ON (mmgr.id_musician=m.id))
ORDER BY name_surname), r2 AS(SELECT name_surname, name, begin FROM
(musicians_music_rewards_relation CROSS JOIN musicians CROSS JOIN
music_rewards CROSS JOIN countries) WHERE
(musicians_music_rewards_relation. id_music_reward = music_rewards.id AND
musicians_music_rewards_relation.id_musician = musicians.id AND
music_rewards.country_m=countries.id AND countries.country = 'USA'))
SELECT name_surname,genre FROM r1 WHERE name_surname in (SELECT
name_surname FROM r2);
```