Санкт-Петербургский государственный университет
Факультет прикладной математики-процессов управления

Статьева Юлия Витальевна

Курсовая работа

УЧЁНЫЕ
(SCIENTISTS)

Направление 010302

Прикладная математика, фундаментальная информатика и программирование

Преподаватель: Филиппов Р.О.

Санкт-Петербург

2017

Содержание

Глава 1

Схема базы данных

Здесь представлена структура базы данных «Учёные».

Таблица «Scientists»

- id
- name_and_surname
- date_of_birth
- date_of_death
- town_of_birth_id
- alma_mater

Таблица «Towns»

- id
- town
- id_country

Таблица «Countries»

- id
- country

Таблица «Scientific_awards»

- id
- the_name_of_the_award
- year
- country_id

Таблица «Scientific_spheres»

- id
- scientific_sphere

Таблица «Scientists_spheres_relation»

- id_scientists
- id_sphere_of_activity

Таблица «Discoveries»

- id
- discovery
- opening_area

- place

Таблица «Scientists_Scientific_awards_relation»

- id_scientist
- id_scientific_award

Таблица «Scientists_Discoveries_relation»

- id_scientist
- id_scientist_discovery

Глава 2

Описание схемы базы данных

База данных «Scientists» содержит информацию об ученых: дата рождения и смерти, город и страну рождения, их альма-матер. Также содержится информация об открытиях и наградах.

«Scientists» предназначена для людей, которые хотят получше узнать об учёных, о различных открытиях, наградах.

Первая «сущность»: Таблица ***«Scientists»***

В первой «главной» таблице находится информация об учёных (имя и фамилия учёного, его дата рождения и дата смерти, город рождения учёного и его альма-матер).

Столбцы имеют следующие названия:

1 столбец – id- нумерация строк (тип данных int);

2 столбец – name_and _surname- имя и фамилия учёного (тип данных character varying(80));

3 столбец – date_of_birth - дата рождения учёного (тип данных date);

4 столбец – date_of_death – дата смерти учёного (тип данных date);

5 столбец – town_of_birth – id города, в котором родился учёный (id берется из таблицы «Towns») (тип данных integer);

6 столбец – alma_mater- альма-матер учёного (тип данных character varying(80)).

В первой строке записываются названия каждого столбца в порядке, указанном выше. Остальные строки заполняются в соответствии с названиями столбцов.

Таблица «Towns»

Названия столбцов:

1 столбец -id – присваивание каждому городу своего порядкового номера (тип данных int);

2 столбец - town- название города (тип данных character varying(80));

3 столбец – id_country- записывается id нужной страны (id берется из таблицы «Countries») (тип данных integer).

В первой строке указываются названия столбцов (id, town, id_country). В остальных строках, согласно названиям столбцов, записываем порядковый номер, название города и id нужной страны.

Данная таблица необходима для того, чтобы в таблице «Scientists» в столбце town_of birth_id указать id того города, в котором родился учёный.

Таблица «Countries»

Названия столбцов:

1 столбец -id – присваивание каждой стране своего порядкового номера (тип данных int);

2 столбец - country- название страны (тип данных character varying(80)).

В первой строчке указываются названия столбцов (id, country). В остальных строках, согласно названиям столбцов, записываем порядковый номер и название страны.

Данная таблица необходима для того, чтобы в таблице «Towns» в столбце id_country указать id той страны, в которой находится город рождения учёного, и чтобы в таблице «Scientific_awards» указать id той страны, в которой была учреждена награда.

С помощью таблиц «Towns» и «Countries» реализуется отношение m:1.

Вторая «сущность»: ***Таблица «Scientific_awards»***

Во второй «главной» таблице находится информация о научных наградах (название научной награды, год, в который она была впервые присуждена, страна, в которой была учреждена награда).

Столбцы имеют следующие названия:

1 столбец – id- нумерация строк (тип данных int);

2 столбец – the_name_of_the_award- название награды (тип данных character varying(80));

3 столбец – year – год, в который награда была впервые присуждена (тип данных integer);

4 столбец - country_id – id страны, в которой была учреждена награда(id страны берем из таблицы «Countries») (тип данных integer).

В первой строке записываются названия каждого столбца в порядке, указанном выше. Остальные строки заполняются в соответствии с названиями столбцов.

Таблица «Scientific_spheres»

Названия столбцов:

1 столбец -id – присваивание каждой научной сфере своего порядковый номер (тип данных int);

2 столбец – scientific_sphere - название научной сферы (тип данных character varying(80)).

В первой строчке указываются названия столбцов (id, scientific_sphere). В остальных строках, согласно названиям столбцов, записываем порядковый номер и название научной сферы.

Данная таблица необходима для того, чтобы в таблице «Scientists_spheres_relation» в столбце id_sphere_of_activity указать id нужной научной сферы.

Таблица «Scientists_spheres_relation»

В этой таблице содержится информация о том, какая область деятельности у учёного.

Названия столбцов:

1 столбец -id – нумерация строк (тип данных int);

2 столбец – id_scientist – id учёного (нужный номер берется из таблицы «Scientists») (тип данных integer);

3 столбец - id_sphere_of_activity - id научной сферы (нужный номер берется из таблицы «Scientific_spheres») (тип данных integer).

В первой строке указываются названия столбцов (id, id_scientist, id_sphere_of_activity). В остальных строках, согласно названиям столбцов, записываем порядковый номер, id учёного, id научной сферы.

Здесь реализуется отношение вида m:1 между таблицами «Scientists» и «Scientific_spheres».

Третья «сущность»: ***Таблица «Discoveries»***

В третьей «главной» таблице находится информация о достижениях учёного в науке (то есть указывается название достижений, год публикации или презентации достижения/открытия, область открытия, место публикации или презентации достижения/открытия).

Столбцы имеют следующие названия:

1 столбец – id- нумерация строк (тип данных int);

2 столбец - discovery – название достижения или открытия учёного (тип данных character varying(80));

3 столбец – year– год упоминания или публикации, или презентации достижения или открытия (тип данных integer);

4 столбец – opening_area – область открытия (тип данных character varying(80));

5 столбец - place- место публикации или презентации достижения/открытия (информационные издательства, конференции и т.п.) (тип данных character varying(180)).

Если нет точной информации о месте публикации, то в ячейку записываем No.

В первой строке записываются названия каждого столбца в порядке, указанном выше. Остальные строки заполняются в соответствии с названиями столбцов.

Таблица «Scientists_scientific_awards_relation»

В этой таблице содержится информация о том, какие награды имеются у учёного.

Названия столбцов:

1 столбец -id – нумерация строк (тип данных int);

2 столбец – id_scientist – id учёного (нужный номер берется из таблицы «Scientists») (тип данных integer);

3 столбец - id_scientific_award - id награды (нужный номер берется из таблицы «Scientific_awards») (тип данных integer).

В первой строке указываются названия столбцов (id, id_scientist, id_scientific_award). В остальных строках, согласно названиям столбцов, записываем порядковый номер), id учёного, id награды.

Таблица «Scientists_discoveries_relation»

В этой таблице содержится информация о том, какое открытие/достижение у ученого.

Названия столбцов:

1 столбец -id – нумерация строк (тип данных int);

2 столбец – id_scientist – id учёного (номер берется из таблицы «Scientists») (тип данных integer);

3 столбец - id_scientist_discovery - id открытия/ достижения (нужный номер берется из таблицы «Discoveries») (тип данных integer).

В первой строчке указываются названия столбцов (id, id_scientist, id_scientist_discovery). В остальных строках, согласно названиям столбцов, записываем порядковый номер, id учёного, id открытия/ достижения.

Глава 3

Легкие запросы и их оптимизация

1) Вывести средний возраст ученых, которые родились в период с 1902 по 1956

SELECT AVG(age(date_of_death,date_of_birth)) FROM Scientists WHERE (Scientists.date_of_birth > '1902-01-01' AND Scientists.date_of_birth < '1956-01-01');

До оптимизации:

QUERY PLAN

Aggregate (cost=1.24..1.25 rows=1 width=16) (actual time=5.994..5.994 rows=1 loops=1)

-> Seq Scan on scientists (cost=0.00..1.23 rows=1 width=8) (actual time=0.206..0.229 rows=4 loops=1)

Filter: ((date_of_birth > '1902-01-01'::date) AND (date_of_birth < '1956-01-01'::date))

Rows Removed by Filter: 11

Planning time: 87.315 ms

Execution time: 10.514 ms

(6 строк)

Создание индекса: Создаем индекс по дате рождения учёного

CREATE INDEX ON Scientists (date_of_birth);

EXPLAIN (ANALYZE) SELECT AVG(age(date_of_death,date_of_birth)) FROM Scientists WHERE (Scientists.date_of_birth > '1902-01-01' AND Scientists.date_of_birth < '1956-01-01');

После оптимизации:

QUERY PLAN

Aggregate (cost=1.24..1.25 rows=1 width=16) (actual time=0.135..0.135 rows=1 loops=1)

-> Seq Scan on scientists (cost=0.00..1.23 rows=1 width=8) (actual time=0.062..0.072 rows=4 loops=1)

Filter: ((date_of_birth > '1902-01-01'::date) AND (date_of_birth < '1956-01-01'::date))

Rows Removed by Filter: 11

Planning time: 1.444 ms

Execution time: 0.259 ms

(6 строк)

2) Вывести все достижения, открытые в 20 веке, которые нигде не публиковались/не афишировались, и год открытия достижения

SELECT discovery, year FROM Discoveries WHERE (year > 1900 AND year <2001) AND Discoveries.place ='No' ORDER BY year;

До оптимизации:

QUERY PLAN

Sort (cost=1.41..1.42 rows=1 width=182) (actual time=0.089..0.090 rows=3 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on discoveries (cost=0.00..1.40 rows=1 width=182) (actual time=0.067..0.072 rows=3 loops=1)

Filter: ((year > 1900) AND (year < 2001) AND ((place)::text = 'No'::text))

Rows Removed by Filter: 20

Planning time: 92.424 ms

Execution time: 0.145 ms

(8 строк)

Создание индекса: создаем индекс по году публикации/презентации достижения

CREATE INDEX ON Discoveries (year);

EXPLAIN (ANALYZE) SELECT discovery, year FROM Discoveries WHERE (year > 1900 AND year <2001) AND Discoveries.place ='No' ORDER BY year;

После оптимизации:

QUERY PLAN

Sort (cost=1.41..1.42 rows=1 width=182) (actual time=0.068..0.070 rows=3 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on discoveries (cost=0.00..1.40 rows=1 width=182) (actual time=0.037..0.042 rows=3 loops=1)

Filter: ((year > 1900) AND (year < 2001) AND ((place)::text = 'No'::text))

Rows Removed by Filter: 20

Planning time: 0.651 ms

Execution time: 0.105 ms

(8 строк)

3) Вывести все награды, которые в названии содержат слово Medal и которые были впервые присуждены, начиная с 1892 года

SELECT the_name_of_the_award, year FROM Scientific_awards WHERE (the_name_of_the_award LIKE '%Medal%' AND Scientific_awards.year >= 1892) ORDER BY year;

До оптимизации:

QUERY PLAN

Sort (cost=1.28..1.28 rows=1 width=182) (actual time=23.764..23.766 rows=5 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on scientific_awards (cost=0.00..1.27 rows=1 width=182) (actual time=23.710..23.723 rows=5 loops=1)

Filter: (((the_name_of_the_award)::text ~~ '%Medal%'::text) AND (year >= 1892))

Rows Removed by Filter: 13

Planning time: 91.165 ms

Execution time: 23.821 ms

(8 строк)

Создание индекса: создаем индекс по году (год, в котором была впервые присуждена та или иная награда)

CREATE INDEX ON Scientific_awards (year);

EXPLAIN (ANALYZE) SELECT the_name_of_the_award, year FROM Scientific_awards WHERE (the_name_of_the_award LIKE '%Medal%' AND Scientific_awards.year >= 1892) ORDER BY year;

После оптимизации:

QUERY PLAN

Sort (cost=1.28..1.28 rows=1 width=182) (actual time=0.079..0.079 rows=5 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on scientific_awards (cost=0.00..1.27 rows=1 width=182) (actual time=0.048..0.058 rows=5 loops=1)

Filter: (((the_name_of_the_award)::text ~~ '%Medal%'::text) AND (year >= 1892))

Rows Removed by Filter: 13

Planning time: 1.196 ms

Execution time: 0.123 ms

(8 строк)

4) Вывести 3 самых молодых ученых, рожденных в 19 веке, их возраст и их альма-матер, в названии которого содержится слово University

SELECT name_and_surname, alma_mater, age(date_of_death,date_of_birth) FROM Scientists WHERE (alma_mater LIKE '%University%' AND

Scientists.date_of_birth > '1800-01-01' AND Scientists.date_of_birth < '1901-01-01') ORDER BY age LIMIT 3;

До оптимизации:

QUERY PLAN

Limit (cost=1.28..1.28 rows=1 width=372) (actual time=7.643..7.644 rows=3 loops=1)

  -> Sort (cost=1.28..1.28 rows=1 width=372) (actual time=7.638..7.639 rows=3 loops=1)

        Sort Key: (age((date_of_death)::timestamp with time zone, (date_of_birth)::timestamp with time zone))

        Sort Method: quicksort Memory: 25kB

        -> Seq Scan on scientists (cost=0.00..1.27 rows=1 width=372) (actual time=0.115..0.134 rows=5 loops=1)

              Filter: (((alma_mater)::text ~~ '%University%'::text) AND (date_of_birth > '1800-01-01'::date) AND (date_of_birth < '1901-01-01'::date))

              Rows Removed by Filter: 10

Planning time: 15.132 ms

Execution time: 13.509 ms

(9 строк)

Создание индекса: создаем индекс по дате рождения учёного

CREATE INDEX ON Scientists (date_of_birth);

EXPLAIN (ANALYZE) SELECT name_and_surname, alma_mater, age(date_of_death,date_of_birth) FROM Scientists WHERE (alma_mater LIKE '%University%' AND Scientists.date_of_birth > '1800-01-01' AND Scientists.date_of_birth < '1901-01-01') ORDER BY age LIMIT 3;

После оптимизации:

QUERY PLAN

```
 Limit  (cost=1.28..1.28 rows=1 width=372) (actual time=0.088..0.089 rows=3 loops=1)
   ->  Sort  (cost=1.28..1.28 rows=1 width=372) (actual time=0.087..0.088 rows=3 loops=1)
         Sort Key: (age((date_of_death)::timestamp with time zone, (date_of_birth)::timestamp with time zone))
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on scientists  (cost=0.00..1.27 rows=1 width=372) (actual time=0.048..0.065 rows=5 loops=1)
               Filter: (((alma_mater)::text ~~ '%University%'::text) AND (date_of_birth > '1800-01-01'::date) AND (date_of_birth < '1901-01-01'::date))
               Rows Removed by Filter: 10
 Planning time: 0.789 ms
 Execution time: 0.131 ms
(9 строк)
```

Глава 4

Средние запросы и их оптимизация

1)Вывести всех ученых из Российской империи и их возраст и отсортировать их от младшего к старшему

SELECT name_and_surname, age(date_of_death,date_of_birth) FROM (Scientists INNER JOIN (Towns INNER JOIN Countries ON (Towns.id_Country = Countries.id)) ON (Scientists.town_of_birth_Id = Towns.id)) WHERE Countries.country ='Russian_Empire' ORDER BY age;

До оптимизации:

QUERY PLAN

Sort (cost=36.20..36.21 rows=1 width=194) (actual time=0.325..0.326 rows=5 loops=1)

Sort Key: (age((scientists.date_of_death)::timestamp with time zone, (scientists.date_of_birth)::timestamp with time zone))

Sort Method: quicksort Memory: 25kB

-> Hash Join (cost=23.43..36.19 rows=1 width=194) (actual time=0.270..0.287 rows=5 loops=1)

Hash Cond: (scientists.town_of_birth_id = towns.id)

-> Seq Scan on scientists (cost=0.00..12.00 rows=200 width=190) (actual time=0.030..0.034 rows=15 loops=1)

-> Hash (cost=23.41..23.41 rows=1 width=4) (actual time=0.104..0.104 rows=5 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Hash Join (cost=8.18..23.41 rows=1 width=4) (actual time=0.070..0.080 rows=5 loops=1)

Hash Cond: (towns.id_country = countries.id)

-> Seq Scan on towns (cost=0.00..13.80 rows=380 width=8) (actual time=0.014..0.016 rows=15 loops=1)

-> Hash (cost=8.17..8.17 rows=1 width=4) (actual time=0.040..0.040 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Index Scan using countries_country_key on countries (cost=0.15..8.17 rows=1 width=4) (actual time=0.034..0.035 rows=1 loops=1)

Index Cond: ((country)::text = 'Russian_Empire'::text)

Planning time: 34.404 ms

Execution time: 0.576 ms

(17 строк)

Создание индекса: создание индекса по стране

CREATE INDEX ON Countries (country);

EXPLAIN (ANALYZE) SELECT name_and_surname, age(date_of_death,date_of_birth) FROM (Scientists INNER JOIN (Towns INNER JOIN Countries ON (Towns.id_Country = Countries.id)) ON (Scientists.town_of_birth_Id = Towns.id)) WHERE Countries.country ='Russian_Empire' ORDER BY age;

После оптимизации:

QUERY PLAN

Sort (cost=17.90..17.90 rows=1 width=194) (actual time=0.320..0.323 rows=5 loops=1)

Sort Key: (age((scientists.date_of_death)::timestamp with time zone, (scientists.date_of_birth)::timestamp with time zone))

Sort Method: quicksort Memory: 25kB

-> Hash Join (cost=2.52..17.89 rows=1 width=194) (actual time=0.247..0.290 rows=5 loops=1)

Hash Cond: (towns.id = scientists.town_of_birth_id)

-> Hash Join (cost=1.19..16.43 rows=27 width=4) (actual time=0.117..0.138 rows=5 loops=1)

Hash Cond: (towns.id_country = countries.id)

-> Seq Scan on towns (cost=0.00..13.80 rows=380 width=8) (actual time=0.029..0.037 rows=15 loops=1)

-> Hash (cost=1.18..1.18 rows=1 width=4) (actual time=0.069..0.069 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on countries (cost=0.00..1.18 rows=1 width=4) (actual time=0.054..0.064 rows=1 loops=1)

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

Rows Removed by Filter: 13

-> Hash (cost=1.15..1.15 rows=15 width=190) (actual time=0.094..0.094 rows=15 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientists (cost=0.00..1.15 rows=15 width=190) (actual time=0.045..0.059 rows=15 loops=1)

Planning time: 1.992 ms

Execution time: 0.453 ms

(18 строк)

2) Вывести ученых, которые имеют британские награды, которые впервые были присуждены в период с 1813 по 1850 года, а также название этой награды и год первого присуждения

SELECT name_and_surname, the_name_of_the_award, year FROM (Countries INNER JOIN (Scientific_awards AS sa INNER JOIN (Scientists_Scientific_awards_relation AS ssar INNER JOIN Scientists ON (ssar.id_scientist = Scientists.id)) ON (ssar.id_scientific_award = sa.id)) ON (sa.country_id= Countries.id)) WHERE (Countries. country='Britain' AND (year >= '1813' AND year <= '1850'));

До оптимизации:

QUERY PLAN

Sort (cost=17.90..17.90 rows=1 width=194) (actual time=0.506..0.507 rows=5 loops=1)

Sort Key: (age((scientists.date_of_death)::timestamp with time zone, (scientists.date_of_birth)::timestamp with time zone))

Sort Method: quicksort Memory: 25kB

-> Hash Join (cost=2.52..17.89 rows=1 width=194) (actual time=0.242..0.268 rows=5 loops=1)

Hash Cond: (towns.id = scientists.town_of_birth_id)

-> Hash Join (cost=1.19..16.43 rows=27 width=4) (actual time=0.091..0.106 rows=5 loops=1)

Hash Cond: (towns.id_country = countries.id)

-> Seq Scan on towns (cost=0.00..13.80 rows=380 width=8) (actual time=0.022..0.024 rows=15 loops=1)

-> Hash (cost=1.18..1.18 rows=1 width=4) (actual time=0.044..0.044 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on countries (cost=0.00..1.18 rows=1 width=4) (actual time=0.028..0.034 rows=1 loops=1)

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

Rows Removed by Filter: 13

-> Hash (cost=1.15..1.15 rows=15 width=190) (actual time=0.064..0.064 rows=15 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientists (cost=0.00..1.15 rows=15 width=190) (actual time=0.031..0.038 rows=15 loops=1)

Planning time: 3.371 ms

Execution time: 0.707 ms

(18 строк)

Создание индекса: создаем индекс по году (год, в котором была впервые присуждена та или иная награда)

CREATE INDEX ON Scientific_awards (year);

EXPLAIN (ANALYZE) SELECT name_and_surname, the_name_of_the_award, year FROM (Countries INNER JOIN (Scientific_awards AS sa INNER JOIN (Scientists_Scientific_awards_relation AS ssar INNER JOIN Scientists ON (ssar.id_scientist = Scientists.id)) ON (ssar.id_scientific_award = sa.id)) ON (sa.country_id= Countries.id)) WHERE (Countries.country='Britain' AND (year >= '1813' AND year <= '1850'));

После оптимизации:

QUERY PLAN

Hash Join (cost=3.81..42.46 rows=113 width=360) (actual time=0.226..0.237 rows=3 loops=1)

Hash Cond: (ssar.id_scientist = scientists.id)

-> Hash Join (cost=2.47..40.62 rows=113 width=186) (actual time=0.095..0.105 rows=3 loops=1)

Hash Cond: (ssar.id_scientific_award = sa.id)

-> Seq Scan on scientists_scientific_awards_relation ssar (cost=0.00..30.40 rows=2040 width=8) (actual time=0.018..0.023 rows=27 loops=1)

-> Hash (cost=2.46..2.46 rows=1 width=186) (actual time=0.066..0.066 rows=3 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=0.00..2.46 rows=1 width=186) (actual time=0.048..0.058 rows=3 loops=1)

Join Filter: (countries.id = sa.country_id)

Rows Removed by Join Filter: 2

-> Seq Scan on countries (cost=0.00..1.18 rows=1 width=4) (actual time=0.023..0.025 rows=1 loops=1)

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

Rows Removed by Filter: 13

-> Seq Scan on scientific_awards sa (cost=0.00..1.27 rows=1 width=190) (actual time=0.021..0.025 rows=5 loops=1)

Filter: ((year >= 1813) AND (year <= 1850))

Rows Removed by Filter: 13

-> Hash (cost=1.15..1.15 rows=15 width=182) (actual time=0.107..0.107 rows=15 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientists (cost=0.00..1.15 rows=15 width=182) (actual time=0.083..0.090 rows=15 loops=1)

Planning time: 0.985 ms

Execution time: 0.393 ms

(21 строка)

3) Вывести достижения, открытые учеными, у которых сфера деятельности – математика и которые родились после 01.01.1800, а также указать этих ученых, место и год публикации

SELECT discovery, name_and_surname, place AS place_of_publication, year AS year_of_publication FROM (Scientific_spheres AS ss INNER JOIN (Scientists_Spheres_relation AS ssr INNER JOIN (Scientists INNER JOIN (Scientists_Discoveries_relation AS sdr INNER JOIN Discoveries ON (sdr.id_scientist_discovery = Discoveries.id)) ON (sdr.id_scientist = Scientists.id)) ON (ssr.id_scientist = Scientists.id)) ON (ssr.id_sphere_of_activity = ss.id)) WHERE (ss.scientific_sphere = 'Mathematics' AND Scientists.date_of_birth >='1800-01-01') ORDER BY year_of_publication;

До оптимизации:

QUERY PLAN

Nested Loop (cost=42.25..1151.18 rows=10336 width=738) (actual time=0.487..0.498 rows=1 loops=1)

Join Filter: (sdr.id_scientist = scientists.id)

Rows Removed by Join Filter: 27

-> Nested Loop (cost=42.25..976.36 rows=2315 width=568) (actual time=0.404..0.451 rows=2 loops=1)

Join Filter: (sdr.id_scientist_discovery = discoveries.id)

Rows Removed by Join Filter: 44

-> Index Scan using discoveries_year_idx on discoveries (cost=0.14..12.48 rows=23 width=564) (actual time=0.087..0.102 rows=23 loops=1)

-> Materialize (cost=42.11..170.99 rows=2315 width=12) (actual time=0.014..0.014 rows=2 loops=23)

-> Hash Join (cost=42.11..159.41 rows=2315 width=12) (actual time=0.300..0.307 rows=2 loops=1)

Hash Cond: (sdr.id_scientist = ssr.id_scientist)

-> Seq Scan on scientists_discoveries_relation sdr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.056..0.059 rows=23 loops=1)

-> Hash (cost=39.27..39.27 rows=227 width=4) (actual time=0.215..0.215 rows=2 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Hash Join (cost=1.13..39.27 rows=227 width=4) (actual time=0.194..0.199 rows=2 loops=1)

Hash Cond: (ssr.id_sphere_of_activity = ss.id)

-> Seq Scan on scientists_spheres_relation ssr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.017..0.019 rows=15 loops=1)

-> Hash (cost=1.11..1.11 rows=1 width=4) (actual time=0.149..0.149 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientific_spheres ss (cost=0.00..1.11 rows=1 width=4) (actual time=0.129..0.132 rows=1 loops=1)

Filter: ((scientific_sphere)::text = 'Mathematics'::text)

Rows Removed by Filter: 8

-> Materialize (cost=0.00..1.21 rows=5 width=182) (actual time=0.014..0.020 rows=14 loops=2)

-> Seq Scan on scientists (cost=0.00..1.19 rows=5 width=182) (actual time=0.020..0.025 rows=14 loops=1)

Filter: (date_of_birth >= '1800-01-01'::date)

Rows Removed by Filter: 1

Planning time: 7.472 ms

Execution time: 0.709 ms

(27 строк)

Создание индекса: создаем индекс по научной сфере

CREATE INDEX ON Scientific_spheres (scientific_sphere);

EXPLAIN (ANALYZE) SELECT discovery, name_and_surname, place AS place_of_publication, year AS year_of_publication FROM (Scientific_spheres AS ss INNER JOIN (Scientists_Spheres_relation AS ssr INNER JOIN (Scientists INNER JOIN (Scientists_Discoveries_relation AS sdr INNER JOIN Discoveries ON (sdr.id_scientist_discovery = Discoveries.id)) ON (sdr.id_scientist = Scientists.id)) ON (ssr.id_scientist = Scientists.id)) ON (ssr.id_sphere_of_activity = ss.id)) WHERE (ss.scientific_sphere = 'Mathematics' AND Scientists.date_of_birth >='1800-01-01') ORDER BY year_of_publication;

После оптимизации:

QUERY PLAN

Nested Loop (cost=42.25..1151.18 rows=10336 width=738) (actual time=0.249..0.261 rows=1 loops=1)

Join Filter: (sdr.id_scientist = scientists.id)

Rows Removed by Join Filter: 27

-> Nested Loop (cost=42.25..976.36 rows=2315 width=568) (actual time=0.166..0.214 rows=2 loops=1)

Join Filter: (sdr.id_scientist_discovery = discoveries.id)

Rows Removed by Join Filter: 44

-> Index Scan using discoveries_year_idx on discoveries (cost=0.14..12.48 rows=23 width=564) (actual time=0.022..0.036 rows=23 loops=1)

-> Materialize (cost=42.11..170.99 rows=2315 width=12) (actual time=0.006..0.007 rows=2 loops=23)

-> Hash Join (cost=42.11..159.41 rows=2315 width=12) (actual time=0.134..0.141 rows=2 loops=1)

Hash Cond: (sdr.id_scientist = ssr.id_scientist)

-> Seq Scan on scientists_discoveries_relation sdr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.036..0.039 rows=23 loops=1)

-> Hash (cost=39.27..39.27 rows=227 width=4) (actual time=0.079..0.079 rows=2 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Hash Join (cost=1.13..39.27 rows=227 width=4) (actual time=0.068..0.074 rows=2 loops=1)

Hash Cond: (ssr.id_sphere_of_activity = ss.id)

-> Seq Scan on scientists_spheres_relation ssr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.015..0.019 rows=15 loops=1)

-> Hash (cost=1.11..1.11 rows=1 width=4) (actual time=0.034..0.034 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientific_spheres ss (cost=0.00..1.11 rows=1 width=4) (actual time=0.024..0.026 rows=1 loops=1)

Filter: ((scientific_sphere)::text = 'Mathematics'::text)

Rows Removed by Filter: 8

-> Materialize (cost=0.00..1.21 rows=5 width=182) (actual time=0.013..0.020 rows=14 loops=2)

-> Seq Scan on scientists (cost=0.00..1.19 rows=5 width=182) (actual time=0.020..0.026 rows=14 loops=1)

Filter: (date_of_birth >= '1800-01-01'::date)

Rows Removed by Filter: 1

Planning time: 2.408 ms

Execution time: 0.410 ms

(27 строк)

Глава 5

Сложные запросы

1) Вывести всех ученых, которые работали в самой популярной сфере и у которых есть американские награды, написать название этих наград.

SELECT name_and_surname, the_name_of_the_award FROM (Countries INNER JOIN (Scientific_awards AS sa INNER JOIN (Scientists_Scientific_awards_relation AS ssar INNER JOIN (Scientists AS sct LEFT JOIN (Scientific_spheres AS ss LEFT JOIN Scientists_Spheres_relation AS ssr ON (ss.id = ssr.id_sphere_of_activity)) ON (ssr. id_scientist = sct.id)) ON (ssar.id_scientist = sct.id)) ON (ssar.id_scientific_award =sa.id)) ON (sa.country_id= Countries.id)) WHERE Countries. country='USA' AND scientific_sphere = (SELECT scientific_sphere FROM (Scientific_spheres AS ss LEFT JOIN (Scientists_Spheres_relation AS ssr LEFT JOIN Scientists AS sct ON (ssr. id_scientist = sct.id)) ON (ss.id = ssr.id_sphere_of_activity)) GROUP BY ss.scientific_sphere ORDER BY count(sct.id) DESC LIMIT 1);

2)Вывести количество достижений, которые открыли ученые из Российской империи, имеющие немецкие награды, учреждённые с 1900 года.

WITH m1 AS (SELECT discovery,name_and_surname FROM (Discoveries AS disc LEFT JOIN(Scientists_Discoveries_relation AS sdr LEFT JOIN (Scientists AS sct LEFT JOIN (Towns LEFT JOIN Countries ON (Towns.id_Country = Countries.id)) ON (sct.town_of_birth_Id = Towns.id)) ON (sdr.id_scientist = sct.id)) ON (disc.id = sdr.id_scientist_discovery AND country ='Russian_Empire'))), m2 AS (SELECT name_and_surname FROM (Countries INNER JOIN (Scientific_awards AS sa INNER JOIN (Scientists_Scientific_awards_relation AS ssar INNER JOIN Scientists AS sct ON (ssar.id_scientist = sct.id)) ON (ssar.id_scientific_award = sa.id)) ON (sa.country_id= Countries.id AND country='Germany' AND year >= '1900'))) SELECT count(discovery) FROM m1 WHERE name_and_surname IN (SELECT name_and_surname FROM m2);

3) Вывести все награды, в названии которых содержится слово Medal и которые имеют ученые, у которых научная сфера входит в тройку самых популярных сфер из имеющихся и которые сделали открытия после 1800 года, также вывести имена и фамилии этих ученых.

WITH g1 AS (SELECT name_and_surname, the_name_of_the_award FROM (Scientific_awards AS sa INNER JOIN (Scientists_Scientific_awards_relation AS ssar INNER JOIN (Scientists AS sct LEFT JOIN (Scientific_spheres AS ss LEFT JOIN Scientists_Spheres_relation AS ssr ON (ss.id = ssr.id_sphere_of_activity)) ON (ssr. id_scientist = sct.id)) ON (ssar.id_scientist = sct.id)) ON

```
(ssar.id_scientific_award =sa.id AND the_name_of_the_award LIKE '%Medal%'
AND scientific_sphere IN (SELECT scientific_sphere FROM (Scientific_spheres
AS ss LEFT JOIN (Scientists_Spheres_relation AS ssr LEFT JOIN Scientists AS
sct ON (ssr. id_scientist = sct.id)) ON (ss.id = ssr.id_sphere_of_activity)) GROUP
BY ss.scientific_sphere ORDER BY count(sct.id) DESC LIMIT 3)))), g2 AS
(SELECT name_and_surname, discovery FROM ( Discoveries AS disc INNER
JOIN( Scientists_Discoveries_relation AS sdr INNER JOIN Scientists AS sct
ON(sdr.id_scientist = sct.id)) ON (disc.id = sdr.id_scientist_discovery)) WHERE
disc.year >= '1900') SELECT the_name_of_the_award, name_and_surname
FROM g1 WHERE name_and_surname IN (SELECT name_and_surname FROM
g2) ORDER BY name_and_surname;
```