Легкие запросы

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=13.01..13.02 rows=1 width=16) (actual time=0.224..0.224 rows=1 loops=1)

-> Seq Scan on scientists (cost=0.00..13.00 rows=1 width=8) (actual time=0.113..0.125 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: 13.306 ms

Execution time: 46.745 ms

(6 строк)

Создание индекса:

CREATE INDEX ON Scientists (AVG);

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=13.01..13.02 rows=1 width=16) (actual time=0.168..0.168 rows=1 loops=1)

-> Seq Scan on scientists (cost=0.00..13.00 rows=1 width=8) (actual time=0.081..0.092 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: 0.347 ms

Execution time: 0.353 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=11.76..11.77 rows=1 width=182) (actual time=0.095..0.096 rows=3 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on discoveries (cost=0.00..11.75 rows=1 width=182) (actual time=0.054..0.062 rows=3 loops=1)

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

Rows Removed by Filter: 20

Planning time: 0.307 ms

Execution time: 0.154 ms

(8 строк)

Создание индекса:

CREATE INDEX ON Discoveries (discovery);

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=0.103..0.104 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=0.062..0.074 rows=3 loops=1)

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

Rows Removed by Filter: 20

Planning time: 2.033 ms

Execution time: 0.166 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=15.56..15.57 rows=1 width=182) (actual time=0.140..0.141 rows=5 loops=1)

Sort Key: year

Sort Method: quicksort Memory: 25kB

-> Seq Scan on scientific\_awards (cost=0.00..15.55 rows=1 width=182) (actual time=0.071..0.080 rows=5 loops=1)

Filter: (((the\_name\_of\_the\_award)::text ~~ '%Medal%'::text) AND (year >= 1892))

Rows Removed by Filter: 13

Planning time: 0.544 ms

Execution time: 0.222 ms

(8 строк)

Создание индекса:

CREATE INDEX ON Scientific\_awards (the\_name\_of\_the\_award);

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=0.132..0.135 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=0.076..0.095 rows=5 loops=1)

Filter: (((the\_name\_of\_the\_award)::text ~~ '%Medal%'::text) AND (year >= 1892))

Rows Removed by Filter: 13

Planning time: 2.813 ms

Execution time: 0.211 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=13.52..13.52 rows=1 width=372) (actual time=0.138..0.138 rows=3 loops=1)

-> Sort (cost=13.52..13.52 rows=1 width=372) (actual time=0.135..0.136 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..13.51 rows=1 width=372) (actual time=0.087..0.104 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: 0.321 ms

Execution time: 0.195 ms

(9 строк)

Создание индекса:

CREATE INDEX ON Scientists (name\_and\_surname);

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=0.161..0.162 rows=3 loops=1)

-> Sort (cost=1.28..1.28 rows=1 width=372) (actual time=0.158..0.160 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=0.084..0.118 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: 1.319 ms

Execution time: 0.249 ms

(9 строк)

5)Вывести 7 самых старших ученых, их альма-матер и возраст

SELECT name\_and\_surname, alma\_mater, age(date\_of\_death,date\_of\_birth) FROM scientists ORDER BY age DESC LIMIT 7;

До оптимизации:

QUERY PLAN

----------------------------------------------------------------------------------------------------------------------

Limit (cost=17.31..17.32 rows=7 width=372) (actual time=0.101..0.103 rows=7 loops=1)

-> Sort (cost=17.31..17.81 rows=200 width=372) (actual time=0.100..0.102 rows=7 loops=1)

Sort Key: (age((date\_of\_death)::timestamp with time zone, (date\_of\_birth)::timestamp with time zone)) DESC

Sort Method: top-N heapsort Memory: 25kB

-> Seq Scan on scientists (cost=0.00..13.50 rows=200 width=372) (actual time=0.037..0.071 rows=15 loops=1)

Planning time: 0.227 ms

Execution time: 0.140 ms

(7 строк)

Создание индекса:

CREATE INDEX ON Scientists (name\_and\_surname);

EXPLAIN (ANALYZE) SELECT name\_and\_surname, alma\_mater, age(date\_of\_death,date\_of\_birth) FROM scientists ORDER BY age DESC LIMIT 7;

После оптимизации:

QUERY PLAN

--------------------------------------------------------------------------------------------------------------------

Limit (cost=1.55..1.57 rows=7 width=372) (actual time=0.315..0.317 rows=7 loops=1)

-> Sort (cost=1.55..1.59 rows=15 width=372) (actual time=0.309..0.310 rows=7 loops=1)

Sort Key: (age((date\_of\_death)::timestamp with time zone, (date\_of\_birth)::timestamp with time zone)) DESC

Sort Method: top-N heapsort Memory: 25kB

-> Seq Scan on scientists (cost=0.00..1.26 rows=15 width=372) (actual time=0.229..0.265 rows=15 loops=1)

Planning time: 1.630 ms

Execution time: 0.379 ms

(7 строк)

Средние запросы

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:

CREATE INDEX ON Scientists (name\_and\_surname);

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=21.29..21.29 rows=1 width=194) (actual time=0.637..0.639 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

-> Nested Loop (cost=1.48..21.28 rows=1 width=194) (actual time=0.274..0.546 rows=5 loops=1)

-> Hash Join (cost=1.34..16.71 rows=15 width=190) (actual time=0.213..0.276 rows=15 loops=1)

Hash Cond: (towns.id = scientists.town\_of\_birth\_id)

-> Seq Scan on towns (cost=0.00..13.80 rows=380 width=8) (actual time=0.102..0.120 rows=15 loops=1)

-> Hash (cost=1.15..1.15 rows=15 width=190) (actual time=0.083..0.083 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=0.034..0.048 rows=15 loops=1)

-> Index Scan using countries\_pkey on countries (cost=0.15..0.29 rows=1 width=4) (actual time=0.012..0.012 rows=0 loops=15)

Index Cond: (id = towns.id\_country)

Filter: ((country)::text = 'Russian\_Empire'::text)

Rows Removed by Filter: 1

Planning time: 2.238 ms

Execution time: 0.833 ms

(16 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

-----------------------------------------------------------------------------------------------------------------------------------------------------------

Nested Loop (cost=15.87..63.21 rows=6 width=360) (actual time=0.438..0.469 rows=3 loops=1)

-> Nested Loop (cost=15.72..62.04 rows=6 width=186) (actual time=0.413..0.429 rows=3 loops=1)

Join Filter: (sa.country\_id = countries.id)

Rows Removed by Join Filter: 2

-> Index Scan using countries\_country\_key on countries (cost=0.15..8.17 rows=1 width=4) (actual time=0.113..0.114 rows=1 loops=1)

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

-> Hash Join (cost=15.58..53.73 rows=11 width=190) (actual time=0.290..0.303 rows=5 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=0.158..0.162 rows=27 loops=1)

-> Hash (cost=15.55..15.55 rows=2 width=190) (actual time=0.091..0.091 rows=5 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientific\_awards sa (cost=0.00..15.55 rows=2 width=190) (actual time=0.067..0.073 rows=5 loops=1)

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

Rows Removed by Filter: 13

-> Index Scan using scientists\_pkey on scientists (cost=0.14..0.19 rows=1 width=182) (actual time=0.009..0.010 rows=1 loops=3)

Index Cond: (id = ssar.id\_scientist)

Planning time: 1.090 ms

Execution time: 0.678 ms

(18 строк)

Создание индекса:

CREATE INDEX ON Scientists (name\_and\_surname);

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=10.80..49.45 rows=113 width=360) (actual time=0.408..0.436 rows=3 loops=1)

Hash Cond: (ssar.id\_scientist = scientists.id)

-> Hash Join (cost=9.46..47.61 rows=113 width=186) (actual time=0.242..0.267 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=0.032..0.040 rows=27 loops=1)

-> Hash (cost=9.45..9.45 rows=1 width=186) (actual time=0.189..0.189 rows=3 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=0.15..9.45 rows=1 width=186) (actual time=0.152..0.170 rows=3 loops=1)

Join Filter: (countries.id = sa.country\_id)

Rows Removed by Join Filter: 2

-> Index Scan using countries\_country\_key on countries (cost=0.15..8.17 rows=1 width=4) (actual time=0.052..0.054 rows=1 loops=1)

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

-> Seq Scan on scientific\_awards sa (cost=0.00..1.27 rows=1 width=190) (actual time=0.047..0.058 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=0.145..0.145 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=0.100..0.112 rows=15 loops=1)

Planning time: 3.486 ms

Execution time: 0.642 ms

(20 строк)

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

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=89.85..89.90 rows=20 width=738) (actual time=0.290..0.291 rows=1 loops=1)

Sort Key: discoveries.year

Sort Method: quicksort Memory: 25kB

-> Nested Loop (cost=47.44..89.42 rows=20 width=738) (actual time=0.257..0.260 rows=1 loops=1)

-> Hash Join (cost=47.30..85.55 rows=20 width=182) (actual time=0.247..0.250 rows=1 loops=1)

Hash Cond: (sdr.id\_scientist = scientists.id)

-> Seq Scan on scientists\_discoveries\_relation sdr (cost=0.00..30.40 rows=2040 width=8) (actual time=0.099..0.101 rows=23 loops=1)

-> Hash (cost=47.27..47.27 rows=2 width=186) (actual time=0.128..0.128 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Nested Loop (cost=8.32..47.27 rows=2 width=186) (actual time=0.108..0.123 rows=1 loops=1)

-> Hash Join (cost=8.18..46.28 rows=5 width=4) (actual time=0.093..0.100 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=0.019..0.023 rows=15 loops=1)

-> Hash (cost=8.17..8.17 rows=1 width=4) (actual time=0.057..0.057 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Index Scan using scientific\_spheres\_scientific\_sphere\_key on scientific\_spheres ss (cost=0.15..8.17 rows=1 width=4) (actual time=0.044..0.045 rows=1 loops=1)

Index Cond: ((scientific\_sphere)::text = 'Mathematics'::text)

-> Index Scan using scientists\_pkey on scientists (cost=0.14..0.19 rows=1 width=182) (actual time=0.008..0.009 rows=1 loops=2)

Index Cond: (id = ssr.id\_scientist)

Filter: (date\_of\_birth >= '1800-01-01'::date)

Rows Removed by Filter: 1

-> Index Scan using discoveries\_pkey on discoveries (cost=0.14..0.18 rows=1 width=564) (actual time=0.007..0.008 rows=1 loops=1)

Index Cond: (id = sdr.id\_scientist\_discovery)

Planning time: 2.038 ms

Execution time: 0.515 ms

(25 строк)

Создание индекса:

CREATE INDEX ON Discoveries (discovery);

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

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=99.12..99.80 rows=272 width=738) (actual time=0.382..0.382 rows=1 loops=1)

Sort Key: discoveries.year

Sort Method: quicksort Memory: 25kB

-> Hash Join (cost=49.11..88.12 rows=272 width=738) (actual time=0.361..0.364 rows=1 loops=1)

Hash Cond: (sdr.id\_scientist = scientists.id)

-> Hash Join (cost=47.86..86.67 rows=51 width=568) (actual time=0.224..0.235 rows=2 loops=1)

Hash Cond: (sdr.id\_scientist\_discovery = discoveries.id)

-> Hash Join (cost=46.34..84.90 rows=51 width=12) (actual time=0.154..0.164 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=0.019..0.022 rows=23 loops=1)

-> Hash (cost=46.28..46.28 rows=5 width=4) (actual time=0.121..0.121 rows=2 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Hash Join (cost=8.18..46.28 rows=5 width=4) (actual time=0.104..0.115 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..0.024 rows=15 loops=1)

-> Hash (cost=8.17..8.17 rows=1 width=4) (actual time=0.072..0.072 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Index Scan using scientific\_spheres\_scientific\_sphere\_key on scientific\_spheres ss (cost=0.15..8.17 rows=1 width=4) (actual time=0.044..0.046 rows=1 loops=1)

Index Cond: ((scientific\_sphere)::text = 'Mathematics'::text)

-> Hash (cost=1.23..1.23 rows=23 width=564) (actual time=0.059..0.059 rows=23 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 11kB

-> Seq Scan on discoveries (cost=0.00..1.23 rows=23 width=564) (actual time=0.021..0.032 rows=23 loops=1)

-> Hash (cost=1.19..1.19 rows=5 width=182) (actual time=0.115..0.115 rows=14 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on scientists (cost=0.00..1.19 rows=5 width=182) (actual time=0.080..0.091 rows=14 loops=1)

Filter: (date\_of\_birth >= '1800-01-01'::date)

Rows Removed by Filter: 1

Planning time: 2.344 ms

Execution time: 0.593 ms

(29 строк)

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, которые имеют ученые, у которых научная сфера входит в тройку самых непопулярных 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;