Домашнее задание № 2

1)demo=# INSERT INTO aircrafts

VALUES ( 'SU9', 'Sukhoi SuperJet-100', 3000 );

ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "aircrafts\_pkey"

ПОДРОБНОСТИ: Ключ "(aircraft\_code)=(SU9)" уже существует.

aircraft\_code | model | range

---------------+---------------------+-------

SU9 | Sukhoi SuperJet-100 | 3000

Ошибка возникла из-за того, что запись с aircraft\_code (PK) = SU9 уже есть

2) demo=# select \* from aircrafts order by (range) desc;

aircraft\_code | model | range

---------------+---------------------+-------

773 | Boeing 777-300 | 11100

763 | Boeing 767-300 | 7900

319 | Airbus A319-100 | 6700

320 | Airbus A320-200 | 5700

321 | Airbus A321-200 | 5600

733 | Boeing 737-300 | 4200

SU9 | Sukhoi SuperJet-100 | 3000

CR2 | Bombardier CRJ-200 | 2700

CN1 | Cessna 208 Caravan | 1200

(9 строк)

3) demo=# update aircrafts set range = range \* 2 where aircraft\_code = 'SU9';

UPDATE 1

aircraft\_code | model | range

---------------+---------------------+-------

SU9 | Sukhoi SuperJet-100 | 6000

4) demo=# delete from aircrafts where range < 100;

DELETE 0

Домашнее задание № 3

2) demo=# select \* from test\_numeric ;

measurement | description

------------------------+----------------------------------------------------

1234567890.0987654321 | Точность 20 знаков, масштаб 10 знаков

1.5 | Точность 2 знака, масштаб 1 знак

0.12345678901234567890 | Точность 21 знак, масштаб 20 знаков

1234567890 | Точность 10 знаков, масштаб 0 знаков (целое число)

(4 строки)

4) SELECT '1e+308'::double precision + '1e+291'::double precision = '1e+308'::double precision + '0.0'::double precision;

?column?

----------

t

(1 строка)

SELECT '1e+38'::real + '1e+30'::real = '1e+38'::real + '0.0'::real;

?column?

----------

t

(1 строка)

8) При вставке записи с явным указанием id = 2 последовательность не обновляется и следующий id при вставке без явно указания будет 2, но запись с id = 2 уже существует из-за чего появляется ошибка об необходимости вводить уникальные значения для PK. После выполнения команды с ошибкой значение id инкрементируется и следующее значение id = 3, т.к. в таблице ещё не существует записи с id = 3 операция завершается успешно.

Убедился:  
id | name

----+------------

1 | Вишневая

2 | Прохладная

3 | Грушевая

5 | Луговая

(4 строки)

12) Повторим одну из команд, выполненных ранее. Теперь она должна вызвать  
ошибку. Почему?  
SELECT '18-05-2016'::date;

ОШИБКА: значение поля типа date/time вне диапазона: "18-05-2016"

СТРОКА 1: SELECT '18-05-2016'::date;

^

ПОДСКАЗКА: Возможно, вам нужно изменить настройку "datestyle".

Это происходит из-за того, что введёный месяц больше чем колличество месяцев в году. Ошибка возникает из-за изменения формата даты. Следующая команда не вызовет ошибок из-за того, что все поля не превышают ограничений для текущего формата даты.

demo=# SET datestyle TO 'German, DMY';

SET

demo=# SELECT '14.7.2000'::date;

date

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

14.07.2000

(1 строка)

demo=# SELECT '11.13.2000'::date;

ОШИБКА: значение поля типа date/time вне диапазона: "11.13.2000"

СТРОКА 1: SELECT '11.13.2000'::date;

^

ПОДСКАЗКА: Возможно, вам нужно изменить настройку "datestyle".

Ошибка из-за ввода месяца больше чем их колличества в году.

demo=# SET datestyle TO 'German, MDY';

SET

demo=# SELECT '11.13.2000'::date;

date

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

13.11.2000

(1 строка)

После изменения формата даты, ошибка не возникает.

15) Сначала попробуйте разобраться, не обращаясь к документации,  
в том, что означает второй параметр этой функции в каждой из приведенных  
команд, а затем проверьте свои предположения по документации:  
 (предроложение) Второй параметр представляет из себя шаблон по которому производится вывод приведённого к символьному типу значения current\_timestamp

(предроложение) Второй параметр представляет из себя шаблон по которому производится вывод приведённого к символьному типу значения current\_timestamp.  
 (Документация)Шаблон вывода to\_char может содержать ряд кодов, которые распознаются при форматировании и заменяются соответствующими данными. Любой текст, который не является кодом, копируется в результат в неизменном виде. Подобным образом, в строке шаблона ввода (для других функций) коды шаблона определяют, какие значения содержит передаваемая текстовая строка.

Поэкспериментируйте с этой функцией, извлекая из значения типа timestamp  
различные поля и располагая их в нужном вам порядке:

demo=# SELECT to\_char( current\_timestamp, 'dd:mm:yyyy:hh:mi:ss' );

to\_char

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

15:10:2022:04:21:09

(1 строка)

demo=# SELECT to\_char( current\_timestamp, 'hh:mi:ss:dd:mm:yyyy');

to\_char

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

04:23:34:15:10:2022

(1 строка)

21)Можно с высокой степенью уверенности предположить, что при прибавлении  
интервалов к датам и временн ́ым отметкам PostgreSQL учитывает тот факт,  
что различные месяцы имеют различное число дней. Но как это реализуется  
на практике? Например, что получится при прибавлении интервала в 1 месяц  
к последнему дню января и к последнему дню февраля? Сначала сделайте обос-  
нованные предположения о результатах следующих двух команд:  
Обе команды должны увеличить значение месяца на единицу, но из-за того что число дней в месяцах различается, придётся отслеживать ещё и дату. В случае если дата превышает кол-во дней в обновлённом месяце берётся последнее число в месяце.

Затем проверьте предположения на практике и проанализируйте полученные результаты:

demo=# SELECT ( '2016-01-31'::date + '1 mon'::interval ) AS new\_date;

new\_date

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

2016-02-29 00:00:00

(1 строка)

Как и ожидалось, дата стала последним днём февраля.

demo=# SELECT ( '2016-02-29'::date + '1 mon'::interval ) AS new\_date;

new\_date

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

2016-03-29 00:00:00

(1 строка)

Месяц просто инкрементировался.

30) Обратимся к таблице, создаваемой с помощью команды  
CREATE TABLE test\_bool  
( a boolean,  
b text  
);  
Как вы думаете, какие из приведенных ниже команд содержат ошибку?

INSERT INTO test\_bool VALUES ( TRUE, 'yes' );  
INSERT INTO test\_bool VALUES ( yes, 'yes' );  
INSERT INTO test\_bool VALUES ( 'yes', true );  
INSERT INTO test\_bool VALUES ( 'yes', TRUE );  
INSERT INTO test\_bool VALUES ( '1', 'true' );  
INSERT INTO test\_bool VALUES ( 1, 'true' );  
INSERT INTO test\_bool VALUES ( 't', 'true' );  
INSERT INTO test\_bool VALUES ( 't', truth );  
INSERT INTO test\_bool VALUES ( true, true );  
INSERT INTO test\_bool VALUES ( 1::boolean, 'true' );  
INSERT INTO test\_bool VALUES ( 111::boolean, 'true' );

Я считаю, что команды №2,3,4,6,8 содержут ошибку

Проверьте свои предположения практически, выполнив эти команды:

demo=# INSERT INTO test\_bool VALUES ( TRUE, 'yes' );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( yes, 'yes' );

ОШИБКА: столбец "yes" не существует

СТРОКА 1: INSERT INTO test\_bool VALUES ( yes, 'yes' );

demo=# INSERT INTO test\_bool VALUES ( 'yes', true );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( 'yes', TRUE );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( '1', 'true' );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( 1, 'true' );

ОШИБКА: столбец "a" имеет тип boolean, а выражение - integer

СТРОКА 1: INSERT INTO test\_bool VALUES ( 1, 'true' );

^

ПОДСКАЗКА: Перепишите выражение или преобразуйте его тип.

demo=# INSERT INTO test\_bool VALUES ( 't', 'true' );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( 't', truth );

ОШИБКА: столбец "truth" не существует

СТРОКА 1: INSERT INTO test\_bool VALUES ( 't', truth );

demo=# INSERT INTO test\_bool VALUES ( true, true );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( 1::boolean, 'true' );

INSERT 0 1

demo=# INSERT INTO test\_bool VALUES ( 111::boolean, 'true' );

INSERT 0 1

33)Задание. Создайте новую версию таблицы и соответственно измените коман-  
ду INSERT, чтобы в ней содержались литералы двумерных массивов. Они будут  
выглядеть примерно так:  
'{ { "сосиска", "макароны", "кофе" },  
{ "котлета", "каша", "кофе" },  
{ "сосиска", "каша", "кофе" },  
{ "котлета", "каша", "чай" } }'::text[][]  
Сделайте ряд выборок и обновлений строк в этой таблице. Для обращения к эле-  
ментам двумерного массива нужно использовать два индекса. Не забывайте,  
что по умолчанию номера индексов начинаются с единицы.

CREATE TABLE pilots

( pilot\_name text,

schedule integer[],

meal text[][]

);

CREATE TABLE

demo=# INSERT INTO pilots

VALUES ( 'Ivan', '{ 1, 3, 5, 6, 7 }'::integer[],

'{ { "сосиска", "макароны", "кофе" },

{ "котлеты", "пюре", "чай" },

{ "куриная грудка", "рис", "сок" },

{ "шашлык", "картофель", "компот" }

}'::text[][]

),

( 'Petr', '{ 1, 2, 5, 7 }'::integer [],

'{ { "лапша", "котлета", "чай" },

{ "рыба", "пюре", "кофе" },

{ "суп куриный", "стейк", "чай" },

{ "сосиска", "гречневая крупа", "кофе" }

}'::text[][]

),

( 'Pavel', '{ 2, 5 }'::integer[],

'{ { "борщ", "отбивная", "компот" },

{ "котлета", "пюре", "компот" },

{ "рыба", "стейк", "компот" },

{ "шашлык", "картофель", "компот" }

}'::text[][]

);'::text[][]"каша", "чай" }й" },[],

INSERT 0 4

demo=# select \* from pilots;

pilot\_name | schedule | meal

------------+-------------+-----------------------------------------------------

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

Ivan | {1,3,5,6,7} | {{сосиска,макароны,кофе},{котлеты,пюре,чай},{"курина

я грудка",рис,сок},{шашлык,картофель,компот}}

Petr | {1,2,5,7} | {{лапша,котлета,чай},{рыба,пюре,кофе},{"суп куриный"

,стейк,чай},{сосиска,"гречневая крупа",кофе}}

Pavel | {2,5} | {{борщ,отбивная,компот},{котлета,пюре,компот},{рыба,

стейк,компот},{шашлык,картофель,компот}}

Boris | {3,5,6} | {{лапша,сосиска,чай},{стейк,пюре,чай},{пюре,отбивная

,чай},{борщ,каша,чай}}

(4 строки)

demo=# SELECT \* FROM pilots WHERE meal[1][3] = 'чай';

pilot\_name | schedule | meal

------------+-----------+--------------------------------------------------------------------------------------

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

Petr | {1,2,5,7} | {{лапша,котлета,чай},{рыба,пюре,кофе},{"суп куриный",стейк,чай},{сосиска,"гречневая к

рупа",кофе}}

Boris | {3,5,6} | {{лапша,сосиска,чай},{стейк,пюре,чай},{пюре,отбивная,чай},{борщ,каша,чай}}

(2 строки)

demo=# SELECT \* FROM pilots WHERE meal[3][2] = 'стейк';

pilot\_name | schedule | meal

------------+-----------+---------------------------------------------------------------------------------------------------

Petr | {1,2,5,7} | {{лапша,котлета,чай},{рыба,пюре,кофе},{"суп куриный",стейк,чай},{сосиска,"гречневая крупа",кофе}}

Pavel | {2,5} | {{борщ,отбивная,компот},{котлета,пюре,компот},{рыба,стейк,компот},{шашлык,картофель,компот}}

(2 строки)

35)Изучая приемы работы с типами JSON, можно, как и в случае с массивами, поль-  
зоваться способностью команды SELECT обходиться без создания таблиц.  
Покажем лишь один пример. Добавить новый ключ и соответствующее ему зна-  
чения в уже существующий объект можно оператором ||:  
SELECT '{ "sports": "хоккей" }'::jsonb || '{ "trips": 5 }'::jsonb;  
?column?  
----------------------------------  
{"trips": 5, "sports": "хоккей"}  
(1 строка)  
Для работы с типами JSON предусмотрено много различных функций и опе-  
раторов, представленных в разделе документации 9.15 «Функции и операторы  
JSON». Самостоятельно ознакомьтесь с ними, используя описанную технологию  
работы с командой SELECT.  
  
demo=# SELECT jsonb\_each('{ "sports": "хоккей" }'::jsonb || '{ "trips": 5 }'::jsonb);

jsonb\_each

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

(trips,5)

(sports,"""хоккей""")

(2 строки)

demo=# SELECT jsonb\_object\_keys('{ "sports": "хоккей" }'::jsonb || '{ "trips": 5 }'::jsonb);

jsonb\_object\_keys

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

trips

sports

(2 строки)

demo=# SELECT jsonb\_typeof('{ "sports": "хоккей" }'::jsonb || '{ "trips": 5 }'::jsonb);

jsonb\_typeof

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

object

(1 строка)

demo=# SELECT jsonb\_pretty('{ "sports": "хоккей" }'::jsonb || '{ "trips": 5 }'::jsonb);

jsonb\_pretty

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

{ +

"trips": 5, +

"sports": "хоккей"+

}

(1 строка)

ДЗ№4

2)Добавьте в таблицу progress  
еще один атрибут — «Форма проверки знаний» (test\_form), который может  
принимать только два значения: «экзамен» или «зачет». Тогда набор допусти-  
мых значений атрибута «Оценка» (mark) будет зависеть от того, экзамен или за-  
чет предусмотрены по данной дисциплине. Если предусмотрен экзамен, тогда  
допускаются значения 3, 4, 5, если зачет — тогда 0 (не зачтено) или 1 (зачтено).

demo=# alter table progress add column test\_form text not null;

ALTER TABLE

demo=# alter table progress add constraint check\_test\_form check (( test\_form = 'экзамен' and mark in(3,4,5)) or (test\_form = 'зачёт' and mark in (0,1)));

ALTER TABLE

В таблице уже было ограничение на допустимые значения атрибута mark. Как  
вы думаете, не будет ли оно конфликтовать с новым ограничением? Да, конфликт будет при добавлении значиений 0, 1 для типа оценивания — зачёт.

Добавляем данные:

INSERT INTO students VALUES (11111, 'Ivanov I.I', 1111, 111111);

INSERT 0 1

demo=# INSERT INTO progress VALUES (11111, 'Math', 'first', 1, 5, 'экзамен');

INSERT 0 1

demo=# INSERT INTO progress VALUES (11111, 'Math', 'first', 1, 1, 'зачёт');

ОШИБКА: новая строка в отношении "progress" нарушает ограничение-проверку "progress\_mark\_check"

ПОДРОБНОСТИ: Ошибочная строка содержит (11111, Math, first, 1, 1, зачёт).

demo=# alter table progress drop CONSTRAINT progress\_mark\_check;

ALTER TABLE

INSERT INTO progress VALUES (11111, 'Math', 'first', 1, 1, 'зачёт');

INSERT 0 1

Теперь нет конфликтов и запись успешно добавляется.

9) В таблице «Студенты» (students) есть текстовый атрибут name, на который на-  
ложено ограничение NOT NULL. Как вы думаете, что будет, если при вводе новой  
строки в эту таблицу дать атрибуту name в качестве значения пустую строку?

Пустая строка не null запрос выполнится.

demo=# INSERT INTO students ( record\_book, name, doc\_ser, doc\_num )

VALUES ( 12300, '', 0402, 543281 );

INSERT 0 1

demo=# ALTER TABLE students ADD CHECK ( name <> '' );

ALTER TABLE

Ожидаемо, проверка условия не позволяет добавить запись в таблицу.

demo=# INSERT INTO students ( record\_book, name, doc\_ser, doc\_num )

VALUES ( 12300, '', 0402, 543281 );

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students\_name\_check"

ПОДРОБНОСТИ: Ошибочная строка содержит (12300, , 402, 543281).

demo=# INSERT INTO students VALUES ( 12346, ' ', 0406, 112233 );

INSERT INTO students VALUES ( 12347, ' ', 0407, 112234 );

INSERT 0 1

INSERT 0 1

demo=# SELECT \*, length( name ) FROM students;

record\_book | name | doc\_ser | doc\_num | length

-------------+------------+---------+---------+--------

11111 | Ivanov I.I | 1111 | 111111 | 10

12346 | | 406 | 112233 | 1

12347 | | 407 | 112234 | 1

(3 строки)

demo=# ALTER TABLE students ADD CHECK (length(trim(name)) <> 0);

ALTER TABLE

demo=# INSERT INTO students VALUES ( 12346, ' ', 0406, 112233 );

INSERT INTO students VALUES ( 12347, ' ', 0407, 112234 );

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students\_name\_check"

ПОДРОБНОСТИ: Ошибочная строка содержит (12346, , 406, 112233).

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students\_name\_check"

ПОДРОБНОСТИ: Ошибочная строка содержит (12347, , 407, 112234).

Есть ли подобные слабые места в таблице «Успеваемость» (progress)?

Есть, в полях subject & acad\_year.

17)Подумайте, какие представления было бы целесообразно создать для нашей  
базы данных «Авиаперевозки». Необходимо учесть наличие различных групп  
пользователей, например: пилоты, диспетчеры, пассажиры, кассиры.  
Создайте представления и проверьте их в работе.

demo=# create view longrange\_boeings as select model, aircraft\_code from aircrafts where range > 6000 and model like 'Boeing%';

CREATE VIEW

demo=# select \* from longrange\_boeings;

model | aircraft\_code

----------------+---------------

Boeing 777-300 | 773

Boeing 767-300 | 763

(2 строки)

Представление которое показывает все самолёты компинии «Boeing», имеющие дальность больше 5тыс км.

create view from\_Moscow\_to\_ROV\_after\_6\_clock as select actual\_departure, arrival\_airport, departure\_airport from flights where departure\_airport in ('VKO', 'SVO', 'DME') and actual\_departure is not null and extract(hour from actual\_departure) > 18 and arrival\_airport = 'ROV' order by actual\_departure;

CREATE VIEW

actual\_departure | arrival\_airport | departure\_airport

------------------------+-----------------+-------------------

2017-07-16 19:27:00+03 | ROV | DME

2017-07-17 19:23:00+03 | ROV | DME

2017-07-18 19:22:00+03 | ROV | DME

2017-07-19 19:24:00+03 | ROV | DME

2017-07-20 19:21:00+03 | ROV | DME

2017-07-21 19:26:00+03 | ROV | DME

2017-07-22 19:26:00+03 | ROV | DME

2017-07-23 19:23:00+03 | ROV | DME

2017-07-24 19:22:00+03 | ROV | DME

2017-07-25 19:22:00+03 | ROV | DME

2017-07-26 19:22:00+03 | ROV | DME

2017-07-27 19:21:00+03 | ROV | DME

2017-07-27 20:12:00+03 | ROV | SVO

2017-07-28 19:24:00+03 | ROV | DME

2017-07-29 19:23:00+03 | ROV | DME

2017-07-30 19:24:00+03 | ROV | DME

2017-07-31 19:21:00+03 | ROV | DME

2017-08-01 19:26:00+03 | ROV | DME

2017-08-02 19:21:00+03 | ROV | DME

2017-08-03 19:23:00+03 | ROV | DME

2017-08-04 19:22:00+03 | ROV | DME

2017-08-05 19:23:00+03 | ROV | DME

2017-08-05 20:59:00+03 | ROV | DME

2017-08-06 19:22:00+03 | ROV | DME

2017-08-07 19:24:00+03 | ROV | DME

2017-08-08 19:24:00+03 | ROV | DME

2017-08-09 19:24:00+03 | ROV | DME

2017-08-10 22:58:00+03 | ROV | DME

2017-08-11 19:22:00+03 | ROV | DME

2017-08-12 19:24:00+03 | ROV | DME

2017-08-13 19:23:00+03 | ROV | DME

2017-08-13 19:51:00+03 | ROV | SVO

2017-08-14 22:00:00+03 | ROV | DME

(33 rows)

Представление — все вылеты из москвы в Ростов-на-дону после 6 вечера по местному времени вылета

8)Задание. Подумайте, какие еще таблицы было бы целесообразно дополнить  
столбцами типа json/jsonb. Вспомните, что, например, в таблице «Билеты»  
(tickets) уже есть столбец такого типа — contact\_data. Выполните модифи-  
кации таблиц и измените в них одну-две строки для проверки правильности  
ваших решений.

demo=# ALTER TABLE airports\_data ADD COLUMN approach\_data jsonb;

ALTER TABLE

demo=# update airports\_data set approach\_data = '{"runway\_count": 3,

demo'# "RNW-14C": {"course": 135, "length": 2370, "navigation\_method": "RNAV"},

demo'# "RNW-14L": {"course": 135, "length": 3800, "navigation\_method": "RNAV"},

demo'# "RNW-14R": {"course": 135, "length": 3500, "navigation\_method": "ILS CAT 3A (110.100)"}

demo'# }'::jsonb where airport\_code = 'DME';

UPDATE 1

Проверяем добавленные данные:  
demo=# select approach\_data from airports\_data where airport\_code = 'DME';  
  
 approach\_data

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

{"RNW-14C": {"course": 135, "length": 2370, "navigation\_method": "RNAV"}, "RNW-14L": {"course": 135, "length": 3800, "navigation\_method": "RNAV"}, "RNW-14R": {"course": 135, "length": 3500, "navigation\_method": "ILS CAT 3A (110.100)"}, "runway\_count": 3}

(1 row)

Получаем данные по конкретной ВПП:

demo=# select approach\_data->'RNW-14C' from airports\_data where airport\_code = 'DME';

?column?

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

{"course": 135, "length": 2370, "navigation\_method": "RNAV"}

(1 row)

ДЗ№5

2) Предложите шаблон поиска в операторе LIKE для выбора из этой таблицы всех  
пассажиров с фамилиями, состоящими из пяти букв.

demo=# SELECT count(passenger\_name)

FROM tickets

WHERE passenger\_name LIKE '% \_\_\_\_\_';

count

-------

14272

(1 row)

7)

demo=# select \* from test;

demo=# create view cities\_boeing\_cover as

SELECT DISTINCT departure\_city, arrival\_city

FROM routes r

JOIN aircrafts a ON r.aircraft\_code = a.aircraft\_code

WHERE a.model = 'Боинг 777-300'

ORDER BY 1;

CREATE VIEW

demo=# select r.arrival\_city, r.departure\_city from cities\_boeing\_cover r join cities\_boeing\_cover rr on r.arrival\_city = rr.departure\_city and r.departure\_city = rr.arrival\_city and r.arrival\_city > rr.arrival\_city;

arrival\_city | departure\_city

--------------+----------------

Москва | Екатеринбург

Новосибирск | Москва

Пермь | Москва

Сочи | Москва

(4 rows)

9)

#### Упражнение 9

Для ответа на вопрос, сколько рейсов выполняется из Москвы в Санкт-Петербург, можно написать совсем простой запрос:

SELECT count( \* )

FROM routes

WHERE departure\_city = 'Москва'

AND arrival\_city = 'Санкт-Петербург'

demo=# SELECT departure\_city, arrival\_city, count( \* )

FROM routes

WHERE departure\_city = 'Москва'

AND arrival\_city = 'Санкт-Петербург' group by departure\_city, arrival\_city;

departure\_city | arrival\_city | count

----------------+-----------------+-------

Москва | Санкт-Петербург | 12

(1 row)

13)

SELECT f.departure\_city, f.arrival\_city,  
max( tf.amount ), min( tf.amount )  
FROM flights\_v f  
JOIN ticket\_flights tf ON f.flight\_id = tf.flight\_id  
GROUP BY 1, 2  
ORDER BY 1, 2;

Модифицируйте запрос, приведенный выше.

SELECT f.departure\_city, f.arrival\_city,  
max( tf.amount ), min( tf.amount )  
FROM flights\_v f  
LEFT JOIN ticket\_flights tf ON f.flight\_id = tf.flight\_id  
GROUP BY 1, 2  
ORDER BY 1, 2;

departure\_city | arrival\_city | max | min

--------------------------+--------------------------+-----------+----------

Абакан | Архангельск | |

Абакан | Грозный | |

Абакан | Кызыл | |

Абакан | Москва | 101000.00 | 33700.00

Абакан | Новосибирск | 5800.00 | 5800.00

Абакан | Томск | 4900.00 | 4900.00

Анадырь | Москва | 185300.00 | 61800.00

Анадырь | Хабаровск | 92200.00 | 30700.00

Анапа | Белгород | 18900.00 | 6300.00

Анапа | Москва | 36600.00 | 12200.00

Анапа | Новокузнецк | |

Архангельск | Абакан | |

Архангельск | Иркутск | |

Архангельск | Москва | 11100.00 | 10100.00

19)

izhodniy variant

WITH RECURSIVE ranges ( min\_sum, max\_sum )  
AS (  
VALUES( 0, 100000 ),  
( 100000, 200000 ),  
( 200000, 300000 )  
UNION ALL  
SELECT min\_sum + 100000, max\_sum + 100000  
FROM ranges  
WHERE max\_sum < ( SELECT max( total\_amount ) FROM bookings )  
)  
SELECT \* FROM ranges;

zadanie 1

demo=# WITH RECURSIVE ranges ( min\_sum, max\_sum, iter )

AS (

VALUES( 0, 100000, 1 ),

( 100000, 200000, 1 ),

( 200000, 300000, 1 )

UNION ALL

SELECT min\_sum + 100000, max\_sum + 100000, iter + 1

FROM ranges

WHERE max\_sum < ( SELECT max( total\_amount ) FROM bookings )

)

SELECT \* FROM ranges;

min\_sum | max\_sum | iter

---------+---------+------

0 | 100000 | 1

100000 | 200000 | 1

200000 | 300000 | 1

100000 | 200000 | 2

200000 | 300000 | 2

300000 | 400000 | 2

200000 | 300000 | 3

300000 | 400000 | 3

400000 | 500000 | 3

300000 | 400000 | 4

400000 | 500000 | 4

500000 | 600000 | 4

400000 | 500000 | 5

500000 | 600000 | 5

600000 | 700000 | 5

500000 | 600000 | 6

600000 | 700000 | 6

700000 | 800000 | 6

600000 | 700000 | 7

700000 | 800000 | 7

800000 | 900000 | 7

700000 | 800000 | 8

800000 | 900000 | 8

900000 | 1000000 | 8

800000 | 900000 | 9

900000 | 1000000 | 9

1000000 | 1100000 | 9

900000 | 1000000 | 10

1000000 | 1100000 | 10

1100000 | 1200000 | 10

1000000 | 1100000 | 11

1100000 | 1200000 | 11

1200000 | 1300000 | 11

1100000 | 1200000 | 12

1200000 | 1300000 | 12

1200000 | 1300000 | 13

(36 rows)

zadanie 2:

demo=# WITH RECURSIVE ranges ( min\_sum, max\_sum)

AS (

VALUES( 0, 100000),

( 100000, 200000),

( 200000, 300000)

UNION

SELECT min\_sum + 100000, max\_sum + 100000

FROM ranges

WHERE max\_sum < ( SELECT max( total\_amount ) FROM bookings )

)

SELECT \* FROM ranges;

min\_sum | max\_sum

---------+---------

0 | 100000

100000 | 200000

200000 | 300000

300000 | 400000

400000 | 500000

500000 | 600000

600000 | 700000

700000 | 800000

800000 | 900000

900000 | 1000000

1000000 | 1100000

1100000 | 1200000

1200000 | 1300000

(13 rows)

21)Вместо знака «?» поставьте в приведенном ниже запросе нужное ключевое сло-  
во — UNION, INTERSECT или EXCEPT — и обоснуйте ваше решение.

Первый запрос формирует множество городов в которых есть аэропорты исключая Москву. Второй запрос формирует множество маршрутов, где город вылета не Москва. В данной ситуации подходит опреация вычитания одного множества из другого.

demo=# SELECT city

FROM airports

WHERE city <> 'Москва'

except

SELECT arrival\_city

FROM routes

WHERE departure\_city = 'Москва'

ORDER BY city;

city

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

Благовещенск

Иваново

Иркутск

Калуга

Когалым

Комсомольск-на-Амуре

Кызыл

Магадан

Нижнекамск

Новокузнецк

Стрежевой

Сургут

Удачный

Усть-Илимск

Усть-Кут

Ухта

Череповец

Чита

Якутск

Ярославль

(20 rows)

23)

Предположим, что департамент развития нашей авиакомпании задался вопро-  
сом: каким будет общее число различных маршрутов, которые теоретически  
можно проложить между всеми городами?  
Если в каком-то городе имеется более одного аэропорта, то это учитывать не  
будем, т. е. маршрутом будем считать путь между городами, а не между аэро-  
портами. Здесь мы используем соединение таблицы с самой собой на основе  
неравенства значений атрибутов.  
SELECT count( \* )  
FROM ( SELECT DISTINCT city FROM airports ) AS a1  
JOIN ( SELECT DISTINCT city FROM airports ) AS a2  
ON a1.city <> a2.city;

count  
-------  
10100  
(1 строка)  
Задание. Перепишите этот запрос с общим табличным выражением.

demo=# with cities as (SELECT DISTINCT city FROM airports)

select count(\*)

from cities c1 join cities c2 on c1.city <> c2.city;

count

-------

10100

(1 row)

ДЗ№6

1) Добавьте в определение таблицы aircrafts\_log значение по умолчанию  
current\_timestamp и соответствующим образом измените команды INSERT,  
приведенные в тексте главы.

demo=# CREATE TEMP TABLE aircrafts\_log AS

SELECT \* FROM aircrafts WITH DATA;

SELECT 9

demo=# ALTER TABLE aircrafts\_log

demo-# ADD COLUMN when\_add timestamp default Current\_timestamp;

ALTER TABLE

demo=# WITH add\_row AS

( INSERT INTO aircrafts\_tmp

SELECT \* FROM aircrafts

RETURNING \*

)

INSERT INTO aircrafts\_log

SELECT add\_row.aircraft\_code, add\_row.model, add\_row.range

FROM add\_row;

INSERT 0 9

2)В предложении RETURNING можно указывать не только символ «∗», означающий  
выбор всех столбцов таблицы, но и более сложные выражения, сформированные  
на основе этих столбцов. В тексте главы мы копировали содержимое таблицы  
«Самолеты» в таблицу aircrafts\_tmp, используя в предложении RETURNING  
именно «∗». Однако возможен и другой вариант запроса:  
WITH add\_row AS  
( INSERT INTO aircrafts\_tmp  
SELECT \* FROM aircrafts  
RETURNING aircraft\_code, model, range,  
current\_timestamp, 'INSERT'  
)  
INSERT INTO aircrafts\_log  
SELECT ? FROM add\_row;

Можно явно указать столбцы: add\_row.aircraft\_code, add\_row.model, add\_row.range, current\_timestamp, 'INSERT'

4)

В тексте главы в предложениях ON CONFLICT команды INSERT мы использова-  
ли только выражения, состоящие из имени одного столбца. Однако в таблице  
«Места» (seats) первичный ключ является составным и включает два столбца.  
Напишите команду INSERT для вставки новой строки в эту таблицу и преду-  
смотрите возможный конфликт добавляемой строки со строкой, уже имеющей-  
ся в таблице. Сделайте два варианта предложения ON CONFLICT: первый — с ис-  
пользованием перечисления имен столбцов для проверки наличия дублирова-  
ния, второй — с использованием предложения ON CONSTRAINT.  
Для того чтобы не изменить содержимое таблицы «Места», создайте ее копию  
и выполняйте все эти эксперименты с таблицей-копией.

demo=# create table seats2 as select \* from seats;

SELECT 1340

demo=# alter table seats2 add primary key (aircraft\_code, seat\_no);

ALTER TABLE

Variant 1:

demo=# insert into seats2 values ('319', '2C', 'Business')

on conflict (aircraft\_code, seat\_no) do nothing;

INSERT 0 0

Varint 2:

demo=# insert into seats2 values ('319', '2C', 'Business')

on conflict on constraint seats2\_pkey do nothing;

INSERT 0 0

ДЗ №7

1)

Предположим, что для какой-то таблицы создан уникальный индекс по двум  
столбцам: column1 и column2. В таблице есть строка, у которой значение ат-  
рибута column1 равно ABC, а значение атрибута column2 — NULL. Мы решили  
добавить в таблицу еще одну строку с такими же значениями ключевых атри-  
бутов, т. е. column1 — ABC, а column2 — NULL.  
Как вы думаете, будет ли операция вставки новой строки успешной или завер-  
шится с ошибкой? Объясните ваше решение.

Я считаю, что операция будет неудачной т. к. в уникальных индексах допускается наличие значений NULL, поскольку  
они считаются не совпадающими ни с какими другими значениями, в том числе и  
друг с другом. Однако начения первых столбцов — одинаковые.

3)Выполните запросы, подсчитывающие количество строк, в которых атрибут  
fare\_conditions принимает одно из трех возможных значений. Каждый из  
запросов выполните три-четыре раза, поскольку время может немного изме-  
няться, и подсчитайте среднее время.

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Comfort';

count

-------

17291

(1 row)

Time: 66,778 ms

среднее время — 68

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Business';

count

--------

107642

(1 row)

Time: 63,114 ms

среднее время — 71

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Economy';

count

--------

920793

(1 row)

Time: 80,450 ms

среднее время — 88

Создайте индекс по столбцу fare\_conditions.

demo=# create index on ticket\_flights (fare\_conditions);

CREATE INDEX

Time: 678,850 ms

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Comfort';

count

-------

17291

(1 row)

Time: 12,174 ms

среднее время — 16,5

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Business';

count

--------

107642

(1 row)

Time: 49,882 ms

среднее время — 62,5

demo=# SELECT count( \* )

FROM ticket\_flights

WHERE fare\_conditions = 'Economy';

count

--------

920793

(1 row)

Time: 94,142 ms

среднее время — 94,3333…

Как видно из результатов, там где результирующая выборка меньше, Удалось достигнуть значительного ускорения. В случае объёмной выборки, как получилось с местами эконом класса, ускорить запрос не получилось из-за большого кол-ва результирующих строк.

ДЗ №8

2)Задание. Модифицируйте сценарий выполнения транзакций: в первой тран-  
закции вместо фиксации изменений выполните их отмену с помощью команды  
ROLLBACK и посмотрите, будет ли удалена строка и какая конкретно.

В результате выполнения модифицированного сценария удалилась 1 строка, где range был < 2000 в исходной таблице т. к. все изменения которые вносились в первом терминале.

До изменений:

aircraft\_code | model | range

---------------+---------------------+-------

773 | Боинг 777-300 | 11100

763 | Боинг 767-300 | 7900

SU9 | Сухой Суперджет-100 | 3000

320 | Аэробус A320-200 | 5700

321 | Аэробус A321-200 | 5600

319 | Аэробус A319-100 | 6700

733 | Боинг 737-300 | 4200

CN1 | Сессна 208 Караван | 1200

CR2 | Бомбардье CRJ-200 | 2700

После изменений:

aircraft\_code | model | range

---------------+---------------------+-------

773 | Боинг 777-300 | 11100

763 | Боинг 767-300 | 7900

SU9 | Сухой Суперджет-100 | 3000

320 | Аэробус A320-200 | 5700

321 | Аэробус A321-200 | 5600

319 | Аэробус A319-100 | 6700

733 | Боинг 737-300 | 4200

CR2 | Бомбардье CRJ-200 | 2700

(8 rows)

3)Можно ли  
говорить, что в такой ситуации имеет место потерянное обновление? Если оно  
имеет место, то что можно предпринять для его недопущения? Обоснуйте ваш  
ответ.

В какой-то мере эта ситуация похожа на потерянное обновление, но с точки зрения СУБД сначала выполнится первый запрос, а затем второй затрёт изменения первого. Решить эту проблему можно с помощью более высокого уровня изоляции.

ДЗ № 9

Самостоятельно выполните команду EXPLAIN для запроса, содержащего общее  
табличное выражение (CTE). Посмотрите, на каком уровне находится узел пла-  
на, отвечающий за это выражение, как он оформляется. Учтите, что общие таб-  
личные выражения всегда материализуются, т. е. вычисляются однократно и  
результат их вычисления сохраняется в памяти, а затем все последующие об-  
ращения в рамках запроса направляются уже к этому материализованному ре-  
зультату.

Обратимся к запросу из предыидущих ДЗ:

emo=# EXPLAIN WITH city\_from AS

demo-# ( SELECT DISTINCT city FROM airports )

demo-# SELECT count( \* )

demo-# FROM city\_from f

demo-# JOIN ( SELECT DISTINCT city FROM airports ) AS a2

demo-# ON f.city <> a2.city;

QUERY PLAN

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

Aggregate (cost=249.15..249.16 rows=1 width=8)

-> Nested Loop (cost=67.57..223.90 rows=10100 width=0)

Join Filter: (((ml.city ->> lang())) <> ((ml\_1.city ->> lang())))

-> Unique (cost=33.78..34.30 rows=101 width=32)

-> Sort (cost=33.78..34.04 rows=104 width=32)

Sort Key: ((ml.city ->> lang()))

-> Seq Scan on airports\_data ml (cost=0.00..30.30 rows=104 width=32)

-> Materialize (cost=33.78..35.82 rows=101 width=32)

-> Unique (cost=33.78..34.30 rows=101 width=32)

-> Sort (cost=33.78..34.04 rows=104 width=32)

Sort Key: ((ml\_1.city ->> lang()))

-> Seq Scan on airports\_data ml\_1 (cost=0.00..30.30 rows=104 width=32)

(12 rows)

6)demo=# select aircraft\_code, model, range, avg(range) OVER() from aircrafts where model like 'Аэробус%';

aircraft\_code | model | range | avg

---------------+------------------+-------+-----------------------

320 | Аэробус A320-200 | 5700 | 6000.0000000000000000

321 | Аэробус A321-200 | 5600 | 6000.0000000000000000

319 | Аэробус A319-100 | 6700 | 6000.0000000000000000

(3 rows)

demo=# explain select aircraft\_code, model, range, avg(range) OVER() from aircrafts where model like 'Аэробус%';

QUERY PLAN

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

WindowAgg (cost=0.00..3.65 rows=1 width=84)

-> Seq Scan on aircrafts\_data ml (cost=0.00..3.39 rows=1 width=52)

Filter: ((model ->> lang()) ~~ 'Аэробус%'::text)

(3 rows)

Узел с WindowAgg т.е. с оконной функцией занимает именно такое место из-за того, что она формирует выборку для исполнения последующих действий над ней.

8)demo=# EXPLAIN ANALYZE

SELECT a.aircraft\_code AS a\_code,

a.model,

( SELECT count( r.aircraft\_code )

FROM routes r

WHERE r.aircraft\_code = a.aircraft\_code

) AS num\_routes

FROM aircrafts a

GROUP BY 1, 2

ORDER BY 3 DESC;

Time: 125,966 ms QUERY PLAN

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

Sort (cost=11757.58..11757.60 rows=9 width=56) (actual time=124.941..124.946 rows=9 loops=1)

Sort Key: ((SubPlan 1)) DESC

Sort Method: quicksort Memory: 25kB

-> HashAggregate (cost=3.41..11757.44 rows=9 width=56) (actual time=7.495..124.928 rows=9 loops=1)

Group Key: ml.aircraft\_code, (ml.model ->> lang())

-> Seq Scan on aircrafts\_data ml (cost=0.00..3.36 rows=9 width=48) (actual time=0.024..0.038 rows=9 loops=1)

SubPlan 1

-> Aggregate (cost=1305.73..1305.74 rows=1 width=8) (actual time=13.871..13.871 rows=1 loops=9)

-> Hash Join (cost=1106.81..1304.52 rows=97 width=240) (actual time=12.759..13.861 rows=79 loops=9)

Hash Cond: (flights.arrival\_airport = ml\_2.airport\_code)

-> Hash Join (cost=1101.47..1298.68 rows=187 width=8) (actual time=12.754..13.832 rows=79 loops=9)

Hash Cond: (flights.departure\_airport = ml\_1.airport\_code)

-> GroupAggregate (cost=1096.13..1288.78 rows=359 width=67) (actual time=12.748..13.804 rows=79 loops=9)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure))

-> Group (cost=1096.13..1204.46 rows=3588 width=39) (actual time=12.735..13.698 rows=422 loops=9)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure)), ((to\_char(flights.scheduled\_departure, 'ID'::text))::integer)

-> Sort (cost=1096.13..1106.48 rows=4140 width=39) (actual time=12.708..12.918 rows=3680 loops=9)

Sort Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, ((flights.scheduled\_arrival - flights.scheduled\_departure)), ((to\_char(flights.scheduled\_departure, 'ID'::text))::integer)

Sort Method: quicksort Memory: 1049kB

-> Seq Scan on flights (cost=0.00..847.41 rows=4140 width=39) (actual time=0.524..5.948 rows=3680 loops=9)

Filter: (aircraft\_code = ml.aircraft\_code)

Rows Removed by Filter: 29441

-> Hash (cost=4.04..4.04 rows=104 width=4) (actual time=0.024..0.024 rows=104 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 12kB

-> Seq Scan on airports\_data ml\_1 (cost=0.00..4.04 rows=104 width=4) (actual time=0.002..0.011 rows=104 loops=1)

-> Hash (cost=4.04..4.04 rows=104 width=4) (actual time=0.028..0.029 rows=104 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 12kB

-> Seq Scan on airports\_data ml\_2 (cost=0.00..4.04 rows=104 width=4) (actual time=0.004..0.015 rows=104 loops=1)

Planning Time: 0.369 ms

Execution Time: 125.087 ms

(30 rows)

среднее время выполнения: 139,73333…

QUERY PLAN

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

Sort (cost=2734.93..2734.96 rows=9 width=56) (actual time=45.681..45.684 rows=9 loops=1)

Sort Key: (count(flights.aircraft\_code)) DESC

Sort Method: quicksort Memory: 25kB

-> GroupAggregate (cost=2732.31..2734.79 rows=9 width=56) (actual time=45.541..45.679 rows=9 loops=1)

Group Key: ml.aircraft\_code, ((ml.model ->> lang()))

-> Sort (cost=2732.31..2732.34 rows=12 width=52) (actual time=45.528..45.565 rows=711 loops=1)

Sort Key: ml.aircraft\_code, ((ml.model ->> lang()))

Sort Method: quicksort Memory: 80kB

-> Hash Right Join (cost=2446.06..2732.09 rows=12 width=52) (actual time=42.307..45.267 rows=711 loops=1)

Hash Cond: (flights.aircraft\_code = ml.aircraft\_code)

-> Hash Join (cost=2444.86..2724.37 rows=276 width=240) (actual time=42.265..43.913 rows=710 loops=1)

Hash Cond: (flights.arrival\_airport = ml\_2.airport\_code)

-> Hash Join (cost=2439.52..2717.60 rows=530 width=8) (actual time=42.232..43.673 rows=710 loops=1)

Hash Cond: (flights.departure\_airport = ml\_1.airport\_code)

-> GroupAggregate (cost=2434.18..2699.33 rows=1020 width=67) (actual time=42.204..43.453 rows=710 loops=1)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure))

-> Sort (cost=2434.18..2459.67 rows=10198 width=39) (actual time=42.196..42.435 rows=3798 loops=1)

Sort Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure)), ((to\_char(flights.scheduled\_departure, 'ID'::text))::integer)

Sort Method: quicksort Memory: 393kB

-> HashAggregate (cost=1551.24..1755.20 rows=10198 width=39) (actual time=31.385..32.269 rows=3798 loops=1)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, (flights.scheduled\_arrival - flights.scheduled\_departure), (to\_char(flights.scheduled\_departure, 'ID'::text))::integer

-> Seq Scan on flights (cost=0.00..1054.42 rows=33121 width=39) (actual time=0.012..16.857 rows=33121 loops=1)

-> Hash (cost=4.04..4.04 rows=104 width=4) (actual time=0.023..0.024 rows=104 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 12kB

-> Seq Scan on airports\_data ml\_1 (cost=0.00..4.04 rows=104 width=4) (actual time=0.002..0.011 rows=104 loops=1)

-> Hash (cost=4.04..4.04 rows=104 width=4) (actual time=0.030..0.030 rows=104 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 12kB

-> Seq Scan on airports\_data ml\_2 (cost=0.00..4.04 rows=104 width=4) (actual time=0.004..0.017 rows=104 loops=1)

-> Hash (cost=1.09..1.09 rows=9 width=48) (actual time=0.013..0.014 rows=9 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on aircrafts\_data ml (cost=0.00..1.09 rows=9 width=48) (actual time=0.008..0.009 rows=9 loops=1)

Planning Time: 0.332 ms

Execution Time: 45.770 ms

(33 rows)

среднее время выполнения: 45,4896667…

select a.aircraft\_code, a.model, a.range, (select count (distinct r.aircraft\_code) from routes r where r.departure\_city = 'Москва') as count\_of\_models\_from\_moscow

from aircrafts a;

QUERY PLAN

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

Seq Scan on aircrafts\_data ml (cost=2753.78..2757.15 rows=9 width=60) (actual time=60.831..60.860 rows=9 loops=1)

InitPlan 1 (returns $1)

-> Aggregate (cost=2753.77..2753.78 rows=1 width=8) (actual time=60.766..60.769 rows=1 loops=1)

-> Nested Loop (cost=2434.32..2753.74 rows=3 width=240) (actual time=55.098..60.686 rows=154 loops=1)

-> Nested Loop (cost=2434.18..2752.84 rows=5 width=8) (actual time=55.063..60.186 rows=154 loops=1)

Join Filter: (flights.departure\_airport = ml\_1.airport\_code)

Rows Removed by Join Filter: 1976

-> Seq Scan on airports\_data ml\_1 (cost=0.00..30.56 rows=1 width=4) (actual time=0.089..0.298 rows=3 loops=1)

Filter: ((city ->> lang()) = 'Москва'::text)

Rows Removed by Filter: 101

-> GroupAggregate (cost=2434.18..2699.33 rows=1020 width=67) (actual time=18.316..19.897 rows=710 loops=3)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure))

-> Sort (cost=2434.18..2459.67 rows=10198 width=39) (actual time=18.310..18.624 rows=3798 loops=3)

Sort Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure)), ((to\_char(flights.scheduled\_departure, 'ID'::text))::integer)

Sort Method: quicksort Memory: 393kB

-> HashAggregate (cost=1551.24..1755.20 rows=10198 width=39) (actual time=43.092..43.928 rows=3798 loops=1)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, (flights.scheduled\_arrival - flights.scheduled\_departure), (to\_char(flights.scheduled\_departure, 'ID'::text))::integer

-> Seq Scan on flights (cost=0.00..1054.42 rows=33121 width=39) (actual time=0.019..23.140 rows=33121 loops=1)

-> Index Only Scan using airports\_data\_pkey on airports\_data ml\_2 (cost=0.14..0.18 rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=154)

Index Cond: (airport\_code = flights.arrival\_airport)

Heap Fetches: 154

Planning Time: 1.085 ms

Execution Time: 61.051 ms

(23 rows)

Time: 62,782 ms

среднее время выполнения 57

select a.aircraft\_code, a.model, a.range, count(r.aircraft\_code) over() as count\_of\_models\_from\_moscow

from aircrafts a LEFT OUTER JOIN routes r on r.aircraft\_code = a.aircraft\_code

where r.departure\_city = 'Москва' group by a.aircraft\_code, a.model, a.range, r.aircraft\_code;

WindowAgg (cost=2755.76..2757.34 rows=3 width=64) (actual time=66.016..66.020 rows=8 loops=1)

-> Group (cost=2755.76..2756.55 rows=3 width=56) (actual time=65.977..66.008 rows=8 loops=1)

Group Key: ml.aircraft\_code, ((ml.model ->> lang())), flights.aircraft\_code

-> Sort (cost=2755.76..2755.77 rows=3 width=56) (actual time=65.975..65.984 rows=154 loops=1)

Sort Key: ml.aircraft\_code, ((ml.model ->> lang()))

Sort Method: quicksort Memory: 37kB

-> Hash Join (cost=2435.52..2755.73 rows=3 width=56) (actual time=61.547..65.888 rows=154 loops=1)

Hash Cond: (flights.aircraft\_code = ml.aircraft\_code)

-> Nested Loop (cost=2434.32..2753.74 rows=3 width=240) (actual time=61.486..65.452 rows=154 loops=1)

-> Nested Loop (cost=2434.18..2752.84 rows=5 width=8) (actual time=61.467..65.063 rows=154 loops=1)

Join Filter: (flights.departure\_airport = ml\_1.airport\_code)

Rows Removed by Join Filter: 1976

-> Seq Scan on airports\_data ml\_1 (cost=0.00..30.56 rows=1 width=4) (actual time=0.167..0.291 rows=3 loops=1)

Filter: ((city ->> lang()) = 'Москва'::text)

Rows Removed by Filter: 101

-> GroupAggregate (cost=2434.18..2699.33 rows=1020 width=67) (actual time=20.426..21.537 rows=710 loops=3)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure))

-> Sort (cost=2434.18..2459.67 rows=10198 width=39) (actual time=20.422..20.630 rows=3798 loops=3)

Sort Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, ((flights.scheduled\_arrival - flights.scheduled\_departure)), ((to\_char(flights.scheduled\_departure, 'ID'::text))::integer)

Sort Method: quicksort Memory: 393kB

-> HashAggregate (cost=1551.24..1755.20 rows=10198 width=39) (actual time=50.687..51.475 rows=3798 loops=1)

Group Key: flights.flight\_no, flights.departure\_airport, flights.arrival\_airport, flights.aircraft\_code, (flights.scheduled\_arrival - flights.scheduled\_departure), (to\_char(flights.scheduled\_departure, 'ID'::text))::integer

-> Seq Scan on flights (cost=0.00..1054.42 rows=33121 width=39) (actual time=0.027..27.275 rows=33121 loops=1)

-> Index Only Scan using airports\_data\_pkey on airports\_data ml\_2 (cost=0.14..0.18 rows=1 width=4) (actual time=0.002..0.002 rows=1 loops=154)

Index Cond: (airport\_code = flights.arrival\_airport)

Heap Fetches: 154

-> Hash (cost=1.09..1.09 rows=9 width=52) (actual time=0.028..0.029 rows=9 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

-> Seq Scan on aircrafts\_data ml (cost=0.00..1.09 rows=9 width=52) (actual time=0.015..0.018 rows=9 loops=1)

Planning Time: 1.085 ms

Execution Time: 45.280 ms

(31 rows)

среднее время выполнения 48

Как видно из приведённых выше запросов объединение позволяет немного ускорить выполнение запроса относительно варианта с коррелированным подзапросом.