Мамонов Антон 3ИсиП-19-1

Массивы

Создадим новую таблицу pilots

```
test=# CREATE TABLE pilots
test-# (
test(# pilot_name text,
test(# schedule integer[]
test(# );
CREATE TABLE
```

```
test=# INSERT INTO pilots
test-# VALUES ( 'Ivan', '{ 1, 3, 5, 6, 7 }'::integer[] ),
test-# ( 'Petr', '{ 1, 2, 5, 7 }'::integer[] ),
test-# ( 'Pavel', '{ 2, 5 }'::integer[] ),
test-# ( 'Boris', '{ 3, 5, 6 }'::integer[] );
INSERT 0 4
```

Заполним ее данными с помощью массивов

Посмотрим, что к чему там. Выглядит здорово

```
test=# SELECT * FROM pilots;
 pilot_name |  schedule
------------+-------------
 Ivan       | {1,3,5,6,7}
 Petr       | {1,2,5,7}
 Pavel      | {2,5}
 Boris      | {3,5,6,7}
(4 строки)
```

Самую малость update(ов)

```
test=# UPDATE pilots
test-# SET schedule = schedule || 7
test-# WHERE pilot_name = 'Boris';
UPDATE 1
```

```
test=# UPDATE pilots
test-# SET schedule = array_append( schedule, 6 )
test-# WHERE pilot_name = 'Pavel';
UPDATE 1
```

```
test=# UPDATE pilots
test-# SET schedule = array_prepend( 1, schedule )
test-# WHERE pilot_name = 'Pavel';
UPDATE 1
```

```
test=# UPDATE pilots
test-# SET schedule = array_remove( schedule, 5 )
test-# WHERE pilot_name = 'Ivan';
UPDATE 1
```

```
test=# UPDATE pilots
test-# SET schedule[ 1 ] = 2, schedule[ 2 ] = 3
test-# WHERE pilot_name = 'Petr';
UPDATE 1
```

```
test=# UPDATE pilots
test-# SET schedule[ 1:2 ] = ARRAY[ 2, 3 ]
test-# WHERE pilot_name = 'Petr';
UPDATE 1
```

Проверим что к чему, все поменялось, отлично

```
test=# SELECT * FROM pilots;
 pilot_name |  schedule
------------+-------------
 Boris      | {3,5,6,7,7}
 Pavel      | {1,2,5,6}
 Ivan       | {1,3,6,7}
 Petr       | {2,3,5,7}
(4 строки)
```

```
test=# SELECT * FROM pilots
test-# WHERE array_position( schedule, 3 ) IS NOT NULL;
 pilot_name |  schedule
------------+-------------
 Boris      | {3,5,6,7,7}
 Ivan       | {1,3,6,7}
 Petr       | {2,3,5,7}
(3 строки)
```

Как видим, нулевой элемент массива нам не показало

Теперь те, у кого есть 1 и 7

```
test=# SELECT * FROM pilots
test-# WHERE schedule @> '{ 1, 7 }'::integer[];
 pilot_name | schedule
------------+-----------
 Ivan       | {1,3,6,7}
(1 строка)
```

```
test=# SELECT * FROM pilots
test-# WHERE schedule && ARRAY[ 2, 5 ];
 pilot_name |  schedule
------------+-------------
 Boris      | {3,5,6,7,7}
 Pavel      | {1,2,5,6}
 Petr       | {2,3,5,7}
(3 строки)
```

Те, у кого есть 2 и/или 5

Теперь те, у кого нет 2 и/или 5

```
test=# SELECT * FROM pilots
test-# WHERE NOT ( schedule && ARRAY[ 2, 5 ] );
 pilot_name | schedule
------------+-----------
 Ivan       | {1,3,6,7}
(1 строка)
```

```
test=# SELECT unnest( schedule ) AS days_of_week
test-# FROM pilots
test-# WHERE pilot_name = 'Ivan';
 days_of_week
--------------
            1
            3
            6
            7
(4 строки)
```

Дни у Ивана

JSON

Для тестов создадим новую таблицу

```
test=# CREATE TABLE pilot_hobbies
test-# (
test(# pilot_name text,
test(# hobbies jsonb
test(# );
CREATE TABLE
```

И заполним ее значениями в формате JSON

```
test=# INSERT INTO pilot_hobbies
test-# VALUES ( 'Ivan',
test(# '{ "sports": [ "футбол", "плавание" ],
test'# "home_lib": true, "trips": 3
test'# }'::jsonb
test(# ),
test-# ( 'Petr',
test(# '{ "sports": [ "теннис", "плавание" ],
test'# "home_lib": true, "trips": 2
test'# }'::jsonb
test(# ),
test-# ( 'Pavel',
test(# '{ "sports": [ "плавание" ],
test'# "home_lib": false, "trips": 4
test'# }'::jsonb
test(# ),
test-# ( 'Boris',
test(# '{ "sports": [ "футбол", "плавание", "теннис" ],
test'# "home_lib": true, "trips": 0
test'# }'::jsonb
test(# );
INSERT 0 4
```

Выглядит интересно

```
test=# SELECT * FROM pilot_hobbies;
 pilot_name |                                  hobbies
------------+-------------------------------------------------------------------------------
 Ivan       | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
 Petr       | {"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
 Pavel      | {"trips": 4, "sports": ["плавание"], "home_lib": false}
 Boris      | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
(4 строки)
```

Выбираем футболистов так

```
test=# SELECT * FROM pilot_hobbies
test-# WHERE hobbies @> '{ "sports": [ "футбол" ] }'::jsonb;
 pilot_name |                                  hobbies
------------+-------------------------------------------------------------------------------
 Ivan       | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
 Boris      | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
(2 строки)
```

Либо же вот так

```
test=# SELECT pilot_name, hobbies->'sports' AS sports
test-# FROM pilot_hobbies
test-# WHERE hobbies->'sports' @> '[ "футбол" ]'::jsonb;
 pilot_name |              sports
------------+----------------------------------
 Ivan       | ["футбол", "плавание"]
 Boris      | ["футбол", "плавание", "теннис"]
(2 строки)
```

```
test=# SELECT count( * )
test-# FROM pilot_hobbies
test-# WHERE hobbies ? 'sport';
 count
-------
     0
(1 строка)
```

Проверка на ключ, ключа sport у нас нет

А вот sports есть, работает!

```
test=# SELECT count( * )
test-# FROM pilot_hobbies
test-# WHERE hobbies ? 'sports';
 count
-------
     4
(1 строка)
```

Пажилой update

```
test=# UPDATE pilot_hobbies
test-# SET hobbies = hobbies || '{ "sports": [ "хоккей" ] }'
test-# WHERE pilot_name = 'Boris';
UPDATE 1
```

Выглядит это вот так

```
test=# SELECT pilot_name, hobbies
test-# FROM pilot_hobbies
test-# WHERE pilot_name = 'Boris';
 pilot_name |                         hobbies
------------+-----------------------------------------------------
 Boris      | {"trips": 0, "sports": ["хоккей"], "home_lib": true}
(1 строка)
```

Делаем из Бориса спортсмена

```
test=# UPDATE pilot_hobbies
test-# SET hobbies = jsonb_set( hobbies, '{ sports, 1 }', '"футбол"' )
test-# WHERE pilot_name = 'Boris';
UPDATE 1
```

И правда спортсмен

```
test=# SELECT pilot_name, hobbies
test-# FROM pilot_hobbies
test-# WHERE pilot_name = 'Boris';
 pilot_name |                          hobbies
------------+-----------------------------------------------------------
 Boris      | {"trips": 0, "sports": ["хоккей", "футбол"], "home_lib": true}
(1 строка)
```

Контрольные вопросы и задания

Задание 1

Создаем новую табличку

```
test=# CREATE TABLE test_numeric
test-# ( measurement numeric(5, 2),
test(# description text
test(# );
CREATE TABLE
```

При попытке ввода данных получаем следующие ошибки

```
test=# INSERT INTO test_numeric
test-# VALUES ( 999.9999, 'Какое-то измерение ' );
ОШИБКА:  переполнение поля numeric
ПОДРОБНОСТИ:  Поле с точностью 5, порядком 2 должно округляться до абсолютного значения меньше чем 10^3.
test=# INSERT INTO test_numeric
test-# VALUES ( 999.9009, 'Еще одно измерение' );
INSERT 0 1
test=# INSERT INTO test_numeric
test-# VALUES ( 999.1111, 'И еще измерение' );
INSERT 0 1
test=# INSERT INTO test_numeric
test-# VALUES ( 998.9999, 'И еще одно' );
INSERT 0 1
```

Задание 2

Удаляем старую таблицу

```
test=# DROP TABLE test_numeric;
DROP TABLE
```

Создаем новую чуть иначе

```
test=# CREATE TABLE test_numeric
test-# ( measurement numeric,
test(# description text
test(# );
CREATE TABLE
```

Заполняем данными

```
test=# INSERT INTO test_numeric
test-# VALUES ( 1234567890.0987654321,
test(# 'Точность 20 знаков, масштаб 10 знаков' );
INSERT 0 1
test=# INSERT INTO test_numeric
test-# VALUES ( 1.5,
test(# 'Точность 2 знака, масштаб 1 знак' );
INSERT 0 1
test=# INSERT INTO test_numeric
test-# VALUES ( 0.12345678901234567890,
test(# 'Точность 21 знак, масштаб 20 знаков' );
INSERT 0 1
test=# INSERT INTO test_numeric
test-# VALUES ( 1234567890,
test(# 'Точность 10 знаков, масштаб 0 знаков (целое число)' );
INSERT 0 1
```

Смотрим данные

```
test=# select * from test_numeric
test-# ;
      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)
```

Задание 3

```
test=# SELECT 'NaN'::numeric > 10000;
 ?column?
----------
 t
(1 строка)
```

Задание 4

Почему так?

```
test=# SELECT '5e-324'::double precision > '4e-324'::double precision;
 ?column?
----------
 f
(1 строка)
```

А числа то одинаковые, получается

```
test=# SELECT '5e-324'::double precision;
 float8
--------
 5e-324
(1 строка)
```

```
test=# SELECT '4e-324'::double precision;
 float8
--------
 5e-324
(1 строка)
```

Самостоятельно

```
test=# SELECT '3e-324'::double precision='4e-324'::double precision;
 ?column?
----------
 t
(1 строка)
```

Задание 5

```
test=# SELECT 'Inf'::double precision > 1E+308;
 ?column?
----------
 t
(1 строка)
```

Самостоятельно

```
test=# SELECT '-Inf'::double precision < '3e-324' ::double precision;
 ?column?
----------
 t
(1 строка)
```

Задание 6

NaN

```
test=# SELECT 0.0 * 'Inf'::real;
 ?column?
----------
      NaN
(1 строка)
```

TRUE NaN

```
test=# select 'NaN'::real > 'Inf'::real
 ?column?
----------
 t
(1 строка)
```

Задание 7

```
test=# CREATE TABLE test_serial
test-# ( id serial,
test(# name text
test(# );
CREATE TABLE
test=# INSERT INTO test_serial ( name ) VALUES ( 'Вишневая' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Зеленая' );
INSERT 0 1
test=# INSERT INTO test_serial ( id, name ) VALUES ( 10, 'Прохладная' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Луговая' );
INSERT 0 1
```

Задание 8

```
test=# CREATE TABLE test_serial
test-# ( id serial PRIMARY KEY,
test(# name text
test(# );
CREATE TABLE
test=# INSERT INTO test_serial ( name ) VALUES ( 'Вишневая' );
INSERT 0 1
test=# INSERT INTO test_serial ( id, name ) VALUES ( 2, 'Прохладная' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "test_serial_pkey"
ПОДРОБНОСТИ:  Ключ "(id)=(2)" уже существует.
test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
INSERT 0 1
```

Ошибка, потому что запись с таким id уже есть

Четверки нет ((((((((((

```
test=# INSERT INTO test_serial ( name ) VALUES ( 'Зеленая' );
INSERT 0 1
test=# DELETE FROM test_serial WHERE id = 4;
DELETE 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Луговая' );
INSERT 0 1
test=# SELECT * FROM test_serial;
 id |    name
----+------------
  1 | Вишневая
  2 | Прохладная
  3 | Грушевая
  5 | Луговая
(4 строки)
```

Задание 9

Григорианский

Задание 10

Из-за ограничений по памяти. Эти данные хранятся в восьми байтах.

Задание 11

```
test=# SELECT current_time;
    current_time
--------------------
 19:01:43.854231+03
(1 строка)

test=# SELECT current_time::time( 0 );
 current_time
--------------
 19:01:47
(1 строка)

test=# SELECT current_time::time( 3 );
 current_time
--------------
 19:01:51.575
(1 строка)

test=# SELECT '2004-10-19 10:23:54.421214'::timestamp(2);
       timestamp
------------------------
 2004-10-19 10:23:54.42
(1 строка)

test=# SELECT '2004-10-19 10:23:54.421214'::timestamp(0);
      timestamp
---------------------
 2004-10-19 10:23:54
(1 строка)

test=# SELECT 'P0001-02-03T04:05:06.321321'::interval(1);
            interval
---------------------------------
 1 year 2 mons 3 days 04:05:06.3
(1 строка)
```

Ответ: Потому что дни целочисленный формат.

Задание 12

```
test=# SET datestyle TO
test-# 'German, DMY';
SET
test=# SELECT
test-# '18-05-2016'
test-# ::timestamp;
      timestamp
--------------------
 18.05.2016 00:00:00
(1 строка)
```

Задание 13-14 is sadge

Задание 15

```
test=# SELECT to_char( current_timestamp, 'hh:mi:ss dd-mm-yyyy');
       to_char
---------------------
 07:13:41 26-10-2021
(1 строка)
```

Задание 16

```
test=# SELECT 'Feb 29, 2015'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "Feb 29, 2015"
СТРОКА 1: SELECT 'Feb 29, 2015'::date;
```

Задание 17

```
test=# SELECT '21:15:16:22'::time;
ОШИБКА:  неверный синтаксис для типа time: "21:15:16:22"
СТРОКА 1: SELECT '21:15:16:22'::time;
```

Задание 18

15!

```
test=# SELECT ( '2016-09-16'::date - '2016-09-01'::date );
 ?column?
----------
       15
(1 строка)
```

Задание 19

Если вместо «-« написать «+», то пишет ошибку. Так как мы плюсуем два времени суток, то ничего и не получается. Вместо другого времени нужно брать тип интервала.

Задание 20

Будет выведена текущая дата + 1 месяц.
Если убрать псевдоним, то в СУБД PostgreSQL будет выведено «?column?»

Задание 21

Он выводит последний день месяца. В случае февраля это 29 число.

```
test=# SELECT ('2016-01-31'::date +'1 mon'::interval ) AS new_date;
         new_date
--------------------------
 Mon 29 Feb 00:00:00 2016
(1 строка)
```

Он выводит тоже 29 число, здесь это получается из-за того, что здесь он уже прибавляет полноценный месяц.

```
test=# SELECT ('2016-02-29'::date +'1 mon'::interval ) AS new_date;
         new_date
--------------------------
 Tue 29 Mar 00:00:00 2016
(1 строка)
```

Задание 22

```
test=# SET intervalstyle TO 'postgres_verbose';
SET
test=# SELECT ('1 mon '::interval) AS new_date;
 new_date
----------
 @ 1 mon
(1 строка)
```

Задание 23

```
test=# SELECT ('2016-09-16'::date -'2015-09-01'::date );SELECT ('2016-09-16'::timestamp -'2015-09-01'::timestamp);
 ?column?
----------
      381
(1 строка)


  ?column?
------------
 @ 381 days
(1 строка)
```

date и timestamp почти одно и тоже. Поэтому результаты одни и те же.

Задание 24

Чтобы работало делаем так:

```
test=# SELECT ('20:34:35'::time - '1 hour'::interval);
 ?column?
----------
 19:34:35
(1 строка)
```

```
test=# SELECT ('2016-09-16'::date - 1 );
  ?column?
------------
 15-09-2016
(1 строка)
```

Из времени нельзя вычесть 1

Задание 25

```
SELECT ( date_trunc( 'mon',
timestamp '1999-11-27 12:34:56.987654' ) );
```

Результат | План выполн... | Сообщения | Notifications

	date_trunc timestamp without time zone
1	1999-11-01 00:00:00

Задание 26

```
SELECT ( date_trunc( 'min',
interval '1 hour 36 seconds' ) );
```

Результат | План выполн... | Сообщения | Not

	date_trunc interval
1	01:00:00

Задание 27

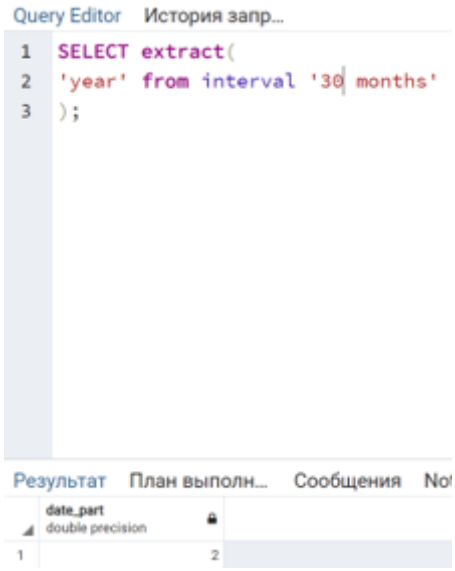
```
SELECT extract(
'year' from timestamp '1999-11-27 12:34:56.123459'
);
```

Результат | План выполн... | Сообщения | Notifications

	date_part double precision
1	1999

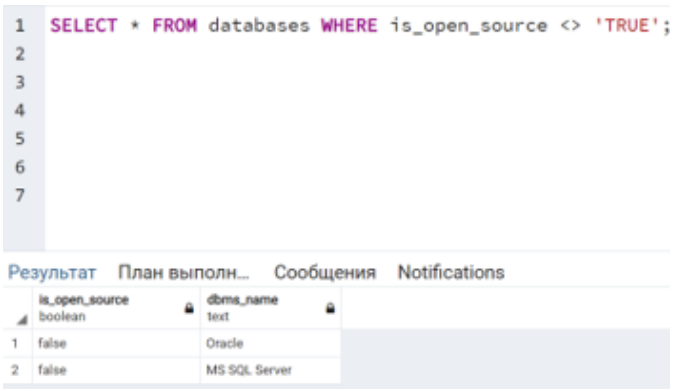
✔ Запрос

Задание 28



Задание 29

Равнозначны. Т.к. непустая строчка равна ИСТИНЕ.

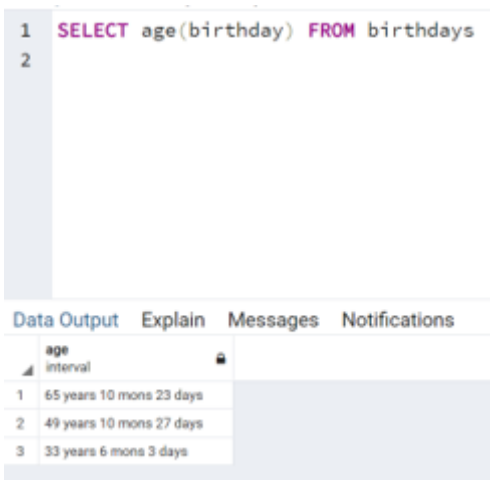


А последний запрос не работает, ибо почему-то нельзя сравнивать boolean и int.

Задание 30

Не работает 2, 6 и 8 запрос, ибо для str нужно добавлять апострофы, а int нельзя без явного преобразования переместить в boolean.

Задание 31



Задание 32

```
SELECT array_length(array[1,2,3], 1)
```

Data Output | Explain | Messages | Notifications

	array_length integer
1	3

Задание 33

```
SELECT meal[1][1] FROM pilots1
```

Data Output | Explain | Messages | Notificatic

	meal text
1	сосиска
2	сосиска
3	сосиска
4	сосиска

Задание 34

UPDATE pilot_hobbies
SET hobbies = jsonb_set(hobbies, '{ home_lib }', 'false')
WHERE pilot_name = 'Pavel';

```
SELECT pilot_name, hobbies->'home_lib' FROM pilot_hobbies;
```

Data Output | Explain | Messages | Notifications

	pilot_name text	?column? jsonb
1	Ivan	true
2	Petr	true
3	Boris	true
4	Pavel	false

Задание 35

```
SELECT '{ "sports": "хоккей"}'::jsonb || '{"trips": 5}'::jsonb->'trips'
```

Data Output | Explain | Messages | Notifications

	?column? jsonb
1	5

Задание 36

UPDATE pilot_hobbies
SET hobbies= hobbies || '{"chips": 5}'::jsonb
WHERE pilot_name='Pavel'

```
SELECT * FROM pilot_hobbies
```

Data Output | Explain | Messages | Notifications

	pilot_name text	hobbies jsonb
1	Ivan	{"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
2	Petr	{"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
3	Boris	{"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
4	Pavel	{"chips": 5, "trips": 4, "sports": ["плавание"], "home_lib": false}

Задание 37

UPDATE pilot_hobbies
SET hobbies= hobbies - 'chips'
WHERE pilot_name='Pavel'

```
SELECT * FROM pilot_hobbies
```

Data Output | Explain | Messages | Notifications

	pilot_name text	hobbies jsonb
1	Ivan	{"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
2	Petr	{"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
3	Boris	{"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
4	Pavel	{"trips": 4, "sports": ["плавание"], "home_lib": false}