МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №3**

**по курсу «Проектирование баз данных»**

Выполнила: Прудникова А. А.

Группа: М8О-114СВ-24

Преподаватель: Моргунов Е. П.

Москва, 2024

## Задание 2

Предположим, что возникла необходимость хранить в одном столбце таблицы данные, представленные с различной точностью. Это могут быть, например, результаты физических измерений разнородных показателей или различные медицинские показатели здоровья пациентов (результаты анализов). В таком случае можно использовать тип numeric без указания масштаба и точности. Теперь сделайте выборку из таблицы и посмотрите, что все эти разнообразные значения сохранены именно в том виде, как вы их вводили.

### Запрос

CREATE TABLE test\_numeric

( measurement numeric,

description text

);

INSERT INTO test\_numeric

VALUES ( 1234567890.0987654321,

'Точность 20 знаков, масштаб 10 знаков' );

INSERT INTO test\_numeric

VALUES ( 1.5,

'Точность 2 знака, масштаб 1 знак' );

INSERT INTO test\_numeric

VALUES ( 0.12345678901234567890,

'Точность 21 знак, масштаб 20 знаков' );

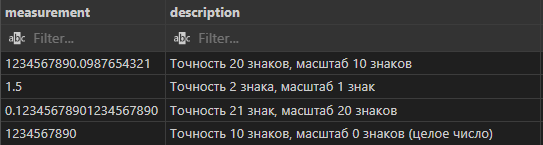
INSERT INTO test\_numeric

VALUES ( 1234567890,

'Точность 10 знаков, масштаб 0 знаков (целое число)' );

SELECT \* FROM test\_numeric;

### Результат



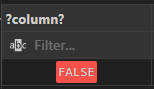
## Задание 4

При работе с числами типов real и double precision нужно помнить, что сравнение двух чисел с плавающей точкой на предмет равенства их значений может привести к неожиданным результатам.

### Запрос

SELECT '5e-324'::double precision > '4e-324'::double precision;

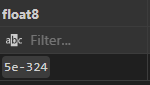
### Результат



### Запрос

SELECT '5e-324'::double precision;

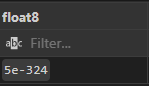
### Результат



### Запрос

SELECT '4e-324'::double precision;

### Результат



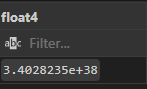
### Задание

Самостоятельно проведите аналогичные эксперименты с очень большими числами, находящимися на границе допустимого диапазона для чисел типов real и double precision.

### Запрос

SELECT '3.4028235e+38'::real;

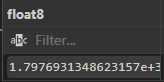
### Результат



### Запрос

SELECT '1.7976931348623157e+308'::double precision;

### Результат

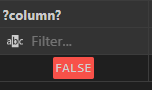
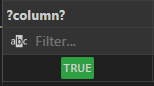


### Запрос

SELECT '3.4028235e+38'::real > '3.4028234e+38'::real;

SELECT '1.7976931348623157e+308'::double precision > '1.7976931348623156e+308'::double precision;

### Результат

## Задание 8

Пусть теперь столбец id будет первичным ключом таблицы.

CREATE TABLE test\_serial

( id serial PRIMARY KEY,

name text

);

Теперь выполните следующие команды для добавления строк в таблицу и удаления одной строки из нее. Для пошагового управления этим процессом выполняйте выборку данных из таблицы с помощью команды SELECT после каждой команды вставки или удаления.

INSERT INTO test\_serial ( name ) VALUES ( 'Вишневая' );

Явно зададим значение столбца id:

INSERT INTO test\_serial ( id, name ) VALUES ( 2, 'Прохладная' );

При выполнении этой команды СУБД выдаст сообщение об ошибке. Почему?

INSERT INTO test\_serial ( name ) VALUES ( 'Грушевая' );

Повторим эту же команду. Теперь все в порядке. Почему?

INSERT INTO test\_serial ( name ) VALUES ( 'Грушевая' );

Добавим еще одну строку.

INSERT INTO test\_serial ( name ) VALUES ( 'Зеленая' );

А теперь удалим ее же.

DELETE FROM test\_serial WHERE id = 4;

Добавим последнюю строку.

INSERT INTO test\_serial ( name ) VALUES ( 'Луговая' );

Теперь сделаем выборку.

SELECT \* FROM test\_serial;

Вы увидите, что в нумерации образовалась «дыра». Это из-за того, что при формировании нового значения из последовательности поиск максимального значения, уже имеющегося в столбце, не выполняется.

### Запрос

CREATE TABLE test\_serial

( id serial PRIMARY KEY,

name text

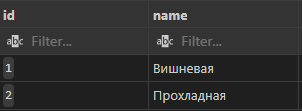
);

INSERT INTO test\_serial ( name ) VALUES ( 'Вишневая' );

INSERT INTO test\_serial ( id, name ) VALUES ( 2, 'Прохладная' );

SELECT \* FROM test\_serial;

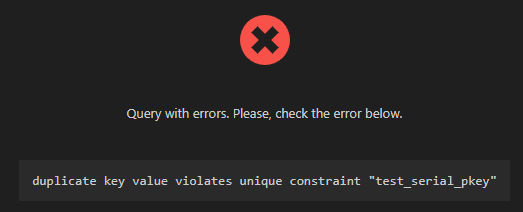
### Результат



### Запрос

INSERT INTO test\_serial ( name ) VALUES ( 'Грушевая' );

### Результат



### Запрос

INSERT INTO test\_serial ( name ) VALUES ( 'Грушевая' );

### Результат



### Решение

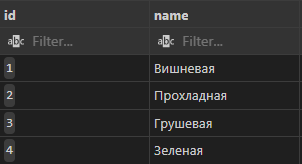
В первом случае при вставке появилось сообщение об ошибке, так как не был указан ID для вставляемой строки, поэтому был использован ID = 2, не являющийся уникальным. Однако при обработке запроса произошел сдвиг, поэтому при повторной попытке вставки был сгенерирован уникальный ID (ID = 3).

### Запрос

INSERT INTO test\_serial ( name ) VALUES ( 'Зеленая' );

SELECT \* FROM test\_serial;

### Результат



### Запрос

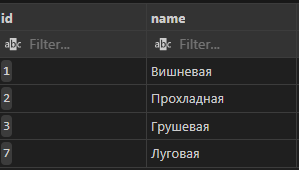
INSERT INTO test\_serial ( name ) VALUES ( 'Зеленая' );

DELETE FROM test\_serial WHERE id = 4;

INSERT INTO test\_serial ( name ) VALUES ( 'Луговая' );

SELECT \* FROM test\_serial;

### Результат



## Задание 12

Формат ввода и вывода даты можно изменить с помощью конфигурационного параметра datestyle. Значение этого параметра состоит из двух компонентов: первый управляет форматом вывода даты, а второй регулирует порядок следования составных частей даты (год, месяц, день) при вводе и выводе. Текущее значение этого параметра можно узнать с помощью команды SHOW:

SHOW datestyle;

Поскольку параметр datestyle состоит фактически из двух частей, которые можно задавать не только обе сразу, но и по отдельности, изменим только порядок следования составных частей даты, не изменяя формат вывода с ISO на какой-либо другой.

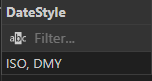
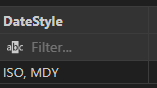
### Запрос

SHOW datestyle;

SET datestyle TO 'MDY';

SHOW datestyle;

### Результат

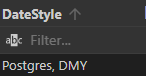
 

### Запрос

SET datestyle TO 'Postgres, DMY';

SHOW datestyle;

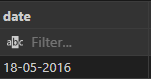
### Результат



### Запрос

SELECT '18-05-2016'::date;

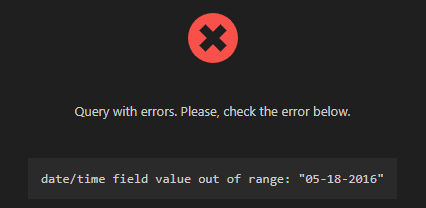
### Результат



### Запрос

SELECT '05-18-2016'::date;

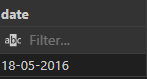
### Результат



### Запрос

SELECT '2016-05-18'::date;

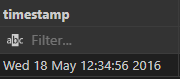
### Результат



### Запрос

SELECT '2016-05-18 12:34:56'::timestamp;

### Результат



### Запрос

SET datestyle TO 'SQL, DMY';

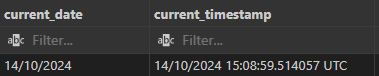
SELECT current\_date, current\_timestamp;

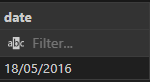
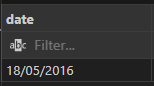
SELECT '18-05-2016'::date;

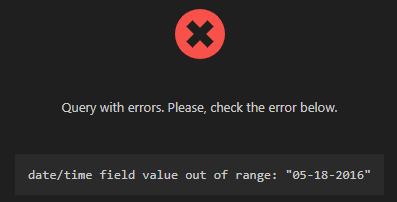
SELECT '2016-05-18'::date;

SELECT '05-18-2016'::date;

### Результат





### Запрос

SET datestyle TO 'German, DMY';

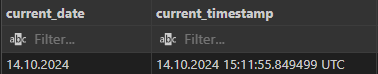
SELECT current\_date, current\_timestamp;

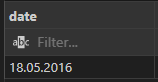
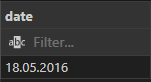
SELECT '18-05-2016'::date;

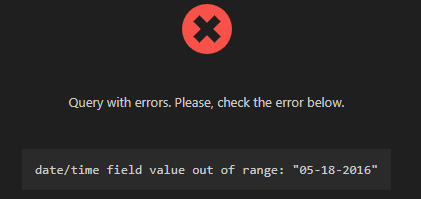
SELECT '2016-05-18'::date;

SELECT '05-18-2016'::date;

### Результат





## Задание 15

Поэкспериментируйте с функцией to\_char, извлекая из значения типа timestamp различные поля и располагая их в нужном вам порядке.

### Запрос

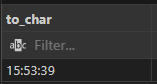
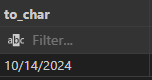
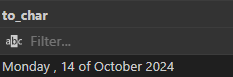
SET datestyle TO 'DMY';

SELECT to\_char(current\_timestamp, 'HH24:MI:SS');

SELECT to\_char(current\_timestamp, 'MM/DD/YYYY');

SELECT to\_char(current\_timestamp, 'Day, DD "of" Month YYYY');

### Результат

## Задание 21

Можно с высокой степенью уверенности предположить, что при прибавлении интервалов к датам и временным 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 обоснованные предположения о результатах следующих двух команд, а затем проверьте предположения на практике и проанализируйте полученные результаты:

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

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

### Решение

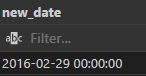
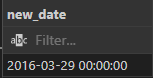
**Предположение**. Поскольку январь имеет 31 день, а при прибавлении 1 месяца мы переходим в февраль, который имеет меньше дней (28 или 29), то результатом первого запроса будет последний день февраля. Также 2016 год был високосным, и 29 февраля существует, при прибавлении 1 месяца мы перейдем в март, который имеет 31 день. Поэтому результатом второго запроса будет 31 марта.

### Запрос

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

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

### Результат

### Решение

Во втором случае предположение не совпало с результатом на практике. Вместо 31 марта мы получили 29 марта. Это связано с тем, что PostgreSQL при прибавлении интервала в 1 месяц, сохраняет день месяца, насколько это возможно.

## Задание 30

Обратимся к таблице, создаваемой с помощью команды

CREATE TABLE test\_bool

( a boolean,

b text

);

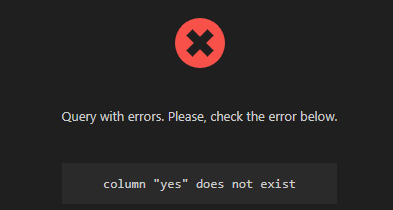
Как вы думаете, какие из приведенных ниже команд содержат ошибку? Проверьте свои предположения практически, выполнив эти команды.

### Запрос / Результат

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 );



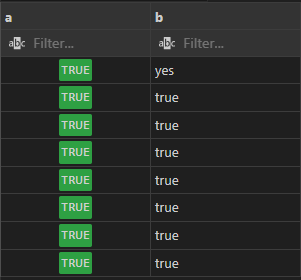
INSERT INTO test\_bool VALUES ( 1::boolean, 'true' );



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



SELECT \* FROM test\_bool;



## Задание 33

Предположим, что пилоты авиакомпании имеют возможность высказывать свои пожелания насчет конкретных блюд, из которых должен состоять их обед во время полета. Для учета пожеланий пилотов необходимо модифицировать таблицу pilots, с которой мы работали в разделе 4.5.

CREATE TABLE pilots

( pilot\_name text,

schedule integer[],

meal text[]

);

INSERT INTO pilots

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

'{ "сосиска", "макароны", "кофе" }'::text[]

),

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

'{ "котлета", "каша", "кофе" }'::text[]

),

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

'{ "сосиска", "каша", "кофе" }'::text[]

),

( 'Boris', '{ 3, 5, 6 }'::integer[],

'{ "котлета", "каша", "чай" }'::text[]

);Обратите внимание, что каждое из текстовых значений, включаемых в литерал массива, заключается в двойные кавычки, а в качестве типа данных указывается text[].

Создайте новую версию таблицы и соответственно измените команду 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[][]

### Запрос

CREATE TABLE pilots

( pilot\_name text,

schedule integer[],

meal text[]

);

INSERT INTO pilots

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

'{ "сосиска", "макароны", "кофе" }'::text[]

),

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

'{ "котлета", "каша", "кофе" }'::text[]

),

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

'{ "сосиска", "каша", "кофе" }'::text[]

),

( 'Boris', '{ 3, 5, 6 }'::integer[],

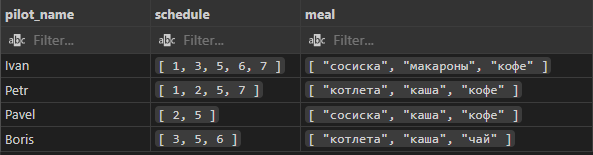
'{ "котлета", "каша", "чай" }'::text[]

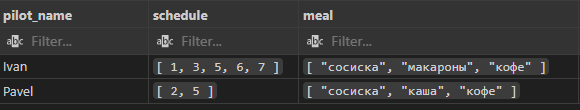
);

SELECT \* FROM pilots;

SELECT \* FROM pilots WHERE meal[ 1 ] = 'сосиска';

### Результат





### Задание

Сделайте ряд выборок и обновлений строк в этой таблице. Для обращения к элементам двумерного массива нужно использовать два индекса. Не забывайте, что по умолчанию номера индексов начинаются с единицы.

### Запрос

CREATE TABLE pilots

(

pilot\_name text,

schedule integer[],

meal text[][]

);

INSERT INTO pilots

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

'{{

"сосиска", "макароны", "кофе"

},

{

"сосиска", "макароны", "кофе"

},

{

"сосиска", "макароны", "кофе"

},

{

"сосиска", "макароны", "кофе"

}}'::text[][]),

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

'{{

"котлета", "каша", "кофе"

},

{

"котлета", "каша", "кофе"

},

{

"котлета", "каша", "кофе"

},

{

"котлета", "каша", "кофе"

}}'::text[][]),

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

'{{

"сосиска", "каша", "кофе"

},

{

"сосиска", "каша", "кофе"

},

{

"сосиска", "каша", "кофе"

},

{

"сосиска", "каша", "кофе"

}}'::text[][]),

('Boris', '{3, 5, 6}'::integer[],

'{{

"котлета", "каша", "чай"

},

{

"котлета", "каша", "чай"

},

{

"котлета", "каша", "чай"

},

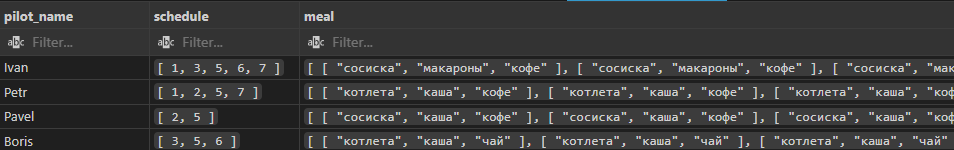
{

"котлета", "каша", "чай"

}}'::text[][]);

SELECT \* FROM pilots;

### Результат



### Запрос

SELECT meal FROM pilots WHERE pilot\_name = 'Ivan';

SELECT pilot\_name, meal[1][3] FROM pilots;

### Результат





### Запрос

UPDATE pilots

SET meal[2][2] = 'омлет';

SELECT meal[2][2] from pilots;

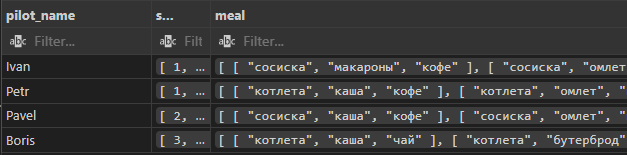
UPDATE pilots

SET meal[2][2] = 'бутерброд'

WHERE pilot\_name = 'Boris';

SELECT \* FROM pilots;

### Результат

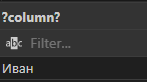


## Задание 35

Для работы с типами JSON предусмотрено много различных функций и операторов, представленных в разделе документации 9.15 «Функции и операторы JSON». Самостоятельно ознакомьтесь с ними, используя описанную технологию работы с командой SELECT.

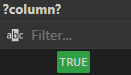
* Извлечение значения по ключу

SELECT '{ "name": "Иван", "age": 30 }'::jsonb->>'name';



* Проверка наличия ключа в JSON-объекте

SELECT '{ "name": "Иван", "age": 30 }'::jsonb ? 'age';



* Преобразование JSON-объекта в текст

SELECT CAST('{ "name": "Иван", "age": 30 }'::jsonb AS text);

