Домашняя работа №3

Попов Матвей, М8О-114СВ-24

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.

Команда для создания таблицы может быть, например, такой:

```
CREATE TABLE test_numeric
( measurement numeric,
  description text
);
```

Если у вас в базе данных уже есть таблица с таким же именем, то можно предварительно ее удалить с помощью команды

Теперь сделайте выборку из таблицы и посмотрите, что все эти разнообразные значения сохранены именно в том виде, как вы их вводили.

Запрос

```
DROP TABLE IF EXISTS test numeric;
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;
```

```
      □ measurement ӯ
      ‡
      □ description ӯ
      ‡

      1
      1234567890.0987654321
      Точность 20 знаков, масштаб 10 знаков

      2
      1.5
      Точность 2 знака, масштаб 1 знак

      3
      0.1234567890123456789
      Точность 21 знак, масштаб 20 знаков

      4
      1234567890
      Точность 10 знаков, масштаб 0 знаков (целое число)
```

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.

Например, сравним два очень маленьких числа (они представлены в экспоненциальной форме записи):

Чтобы понять, почему так получается, выполните еще два запроса.

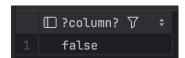
Самостоятельно проведите аналогичные эксперименты с очень большими числами, находящимися на границе допустимого диапазона для чисел типов real и double precision.

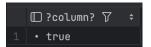
```
SELECT '5e-324'::double precision >'4e-324'::double precision;
SELECT '5e-324'::double precision;
SELECT '4e-324'::double precision;

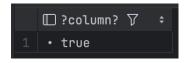
SELECT '5e256'::double precision > '4e256'::double precision;
SELECT '5e256'::double precision;
SELECT '4e256'::double precision;
```

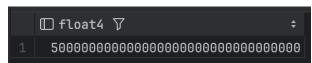
```
SELECT '5e32'::real > '4e32'::real;
SELECT '5e32'::real;
SELECT '4e32'::real;
```

Результат









Немного усложним определение таблицы из предыдущего задания. Пусть теперь столбец id будет первичным ключом этой 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

```
DROP TABLE IF EXISTS test_serial;

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' );

INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );

INSERT INTO test_serial ( name ) VALUES ('Грушевая' );

INSERT INTO test_serial ( name ) VALUES ('Зеленая' );

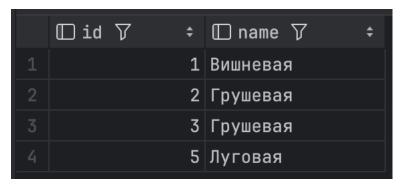
INSERT INTO test_serial ( name ) VALUES ('Зеленая' );

DELETE FROM test_serial WHERE id = 4;

INSERT INTO test_serial ( name ) VALUES ('Луговая' );

SELECT * FROM test serial;
```

Результат



Задание 12

Самостоятельно выполните команды SELECT, приведенные выше, как для значения типа date, так и для значения типа timestamp. Обратите внимание, что если выбран формат Postgres, то порядок следования составных частей даты (день, месяц, год), заданный в параметре datestyle, используется не только при вводе значений, но и при выводе. Напомним, что вводом мы считаем команду SELECT, а выводом — результат ее выполнения, выведенный на экран.

Запрос

```
SET datestyle TO 'MDY';

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

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

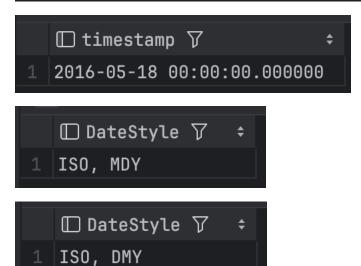
SET datestyle TO DEFAULT;

SHOW datestyle;

SET datestyle TO 'ISO, DMY';

SHOW datestyle;
```

```
[22008] ERROR: date/time field value out of range: "18-05-2016" Hint: Perhaps you need a different "datestyle" setting. Position: 8
```



Приведем несколько команд, иллюстрирующих использование этой функции. Ее первым параметром является форматируемое значение, а вторым — шаблон, описывающий формат, в котором это значение будет представлено при вводе или выводе. 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.

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

Запрос

```
SELECT to_char(current_timestamp, 'dd-yyyy-mm');
SELECT to_char(current_timestamp, 'mi:hh');
```



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;

3aπpoc
SELECT ('2016-01-31'::date + '1 mon'::interval ) AS new_date;
SELECT ('2016-02-29'::date + '1 mon'::interval ) AS new_date;
```

Результат

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?

```
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 ( 't', '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 ( 1::boolean, 'true' );
```

Проверьте свои предположения практически, выполнив эти команды.

```
DROP TABLE IF EXISTS test bool;
CREATE TABLE test bool
   a boolean,
   b text
);
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' );
Результат
INSERT INTO test bool VALUES ( TRUE, 'yes' )
ERROR: column "yes" does not exist
INSERT INTO test bool VALUES ('yes', true )
INSERT INTO test bool VALUES ('yes', TRUE )
INSERT INTO test bool VALUES ('1', 'true')
ERROR: column "a" is of type boolean but expression is of type
integer
```

```
INSERT INTO test_bool VALUES ('t','true')
ERROR: column "truth" does not exist
INSERT INTO test_bool VALUES ( true, true )
INSERT INTO test_bool VALUES ( 1::boolean,'true' )
INSERT INTO test_bool VALUES ( 111::boolean,'true' )
```

Задание. Создайте новую версию таблицы и соответственно измените команду INSERT, чтобы в ней содержались литералы $\partial в y м e p h b x$ массивов. Они будут выглядеть примерно так:

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.

```
DROP TABLE IF EXISTS pilots;
CREATE TABLE pilots
   pilot name text,
   schedule integer[],
   meal text[][]
);
INSERT INTO pilots
VALUES ('NBAH',
        '{1, 3, 5, 6, 7}'::integer[],
        '{ "сосиска", "макароны", "кофе" },
        { "котлета", "каша", "кофе" },
        { "сосиска", "каша", "кофе" },
        { "котлета", "каша", "чай" } }'::text[][]);
SELECT meal[2][2] FROM pilots;
UPDATE pilots
SET meal[2][2] = 'cyn';
```

```
SELECT meal[2][2] FROM pilots;
```

Результат



Задание 35

Изучая приемы работы с типами JSON, можно, как и в случае с массивами, пользоваться способностью команды SELECT обходиться без создания таблиц.

Покажем лишь один пример. Добавить новый ключ и соответствующее ему значения в уже существующий объект можно оператором | |:

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.

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

SELECT '{
    "sports": "хоккей"
}'::jsonb || '{
    "trips": 5
}'::jsonb || '{
    "pilots": "ИВАН"
}'::jsonb;
```