МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Московский авиационный институт”

(национальный исследовательский университет)

Факультет №3 «Системы управления, информатика и электроэнергетика»

Отчет по домашней работе №3

по курсу «**Управление базами данных**»

Выполнили:

студенты группы 3О-218М-19:

Пономарев Роман

Принял:

Доцент, к.т.н. Моргунов Е.П.

Москва, 2020

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 **DROP TABLE test_numeric;**

Вставьте в таблицу несколько строк:

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)');

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

Ответ: SELECT * FROM text_numeric;

```
demo=# CREATE TABLE test_numeric
demo-# (measurement numeric,
demo(# description text
demo(# );
CREATE TABLE
demo=# INSERT INTO test_numeric
demo-# VALUES (1234567890.0987654321,
demo(# 'Точность 20 знаков, масштаб 10 знаков');
INSERT 0 1
demo=# INSERT INTO test_numeric
VALUES (1.5,
'Точность 2 знака, масштаб 1 знак');
INSERT 0 1
demo=# INSERT INTO test_numeric
VALUES (0.12345678901234567890,
'Точность 21 знак, масштаб 20 знаков');
INSERT 0 1
demo=# INSERT INTO test_numeric
VALUES (1234567890,
'Точность 10 знаков, масштаб 0 знаков(целое число)');
INSERT 0 1
demo=# SELECT * FR

demo=# SELECT * FRO

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) 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):
SELECT '5e-324'::double precision > '4e-324'::double precision;
Чтобы понять, почему так получается, выполните еще два запроса.
SELECT '5e-324'::double precision;
SELECT '4e-324'::double precision;
Самостоятельно проведите аналогичные эксперименты с очень большими числами, находящимися на границе допустимого диапазона для чисел типов **real** и **double precision**.

Ответ:

```
demo=# SELECT '1E+37'::real >'0E+37'::real;
 ?column?
----------
 t
(1 строка)

demo=# SELECT '1E+35'::real >'0E+37'::real;
 ?column?
----------
 t
(1 строка)

demo=# SELECT '1E+308'::double precision >'0E+308'::double precision;
 ?column?
----------
 t
(1 строка)

demo=# SELECT '1E+307'::double precision >'0E+308'::double precision;
 ?column?
----------
 t
(1 строка)
```

8) 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.

```
demo=# INSERT INTO test_serial (id,name) VALUES (2, 'Прохладная');
INSERT 0 1
demo=# SELECT * FROM test_serial;
 id |    name
----+------------
  1 | Вишневая
  2 | Прохладная
(2 строки)

demo=# INSERT INTO test_serial (name) VALUES ('Грушевая');
ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "test_serial_pkey"
ПОДРОБНОСТИ:  Ключ "(id)=(2)" уже существует.
demo=# INSERT INTO test_serial (name) VALUES ('Грушевая');
INSERT 0 1
demo=# SELECT * FROM test_serial;
 id |    name
----+------------
  1 | Вишневая
  2 | Прохладная
  3 | Грушевая
(3 строки)

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

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

12) SET datestyle TO 'MDY';
Повторим одну из команд, выполненных ранее. Теперь она должна вызвать ошибку. Почему?
SELECT '18-05-2016'::date;
А такая команда, наоборот, теперь будет успешно выполнена:
SELECT '05-18-2016'::date;
Ответ: Это происходит потому что мы изменили порядок ввода данных типа date на месяц, день, год. А в году не может быть 18 месяцев, отсюда и получается ошибка.

Самостоятельно выполните команды SELECT, приведенные выше, 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.

```
demo=# SET datestyle TO 'SQL, DMY';
SET
demo=# SELECT '18-05-2016'::date;
    date
------------
 18/05/2016
(1 строка)

demo=# SELECT '05-18-2016'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::date;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
demo=# SELECT '18-05-2016'::timestamp;
      timestamp
---------------------
 18/05/2016 00:00:00
(1 строка)

demo=# SELECT '05-18-2016'::timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
demo=# SET datestyle TO 'German, DMY';
SET
demo=# SELECT '18-05-2016'::date;
    date
------------
 18.05.2016
(1 строка)

demo=# SELECT '05-18-2016'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::date;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
demo=# SELECT '05-18-2016'::timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
demo=# SELECT '18-05-2016'::timestamp;
      timestamp
---------------------
 18.05.2016 00:00:00
(1 строка)
```

15) В документации в разделе 9.8 «Функции форматирования данных» представлены описания множества полезных функций, позволяющих преобразовать в строку данные других типов, например, **timestamp**. Одна из таких функций **to_char**. 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.
SELECT to_char(current_timestamp, 'mi:ss');
SELECT to_char(current_timestamp, 'dd'); SELECT to_char(current_timestamp, 'yyyy-mm-dd');
Поэкспериментируйте с этой функцией, извлекая из значения типа **timestamp** различные поля и располагая их в нужном вам порядке.

```
demo=# SELECT to_char('2014-04-04'::timestamp, 'yyyy-mm-dd');
  to_char
------------
 2014-04-04
(1 строка)

demo=# SELECT to_char('2014-04-04'::timestamp, 'yyyy');
 to_char
---------
 2014
(1 строка)

demo=# SELECT to_char('2014-04-04'::timestamp, 'dd');
 to_char
---------
 04
(1 строка)
```

21) Можно с высокой степенью уверенности предположить, что при прибавлении интервалов к датам и временны́м 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:

```
demo=# SELECT ('2016-01-31'::date + '1 mon'::interval) AS new_date;
      new_date
---------------------
 2016-02-29 00:00:00
(1 строка)

demo=# SELECT ('2016-02-29'::date + '1 mon'::interval) AS new_date;
      new_date
---------------------
 2016-03-29 00:00:00
(1 строка)
```

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');
Ответ: во втором запросе будет ошибка, что колонки yes не существует;
В 6 запросе будет ошибка, так как мы в boolean пытаемся занести integer;
В 8 запросе будет ошибка, что колонки truth не существует;

```
demo=# CREATE TABLE test_bool
demo-# (a boolean,
demo(# b text
demo(# );
CREATE TABLE
demo=# INSERT INTO test_bool VALUES (TR

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', 'yes');
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
demo=# SELECT * FROM test_bool;
 a |  b
---+------
 t | yes
```

33)

```
demo=# INSERT INTO pilots
VALUES ('Ivan', '{1, 3, 5, 6, 7}'::integer[],
'{{"сосиска","макароны","кофе"},{"котлета","каша","кофе"}, {"сосиска","каша","кофе"}, {"котлета","каша","чай"}}'::t
ext[][]
),
('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[]
);
INSERT 0 4
```

```
demo=# SELECT * FROM pilots;
 pilot_name |  schedule   |                                          meal
------------+-------------+-----------------------------------------------------------------------------------------
 Ivan       | {1,3,5,6,7} | {сосиска,макароны,кофе}
 Petr       | {1,2,5,7}   | {котлета,каша,кофе}
 Pavel      | {2,5}       | {сосиска,каша,кофе}
 Boris      | {3,5,6}     | {котлета,каша,чай}
 Ivan       | {1,3,5,6,7} | {{сосиска,макароны,кофе},{котлета,каша,кофе},{сосиска,каша,кофе},{котлета,каша,чай}}
 Petr       | {1,2,5,7}   | {котлета,каша,кофе}
 Pavel      | {2,5}       | {сосиска,каша,кофе}
 Boris      | {3,5,6}     | {котлета,каша,чай}
(8 строк)

demo=# SELECT * FROM pilots WHERE meal[1] = 'сосиска';
 pilot_name |  schedule   |          meal
------------+-------------+------------------------
 Ivan       | {1,3,5,6,7} | {сосиска,макароны,кофе}
 Pavel      | {2,5}       | {сосиска,каша,кофе}
 Pavel      | {2,5}       | {сосиска,каша,кофе}
(3 строки)

demo=# SELECT * FROM pilots WHERE meal[1][2] = 'каша';
 pilot_name | schedule | meal
------------+----------+------
(0 строк)

demo=# SELECT * FROM pilots WHERE meal[1][2] = 'макароны';
 pilot_name |  schedule   |                                          meal
------------+-------------+-----------------------------------------------------------------------------------------
 Ivan       | {1,3,5,6,7} | {{сосиска,макароны,кофе},{котлета,каша,кофе},{сосиска,каша,кофе},{котлета,каша,чай}}
(1 строка)
```

35)

```
demo=# SELECT '{"sports": "football"}'::jsonb || '{"type":"american"}'::jsonb;
                    ?column?
-----------------------------------------------
 {"type": "american", "sports": "football"}
(1 строка)
```