Федеральное государственное автономное

Образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

«Типы данных СУБД PostgreSQL»

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Е.П. Моргунов

подпись, дата инициалы, фамилия

Студент КИ16-16б, №031620303 \_\_\_\_\_\_\_\_\_ В.В. Кокташев

Номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

# Цель работы

# ознакомиться с типами данных СУБД PostgreSQL;

# выполнить задания в конце четвертого раздела учебного пособия.

# Ход работы

1. Создайте таблицу, содержащую атрибут типа numeric (precision, scale). Пусть это будет таблица, содержащая результаты каких-то измерений. Попробуйте с помощью команды INSERT продемонстрировать округление вводимого числа до той точности, которая задана при создании таблицы (рис. 1).

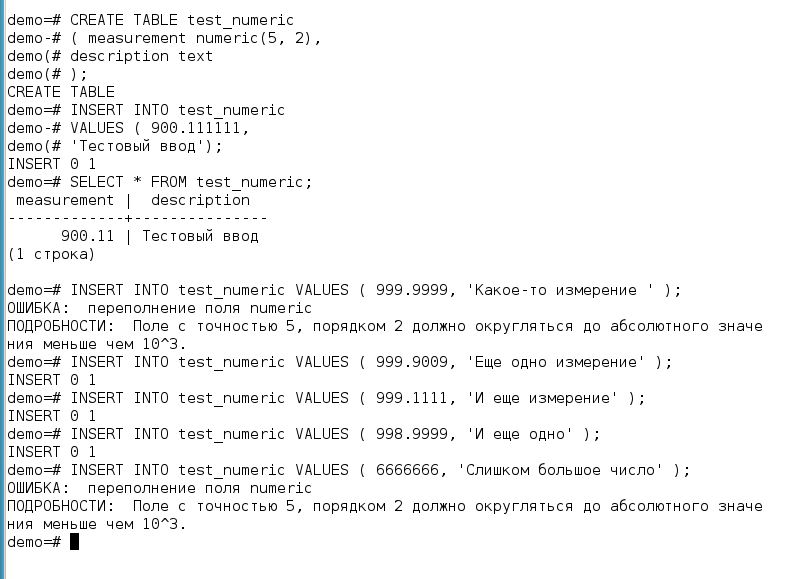


Рисунок 1 – Задание 1

Подумайте, какая из следующих команд вызовет ошибку и почему?

INSERT INTO test\_numeric VALUES (999.9999, 'Какое-то измерение ' );

INSERT INTO test\_numeric VALUES ( 999.9009, 'Еще одно измерение' )

INSERT INTO test\_numeric VALUES ( 999.1111, 'И еще измерение' );

INSERT INTO test\_numeric VALUES ( 998.9999, 'И еще одно' );

Ответ: ошибку вызовет выполнение первой строки, так как при округлении число превысит допустимый размер (numeric(5, 2))

1. Тип данных numeric поддерживает специальное значение NaN, которое означает «не число» (not a number). В документации утверждается, что значение NaN считается равным другому значению NaN, а также что значение NaN считается большим любого другого «нормального» значения, т. е. не - NaN. Проверьте эти ´ утверждения с помощью SQL команды SELECT (рис. 2).

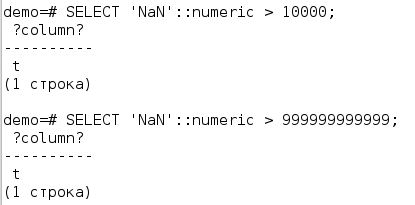


Рисунок 2 – Задание 3

1. Выполните сравнение Infinity с наименьшим значения типа double precision (рис. 3).

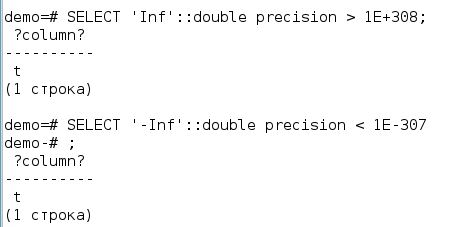


Рисунок 3 – Задание 5

1. В качестве иллюстрация применения типа serial предложим таблицу, содержащую наименования улиц и площадей.

CREATE TABLE test\_serial (id serial, name text );

Введите несколько строк. Значения для столбца id указывать не обязательно (и даже не нужно).

Сделайте выборку данных из таблицы, вы увидите, что значения столбца id имеют последовательные значения, начиная с 1 (рис. 4). Давайте проведем эксперимент со столбцом id (рис. 4). Выполните команду INSERT, в которой укажите явное значение столбца id:

INSERT INTO test\_serial ( id, name ) VALUES ( 10, 'Прохладная' );

А теперь добавьте еще одну строку, но уже не указывая явно значение для столбца id (как мы поступали в предыдущих командах):

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

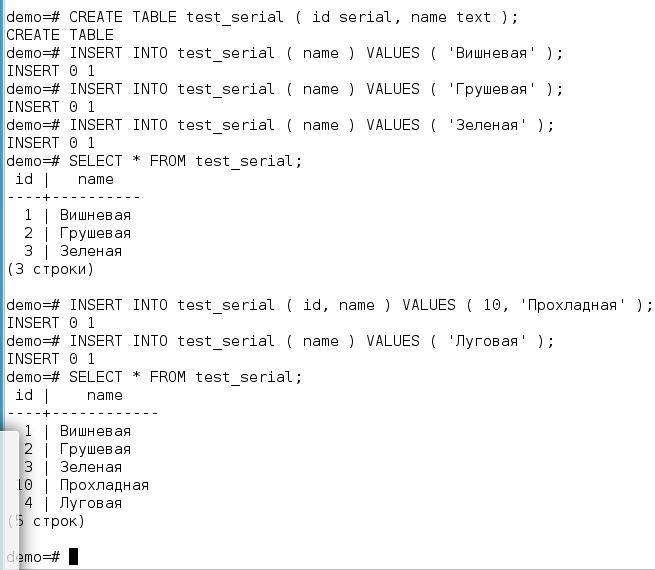


Рисунок 4 – Задание 7

1. Какой календарь используется в PostgreSQL для работы с датами: юлианский или григорианский?

Ответ: григорианский.

1. Типы timestamp, time и interval позволяют задать точность ввода и вывода значений. Точность предписывает количество десятичных цифр в поле секунд. Проиллюстрируем эту возможность на примере типа time, выполнив три запроса: в первом запросе вообще не используем параметр точности, во втором назначим его равным 0, в третьем запросе сделаем его равным 3.

Выполните подобные команды для типа interval (рис. 5)

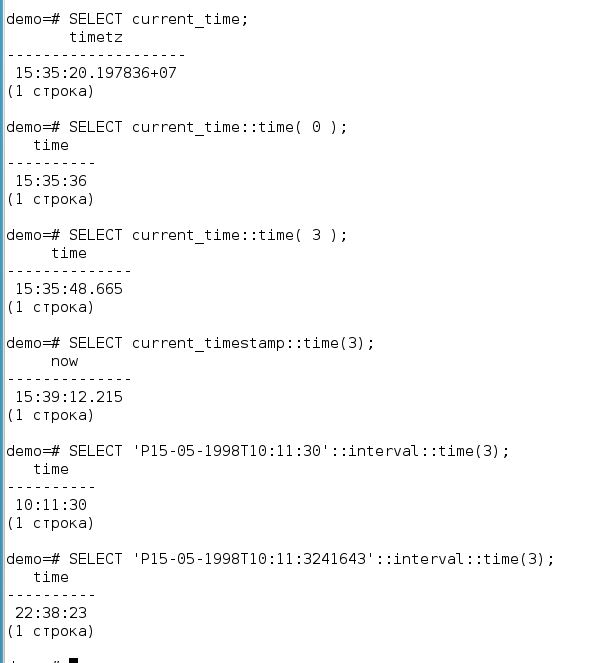


Рисунок 5 – Задание 11

Тип date такой возможности — задавать точность — не имеет. Как вы думаете, почему?

Ответ: потому что у даты нет нецелых значений.

1. Установите новое значение параметра datestyle с помощью создания переменной системного окружения PGDATESTYLE (рис. 6).

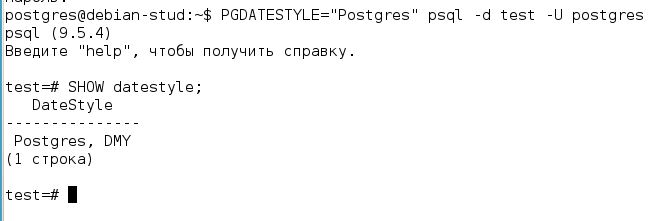


Рисунок 6 – Задание 13

1. 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().

Поэкспериментируйте с этой функцией, извлекая из значения типа timestamp различные поля и располагая их в нужном вам порядке (рис. 7).

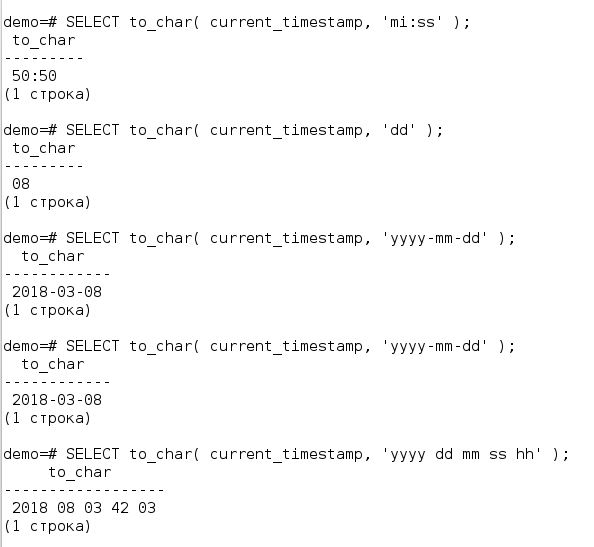


Рисунок 7 – Задание 15

1. При выполнении приведения типа данных производится проверка значения на допустимость. Попробуйте ввести недопустимое значение времени, например, с нарушением формата (рис. 8).

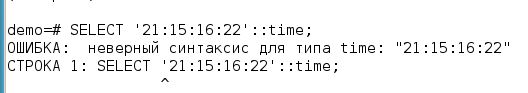


Рисунок 8 – Задание 17

1. С типами даты и времени можно выполнять различные арифметические операции. Как правило, их применение является интуитивно понятным. Выполните следующую команду и проанализируйте результат.

SELECT ( '20:34:35'::time - '19:44:45'::time );

Попробуйте предположить, какой результат будет получен, если в этой команде знак «минус» заменить на знак «плюс» (рис. 9)?

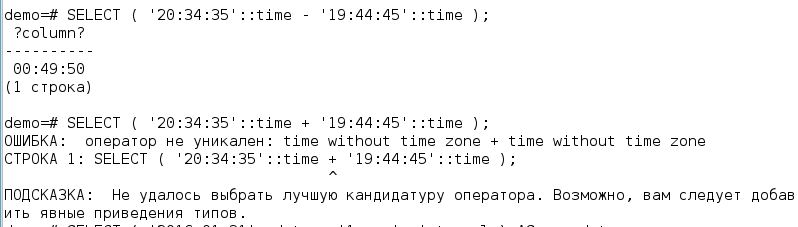


Рисунок 9 – Задание 19

1. Что получится при прибавлении интервала в 1 месяц к последнему дню января и к последнему дню февраля (рис. 10)?

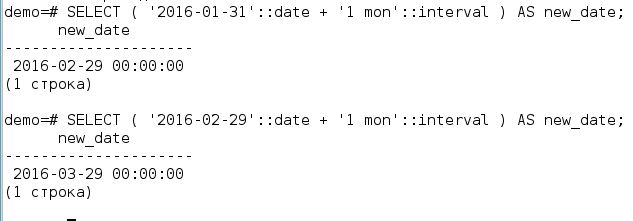


Рисунок 10 – Задание 21

1. Выполните следующие две команды и объясните различия в выведенных результатах (рис. 11):

SELECT ( '2016-09-16'::date - '2015-09-01'::date );

SELECT ( '2016-09-16'::timestamp - '2015-09-01'::timestamp )

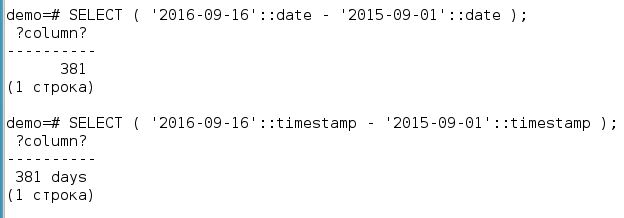


Рисунок 11 – Задание 23

Разница, в том, что date считает с точностью до дней, а timestamp – до микросекунд.

1. Значения временных отметок можно усекать с той или иной точностью с помощью функции date\_trunc().

Выполните эту команду, последовательно указывая в качестве первого параметра значения microsecond, millisecond, second, minute, hour, day, week, month, year, decade, century, millennium (рис. 12).

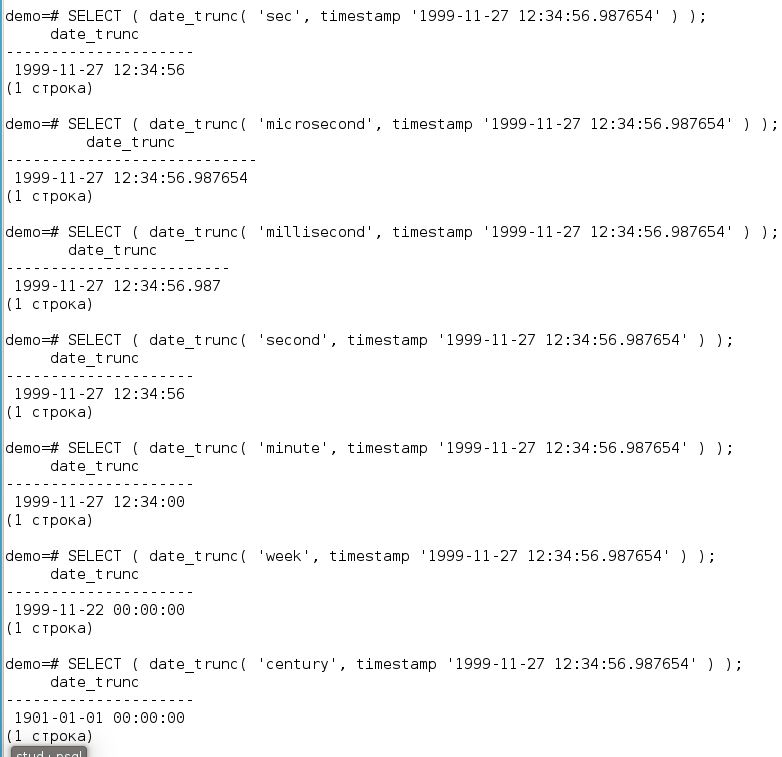


Рисунок 12 – Задание 25

1. Весьма полезна функция extract(). С ее помощью можно извлечь значение отдельного поля из временной отметки timestamp.

Выполните эту команду, последовательно указывая в качестве первого параметра значения microsecond, millisecond, second, minute, hour, day, week, month, year, decade, century, millennium (рис. 13).

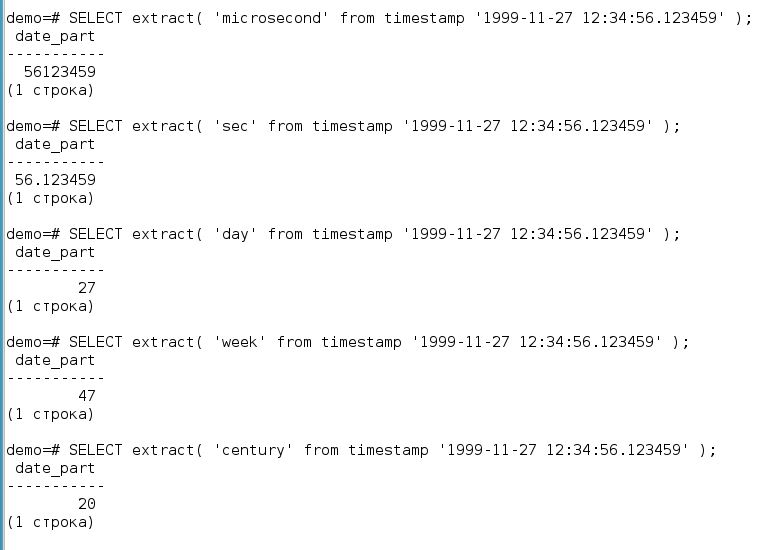


Рисунок 13 – Задание 27

1. Как вы думаете, являются ли все приведенные ниже команды равнозначными в смысле результатов, получаемых с их помощью?

SELECT \* FROM databases WHERE NOT is\_open\_source;

SELECT \* FROM databases WHERE is\_open\_source <> 'yes';

SELECT \* FROM databases WHERE is\_open\_source <> 't';

SELECT \* FROM databases WHERE is\_open\_source <> '1';

SELECT \* FROM databases WHERE is\_open\_source <> 1;

Ответ: нет, не является. Результаты выполнения первой и последней строчки различаются, по сравнению с другими.

1. Пусть в таблице birthdays хранятся даты рождения какой-то группы людей.

Если мы захотим определить точный возраст каждого человека на текущий момент времени, то как получить этот результат (рис. 14)?

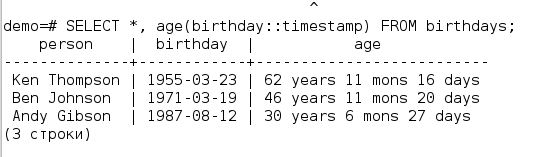


Рисунок 14 – Задание 31

1. Создайте новую версию таблицы и соответственно измените команду INSERT, чтобы в ней содержались литералы двумерных массивов (рис. 29).

Сделайте ряд выборок и обновлений строк в этой таблице (рис. 15-17).



Рисунок 15 – Задание 33, окно вывода 1

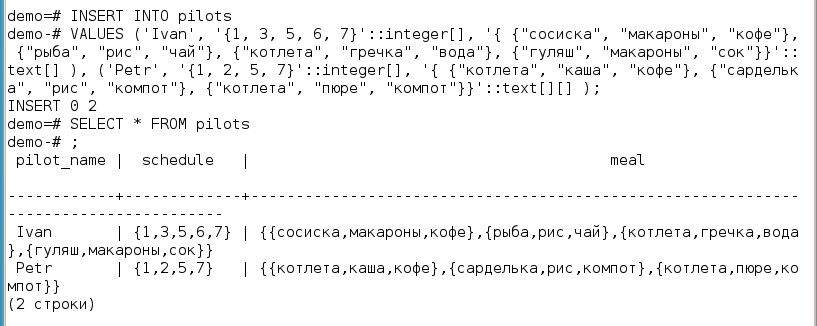


Рисунок 16 – Задание 33, окно вывода 2

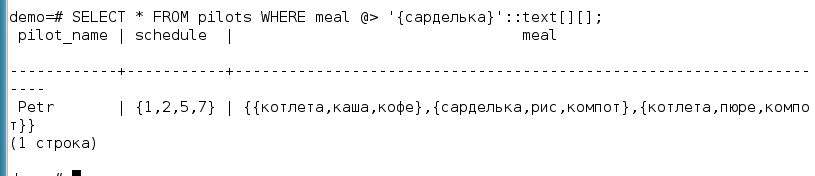


Рисунок 17 – Задание 33, окно вывода 3

1. В тексте раздела 4.6 мы выполняли обновление JSON-объекта с помощью функции jsonb\_set: добавляли значение в массив.

Самостоятельно выполните изменение значения по ключу home\_lib в одной из строк таблицы (рис. 18).

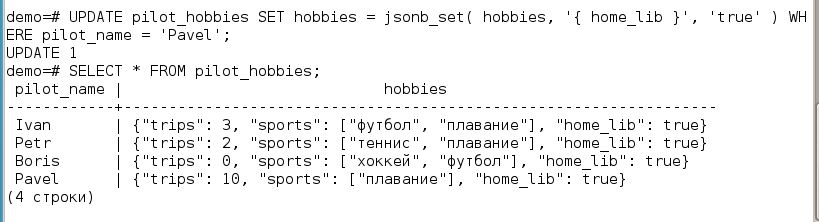


Рисунок 18 – Задание 34

1. Объекты JSON в разных строках таблицы могут иметь различные наборы ключей. Добавьте дополнительный ключ и соответствующее ему значение в JSON - объект какой-нибудь строки таблицы pilots. Воспользуйтесь оператором «| |» (рис. 19).

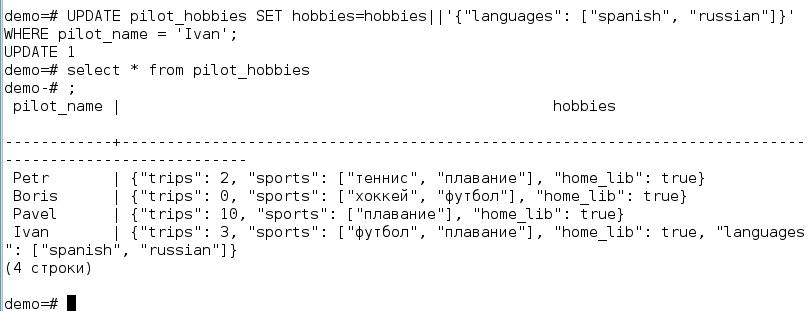


Рисунок 19 – Задание 36

# Вывод

В ходе выполнения практической работы изучены типы данных СУБД PostgreSQL и полученная теория закреплена на практике.