**Установка PostgreSQL**

Для установки СУБД нужно скачать установщик на сайте (рисунок 1) [1].

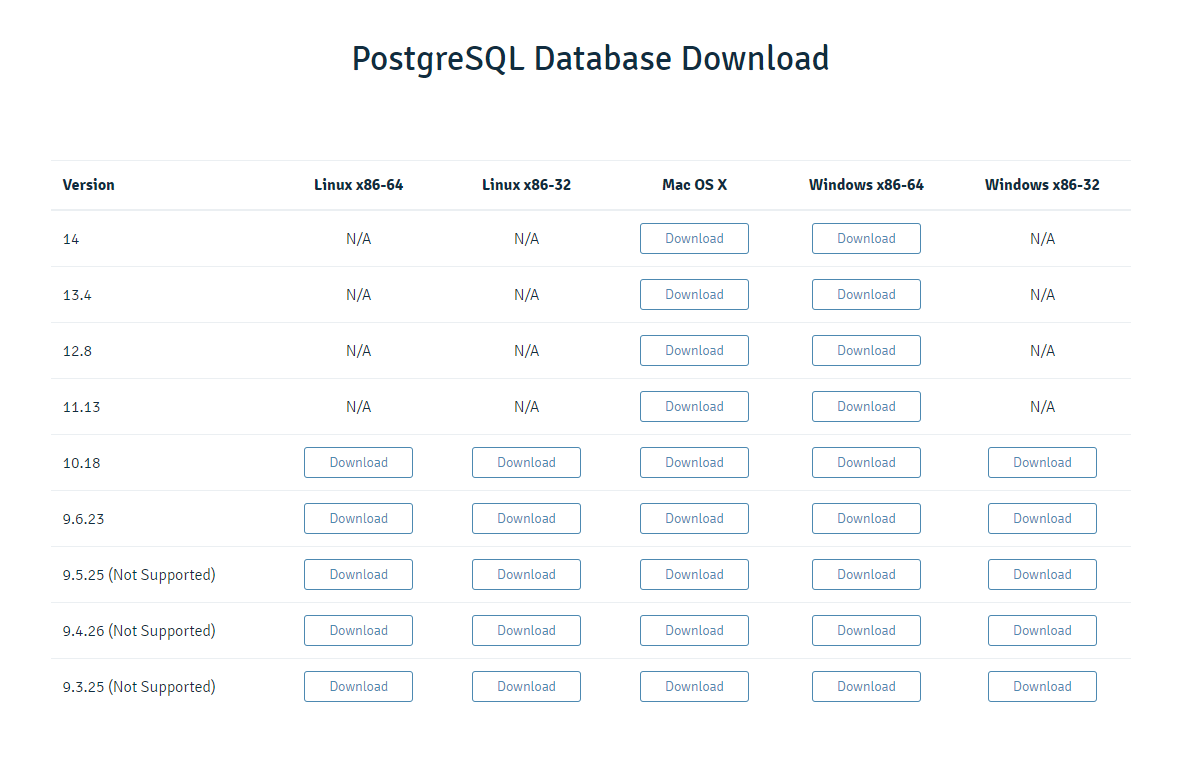


Рисунок 1 – Предложенные версии PostgreSQL

После запуска установщика запустится окно приветствия (рисунок 2). Далее будет предложено выбрать директорию, по которой будет установлено СУБД (рисунок 3). Затем будет предложено выбрать компоненты. На всех необходимо установить галочку для нормального взаимодействия с СУБД (рисунок 4). После этого будет предложено указать путь к данным, которые хранятся в СУБД (рисунок 5). Далее необходимо придумать пароль, который будет использоваться при подключении к базе данных (рисунок 6). Затем предлагают номер порта, который используется при подключении к базе данных. Можно оставить предложенный установщиком номер порта (рисунок 7). Далее будет предложено выбрать региональные настройки, оставляем по умолчанию (рисунок 8). После установки СУБД будет предложено запустить Stack Builder. Он пригодится для создания хранимых процедур и функций на языке python, поэтому оставляем галочку (рисунок 9).

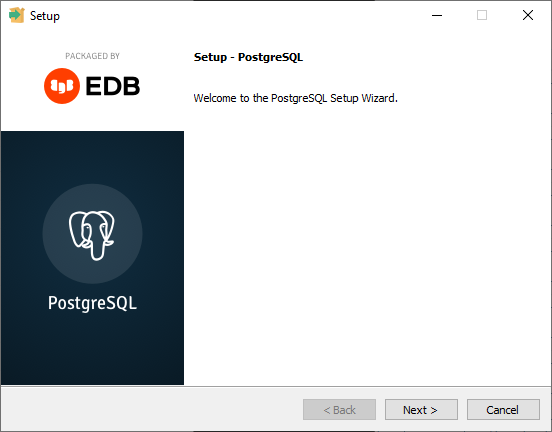


Рисунок 2 – Окно приветствия установщика СУБД

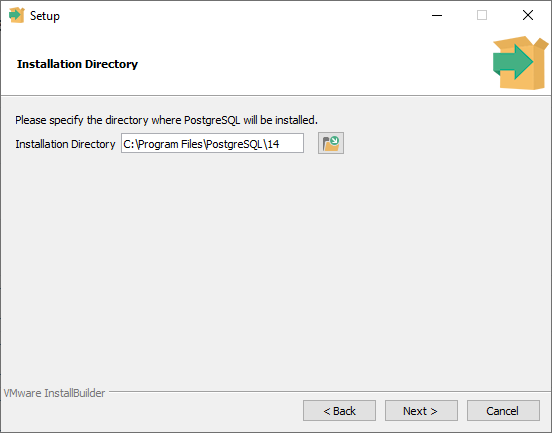


Рисунок 3 – Директория СУБД

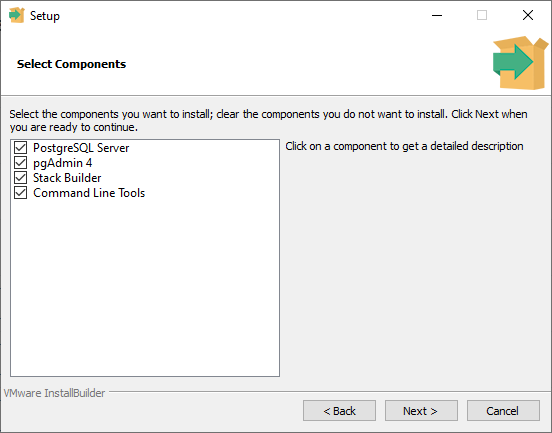


Рисунок 4 – Компоненты СУБД

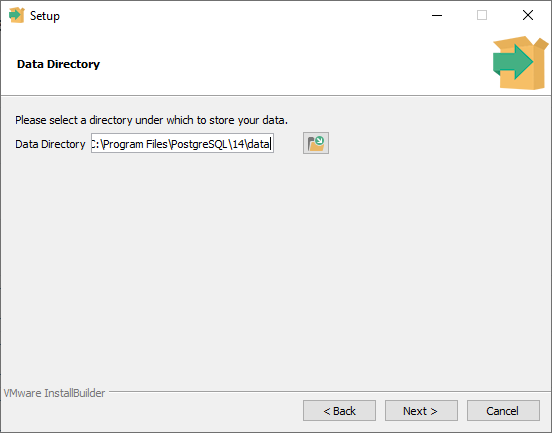


Рисунок 5 – Данные СУБД

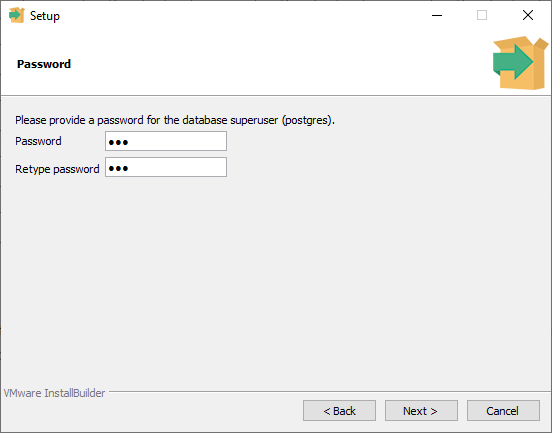


Рисунок 6 – Пароль СУБД

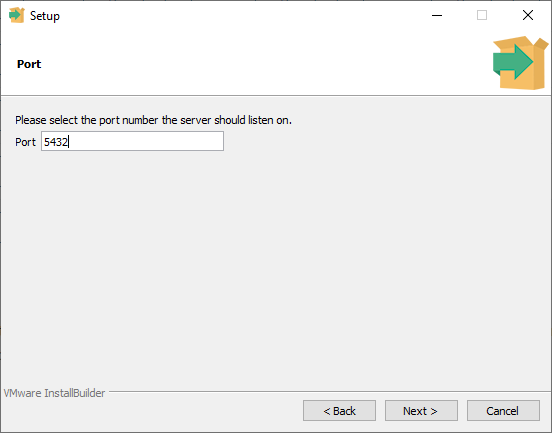


Рисунок 7 – Номер порта СУБД

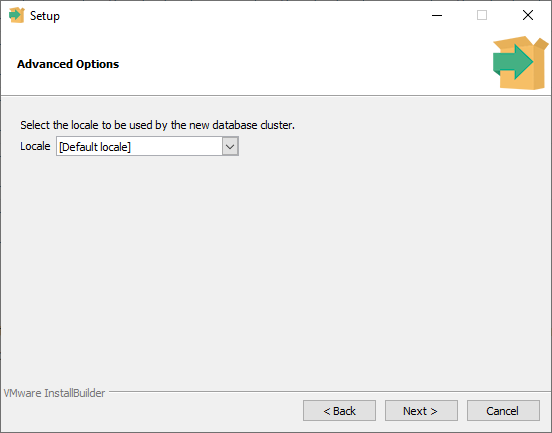


Рисунок 8 – Региональные настройки СУБД



Рисунок 9 – Завершение установки СУБД

**Настройка plpython3u**

После установки PostgreSQL необходимо запустить Stack Builder. При запуске будет выведено окно приветствия, на котором необходимо выбрать версию СУБД (рисунок 10). В категории Add-ons, tools and utilities выбираем опцию EDB Language Pack v2.0-1 (рисунок 11). Далее выбираем каталог загрузки (рисунок 12).

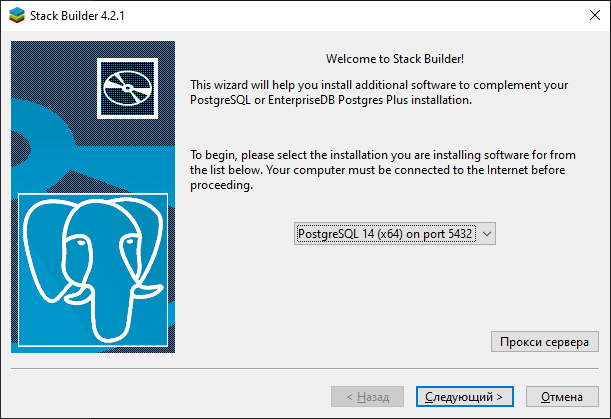


Рисунок 10 – Окно приветствия Stack Builder

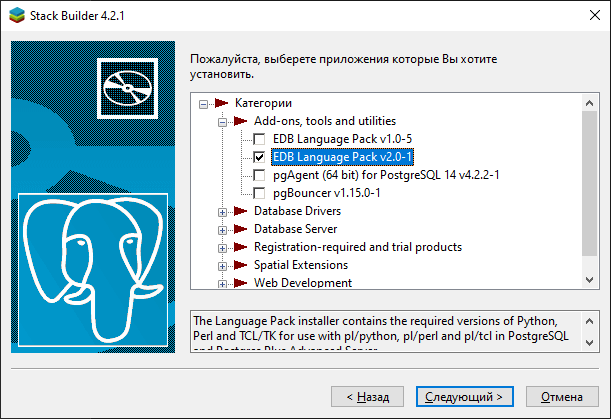


Рисунок 11 – Выбор дополнительных инструментов СУБД

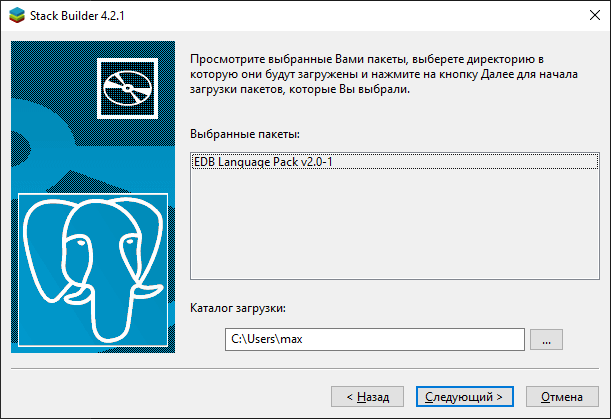


Рисунок 12 – Директория установки инструментов

После установки необходимо узнать версию python, которая требуется для PostgreSQL для этого необходимо перейти в папку с документацией. Если при установке postgre задать путь по умолчанию, то путь до документации, будет следующий: C:\Program Files\PostgreSQL\14\doc. В этой папке следует запустить файл «installation-notes.html» и узнать версию python необходимую для версии postgrSQL, которую вы скачали (рисунок 13). Если версия python, указанная в документации, ниже, чем версия, которая была установлена через Stack Buider, то следует перейти в папку, в которую был установлен python. По умолчанию: C:\edb\languagepack\v2\Python-3.9. В этой папке необходимо файл «python39.dll» переименовать таким образом, чтобы она соответствовала версии, указанной в документации. То есть, если в документации указана версия python 3.7, то файл «python39.dll» переименовываем в файл «python37.dll» (рисунок 14).

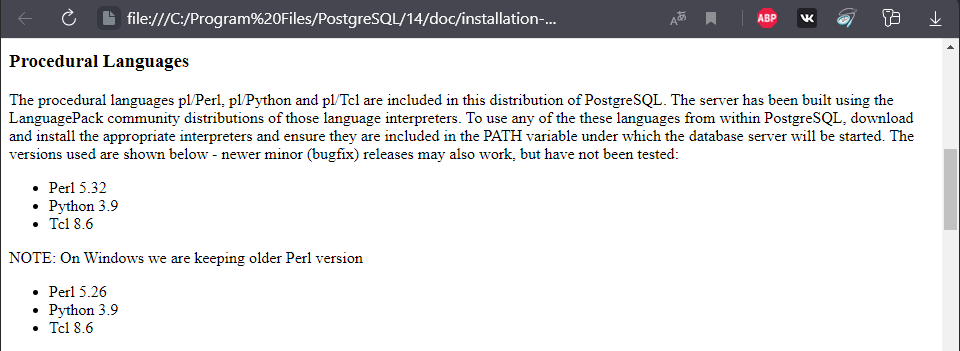


Рисунок 13 – Документация PostgreSQL

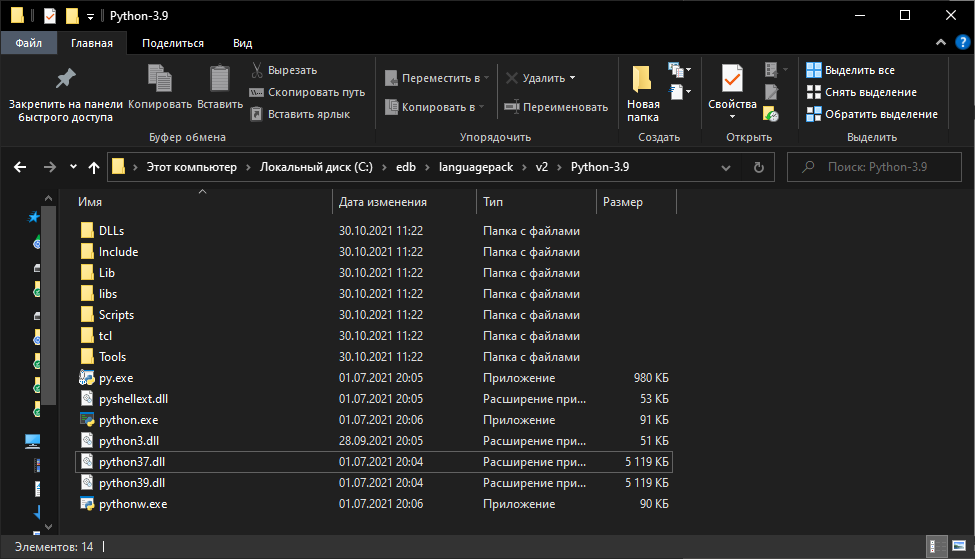


Рисунок 14 – Директория python-3.9

После этого в поиске прописываем система (рисунок 15). В открывшимся окне выбираем параметр «Дополнительные параметры системы» (рисунок 16). В новом окне нажимаем на кнопку «Переменные среды…» (рисунок 17).

В группе «Системные переменные» нажимаем на кнопку «Создать…» и задаем имя переменной «PYTHONHOME», а значение «C:\edb\languagepack\v2\Python-3.9» (рисунок 18). Далее дважды кликаем по элементу «Path» (рисунок 19). В появившемся окне нужно нажать на кнопку «Создать» и прописать %PYTHONHOME%, после чего снова нажать на кнопку «Создать» и прописать путь до директории с python (рисунок 20). После выше проделанных действий нужно обязательно перезагрузить компьютер.

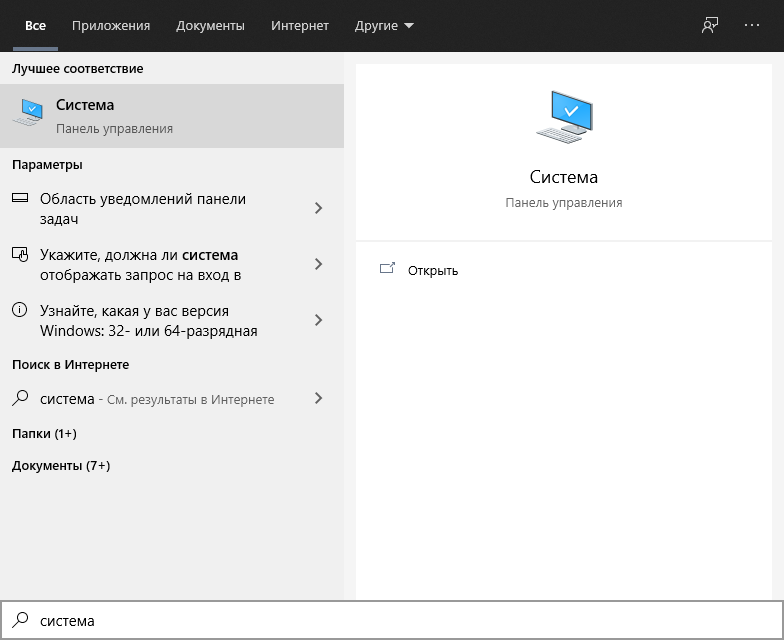


Рисунок 15 – Поиск системы

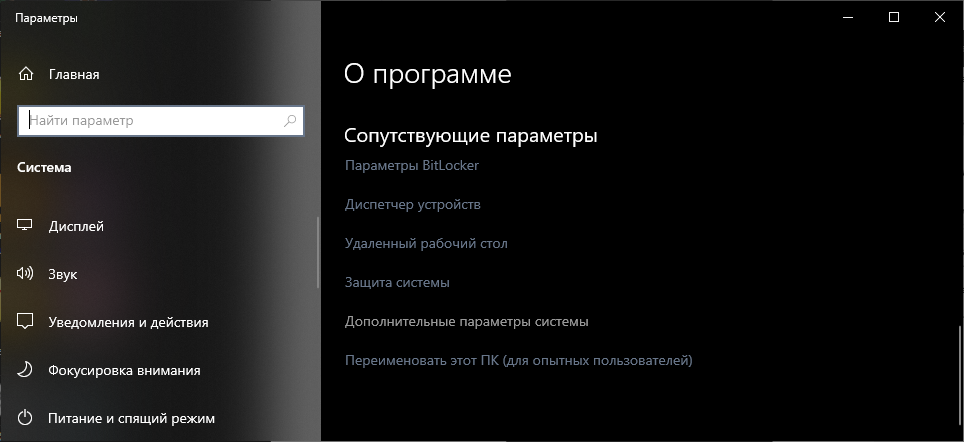


Рисунок 16 – Дополнительные параметры системы

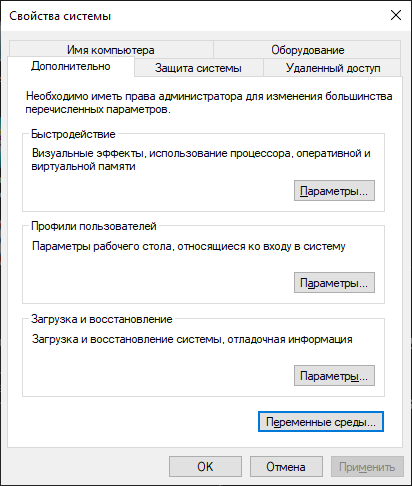


Рисунок 17 – Свойства системы

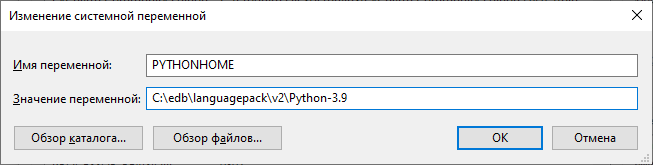


Рисунок 18 – Изменение системной переменной

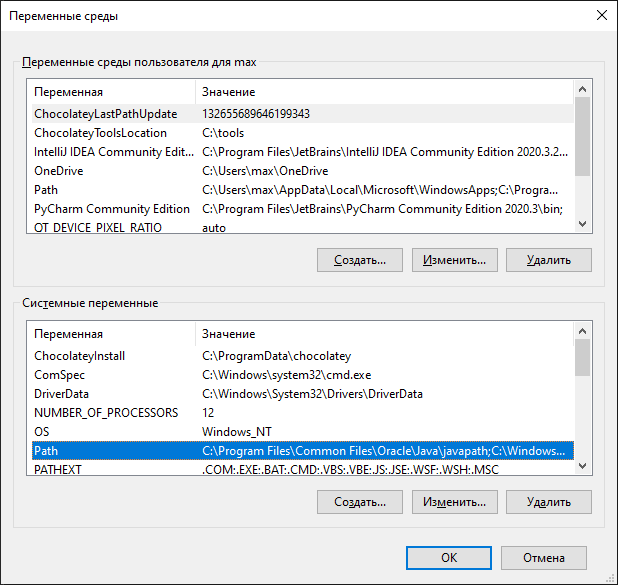


Рисунок 19 – Переменные среды

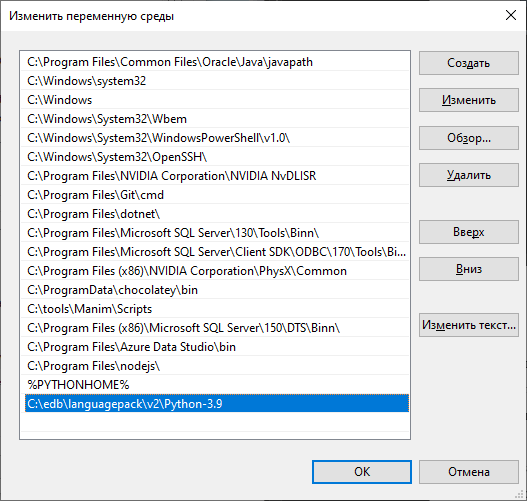


Рисунок 20 – Изменение переменной среды

**Взаимодействие с СУБД через графический интерфейс**

Для запуска графического интерфейса для взаимодействия с СУБД необходимо запустить pgAdmin4, которую можно найти через поиск (рисунок 21).

При запуске pgAdmin4 необходимо указать пароль, который был указан при установке СУБД (рисунок 22).

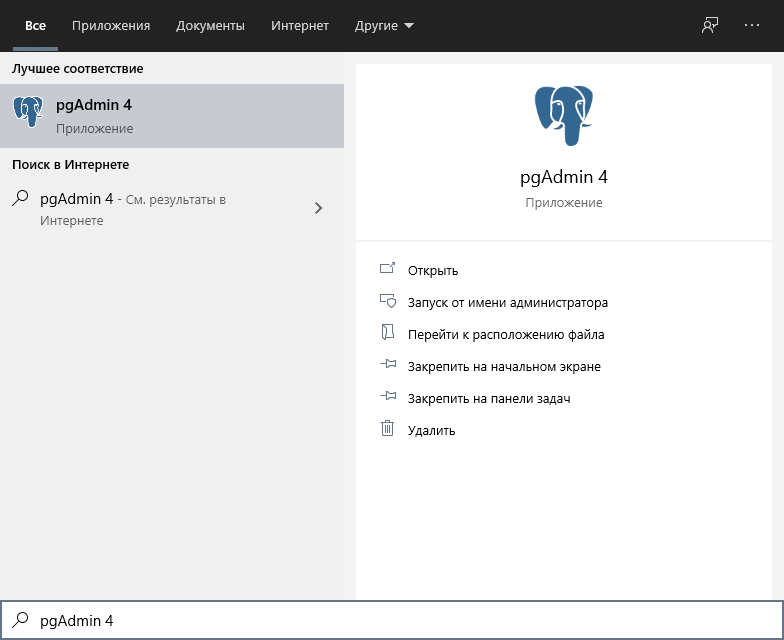


Рисунок 21 – Поиск pgAdmin 4

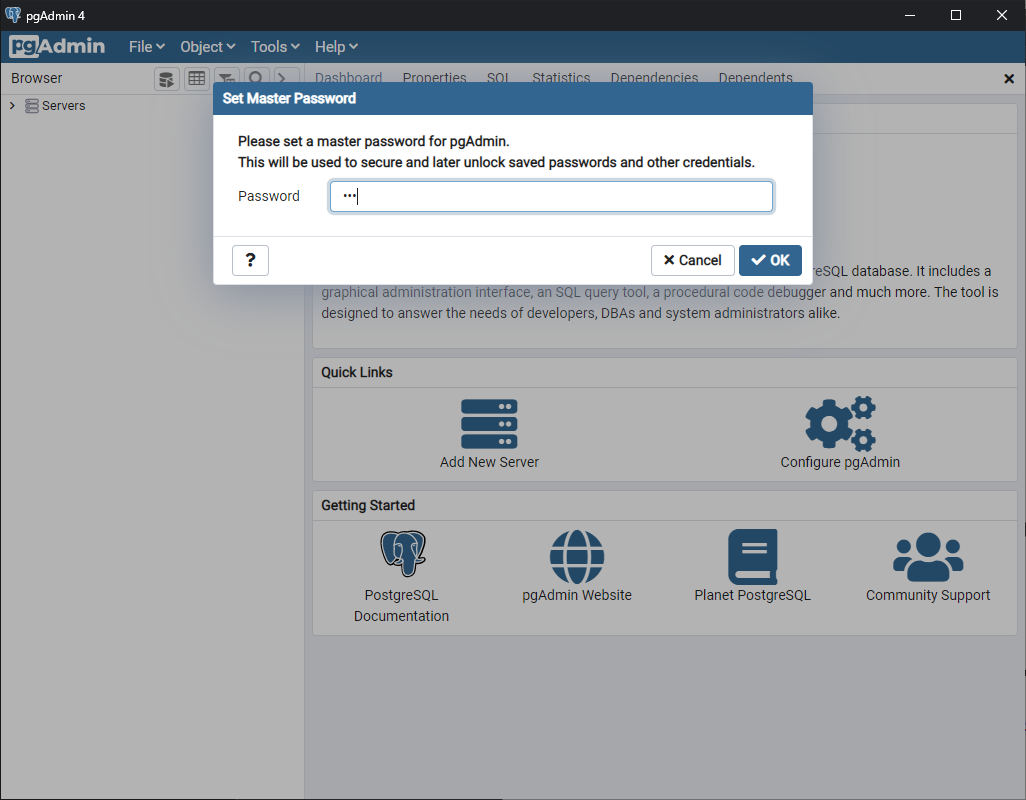


Рисунок 22 – Подключение к СУБД

После подключения к СУБД можно создать новую базу данных кликнув правой кнопкой мыши на Databases (рисунок 23). Создадим базу данных с именем Test. Кликнем на нее правой кнопкой мыши и выберем пункт Query Tool (рисунок 24).

Пропишем в открывшемся окне команду «create language plpython3u» и убедимся, что команда работает (рисунок 25).

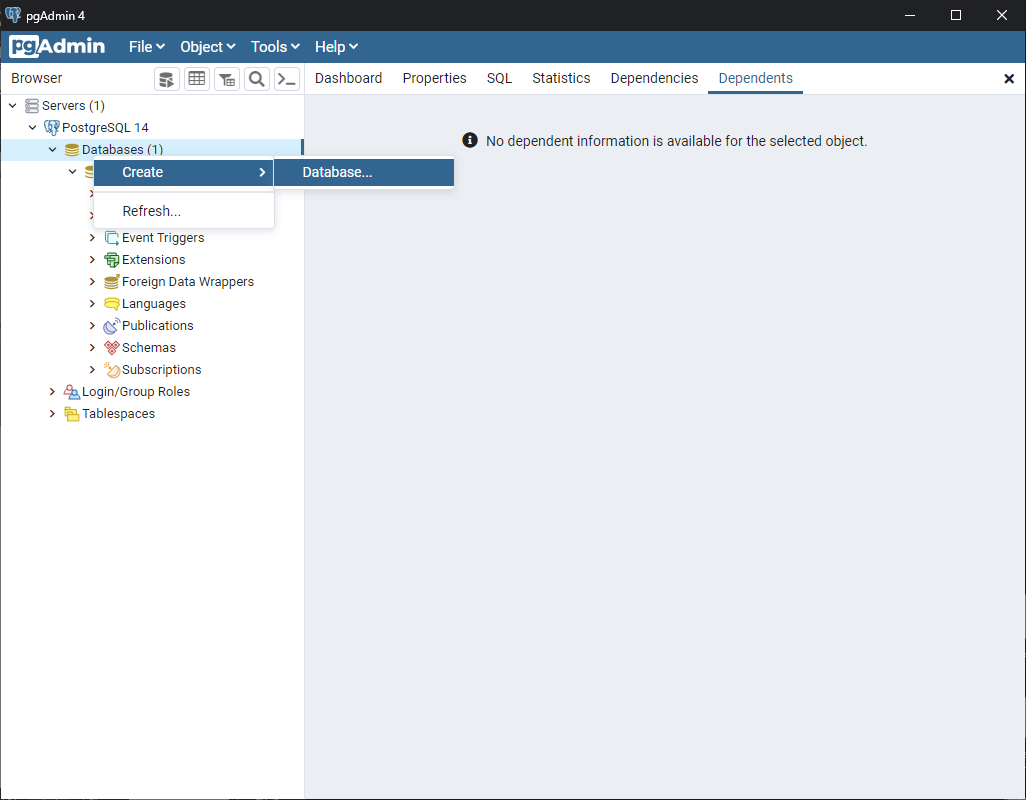


Рисунок 23 – Создание новой базы данных

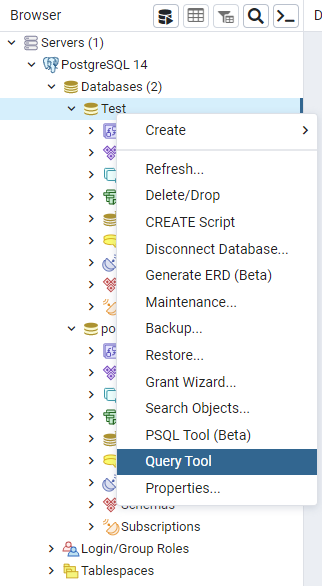


Рисунок 24 – Инструмент создания запросов

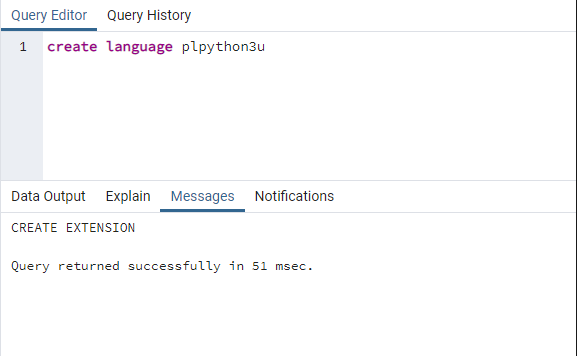


Рисунок 25 – Создание расширения plpython3u

В листинге 1 представлен код запросов, который создает функции на языке python3, а также вызов созданных функций.

Листинг 1 – Код файла pythonFunctions

|  |
| --- |
| -- пример функции на python3  -- возвращает наибольшее из двух чисел  CREATE FUNCTION pymax (a integer, b integer)  RETURNS integer  AS $$  if a > b:  return a  return b  $$ LANGUAGE plpython3u;  -- пример перегрузки функции  -- возвращает наибольшее из трех чисел  CREATE FUNCTION pymax (a integer, b integer, c integer)  RETURNS integer  AS $$  if a > b and a > c:  return a  elif b > a and b > c:  return b  else:  return c  $$ LANGUAGE plpython3u;  -- пример использования стандартного модуля random  CREATE FUNCTION get\_random\_numbers (min integer, max integer)  RETURNS int[]  AS $$  import random  list\_of\_random\_numbers = []  for i in range(10):  list\_of\_random\_numbers.append(random.randint(min, max))  return list\_of\_random\_numbers;  $$ LANGUAGE plpython3u;  -- 2 аргумента  select \* from pymax(111, 50)  -- 3 аргумента  select \* from pymax(1, 3, 2)  -- генерация массива случайных числел  select \* from get\_random\_numbers(1, 100) |

Результаты вызова функций показаны на рисунках 26 - 28.



Рисунок 26 – Результат функции, возвращающей наибольшее из двух чисел



Рисунок 27 – Результат функции, возвращающей наибольшее из трех чисел

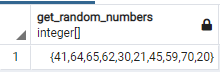


Рисунок 28 – Результат функции, возвращающей массив случайных чисел

В листинге 2 представлен код запросов, который демонстрирует применение механизма наследования.

Листинг 2 – Код файла inherition

|  |
| --- |
| -- города штата  CREATE TABLE cities (  name text,  population float,  elevation int -- высота в футах  );  -- столицы штата  CREATE TABLE capitals (  state char(2) -- штат  ) INHERITS (cities);  /\* добавление записей \*/  insert  into cities ("name", "population", "elevation")  values ('Las Vegas',1232113,2174);  insert  into cities ("name", "population", "elevation")  values ('Mariposa',1342132,1953);  insert  into capitals ("name", "population", "elevation", "state")  values ('Madison',1236745,845,'WI');  -- все города включая столицы  SELECT name, elevation  FROM cities    -- только города  SELECT name, elevation  FROM ONLY cities  -- вывод записей с названием классов  SELECT p.relname, c.name, c.elevation  FROM cities c, pg\_class p  WHERE c.tableoid = p.oid; |

Результаты использования запроса select после создания и наполнения таблиц показаны на рисунках 29 - 31.

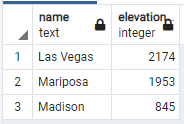


Рисунок 29 – Вывод всех городов включая столицы

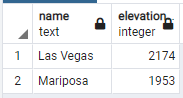


Рисунок 30 – Вывод только городов

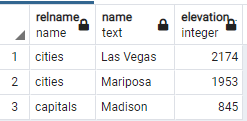


Рисунок 31 – Вывод всех записей с выводом названия таблицы

В листинге 3 представлен код запросов, который демонстрирует использование массивов.

Листинг 3 – Код файла arrays

|  |
| --- |
| CREATE TABLE sal\_emp (  name text,  pay\_by\_quarter integer[], -- плата за квартал  schedule text[][] -- расписание  );  -- добавление массива с фигурными скобками  INSERT INTO sal\_emp  VALUES ('Bill',  '{10000, 10000, 10000, 10000}',  '{{"meeting", "lunch"}, {"training", "presentation"}}');  -- с конструктором ARRAY:  INSERT INTO sal\_emp  VALUES ('Carol',  ARRAY[20000, 25000, 25000, 25000],  ARRAY[['breakfast', 'consulting'], ['meeting', 'lunch']]);  select \* from sal\_emp    -- обращение к элементу массива  SELECT \* FROM sal\_emp WHERE pay\_by\_quarter[1] <> pay\_by\_quarter[2];  -- полная замена массива  UPDATE sal\_emp SET pay\_by\_quarter = '{25000,25000,27000,27000}'  WHERE name = 'Carol';  SELECT \* FROM sal\_emp  WHERE name = 'Carol';  -- замена среза  UPDATE sal\_emp SET pay\_by\_quarter[1:2] = '{27000,27000}'  WHERE name = 'Bill';  SELECT \* FROM sal\_emp  WHERE name = 'Bill'; |

Результаты использования запроса select после создания и наполнения таблиц показаны на рисунках 32 - 35.

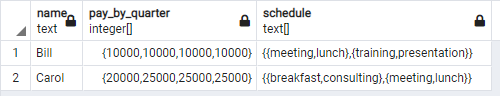


Рисунок 32 – Вывод добавленных записей

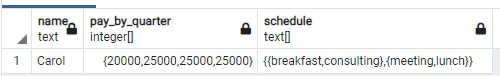


Рисунок 33 – Вывод записей, у которых в поле pay\_by\_quarter  
первый элемент отличается от второго

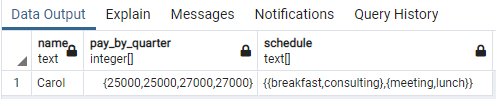


Рисунок 34 – Вывод записей о «Carol» после полной замены массива в поле  
pay\_by\_quarter

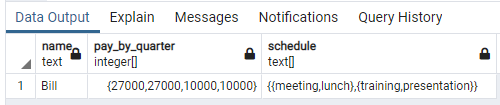


Рисунок 35 – Вывод записей о «Bill» после замены среза [1:2] у массива в поле pay\_by\_quarter

**Список использованных источников**

1. <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>