

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Тихоокеанский государственный университет»

Тимошко А.М., Тусикова А.А.

СБОРКА И НАСТРОЙКА PostgreSQL Server БЕЗ ПРАВ АДМИНИСТРАТОРА

Методические указания к лабораторной работе №1 по дисциплине
«Проектирование приложений баз данных» для студентов
специальности 231000.62 «Программная инженерия»



Хабаровск
2016

Сборка и настройка PostgreSQL Server без прав администратора: методические указания к лабораторной работе №1 по дисциплине «Проектирование приложений баз данных» для студентов специальности 231000.62 «Программная инженерия» / сост. Тимошко А.А., Тусикова А.А. – Хабаровск: 2016. – 9 с.

Методические указания к лабораторной работе №1 по дисциплине «Проектирование приложений баз данных» составлены студентами для помощи другим студентам при выполнении данной работы. В них изложен доступным языком материал для практического применения: сборка, установка и настройка PostgreSQL, создание баз данных, пользователей и подключений.

Хабаровск, 2016

Итак, для данной работы мы будем использовать PostgreSQL версии 9.3, однако порядок и сами действия не должны зависеть от выбранной версии.

Шаг 1. Установка PostgreSQL

Для начала скачайте исходные файлы PostgreSQL и распакуйте в отдельную папку. После того, как вы это сделали, необходимо запустить автоконфигурацию пакета, для этого:

1. Откройте терминал.
2. Перейдите в папку с исходниками при помощи команды **cd**.
Например:
`cd /home/user/postgresql-9.3.0`
3. Создайте папку в домашней директории, в которую будет установлен PostgreSQL. Избегайте кириллицы в пути к данной папке.
4. Запустите `configure` со следующими параметрами:
`-prefix=PATH`
`--without-readline`

Вместо `PATH` укажите путь к созданной папке для установки. В итоге команда должна выглядеть примерно таким образом:

- ```
./configure -prefix=/home/user/POSTGRESQL --without-readline
```
5. `make`
  6. `make install`

## Шаг 2. Запуск и настройка сервера PostgreSQL

После выполнения предыдущего шага вы установили PostgreSQL в выбранную папку. Далее необходимо инициализировать кластер баз данных (сервер) при помощи утилиты **initdb**, которая находится в папке *bin*. В случае данных, используемых на этапе 4 шага № 1, эта папка располагается по адресу: `/home/user/POSTGRESQL/bin`. Данное действие нужно выполнить лишь единожды. Создайте в домашней папке директорию для сервера (системную папку). В терминале перейдите в папку *bin* и выполните команду:

```
./initdb -D /home/user/SERVER -W
```

Необходимо сразу уточнить, что при создании системной папки создаётся и суперпользователь базы данных. Данный пользователь обладает неограниченными правами по администрированию сервера

PostgreSQL. По умолчанию, имя суперпользователя PostgreSQL совпадает с именем пользователя, запустившего команду `initdb`. В приведённом выше случае суперпользователь будет иметь имя `user`. Пароль для суперпользователя генерируется автоматически, что в некоторых случаях является неудобным. Ключ `-W` вынуждает команду `initdb` запросить пароль суперпользователя у оператора.

После этого Вам предложат запустить сервер двумя способами:

```
./postgres -D /home/user/SERVER
```

или

```
./pg_ctl -D /home/user/SERVER -l logfile start
```

В первом случае Вы запускаете сервер в режиме реального времени. В другом случае – в фоновом режиме с ведением log-файла. Следует обратить внимание на содержимое log-файла. В большинстве случаев это помогает найти ошибки запуска сервера. Для работы можно применять любой из вариантов запуска сервера. Если сервер запущен в режиме реального времени, то его системные сообщения будут отображаться в терминале, в котором невозможно проводить никаких действий. Для работы с сервером запустите новый терминала.

Чтобы для запуска команд сервера, постоянно не входить в терминале в папку `bin` установки PostgreSQL, можно заранее установить переменную окружения `PATH`:

```
export PATH=${PATH}:/home/user/POSTGRESQL/bin
```

Если эту команду написать в конце файла `~/.bashrc`, то новое значение `PATH` сохранится между сеансами работы.

Аналогично, можно избежать постоянного указания системной папки PostgreSQL (ключ `-D` вместе с аргументом). Для этого достаточно выполнить команду:

```
export PGDATA=/home/user/SERVER
```

Также как и случае `PATH`, значение переменной `PGDATA` можно сохранить путём редактирования файла `~/.bashrc`.

Чтобы корректно завершить процесс работы сервера введите команду:

```
./pg_ctl -D /home/user/SERVER stop
```

```
alice@alice-X202E: ~/POSTGRESQL/bin
alice@alice-X202E:~/POSTGRESQL/bin$./postgres -D /home/alice/SERVER
LOG: database system was shut down at 2016-02-11 20:56:43 VLAT
LOG: database system is ready to accept connections
LOG: autovacuum launcher started
```

Рис. 1. Сервер запущен в режиме реального времени.

После того, как вы запустите сервер, следует создать саму базу данных. Сделать это можно с помощью **createdb**. При использовании данной команды следует указать название базы данных. Это может выглядеть так:

```
./createdb database
```

Так же для работы с БД необходимо создать пользователя. Сделать это можно 2-мя способами:

1) используя **createuser**:

```
./createuser -S user1
```

Расшифровка ключей:

-S – роль обычного пользователя.

2) используя **psql**.

**psql** позволяет работать в среде PostgreSQL (т.е. использовать запросы). Для запуска утилиты введите строку:

```
./psql database
```

Для создания пользователя запрос выглядит так:

```
CREATE USER user1 WITH password '12345';
```

В обоих примерах создаётся пользователь PostgreSQL с именем user1. Для выхода из среды PostgreSQL, введите “\q”.

После данных манипуляций необходимо настроить сервер для того, чтобы клиент мог подключиться с другого устройства на Ваш сервер. Для этого необходимо изменить два файла, которые находятся в системной папке сервера.

В файле postgresql.conf необходимо раскомментировать строку `#listen_addresses='localhost'` и заменить на `listen_addresses = '*'`.

В нашем случае, в файле `pg_hba.conf` нужно задать настройки для подключения суперпользователя с именем `user` и обычного пользователя с именем `user1` с использованием парольной авторизации.

```
local all user md5
local all user1 md5
host all user 0.0.0.0/0 md5
host all user1 0.0.0.0/0 md5
```

Первый параметр – способ подключения (`local` или `host`).

Второй параметр – имя БД, к которой происходит подключение (`all` – подключение ко всем БД на сервере).

Третий параметр – имя пользователя.

Четвертый параметр – IP-адрес или диапазон IP-адресов для подключения с использования механизма масок. Значение `0.0.0.0/0` означает, что пользователь может подключаться к серверу с любого IP-адреса. Ограниченный диапазон адресов можно задать так: `192.168.1.0/24`. Для локальных подключений параметр опускается.

Пятый параметр – способ авторизации. Существует несколько значений параметра:

`trust` — доступ к базе может получить кто угодно под любым именем, имеющий с ней соединение.

`reject` — отклонить безоговорочно! Это подходит для фильтрации определенных IP адресов.

`md5` — требует обязательного ввода пароля для защиты от несанкционированного доступа.

После настройки можно перезапустить сервер:

```
./pg_ctl -D /home/user/SERVER restart
```

В качестве альтернативы, можно заставить сервер прочитать новое содержимое файла `pg_hba.conf`:

```
./pg_ctl -D /home/user/SERVER reload
```

### Шаг 3. Подключение к серверу PostgreSQL

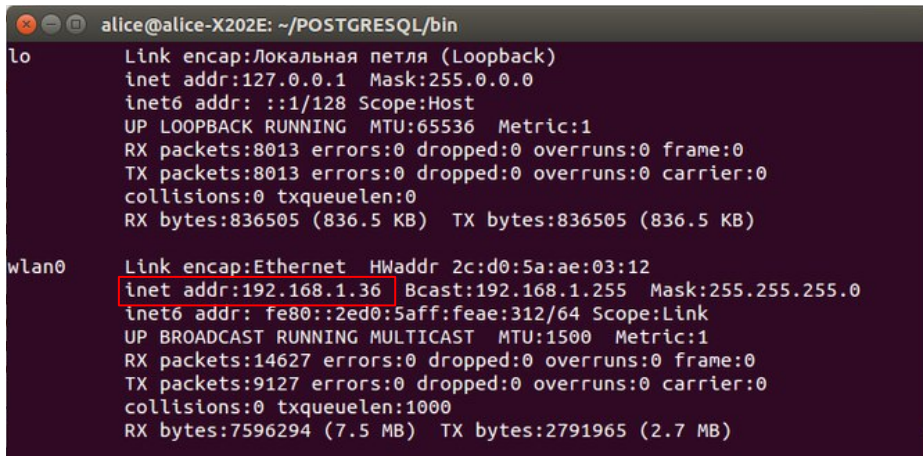
Для подключения к серверу локально достаточно выполнить команду:

```
./psql database user
```

Для подключения к серверу с другого компьютера необходимо добавить в команду IP-адрес сервера:

```
./psql -h 192.168.1.36 database user
```

Узнать IP-адрес сервера можно через окно «Сведения о соединении» или выполнив команду **ifconfig** на компьютере, играющего роль сервера.



```
alice@alice-X202E: ~/POSTGRESQL/bin
lo Link encap:Локальная петля (Loopback)
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:8013 errors:0 dropped:0 overruns:0 frame:0
 TX packets:8013 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:836505 (836.5 KB) TX bytes:836505 (836.5 KB)

wlan0 Link encap:Ethernet HWaddr 2c:d0:5a:ae:03:12
 inet addr:192.168.1.36 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::2ed0:5aff:feae:312/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:14627 errors:0 dropped:0 overruns:0 frame:0
 TX packets:9127 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:7596294 (7.5 MB) TX bytes:2791965 (2.7 MB)
```

Рис. 2. Результат команды ifconfig.

К Вашему серверу можно подключиться со смартфона на OS Android при помощи приложения SQL Buddy.

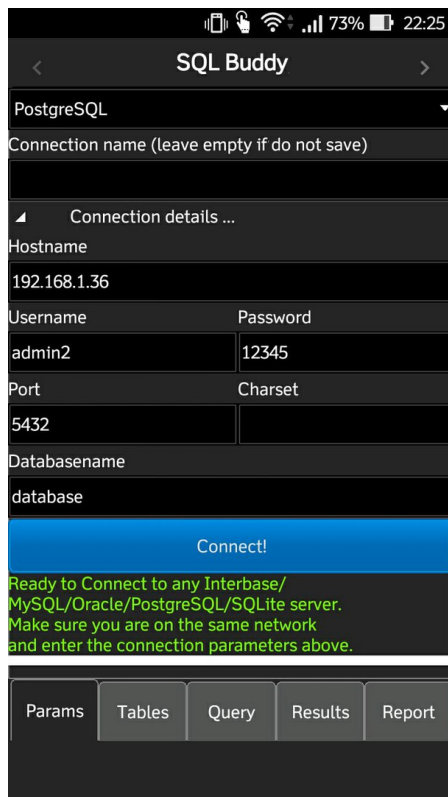


Рис. 3. Настройки для подключения к серверу через приложение SQL Buddy на OS Android.

#### Шаг 4. Создание простейших запросов в PostgreSQL

Обратите внимание, что в отличие от некоторых других СУБД, в PostgreSQL нет столбцов со свойством `auto_increment`. Вместо этого используются последовательности (sequences). На данный момент достаточно знать, что с помощью функции **nextval** мы можем получать уникальные числа для заданной последовательности. Примечание: начиная с версии 9 СУБД PostgreSQL, в ней реализован тип `serial`, который полностью выполняет функции свойства `auto_increment`. Важно отметить, что в PostgreSQL по умолчанию имена таблиц и столбцов приводятся к нижнему регистру. Если это поведение нежелательно, можно воспользоваться двойными кавычками.

Создайте таблицу `humans` с тремя столбцами: `id`, `name`, `phone`:

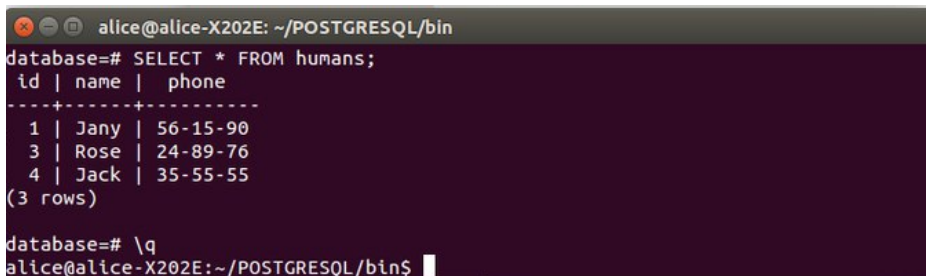


```
CREATE SEQUENCE ids;
CREATE TABLE humans (id INTEGER PRIMARY KEY DEFAULT
NEXTVAL('ids'), name VARCHAR, phone VARCHAR);
```

Прописав в качестве значения по умолчанию для поля `id` таблицы `humans` значение `NEXTVAL('ids')`, мы добились того же эффекта, что дает `auto_increment`. При добавлении новых записей в таблицу мы можем не указывать `id`, потому что уникальный `id` будет сгенерирован автоматически. Несколько таблиц могут использовать одну и ту же последовательность. Таким образом мы сможем гарантировать, что значения некоторых полей у этих таблиц не пересекаются. В этом смысле последовательности более гибки, чем `auto_increment`.

Теперь заполним таблицу `humans` значениями:

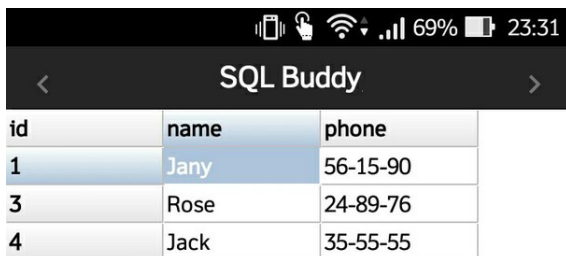
```
INSERT INTO humans(name, phone) VALUES ('Jany','56-15-90');
INSERT INTO humans(name, phone) VALUES ('Rose','24-89-76');
INSERT INTO humans(name, phone) VALUES ('Jack','35-55-55');
```



```
alice@alice-X202E: ~/POSTGRESQL/bin
database=# SELECT * FROM humans;
 id | name | phone
----+-----+-----
 1 | Jany | 56-15-90
 3 | Rose | 24-89-76
 4 | Jack | 35-55-55
(3 rows)

database=# \q
alice@alice-X202E:~/POSTGRESQL/bin$
```

Рис. 4. Результат запроса `SELECT` в терминале.



| id | name | phone    |
|----|------|----------|
| 1  | Jany | 56-15-90 |
| 3  | Rose | 24-89-76 |
| 4  | Jack | 35-55-55 |

Рис. 5. Результат запроса `SELECT` в SQL Buddy.

## **Используемые источники**

1. <https://habrahabr.ru/post/168601/>
2. <http://eax.me/postgresql-install/>
3. <http://www.proft.com.ua/2010/05/6/ustanovka-postgresql-pod-ubuntu-centos/>
4. <http://unixhelp.org/открыть-удаленный-доступ-к-postgresql.html>