

изменить root пароль в PostgreSQL

```
$ *psql postgres postgres
Password: (oldpassword)
# ALTER USER postgres WITH PASSWORD 'tmppassword';
$ *psql postgres postgres
Password: (tmppassword)
```

Create user PostgreSQL

Метод 1: Создаем пользователя в через PSQL шелл, командой CREATE USER.

```
# CREATE USER ramesh WITH password 'tmppassword';
CREATE ROLE
```

Метод 2: Создаем пользователя в через шелл команду create-user.

```
$ */createuser sathiya
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
CREATE ROLE
```

Help

Команда \? отобразит строку помощи для команда PSQL. \h CREATE покажет хелп для всех команд который начинаются с CREATE.

```
# \?
# \h CREATE
# \h CREATE INDEX
```

список всех таблиц в базе

```
# \d
Для пустой базы вы получите сообщение "No relations found."
```

psql команды

```
$ su - postgres
psql -l - список баз данных.
psql -d dbname - подключение к БД dbname.
psql -f file.sql - выполнение команд из файла file.sql.
psql -U postgres -d dbname -c "CREATE TABLE test(some_id serial PRIMARY KEY, some_text text);" - выполнение команды в базе dbname.
psql -d dbname -H -c "SELECT * FROM test" -o test.html - вывод результата запроса в html-файл.
```

Все команды запускаются под пользователем postgres (postgres-ql-суперпользователь)

Backup

В PostgreSQL есть две утилиты для бекапа pg_dump и pg_dumpall. pg_dump используется для бекапа одной базы, pg_dumpall для бекапа всех баз и сервера в целом (необходимо запускать под postgresql-суперпользователем).

Создание бекапа базы mydb, в сжатом виде

```
pg_dump -h localhost -p 5432 -U someuser -F c -b -v -f mydb.backup mydb
```

Создание бекапа базы mydb, в виде обычного текстового файла, включая команду для создания БД

```
pg_dump -h localhost -p 5432 -U someuser -C -F p -b -v -f mydb.backup mydb
```

Создание бекапа базы mydb, в сжатом виде, с таблицами которые содержат в имени payments

```
pg_dump -h localhost -p 5432 -U someuser -F c -b -v -t payments -f payment_tables.backup mydb
```

Дамп данных только одной, конкретной таблицы. Если нужно создать резервную копию нескольких таблиц, то имена этих таблиц перечисляются с помощью ключа -t для каждой таблицы.

```
pg_dump -a -t table_name -f file_name database_name
```

Создание резервной копии с сжатием в gz

```
pg_dump -h localhost -O -F p -c -U postgres mydb | gzip -c > mydb.gz
```

Список наиболее часто используемых опций:

-h host - хост, если не указан то используется localhost или значение из переменной окружения PGHOST.
-p port - порт, если не указан то используется 5432 или значение из переменной окружения PGPORT.

-u - пользователь, если не указан то используется текущий пользователь, также значение можно указать в переменной окружения PGUSER.

-a, --data-only - дамп только данных, по-умолчанию сохраняются данные и схема.

-b - включать в дамп большие объекты (blog'и).

-s, --schema-only - дамп только схемы.

-C, --create - добавляет команду для создания БД.

-c - добавляет команды для удаления (drop) объектов (таблиц, видов и т.д.).

-O - не добавлять команды для установки владельца объекта (таблиц, видов и т.д.).

-F, --format {c|t|p} - выходной формат дампа, custom, tar, или plain text.

-t, --table=TABLE - указываем определенную таблицу для дампа.

-v, --verbose - вывод подробной информации.

-D, --attribute-inserts - дамп используя команду INSERT с списком имен свойств.

Бекап всех баз данных используя команду pg_dumpall.

Check server status

Linux:

\$ status

Password:

pg_ctl: server is running (PID: 6171)

[Замечание: Это сообщение говорит о том, что сервер запущен и работает нормально]

\$ status

Password:

pg_ctl: no server running

[Замечание: Это сообщение говорит о том, что сервер не запущен]

Mac OS:

ps auxwww | grep postgres

Create database

Linux

Метод 1: Создаем базу через PSQL шелл, с помощью команды CREATE DATABASE.

```
# CREATE DATABASE mydb WITH OWNER ramesh;
CREATE DATABASE
```

Метод 2: Используем команду createdb.

```
$ */createdb mydb -O ramesh
```

```
CREATE DATABASE
```

Mac OS

```
createdb -Otunnelsup -Eutf8 mysite_development
```

[**-O** означает что пользователь является владельцем базы данных]

удалить базу в PostgreSQL

```
# \l
```

List of databases

```
Name | Owner | Encoding
```

```
-----+-----+-----
```

```
backup | postgres | UTF8
```

```
mydb | ramesh | UTF8
```

```
postgres | postgres | UTF8
```

```
template0 | postgres | UTF8
```

```
template1 | postgres | UTF8
```

```
# DROP DATABASE mydb;
```

```
DROP DATABASE
```

psql команды

\c dbname - подключение к БД dbname.

\l - список баз данных.

\dt - список всех таблиц.

\d table - структура таблицы table.

\du - список всех пользователей и их привилегий.

\dt+ - список всех таблиц с описанием.

\dt s - список всех таблиц, содержащих s в имени.

\i FILE - выполнить команды из файла FILE.

\o FILE - сохранить результат запроса в файл FILE.

\a - переключение между режимами вывода: с/без выравнивания.

Recovery

В PostgreSQL есть две утилиты для восстановления базы из бекапа.

psql - восстановление бекапов, которые хранятся в обычном текстовом файле (plain text);

pg_restore - восстановление сжатых бекапов (tar);

Восстановление всего бекапа с игнорированием ошибок

```
psql -h localhost -U someuser -d dbname -f mydb.sql
```

Восстановление всего бекапа с остановкой на первой ошибке

```
psql -h localhost -U someuser --set ON_ERROR_STOP=on -f
```

```
mydb.sql
```

Для восстановления из tar-архива нам понадобится сначала

создать базу с помощью CREATE DATABASE mydb; (если при

создании бекапа не была указана опция -C) и восстановить

```
pg_restore --dbname=mydb --jobs=4 --verbose mydb.backup
```

Восстановление резервной копии БД, сжатой gz

```
gunzip mydb.gz
```

```
psql -U postgres -d mydb -f mydb
```

Начиная с версии 9.2 можно восстановить только структуру

таблиц с помощью опции --section

```
# создаем БД
```

```
CREATE DATABASE mydb2;
```

```
# восстанавливаем
```

```
pg_restore --dbname=mydb2 --section=pre-data --jobs=4 mydb.b-
ackup
```

Run/Stop/Restart

```
# service postgresql stop
```

Stopping PostgreSQL: server stopped

```
ok
```

```
# service postgresql start
```

Starting PostgreSQL:

```
ok
```



Run/Stop/Restart (cont)

```
# service postgresql restart
Restarting PostgreSQL: server stopped
ok
```

Other (cont)

```
Запуск
pgcli -U postgres -W dbname
```

список всех баз в PostgreSQL

```
# \l
List of databases
Name | Owner | Encoding
-----+-----+-----
backup | postgres | UTF8
mydb | ramesh | UTF8
postgres | postgres | UTF8
template0 | postgres | UTF8
template1 | postgres | UTF8
```

время выполнения запроса

```
# \timing — после выполнения данной команды каждый послед-
ующий запрос будет показывать время выполнения.
# \timing
Timing is on.
# SELECT * from pg_catalog.pg_attribute;
Time: 9.583 ms
```

файл истории postgresSQL

```
$ cat ~/.psql_history
alter user postgres with password 'tmppassword';
\h alter user
select version();
create user ramesh with password 'tmppassword';
\timing
select * from pg_catalog.pg_attribute;
```

Other

Очищение таблицы
Очищение таблицы tablename и обнуление счетчика с ID.
TRUNCATE TABLE tablename RESTART IDENTITY CASCADE;
CASCADE нужен на случай если tablename связана с другой таблицей.

Удаление NULL у поля
ALTER TABLE movies ALTER COLUMN year DROP NOT NULL;

Утилиты
pgcli утилита командной строки с авто-дополнениям и подсветкой синтаксиса.

Установка
pip install pgcli

