

Обзор базового инструментария

Установка и управление, psql

**16**

|  |
| --- |
| Темы |
| Варианты установки PostgreSQL  Управление сервером  Журнал сообщений сервера  Настройка параметров конфигурации  Использование psql  2 |

Начнем с основных понятий.

PostgreSQL — программа, которая относится к классу *систем управления базами данных*.

Какие бывают СУБД?

Классификация. Примеры.

Когда эта программа выполняется, мы называем ее *сервером*

PostgreSQL или *экземпляром сервера*. Пока сервер представляется для нас «черным ящиком», но постепенно мы познакомимся с тем, как он устроен.

|  |
| --- |
| Кластер БД, сервер, клиент |
| PostgreSQL  база  данных    кластер баз данных  клиент  д  р  а  й  в  е  р  подключение  3 |

Данные, которыми управляет PostgreSQL, хранятся в *базах данных*. Один экземпляр PostgreSQL одновременно работает с несколькими базами данных. Этот набор баз данных называется *кластером баз данных*. Подробнее мы будем говорить о базах данных в теме «Организация данных. Логическая структура».

С сервером взаимодействуют клиенты — внешние приложения, которые могут подключаться к одной из баз сервера и посылать *запросы* для выполнения.

Итак: кластер баз данных — это данные в файлах; сервер или экземпляр сервера — программа, управляющая кластером баз данных, а клиент — программа, позволяющая «общаться» с сервером.

Предпочтительный вариант установки PostgreSQL — использование пакетных менеджеров (таких, как apt или rpm) и готовых пакетов. В этом случае получается понятная, поддерживаемая и легко обновляемая установка. Пакеты существуют для большинства операционных систем.

Другой вариант — самостоятельная сборка PostgreSQL из исходных кодов. Это может понадобиться для установки нестандартных значений параметров или при использовании не популярной платформы. Готовые пакеты и исходные коды: http://www.postgresql.org/download/

|  |
| --- |
| Установка |
| Варианты  готовые пакеты (предпочтительный способ) установка из исходных кодов без установки — облачные сервисы  Расширения  дополнительный функционал устанавливаются отдельно  в поставке с сервером — модули и программы (~50 штук)  4 |

Кроме того, можно использовать готовые облачные решения, что позволяет обойтись вообще без установки. Такую возможность дают многие ведущие зарубежные (Amazon RDS, Google Cloud SQL, Microsoft Azure) и отечественные (Yandex Cloud, Облако Mail.ru) платформы.

В курсе мы будем использовать виртуальную машину с ОС Xubuntu 22 и сервер PostgreSQL 16, установленный из пакета для этой ОС. В этом случае сразу настраивается автоматический запуск и останов сервера PostgreSQL при запуске и останове операционной системы.

Для PostgreSQL существует большое количество расширений, которые подключают новый функционал к СУБД «на лету», без изменения ядра системы. В состав дистрибутива входит 50 расширений.

https://postgrespro.ru/docs/postgresql/16/contrib https://postgrespro.ru/docs/postgresql/16/contrib-prog

Список доступных расширений и статус их установки можно посмотреть в представлении pg\_available\_extensions.

К основным операциям управления сервером относятся начальная инициализация кластера баз данных, запуск и останов сервера, обновление конфигурации и некоторые другие. Для выполнения этих действий предназначена утилита pg\_ctl, входящая в состав PostgreSQL. В пакетном дистрибутиве для Ubuntu доступ к утилите pg\_ctl осуществляется не напрямую, а через специальную обертку pg\_ctlcluster. Справку по использованию pg\_ctlcluster можно получить командой:

|  |
| --- |
| Управление сервером |
| Утилита для управления pg\_ctlcluster pg\_ctl  Основные задачи  запуск сервера останов сервера  обновление параметров конфигурации  5 |

$ man pg\_ctlcluster

Также можно получить информацию об установленных кластерах и их текущем состоянии при помощи команд:

$ pg\_lsclusters

$ pg\_ctlcluster status

Более подробная информация об управлении сервером для администраторов баз данных:

https://postgrespro.ru/docs/postgresql/16/app-pg-ctl https://postgrespro.ru/docs/postgresql/16/runtime

# Установка и управление

В виртуальной машине курса установка выполнена из пакета. Каталог установки PostgreSQL:

## student$ ls -l /usr/lib/postgresql/16

total 8

drwxr-xr-x 2 root root 4096 июн 21 16:28 bin

drwxr-xr-x 4 root root 4096 июл 3 15:56 lib

Кто владелец сервера?

Владелец ПО сервера — пользователь root.

Что означаюn буквы drwxr-xr-x?

**drwxr**-**xr**-**x** (755) — Все могут читать этот каталог, но его содержимое может изменить только владелец.

- (rw-) (rw-) (r--)

| | |

4+2+0 4+2+0 4+0+0

<http://www.rhd.ru/docs/manuals/enterprise/RHEL-AS-2.1-Manual/getting-started-guide/s1-navigating-chmodnum.html>

Кластер баз данных автоматически инициализируется при установке из пакета и находится в каталоге **/var/lib/postgresql/16/main.**

В последующих темах мы будем ссылаться на этот каталог как **PGDATA**, по имени переменной ОС, которую можно установить для использования в некоторых утилитах сервера.

Владельцем каталога является пользователь postgres. Вот содержимое каталога:

## student$ sudo ls -l /var/lib/postgresql/16/main

**total 96**

**drwx------ 1 postgres postgres 4096 июл 5 16:17 base**

**drwx------ 1 postgres postgres 4096 июл 5 16:17 global**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_commit\_ts**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_dynshmem**

**drwx------ 4 postgres postgres 4096 июл 3 15:57 pg\_logical**

**drwx------ 4 postgres postgres 4096 июл 3 15:57 pg\_multixact**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_notify**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_replslot**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_serial**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_snapshots**

**drwx------ 1 postgres postgres 4096 июл 5 16:17 pg\_stat**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_stat\_tmp**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_subtrans**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_tblspc**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_twophase**

**-rw------- 1 postgres postgres 3 июл 3 15:57 PG\_VERSION**

**drwx------ 1 postgres postgres 4096 июл 3 15:57 pg\_wal**

**drwx------ 2 postgres postgres 4096 июл 3 15:57 pg\_xact**

**-rw------- 1 postgres postgres 88 июл 3 15:57 postgresql.auto.conf**

**-rw------- 1 postgres postgres 130 июл 5 16:17 postmaster.opts**

**-rw------- 1 postgres postgres 108 июл 5 16:17 postmaster.pid**

При установке из пакета в настройки запуска ОС добавляется автоматический запуск PostgreSQL. Поэтому после загрузки операционной системы отдельно стартовать PostgreSQL не нужно.

Можно явным образом управлять сервером с помощью следующих команд, которые выдаются от имени привилегированного пользователя ОС через sudo:

Остановить сервер:

**student$ sudo pg\_ctlcluster 16 main stop**

Запустить сервер:

**student$ sudo pg\_ctlcluster 16 main start**

Перезапустить:

**student$ sudo pg\_ctlcluster 16 main restart** Обновить конфигурацию:

**student$ sudo pg\_ctlcluster 16 main reload** Получить информацию о сервере:

## student$ sudo pg\_ctlcluster 16 main status

**pg\_ctl: server is running (PID: 3440)**

**/usr/lib/postgresql/16/bin/postgres "-D" "/var/lib/postgresql/16/main" "-c" "config\_file=/etc/postgresql/16/main/postgresql.conf"**

Список установленных экземпляров (можно без sudo):

## student$ pg\_lsclusters

**Ver Cluster Port Status Owner Data directory Log file 16**

**main 5432 online postgres /var/lib/postgresql/16/main /var/log/postgresql/postgresql-16-main.log**

|  |
| --- |
| Журнал сообщений |
| В журнал записываются  служебные сообщения сервера сообщения пользовательских сеансов сообщения приложений  Настройка журнала  расположение формат записей  какие события регистрировать  7 |

Информация о ходе работы СУБД записывается в журнал сообщений сервера. Сюда попадают сведения о запуске и останове сервера, различная служебная информация, в том числе сообщения о возникающих проблемах.

Также сюда могут выводиться сообщения о выполняющихся командах и времени их работы, о возникающих блокировках и тому подобное. Это позволяет выполнять трассировку пользовательских сеансов.

Разработчики приложений могут формировать и записывать в журнал сервера свои собственные сообщения.

Настройки PostgreSQL позволяют гибко определять, какие именно сообщения и в каком формате должны попадать в журнал сервера.

Например, вывод в форматах csv и json удобен для автоматизации анализа журнала. https://postgrespro.ru/docs/postgresql/16/runtime-config-logging

Журнал сообщений сервера

Журнал сообщений сервера находится здесь:

## student$ ls -l /var/log/postgresql/postgresql-16-main.log

-rw-r----- 1 postgres adm 5832 июл 5 16:17 /var/log/postgresql/postgresql-16-main.log

Как посмотреть конец журнала (последние 10-записей):

## student$ tail -n 10 /var/log/postgresql/postgresql-16-main.log

2024-07-05 16:17:19.409 MSK [616] LOG: background worker "logical replication launcher"

(PID 674) exited with exit code 1

2024-07-05 16:17:19.409 MSK [669] LOG: shutting down

2024-07-05 16:17:19.419 MSK [669] LOG: checkpoint starting: shutdown immediate

2024-07-05 16:17:19.489 MSK [669] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0

WAL file(s) added, 0 removed, 0 recycled; write=0.021 s, sync=0.010 s, total=0.081 s; sync files=2, longest=0.005 s, average=0.005 s; distance=0 kB, estimate=0 kB; lsn=0/1956878, redo lsn=0/1956878

2024-07-05 16:17:19.495 MSK [616] LOG: database system is shut down

2024-07-05 16:17:20.100 MSK [3440] LOG: starting PostgreSQL 16.3 (Ubuntu 16.3-1.pgdg22.04+1) on x86\_64-pc-linux-gnu, compiled by gcc (Ubuntu

11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit

2024-07-05 16:17:20.101 MSK [3440] LOG: listening on IPv4 address "127.0.0.1", port 5432

2024-07-05 16:17:20.114 MSK [3440] LOG: listening on Unix socket

"/var/run/postgresql/.s.PGSQL.5432"

2024-07-05 16:17:20.162 MSK [3443] LOG: database system was shut down at 2024-07-03 15:57:35 MSK

2024-07-05 16:17:20.186 MSK [3440] LOG: database system is ready to accept connections

|  |
| --- |
| Параметры конфигурации |
| Для всего экземпляра  основной файл параметров — postgresql.conf  ALTER SYSTEM — postgresql.auto.conf  Для текущего сеанса  SET/RESET set\_config()  Просмотр текущего значения  SHOW  current\_setting() pg\_settings  9 |

Сервер PostgreSQL настраивается с помощью разнообразных параметров конфигурации, которые позволяют управлять ресурсами, настраивать служебные процессы и пользовательские сеансы, управлять журналом сервера и решать многие другие задачи. Поэтому нужно знать, как проверить текущие значения параметров и установить новые.

Настройки всего сервера обычно задаются в конфигурационных файлах. Основной конфигурационный файл — postgresql.conf, он редактируется вручную. Второй конфигурационный файл — postgresql.auto.conf — предназначен для изменения специальной командой ALTER SYSTEM. Параметры, установленные через ALTER SYSTEM, имеют приоритет над параметрами в postgresql.conf.

Директивы включения файлов и каталогов include и include\_dir позволяют разделять сложные файлы postgresql.conf на части. Это может быть удобно, например, при управлении несколькими серверами с похожими конфигурациями.

Большинство параметров конфигурации допускает изменение значений в пользовательских сеансах прямо во время выполнения. Помимо системных, можно определять и пользовательские параметры и работать с ними с помощью этих же команд и функций.

Варианты установки и управления параметрами:

https://postgrespro.ru/docs/postgresql/16/config-setting

Текущие значения параметров доступны в представлении pg\_settings:

https://postgrespro.ru/docs/postgresql/16/view-pg-settings

# Параметры конфигурации

Основной файл конфигурации postgresql.conf расположен в этом каталоге:

## student$ ls -l /etc/postgresql/16/main

total 60 drwxr-xr-x 2 postgres postgres 4096 июл 3 15:57 conf.d

-rw-r--r-- 1 postgres postgres 315 июл 3 15:57 environment

-rw-r--r-- 1 postgres postgres 143 июл 3 15:57 pg\_ctl.conf

-rw-r----- 1 postgres postgres 5743 июл 3 15:57 pg\_hba.conf

-rw-r----- 1 postgres postgres 2640 июл 3 15:57 pg\_ident.conf

-rw-r--r-- 1 postgres postgres 29960 июл 3 15:57 postgresql.conf

-rw-r--r-- 1 postgres postgres 317 июл 3 15:57 start.conf

Здесь же находятся и другие конфигурационные файлы.

Попробуйте вручную посмотреть его содержимое

cat /etc/postgresql/16/main/postgresql.conf

Как найти файл postgresql.auto.conf

**sudo cat /var/lib/postgresql/16/main/postgresql.auto.conf**

Почему не получается?

\c postgres

Как посмотреть все настройки ?

**select \* from pg\_settings;**

**Что умеет делать select?**

**select (5\*5+20)::text as калькулятор;**

**Почему не работает?**

**select work\_mem from pg\_settings;**

**Как быть?**

**SELECT setting FROM pg\_settings WHERE name = 'work\_mem';**

**setting**

**---------**

**4096**

**(1 row)**

Проверим значение параметра work\_mem:

## => SHOW work\_mem;

work\_mem ----------

4MB

(1 row)

Параметр work\_mem задает объем памяти, который будет использоваться для внутренних операций сортировки и размещения хеш-таблиц, прежде чем будут задействованы временные файлы на диске.

4MB — это значение по умолчанию и оно слишком мало. Допустим, мы хотим увеличить его до 16MB для всего экземпляра. Для этого есть различные пути.

Во-первых, можно внести изменение в postgresql.conf, раскомментировав и изменив строку, где определяется параметр:

## student$ grep '#work\_mem' /etc/postgresql/16/main/postgresql.conf

#work\_mem = 4MB # min 64kB

Во-вторых, можно поместить определение параметра в файл с суффиксом .conf в каталоге

/etc/postgresql/16/main/conf.d или в пользовательский файл конфигурации, местоположение которого следует задать в параметре include основного конфигурационного файла postgresql.conf.

В-третьих, можно изменить значение параметра с помощью команды SQL — что мы и сделаем:

**=> ALTER SYSTEM SET work\_mem TO '16MB';**

ALTER SYSTEM

Такое изменение попадает не в postgresql.conf, а в файл postgresql.auto.conf, который находится в каталоге **PGDATA**:

## student$ sudo cat /var/lib/postgresql/16/main/postgresql.auto.conf

# Do not edit this file manually!

# It will be overwritten by the ALTER SYSTEM command. work\_mem = '16MB'

Чтобы изменение вступило в силу, нужно перечитать конфигурационные файлы. Для этого можно воспользоваться pg\_ctlcluster, либо использовать **функцию** SQL:

**=> SELECT pg\_reload\_conf();**

pg\_reload\_conf --------------- t

(1 row)

Убедимся, что новое значение параметра применилось. Кроме команды SHOW, можно сделать это таким образом:

**=> SELECT current\_setting('work\_mem');**

current\_setting -----------------

16MB

(1 row)

Что такое **current\_setting? Что под капотом**

**\df current\_setting;**

SELECT current\_setting('port')

Чтобы восстановить значение параметра по умолчанию, достаточно вместо SET использовать команду RESET (и, конечно, перечитать конфигурационные файлы):

**=> ALTER SYSTEM RESET work\_mem;**

ALTER SYSTEM

**=> SELECT pg\_reload\_conf();**

pg\_reload\_conf --------------- t

(1 row)

Большинству параметров можно установить новое значение для текущего сеанса прямо во время его выполнения. Например, если мы собираемся выполнить запрос, сортирующий большой объем данных, то для сеанса можно увеличить значение work\_mem:

## => SET work\_mem = '64MB';

## SET

Новое значение действует только в текущем сеансе или даже в текущей транзакции (при указании SET LOCAL).

Еще один способ проверить текущее значение — выполнить запрос к представлению:

**=> SELECT name, setting, unit FROM pg\_settings WHERE name = 'work\_mem';**

name | setting | unit ----------+---------+----- work\_mem | 65536 | kB (1 row)

Можно увидеть значение параметра и с помощью команды \dconfig:

## => \dconfig work\_mem

List of configuration parameters

Parameter | Value -----------+------ work\_mem | 64MB (1 row)

|  |
| --- |
| Клиент psql |
| Терминальный клиент для работы с PostgreSQL  Поставляется вместе с СУБД  Используется администраторами и разработчиками для интерактивной работы и выполнения скриптов  11 |

Для работы с СУБД PostgreSQL существуют различные сторонние инструменты, рассмотрение которых не входит в рамки курса.

В курсе мы будем использовать терминальный клиент psql:

1. psql — это единственный клиент, поставляемый вместе с СУБД.
2. Навыки работы c psql пригодятся разработчикам и администраторам вне зависимости от того, с каким инструментом они будут работать.

Для интерактивной работы в psql встроена поддержка readline, программ постраничного просмотра результатов запросов (таких, как less и pspg), а также подключения внешних редакторов. Возможности psql позволяют взаимодействовать с ОС, просматривать содержимое системного каталога, создавать скрипты для автоматизации повторяющихся задач. https://postgrespro.ru/docs/postgresql/16/app-psql

## Подключение

При запуске psql нужно указать параметры подключения. К обязательным параметрам относятся:

имя базы данных, по умолчанию совпадает с именем пользователя; имя пользователя (роль), по умолчанию совпадает с именем пользователя ОС; узел (host), по умолчанию — локальное соединение; порт, по умолчанию — обычно 5432.

Параметры указываются так:

**student$ psql -d база -U роль -h узел -p порт**

**Как подключиться к базе данных student?**

psql -d postgres -U student -h localhost -p 5432

Настройки, сделанные в виртуальной машине курса, позволяют подключаться к PostgreSQL без указания параметров:

**student$ psql**

Проверим текущее подключение:

### => \conninfo

You are connected to database "student" as user "student" via socket in "/var/run/postgresql" at port "5432".

Команда \connect выполняет новое подключение, не покидая psql. Ее можно сократить до \c. Мы будем указывать необязательную часть имени команды в квадратных скобках: \c[onnect].

## Справочная информация

Справку по psql можно получить не только в документации, но и прямо в системе. Команда **student$ psql --help** выдает справку по запуску. А если PostgreSQL устанавливался с документацией, то справочное руководство можно получить командой **student$ man psql**

Утилита psql умеет выполнять команды SQL и свои собственные команды, которые начинаются с обратной косой черты, как \conninfo. Команды psql всегда однострочные — в отличие от команд SQL.

Внутри psql есть возможность получить список и краткое описание его собственных команд:

\? выдает список команд psql,

\h[elp] выдает список команд SQL, которые поддерживает сервер, а также синтаксис

конкретной команды SQL.

## Форматирование вывода

Клиент psql умеет выводить результаты запросов в разных форматах:

формат с выравниванием значений; формат без выравнивания; расширенный формат.

Формат с выравниванием используется по умолчанию:

**=> SELECT name, setting, unit FROM pg\_settings LIMIT 7;**

name | setting | unit

----------------------------+------------+------

allow\_in\_place\_tablespaces | off |

allow\_system\_table\_mods | off |

application\_name | psql |

archive\_cleanup\_command | |

archive\_command | (disabled) |

archive\_library | |

archive\_mode | off |

(7 rows)

Ширина столбцов выровнена по значениям. Также выводится строка заголовков и итоговая строка.

Команды psql для переключения режима выравнивания:

\a — переключатель режима: с выравниванием/без выравнивания.

\t — переключатель отображения строки заголовка и итоговой строки.

Отключим выравнивание, заголовок и итоговую строку:

### => \a \t

Output format is unaligned. Tuples only is on.

#### => SELECT name, setting, unit FROM pg\_settings LIMIT 7;

allow\_in\_place\_tablespaces|off| allow\_system\_table\_mods|off| application\_name|psql| archive\_cleanup\_command|| archive\_command|(disabled)| archive\_library|| archive\_mode|off|

### => \a \t

Output format is aligned. Tuples only is off.

Такой формат неудобен для просмотра, но может оказаться полезным для автоматической обработки результатов.

Расширенный формат удобен, когда нужно вывести много столбцов для одной или нескольких записей.

Для этого вместо точки с запятой указываем в конце команды \gx:

#### => SELECT name, setting, unit, category, context, vartype, min\_val, max\_val, boot\_val, reset\_val FROM pg\_settings

**WHERE name = 'work\_mem' \gx**

-[ RECORD 1 ]----------------------

name | work\_mem

setting | 4096

unit | kB

category | Resource Usage / Memory

context | user

vartype | integer

min\_val | 64

max\_val | 2147483

Если расширенный формат нужен не для одной команды, а постоянно, можно включить его переключателем \x. Все возможности форматирования результатов запросов доступны через команду \pset.

## Взаимодействие с ОС и выполнение скриптов

Из psql можно выполнять команды shell:

### => \! pwd

/home/student

С помощью запроса SQL можно сформировать несколько других запросов SQL и записать их в файл, используя команду \o[ut]:

### => \a \t \pset fieldsep ''

Output format is unaligned.

Tuples only is on.

Field separator is "".

### => \o dev1\_tools.sql

**=> SELECT \* FROM pg\_tables LIMIT 3;**

На экран (в стандартный вывод) ничего не попало. Посмотрим в файле:

### => \! cat dev1\_tools.sql

### Усложняем запрос

**=> SELECT format('SELECT %L AS tbl, count(\*) FROM %I;', tablename, tablename) FROM pg\_tables LIMIT 3;**

На экран (в стандартный вывод) ничего не попало. Посмотрим в файле:

### => \! cat dev1\_tools.sql

SELECT 'pg\_statistic' AS tbl, count(\*) FROM pg\_statistic;

SELECT 'pg\_type' AS tbl, count(\*) FROM pg\_type;

SELECT 'pg\_foreign\_table' AS tbl, count(\*) FROM pg\_foreign\_table;

Вернем вывод на экран и восстановим форматирование по умолчанию.

### => \o \t \a

Tuples only is off.

Output format is aligned.

Выполним теперь эти команды из файла с помощью \i[nclude]:

**=> \i dev1\_tools.sql**

**tbl | count**

**--------------+-------**

**pg\_statistic | 409**

**(1 row)**

**tbl | count**

**---------+-------**

**pg\_type | 613**

**(1 row)**

**tbl | count**

**------------------+-------**

**pg\_foreign\_t**

Есть и другие способы выполнить команды, в том числе из файлов. После выполнения команд сеанс psql будет завершен:

psql < имя\_файла

psql -f имя\_файла

psql -c 'команда' (работает только для одной команды)

## Переменные psql

По аналогии с shell, psql имеет собственные переменные.

Установим переменную:

**=> \set TEST Hi!**

Чтобы подставить значение переменной, надо предварить ее имя двоеточием:

### => \echo :TEST

Hi!

Значение переменной можно сбросить:

**=> \unset TEST**

### => \echo :TEST

:TEST

Переменные можно использовать, например, для хранения текста часто используемых запросов.

Вот запрос на получение списка пяти самых больших по размеру таблиц:

**=> \set top5 'SELECT tablename, pg\_total\_relation\_size(schemaname||''.''||tablename) AS bytes FROM pg\_tables ORDER BY bytes DESC LIMIT 5;'** Для выполнения запроса достаточно набрать:

### => :top5

tablename | bytes

----------------+---------

pg\_proc | 1245184

pg\_rewrite | 745472

pg\_attribute | 720896

pg\_description | 630784

pg\_statisti

Присвоение значения переменной top5 удобно записать в стартовый файл .psqlrc в домашнем каталоге пользователя. Команды из .psqlrc будут автоматически выполняться каждый раз при старте psql.

Без параметров \set выдает значения всех переменных, включая встроенные. Справку по встроенным переменным можно получить так:

#### \? variables

Выйти из psql можно с помощью команд \q[uit], quit, exit или нажав Ctrl+D: **=> \q**

|  |
| --- |
| Итоги |
| Установка PostgreSQL из готовых пакетов — предпочтительный способ установки  Пакетные дистрибутивы учитывают особенности ОС, которые нужно знать  как запускать и останавливать сервер расположение файлов конфигурации расположение журнала сервера psql — клиент для работы с PostgreSQL  13 |

|  |
| --- |
| Практика |
| 1. Установите для всех сеансов значение параметра *work\_mem* равным 8 Мбайт.   Обновите конфигурацию и проверьте, что изменения вступили в силу.  Восстановите значение по умолчанию.   1. Запишите в файл ddl.sql команду CREATE TABLE на создание любой таблицы.   Запишите в файл populate.sql команды на вставку строк в эту таблицу.  Выполните оба скрипта и проверьте, что таблица создалась и в ней появились записи.   1. Найдите в журнале сервера строки за сегодняшний день.   14 |

Для запуска psql в окне терминала наберите psql без параметров: student:~$ **psql**

Для выполнения заданий каждой темы удобно создавать отдельную базу данных: student/student=# **CREATE DATABASE tools\_overview;** CREATE DATABASE

student/student=# **\c tools\_overview**

You are now connected to database "tools\_overview" as user

"student". student/tools\_overview=#

1. Воспользуйтесь командой ALTER SYSTEM.
2. Если результат запроса не помещается в окно терминала, psql передает его утилите less для постраничной выдачи. Чтобы вернуться в режим ввода команд, нажмите клавишу **q**.

# Параметры конфигурации

Подключаемся к базе данных:

## student$ psql

Задаем значение параметра:

**=> ALTER SYSTEM SET work\_mem = '8MB';**

ALTER SYSTEM

Обновляем конфигурацию: **=> SELECT pg\_reload\_conf();**

pg\_reload\_conf --------------- t

(1 row)

Проверяем значение параметра:

## => \dconfig work\_mem

List of configuration parameters

Parameter | Value -----------+------ work\_mem | 8MB (1 row)

Вернем значение по умолчанию:

**=> ALTER SYSTEM RESET work\_mem;**

ALTER SYSTEM

Применяем изменения: **=> SELECT pg\_reload\_conf();**

pg\_reload\_conf --------------- t

(1 row)

# Выполнение скриптов в psql

Запишем в файл ddl.sql команду на создание таблицы с ключевыми словами PostgreSQL (для этого можно использовать и любой текстовый редактор):

**student$ cat > ~/ddl.sql <<EOF CREATE TABLE keywords ( word text, category text, description text**

**);**

**EOF**

Запишем команды для заполнения таблицы keywords в файл populate.sql:

## student$ cat > ~/populate.sql <<EOF

**INSERT INTO keywords**

**SELECT word, catcode, catdesc FROM pg\_get\_keywords(); EOF**

Создаем базу данных и подключаемся к ней:

**=> CREATE DATABASE tools\_overview;**

CREATE DATABASE

## => \c tools\_overview

You are now connected to database "tools\_overview" as user "student".

Выполняем скрипты и проверяем записи в таблице:

**=> \i ddl.sql**

CREATE TABLE

## => \i populate.sql

INSERT 0 471

**=> SELECT \* FROM keywords LIMIT 10;**

word | category | description

-----------+----------+-------------

abort | U | unreserved

absent | U | unreserved

absolute | U | unreserved

access | U | unreserved

action | U | unreserved

add | U | unreserved

admin | U | unreserved

after | U | unreserved

aggregate | U | unreserved

all | R | reserved

(10 rows)

Просмотр журнала

Журнал можно открыть любым текстовым редактором. Каждая запись в журнале начинается с даты, содержит номер серверного процесса (таковы настройки журнала после установки из пакета) и может состоять из нескольких строк. Последние записи будут в конце файла.

## student$ tail /var/log/postgresql/postgresql-16-main.log

2024-07-05 16:28:23.857 MSK [43402] LOG: database system is shut down

2024-07-05 16:28:24.499 MSK [45377] LOG: starting PostgreSQL 16.3 (Ubuntu 16.3-1.pgdg22.04+1) on x86\_64-pc-linux-gnu, compiled by gcc (Ubuntu

11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit

2024-07-05 16:28:24.500 MSK [45377] LOG: listening on IPv4 address "127.0.0.1", port 5432

2024-07-05 16:28:24.514 MSK [45377] LOG: listening on Unix socket

"/var/run/postgresql/.s.PGSQL.5432"

2024-07-05 16:28:24.549 MSK [45380] LOG: database system was shut down at 2024-07-03 15:57:35 MSK

2024-07-05 16:28:24.569 MSK [45377] LOG: database system is ready to accept connections

2024-07-05 16:28:27.722 MSK [45377] LOG: received SIGHUP, reloading configuration files

2024-07-05 16:28:27.722 MSK [45377] LOG: parameter "work\_mem" changed to "8MB"

2024-07-05 16:28:28.071 MSK [45377] LOG: received SIGHUP, reloading configuration files

2024-07-05 16:28:28.071 MSK [45377] LOG: parameter "work\_mem" removed from configuration file, reset to default

Созданную нами базу можно удалить, предварительно отключившись от нее (подробности будут в модуле «Организация данных»)

**=> \c postgres**

You are now connected to database "postgres" as user "student".

## => DROP DATABASE tools\_overview;

DROP DATABASE