Обзор базового инструментария

Установка и управление, psql

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов, Игорь Гнатюк
Фото: Олег Бартунов (монастырь Пху и пик Бхрикути, Непал)

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Варианты установки PostgreSQL

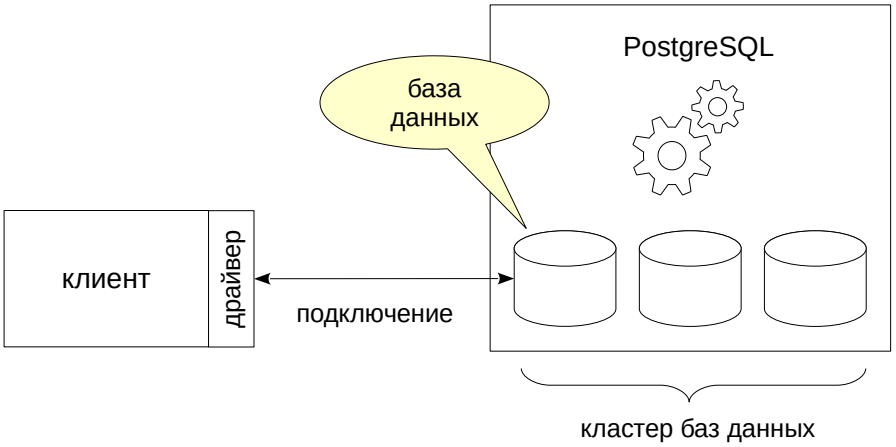
Управление сервером

Журнал сообщений сервера

Настройка параметров конфигурации

Использование psql

Кластер БД, сервер, клиент



Начнем с основных понятий.

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

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.

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.

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)

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/

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.

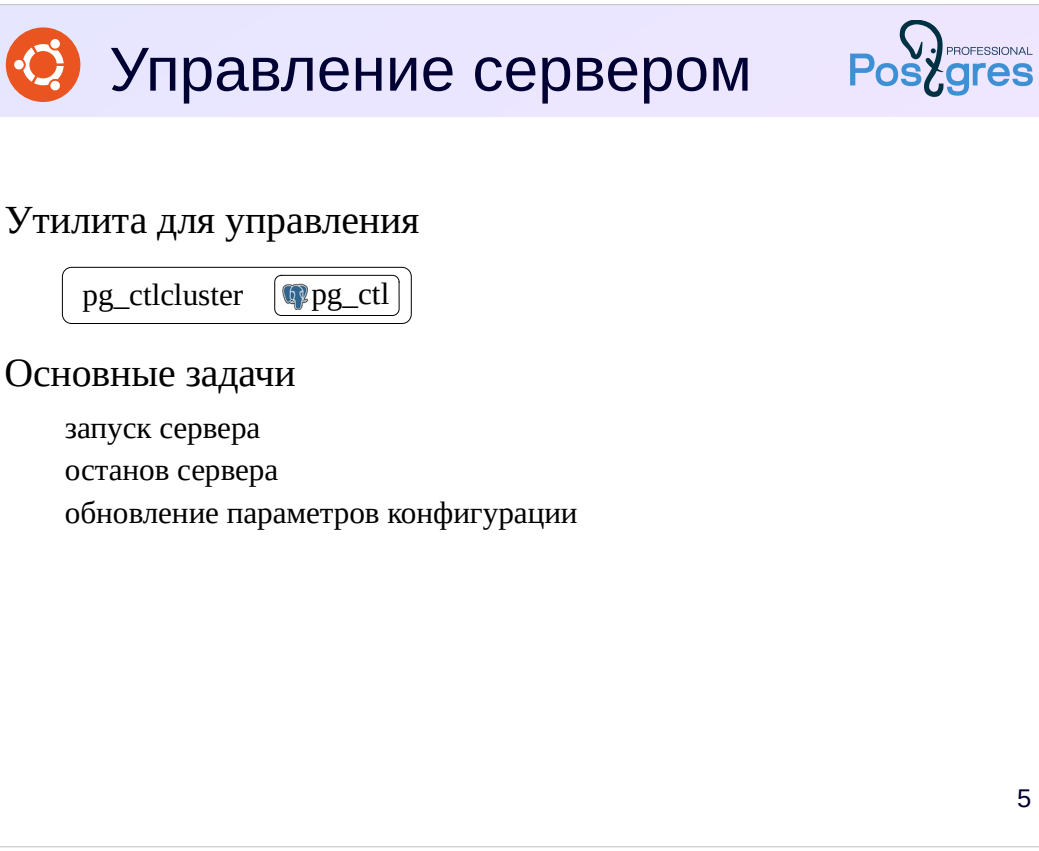
В курсе мы будем использовать виртуальную машину с ОС Xubuntu 22 и сервер PostgreSQL 16, установленный из пакета для этой ОС. В этом случае сразу настраивается автоматический запуск и останов сервера PostgreSQL при запуске и останове операционной системы.

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:

$ 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 16
drwxr-xr-x 2 root root  4096 янв 15 10:24 bin
drwxr-xr-x 4 root root 12288 янв 15 10:24 lib
```

Владелец ПО сервера — пользователь root.

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 84
drwx------ 9 postgres postgres 4096 янв 18 12:39 base
drwx------ 2 postgres postgres 4096 янв 18 12:39 global
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_commit_ts
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_dynshmem
drwx------ 4 postgres postgres 4096 янв 18 12:39 pg_logical
drwx------ 4 postgres postgres 4096 дек 21 11:35 pg_multixact
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_notify
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_replslot
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_serial
drwx------ 2 postgres postgres 4096 янв  9 22:55 pg_snapshots
drwx------ 2 postgres postgres 4096 янв 18 12:39 pg_stat
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_stat_tmp
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_subtrans
drwx------ 2 postgres postgres 4096 янв 13 12:08 pg_tblspc
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_twophase
-rw------- 1 postgres postgres    3 дек 21 11:35 PG_VERSION
drwx------ 3 postgres postgres 4096 янв 18 11:03 pg_wal
drwx------ 2 postgres postgres 4096 дек 21 11:35 pg_xact
-rw------- 1 postgres postgres   88 янв 18 12:39 postgresql.auto.conf
-rw------- 1 postgres postgres  130 янв 18 12:39 postmaster.opts
-rw------- 1 postgres postgres  108 янв 18 12:39 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
```

Список установленных экземпляров (можно без sudo):

```
student$ pg_lsclusters
```

По-прежнему есть возможность использования утилиты pg_ctl, запускаемой от имени пользователя ОС postgres.

Журнал сообщений

В журнал записываются

- служебные сообщения сервера
- сообщения пользовательских сеансов
- сообщения приложений

Настройка журнала

- расположение
- формат записей
- какие события регистрировать

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 322549 янв 18 12:39 /var/log/postgresql/postgresql-16-main.log
```

Заглянем в конец журнала:

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

```
2024-01-18 12:39:32.818 MSK [9352] LOG:  background worker "logical replication launcher"
(PID 9358) exited with exit code 1
2024-01-18 12:39:32.818 MSK [9353] LOG:  shutting down
2024-01-18 12:39:32.819 MSK [9353] LOG:  checkpoint starting: shutdown immediate
2024-01-18 12:39:32.825 MSK [9353] LOG:  checkpoint complete: wrote 1 buffers (0.0%); 0
WAL file(s) added, 0 removed, 0 recycled; write=0.002 s, sync=0.001 s, total=0.007 s;
sync files=1, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=14 kB;
lsn=2/295DC9C0, redo lsn=2/295DC9C0
2024-01-18 12:39:32.829 MSK [9352] LOG:  database system is shut down
2024-01-18 12:39:33.022 MSK [9518] LOG:  starting PostgreSQL 16.1 (Ubuntu
16.1-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
2024-01-18 12:39:33.022 MSK [9518] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2024-01-18 12:39:33.024 MSK [9518] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2024-01-18 12:39:33.028 MSK [9521] LOG:  database system was shut down at 2024-01-18
12:39:32 MSK
2024-01-18 12:39:33.033 MSK [9518] 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`

Сервер PostgreSQL настраивается с помощью разнообразных параметров конфигурации, которые служат для управления потреблением ресурсов, настройки служебных процессов и пользовательских сеансов, управления журналом сервера и для решения многих других задач. В ходе курса мы будем встречаться с некоторыми из этих параметров. А сейчас важно разобраться с тем, как проверить текущие значения 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.

PostgreSQL предоставляет возможности для разделения сложных файлов postgresql.conf на физически отдельные части, для этого предусмотрены директивы включения файлов и каталогов конфигураций include и include_dir. 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.

Варианты установки и управления параметрами:
https://postgrespro.ru/docs/postgresql/16/config-setting
Текущие значения доступны для просмотра 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 68
drwxr-xr-x 2 postgres postgres  4096 янв  9 22:56 conf.d
-rw-r--r-- 1 postgres postgres   315 дек 21 11:35 environment
-rw-r--r-- 1 postgres postgres   143 дек 21 11:35 pg_ctl.conf
-rw-r----- 1 postgres postgres  5924 дек 25 10:05 pg_hba.000
-rw-r----- 1 postgres postgres  5925 янв  9 23:02 pg_hba.conf
-rw-r----- 1 postgres postgres  2640 дек 21 11:35 pg_ident.conf
-rw-r--r-- 1 postgres postgres 30013 дек 21 20:35 postgresql.conf
-rw-r--r-- 1 postgres postgres   317 дек 21 11:35 start.conf
```

Здесь же находятся и другие конфигурационные файлы.

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)
```

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
```

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)
```

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

Клиент psql

Терминальный клиент для работы с PostgreSQL

Поставляется вместе с СУБД

Используется администраторами и разработчиками для интерактивной работы и выполнения скриптов

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 с 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
```

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:

- \h[elp] выдает список команд SQL, которые поддерживает сервер, а также синтаксис конкретной команды SQL.
- \? выдает список команд psql.

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   | 2147483647
boot_val  | 4096
reset_val | 4096
```

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

В версии 15 появилась возможность получения значений параметров с помощью короткой команды \dconfig. Возможно использование шаблонов и вывод расширенной информации о параметре.

```
=> \dconfig work_mem
```

```
List of configuration parameters
 Parameter | Value
-----------+-------
 work_mem  | 4MB
(1 row)
```

Взаимодействие с ОС и выполнение скриптов

Из psql можно выполнять команды shell:

```
=> \! pwd
```

```
/home/student
```

Можно установить переменную окружения операционной системы:

```
=> \setenv TEST Hello
```

```
=> \! echo $TEST
```

```
Hello
```

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

```
=> \a \t
```

```
Output format is unaligned.
Tuples only is on.
```

```
=> \pset fieldsep ''
```

```
Field separator is "".
```

```
=> \o dev1_psql.log
```

```
=> 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_psql.log
```

```
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_psql.log
```

```
     tbl      | count
--------------+-------
 pg_statistic |   416
(1 row)

   tbl   | count
---------+-------
 pg_type |   615
(1 row)

       tbl        | count
------------------+-------
 pg_foreign_table |     0
(1 row)
```

То же самое можно получить за один шаг, используя команду \gexec:

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

```
     tbl      | count
--------------+-------
 pg_statistic |   416
(1 row)

   tbl   | count
---------+-------
 pg_type |   615
(1 row)

       tbl        | count
------------------+-------
 pg_foreign_table |     0
(1 row)
```

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 получить значение переменной, надо предварить ее имя двоеточием:

```
=> \echo :TEST
```

```
Hi!
```

Есть возможность получить значение переменной окружения ОС и присвоить его переменной psql:

```
=> \getenv TEST HOME
```

```
=> \echo :TEST
```

```
/home/student
```

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        | 1286144
 pg_rewrite     |  753664
 pg_attribute   |  737280
 pg_description |  630784
 pg_depend      |  303104
(5 rows)
```

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.

Результат запроса можно записать в переменную с помощью \gset:

```
=> SELECT current_setting('work_mem') AS current_work_mem \gset
```

```
=> \echo Значение work_mem: :current_work_mem
```

```
Значение work_mem: 4MB
```

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

```
\? variables
```

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

Практика

1. Создайте конфигурационный файл, устанавливающий значение параметра *work_mem* 8 Мбайт и разместите его в каталоге conf.d.
   Обновите конфигурацию и проверьте, что изменения вступили в силу.
2. Запишите в файл ddl.sql команду CREATE TABLE на создание любой таблицы.
   Запишите в файл populate.sql команды на вставку строк в эту таблицу.
   Войдите в psql, выполните оба скрипта и проверьте, что таблица создалась и в ней появились записи.
3. Найдите в журнале сервера строки за сегодняшний день.

Для выполнения практических заданий нужно войти в операционную систему под пользователем student (пароль student).

Для запуска psql в окне терминала наберите psql без параметров. Для подключения будут использованы настройки по умолчанию.

```
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 любым текстовым редактором. В виртуальной машине установлены mousepad, gedit, vim, nano.

Обратите внимание: если вы запускаете редактор из графической среды, он будет запущен под пользователем ОС student. В этом случае у вас не будет прав на редактирование файла postgresql.conf, который принадлежит пользователю postgres. Поэтому вам необходимо либо переключиться в пользователя postgres, либо действовать с правами суперпользователя ОС:

```
student:~$ sudo gedit
```

1. Параметры конфигурации

Создадим подключаемый конфигурационный файла в нужном каталоге и запишем в него строку:

```
student$ echo 'work_mem = 8MB' | sudo tee -a /etc/postgresql/16/main/conf.d/misc.conf
```

```
work_mem = 8MB
```

Это можно сделать и любым текстовым редактором.

Обновляем конфигурацию:

```
student$ sudo pg_ctlcluster 16 main reload
```

Проверяем:

```
student$ psql
```

```
=> SELECT current_setting('work_mem') AS work_mem;
```

```
 work_mem
----------
 8MB
(1 row)
```

При необходимости вернуться к исходной настройке удалим созданный файл:

```
student$ sudo rm /etc/postgresql/16/main/conf.d/misc.conf
```

Применяем изменения:

```
student$ sudo pg_ctlcluster 16 main reload
```

2. 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)
```

3. 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
```

```
2024-01-18 12:48:04.350 MSK [57747] LOG:  database system is shut down
2024-01-18 12:48:04.534 MSK [57917] LOG:  starting PostgreSQL 16.1 (Ubuntu
16.1-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
2024-01-18 12:48:04.534 MSK [57917] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2024-01-18 12:48:04.536 MSK [57917] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2024-01-18 12:48:04.541 MSK [57920] LOG:  database system was shut down at 2024-01-18
12:48:04 MSK
2024-01-18 12:48:04.548 MSK [57917] LOG:  database system is ready to accept connections
2024-01-18 12:48:07.035 MSK [57917] LOG:  received SIGHUP, reloading configuration files
2024-01-18 12:48:07.036 MSK [57917] LOG:  parameter "work_mem" changed to "8MB"
2024-01-18 12:48:07.419 MSK [57917] LOG:  received SIGHUP, reloading configuration files
2024-01-18 12:48:07.419 MSK [57917] LOG:  parameter "work_mem" removed from configuration
file, reset to default
```