Темы

Параметры конфигурации

Файлы конфигурации

Управление параметрами на уровнях экземпляра и сеанса

Параметры

Задача

управление работой и поведением СУБД

Установка

для экземпляра — файлы конфигурации

для отдельной базы или пользователя

для текущего сеанса

В PostgreSQL существует большое количество параметров, влияющих на работу СУБД. Параметры позволяют управлять потреблением ресурсов, настраивать работу серверных процессов и многое другое.

Например, при помощи параметра *max_connections* можно ограничить количество одновременных подключений к серверу.

Полный список и описание параметров конфигурации: https://postgrespro.ru/docs/postgresql/16/runtime-config

В этой теме мы не изучаем назначение отдельных параметров конфигурации, а лишь рассматриваем, какими способами им можно устанавливать значения.

Для установки параметров, в первую очередь, используются файлы конфигурации. Если не определено иное, значения, установленные в этих файлах, действуют для всего экземпляра СУБД.

Ряд параметров можно установить для сеансов в отдельной базе данных или для сеансов отдельного пользователя. Такие установки будут иметь предпочтение перед файлами конфигурации. Мы подробнее коснемся этого варианта в следующих темах курса.

Наконец, многими параметрами можно управлять на уровне отдельного сеанса, прямо во время работы.

postgresql.conf

Основной файл конфигурации

считывается при старте сервера

возможность подключения дополнительных файлов

по умолчанию находится в каталоге с данными (PGDATA)

`/etc/postgresql/16/main`

При изменении параметров необходимо перечитать файл

```
$ pg_ctl reload
$ pg_ctlcluster 16 main reload
=> SELECT pg_reload_conf();
```

изменение ряда параметров требует перезапуска сервера

Основной конфигурационный файл — postgresql.conf.

Расположение файла по умолчанию задается при сборке PostgreSQL. При запуске исполняемого файла сервера с помощью аргумента командной строки -c config_file можно задать требуемое местонахождение файла конфигурации.

По умолчанию файл располагается в каталоге с данными (PGDATA), но пакетные дистрибутивы обычно размещают этот файл в другом месте, в соответствии с правилами, принятыми в конкретной ОС.

Это текстовый, хорошо документированный файл, хранящий параметры в формате «ключ=значение».

Есть возможность подключать дополнительные файлы. По умолчанию в Ubuntu подключаются все файлы из каталога /etc/postgresql/16/main/conf.d.

Если один и тот же параметр указан в конфигурационных файлах несколько раз, будет использоваться значение, считанное последним.

Для вступления в силу внесенных в файл изменений необходимо, чтобы сервер перечитал файл. Для некоторых параметров требуется перезагрузка сервера.

Файл postgresql.conf и представление pg_file_settings

Имя конфигурационного файла содержится в доступном для чтения параметре config_file. Имя конфигурационного файла можно указать с помощью ключа командной строки при запуске postgres.

```
=> SHOW config_file;
```

```
               config_file
----------------------------------------
 /etc/postgresql/16/main/postgresql.conf
(1 row)
```

Посмотрим небольшой фрагмент конфигурационного файла.

```
=> SELECT pg_read_file('/etc/postgresql/16/main/postgresql.conf', 1516, 861) \g (tuples_only=on format=unaligned)
```

```
#------------------------------------------------------------------------------
# FILE LOCATIONS
#------------------------------------------------------------------------------

# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.

data_directory = '/var/lib/postgresql/16/main'          # use data in another directory
                                        # (change requires restart)
hba_file = '/etc/postgresql/16/main/pg_hba.conf'        # host-based authentication file
                                        # (change requires restart)
ident_file = '/etc/postgresql/16/main/pg_ident.conf'    # ident configuration file
                                        # (change requires restart)

# If external_pid_file is not explicitly set, no extra PID file is written.
external_pid_file = '/var/run/postgresql/16-main.pid'                   # write an extra PID file
                                        # (change requires restart)
```

К основному конфигурационному файлу postgresql.conf можно подключать дополнительные файлы конфигурации. Директивы подключения:

- include_dir — каталог с дополнительными файлами конфигурации;
- include — включает дополнительный файл конфигурации;
- include_if_exists — включает дополнительный файл конфигурации, если он существует.

Обычно эти директивы располагаются в завершающей части файла postgresql.conf:

```
student$ sudo grep -A3 ^include /etc/postgresql/16/main/postgresql.conf
```

```
include_dir = 'conf.d'                  # include files ending in '.conf' from
                                        # a directory, e.g., 'conf.d'
#include_if_exists = '...'              # include file only if it exists
#include = '...'                        # include file
```

Чтобы увидеть настройки в конфигурационных файлах, можно обратиться к представлению pg_file_settings:

```
=> SELECT sourceline, name, setting, applied, error FROM pg_file_settings;
```

```
 sourceline |            name            |                setting                 |
applied | error
------------+----------------------------+----------------------------------------+-------
--+-------
         42 | data_directory             | /var/lib/postgresql/16/main            | t
  |
         44 | hba_file                   | /etc/postgresql/16/main/pg_hba.conf    | t
  |
         46 | ident_file                 | /etc/postgresql/16/main/pg_ident.conf  | t
  |
         50 | external_pid_file          | /var/run/postgresql/16-main.pid        | t
  |
         64 | port                       | 5432                                   | t
  |
         65 | max_connections            | 100                                    | t
  |
         68 | unix_socket_directories    | /var/run/postgresql                    | t
  |
        108 | ssl                        | on                                     | t
  |
        110 | ssl_cert_file              | /etc/ssl/certs/ssl-cert-snakeoil.pem   | t
  |
        113 | ssl_key_file               | /etc/ssl/private/ssl-cert-snakeoil.key | t
  |
        130 | shared_buffers             | 128MB                                  | t
  |
        153 | dynamic_shared_memory_type | posix                                  | t
  |
        247 | max_wal_size               | 1GB                                    | t
  |
        248 | min_wal_size               | 80MB                                   | t
  |
        565 | log_line_prefix            | %m [%p] %q%u@%d                        | t
  |
        603 | log_timezone               | Europe/Moscow                          | t
  |
        607 | cluster_name               | 16/main                                | t
  |
        715 | datestyle                  | iso, dmy                               | t
  |
        717 | timezone                   | Europe/Moscow                          | t
  |
        731 | lc_messages                | en_US.UTF-8                            | t
  |
        733 | lc_monetary                | ru_RU.UTF-8                            | t
  |
        734 | lc_numeric                 | ru_RU.UTF-8                            | t
  |
        735 | lc_time                    | ru_RU.UTF-8                            | t
  |
        741 | default_text_search_config | pg_catalog.english                     | t
  |
(24 rows)
```

Представление выводит незакомментированные строки конфигурационных файлов. Столбец applied показывает, будет ли заданное значение применено при перечитывании. В частности, в столбце будет false, если:

- изменение требует рестарта сервера;
- существует строка с тем же параметром, которая будет прочитана позже;
- в одной из строк, где задается параметр, есть ошибка.

Представление также показывает имя файла конфигурации и номер строки, что удобно для поиска ошибок.

Представление pg_settings

Возьмем для примера параметр work_mem. Он определяет объем памяти, выделяемый для таких операций, как сортировка или хеш-соединение. Не для всех запросов значения по умолчанию может быть достаточно. Подробнее о параметре work_mem можно узнать в курсе QPT «Оптимизация запросов».

Действующие значения всех параметров доступны в представлении pg_settings. Вот что в нем содержится для параметра work_mem:

```
=> SELECT name, unit, setting, boot_val, reset_val,
  source, sourcefile, sourceline, pending_restart, context
FROM pg_settings
WHERE name = 'work_mem' \gx
```

```
-[ RECORD 1 ]---+---------
name            | work_mem
unit            | kB
setting         | 4096
boot_val        | 4096
reset_val       | 4096
source          | default
sourcefile      |
sourceline      |
pending_restart | f
context         | user
```

Рассмотрим ключевые столбцы представления pg_settings:

- name, unit — название и единица измерения параметра;
- setting — текущее значение;
- boot_val — значение по умолчанию;
- reset_val — начальное значение для сеансов;
- source — источник текущего значения параметра;
- sourcefile, sourceline — файл конфигурации и номер строки, если текущее значение было задано в файле;
- pending_restart — true, если значение изменено в файле конфигурации, но для применения требуется перезапуск сервера.

Столбец context определяет действия, необходимые для применения параметра. Среди возможных значений:

- internal — изменить нельзя, значение задано при установке;
- postmaster — требуется перезапуск сервера;
- sighup — требуется перечитать файлы конфигурации,
- superuser — суперпользователь может изменить для своего сеанса;
- user — любой пользователь может изменить для своего сеанса.

Порядок применения строк

При перечитывании конфигурации сначала читается основной файл, а затем дополнительные. Если один и тот же параметр встречается несколько раз, то устанавливается значение из последней считанной строки.

Например, укажем дважды параметр work_mem в дополнительном файле конфигурации:

```
student$ echo work_mem=12MB | sudo tee /etc/postgresql/16/main/conf.d/work_mem.conf
```

```
work_mem=12MB
```

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

```
work_mem=8MB
```

Содержимое файла /etc/postgresql/16/main/conf.d/work_mem.conf:

```
=> SELECT sourcefile, sourceline, name, setting, applied
FROM pg_file_settings WHERE sourcefile LIKE '%/work_mem.conf';
```

```
                  sourcefile                  | sourceline |   name   | setting | applied
----------------------------------------------+------------+----------+---------+---------
 /etc/postgresql/16/main/conf.d/work_mem.conf |          1 | work_mem | 12MB    | f
 /etc/postgresql/16/main/conf.d/work_mem.conf |          2 | work_mem | 8MB     | t
(2 rows)
```

Значение applied = f для первой строки показывает, что она не будет применена.

Для параметра work_mem поле context имеет значение user. Значит, параметр можно менять прямо во время сеанса, и позже мы увидим, как это сделать.

А чтобы изменить значение во всех сеансах, достаточно перечитать файлы конфигурации:

```
=> SELECT pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)
```

Убедимся, что параметр work_mem получил значение из второй строки:

```
=> SELECT name, unit, setting, boot_val, reset_val,
  source, sourcefile, sourceline, pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-----------------------------------------
name            | work_mem
unit            | kB
setting         | 8192
boot_val        | 4096
reset_val       | 8192
source          | configuration file
sourcefile      | /etc/postgresql/16/main/conf.d/work_mem.conf
sourceline      | 2
pending_restart | f
context         | user
```

postgresql.auto.conf

Файл конфигурации, управляемый командами SQL

`ALTER SYSTEM SET` *`параметр`* `TO` *`значение`*`;`	добавляет или изменяет строку
`ALTER SYSTEM RESET` *`параметр`*`;`	удаляет строку
`ALTER SYSTEM RESET ALL;`	удаляет все строки

считывается после `postgresql.conf`

Расположение

всегда в каталоге с данными (PGDATA)

Действия при изменении

аналогично `postgresql.conf`

Последним считывается файл postgresql.auto.conf. Он всегда располагается в каталоге данных (PGDATA).

Этот файл не следует изменять вручную, для его редактирования предназначена команда ALTER SYSTEM. По сути, ALTER SYSTEM представляет собой SQL-интерфейс для управления параметрами конфигурации.

Для применения изменений, сделанных командой ALTER SYSTEM, сервер должен перечитать конфигурационные файлы, как и в случае с изменением файла postgresql.conf.

Содержимое файлов конфигурации можно увидеть в представлении pg_file_settings. А действующие значения параметров — в представлении pg_settings.

Более подробная информация о команде ALTER SYSTEM:

https://postgrespro.ru/docs/postgresql/16/sql-altersystem

Команда ALTER SYSTEM и файл postgresql.auto.conf

Для примера установим параметр work_mem:

```
=> ALTER SYSTEM SET work_mem TO '16mb';
```

```
ERROR:  invalid value for parameter "work_mem": "16mb"
HINT:  Valid units for this parameter are "B", "kB", "MB", "GB", and "TB".
```

Что случилось?

ALTER SYSTEM выполняет проверку на допустимые значения.

```
=> ALTER SYSTEM SET work_mem TO '16MB';
```

```
ALTER SYSTEM
```

Вот теперь все правильно.

В результате выполнения команды значение 16MB записано в файл postgresql.auto.conf:

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
work_mem = '16MB'
```

Но это значение не применено:

```
=> SHOW work_mem;
```

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

Чтобы применить изменение work_mem, перечитаем файлы конфигурации:

```
=> SELECT pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)
```

```
=> SELECT name, unit, setting, boot_val, reset_val,
  source, sourcefile, sourceline, pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-------------------------------------------------
name            | work_mem
unit            | kB
setting         | 16384
boot_val        | 4096
reset_val       | 16384
source          | configuration file
sourcefile      | /var/lib/postgresql/16/main/postgresql.auto.conf
sourceline      | 3
pending_restart | f
context         | user
```

Для удаления строк из postgresql.auto.conf используется команда ALTER SYSTEM RESET:

```
=> ALTER SYSTEM RESET work_mem;
```

```
ALTER SYSTEM
```

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
```

Еще раз перечитаем конфигурацию. Теперь восстановится значение из work_mem.conf:

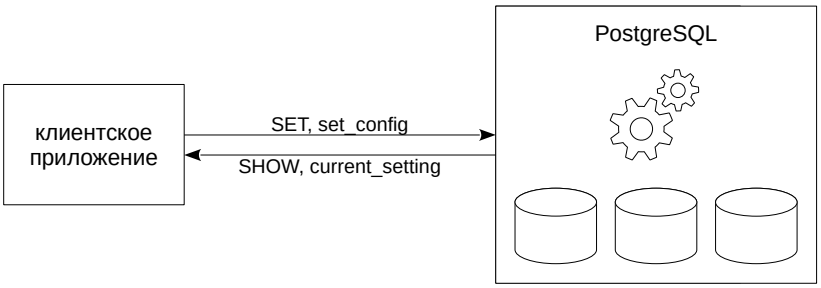
```
=> SELECT pg_reload_conf();

 pg_reload_conf
----------------
 t
(1 row)


=> SELECT name, unit, setting, boot_val, reset_val,
  source, sourcefile, sourceline, pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx

-[ RECORD 1 ]---+-----------------------------------------
name            | work_mem
unit            | kB
setting         | 8192
boot_val        | 4096
reset_val       | 8192
source          | configuration file
sourcefile      | /etc/postgresql/16/main/conf.d/work_mem.conf
sourceline      | 2
pending_restart | f
context         | user
```

В текущем сеансе



установка до конца сеанса или транзакции

установка параметров транзакционна

допускаются пользовательские параметры

Значения параметров можно изменить прямо во время сеанса командой SET или функцией set_config. А для получения текущего значения служит команда SHOW или функция current_setting.

Устанавливая новое значение, можно указать срок его действия: до конца сеанса (по умолчанию) или до конца транзакции (SET LOCAL).

В любом случае установка параметров транзакционна: в случае отмены текущей транзакции, значения измененных в ней параметров вернутся к состоянию на начало транзакции.

Помимо системных параметров PostgreSQL, этими же командами и функциями можно пользоваться для создания и получения значений пользовательских параметров.

Установка параметров для текущего сеанса

Для изменения параметров во время сеанса можно использовать команду SET:

```
=> SET work_mem TO '24MB';
```

```
SET
```

Или функцию set_config:

```
=> SELECT set_config('work_mem', '32MB', false);
```

```
 set_config
------------
 32MB
(1 row)
```

Третий параметр функции говорит о том, нужно ли устанавливать значение только для текущей транзакции (true) или до конца работы сеанса (false). Это важно при работе приложения через пул соединений, когда в одном сеансе могут выполняться транзакции разных пользователей.

Чтение значений параметров во время выполнения

Получить значение параметра можно разными способами:

```
=> SHOW work_mem;
```

```
 work_mem
----------
 32MB
(1 row)
```

```
=> \dconfig work_mem
```

```
List of configuration parameters
 Parameter | Value
-----------+-------
 work_mem  | 32MB
(1 row)
```

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

```
 current_setting
-----------------
 32MB
(1 row)
```

```
=> SELECT name, setting, unit FROM pg_settings WHERE name = 'work_mem';
```

```
   name   | setting | unit
----------+---------+------
 work_mem | 32768   | kB
(1 row)
```

Сбросим значение к тому, которое действовало в начале сеанса:

```
=> RESET work_mem;
```

```
RESET
```

Установка параметров внутри транзакции

Откроем транзакцию и установим новое значение work_mem:

```
=> BEGIN;
```

```
BEGIN
```

```
=> SET work_mem TO '64MB';
```

```
SET
```

```
=> SHOW work_mem;

 work_mem
----------
 64MB
(1 row)
```

Если транзакция откатывается, установка параметра отменяется, хотя при успешной фиксации новое значение продолжало бы действовать.

```
=> ROLLBACK;

ROLLBACK

=> SHOW work_mem;

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

Можно установить значение только до конца текущей транзакции:

```
=> BEGIN;

BEGIN

=> SET LOCAL work_mem TO '64MB'; -- или set_config('work_mem','64MB',true);

SET

=> SHOW work_mem;

 work_mem
----------
 64MB
(1 row)


=> COMMIT;

COMMIT
```

По завершении транзакции значение восстанавливается:

```
=> SHOW work_mem;

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

Пользовательские параметры

Параметры можно создавать прямо во время сеанса, в том числе с предварительной проверкой на существование.

В имени пользовательских параметров обязательно должна быть точка, чтобы отличать их от стандартных параметров.

```
=> SELECT CASE
  WHEN current_setting('myapp.currency_code', true) IS NULL
    THEN set_config('myapp.currency_code', 'RUB', false)
  ELSE
    current_setting('myapp.currency_code')
END;

 current_setting
-----------------
 RUB
(1 row)
```

Теперь myapp.currency_code можно использовать как глобальную переменную сеанса:

```
=> SELECT current_setting('myapp.currency_code');

 current_setting
-----------------
 RUB
(1 row)
```

Пользовательские параметры можно указывать и в конфигурационных файлах, тогда они автоматически будут инициализироваться во всех сеансах.

Итоги

Основной файл конфигурации — postgresql.conf

`ALTER SYSTEM` — SQL-интерфейс для управления параметрами конфигурации в postgresql.auto.conf

При изменениях в файлах нужно перечитать конфигурацию

Многие параметры можно изменять для текущего сеанса

Изменение части параметров требует перезапуска сервера