**ИНСТРУКЦИЯ АДМИНИСТРАТОРА**

**RT.WAREHOUSE**

2021

ОГЛАВЛЕНИЕ

[1 ЗАПУСК И ОСТАНОВКА RT.WAREHOUSE 5](#_Toc74742885)

[1.1 Запуск RT.Warehouse 5](#_Toc74742886)

[1.2 Перезапуск RT.Warehouse 5](#_Toc74742887)

[1.3 Повторная загрузка только изменений файла конфигурации 6](#_Toc74742888)

[1.4 Запуск мастера в режиме обслуживания 6](#_Toc74742889)

[1.5 Остановка RT.Warehouse 6](#_Toc74742890)

[1.6 Остановка клиентских процессов 7](#_Toc74742891)

[2 ДОСТУП К RT.WAREHOUSE 9](#_Toc74742892)

[2.1 Подключение к базе данных 9](#_Toc74742893)

[2.2 Поддерживаемые клиентские приложения 9](#_Toc74742894)

[2.3 Клиентские приложения RT.Warehouse 10](#_Toc74742895)

[2.4 Подключение с помощью psql 11](#_Toc74742896)

[2.5 Использование пула подключений PgBouncer 11](#_Toc74742897)

[2.5.1 Обзор 12](#_Toc74742898)

[2.5.2 Перенос PgBouncer 13](#_Toc74742899)

[2.5.3 Настройка PgBouncer 13](#_Toc74742900)

[2.5.4 Запуск PgBouncer 15](#_Toc74742901)

[2.5.5 Управление PgBouncer 16](#_Toc74742902)

[2.6 Интерфейсы приложений базы данных 18](#_Toc74742903)

[2.7 Устранение проблем с подключением 18](#_Toc74742904)

[3 КОНТРОЛЬ ЦЕЛОСТНОСТИ В RT.WAREHOUSE (ИЗОЛЯЦИЯ ТРАНЗАКЦИЙ) 20](#_Toc74742905)

[3.1 Снапшоты 20](#_Toc74742906)

[3.2 Зацикливание ID транзакций 21](#_Toc74742907)

[3.3 Режимы изоляции транзакций 22](#_Toc74742908)

[3.4 Удаление неиспользуемых строк из таблиц 24](#_Toc74742909)

[4 ПАРАЛЛЕЛЬНАЯ ЗАГРУЗКА ДАННЫХ 26](#_Toc74742910)

[5 ИНФОРМАЦИЯ ОБ УТИЛИТАХ RT.WAREHOUSE 27](#_Toc74742911)

[5.1 Ссылка на IP-адреса 27](#_Toc74742912)

[5.2 Запуск программ внутреннего сервера 27](#_Toc74742913)

[5.3 Справочник по утилитам 28](#_Toc74742914)

[5.3.1 analyzedb 29](#_Toc74742915)

[5.3.2 clusterdb 34](#_Toc74742916)

[5.3.3 createdb 36](#_Toc74742917)

[5.3.4 createlang 38](#_Toc74742918)

[5.3.5 createuser 40](#_Toc74742919)

[5.3.6 dropdb 43](#_Toc74742920)

[5.3.7 droplang 44](#_Toc74742921)

[5.3.8 dropuser 46](#_Toc74742922)

[5.3.9 gpactivatestandby 48](#_Toc74742923)

[5.3.10 gpaddmirrors 50](#_Toc74742924)

[5.3.11 gpbackup 54](#_Toc74742925)

[5.3.12 gpcheckcat 62](#_Toc74742926)

[5.3.13 gpcheckperf 66](#_Toc74742927)

[5.3.14 gpconfig 70](#_Toc74742928)

[5.3.15 gpdeletesystem 73](#_Toc74742929)

[5.3.16 gpexpand 75](#_Toc74742930)

[5.3.17 gpfdist 79](#_Toc74742931)

[5.3.18 gpinitstandby 84](#_Toc74742932)

[5.3.19 gpinitsystem 87](#_Toc74742933)

[5.3.20 gpload 95](#_Toc74742934)

[5.3.21 gplogfilter 110](#_Toc74742935)

[5.3.22 gpmapreduce 113](#_Toc74742936)

[5.3.23 gpmapreduce.yaml 114](#_Toc74742937)

[5.3.24 gpmovemirrors 124](#_Toc74742938)

[5.3.25 gpperfmon\_install 125](#_Toc74742939)

[5.3.26 gppkg 130](#_Toc74742940)

[5.3.27 gprecoverseg 132](#_Toc74742941)

[5.3.28 gpreload 137](#_Toc74742942)

[5.3.29 gprestore 139](#_Toc74742943)

[5.3.30 gpscp 146](#_Toc74742944)

[5.3.31 gpssh 148](#_Toc74742945)

[5.3.32 gpssh-exkeys 152](#_Toc74742946)

[5.3.33 gpstart 155](#_Toc74742947)

[5.3.34 gpstate 157](#_Toc74742948)

[5.3.35 gpstop 162](#_Toc74742949)

[5.3.36 gpsys1 165](#_Toc74742950)

[5.3.37 pg\_config 166](#_Toc74742951)

[5.3.38 pg\_dump 168](#_Toc74742952)

[5.3.39 pg\_dumpall 181](#_Toc74742953)

[5.3.40 pg\_restore 187](#_Toc74742954)

[5.3.41 pgbouncer 194](#_Toc74742955)

[5.3.42 pgbouncer.ini 196](#_Toc74742956)

[5.3.43 pgbouncer-admin 211](#_Toc74742957)

[5.3.44 psql 221](#_Toc74742958)

[5.3.45 pxf 256](#_Toc74742959)

[5.3.46 pxf cluster 258](#_Toc74742960)

[5.3.47 reindexdb 260](#_Toc74742961)

[5.3.48 vacuumdb 262](#_Toc74742962)

ЗАПУСК И ОСТАНОВКА RT.WAREHOUSE

В СУБД RT.Warehouse инстансы серверов базы данных (мастер и все сегменты) запускаются или останавливаются на всех хостах в системе таким образом, что они могут работать вместе как единая СУБД.

Поскольку система RT.Warehouse распределена по множеству машин, процесс запуска и остановки системы RT.Warehouse отличается от процесса запуска и остановки обычной СУБД PostgreSQL.

Используйте утилиты gpstart и gpstop для запуска и остановки RT.Warehouse соответственно. Эти утилиты находятся в каталоге $GPHOME/bin на хосте мастера RT.Warehouse.

|  |
| --- |
| **Внимание.**  Не используйте команду kill для завершения любого процесса Postgres. Вместо этого используйте команду базы данных pg\_cancel\_backend().  Выполнение kill -9 или kill -11 может привести к повреждению базы данных и помешать выполнению анализа первопричин. |

Для получения информации о gpstart и gpstop см. Справочник по утилитам.

Запуск RT.Warehouse

Запустите инициализированную систему RT.Warehouse, запустив утилиту gpstart на инстансе мастера.

Используйте утилиту gpstart для запуска системы RT.Warehouse, которая уже была инициализирована утилитой gpinitsystem, но остановлена утилитой gpstop. Утилита gpstart запускает RT.Warehouse, запустив все инстансы базы данных Postgres в кластере RT.Warehouse. gpstart управляет этим процессом и выполняет его параллельно.

Запустите gpstart на хосте мастера, чтобы запустить RT.Warehouse:

|  |
| --- |
| $ gpstart |

Перезапуск RT.Warehouse

Остановите систему RT.Warehouse, а затем перезапустите её.

Утилита gpstop с параметром -r может остановить, а затем перезапустить RT.Warehouse после завершения работы.

Чтобы перезапустить RT.Warehouse, введите следующую команду на хосте мастера:

|  |
| --- |
| $ gpstop -r |

Повторная загрузка только изменений файла конфигурации

Повторно загрузите изменения в файлы конфигурации RT.Warehouse, не прерывая работу системы.

Утилита gpstop может перезагружать изменения в файл конфигурации pg\_hba.conf и параметры времени выполнения в файле мастера postgresql.conf и файле pg\_hba.conf без прерывания обслуживания. Активные сеансы принимают изменения при повторном подключении к базе данных. Многие параметры конфигурации сервера требуют полной перезагрузки системы (gpstop -r) для активации.

Перезагрузите изменения файла конфигурации, не выключая систему, с помощью утилиты gpstop:

|  |
| --- |
| $ gpstop -u |

Запуск мастера в режиме обслуживания

Запустите только мастер для выполнения задач обслуживания или администрирования, не затрагивая данные в сегментах.

Например, вы можете подключиться к базе данных только на инстансе мастера в режиме обслуживания и редактировать настройки системного каталога.

Запустите gpstart с параметром -m:

|  |
| --- |
| $ gpstart -m |

Подключитесь к мастеру в режиме обслуживания для обслуживания каталога. Например:

|  |
| --- |
| $ PGOPTIONS='-c gp\_session\_role=utility' psql postgres |

После завершения административных задач остановите мастер в режиме утилит. Затем перезапустите его в производственном режиме.

|  |
| --- |
| $ gpstop -mr |

|  |
| --- |
| **Внимание.**  Неправильное использование подключений в режиме обслуживания может привести к несогласованному состоянию системы. |

Остановка RT.Warehouse

Утилита gpstop останавливает или перезапускает вашу систему RT.Warehouse и всегда запускается на хосте мастера. При активации gpstop останавливает все процессы postgres в системе, включая мастер и все инстансы сегментов. Утилита gpstop по умолчанию использует до 64 параллельных рабочих потоков для остановки инстансов Postgres, составляющих кластер RT.Warehouse. Перед завершением работы система ожидает завершения всех активных транзакций. Чтобы немедленно остановить RT.Warehouse, используйте режим fast.

Чтобы остановить RT.Warehouse:

|  |
| --- |
| $ gpstop |

Чтобы остановить RT.Warehouse в режиме fast:

|  |
| --- |
| $ gpstop -M fast |

По умолчанию вам не разрешено останавливать RT.Warehouse, если есть какие-либо клиентские подключения к базе данных. Используйте параметр -M fast, чтобы откатить все выполняющиеся транзакции и разорвать все соединения перед завершением работы.

Остановка клиентских процессов

RT.Warehouse запускает новый внутренний процесс для каждого клиентского подключения. Пользователь RT.Warehouse с привилегиями superuser может отменить и завершить эти клиентские внутренние процессы.

Отмена внутреннего процесса с помощью функции pg\_cancel\_backend() завершает конкретный поставленный в очередь или активный клиентский запрос. Завершение внутреннего процесса с помощью функции pg\_terminate\_backend() завершает клиентское соединение с базой данных.

Функция pg\_cancel\_backend() имеет две сигнатуры:

pg\_cancel\_backend( pid int4 );

pg\_cancel\_backend( pid int4, msg text ).

Функция pg\_terminate\_backend() имеет две похожие сигнатуры:

pg\_terminate\_backend( pid int4 );

pg\_terminate\_backend( pid int4, msg text ).

Если вы предоставите msg, RT.Warehouse включает текст в сообщения об отмене, который возвращается клиенту. msg ограничен 128 байтами. RT.Warehouse обрезает всё, что больше.

Функции pg\_cancel\_backend() и pg\_terminate\_backend() возвращают true в случае успеха и false в ином случае.

Чтобы отменить или завершить серверный процесс, вы должны сначала определить идентификатор процесса серверной части. Вы можете получить идентификатор процесса из столбца pid представления pg\_stat\_activity. Например, чтобы просмотреть информацию о процессе, связанную со всеми запущенными запросами и запросами в очереди:

|  |
| --- |
| =# SELECT usename, pid, waiting, state, query, datname  FROM pg\_stat\_activity; |

Пример частичного вывода запроса:

|  |
| --- |
| usename | pid | waiting | state | query | datname  ---------+----------+---------+--------+------------------------+---------  sammy | 31861 | f | idle | SELECT \* FROM testtbl; | testdb  billy | 31905 | t | active | SELECT \* FROM topten; | testdb |

Используйте выходные данные для определения идентификатора процесса (pid) запроса или клиентского соединения.

Например, чтобы отменить ожидающий запрос, указанный в приведённом выше примере выходных данных, и включить 'Admin canceled long-running query.' в качестве возвращаемого сообщения клиенту:

|  |
| --- |
| =# SELECT pg\_cancel\_backend(31905 ,'Admin canceled long-running query.');  ERROR: canceling statement due to user request: "Admin canceled long-running query." |

ДОСТУП К RT.WAREHOUSE

В этом разделе описываются различные клиентские инструменты, которые вы можете использовать для подключения к RT.Warehouse, а как установить подключение.

Подключение к базе данных

Пользователи могут подключаться к RT.Warehouse с помощью клиентской программы, совместимой с PostgreSQL, например psql. Пользователи и администраторы всегда подключаются к RT.Warehouse через машину мастера, сегменты не могут принимать клиентские соединения.

Чтобы установить соединение с сервером мастера RT.Warehouse, вам необходимо знать следующую информацию о подключении и соответствующим образом настроить клиентскую программу.

Таблица — Параметры подключения

| Параметр подключения | Описание | Переменная окружения |
| --- | --- | --- |
| Имя приложения | Имя приложения, которое подключается к базе данных. Значение по умолчанию, содержащееся в параметре подключения application\_name, —psql. | $PGAPPNAME |
| Имя базы данных | Имя базы данных, к которой вы хотите подключиться. Для вновь инициализированной системы используйте базу данных postgres для первого подключения. | $PGDATABASE |
| Имя хоста | Имя хоста сервера мастера RT.Warehouse. Хост по умолчанию — это локальный хост. | $PGHOST |
| Порт | Номер порта, на котором работает инстанс мастера RT.Warehouse. По умолчанию 5432. | $PGPORT |
| Имя пользователя | Имя пользователя (роли) базы данных для подключения. Это не обязательно имя пользователя вашей ОС. Если вы не уверены, какое имя пользователя базы данных у вас, обратитесь к администратору RT.Warehouse. Обратите внимание, что каждая система RT.Warehouse имеет одну учётную запись суперпользователя, которая создается автоматически во время инициализации. Эта учётная запись имеет то же имя, что и имя пользователя ОС, инициализировавшего систему RT.Warehouse (обычно gpadmin). | $PGUSER |

Поддерживаемые клиентские приложения

Пользователи могут подключаться к RT.Warehouse с помощью различных клиентских приложений:

1. Ряд клиентских приложений RT.Warehouse входит в комплект поставки RT.Warehouse. Клиентское приложение psql предоставляет интерактивный интерфейс командной строки для RT.Warehouse.

Используя стандартные интерфейсы приложений баз данных, такие как ODBC и JDBC, пользователи могут создавать свои собственные клиентские приложения, которые взаимодействуют с RT.Warehouse.

Большинство клиентских инструментов, использующих стандартные интерфейсы базы данных, такие как ODBC и JDBC, можно настроить для подключения к RT.Warehouse.

Клиентские приложения RT.Warehouse

RT.Warehouse поставляется с установленным рядом клиентских служебных приложений, расположенных в каталоге $GPHOME/bin вашей установки хоста мастера RT.Warehouse. Ниже приведены наиболее часто используемые клиентские служебные приложения:

Таблица — Часто используемые клиентские приложения

| Имя | Использование |
| --- | --- |
| createdb | Создание новой базы данных. |
| createlang | Определение нового процедурного языка. |
| createuser | Определение новуой роли базы данных. |
| dropdb | Удаление базы данных. |
| droplang | Удаление процедурного языка. |
| dropuser | Удаление роли. |
| psql | Интерактивный терминал PostgreSQL. |
| reindexdb | Переиндексация базы данных. |
| vacuumdb | Сбор мусора и анализ базы данных. |

|  |
| --- |
| **Примечание.**  createlang и droplang устарели и могут быть удалены в будущем выпуске. |

При использовании этих клиентских приложений вы должны подключиться к базе данных через инстанс мастера RT.Warehouse. Вам нужно будет знать имя вашей целевой базы данных, имя хоста и номер порта мастера, а также имя пользователя базы данных для подключения. Эта информация может быть предоставлена ​​в командной строке с помощью параметров -d, -h, -p и -U соответственно. Если найден аргумент, не принадлежащий ни одному параметру, он будет сначала интерпретирован как имя базы данных.

Все эти параметры имеют значения по умолчанию, которые будут использоваться, если параметр не указан. Хост по умолчанию — это локальный хост. Номер порта по умолчанию — 5432. Имя пользователя по умолчанию — это имя пользователя вашей ОС, как и имя базы данных по умолчанию. Обратите внимание, что имена пользователей ОС и имена пользователей RT.Warehouse не обязательно совпадают.

Если значения по умолчанию неверны, вы можете установить для переменных окружения PGDATABASE, PGHOST, PGPORT и PGUSER соответствующие значения или использовать файл psql ~/.pgpass для хранения часто используемых паролей.

Подключение с помощью psql

В зависимости от используемых значений по умолчанию или установленных вами переменных окружения следующие примеры показывают, как получить доступ к базе данных через psql:

|  |
| --- |
| $ psql -d gpdatabase -h master\_host -p 5432 -U gpadmin |

|  |
| --- |
| $ psql gpdatabase |

|  |
| --- |
| $ psql |

Если пользовательская база данных ещё не создана, вы можете получить доступ к системе, подключившись к базе данных postgres. Например:

|  |
| --- |
| $ psql postgres |

После подключения к базе данных psql отображает подсказку с именем базы данных, к которой в настоящее время подключён psql, за которым следует строка => (или =#, если вы являетесь суперпользователем базы данных). Например:

|  |
| --- |
| gpdatabase => |

В командной строке вы можете ввести команды SQL. Команда SQL должна заканчиваться на ; (точка с запятой) для отправки на сервер и выполнения. Например:

|  |
| --- |
| => SELECT \* FROM mytable; |

Использование пула подключений PgBouncer

Утилита PgBouncer управляет пулами соединений для соединений PostgreSQL и RT.Warehouse.

В следующих разделах описывается, как настроить и использовать PgBouncer с RT.Warehouse. Обратитесь к [веб-сайту PgBouncer](https://pgbouncer.github.io/) для получения информации об использовании PgBouncer с PostgreSQL.

Обзор

Пул соединений с базой данных — это кэш соединений с базой данных. После того, как пул подключений создан, пул подключений минимизирует издержки на создание новых подключений к базе данных, поэтому клиенты подключаются намного быстрее и нагрузка на сервер снижается.

PgBouncer — это лёгкий менеджер пула соединений для RT.Warehouse и PostgreSQL. PgBouncer поддерживает пул соединений для каждой комбинации базы данных и пользователя. PgBouncer либо создаёт новое соединение с базой данных для клиента, либо повторно использует существующее соединение для того же пользователя и базы данных. Когда клиент отключается, PgBouncer возвращает соединение в пул для повторного использования.

PgBouncer разделяет соединения в одном из трёх режимов пула:

Пул сеансов — когда клиент подключается, ему назначается одно серверное подключение на всё время, пока клиент остаётся подключённым. Когда клиент отключается, это подключение к серверу возвращается в пул;

Пул транзакций — подключение к серверу назначается клиенту только на время транзакции. Когда PgBouncer замечает, что транзакция завершена, это подключение возвращается в пул. Этот режим можно использовать только с приложениями, которые не используют функции, зависящие от сеанса;

Пул операторов — подключение к серверу будет возвращаться в пул сразу после завершения каждого запроса. Транзакции с несколькими операторами в этом режиме запрещаются, так как они не будут работать. Этот режим предназначен для обеспечения режима автоматической фиксации на клиенте и предназначен для PL/Proxy в PostgreSQL.

Вы можете установить режим пула по умолчанию для инстанса PgBouncer. Вы можете переопределить этот режим для отдельных баз данных и пользователей.

PgBouncer поддерживает стандартный интерфейс подключения, используемый PostgreSQL и RT.Warehouse. Клиентское приложение RT.Warehouse (например, psql) подключается к хосту и порту, на котором работает PgBouncer, а не к хосту и порту мастера RT.Warehouse.

PgBouncer включает консоль администрирования, подобную psql. Авторизованные пользователи могут подключаться к виртуальной базе данных для мониторинга и управления PgBouncer. Вы можете управлять процессом демона PgBouncer через консоль администратора. Вы также можете использовать консоль для обновления и перезагрузки конфигурации PgBouncer во время выполнения без остановки и перезапуска процесса.

PgBouncer изначально поддерживает TLS.

Перенос PgBouncer

При переходе на новую версию RT.Warehouse необходимо перенести свой инстанс PgBouncer на инстанс в новой установке RT.Warehouse.

1. Если вы переходите на версию RT.Warehouse 5.8.x или более раннюю, вы можете выполнить миграцию PgBouncer, не разрывая соединения. Запустите новый процесс PgBouncer с параметром -R и файлом конфигурации, с которого вы запустили процесс:

|  |
| --- |
| $ pgbouncer -R -d pgbouncer.ini |

Параметр -R (reboot) заставляет новый процесс подключаться к консоли старого процесса через сокет Unix и выполнять следующие команды:

|  |
| --- |
| SUSPEND;  SHOW FDS;  SHUTDOWN; |

Когда новый процесс обнаруживает, что старый процесс завершён он возобновляет работу со старыми соединениями. Это возможно, потому что команда SHOW FDS отправляет фактические файловые дескрипторы новому процессу. Если переход не удастся по какой-либо причине, завершите новый процесс, и старый процесс возобновится.

Если вы переходите на RT.Warehouse версии 5.9.0 или новее, вы должны закрыть инстанс PgBouncer в своей старой установке и перенастроить и перезапустить PgBouncer в новой установке.

Если вы использовали stunnel для защиты соединений PgBouncer в своей старой установке, вы должны настроить SSL/TLS в новой установке, используя встроенные возможности TLS в PgBouncer 1.8.1 и более поздних версиях.

Если вы использовали аутентификацию LDAP в своей старой установке, вы должны настроить LDAP в новой установке, используя встроенные возможности интеграции PAM в PgBouncer 1.8.1 и более поздних версиях. Вы также должны удалить или заменить все строки пароля с префиксом ldap:// в auth\_file.

Настройка PgBouncer

Вы настраиваете PgBouncer и его доступ к RT.Warehouse через файл конфигурации. Этот файл конфигурации, обычно называемый pgbouncer.ini, предоставляет информацию о расположении баз данных RT.Warehouse. Файл pgbouncer.ini также определяет процесс, пул соединений, авторизованных пользователей и конфигурацию аутентификации для PgBouncer.

Пример содержимого файла pgbouncer.ini:

|  |
| --- |
| [databases]  postgres = host=127.0.0.1 port=5432 dbname=postgres  pgb\_mydb = host=127.0.0.1 port=5432 dbname=mydb  [pgbouncer]  pool\_mode = session  listen\_port = 6543  listen\_addr = 127.0.0.1  auth\_type = md5  auth\_file = users.txt  logfile = pgbouncer.log  pidfile = pgbouncer.pid  admin\_users = gpadmin |

См. pgbouncer.ini для получения информации о формате файла конфигурации PgBouncer и списке свойств конфигурации, которые он поддерживает.

Когда клиент подключается к PgBouncer, пул соединений ищет конфигурацию для запрошенной базы данных (которая может быть псевдонимом для фактической базы данных), которая была указана в файле конфигурации pgbouncer.ini, чтобы найти имя хоста, порт и базу данных. имя для подключения к базе данных. Файл конфигурации также определяет режим аутентификации, действующий для базы данных.

PgBouncer требует файл аутентификации — текстовый файл, содержащий список пользователей и пароли RT.Warehouse. Содержимое файла зависит от auth\_type, который вы настраиваете в файле pgbouncer.ini. Пароли могут быть в виде открытого текста или строк в кодировке MD5. Вы также можете настроить PgBouncer для запроса целевой базы данных для получения информации о паролях для пользователей, которых нет в файле аутентификации.

Формат файла аутентификации PgBouncer

PgBouncer требует собственного файла аутентификации пользователя. Вы указываете имя этого файла в свойстве auth\_file конфигурационного файла pgbouncer.ini. auth\_file — это текстовый файл следующего формата:

|  |
| --- |
| "username1" "password" ...  "username2" "md5abcdef012342345" ... |

auth\_file содержит по одной строке для каждого пользователя. Каждая строка должна иметь как минимум два поля, каждое из которых заключено в двойные кавычки (""). Первое поле определяет имя пользователя RT.Warehouse. Второе поле — это либо обычный текст, либо пароль в кодировке MD5. PgBouncer игнорирует оставшуюся часть строки.

Формат auth\_file аналогичен формату текстового файла pg\_auth, который RT.Warehouse использует для информации об аутентификации. PgBouncer может работать напрямую с этим файлом аутентификации RT.Warehouse.

Используйте пароль в кодировке MD5. Формат пароля в кодировке MD5:

|  |
| --- |
| "md5" + MD5\_encoded(<password><username>) |

Вы также можете получить пароли в кодировке MD5 всех пользователей RT.Warehouse из представления pg\_shadow.

Настройка аутентификации на основе HBA для PgBouncer

PgBouncer поддерживает аутентификацию на основе HBA. Чтобы настроить аутентификацию на основе HBA для PgBouncer, установите auth\_type=hba в файле конфигурации pgbouncer.ini. Также укажите имя файла формата HBA в параметре auth\_hba\_file файла pgbouncer.ini.

Пример содержимого файла HBA PgBouncer с именем hba\_bouncer.conf:

|  |
| --- |
| local all bouncer trust  host all bouncer 127.0.0.1/32 trust |

Пример отрывка из связанного файла конфигурации pgbouncer.ini:

|  |
| --- |
| [databases]  p0 = port=15432 host=127.0.0.1 dbname=p0 user=bouncer pool\_size=2  p1 = port=15432 host=127.0.0.1 dbname=p1 user=bouncer  ...  [pgbouncer]  ...  auth\_type = hba  auth\_file = userlist.txt  auth\_hba\_file = hba\_bouncer.conf  ... |

Обратитесь к обсуждению [формата файла HBA](https://pgbouncer.github.io/config.html#hba-file-format) в документации PgBouncer для получения информации о поддержке PgBouncer формата файла аутентификации HBA.

Запуск PgBouncer

Вы можете запустить PgBouncer на сервере мастера RT.Warehouse или на другом сервере. Если вы устанавливаете PgBouncer на отдельный сервер, вы можете легко переключить клиентов на резервный мастер, обновив файл конфигурации PgBouncer и перезагрузив конфигурацию с помощью консоли администрирования PgBouncer.

Выполните следующие действия, чтобы настроить PgBouncer:

1. Создайте файл конфигурации PgBouncer. Например, добавьте следующий текст в файл с именем pgbouncer.ini:

|  |
| --- |
| [databases]  postgres = host=127.0.0.1 port=5432 dbname=postgres  pgb\_mydb = host=127.0.0.1 port=5432 dbname=mydb  [pgbouncer]  pool\_mode = session  listen\_port = 6543  listen\_addr = 127.0.0.1  auth\_type = md5  auth\_file = users.txt  logfile = pgbouncer.log  pidfile = pgbouncer.pid  admin\_users = gpadmin |

В файле перечислите базы данных и детали их подключения. Файл также настраивает инстанс PgBouncer. См. pgbouncer.ini для получения информации о формате и содержимом файла конфигурации PgBouncer.

Создайте файл аутентификации. Имя файла должно быть именем, указанным вами для параметра auth\_file файла pgbouncer.ini — users.txt. Каждая строка содержит имя пользователя и пароль. Формат строки пароля соответствует auth\_type, который вы настроили в файле конфигурации PgBouncer. Если параметр auth\_type является plain, строка пароля представляет собой открытый текстовый пароль, например:

|  |
| --- |
| "gpadmin" "gpadmin1234" |

Если auth\_type как в примере — md5, поле аутентификации должно быть закодировано MD5. Формат пароля в кодировке MD5:

|  |
| --- |
| "md5" + MD5\_encoded(<password><username>) |

Запустите pgbouncer:

|  |
| --- |
| $ $GPHOME/bin/pgbouncer -d pgbouncer.ini |

Параметр -d запускает PgBouncer как фоновый (демон) процесс. См. pgbouncer, чтобы узнать о синтаксисе и параметрах команды pgbouncer.

Обновите клиентские приложения для подключения к pgbouncer, а не напрямую к серверу RT.Warehouse. Например, чтобы подключиться к настроенной выше RT.Warehouse с именем mydb, запустите psql следующим образом:

|  |
| --- |
| $ psql -p 6543 -U someuser pgb\_mydb |

Значение параметра -p — это порт прослушивания, который вы настроили для инстанса PgBouncer.

Управление PgBouncer

PgBouncer предоставляет консоль администрирования, подобную psql. Войдите в консоль администрирования PgBouncer, указав номер порта PgBouncer и виртуальную базу данных с именем pgbouncer. Консоль принимает команды, подобные SQL, которые можно использовать для мониторинга, перенастройки и управления PgBouncer.

Полную документацию по командам консоли администрирования PgBouncer см. в pgbouncer-admin.

Выполните следующие действия, чтобы начать работу с консолью администрирования PgBouncer.

1. С помощью psql войдите в виртуальную базу данных pgbouncer:

|  |
| --- |
| $ psql -p 6543 -U username pgbouncer |

Указанное вами имя пользователя должно быть указано в параметре admin\_users в файле конфигурации pgbouncer.ini. Вы также можете войти в консоль администрирования PgBouncer с текущим именем пользователя Unix, если процесс pgbouncer выполняется под UID этого пользователя.

Чтобы просмотреть доступные команды консоли администрирования PgBouncer, запустите команду SHOW help:

|  |
| --- |
| pgbouncer=# SHOW help;  NOTICE: Console usage  DETAIL:  SHOW HELP|CONFIG|DATABASES|POOLS|CLIENTS|SERVERS|VERSION  SHOW FDS|SOCKETS|ACTIVE\_SOCKETS|LISTS|MEM  SHOW DNS\_HOSTS|DNS\_ZONES  SHOW STATS|STATS\_TOTALS|STATS\_AVERAGES  SET key = arg  RELOAD  PAUSE [<db>]  RESUME [<db>]  DISABLE <db>  ENABLE <db>  KILL <db>  SUSPEND  SHUTDOWN |

Если вы обновляете конфигурацию PgBouncer, редактируя файл конфигурации pgbouncer.ini, используйте команду RELOAD для перезагрузки файла:

|  |
| --- |
| pgbouncer=# RELOAD; |

Сопоставление клиентов PgBouncer с подключениями к серверу RT.Warehouse

Чтобы сопоставить клиент PgBouncer с подключением к серверу RT.Warehouse, используйте команды консоли администрирования PgBouncer SHOW CLIENTS и SHOW SERVERS:

1. С помощью ptr и link сопоставьте подключение локального клиента с подключением к серверу.

Используйте адрес и порт клиентского соединения, чтобы идентифицировать TCP-соединение от клиента.

Используйте local\_addr и local\_port, чтобы определить TCP-соединение с сервером.

Интерфейсы приложений базы данных

Возможно, вы захотите разработать свои собственные клиентские приложения, которые будут взаимодействовать с RT.Warehouse. PostgreSQL предоставляет ряд драйверов баз данных для наиболее часто используемых API баз данных, которые также можно использовать с RT.Warehouse. Эти драйверы доступны для отдельной загрузки. Каждый драйвер (кроме libpq, который поставляется с PostgreSQL) является независимым проектом разработки PostgreSQL и должен быть загружен, установлен и настроен для подключения к RT.Warehouse. Доступны драйверы, представленные в таблице.

Таблица — Интерфейсы RT.Warehouse

| API | Драйвер PostgreSQL | Ссылка для загрузки драйвера |
| --- | --- | --- |
| ODBC | psqlODBC | <https://odbc.postgresql.org/> |
| JDBC | pgjdbc | <https://jdbc.postgresql.org/> |
| Perl DBI | pgperl | <https://metacpan.org/release/DBD-Pg> |
| Python DBI | pygresql | <http://www.pygresql.org/> |
| libpq C Library | libpq | <https://www.postgresql.org/docs/9.4/libpq.html> |

Загрузите платформу языка программирования и соответствующий API из соответствующего источника. Например, вы можете получить Java Development Kit (JDK) и JDBC API от Oracle.

Напишите свое клиентское приложение в соответствии со спецификациями API. При программировании приложения помните о поддержке SQL в RT.Warehouse, не включайте неподдерживаемый синтаксис SQL.

Загрузите соответствующий драйвер и настройте подключение к инстансу мастера RT.Warehouse.

Устранение проблем с подключением

Ряд вещей может помешать успешному подключению клиентского приложения к RT.Warehouse. В этом разделе объясняются некоторые из распространённых причин проблем с подключением и способы их устранения.

Таблица — Общие проблемы с подключением

| Проблема | Решение |
| --- | --- |
| Отсутствует запись в pg\_hba.conf для хоста или пользователя | Чтобы RT.Warehouse могла принимать удалённые клиентские подключения, необходимо настроить гинстанс мастера RT.Warehouse таким образом, чтобы разрешались подключения от клиентских хостов и пользователей базы данных, которые будут подключаться к RT.Warehouse. Это делается путём добавления соответствующих записей в файл конфигурации pg\_hba.conf (расположенный в каталоге данных инстанса мастера). |
| RT.Warehouse не работает | Если инстанс мастера RT.Warehouse не работает, пользователи не смогут подключиться. Вы можете проверить, что система RT.Warehouse запущена, запустив утилиту gpstate на хосте мастера RT.Warehouse. |
| Проблемы с сетью, таймауты интерконнекта | Если пользователи подключаются к хосту мастера RT.Warehouse с удалённого клиента, сетевые проблемы могут помешать подключению (например, проблемы с разрешением имени хоста DNS, хост-система не работает и т.д.). Чтобы убедиться, что сетевые проблемы не являются причиной, подключитесь к хосту мастера RT.Warehouse с удалённого клиентского хоста. Например: ping *hostname*.  Если система не может разрешить имена хостов и IP-адреса хостов, задействованных в RT.Warehouse, запросы и соединения не будут выполнены. Для некоторых операций соединения с мастером RT.Warehouse используют localhost, а другие используют фактическое имя хоста, поэтому вы должны иметь возможность разрешить оба. Если вы столкнулись с этой ошибкой, сначала убедитесь, что вы можете подключиться к каждому хосту в массиве RT.Warehouse с хоста мастера по сети. Убедитесь, что в файле /etc/hosts мастера и всех сегментов указаны правильные имена хостов и IP-адреса для всех хостов, входящих в массив RT.Warehouse. IP-адрес 127.0.0.1 должен разрешаться в localhost. |
| Слишком много клиентов | По умолчанию RT.Warehouse настроена так, чтобы разрешить до 250 одновременных подключений пользователей к мастеру и 750 к сегменту. Попытка подключения, которая приводит к превышению этого лимита, будет отклонена. Этот предел контролируется параметром max\_connections в файле конфигурации postgresql.conf сервера мастера RT.Warehouse. Если вы измените этот параметр для мастера, вы также должны внести соответствующие изменения в сегменты. |

КОНТРОЛЬ ЦЕЛОСТНОСТИ В RT.WAREHOUSE (ИЗОЛЯЦИЯ ТРАНЗАКЦИЙ)

Контроль целостности позволяет согласовать конкурентные запросы с правильными результатами, обеспечивая целостность базы данных. Традиционные базы данных используют протокол двухфазной блокировки, что предотвращает транзакцию от изменения данных, которые были прочитаны другой параллельной транзакцией, и не позволяет любой параллельной транзакции считывать или записывать данные, обновленные другой транзакцией. Блокировки, необходимые для координации транзакций, добавляют нагрузку на базу данных, уменьшая общую пропускную способность.

В RT.Warehouse используется PostgreSQL Multiversion Concurrency Control (MVCC) для управления параллельными транзакциями heap-таблиц. С MVCC каждый запрос работает со снапшотом базы данных, создаваемым в тот момент, когда запрос объявляется. Пока выполняется снапшот, запрос не может видеть сделанные другими параллельными транзакциями изменения. Это гарантирует, что запрос видит последовательное представление базы данных. Запросы, которые читают строки, никогда не блокируют транзакции, пишущие строки. И наоборот, запросы, которые записывают строки, не могут быть заблокированы транзакциями, которые читают строки. Это позволяет значительно увеличить параллелизм в сравнении с традиционными базами данных, в которых используется блокировка для координации доступа между транзакциями, которые считывают и записывают данные.

|  |
| --- |
| **Внимание.**  Append optimized таблицы управляются с помощью иной модели управления параллелизмом, нежели чем модель MVCC, которая обсуждалась в данном разделе. Они предназначены для приложений write-once, read-many, что никогда (или только очень редко) подразумевает обновления на уровне строк. |

Снапшоты

Модель MVCC основывается на способности системы управлять несколькими версиями строк данных. Запрос работает со снапшотом базы данных. Снапшот — это набор строк, которые видны в момент транзакции. Снапшот гарантирует, что запрос имеет действительный и корректный снимок базы данных на время выполнения запроса или транзакции.

Каждой транзакции присваивается уникальный идентификатор (XID), 32-битное значение. Оператор SQL, не являющийся частью транзакции, обрабатывается как одиночная транзакция (с добавлением BEGIN и COMMIT). Это похоже на autocommit, используемый в некоторых базах данных.

Когда транзакция вставляет строку, XID сохраняется со строкой в столбце xmin. Когда транзакция удаляет строку, XID сохраняется в столбце системы xmax. Обновление строки рассматривается как delete и insert, поэтому XID сохраняется в xmax текущей строки и xmin только что вставленного ряда. Столбцы xmin и xmax вместе со статусом завершения транзакции определяют диапазон транзакций, для которых видна текущая версия строки. Транзакция видит результаты работы всех транзакций меньше, чем xmin, но не может видеть результаты работы любой транзакции больше или равной xmax.

Операции с несколькими действиями также должны записывать, какая команда в транзакции вставляла строку (cmin) или удаляла строку (cmax), чтобы транзакция могла видеть изменения, сделанные предыдущими командами в транзакции. Последовательность команд имеет смысл только во время выполнения транзакции, поэтому она сбрасывается до 0 в самом начале.

Каждый сегмент RT.Warehouse имеет свою собственную последовательность XID, которая не может сравниваться с XID других инcтансов. Мастер координирует распределённые транзакции с сегментами, использующими идентификационный номер сеанса, называемый gp\_session\_id. Сегменты сопоставляют идентификаторы распределённых транзакций с их локальными XID. Мастер координирует распределённые транзакции с помощью двухфазового протокола фиксации. Если транзакция не выполняется на одном из сегментов, транзакция прекращается на всех сегментах, и данные возвращаются в первоначальный вид.

Столбцы xmin, xmax, cmin и cmax для любой строки можно увидеть при помощи оператора SELECT:

|  |
| --- |
| SELECT xmin, xmax, cmin, cmax, \* FROM tablename; |

Поскольку команда SELECT выполняется на мастере, XID являются идентификаторами распределённых транзакций. В случае если команда выполняется в отдельной базе данных сегментов, значения xmin и xmax соответствуют локальным значениям XID.

Зацикливание ID транзакций

Модель MVCC использует идентификаторы транзакций (XID), чтобы определить, какие строки видны в начале запроса или транзакции. XID — это 32-битное значение, поэтому база данных может теоретически выполнить более четырех миллиардов транзакций до того, как значение переполнится и обнулится. Тем не менее, в RT.Warehouse используется арифметика по модулю 2^32, что позволяет XID зацикливаться. Для любого XID может быть около двух миллиардов предыдущих и новых XID. Это работает до тех пор, пока текущая версия строки примерно через два миллиарда транзакций неожиданно не станет новой строкой. Чтобы предотвратить это, RT.Warehouse имеет специальный XID, называемый FrozenXID, который всегда считается старше обычного XID, с которым он сравнивается. xmin строки должен быть заменён на FrozenXID в течение двух миллиардов транзакций, и это одна из функций, выполняемых командой VACUUM.

Очистка (vacuuming) как минимум раз в два миллиарда транзакций предотвращает зацикливание XID. RT.Warehouse отслеживает XID и предупреждает, когда требуется произвести очистку (операция VACUUM).

Когда значительная часть идентификаторов больше недоступна и до того, как происходит зацикливание XID, выдается предупреждение:

|  |
| --- |
| WARNING: database "database\_name" must be vacuumed within number\_of\_transactions  Transactions |

Если операция VACUUM не выполняется, RT.Warehouse перестаёт создавать транзакции, чтобы избежать возможной потери данных. При достижении лимита, система сообщает об ошибке:

|  |
| --- |
| FATAL: database is not accepting commands to avoid wraparound data loss in database  "database\_name" |

Параметры конфигурации сервера xid\_warn\_limit и xid\_stop\_limit управляют отображением предупреждений и ошибок. Параметр xid\_warn\_limit показывает количество идентификаторов транзакций перед значением xid\_stop\_limit. А параметр xid\_stop\_limit показывает количество XID перед тем, как происходит зацикливание и выдаётся ошибка.

Режимы изоляции транзакций

Стандарт SQL описывает три явления, которые могут возникать при запуске транзакций базы данных одновременно:

Грязное чтение — явление, которое возникает, когда транзакция считывает незафиксированные данные из другой параллельной транзакции;

Неповторяющееся чтение — ситуация, когда при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными;

Чтение фантомов — ситуация, когда при повторном чтении в рамках одной транзакции одна и та же выборка даёт разные множества строк.

Стандарт SQL определяет четыре режима изоляции транзакций, которые должны поддерживать системы баз данных:

Команды SQL RT.Warehouse позволяют запросить READ UNCOMMITTED, READ COMMITTED или SERIALIZABLE. RT.Warehouse рассматривает READ UNCOMMITTED так же, как READ COMMITTED. Запрос REPEATABLE READ вызывает ошибку, вместо этого необходимо использовать SERIALIZABLE. Режим изоляции по умолчанию — READ COMMITTED.

Разница между READ COMMITTED и SERIALIZABLE заключается в том, что в режиме READ COMMITTED каждый оператор в транзакции видит только строки, обновлённые до начала выполнения оператора, в то время как в режиме SERIALIZABLE все операторы транзакции видят строки, обновлённые до начала транзакции.

Режим изоляции READ COMMITTED обеспечивает лучшее распараллеливание и лучшую производительность, чем режим SERIALIZABLE. Он допускает неповторяющиеся чтения, где значения в строке, полученные дважды в транзакции, могут отличаться, поскольку другая параллельная транзакция совершила изменения. Режим READ COMMITTED также позволяет делать фантомные чтения, где запрос, выполненный дважды в одной и той же транзакции, может возвращать два разных набора строк.

Режим изоляции SERIALIZABLE предотвращает как неповторяющиеся чтения, так и фантомные чтения, но за счёт ухудшения распараллеливания и производительности. Все параллельные транзакции имеют одинаковый снимок базы данных, полученный перед началом транзакций. Параллельная транзакция, которая пытается изменить данные, уже изменённые другой транзакцией, получает запрет на это действие. Приложения, выполняющие транзакции в режиме SERIALIZABLE, должны быть подготовлены к обработке транзакций, не выполняющихся из-за ошибок сериализации. Если режим изоляции SERIALIZABLE жёстко не требуется, лучше использовать режим READ COMMITTED.

Стандарт SQL подразумевает, что параллельные упорядоченные транзакции приводят базу данных в одно состояние в независимости от типа их обработки (т.е. в параллели или друг за другом). Модель изоляции снапшотов MVCC предотвращает грязные чтения, неповторяющиеся чтения и фантомные чтения, не используя блокировку. Однако, иные взаимодействия, которые могут возникнуть между некоторыми транзакциями SERIALIZABLE в RT.Warehouse, не позволяют им называться полностью упорядоченными. Эти аномалии можно отнести к тому факту, что RT.Warehouse не выполняет блокировку предикатов, а это означает, что запись одной транзакции может повлиять на результат предыдущего чтения в другой параллельной транзакции.

Транзакции, которые выполняются одновременно, должны быть проверены на предмет взаимодействий, которые не могут быть предотвращены путём запрета параллельных обновлений одних и тех же данных. Определённые проблемы можно предотвратить, используя блокировки таблиц или потребовав, чтобы конфликтующие транзакции обновили фиктивную строку, введённую для представления конфликта.

Оператор SQL SET TRANSACTION ISOLATION LEVEL устанавливает режим изоляции для текущей транзакции. Режим должен быть установлен перед любыми операциями SELECT, INSERT, DELETE, UPDATE или COPY:

|  |
| --- |
| BEGIN;  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  ...  COMMIT; |

Режим изоляции также может быть указан как часть инструкции BEGIN:

|  |
| --- |
| BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE; |

Режим изоляции транзакции по умолчанию можно изменить с помощью свойства default\_transaction\_isolation.

Удаление неиспользуемых строк из таблиц

Обновление или удаление строки оставляет предыдущую версию строки в таблице. Если строка с истёкшим сроком действия указана в активных транзакциях, её можно удалить, а занимаемое ею пространство повторно использовать. За это действие отвечает команда VACUUM.

Когда недействительные строки накапливаются в таблице, дисковые файлы должны быть расширены для размещения новых строк. Увеличенная нагрузка на ввод/вывод дисков, используемых для обработки запросов, негативно влияет на производительность. Эта ситуация называется раздутие (bloat), и её следует контролировать с помощью регулярной чистки.

Команда VACUUM может работать одновременно с другими запросами. Она удаляет неиспользуемые строки со страниц и переупаковывает оставшиеся строки для консолидации свободного места. Если после очистки осталось много свободного пространства, в таблицу свободного пространства добавляется страница. Позднее, когда базе данных требуется пространство для новых строк, она обращается к карте свободного пространства таблицы. Если ни одна страница не найдена, добавляются новые страницы.

VACUUM не консолидирует страницы и не уменьшает размер таблицы на диске. Пространство, которое эта команда восстанавливает, доступно только через карту свободного пространства. Чтобы размер файлов на диске не рос, важно хотя бы один раз в день запускать VACUUM. Также важно запускать VACUUM после выполнения транзакции, которая обновляет или удаляет большое количество строк.

Команда VACUUM FULL перезаписывает таблицу без неиспользуемых строк, сводя её к минимальному размеру. Для создания новой таблицы должно быть достаточно места на диске. При этом таблица блокируется до тех пор, пока команда VACUUM FULL не завершится. Это очень ресурсоёмко по сравнению с обычной командой VACUUM, и её можно избежать или отложить путём регулярной чистки. Лучше всего запускать VACUUM FULL в течение периода технического обслуживания. Альтернативой VACUUM FULL является воссоздание таблицы с помощью инструкции CREATE TABLE AS с последующим удалением старой таблицы.

Карта свободного пространства находится в общей памяти и отслеживает свободное пространство для всех таблиц и индексов. Каждая таблица или индекс использует около 60 байт памяти, и каждая страница со свободным пространством занимает 6 байт. Два параметра конфигурации системы определяют размер карты свободного пространства:

max\_fsm\_pages — данный параметр устанавливает максимальное количество дисковых страниц, которые могут быть добавлены в общую карту свободного пространства. Для каждого слота страницы потребляется 6 байт общей памяти. Значение по умолчанию — 200000. Этот параметр должен быть установлен как минимум в 16 раз больше значения max\_fsm\_relations;

max\_fsm\_relations — параметр устанавливает максимальное количество отношений, которые отслеживаются в карте свободного пространства. Этот параметр должен быть установлен больше, чем общее количество таблиц + индексов + системных таблиц. Значение по умолчанию — 1000. Для каждого отношения к каждому сегменту потребляется около 60 байт памяти. Рекомендуется устанавливать параметр на более высокое значение.

Если карта свободного пространства слишком маленькая, дисковые страницы с доступным пространством не добавляются на карту, и это пространство не может быть повторно использовано до тех пор, пока не будет запущена, по меньшей мере, следующая команда VACUUM. Это приводит к росту файлов.

Можно запустить VACUUM VERBOSE *tablename*, чтобы получить отчёт по сегменту о количестве удалённых строк, количестве затронутых страниц и количестве страниц с полезным свободным пространством.

Чтобы узнать, сколько страниц таблица использует во всех сегментах, необходимо запросить системную таблицу pg\_class. Чтобы получить точные данные обязательно следует сначала выполнить ANALYZE для таблицы.

|  |
| --- |
| SELECT relname, relpages, reltuples FROM pg\_class WHERE relname = 'tablename'; |

Другим полезным инструментом является gp\_bloat\_diag в схеме gp\_toolkit, который идентифицирует раздутие (bloat) в таблицах путём сравнения фактического количества используемых таблицей страниц с ожидаемым числом.

ПАРАЛЛЕЛЬНАЯ ЗАГРУЗКА ДАННЫХ

В разделе приводится краткое описание методов загрузки данных в RT.Warehouse.

В крупномасштабном хранилище большие объёмы данных должны загружаться за довольно короткий промежуток времени. RT.Warehouse поддерживает быструю параллельную загрузку данных с помощью функции внешних таблиц. Администраторы также могут загружать внешние таблицы в режиме изоляции ошибочных строк, чтобы фильтровать ошибочные строки в отдельную таблицу, продолжая загрузку правильно отформатированных строк. Администраторы могут указать порог ошибок для операции загрузки, чтобы контролировать, какое количество неправильно отформатированных строк заставляет RT.Warehouse прерывать операцию загрузки.

Используя внешние таблицы в сочетании с параллельным файловым сервером (gpfdist) можно достичь максимального распараллеливания и пропускной способности базы данных.

Другая утилита RT.Warehouse gpload запускает задачу загрузки, указанную в управляемом файле в формате YAML. Необходимо описать местоположение исходных данных, формат, необходимые преобразования, участвующие хосты, адресаты баз данных и другие данные в файле управления, после чего gpload выполняет загрузку. Это позволяет описать сложную задачу и выполнить её контролируемым системным образом.

ИНФОРМАЦИЯ ОБ УТИЛИТАХ RT.WAREHOUSE

Раздел содержит общую информацию об использовании служебных программ RT.Warehouse.

Ссылка на IP-адреса

Когда вы ссылаетесь на адреса IPv6 в служебных программах RT.Warehouse или когда используете числовые IP-адреса вместо имён хостов в любой служебной программе управления, всегда заключайте IP-адрес в скобки. При указании IP-адреса в командной строке лучше всего избегать скобок или заключать их в одинарные кавычки. Например, используйте:

|  |
| --- |
| \[2620:0:170:610::11\] |

Или:

|  |
| --- |
| '[2620:0:170:610::11]' |

Запуск программ внутреннего сервера

RT.Warehouse модифицировал некоторые программы внутреннего сервера PostgreSQL для обеспечения параллелизма и распределения системы RT.Warehouse. Вы получаете доступ к этим программам только через инструменты и утилиты управления RT.Warehouse. Не запускайте эти программы напрямую.

В следующей таблице указаны некоторые программы внутреннего сервера PostgreSQL и служебная команда RT.Warehouse, которые следует запускать вместо них.

Таблица — Программы внутреннего сервера RT.Warehouse

| Программа PostgreSQL | Описание | Программа RT.Warehouse, которую следует использовать |
| --- | --- | --- |
| initdb | Программа вызывается gpinitsystem при инициализации массива RT.Warehouse. Он используется внутри компании для создания отдельных инстансов сегмента и инстанса мастера. | gpinitsystem |
| ipcclean | Не используется в RT.Warehouse. | — |
| pg\_basebackup | Программа создаёт бинарную копию единственного инстанса базы данных. RT.Warehouse использует его для таких задач, как создание инстанса резервного мастера или восстановление зеркального сегмента, когда требуется полная копия. Не используйте эту утилиту для резервного копирования инстансов сегментов RT.Warehouse, поскольку она не создаёт резервные копии, согласованные с MPP. | gpinitstandby, gprecoverseg |
| pg\_controldata | Не используется в RT.Warehouse. | gpstate |
| pg\_ctl | Программа вызывается gpstart и gpstop при запуске или остановке массива RT.Warehouse. Она используется внутри для остановки и запуска инстансов отдельных сегментов и инстанса мастера параллельно и с правильными параметрами. | gpstart, gpstop |
| pg\_resetxlog | Не используйте!  **Внимание**. Эта программа может вызвать потерю данных или стать причиной их недоступности. Если эта программа используется, кластер необходимо повторно инициализировать и восстановить. | — |
| postgres | Исполняемый файл postgres — это фактический серверный процесс PostgreSQL, обрабатывающий запросы. | Основной процесс postgres (postmaster) создаёт другие подпроцессы postgres и сеанс postgres по мере необходимости для обработки клиентских подключений. |
| postmaster | postmaster запускает процесс прослушивания сервера базы данных postgres, который принимает клиентские соединения. В RT.Warehouse процесс прослушивания базы данных postgres выполняется на инстансе мастера RT.Warehouse и на каждом инстансе сегмента. | В RT.Warehouse используйте gpstart и gpstop для одновременного запуска всех postmaster (процессов postgres) в системе в правильном порядке и с правильными параметрами. |

Справочник по утилитам

В разделе представлены утилиты командной строки, входящие в состав RT.Warehouse.

RT.Warehouse использует стандартные клиентские и серверные программы PostgreSQL и предоставляет дополнительные служебные программы для администрирования распределённой СУБД RT.Warehouse.

При установке сервера RT.Warehouse устанавливается несколько утилит.

Эти утилиты находятся в $GPHOME/bin.

analyzedb

Утилита, которая выполняет операции ANALYZE над таблицами инкрементально и одновременно. Для добавления оптимизированных таблиц analyzedb обновляет статистику только в том случае, если статистика не актуальна.

Синтаксис

|  |
| --- |
| analyzedb -d *dbname*  { -s *schema* |  { -t *schema.table*  [ -i *col1*[, *col2*, ...] |  -x *col1*[, *col2*, ...] ] } |  { -f | --file} *config-file* }  [ -l | --list ]  [ --gen\_profile\_only ]  [ -p *parallel-level* ]  [ --full ]  [ --skip\_root\_stats ]  [ -v | --verbose ]  [ --debug ]  [ -a ]  analyzedb { --clean\_last | --clean\_all }  analyzedb --version  analyzedb { -? | -h | --help } |

Описание

Утилита analyzedb обновляет статистику табличных данных для указанных таблиц в RT.Warehouse инкрементально и одновременно.

Выполняя операции ANALYZE, analyzedb создаёт снапшот метаданных таблицы и сохраняет его на диске на хосте мастера. Операция ANALYZE выполняется, только если таблица была изменена. Если таблица или партиция не изменялись с момента последнего анализа, analyzedb автоматически пропускает таблицу или партицию, поскольку они уже содержат актуальную статистику.

1. Для добавления оптимизированных таблиц, analyzedb обновляет статистику инкрементально, если статистика устарела. Например, если данные таблицы были изменены после сбора статистики для таблицы. Если статистики для таблицы нет, статистика собирается.

Для heap-таблиц статистика всегда обновляется.

Укажите параметр --full для обновления статистики таблицы, оптимизированной для добавления, даже если статистика таблицы актуальна.

По умолчанию analyzedb создаёт максимум 5 одновременных сеансов для параллельного анализа таблиц. Для каждого сеанса analyzedb выдаёт команду ANALYZE базе данных и указывает разные имена таблиц. Параметр ‑p управляет максимальным количеством одновременных сеансов.

Партиционированные таблицы, оптимизированные для добавления

Для партиционированной таблицы, оптимизированной для добавления, analyzedb проверяет корневую партицию партиционированной таблицы и leaf-партиции. При необходимости утилита обновляет статистику для устаревших партиций и корневой партиции.

Статистика корневой партиции требуется GPORCA. По умолчанию утилита analyzedb собирает статистику в корневой партиции партиционированной таблицы, если статистика не существует. Если какая-либо из leaf-партиций имеет устаревшую статистику, analyzedb также обновляет статистику корневой партиции. Стоимость обновления статистики корневого уровня сопоставима с анализом одной leaf-партиции. Вы можете указать параметр ‑-skip\_root\_stats, чтобы отключить сбор статистики в корневой партиции партиционированной таблицы.

Примечания

Утилита analyzedb обновляет статистику оптимизированной таблицы, если таблица была изменена командами DML или DDL, включая INSERT, DELETE, UPDATE, CREATE TABLE, ALTER TABLE и TRUNCATE. Утилита определяет, была ли таблица изменена, сравнивая метаданные каталога таблиц с предыдущим снапшотом метаданных, сделанным во время предыдущей операции analyzedb. Снапшоты метаданных таблицы хранятся в виде файлов состояния в каталоге db\_analyze/<*db\_name*>/<*timestamp*> в каталоге основных данных RT.Warehouse.

Утилита не удаляет автоматически старую информацию о снапшотах. Со временем снапшоты могут занимать большой объём дискового пространства. Чтобы освободить место на диске, вы можете сохранить самую свежую информацию о снапшотах и удалить старые снапшоты. Вы также можете указать параметр --clean\_last или --clean\_all, чтобы удалить файлы состояния, созданные с помощью analyzedb.

Если вы не укажете таблицу, набор таблиц или схему, утилита analyzedb собирает статистику по мере необходимости по всем таблицам системного каталога и пользовательским таблицам в базе данных.

На внешние таблицы analyzedb не влияет.

Имена таблиц, содержащие пробелы, не поддерживаются.

Выполнение команды ANALYZE для таблицы без использования утилиты analyzedb не обновляет метаданные таблицы, которые использует утилита analyzedb для определения актуальности статистики таблицы.

Параметры

Таблица — Параметры analyzedb

| Параметр | Описание |
| --- | --- |
| --clean\_last | Удаление файлов состояний, созданных последней операцией analyzedb. Все остальные параметры, кроме -d, игнорируются. |
| --clean\_all | Удаление всех файлов состояний, сгенерированных analyzedb. Все остальные параметры, кроме -d, игнорируются. |
| -d *dbname* | Задаёт имя базы данных, содержащей таблицы для анализа. Если этот параметр не указан, имя базы данных считывается из переменной среды PGDATABASE. Если PGDATABASE не задана, используется имя пользователя, указанное для соединения. |
| --debug | Если параметр указан, устанавливается уровень ведения журнала для отладки. Во время выполнения команды информация об уровне отладки записывается в файл журнала и в командную строку. Информация включает команды, выполняемые утилитой, и продолжительность каждой операции ANALYZE. |
| -f *config-file* | --file *config-file* | Текстовый файл, содержащий список таблиц для анализа. Можно указать относительный путь к файлу из текущего каталога.  Файл содержит одну таблицу в строке. Имена таблиц должны быть дополнены именем схемы. Опционально список столбцов можно указать с помощью -i или -x. Никакие другие параметры в файле не допускаются. В командной строке должны быть указаны другие параметры, такие как --full.  Только один из параметров может использоваться для указания файлов для анализа: -f или --file, -t или -s. При выполнении операций ANALYZE на несколько таблиц, analyzedb создаёт параллельные сеансы для параллельного анализа таблиц. Параметр -p управляет максимальным количеством одновременных сеансов.  В следующем примере первая строка выполняет операцию ANALYZE для таблицы public.nation, вторая строка выполняет операцию ANALYZE только для столбцов l\_shipdate и l\_receiptdate в таблице public.lineitem.   |  | | --- | | public.nation  public.lineitem -i l\_shipdate, l\_receiptdate | |
| --full | Выполнение операции ANALYZE для всех указанных таблиц. Операция выполняется, даже если статистика актуальна. |
| --gen\_profile\_only | Обновление снапшота analyzedb таблицы статистической информации без выполнения каких-либо операций ANALYZE. Если другие параметры определяют таблицы или схему, утилита обновляет информацию о снапшотах только для указанных таблиц. Укажите этот параметр, если команда ANALYZE была запущена для таблиц базы данных и вы хотите обновить analyzedb снапшот для таблиц. |
| -i *col1*, *col2*, ... | Опционально. Должен быть указан с параметром -t. Для таблицы, указанной с параметром -t, собирает статистику только для указанных столбцов. Можно указать только -i или -x. Оба варианта указать нельзя. |
| -l | --list | Перечисляет таблицы, которые были бы проанализированы с указанными параметрами. Операции ANALYZE не выполняются. |
| -p *parallel-level* | Количество таблиц, которые анализируются параллельно. *parallel level* может быть целым числом от 1 до 10 включительно. Значение по умолчанию — 5. |
| --skip\_root\_stats | Для партиционированной таблицы пропустите обновление статистики корневых партиций, если требуется обновить только некоторые статистические данные leaf-партиций.  Если обновляется статистика для партиционированной таблицы и вы знаете, какая статистика конечных партиций требует обновления, вы можете указать эти разделы и этот параметр для повышения производительности. |
| -s *schema* | Укажите схему для анализа. Все таблицы в схеме будут проанализированы. В командной строке можно указать только одно имя схемы. Только один из параметров может использоваться для указания файлов для анализа: -f или --file, -t или -s. |
| -t *schema.table* | Собирает статистику только по schema.table. Имя таблицы должно быть дополнено именем схемы. В командной строке можно указать только одно имя таблицы. Вы можете указать параметр -f, чтобы указать несколько таблиц в файле, или параметр -s, чтобы указать все таблицы в схеме. Только один из этих параметров может использоваться для указания файлов для анализа: -f или --file, -t или –s. |
| -x *col1*, *col2*, ... | Опционально. Должен быть указан с параметром -t. Для таблицы, указанной с параметром -t, исключить сбор статистики для указанных столбцов. Статистика собирается только по столбцам, которых нет в списке. Можно указать только -i или -x. Оба варианта указать нельзя. |
| -a | Бесшумный режим. Не запрашивать подтверждение пользователя. |
| -h | -? | --help | Отображает интерактивную справку. |
| -v | --verbose | Если параметр указан, устанавливается уровень ведения журнала как подробный для записи дополнительной информации в файл журнала и в командную строку во время выполнения команды. Информация включает в себя список всех таблиц, которые нужно проанализировать (включая дочерние leaf-партиции партиционированных таблиц). Вывод также включает продолжительность каждой операции ANALYZE. |
| --version | Отображает версию этой утилиты. |

Примеры

Пример, который собирает статистику только по набору столбцов таблицы. В базе данных mytest соберается статистика по столбцам shipdate и receivetdate в таблице public.orders:

|  |
| --- |
| analyzedb -d mytest -t public.orders -i shipdate, receiptdate |

Пример, который собирает статистику по таблице и исключает набор столбцов. В базе данных mytest собирается статистика по таблице public.foo и не собирается статистика по столбцам bar и test2.

|  |
| --- |
| analyzedb -d mytest -t public.foo -x bar, test2 |

Пример, указывающий файл, содержащий список таблиц. Эта команда собирает статистику по таблицам, перечисленным в файле analysis-tables в базе данных mytest.

|  |
| --- |
| analyzedb -d mytest -f analyze-tables |

Если вы не укажете таблицу, набор таблиц или схему, утилита analyzedb собирает статистику по мере необходимости для всех таблиц каталога и пользовательских таблиц в указанной базе данных. Эта команда обновляет статистику таблиц в таблицах системного каталога и пользовательских таблицах в базе данных mytest.

|  |
| --- |
| analyzedb -d mytest |

Вы можете создать функцию PL/Python для запуска утилиты analyzedb как функции RT.Warehouse. В этом примере команда CREATE FUNCTION создаёт определяемую пользователем функцию PL/Python, которая запускает утилиту analyzedb и отображает вывод в командной строке. В качестве параметра функции укажите параметры analyzedb.

|  |
| --- |
| CREATE OR REPLACE FUNCTION analyzedb(params TEXT)  RETURNS VOID AS  $BODY$  import subprocess  cmd = ['analyzedb', '-a' ] + params.split()  p = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)  # verbose output of process  for line in iter(p.stdout.readline, ''):  plpy.info(line);  p.wait()  $BODY$  LANGUAGE plpythonu VOLATILE; |

Когда эта команда SELECT запускается пользователем gpadmin, служебная программа analyzedb выполняет операцию анализа таблицы public.mytable, которая находится в базе данных mytest.

|  |
| --- |
| SELECT analyzedb('-d mytest -t public.mytable') ; |

|  |  |
| --- | --- |
| **Примечание.**  Для создания функции PL/Python процедурный язык PL/Python должен быть зарегистрирован как язык в базе данных. Например, эта команда CREATE LANGUAGE, запускаемая как gpadmin, регистрирует PL/Python как ненадежный язык:   |  | | --- | | CREATE LANGUAGE plpythonu; | |

clusterdb

Выполняет повторную кластеризацию таблиц, которые ранее были кластеризованы с помощью CLUSTER.

Синтаксис

|  |
| --- |
| clusterdb [*connection-option* ...] [--verbose | -v] [--table | -t *table*] [[--dbname | -d] *dbname*]  clusterdb [*connection-option* ...] [--all | -a] [--verbose | -v]  clusterdb -? | --help  clusterdb -V | --version |

Описание

Кластеризация таблицы означает физическое переупорядочение таблицы на диске в соответствии с индексом, чтобы операции сканирования индекса могли обращаться к данным на диске в некотором последовательном порядке, тем самым повышая производительность поиска по индексу для запросов, использующих этот индекс.

Утилита clusterdb найдёт любые таблицы в базе данных, которые ранее были кластеризованы с помощью SQL-команды CLUSTER, и снова кластеризует их по тому же индексу, который использовался последним. Таблицы, которые никогда не были кластеризованы, не затрагиваются.

clusterdb — это оболочка для SQL-команды CLUSTER. Хотя кластеризация таблицы таким образом поддерживается в RT.Warehouse, это не рекомендуется, поскольку сама операция CLUSTER выполняется очень медленно.

Если вам действительно нужно упорядочить таблицу таким образом, чтобы повысить производительность запроса, используйте оператор CREATE TABLE AS для изменения порядка таблицы на диске вместо использования CLUSTER. Если вы выполните “кластеризацию” таблицы таким образом, clusterdb не будет иметь значения.

Параметры

Таблица — Параметры clusterdb

| Параметр | Описание |
| --- | --- |
| -a | --all | Кластеризация всей базы данных. |
| [-d] *dbname* | [--dbname=]*dbname* | Задаёт имя базы данных для кластеризации. Если параметр не указан, имя базы данных считывается из переменной среды PGDATABASE. Если она не установлена, используется имя пользователя, указанное для соединения. |
| -e | --echo | Просмотр команд, которые clusterdb генерирует и отправляет на сервер. |
| -q | --quiet | Не отображать ответ. |
| -t *table* | --table=*table* | Кластеризирует только указанную таблицу. Несколько таблиц можно кластеризовать, написав несколько ключей -t. |
| -v | --verbose | Распечатывает подробную информацию в процессе обработки. |
| -V | --version | Распечатывает версию clusterdb и выходит. |
| -? | --help | Показывает справку об аргументах командной строки clusterdb и выходит. |

Таблица — Параметры подключения clusterdb

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p port | --port=port | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию используется 5432. |
| -U username | --username=username | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |
| --maintenance-db=dbname | Задаёт имя базы данных для подключения, чтобы определить, какие другие базы данных следует кластеризовать. Если параметр не указан, в качестве значения будет использоваться база данных postgres, а если она не существует, будет использоваться template1. |

Примеры

Для кластеризации базы данных test:

|  |
| --- |
| clusterdb test |

Для кластеризации одной таблицы foo в базе данных с именем xyzzy:

|  |
| --- |
| clusterdb --table foo xyzzyb |

createdb

Создаёт новую базу данных.

Синтаксис

|  |
| --- |
| createdb [*connection-option* ...] [*option* ...] [*dbname* ['*description*']]  createdb -? | --help  createdb -V | --version |

Описание

createdb создаёт новую базу данных в системе RT.Warehouse.

Обычно пользователь базы данных, выполняющий эту команду, становится владельцем новой базы данных. Однако можно указать другого владельца с помощью параметра -O, если у выполняющего пользователя есть соответствующие привилегии.

createdb — это оболочка для SQL-команды CREATE DATABASE.

Параметры

Таблица — Параметры createdb

| Параметр | Описание |
| --- | --- |
| dbname | Имя создаваемой базы данных. Имя должно быть уникальным среди всех других баз данных в системе RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGDATABASE, затем PGUSER или по умолчанию используется текущий системный пользователь. |
| description | Комментарий, который будет связан с вновь созданной базой данных. Описания, содержащие пробелы, должны быть заключены в кавычки. |
| -D *tablespace* | --tablespace=*tablespace* | Задаёт табличное пространство по умолчанию для базы данных. (Это имя обрабатывается как идентификатор в двойных кавычках.) |
| -e echo | Повторение команд, которые createdb генерирует и отправляет на сервер. |
| -E *encoding* | --encoding *encoding* | Кодировка набора символов для использования в новой базе данных. Укажите строковую константу (например, 'UTF8'), целочисленный номер кодировки или DEFAULT, чтобы использовать кодировку по умолчанию. |
| -l *locale* | --locale *locale* | Задаёт языковой стандарт, который будет использоваться в этой базе данных. Это эквивалентно указанию --lc-collate и --lc-ctype. |
| --lc-collate *locale* | Задаёт параметр LC\_COLLATE, который будет использоваться в этой базе данных. |
| --lc-ctype *locale* | Задаёт параметр LC\_CTYPE, который будет использоваться в этой базе данных. |
| --maintenance-db=*dbname* | Задаёт имя базы данных, к которой нужно подключиться при создании новой базы данных. Если параметр не указан, будет использоваться база данных postgres; если её не существует (или если это имя создаваемой новой базы данных), будет использоваться template1. |
| -O *owner* | --owner=*owner* | Имя пользователя базы данных, которому будет принадлежать новая база данных. По умолчанию это пользователь, который выполняет эту команду. (Это имя обрабатывается как идентификатор в двойных кавычках.) |
| -T template | --template=template | Имя шаблона, из которого будет создана новая база данных. По умолчанию template1. (Это имя обрабатывается как идентификатор в двойных кавычках.) |
| -V | --version | Распечатывает созданную версию и выходит. |
| -? | --help | Показывает справку об аргументах командной строки createdb и выходит. |

Параметры -D, -l, -E, -O и -T соответствуют параметрам базовой SQL-команды CREATE DATABASE.

Таблица — Параметры подключения createdb

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию используется 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |

Примеры

Чтобы создать базу данных test с использованием параметров по умолчанию:

|  |
| --- |
| createdb test |

Чтобы создать базу данных demo с помощью мастера RT.Warehouse на хосте gpmaster, порт 54321, с использованием схемы кодирования LATIN1:

|  |
| --- |
| createdb -p 54321 -h gpmaster -E LATIN1 demo |

createlang

Определяет новый процедурный язык для базы данных.

Синтаксис

|  |
| --- |
| createlang [*connection\_option* ...] [-e] *langname* [[-d] *dbname*]  createlang [*connection-option* ...] -l *dbname*  createlang -? | --help  createlang -V | --version |

Описание

Утилита createlang добавляет в базу данных новый процедурный язык. createlang — это оболочка для SQL-команды CREATE EXTENSION.

|  |
| --- |
| **Примечание.**  createlang устарел и может быть удалён в следующем выпуске. В RT.Warehouse 6.x использование createlang для добавления пакета процедурного языка приводит к ошибке. Вместо этого используйте команду CREATE EXTENSION. |

Пакеты процедурного языка, включенные в стандартный дистрибутив RT.Warehouse:

PL/pgSQL;

PL/Perl;

PL/Python.

Язык PL/pgSQL по умолчанию зарегистрирован во всех базах данных.

RT.Warehouse также имеет обработчики языков для PL/Java и PL/R, но эти языки не устанавливаются вместе с RT.Warehouse.

Параметры

Таблица — Параметры createlang

| Параметр | Описание |
| --- | --- |
| langname | Задаёт имя устанавливаемого процедурного языка. (Это имя в нижнем регистре.) |
| [-d] *dbname* | [--dbname=]*dbname* | Задаёт базу данных, в которую следует добавить язык. По умолчанию используется настройка переменной среды PGDATABASE или то же имя, что и у текущего пользователя системы. |
| -e | --echo | Повторяет команды, которые createlang генерирует и отправляет на сервер. |
| -l *dbname* | --list *dbname* | Показывает список уже установленных языков в целевой базе данных. |
| -V | --version | Распечатывает версию createlang и выходит. |
| -? | --help | Показывает справку об аргументах командной строки createlang и завершает работу. |

Таблица — Параметры подключения createlang

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста компьютера, на котором работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |

Примеры

Чтобы установить язык plperl в базу данных mytestdb:

|  |
| --- |
| createlang plperl mytestdb |

createuser

Создаёт новую роль базы данных.

Синтаксис

|  |
| --- |
| createuser [*connection-option* ...] [*role\_attribute* ...] [-e] *role\_name*  createuser -? | --help  createuser -V | --version |

Описание

createuser создаёт новую роль RT.Warehouse. Для создания новых ролей вы должны быть суперпользователем или иметь право CREATEROLE. Вы должны подключиться к базе данных как суперпользователь, чтобы создавать новых суперпользователей.

Суперпользователи могут обойти все проверки прав доступа в базе данных, поэтому привилегии суперпользователя не следует предоставлять легкомысленно.

createuser — это оболочка для SQL-команды CREATE ROLE.

Параметры

Таблица — Параметры createuser

| Параметр | Описание |
| --- | --- |
| role\_name | Имя создаваемой роли. Это имя должно отличаться от всех существующих ролей в данной установке RT.Warehouse. |
| -c number | --connection-limit=number | Устанавливает максимальное количество подключений для новой роли. По умолчанию ограничение не установлено. |
| -d | --createdb | Новой роли будет разрешено создавать базы данных. |
| -D | --no-createdb | Новой роли не будет разрешено создавать базы данных. Это значение по умолчанию. |
| -e | --echo | Повторяет команды, которые createuser генерирует и отправляет на сервер. |
| -E | --encrypted | Шифрует пароль роли, хранящийся в базе данных. Если не указано, используется пароль по умолчанию. |
| -i | --inherit | Новая роль автоматически унаследует привилегии ролей, членом которых она является. Это значение по умолчанию. |
| -I | --no-inherit | Новая роль не будет автоматически наследовать привилегии ролей, членом которых она является. |
| --interactive | Запрашивать имя пользователя, если оно не указано в командной строке, а также запрашивать, какой из параметров ‑d/-D, -r/-R, -s/-S не указан в командной строке. |
| -l | --login | Новой роли будет разрешён вход в RT.Warehouse. Это значение по умолчанию. |
| -L | --no-login | Новая роль не сможет войти в систему (роль на уровне группы). |
| -N | --unencrypted | Не шифрует пароль роли, хранящийся в базе данных. Если не указано, используется пароль по умолчанию. |
| -P | --pwprompt | Если параметр указан, createuser выдаст запрос на ввод пароля для новой роли. В этом нет необходимости, если вы не планируете использовать аутентификацию по паролю. |
| -r | --createrole | Новой роли будет разрешено создавать новые роли (привилегия CREATEROLE). |
| -R | --no-createrole | Новая роль не сможет создавать новые роли. Это значение по умолчанию. |
| -s | --superuser | Новая роль будет суперпользователем. |
| -S | --no-superuser | Новая роль не будет суперпользователем. Это значение по умолчанию. |
| -V | --version | Распечатывает версию createuser и выходит. |
| --replication | Новый пользователь будет иметь привилегию REPLICATION. |
| --no-replication | У нового пользователя не будет привилегии REPLICATION. |
| -? | --help | Показывает справку об аргументах командной строки createuser и выходит. |

Таблица — Параметры подключения createuser

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которй работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |

Примеры

Чтобы создать роль joe на сервере базы данных по умолчанию:

|  |
| --- |
| $ createuser joe |

Чтобы создать роль joe на сервере базы данных по умолчанию с запросом некоторых дополнительных атрибутов:

|  |
| --- |
| $ createuser --interactive joe  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 |

Чтобы создать ту же роль, joe, используя параметры подключения с явно указанными атрибутами и опираясь на базовую команду:

|  |
| --- |
| createuser -h masterhost -p 54321 -S -D -R -e joe  CREATE ROLE joe NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT  LOGIN;  CREATE ROLE |

Чтобы создать роль joe как суперпользователя и сразу же назначить пароль admin123:

|  |
| --- |
| createuser -P -s -e joe  Enter password for new role: admin123  Enter it again: admin123  CREATE ROLE joe PASSWORD 'admin123' SUPERUSER CREATEDB  CREATEROLE INHERIT LOGIN;  CREATE ROLE |

В приведённом выше примере новый пароль не отображается при вводе, но для ясности показано, что было введено. Однако пароль будет отображаться в отображаемой команде, как показано, если используется параметр -e.

dropdb

Удаляет базу данных.

Синтаксис

|  |
| --- |
| dropdb [*connection-option* ...] [-e] [-i] *dbname*  dropdb -? | --help  dropdb -V | --version |

Описание

dropdb уничтожает существующую базу данных. Пользователь, выполняющий эту команду, должен быть суперпользователем или владельцем удаляемой базы данных.

dropdb — это оболочка для SQL-команды DROP DATABASE.

Параметры

Таблица — Параметры dropdb

| Параметр | Описание |
| --- | --- |
| dbname | Имя удаляемой базы данных. |
| -e | --echo | Повторяет команды, которые dropdb генерирует и отправляет на сервер. |
| -i | --interactive | Перед выполнением каких-либо деструктивных действий выдаёт запрос на подтверждение. |
| -V | --version | Распечатывает версию dropdb и выходит. |
| --if-exists | Не выдавать ошибку, если база данных не существует. В этом случае выдаётся уведомление. |
| -? | --help | Показывает справку об аргументах командной строки dropdb и выходит. |

Таблица — Параметры подключения dropdb

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |
| --maintenance-db=dbname | Задаёт имя базы данных, к которой нужно подключиться, чтобы удалить целевую базу данных. Если параметр не указан, в качестве значения будет использоваться база данных postgres; если её не существует (или если это имя удаляемой базы данных), будет использоваться template1. |

Примеры

Чтобы удалить базу данных с именем demo, используя параметры соединения по умолчанию:

|  |
| --- |
| dropdb demo |

Чтобы удалить базу данных с именем demo, используя параметры подключения, с проверкой и просмотром базовой команды:

|  |
| --- |
| dropdb -p 54321 -h masterhost -i -e demo  Database "demo" will be permanently deleted.  Are you sure? (y/n) y  DROP DATABASE "demo"  DROP DATABASE |

droplang

Удаляет процедурный язык.

Синтаксис

|  |
| --- |
| droplang [*connection-option* ...] [-e] *langname* [[-d] *dbname*]  droplang [*connection-option* ...] [-e] -l *dbname*  droplang -? | --help  droplang -V | --version |

Описание

droplang удаляет существующий процедурный язык из базы данных.

droplang — это оболочка для SQL-команды DROP EXTENSION.

|  |
| --- |
| **Примечание.**  droplang устарел и может быть удалён в следующем выпуске. В RT.Warehouse 6.x при использовании droplang для удаления пакета процедурного языка возникает ошибка. Вместо этого используйте команду DROP EXTENSION. |

Параметры

Таблица — Параметры droplang

| Параметр | Описание |
| --- | --- |
| langname | Задаёт имя процедурного языка, который нужно удалить. (Это имя в нижнем регистре.) |
| [-d] *dbname* | [--dbname=]*dbname* | Определяет, из какой базы данных следует удалить язык. По умолчанию используется настройка переменной среды PGDATABASE или то же имя, что и у текущего пользователя системы. |
| -e | --echo | Повторяет команды, которые droplang генерирует и отправляет на сервер. |
| -l | --list | Показывает список уже установленных языков в целевой базе данных. |
| -V | --version | Распечатывает версию dropplang и выходит. |
| -? | --help | Показывает справку об аргументах командной строки dropplang и выходит. |

Таблица — Параметры подключения droplang

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими способами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |

Примеры

Чтобы удалить язык pltcl из базы данных mydatabase:

|  |
| --- |
| droplang pltcl mydatabase |

dropuser

Удаляет роль базы данных.

Синтаксис

|  |
| --- |
| dropuser [*connection-option* ...] [-e] [-i] *role\_name*  dropuser -? | --help  dropuser -V | --version |

Описание

dropuser удаляет существующую роль из RT.Warehouse. Только суперпользователи и пользователи с привилегией CREATEROLE могут удалять роли. Чтобы удалить роль суперпользователя, вы сами должны быть суперпользователем.

dropuser — это оболочка для SQL-команды DROP ROLE.

Параметры

Таблица — Параметры dropuser

| Параметр | Описание |
| --- | --- |
| role\_name | Название удаляемой роли. Вам будет предложено ввести имя, если оно не указано в командной строке и используется параметр -i/--interactive. |
| -e | --echo | Повторение команд, которые dropuser генерирует и отправляет на сервер. |
| -i | --interactive | Запрашивает подтверждение перед фактическим удалением роли и запрашивает имя роли, если оно не указано в командной строке. |
| --if-exists | Не выдаёт ошибку, если пользователь не существует. В этом случае выдаётся уведомление. |
| -V | --version | Распечатывает версию dropuser и выходит. |
| -? | --help | Показывает справку об аргументах командной строки dropuser и выходит. |

Таблица — Параметры подключения dropuser

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими способами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптахх, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |

Примеры

Чтобы удалить роль joe с использованием параметров подключения по умолчанию:

|  |
| --- |
| dropuser joe  DROP ROLE |

Чтобы удалить роль joe с помощью параметров подключения, с проверкой и просмотром базовой команды:

|  |
| --- |
| dropuser -p 54321 -h masterhost -i -e joe  Role "joe" will be permanently removed.  Are you sure? (y/n) y  DROP ROLE "joe"  DROP ROLE |

gpactivatestandby

Активирует хост резервного мастера и делает его активным мастером для системы RT.Warehouse.

Синтаксис

|  |
| --- |
| gpactivatestandby [-d *standby\_master\_datadir*] [-f] [-a] [-q]  [-l *logfile\_directory*]  gpactivatestandby -v  gpactivatestandby -? | -h | --help |

Описание

Утилита gpactivatestandby активирует хост резервного мастера и запускает его в качестве активного инстанса мастера для системы RT.Warehouse. Активированный резервный мастер фактически становится мастером RT.Warehouse, принимая клиентские соединения на главном порту.

Когда вы инициализируете резервный мастер, по умолчанию будет использоваться тот же порт, что и у активного мастера. Для получения информации о главном порте для резервного мастер-сервера см. gpinitstandby.

Вы должны запускать эту утилиту с хоста мастера, который вы активируете, а не с отказавшего хоста мастера, который вы отключаете. При запуске этой утилиты предполагается, что для системы настроен резервный мастер-хост (см. gpinitstandby).

Утилита выполнит следующие действия:

Останавливает процесс синхронизации (walreceiver) на резервном мастере;

Обновляет таблицы системного каталога резервного мастера с помощью журналов;

Активирует резервный мастер в качестве нового активного мастера для системы;

Перезапускает систему RT.Warehouse с новым мастер-хостом.

Резервный мастер-хост RT.Warehouse служит «горячим резервом» в случае выхода из строя основного мастер-хоста RT.Warehouse. Резервный мастер-сервер поддерживается в актуальном состоянии с помощью процессов репликации журнала транзакций (walsender и walreceiver), которые выполняются на хостах основного мастера и резервного мастера и ​​обеспечивают синхронизацию данных между хостами основного и резервного мастера.

Если основной мастер выходит из строя, процесс репликации журнала завершается, а резервный мастер может быть активирован вместо него с помощью утилиты gpactivatestandby. После активации резервного мастера реплицированные журналы используются для восстановления состояния мастер-хоста RT.Warehouse на момент последней успешно совершенной транзакции.

Чтобы использовать gpactivatestandby для активации нового основного мастер-хоста, мастер-хост, который ранее служил в качестве основного мастер-хоста, не может быть запущен.

Утилита проверяет наличие файла postmaster.pid в каталоге данных отключённого мастер-хоста и, если она обнаруживает его, предполагает, что старый мастер-хост все ещё активен. В некоторых случаях вам может потребоваться удалить файл postmaster.pid из каталога данных отключённого мастер-хоста перед запуском gpactivatestandby (например, если процесс отключённого мастер-хоста был неожиданно завершён).

После активации резервного мастера запустите ANALYZE, чтобы обновить статистику запросов к базе данных. Например:

|  |
| --- |
| psql dbname -c 'ANALYZE;' |

После того, как вы активируете резервный мастер в качестве основного, в системе RT.Warehouse больше не будет настроен резервный мастер. Вы можете указать другой хост в качестве нового резервного с помощью утилиты gpinitstandby.

Параметры

Таблица — Параметры gpactivatestandby

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -d *standby\_master\_datadir* | Абсолютный путь к каталогу данных для мастер-хоста, который вы активируете. Если этот параметр не указан, gpactivatestandby использует значение параметра среды MASTER\_DATA\_DIRECTORY, установленное на мастер-хосте, который вы активируете. Если этот параметр указан, она переопределяет любые настройки MASTER\_DATA\_DIRECTORY. Если каталог не может быть определён, утилита возвращает ошибку. |
| -f | Используйте этот параметр для принудительной активации резервного мастер-хоста. Используйте эту опцию, только если вы уверены, что хосты резервного мастера и основного мастера имеют связность. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -q | Работа в тихом режиме. Вывод команды не отображается на экране, но все равно записывается в файл журнала. |
| -v | Отображает версию, статус, дату последнего обновления и контрольную сумму этой утилиты. |
| -? | -h | --help | Отображает интерактивную справку. |

Примеры

Активируйте резервный мастер-хост и сделайте его активным инстансом мастера для системы RT.Warehouse (запускается с резервного мастер-хоста, который вы активируете):

|  |
| --- |
| gpactivatestandby -d /gpdata |

gpaddmirrors

Добавляет зеркальные сегменты в систему RT.Warehouse, которая изначально была настроена без зеркалирования.

Синтаксис

|  |
| --- |
| gpaddmirrors [-p *port\_offset*] [-m *datadir\_config\_file* [-a]] [-s]  [-d *master\_data\_directory*] [-B *parallel\_processes*] [-l *logfile\_directory*]  [-v]  gpaddmirrors -i *mirror\_config\_file* [-a] [-d *master\_data\_directory*]  [-B *parallel\_processes*] [-l *logfile\_directory*] [-v]  gpaddmirrors -o *output\_sample\_mirror\_config* [-s] [-m *datadir\_config\_file*]  gpaddmirrors -?  gpaddmirrors --version |

Описание

Утилита gpaddmirrors настраивает инстансы зеркального сегмента для существующей системы RT.Warehouse, которая изначально была настроена только для инстансов первичного сегмента. Утилита создаст зеркальные инстансы и начнёт процесс онлайн-репликации между инстансами основного и зеркального сегментов. После того, как все зеркала будут синхронизированы с их основными, ваша система RT.Warehouse станет полностью избыточной.

|  |
| --- |
| **Внимание.**  Во время онлайн-репликации RT.Warehouse должна находиться в состоянии покоя, рабочие нагрузки и другие запросы не должны выполняться. |

По умолчанию утилита запрашивает у вас расположение файловой системы, в которых она будет создавать каталоги данных зеркальных сегментов. Если вы не хотите, чтобы вас спрашивали, вы можете передать файл, содержащий расположение файловой системы, используя параметр ‑m.

Расположение и порты зеркала должны отличаться от местоположений и портов данных основного сегмента.

Утилита создаёт уникальный каталог данных для каждого инстанса зеркального сегмента в указанном месте, используя предопределённое соглашение об именах. Для инстансов зеркального сегмента должно быть объявлено такое же количество расположений файловой системы, как и для инстансов основного сегмента. Можно указать одно и то же имя каталога несколько раз, если вы хотите, чтобы ваши зеркальные каталоги данных создавались в одном и том же месте, или вы можете ввести другое местоположение данных для каждого зеркала. Введите абсолютный путь. Например:

|  |
| --- |
| Enter mirror segment data directory location 1 of 2 > /gpdb/mirror  Enter mirror segment data directory location 2 of 2 > /gpdb/mirror |

Или:

|  |
| --- |
| Enter mirror segment data directory location 1 of 2 > /gpdb/m1  Enter mirror segment data directory location 2 of 2 > /gpdb/m2 |

Кроме того, вы можете запустить утилиту gpaddmirrors и предоставить подробный файл конфигурации, используя параметр -i. Это полезно, если вы хотите, чтобы зеркальные сегменты находились на совершенно ином наборе хостов, чем ваши основные сегменты. Формат файла конфигурации зеркала:

|  |
| --- |
| <row\_id>=<contentID>|<address>|<port>|<data\_dir> |

Где row\_id — это строка в файле, contentID — это идентификатор содержимого инстанса сегмента, address — это имя хоста или IP-адрес хоста сегмента, port — это порт связи, а data\_dir — это каталог данных инстанса сегмента.

Например:

|  |
| --- |
| 0=0|sdw1-1|60000|/gpdata/mir1/gp0  1=1|sdw1-1|60001|/gpdata/mir2/gp1 |

Таблица системного каталога gp\_segment\_configuration может помочь вам определить текущую конфигурацию основного сегмента, чтобы вы могли спланировать конфигурацию зеркального сегмента. Например, выполните следующий запрос:

|  |
| --- |
| =# SELECT dbid, content, address as host\_address, port, datadir  FROM gp\_segment\_configuration  ORDER BY dbid; |

При создании зеркал на альтернативных зеркальных хостах новые хосты зеркального сегмента должны быть предварительно установлены с программным обеспечением RT.Warehouse и настроены точно так же, как существующие хосты первичного сегмента.

Вы должны убедиться, что пользователь, запускающий gpaddmirrors (пользователь gpadmin), имеет права на запись в указанные каталоги данных. Вы можете создать эти каталоги на хостах сегментов и передать их соответствующему пользователю перед запуском gpaddmirrors.

|  |
| --- |
| **Примечание.**  Эта утилита использует SSH-соединения между системами для выполнения своих задач. В больших развёртываниях RT.Warehouse, облачных развёртываниях или развёртываниях с большим количеством сегментов на хост эта утилита может превышать максимальный порог хоста для неаутентифицированных соединений. Рассмотрите возможность обновления параметра конфигурации SSH MaxStartups, чтобы увеличить этот порог. Дополнительные сведения о параметрах конфигурации SSH см. в документации по SSH для вашего дистрибутива Linux. |

Параметры

Таблица — Параметры gpaddmirrors

| Параметр | Описание |
| --- | --- |
| -a | Запуск в тихом режиме, без запроса информации. Если используется этот параметр, необходимо предоставить файл конфигурации с параметром -m или -i. |
| -B *parallel\_processes* | Количество процессов настройки зеркала, запускаемых параллельно. Если не указано иное, утилита запустит до 10 параллельных процессов в зависимости от того, сколько инстансов зеркальных сегментов ей нужно настроить. |
| -d *master\_data\_directory* | Каталог основных данных. Если параметр не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -i *mirror\_config\_file* | Файл конфигурации, содержащий по одной строке для каждого зеркального сегмента, который вы хотите создать. Для каждого основного сегмента в системе должен быть указан один зеркальный сегмент. Формат этого файла следующий (согласно атрибутам в таблице каталога gp\_segment\_configuration):   |  | | --- | | <row\_id>=<contentID>|<address>|<port>|<data\_dir> |   Где row\_id — это строка в файле, contentID — это идентификатор содержимого инстанса сегмента, address — это имя хоста или IP-адрес хоста сегмента, port — это порт связи, а data\_dir — это каталог данных инстанса сегмента. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -m *datadir\_config\_file* | Файл конфигурации, содержащий список расположений файловой системы, в которых будут созданы каталоги зеркальных данных. Если параметр не указан, утилита запросит расположение. Каждая строка в файле указывает расположение каталога данных зеркала. Например:   |  | | --- | | /gpdata/m1  /gpdata/m2  /gpdata/m3  /gpdata/m4 | |
| -o *output\_sample\_mirror\_config* | Если вы не знаете, как разместить файл конфигурации зеркала, используемый параметром -i, вы можете запустить gpaddmirrors с данным параметром, чтобы сгенерировать образец файла конфигурации зеркала на основе конфигурации вашего основного сегмента. Утилита предложит вам указать расположение каталога данных зеркального сегмента (если вы не укажете его в файле с помощью -m). Затем вы можете отредактировать этот файл, чтобы при необходимости изменить имена хостов на альтернативные зеркальные хосты. |
| -p *port\_offset* | Опционально. Это число используется для расчёта портов базы данных, используемых для зеркальных сегментов. Смещение по умолчанию — 1000. Назначения портов зеркала рассчитываются следующим образом: основной порт + смещение = порт зеркальной базы данных. Например, если основной сегмент имеет порт 50001, то его зеркало по умолчанию будет использовать порт базы данных 51001. |
| -s | Распределяет зеркальные сегменты по доступным хостам. По умолчанию набор зеркальных сегментов группируется вместе на альтернативном хосте из их основного набора сегментов. При распределении зеркал каждое зеркало размещается на разных хостах в массиве RT.Warehouse. Распространение разрешено только при наличии достаточного количества хостов в массиве (количество хостов больше, чем количество инстансов сегмента на хост). |
| -v | Устанавливает подробный вывод журнала. |
| --version | Отображает версию этой утилиты. |
| -? | Отображает интерактивную справку. |

Примеры

Добавьте зеркалирование в существующую систему RT.Warehouse, используя тот же набор хостов, что и ваши основные данные. Вычислите порты зеркальной базы данных, добавив 100 к текущим номерам портов первичного сегмента:

|  |
| --- |
| $ gpaddmirrors -p 100 |

Добавьте зеркалирование в существующую систему RT.Warehouse, используя набор хостов, отличный от ваших первичных данных:

|  |
| --- |
| $ gpaddmirrors -i mirror\_config\_file |

Где mirror\_config\_file выглядит примерно так:

|  |
| --- |
| 0=0|sdw1-1|52001|/gpdata/mir1/gp0  1=1|sdw1-2|52002|/gpdata/mir2/gp1  2=2|sdw2-1|52001|/gpdata/mir1/gp2  3=3|sdw2-2|52002|/gpdata/mir2/gp3 |

Создайте образец файла конфигурации зеркала с параметром -o для использования с gpaddmirrors -i:

|  |
| --- |
| $ gpaddmirrors -o /home/gpadmin/sample\_mirror\_config |

gpbackup

Создаёт резервную копию RT.Warehouse для использования с утилитой gprestore.

Синтаксис

|  |
| --- |
| gpbackup --dbname *database\_name*  [--backup-dir *directory*]  [--compression-level *level*]  [--data-only]  [--debug]  [--exclude-schema *schema\_name*]  [--exclude-table *schema.table*]  [--exclude-table-file *file\_name*]  [--include-schema *schema\_name*]  [--include-table *schema.table*]  [--include-table-file *file\_name*]  [--incremental [--from-timestamp *backup-timestamp*]]  [--jobs *int*]  [--leaf-partition-data]  [--metadata-only]  [--no-compression]  [--plugin-config *config\_file\_location*]  [--quiet]  [--single-data-file]  [--verbose]  [--version]  [--with-stats]  gpbackup --help |

Описание

Утилита gpbackup выполняет резервное копирование содержимого базы данных в коллекцию файлов метаданных и файлов данных, которые можно использовать позже для восстановления базы данных с помощью gprestore. При резервном копировании базы данных вы можете указать параметры фильтрации на уровне таблицы и на уровне схемы для резервного копирования определённых таблиц. Например, вы можете комбинировать параметры уровня схемы и уровня таблицы для резервного копирования всех таблиц в схеме, кроме одной.

По умолчанию gpbackup выполняет резервное копирование объектов в указанной базе данных, а также глобальных системных объектов RT.Warehouse. Вы можете дополнительно указать параметр --with-globals с gprestore для восстановления глобальных объектов.

gpbackup по умолчанию сохраняет файлы метаданных объекта и файлы DDL для резервного копирования в каталоге основных данных RT.Warehouse. Сегменты RT.Warehouse используют команду COPY ... ON SEGMENT для сохранения данных для резервных копий таблиц в сжатых файлах данных CSV, расположенных в каталоге данных каждого сегмента.

Вы можете добавить параметр --backup-dir, чтобы скопировать все файлы резервных копий с главной RT.Warehouse и сегментировать хосты по абсолютному пути для дальнейшего использования. Предусмотрены дополнительные параметры для фильтрации набора резервных копий, чтобы включить или исключить определённые таблицы.

Вы можете создать инкрементную резервную копию с параметром ‑‑incremental. Инкрементные резервные копии эффективны, когда общий объём данных в таблицах, оптимизированных для добавления, или в партициях таблиц, которые изменились, невелик по сравнению с данными, которые не изменились.

С параметром по умолчанию --jobs (1 задание) каждая операция gpbackup использует одну транзакцию на мастер-хосте RT.Warehouse. Команда COPY ... ON SEGMENT выполняет задачу резервного копирования параллельно на каждом хосте сегмента. Процесс резервного копирования устанавливает блокировку ACCESS SHARE для каждой резервной копии таблицы. Во время процесса блокировки таблицы база данных должна находиться в состоянии покоя.

Когда операция резервного копирования завершается, gpbackup возвращает код состояния. См. Коды возврата.

Утилиту gpbackup нельзя запустить, пока gpexpand инициализирует новые сегменты. Резервные копии, созданные до расширения, невозможно восстановить с помощью gprestore после завершения расширения кластера.

gpbackup может отправлять уведомления о статусе по электронной почте после завершения операции резервного копирования. Вы указываете момент, когда утилита отправляет уведомление, и получателей электронной почты в файле конфигурации.

|  |
| --- |
| **Примечание.**  Эта утилита использует SSH-соединения между системами для выполнения своих задач. В больших развёртываниях RT.Warehouse, облачных развёртываниях или развёртываниях с большим количеством сегментов на хост эта утилита может превышать максимальный порог хоста для неаутентифицированных соединений. Рассмотрите возможность обновления параметра конфигурации SSH MaxStartups, чтобы увеличить этот порог. Дополнительные сведения о параметрах конфигурации SSH см. в документации по SSH для вашего дистрибутива Linux. |

Параметры

Таблица — Параметры gpbackup

| Параметр | Описание |
| --- | --- |
| --dbname *database\_name* | Обязательный параметр. Задаёт базу данных для резервного копирования. |
| --backup-dir *directory* | Опционально. Копирует все необходимые файлы резервных копий (файлы метаданных и файлы данных) в указанный каталог. Вы должны указать каталог как абсолютный путь (не относительный). Если вы не укажете этот параметр, файлы метаданных будут созданы на мастер-хосте RT.Warehouse в каталоге $MASTER\_DATA\_DIRECTORY/backups/YYYYMMDD/YYYYMMDDhhmmss/. Хосты сегментов создают файлы данных CSV в каталоге <seg\_dir>/backups/YYYYMMDD/YYYYMMDDhhmmss/. Когда вы указываете настраиваемый каталог резервного копирования, файлы копируются по этим путям в подкаталогах каталога резервного копирования. Вы не можете комбинировать этот параметр с параметром --plugin-config. |
| --compression-level *level* | Опционально. Определяет уровень сжатия gzip (от 1 до 9), используемый для сжатия файлов данных. По умолчанию — 1. Обратите внимание, что gpbackup по умолчанию использует сжатие. |
| --data-only | Опционально. Выполняет резервное копирование только табличных данных в файлы CSV, но не выполняет резервное копирование файлов метаданных, необходимых для воссоздания таблиц и других объектов базы данных. |
| --debug | Опционально. Отображает подробные отладочные сообщения во время работы. |
| --exclude-schema *schema\_name* | Опционально. Задаёт схему базы данных, которую нужно исключить из резервной копии. Вы можете указать этот параметр несколько раз, чтобы исключить несколько схем. Вы не можете комбинировать этот параметр с параметром ‑‑include-schema или параметром фильтрации таблиц, таким как --include-table. |
| --exclude-table *schema.table* | Опционально. Задаёт таблицу, которую нужно исключить из резервной копии. Таблица должна быть в формате <schema-name>.<table-name>. Если в имени таблицы или схемы используется любой символ, кроме строчной буквы, числа или символа подчеркивания, то вы должны заключить это имя в двойные кавычки. Вы можете указать этот параетр несколько раз. Вы не можете комбинировать этот параметр с параметром --exclude-schema или другим параметром фильтрации таблиц, таким как --include-table. Вы не можете использовать этот параметр в сочетании с --leaf-partition-data. Хотя вы можете указать имена leaf-партиций, gpbackup игнорирует имена партиций. |
| --exclude-table-file *file\_name* | Опционально. Задаёт текстовый файл, содержащий список таблиц, которые нужно исключить из резервной копии. Каждая строка в текстовом файле должна определять одну таблицу в формате <schema-name>.<table-name>. Файл не должен содержать завершающих строк. Если в имени таблицы или схемы используется любой символ, кроме строчной буквы, числа или символа подчеркивания, то вы должны заключить это имя в двойные кавычки. Вы не можете комбинировать этот параметр с параметром --exclude-schema или другим параметром фильтрации таблиц, таким как --include-table. Вы не можете использовать этот параметр в сочетании с --leaf-partition-data. Хотя вы можете указать имена leaf-партиций в файле, указанном с помощью --exclude-table-file, gpbackup игнорирует имена партиций. |
| --include-schema *schema\_name* | Опционально. Задаёт схему базы данных для включения в резервную копию. Вы можете указать этот параметр несколько раз, чтобы включить несколько схем. Если вы укажете этот параметр, любые схемы, не включённые в последующие параметры --include-schema, будут исключены из резервного набора. Вы не можете комбинировать этот параметр с параметрами --exclude-schema, --include-table или --include-table-file. |
| --include-table *schema.table* | Опционально. Задаёт таблицу для включения в резервную копию. Таблица должна быть в формате <schema-name>.<table-name>. Если в имени таблицы или схемы используется какой-либо символ, кроме строчной буквы, числа или символа подчеркивания, то вы должны заключить это имя в одинарные кавычки. Для получения информации о символах, которые поддерживаются в именах схем и таблиц, см. Схема и имена таблиц. Этот параметр можно указывать несколько раз. Вы не можете комбинировать этот параметр с параметром фильтрации схемы, например --include-schema, или другим параметром фильтрации таблиц, например --exclude-table-file. Вы также можете указать полное имя последовательности или представления. Если вы укажете этот параметр, утилита не будет автоматически создавать резервные копии зависимых объектов. Вы также должны явно указать требуемые зависимые объекты. Например, если вы создаёте резервную копию представления, вы должны также создать резервную копию таблиц, которые использует представление. Если вы создаёте резервную копию таблицы, которая использует последовательность, вы также должны создать резервную копию последовательности. Вы можете дополнительно указать имя leaf-партиции таблицы вместо имени таблицы, чтобы включить в резервную копию только определённые leaf-партиции с параметром --leaf-partition-data. При резервном копировании leaf-партиции выполняется резервное копирование данных leaf-партиции вместе с метаданными для партиционированной таблицы. |
| --include-table-file *file\_name* | Опционально. Задаёт текстовый файл, содержащий список таблиц для включения в резервную копию. Каждая строка в текстовом файле должна определять одну таблицу в формате <schema-name>.<table-name>. Файл не должен содержать завершающих строк. См. в Схема и имена таблиц информацию о символах, которые поддерживаются в именах схем и таблиц. Любые таблицы, не указанные в этом файле, не включаются в резервный набор. Вы не можете комбинировать этот параметр с параметром фильтрации схемы, например --include-schema, или другим параметром фильтрации таблиц, например --exclude-table-file. Вы также можете указать полное имя последовательности или представления. Если вы укажете этот параметр, утилита не будет автоматически создавать резервные копии зависимых объектов. Вы также должны явно указать требуемые зависимые объекты. Например, если вы создаёте резервную копию представления, вы также должны указать таблицы, которые оно использует. Если вы указываете таблицу, в которой используется последовательность, вы также должны указать последовательность. Вы можете дополнительно указать имя leaf-партиции таблицы вместо имени таблицы, чтобы включить в резервную копию только leaf-партиции с параметром --leaf-partition-data. При резервном копировании leaf-партиции выполняется резервное копирование данных leaf-партиции вместе с метаданными для партиционированной таблицы. |
| --incremental | Укажите этот параметр, чтобы добавить инкрементную резервную копию в набор инкрементных резервных копий. Набор резервных копий — это полная резервная копия и одна или несколько инкрементных резервных копий. Резервные копии в наборе должны быть созданы с согласованным набором параметров резервного копирования, чтобы гарантировать, что набор резервных копий можно использовать в операции восстановления. По умолчанию gpbackup пытается найти самую последнюю существующую резервную копию с согласованным набором параметров. Если резервная копия является полной, утилита создаёт набор резервных копий. Если резервная копия является инкрементной, утилита добавляет резервную копию к существующему набору резервных копий. Инкрементная резервная копия добавляется как последняя резервная копия в набор резервных копий. Вы можете указать --from-timestamp, чтобы переопределить поведение по умолчанию. |
| --from-timestamp *backup-timestamp* | Опционально. Задаёт метку времени резервной копии. Указанная резервная копия должна иметь параметры резервного копирования, совместимые с создаваемой инкрементной резервной копией. Если указанная резервная копия является полной, утилита создает набор резервных копий. Если указанная резервная копия является инкрементальной, утилита добавляет инкрементную резервную копию к существующему набору резервных копий. В этом параметре необходимо указать --leaf-partition-data. Вы не можете комбинировать этот параметр с --data-only или --metadata-only. Резервная копия не создаётся, и утилита возвращает ошибку, если резервная копия не может добавить резервную копию к существующему набору инкрементных резервных копий или не может использовать резервную копию для создания. |
| --jobs *int* | Опционально. Задаёт количество заданий, которые будут выполняться параллельно при резервном копировании таблиц. По умолчанию gpbackup использует 1 задание (подключение к базе данных). Увеличение этого числа может повысить скорость резервного копирования данных. При выполнении нескольких заданий каждое задание выполняет резервное копирование таблиц в отдельной транзакции. Например, если вы укажете --jobs 2, утилита создаст два процесса, каждый процесс запускает одну транзакцию, а утилита выполняет резервное копирование таблиц параллельно, используя два процесса.  **Внимание**: если вы укажете значение больше 1, база данных должна находиться в состоянии покоя, пока утилита устанавливает блокировку таблиц, для которых выполняется резервное копирование. Если операции с базой данных выполняются с таблицами, резервные копии которых создаются в процессе блокировки таблицы, согласованность между таблицами, резервные копии которых создаются в различных транзакциях, не может быть гарантирована.  Вы не можете использовать этот параметр в сочетании с параметрами --metadata-only, --single-data-file или --plugin-config. |
| --leaf-partition-data | Опционально. Для партиционированных таблиц создаёт один файл данных на каждую leaf-партицию вместо одного файла данных для всей таблицы (по умолчанию). Использование этого параметра также позволяет указать отдельные leaf-партиции для включения в резервную копию с параметром --include-table-file. Вы не можете использовать этот параметр в сочетании с --exclude-table-file или --exclude-table. |
| --metadata-only | Опционально. Создаёт только файлы метаданных (DDL), необходимые для воссоздания объектов базы данных, но не выполняет резервное копирование фактических данных таблицы. |
| --no-compression | Опционально. Не сжимать файлы CSV с данными таблицы. |
| --plugin-config *config-file\_location* | Указывает расположение файла конфигурации подключаемого модуля gpbackup, текстового файла в формате YAML. Файл содержит информацию о конфигурации для приложения подключаемого модуля, которое gpbackup использует во время операции резервного копирования. Если вы укажете параметр --plugin-config при резервном копировании базы данных, вы должны указать этот параметр с информацией о конфигурации для соответствующего приложения подключаемого модуля при восстановлении базы данных из резервной копии. Этот параметр нельзя комбинировать с параметром --backup-dir. |
| --quiet | Опционально. Не отображать все сообщения журнала без предупреждений и ошибок. |
| --single-data-file | Опционально. Создаёт один файл данных на каждом хосте сегмента для всех таблиц, резервные копии которых созданы в этом сегменте. По умолчанию каждый gpbackup создаёт один сжатый файл CSV для каждой таблицы, резервная копия которой выполняется в сегменте.  **Примечание**. Если вы используете параметр --single-data-file для объединения резервных копий таблиц в один файл для каждого сегмента, вы не можете установить для параметра gprestore --jobs значение выше 1 для выполнения операции параллельного восстановления. |
| --verbose | Опционально. Печатает подробные сообщения журнала. |
| --version | Опционально. Распечатывает номер версии и выходит. |
| --with-stats | Опционально. Включает статистику плана запроса в резервный набор. |
| --help | Отображает интерактивную справку. |

Коды возврата

Один из этих кодов возвращается после завершения gpbackup:

0 — резервное копирование выполнено без проблем;

1 — резервное копирование завершено с некритическими ошибками. Смотрите файл журнала для получения дополнительной информации.

2 — резервное копирование завершилось фатальной ошибкой. Смотрите файл журнала для получения дополнительной информации.

Схема и имена таблиц

При указании параметра фильтрации таблиц --include-table или --include-table-file для вывода списка таблиц для резервного копирования, утилита gpbackup поддерживает резервное копирование схем или таблиц, когда имя содержит символы верхнего регистра или эти специальные символы.

~ # $ % ^ & \* ( ) \_ - + [ ] { } > < \ | ; : / ? ! ,

Если имя содержит верхний регистр или специальный символ и указано в командной строке с помощью --include-table, имя должно быть заключено в одинарные кавычки.

|  |
| --- |
| gpbackup --dbname test --include-table 'my#1schema'.'my\_$42\_Table' |

Когда таблица указана в файле для использования с --include-table-file, одинарные кавычки не требуются. Например, это содержимое текстового файла, который используется с --include-table-file для резервного копирования двух таблиц.

|  |
| --- |
| my#1schema.my\_$42\_Table  my#1schema.my\_$590\_Table |

|  |
| --- |
| **Примечание.**  Параметры --include-table и --include-table-file не поддерживают имена схем или таблиц, содержащие символы двойной кавычки ("), точку (.), новую строку (\n) или пробел ( ). |

Примеры

Сделать резервную копию всех схем и таблиц в базе данных demo, включая глобальную статистику системных объектов RT.Warehouse:

|  |
| --- |
| $ gpbackup --dbname demo |

Сделать резервную копию всех схем и таблиц в базе данных demo, кроме схемы twitter:

|  |
| --- |
| $ gpbackup --dbname demo --exclude-schema twitter |

Сделать резервную копию только схемы twitter в базе данных demo:

|  |
| --- |
| $ gpbackup --dbname demo --include-schema twitter |

Сделать резервную копию всех схем и таблиц в базе данных demo, включая глобальные системные объекты RT.Warehouse и статистику запросов, и скопировать все файлы резервных копий в каталог /home/gpadmin/backup:

|  |
| --- |
| $ gpbackup --dbname demo --with-stats --backup-dir /home/gpadmin/backup |

В этом примере --include-schema с --exclude-table используется для резервного копирования схемы, за исключением одной таблицы:

|  |
| --- |
| $ gpbackup --dbname demo --include-schema mydata --exclude-table mydata.addresses |

Вы не можете использовать параметр --exclude-schema с параметром фильтрации таблиц, например --include-table.

gpcheckcat

Утилита gpcheckcat проверяет таблицы каталога RT.Warehouse на несоответствия.

Утилита находится в $GPHOME/bin/lib.

Синтаксис

|  |
| --- |
| gpcheckcat [ *options*] [ *dbname*]  Options:  -g *dir*  -p *port*  -P *password*  -U *user\_name*  -S {none | only}  -O  -R *test\_name*  -C *catalog\_name*  -B *parallel\_processes*  -v  -A  gpcheckcat -l  gpcheckcat -? |

Описание

Утилита gpcheckcat запускает несколько тестов, которые проверяют несоответствие каталога базы данных. Некоторые тесты нельзя запускать одновременно с другими операторами рабочей нагрузки, иначе их нельзя будет использовать. При запуске gpcheckcat перезапустите базу данных в ограниченном режиме, иначе gpcheckcat может сообщать о несоответствиях из-за текущих операций с базой данных, а не о фактическом количестве несоответствий. Если вы запускаете gpcheckcat без остановки работы базы данных, запустите его с параметром -O.

|  |
| --- |
| **Примечание.**  Каждый раз, когда вы запускаете утилиту, она проверяет и удаляет потерянные временные схемы базы данных (временные схемы без идентификатора сеанса) в указанных базах данных. Утилита отображает результаты проверки временной схемы без родителя в командной строке, а также регистрирует результаты. |

Несоответствия каталога — это несоответствия, возникающие между системными таблицами RT.Warehouse. В общем, существует три типа несоответствий:

1. Несоответствия в системных таблицах на уровне сегмента. Например, несоответствие между системной таблицей, содержащей данные таблицы, и системной таблицей, содержащей данные столбцов. Другой вариант — системная таблица, содержащая дубликаты в столбце, который должен быть уникальным.
2. Несоответствия между одной и той же системной таблицей по сегментам. Например, в системной таблице отсутствует строка в одном сегменте, но в других сегментах эта строка есть. В качестве другого примера, значения конкретных данных столбца строки различаются для разных сегментов, например, права владельца таблицы или доступа к таблице.
3. Несоответствие между таблицей каталога и файловой системой. Например, файл существует в каталоге базы данных, но для него нет записи в таблице pg\_class.

Параметры

Таблица — Параметры gpcheckcat

| Параметр | Описание |
| --- | --- |
| -A | Запускает gpcheckcat для всех баз данных в установке RT.Warehouse. |
| -B *parallel\_processes* | Количество процессов, запускаемых параллельно. Утилита gpcheckcat пытается определить количество одновременно используемых процессов (размер пакета). Утилита предполагает, что она может использовать буфер размером не менее 20 МБ для каждого процесса. Максимальное количество параллельных процессов — это количество инстансов сегмента RT.Warehouse. Утилита отображает количество используемых параллельных процессов, когда начинает проверку каталога.  **Примечание**. Утилите может не хватить памяти, если количество возвращаемых ошибок превышает размер буфера. Если возникает ошибка нехватки памяти, вы можете уменьшить размер пакета с помощью параметра -B. Например, если утилита отображает размер пакета 936 и не хватает памяти, вы можете указать -B 468 для параллельного запуска 468 процессов. |
| -C *catalog\_table* | Запускает тесты кросс-последовательности, внешнего ключа и ACL для указанной таблицы каталога. |
| -g *data\_director* | Создаёт скрипт SQL для устранения несоответствий каталога. Скрипты помещаются в data\_directory. |
| -l | Перечисляет тесты gpcheckcat. |
| -O | Запускает только тесты gpcheckcat, которые можно запускать в онлайн-режиме (без ограничений). |
| -p *port* | Этот параметр указывает порт, который используется RT.Warehouse. |
| -P *password* | Пароль пользователя, подключающегося к RT.Warehouse. |
| -R *test\_name* | Указывает тест для запуска. Некоторые тесты можно запускать только в том случае, если RT.Warehouse находится в ограниченном режиме. Это те тесты, которые можно выполнять:   * acl — проверка кросс-согласованности для прав доступа; * duplicate — проверка наличия повторяющихся записей; * foreign\_key — проверка внешних ключей; * inconsistent — кросс-проверка непротиворечивости основного сегмента; * missing\_extraneous — кросс-проверка целостности отсутствующих или посторонних записей; * owner — проверка владения таблицей, несовместимое с основной базой данных; * orphaned\_toast\_tables — проверка наличия потерянных TOAST-таблиц;   **Примечание**. Существует несколько причин «осиротения» TOAST-таблицы, когда невозможно создать скрипта восстановления и требуется изменение каталога вручную. Одна из причин — если запись reltoastrelid в pg\_class указывает на неправильную таблицу TOAST (несоответствие TOAST-таблицы). Другая причина — reltoastrelid в pg\_class отсутствуют и запись pg\_depend отсутствует (дважды «сиротская» TOAST-таблица). Если необходимо изменить каталог вручную, gpcheckcat отобразит подробные шаги, которые вы можете выполнить, чтобы обновить каталог.   * part\_integrity — проверка целостности ветки pg\_partition, партиций с OID, политики распределения партиций; * part\_constraint — проверка ограничений для партиционированных таблиц; * unique\_index\_violation — проверка таблиц, в которых есть столбцы с ограничением уникального индекса, на наличие повторяющихся записей; * dependency — проверка зависимости от несуществующих объектов (только в ограниченном режиме); * distribution\_policy — проверка ограничений для случайно распределённых таблиц (только в ограниченном режиме); * namespace — проверка схем с отсутствующим определением схемы (только в ограниченном режиме); * pgclass — проверка записи pg\_class, не имеющей соответствующей записи pg\_attribute (только в ограниченном режиме). |
| -S {none | only} | Укажите этот параметр, чтобы контролировать тестирование таблиц каталога, которые используются во всех базах данных в установке RT.Warehouse, например, pg\_database. Значение none отключает тестирование таблиц общего каталога. Значение only проверяет только таблицы общего каталога. |
| -U *user\_name* | Пользователь, подключающийся к RT.Warehouse. |
| -? | Отображает интерактивную справку. |
| -v | Отображает подробную информацию о выполненных тестах. |

Примечания

Утилита определяет таблицы с отсутствующими атрибутами и отображает их в различных местах на выводе и в нестандартизированном формате. Утилита также отображает сводный список таблиц с отсутствующими атрибутами в формате database.schema.table.segment\_id после отображения выходной информации.

Если gpcheckcat обнаруживает несогласованную информацию OID (Object ID, идентификатор объекта), он генерирует один или несколько файлов проверки, содержащих SQL-запрос. Вы можете запустить SQL-запрос, чтобы просмотреть подробную информацию о несоответствиях OID и исследовать несоответствия. Файлы создаются в каталоге, в котором запускается gpcheckcat.

Формат файла:

|  |
| --- |
| gpcheckcat.verify.*dbname*.*catalog\_table\_name*.*test\_name*.*TIMESTAMP*.sql |

Пример файла проверки, созданный gpcheckcat при обнаружении несогласованных OID в таблице каталога pg\_type в базе данных mydb:

|  |
| --- |
| gpcheckcat.verify.mydb.pg\_type.missing\_extraneous.20150420102715.sql |

Пример запроса из файла подтверждения:

|  |
| --- |
| SELECT \*  FROM (  SELECT relname, oid FROM pg\_class WHERE reltype  IN (1305822,1301043,1301069,1301095)  UNION ALL  SELECT relname, oid FROM gp\_dist\_random('pg\_class') WHERE reltype  IN (1305822,1301043,1301069,1301095)  ) alltyprelids  GROUP BY relname, oid ORDER BY count(\*) desc ; |

gpcheckperf

Проверяет базовую производительность оборудования указанных хостов.

Синтаксис

|  |
| --- |
| gpcheckperf -d *test\_directory* [-d *test\_directory* ...]  {-f *hostfile\_gpcheckperf* | - h *hostname* [-h *hostname* ...]}  [-r ds] [-B *block\_size*] [-S *file\_size*] [-D] [-v|-V]  gpcheckperf -d *temp\_directory*  {-f *hostfile\_gpchecknet* | - h *hostname* [-h *hostname* ...]}  [ -r n|N|M [--duration *time*] [--netperf] ] [-D] [-v | -V]  gpcheckperf -?  gpcheckperf --version |

Описание

Утилита gpcheckperf запускает сеанс на указанных хостах и ​​выполняет следующие тесты производительности:

1. Тест производительности ввода-вывода диска (dd-тест) — для тестирования последовательной пропускной способности логического диска или файловой системы утилита использует команду dd, которая является стандартной утилитой UNIX. Она подсчитывает время, необходимое для записи и чтения большого файла на диск и с диска, и вычисляет производительность дискового ввода-вывода в мегабайтах в секунду (МБ/с). По умолчанию размер файла, который используется для теста, рассчитывается как удвоенный общий объём оперативной памяти (RAM) на хосте. Это гарантирует, что тест действительно тестирует дисковый ввод-вывод и не использует кэш памяти.

Тест пропускной способности памяти (потока) — для проверки пропускной способности памяти утилита использует программу тестирования STREAM для измерения устойчивой пропускной способности памяти (в МБ/с). Она проверяет, что производительность вашей системы не ограничена пропускной способностью памяти системы по сравнению с вычислительной производительностью ЦПУ. В приложениях с большим набором данных (например, в RT.Warehouse) низкая пропускная способность памяти является серьёзной проблемой для производительности. Если пропускная способность памяти значительно ниже теоретической пропускной способности ЦПУ, это может привести к тому, что ЦПУ будет тратить значительное количество времени на ожидание данных, поступающих из системной памяти.

Тест производительности сети (gpnetbench\*) — для проверки производительности сети (и, следовательно, производительности соединения с RT.Warehouse) утилита запускает программу тестирования производительности сети, которая передает 5-секундный поток данных с текущего хоста на каждый удалённый хост, включённый в тест. Данные передаются параллельно каждому удалённому хосту, а минимальная, максимальная, средняя и медианная скорость передачи данных по сети указывается в мегабайтах в секунду (МБ/с). Если суммарная скорость передачи ниже ожидаемой (менее 100 МБ/с), вы можете запустить тест сети последовательно, используя параметр -r n, чтобы получить результаты для каждого хоста. Чтобы запустить полноматричный тест пропускной способности, вы можете указать -r M, который заставит каждый хост отправлять и получать данные с любого другого указанного хоста. Этот тест лучше всего использовать для проверки того, может ли коммутационная матрица выдерживать полноматричную рабочую нагрузку.

Чтобы указать хосты для тестирования, используйте параметр -f, чтобы указать файл, содержащий список имён хостов, или используйте параметр ‑h, чтобы указать имена отдельных хостов в командной строке. При выполнении теста производительности сети все записи в файле хоста должны относиться к сетевым интерфейсам в одной подсети. Если хосты вашего сегмента имеют несколько сетевых интерфейсов, настроенных в разных подсетях, запустите тест сети один раз для каждой подсети.

Вы также должны указать хотя бы один тестовый каталог (с -d). Пользователь, запускающий gpcheckperf, должен иметь доступ на запись в указанные тестовые каталоги на всех удалённых хостах. Для теста дискового ввода-вывода тестовые каталоги должны соответствовать каталогам данных вашего сегмента (основного и/или зеркального). Для тестирования пропускной способности памяти и сети требуется временный каталог для файлов тестовой программы.

Перед использованием gpcheckperf вы должны настроить доверенный доступ между хостами, участвующими в тесте производительности. Вы можете использовать утилиту gpssh-exkeys для обновления известных файлов хоста и обмена открытыми ключами между хостами, если вы этого еще не сделали. Обратите внимание, что gpcheckperf вызывает gpssh и gpscp, поэтому эти утилиты RT.Warehouse также должны быть в вашем $PATH.

Параметры

Таблица — Параметры gpcheckperf

| Параметр | Описание |
| --- | --- |
| -B *block\_size* | Задаёт размер блока (в КБ или МБ) для использования при тестировании дискового ввода-вывода. Значение по умолчанию — 32 КБ, что соответствует размеру страницы RT.Warehouse. Максимальный размер блока — 1 МБ. |
| -d *test\_directory* | Для теста дискового ввода-вывода указывает расположение каталогов файловой системы для тестирования. У вас должен быть доступ для записи в тестовый каталог на всех хостах, участвующих в тесте производительности. Вы можете использовать параметр -d несколько раз, чтобы указать несколько тестовых каталогов (например, для тестирования дискового ввода-вывода ваших основных и зеркальных каталогов данных). |
| -d *temp\_directory* | Для сетевых и потоковых тестов указывает единственный каталог, в который файлы тестовой программы будут скопированы на время теста. У вас должен быть доступ на запись в этот каталог на всех хостах, участвующих в тесте. |
| -D | Сообщает результаты производительности для каждого хоста для тестов дискового ввода-вывода. По умолчанию в отчёт выводятся результаты только для хостов с минимальной и максимальной производительностью, а также общая и средняя производительность всех хостов. |
| --duration *time* | Задаёт продолжительность проверки сети в секундах (s), минутах (m), часах (h) или днях (d). По умолчанию — 15 секунд. |
| -f *hostfile\_gpcheckperf* | Для тестов дискового ввода-вывода и потоков указывает имя файла, содержащего одно имя хоста для каждого хоста, который будет участвовать в тесте производительности. Имя хоста является обязательным, и вы можете дополнительно указать альтернативное имя пользователя и/или номер порта SSH для каждого хоста. Синтаксис файла хоста — по одному хосту в строке следующим образом:   |  | | --- | | [*username*@]*hostname*[:*ssh\_port*] | |
| -f *hostfile\_gpchecknet* | Для теста производительности сети все записи в файле хоста должны относиться к адресам хоста в одной подсети. Если хосты вашего сегмента имеют несколько сетевых интерфейсов, настроенных в разных подсетях, запустите тест сети один раз для каждой подсети. Например (файл хоста, содержащий сегментные имена адресов хоста для интерконнект-подсети 1):   |  | | --- | | sdw1-1  sdw2-1  sdw3-1 | |
| -h *hostname* | Задаёт одно имя хоста (или адрес хоста), который будет участвовать в тесте производительности. Вы можете использовать параметр -h несколько раз, чтобы указать несколько имён хостов. |
| --netperf | Указывает, что для выполнения теста сети следует использовать бинарный файл netperf вместо теста сети RT.Warehouse. Чтобы использовать этот параметр, вы должны загрузить netperf с <http://www.netperf.org> и установить его в $GPHOME/bin/lib на всех хостах RT.Warehouse (мастере и сегментах). |
| -r ds{n|N|M} | Указывает, какие тесты производительности следует запустить.  По умолчанию используется dsn:   * Тест ввода-вывода диска (d). * Потоковый тест (s). * Тест производительности сети в последовательном (n), параллельном (N) или полноматричном (M) режиме. Необязательный параметр --duration указывает, как долго (в секундах) запускать сетевой тест. Чтобы использовать параллельный (N) режим, вы должны запустить тест на чётном количестве хостов. Если вы предпочитаете использовать netperf (<http://www.netperf.org>) вместо сетевого теста RT.Warehouse, вы можете загрузить его и установить в $GPHOME/bin/lib на всех хостах RT.Warehouse (мастере и сегментах). Вы должны указать необязательный параметр --netperf, чтобы использовать бинарный файл netperf вместо стандартных утилит gpnetbench\*. |
| -S *file\_size* | Задаёт общий размер файла, который будет использоваться для теста дискового ввода-вывода для всех каталогов, указанных с помощью -d. *file\_size* должен в два раза превышать общий объём RAM на хосте. Если параметр не указан, значение по умолчанию рассчитывается как удвоенный общий объём RAM на хосте, на котором выполняется gpcheckperf. Это гарантирует, что тест действительно тестирует дисковый ввод-вывод и не использует кэш памяти. Вы можете указать размер в КБ, МБ или ГБ. |
| -v | -V | В подробном режиме -v (verbose) отображаются сообщения о ходе и состоянии тестов производительности по мере их выполнения. В очень подробном режиме –V (very verbose) отображаются все выходные сообщения, генерируемые этой утилитой. |
| --version | Отображает версию этой утилиты. |
| -? (help) | Отображает интерактивную справку. |

Примечания

Запускает тесты дискового ввода-вывода и пропускной способности памяти на всех хостах в файле host\_file, используя тестовые каталоги /data1 и /data2:

|  |
| --- |
| $ gpcheckperf -f hostfile\_gpcheckperf -d /data1 -d /data2 -r ds |

Запускает только тест ввода-вывода диска на хостах с именами sdw1 и sdw2, используя тестовый каталог /data1. Показывает результаты отдельных хостов и запускает в подробном режиме:

|  |
| --- |
| $ gpcheckperf -h sdw1 -h sdw2 -d /data1 -r d -D -v |

Запускает тест параллельной сети, используя тестовый каталог /tmp, где hostfile\_gpcheck\_ic\* указывает имена всех адресов хостов сетевого интерфейса в одной подсети интерконнекта:

|  |
| --- |
| $ gpcheckperf -f hostfile\_gpchecknet\_ic1 -r N -d /tmp  $ gpcheckperf -f hostfile\_gpchecknet\_ic2 -r N -d /tmp |

Выполнение того же теста, что и выше, но с использованием netperf вместо сетевого теста RT.Warehouse (обратите внимание, что netperf должен быть установлен в $GPHOME/bin/lib на всех хостах RT.Warehouse):

|  |
| --- |
| $ gpcheckperf -f hostfile\_gpchecknet\_ic1 -r N --netperf -d /tmp  $ gpcheckperf -f hostfile\_gpchecknet\_ic2 -r N --netperf -d /tmp |

gpconfig

Устанавливает параметры конфигурации сервера для всех сегментов в системе RT.Warehouse.

Синтаксис

|  |
| --- |
| gpconfig -c *param\_name* -v *value* [-m *master\_value* | --masteronly]  | -r *param\_name* [--masteronly]  | -l  [--skipvalidation] [--verbose] [--debug]  gpconfig -s *param\_name* [--file | --file-compare] [--verbose] [--debug]  gpconfig --help |

Описание

Утилита gpconfig позволяет вам устанавливать, отменять или просматривать параметры конфигурации из файлов postgresql.conf всех инстансов (мастера, сегментов и зеркал) в вашей системе RT.Warehouse. При установке параметра вы также можете указать другое значение для мастера, если это необходимо. Например, такие параметры, как max\_connections, требуют настройки на мастере, отличной от той, которая используется для сегментов. Если вы хотите установить или отключить глобальный параметр или параметр только для мастера, используйте параметр --masteronly.

gpconfig можно использовать только для управления определёнными параметрами. Например, вы не можете использовать его для установки таких параметров, как порт, который должен быть отдельным для каждого инстанса сегмента. Используйте параметр -l (list), чтобы увидеть полный список параметров конфигурации, поддерживаемых gpconfig.

Когда gpconfig задаёт параметр конфигурации в файле postgresql.conf сегмента, новая установка параметра всегда отображается внизу файла. Когда вы используете gpconfig для удаления настройки параметра конфигурации, gpconfig комментирует параметр во всех сегментных файлах postgresql.conf, тем самым восстанавливая системные настройки по умолчанию. Например, если вы используете gpconfig, чтобы удалить (закомментировать) параметр, а затем добавить его обратно (установить новое значение), будет два экземпляра параметра; один закомментирован, а другой включён и вставлен в конец файла postgresql.conf.

После установки параметра необходимо перезапустить систему RT.Warehouse или перезагрузить файлы postgresql.conf, чтобы изменения вступили в силу. Требуется ли вам перезапуск или перезагрузка, зависит от параметра.

Чтобы показать текущие установленные значения параметра в системе, используйте параметр -s.

gpconfig использует следующие переменные среды для подключения к инстансу мастера RT.Warehouse и получения информации о конфигурации системы:

PGHOST;

PGPORT;

PGUSER;

PGPASSWORD;

PGDATABASE.

Параметры

Таблица — Параметры gpconfig

| Параметр | Описание |
| --- | --- |
| -c | --change *param\_name* | Изменяет настройку параметра конфигурации, добавляя новую настройку в конец файлов postgresql.conf. |
| -v | --value *value* | Значение, которое следует использовать для параметра конфигурации, указанного с параметром -c. По умолчанию это значение применяется ко всем сегментам, их зеркалам, мастеру и резервному мастеру. Утилита правильно цитирует значение при добавлении параметра в файлы postgresql.conf. Чтобы установить пустую строку вместо значения, введите пустые одинарные кавычки (''). |
| -m | --mastervalue *master\_value* | Значение для мастера, используемое для параметра конфигурации, указанного с помощью параметра -c. Если параметр указан, значение применяется только к мастеру и резервному мастеру. Этот параметр можно использовать только с -v. |
| --masteronly | Если параметр указан, gpconfig будет редактировать только файл postgresql.conf мастера. |
| -r | --remove *param\_name* | Удаляет настройку параметра конфигурации, закомментировав запись в файлах postgresql.conf. |
| -l | --list | Перечисляет все параметры конфигурации, поддерживаемые утилитой gpconfig. |
| -s | --show *param\_name* | Показывает значение параметра конфигурации, используемого на всех инстансах (мастере и сегментах) в системе RT.Warehouse. Если значения параметра среди инстансов различаются, утилита отображает сообщение об ошибке. Запуск gpconfig с параметром -s считывает значения параметров непосредственно из базы данных, а не из файла postgresql.conf. Если вы используете gpconfig для установки параметров конфигурации для всех сегментов, а затем запускаете gpconfig -s для проверки изменений, вы всё равно можете увидеть предыдущие (старые) значения. Вы должны перезагрузить файлы конфигурации (gpstop -u) или перезапустить систему (gpstop -r), чтобы изменения вступили в силу. |
| --file | Для параметра конфигурации показывает значение из файла postgresql.conf для всех инстансов (мастера и сегментов) в системе RT.Warehouse. Если значения параметра среди инстансов различаются, утилита отображает сообщение. Должен быть указан с параметром -s. Например, параметр конфигурации statement\_mem установлен на 64 МБ для пользователя с командой ALTER ROLE, а значение в файле postgresql.conf — 128 МБ. Выполнение команды gpconfig -s statement\_mem --file отображает 128 МБ. Команда gpconfig -s statement\_mem, запущенная пользователем, отображает 64 МБ. Невалидно с параметром --file-compare. |
| --file-compare | Для параметра конфигурации сравнивает текущее значение RT.Warehouse со значением в файлах postgresql.conf на хостах (мастере и сегментах). Значения в файлах postgresql.conf представляют значения RT.Warehouse при перезапуске. Если значения не совпадают, утилита отображает значения со всех хостов. Если все хосты имеют одинаковое значение, утилита отображает сводный отчёт. Невалидно с параметром --file. |
| --skipvalidation | Переопределяет системные проверки gpconfig и позволяет вам работать с любым параметром конфигурации сервера, включая скрытые параметры и параметры с ограничениями, которые не могут быть изменены с помощью gpconfig. При использовании с параметром -l (list) показывает список ограниченных параметров.  **Предупреждение**. Будьте предельно осторожны при настройке параметров конфигурации с помощью этого параметра. |
| --verbose | Отображает дополнительную информацию журнала во время выполнения команды gpconfig. |
| --debug | Устанавливает вывод журнала на уровень отладки. |
| -? | -h | --help | Отображает интерактивную справку. |

Примеры

Установить для параметра max\_connections значение 100 для всех сегментов и значение 10 для мастера:

|  |
| --- |
| gpconfig -c max\_connections -v 100 -m 10 |

Примеры синтаксиса, необходимого для обработки строкой оболочки bash:

|  |
| --- |
| gpconfig -c search\_path -v '"\$user",public'  gpconfig -c dynamic\_library\_path -v '\$libdir' |

Параметры конфигурации добавить в файл postgresql.conf:

|  |
| --- |
| search\_path='"$user",public'  dynamic\_library\_path='$libdir' |

Закомментировать все инстансы параметра конфигурации default\_statistics\_target и восстановить системное значение по умолчанию:

|  |
| --- |
| gpconfig -r default\_statistics\_target |

Перечислить все параметры конфигурации, поддерживаемые gpconfig:

|  |
| --- |
| gpconfig -l |

Показывать значения определённого параметра конфигурации в системе:

|  |
| --- |
| gpconfig -s max\_connections |

gpdeletesystem

Удаляет систему RT.Warehouse, которая была инициализирована с помощью gpinitsystem.

Синтаксис

|  |
| --- |
| gpdeletesystem [-d *master\_data\_directory*] [-B *parallel\_processes*]  [-f] [-l *logfile\_directory*] [-D]  gpdeletesystem -?  gpdeletesystem -v |

Описание

Утилита gpdeletesystem выполняет следующие действия:

1. Останавливает все процессы postgres (инстансы сегментов и инстанс мастера).

Удаляет все каталоги данных.

Перед запуском gpdeletesystem:

1. Переместите все файлы резервных копий из каталогов данных сегментов и мастера.

Убедитесь, что RT.Warehouse запущена.

1. Если вы в данный момент находитесь в каталоге данных сегмента, смените каталог на другое место. Утилита выдаёт ошибку при запуске из каталога данных сегмента.

Эта утилита не удаляет программное обеспечение RT.Warehouse.

Параметры

Таблица — Параметры gpdeletesystem

| Параметр | Описание |
| --- | --- |
| -d *master\_data\_directory* | Задаёт каталог данных мастер-хоста. Если этот параметр не указан, используется значение переменной среды MASTER\_DATA\_DIRECTORY. Если параметр указан, он переопределяет любые настройки MASTER\_DATA\_DIRECTORY. Если master\_data\_directory не может быть определён, утилита возвращает ошибку. |
| -B *parallel\_processes* | Количество сегментов для параллельного удаления. Если параметр не указан, утилита запустит до 60 параллельных процессов в зависимости от того, сколько инстансов сегментов необходимо удалить. |
| -f | Принудительное удаление, даже если файлы резервных копий найдены в каталогах данных. По умолчанию инстансы RT.Warehouse не удаляются при наличии файлов резервных копий. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -D | Устанавливает уровень ведения журнала для отладки. |
| -? | Отображает интерактивную справку. |
| -v | Отображает версию, статус, дату последнего обновления и контрольную сумму утилиты. |

Примеры

Удалить систему RT.Warehouse:

|  |
| --- |
| gpdeletesystem -d /gpdata/gp-1 |

Удалить систему RT.Warehouse, даже если файлы резервных копий присутствуют:

|  |
| --- |
| gpdeletesystem -d /gpdata/gp-1 -f |

gpexpand

Расширяет существующую RT.Warehouse на новые хосты в системе.

Синтаксис

|  |
| --- |
| gpexpand [{-f|--hosts-file} *hosts\_file*]  | {-i|--input} *input\_file* [-B *batch\_size*]  | [{-d | --duration} *hh:mm:ss* | {-e|--end} *'YYYY-MM-DD hh:mm:ss*']  [-a|-analyze]  [-n *parallel\_processes*]  | {-r|--rollback}  | {-c|--clean}  [-v|--verbose] [-s|--silent]  [{-t|--tardir} *directory* ]  [-S|--simple-progress ]  gpexpand -? | -h | --help  gpexpand --version |

Необходимые условия

1. Вы вошли в систему как суперпользователь RT.Warehouse (gpadmin).

Новые хосты сегмента были установлены и настроены в соответствии с существующими хостами сегмента, а именно:

Настроено оборудования и ОС;

Установлено программное обеспечение RT.Warehouse;

Создана учётная запись пользователя gpadmin;

Произведён обмен ключами SSH.

Имеется достаточно дискового пространства на хостах сегмента, чтобы временно разместить копию самой большой таблицы.

При перераспределении данных RT.Warehouse должна работать в производственном режиме. RT.Warehouse не может работать в ограниченном режиме или в мастер-режиме. Параметры gpstart -R или -m не могут быть указаны для запуска RT.Warehouse.

|  |
| --- |
| **Примечание.**  Данные утилиты нельзя запустить, пока gpexpand выполняет инициализацию сегмента:  gpbackup;  gpcheckcat;  gpconfig;  gppkg;  gprestore. |

Описание

Утилита gpexpand выполняет расширение системы в два этапа: инициализация инстанса сегмента и затем перераспределение данных таблицы.

На этапе инициализации gpexpand запускается с входным файлом, в котором указаны каталоги данных, значения dbid и другие характеристики инстансов нового сегмента. Вы можете создать входной файл вручную или следуя подсказкам в интерактивном режиме.

Если вы решили создать входной файл с помощью интерактивного режима, вы можете дополнительно указать файл, содержащий список хостов системы расширения. Если ваша платформа или командная оболочка ограничивает длину списка имён хостов, которые вы можете ввести при запросе в интерактивном режиме, указание хостов с помощью -f может быть обязательным.

Помимо инициализации инстансов сегмента, на этапе инициализации также создаётся схема расширения с именем gpexpand в базе данных postgres для хранения статуса операции расширения, включая подробный статус для таблиц.

На этапе перераспределения табличных данных gpexpand перераспределяет табличные данные, чтобы повторно сбалансировать данные между старым и новым инстансами сегментов.

|  |
| --- |
| **Примечание.**  Перераспределение данных следует выполнять в часы низкой нагрузки. Распределение можно разделить на batch-пакеты в течение длительного периода. |

Чтобы начать этап распределения, запустите gpexpand с параметрами ‑d (duration) или -e (end time) или без параметров. Если вы укажете время окончания или продолжительность, утилита будет перераспределять таблицы в схеме расширения до тех пор, пока не будет достигнуто указанное время окончания или продолжительность. Если вы не укажете никаких параметров, то этап перераспределения утилиты будет продолжаться до тех пор, пока все таблицы в схеме расширения не будут реорганизованы. Каждая таблица реорганизуется с помощью команд ALTER TABLE, чтобы повторно сбалансировать таблицы по новым сегментам и установить для таблиц их исходную политику распределения. Если gpexpand завершает реорганизацию всех таблиц, он отображает сообщение об успешном завершении и завершается.

|  |
| --- |
| **Примечание.**  Эта утилита использует SSH-соединения между системами для выполнения своих задач. В больших развёртываниях RT.Warehouse, облачных развёртываниях или развёртываниях с большим количеством сегментов на хост эта утилита может превышать максимальный порог хоста для неаутентифицированных соединений. Рассмотрите возможность обновления параметра конфигурации SSH MaxStartups, чтобы увеличить этот порог. Дополнительные сведения о параметрах конфигурации SSH см. в документации по SSH для вашего дистрибутива Linux. |

Параметры

Таблица — Параметры gpexpand

| Параметр | Описание |
| --- | --- |
| -a | --analyze | Запускает ANALYZE, чтобы обновить статистику таблицы после расширения. По умолчанию ANALYZE не запускается. |
| -B *batch\_size* | Размер batch-пакета удалённых команд для отправки на данный хост перед выполнением односекундной паузы. По умолчанию — 16. Допустимые значения — 1–128.  Утилита gpexpand выдаёт ряд команд настройки, которые могут превышать максимальный порог хоста для неаутентифицированных подключений, как определено MaxStartups в конфигурации демона SSH. Пауза в одну секунду позволяет завершить аутентификацию до того, как gpexpand выдаст еще какие-либо команды.  Значение по умолчанию обычно не нужно менять. Однако может потребоваться уменьшить максимальное количество команд, если gpexpand не работает с ошибками подключения, такими как 'ssh\_exchange\_identification: Connection closed by remote host.' |
| -c | --clean | Удаляет схему расширения. |
| -d | --duration *hh:mm:ss* | Продолжительность сеанса расширения от начала до конца. |
| -e | --end '*YYYY-MM-DD hh:mm:ss*' | Дата и время окончания сеанса расширения. |
| -f | --hosts-file *filename* | Задаёт имя файла, содержащего список новых хостов для расширения системы. Каждая строка файла должна содержать одно имя хоста.  Этот файл может содержать имена хостов с указанием сетевых интерфейсов или без них. Утилита gpexpand обрабатывает оба случая, добавляя номера интерфейсов в конец имени хоста, если исходные хосты настроены с несколькими сетевыми интерфейсами.  **Примечание**. Соглашение об именах хостов сегмента RT.Warehouse — sdwN, где sdw — префикс, а N — целое число. Например, sdw1, sdw2 и так далее. Для хостов с несколькими интерфейсами принято добавлять тире (-) и число к имени хоста. Например, sdw1-1 и sdw1-2 — это два имени интерфейса для хоста sdw1. |
| -i | --input *input\_file* | Задаёт имя файла конфигурации расширения, который содержит по одной строке для каждого добавляемого сегмента в формате: *hostname*|*address*|*port*|*datadir*|*dbid*|*content*|*preferred\_role*. |
| -n *parallel\_processes* | Количество таблиц для одновременного распределения. Допустимые значения — от 1 до 96. Каждый процесс перераспределения таблиц требует двух соединений с базой данных: одно для изменения таблицы, а другое для обновления статуса таблицы в схеме расширения. Перед увеличением -n проверьте текущее значение параметра конфигурации сервера max\_connections и убедитесь, что максимальный лимит подключений не превышен. |
| -r | --rollback | Откатывает неудачную операцию настройки расширения. |
| -s | --silent | Работает в тихом режиме. Не запрашивает подтверждение для продолжения предупреждений. |
| -S | --simple-progress | Если параметр указан, утилита gpexpand записывает только минимальную информацию о ходе выполнения в таблице gpexpand.expansion\_progress RT.Warehouse. Утилита не записывает информацию о размере отношения и информацию о состоянии в таблицу gpexpand.status\_detail. Указание этого параметра может повысить производительность за счёт уменьшения объёма информации о ходе выполнения, записываемой в таблицы gpexpand. |
| [-t | --tardir] *directory* | Полный путь к каталогу на хостах сегмента, куда утилита gpexpand копирует временный tar-файл. Этот файл содержит файлы RT.Warehouse, которые используются для создания инстансов сегментов. Каталог по умолчанию — это домашний каталог пользователя. |
| -v | --verbose | Подробный отладочный вывод. С этим параметром утилита выводит все DDL и DML, используемые для расширения базы данных. |
| --version | Отображает номер версии утилиты и выходит. |
| -? | -h | --help | Отображает интерактивную справку. |

Примеры

Запустить gpexpand с входным файлом, чтобы инициализировать новые сегменты и создать схему расширения в базе данных postgres:

|  |
| --- |
| $ gpexpand -i input\_file |

Запустить gpexpand на максимальную продолжительность шестьдесят часов, чтобы перераспределить таблицы в новые сегменты:

|  |
| --- |
| $ gpexpand -d 60:00:00 |

gpfdist

Обслуживает файлы данных или записывает файлы данных из сегментов RT.Warehouse.

Синтаксис

|  |
| --- |
| gpfdist [-d *directory*] [-p *http\_port*] [-P *last\_http\_port*] [-l *log\_file*]  [-t *timeout*] [-S] [-w *time*] [-v | -V] [-s] [-m *max\_length*]  [--ssl *certificate\_path* [--sslclean *wait\_time*] ]  [-c *config.yml*]  gpfdist -? | --help  gpfdist --version |

Описание

gpfdist — программа параллельного распространения файлов RT.Warehouse. Она используется читаемыми внешними таблицами и gpload для параллельного обслуживания файлов внешних таблиц для всех сегментов RT.Warehouse. Она используется внешними таблицами с возможностью записи для параллельного приёма выходных потоков из сегментов RT.Warehouse и их записи в файл.

|  |
| --- |
| **Примечание.**  gpfdist и gpload совместимы только с основной версией RT.Warehouse, в которой они поставляются. Например, утилита gpfdist, установленная с RT.Warehouse 4.x, не может использоваться с RT.Warehouse 5.x или 6.x. |

Чтобы gpfdist мог использоваться внешней таблицей, в предложении LOCATION определения внешней таблицы должны быть указаны данные внешней таблицы с использованием протокола gpfdist://.

|  |
| --- |
| **Примечание.**  Если для включения безопасности SSL указан параметр --ssl, создайте внешнюю таблицу с протоколом gpfdists://. |

Преимущество использования gpfdist заключается в том, что вам гарантируется максимальный параллелизм при чтении или записи во внешние таблицы, что обеспечивает наилучшую производительность, а также упрощает администрирование внешних таблиц.

Для доступных для чтения внешних таблиц gpfdist анализирует и равномерно передает файлы данных всем инстансам сегментов в системе RT.Warehouse, когда пользователи SELECT из внешней таблицы. Для внешних таблиц с возможностью записи gpfdist принимает параллельные выходные потоки из сегментов, когда пользователи INSERT во внешнюю таблицу, и записывает в выходной файл.

|  |
| --- |
| **Примечание.**  Когда gpfdist считывает данные и обнаруживает ошибку форматирования данных, сообщение об ошибке включает номер строки, указывающий местонахождение ошибки форматирования. gpfdist пытается захватить строку, содержащую ошибку. Однако gpfdist может не захватить точную строку для некоторых ошибок форматирования. |

Для читаемых внешних таблиц, если файлы загрузки сжимаются с использованием gzip или bzip2 (имеют расширение файла .gz или .bz2), gpfdist распаковывает файлы перед загрузкой. Для внешних таблиц с возможностью записи gpfdist сжимает данные с помощью gzip, если целевой файл имеет расширение .gz.

|  |
| --- |
| **Примечание.**  Сжатие не поддерживается для доступных для чтения и записи внешних таблиц, когда утилита gpfdist работает на платформах Windows. |

При чтении или записи данных с помощью протокола gpfdist или gpfdists RT.Warehouse включает X-GP-PROTO в заголовок запроса HTTP, чтобы указать, что запрос поступает из RT.Warehouse. Утилита отклоняет HTTP-запросы, не содержащие X-GP-PROTO в заголовке запроса.

Скорее всего, вы захотите запустить gpfdist на своих машинах ETL, а не на хостах, на которых установлена RT.Warehouse. Чтобы установить gpfdist на другой хост, просто скопируйте утилиту на этот хост и добавьте gpfdist в свой $PATH.

|  |
| --- |
| **Примечание.**  При использовании IPv6 всегда заключайте числовой IP-адрес в скобки. |

Параметры

Таблица — Параметры gpfdist

| Параметр | Описание |
| --- | --- |
| -d *directory* | Каталог, из которого gpfdist будет обслуживать файлы для читаемых внешних таблиц или создавать выходные файлы для доступных для записи внешних таблиц. Если параметр не указан, по умолчанию используется текущий каталог. |
| -l *log\_file* | Полный путь и имя файла журнала, в котором должны регистрироваться стандартные выходные сообщения. |
| -p *http\_port* | Порт HTTP, на котором gpfdist будет обслуживать файлы. По умолчанию — 8080. |
| -P *last\_http\_port* | Последний номер порта в диапазоне номеров портов HTTP (от http\_port до last\_http\_port включительно), на котором gpfdist будет пытаться обслуживать файлы. gpfdist обслуживает файлы по первому номеру порта в диапазоне, к которому он успешно привязывается. |
| -t *timeout* | Устанавливает время, отведённое RT.Warehouse для установления соединения с процессом gpfdist. По умолчанию — 5 секунд. Допустимые значения: от 2 до 7200 секунд (2 часа). Может потребоваться увеличить в системах с большим сетевым трафиком. |
| -m *max\_length* | Устанавливает максимально допустимую длину строки данных в байтах. По умолчанию — 32768. Параметр следует использовать, когда пользовательские данные содержат очень широкие строки (или когда появляется сообщение об ошибке line too long). Параметр не следует использовать иным способом, так как это увеличивает выделение ресурсов. Допустимый диапазон — от 32 КБ до 256 МБ (в системах Windows верхний предел составляет 1 МБ.)  **Примечание**. Проблемы с памятью могут возникнуть, если вы укажете большую максимальную длину строки и запустите большое количество одновременных подключений gpfdist. Например, установка этого значения на максимум 256 МБ с 96 параллельными процессами gpfdist потребует примерно 24 ГБ памяти ((96 + 1) x 246 МБ). |
| -s | Обеспечивает упрощённое ведение журнала. Когда указан этот параметр, в файл журнала gpfdist записываются только сообщения с уровнем WARN и выше. Сообщения уровня INFO не записываются в файл журнала. Если этот параметр не указан, все сообщения gpfdist записываются в файл журнала.  Вы можете указать этот параметр, чтобы уменьшить количество информации, записываемой в файл журнала. |
| -S | Открывает файл для синхронного I/O с флагом O\_SYNC. Любая запись фиксируется в результирующий блок дескриптора файла gpfdist до тех пор, пока данные не будут физически записаны на базовое оборудование. |
| -w *time* | Устанавливает количество секунд, в течение которых RT.Warehouse задерживает перед закрытием целевого файла, например, именованного канала. Значение по умолчанию — 0, т.е. без задержки. Максимальное значение — 7200 секунд (2 часа). Для RT.Warehouse с несколькими сегментами может быть задержка между сегментами при записи данных из разных сегментов в файл. Вы можете указать время ожидания, прежде чем RT.Warehouse закроет файл, чтобы гарантировать, что все данные будут записаны в файл. |
| --ssl *certificate\_path* | Добавляет SSL-шифрование к данным, передаваемым с помощью gpfdist. После выполнения gpfdist с параметром --ssl *certificate\_path* единственный способ загрузить данные с этого файлового сервера — использовать протокол gpfdist://.  Местоположение, указанное в *certificate\_path*, должно содержать следующие файлы:   * Файл сертификата сервера server.crt. * Файл закрытого ключа сервера server.key. * Доверенные центры сертификации root.crt.   Корневой каталог (/) не может быть указан как *certificate\_path*. |
| --sslclean *wait\_time* | Когда утилита запускается с параметром --ssl, задаётся количество секунд, на которое утилита задерживается перед закрытием SSL-сеанса и очисткой SSL-ресурсов после того, как завершится запись данных в сегмент RT.Warehouse или из него. Значение по умолчанию — 0, т.е. без задержки. Максимальное значение — 500 секунд. При увеличении задержки скорость передачи снижается. В некоторых случаях может возникать данная ошибка при копировании больших объёмов данных: gpfdist server closed connection. Чтобы избежать ошибки, вы можете добавить задержку, например --sslclean 5. |
| -c *config.yaml* | Задаёт правила, которые gpfdist использует для выбора преобразования, применяемого при загрузке или извлечении данных. Файл конфигурации gpfdist — это документ YAML 1.1. Данный параметр недоступен на платформах Windows. |
| -v | В подробном (verbose) режиме отображаются сообщения о ходе и состоянии. |
| -V | В очень подробном (very verbose) режиме отображаются все выходные сообщения, генерируемые этой утилитой. |
| -? | Отображает интерактивную справку. |
| --version | Отображает версию этой утилиты. |

Примечания

Параметр конфигурации сервера verify\_gpfdists\_cert определяет, включена ли проверка подлинности сертификата SSL, когда RT.Warehouse связывается с утилитой gpfdist для чтения или записи данных из внешнего источника данных. Вы можете установить для параметра значение false, чтобы отключить аутентификацию при тестировании связи между внешней таблицей RT.Warehouse и утилитой gpfdist, обслуживающей внешние данные. Если значение false, данные исключения SSL игнорируются:

1. Самоподписанный сертификат SSL, используемый gpfdist, не является доверенным для RT.Warehouse.

Имя хоста, содержащееся в сертификате SSL, не соответствует имени хоста, на котором выполняется gpfdist.

|  |
| --- |
| **Внимание.**  Отключение проверки подлинности SSL-сертификата представляет угрозу безопасности, поскольку не проверяет SSL-сертификат gpfdists. |

Если утилита gpfdist зависает при отсутствии операций чтения или записи, вы можете создать дамп ядра в следующий раз, когда произойдет зависание, чтобы помочь отладить проблему. Задайте для переменной среды GPFDIST\_WATCHDOG\_TIMER количество секунд бездействия для ожидания перед принудительным завершением работы gpfdist. Когда переменная среда установлена ​​и gpfdist зависает, утилита прекращает работу через указанное количество секунд, создаёт дамп ядра и отправляет информацию об аварийном завершении в файл журнала.

В этом примере переменная среда в системе Linux устанавливается так, что gpfdist завершает работу через 300 секунд (5 минут) бездействия.

|  |
| --- |
| export GPFDIST\_WATCHDOG\_TIMER=300 |

Примеры

Чтобы обслуживать файлы из указанного каталога через порт 8081 (и запускать gpfdist в фоновом режиме):

|  |
| --- |
| gpfdist -d /var/load\_files -p 8081 & |

Чтобы запустить gpfdist в фоновом режиме и перенаправить вывод и ошибки в файл журнала:

|  |
| --- |
| gpfdist -d /var/load\_files -p 8081 -l /home/gpadmin/log & |

Чтобы остановить gpfdist, когда он работает в фоновом режиме:

Сначала найдите его идентификатор процесса:

|  |
| --- |
| ps ax | grep gpfdist |

Затем завершите процесс, например:

|  |
| --- |
| kill 3456 |

gpinitstandby

Добавляет и/или инициализирует хост резервного мастера для системы RT.Warehouse.

Синтаксис

|  |
| --- |
| gpinitstandby { -s *standby\_hostname* [-P *port*] | -r | -n } [-a] [-q]  [-D] [-S *standby\_data\_directory*] [-l *logfile\_directory*]  gpinitstandby -v  gpinitstandby -? |

Описание

Утилита gpinitstandby добавляет инстанс резервного мастера в вашу систему RT.Warehouse. Если в вашей системе настроен существующий инстанс резервного мастера, используйте параметр -r, чтобы удалить его перед добавлением нового инстанса резервного мастера.

Перед запуском этой утилиты убедитесь, что программное обеспечение RT.Warehouse установлено на хосте резервного мастер и что вы обменялись ключами SSH между хостами. Рекомендуется установить в качестве порта мастера один и тот же номер порта на хосте мастера и хосте резервного мастера.

Данную утилиту следует запускать на активном в данный момент хосте основного мастера.

Утилита выполняет следующие действия:

1. Обновляет системный каталог RT.Warehouse для удаления информации о существующем резервном мастере (если указан параметр -r).

Обновляет системный каталог RT.Warehouse, чтобы добавить информацию об инстансе нового резервного мастера.

Редактирует файл pg\_hba.conf мастера RT.Warehouse, чтобы разрешить доступ только что добавленному резервному мастеру.

Устанавливает инстанс резервного мастера на альтернативном хосте мастера.

Запускает процесс синхронизации.

Инстанс резервного мастера служит «горячим резервом» на случай, если основной мастер перестанет работать. Резервный мастер поддерживается в актуальном состоянии с помощью процессов репликации журнала транзакций (walsender и walreceiver), которые выполняются на хостах основного мастера и резервного мастера и ​​обеспечивают синхронизацию данных между инстансами основного и резервного мастеров. Если основной мастер выходит из строя, процесс репликации журнала завершается, а резервный мастер может быть активирован вместо него с помощью утилиты gpactivatestandby. После активации резервного мастера реплицированные журналы используются для восстановления состояния инстанса мастера на момент последней успешно зафиксированной транзакции.

Активированный резервный мастер фактически становится мастером RT.Warehouse, принимая клиентские соединения на главном порту и выполняя обычные операции мастера, такие как обработка SQL-команд и управление ресурсами.

|  |
| --- |
| **Внимание.**  Если утилите gpinitstandby ранее не удалось инициализировать резервный мастер, перед повторным запуском gpinitstandby необходимо удалить файлы в каталоге данных резервного мастера. Каталог данных резервного мастера не очищается после сбоя инициализации, поскольку он содержит файлы журнала, которые могут помочь в определении причины сбоя.  В случае сбоя инициализации в каталоге хоста резервного мастера /tmp создаётся файл сводного отчёта. В файле отчёта перечислены каталоги на хосте резервного мастера, которые необходимо очистить. |

Параметры

Таблица — Параметры gpinitstandby

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -D | Устанавливает уровень ведения журнала для отладки. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -n | Укажите этот параметр, чтобы запустить резервный мастер RT.Warehouse, который был настроен, но остановлен по какой-либо причине. |
| -P *port* | Этот параметр указывает порт, который используется резервным мастером RT.Warehouse. По умолчанию используется тот же порт, который используется активным мастером RT.Warehouse. Если резервный мастер RT.Warehouse находится на том же хосте, что и активный мастер, порты должны быть разными. Если порты для активного и резервного мастера совпадают, а хост — один и тот же, утилита возвращает ошибку. |
| -q | Работает в тихом режиме. Вывод команды не отображается на экране, но всё равно записывается в файл журнала. |
| -r | Удаляет текущий настроенный инстанс резервного мастера из вашей системы RT.Warehouse. |
| -s *standby\_hostname* | Имя хоста резервного мастера. |
| -S *standby\_data\_directory* | Каталог данных, который будет использоваться для нового резервного мастера.  По умолчанию используется тот же каталог, что и у активного мастера. Если резервный мастер находится на том же хосте, что и активный мастер, с помощью этого параметра необходимо указать другой каталог. |
| -v | Отображает версию, статус, дату последнего обновления и контрольную сумму этой утилиты. |
| -? | Отображает интерактивную справку. |

Примеры

Добавить инстанс резервного мастера в свою систему RT.Warehouse и запустить процесс синхронизации:

|  |
| --- |
| gpinitstandby -s host09 |

Запустить существующий инстанс резервного мастера и синхронизировать данные с текущим инстансом мастера:

|  |
| --- |
| gpinitstandby -n |

|  |
| --- |
| **Примечание.**  Не указывайте параметры -n и -s в одной команде. |

Добавить инстанс резервного мастера в систему RT.Warehouse, указав другой порт:

|  |
| --- |
| gpinitstandby -s myhost -P 2222 |

Если вы указываете то же имя хоста, что и у активного мастер RT.Warehouse, вы также должны указать другой номер порта с параметром -P и каталог данных резервного мастера с параметром -S.

Удалить существующий резервный мастер из конфигурации системы RT.Warehouse:

|  |
| --- |
| gpinitstandby -r |

gpinitsystem

Инициализирует систему RT.Warehouse, используя параметры конфигурации, указанные в файле gpinitsystem\_config.

Синтаксис

|  |
| --- |
| gpinitsystem -c *cluster\_configuration\_file*  [-h *hostfile\_gpinitsystem*]  [-B *parallel\_processes*]  [-p *postgresql\_conf\_param\_file*]  [-s *standby\_master\_host*  [-P *standby\_master\_port*]  [-S *standby\_master\_datadir* | --standby\_datadir=*standby\_master\_datadir*]]  [-m *number* | --max\_connections=*number*]  [-b *size* | --shared\_buffers=*size*]  [-n *locale* | --locale=*locale*] [--lc-collate=*locale*]  [--lc-ctype=*locale*] [--lc-messages=*locale*]  [--lc-monetary=*locale*] [--lc-numeric=*locale*]  [--lc-time=*locale*] [-e *password* | --su\_password=*password*]  [--mirror-mode={group|spread}] [-a] [-q] [-l *logfile\_directory*] [-D]  [-I *input\_configuration\_file*]  [-O *output\_configuration\_file*]  gpinitsystem -v | --version  gpinitsystem -? | --help |

Описание

Утилита gpinitsystem создаёт инстанс RT.Warehouse или записывает входной файл конфигурации, используя значения, определённые в файле конфигурации кластера, и любые предоставленные вами параметры командной строки. См. Формат файла конфигурации инициализации для получения дополнительной информации о файле конфигурации. Перед запуском этой утилиты убедитесь, что вы установили программное обеспечение RT.Warehouse на все хосты в массиве.

С параметром -O *output\_configuration\_file* gpinitsystem не создаёт новый инстанс базы данных, а вместо этого записывает всю предоставленную информацию о конфигурации в указанный выходной файл. Этот файл использует параметры QD\_PRIMARY\_ARRAY и PRIMARY\_ARRAY для определения каждого члена с использованием его имени хоста, порта, каталога данных, префикса сегмента, идентификатора сегмента и идентификатора содержимого. Детали конфигурации массива могут быть изменены по мере необходимости для соответствия значениям, доступным в резервной копии RT.Warehouse, или могут просто использоваться для воссоздания той же конфигурации кластера в более позднее время. Файлы конфигурации, которые используют QD\_PRIMARY\_ARRAY и PRIMARY\_ARRAY, должны быть переданы в gpinitsystem с использованием параметра -I input\_configuration\_file. См. Формат файла конфигурации инициализации для получения дополнительной информации.

В СУБД RT.Warehouse каждый инстанс базы данных (мастер и все сегменты) должен быть инициализирован на всех хостах в системе таким образом, чтобы все они могли работать вместе как единая СУБД. Утилита gpinitsystem выполняет инициализацию инстансов мастера и каждого сегмента RT.Warehouse, а также настраивает систему в целом.

Перед запуском gpinitsystem вы должны установить переменную среды $GPHOME так, чтобы она указывала на расположение вашей установки RT.Warehouse на хосте мастера и обменивалась SSH-ключами между адресами всех хостов в массиве с помощью gpssh-exkeys.

Эта утилита выполняет следующие задачи:

1. Проверяет правильность параметров в файле конфигурации.

Обеспечивает возможность установления соединения с каждым адресом хоста. Если адрес хоста не может быть достигнут, утилита завершит работу.

Проверяет настройки локали.

Отображает конфигурацию, которая будет использоваться, и запрашивает у пользователя подтверждение.

Инициализирует инстанс мастера.

Инициализирует инстанс резервного мастера (если указан).

Инициализирует инстансы основных сегментов.

Инициализирует инстансы резервных сегментов (если настроено зеркалирование).

Настраивает систему RT.Warehouse и проверяет наличие ошибок.

Запускает систему RT.Warehouse.

|  |
| --- |
| **Примечание.**  Данная утилита использует SSH-соединения между системами для выполнения своих задач. В больших развёртываниях RT.Warehouse, облачных развёртываниях или развёртываниях с большим количеством сегментов на хост эта утилита может превышать максимальный порог хоста для неаутентифицированных соединений. Рассмотрите возможность обновления параметра конфигурации SSH MaxStartups, чтобы увеличить этот порог. Дополнительные сведения о параметрах конфигурации SSH смотрите в документации по SSH для вашего дистрибутива Linux. |

Параметры

Таблица — Параметры gpinitsystem

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -B *parallel\_processes* | Количество сегментов, которые нужно создать параллельно. Если не указано иное, утилита будет запускать до 4 параллельных процессов одновременно. |
| -c *cluster\_configuration\_file* | Обязательный. Полный путь и имя файла конфигурации, который содержит все определённые параметры для настройки и инициализации новой системы RT.Warehouse. Описание этого файла см. в Формат файла конфигурации инициализации Вы должны предоставить gpinitsystem либо параметр -c *cluster\_configuration\_file*, либо параметр -I *input\_configuration\_file*. |
| -D | Устанавливает уровень вывода журнала для отладки. |
| -h *hostfile\_gpinitsystem* | Опционально. Полный путь и имя файла, содержащего адреса хостов ваших сегментов. Если не указан в командной строке, вы можете указать файл хоста с помощью параметра MACHINE\_LIST\_FILE в файле gpinitsystem\_config. |
| -I *input\_configuration\_file* | Полный путь и имя файла входной конфигурации, который определяет элементы и сегменты RT.Warehouse с помощью параметров QD\_PRIMARY\_ARRAY и PRIMARY\_ARRAY. Входной файл конфигурации обычно создается с помощью gpinitsystem с параметром -O *output\_configuration\_file*. Вы должны предоставить gpinitsystem либо параметр -c *cluster\_configuration\_file*, либо параметр -I *input\_configuration\_file*. |
| -n *locale* | --locale=*locale* | Устанавливает языковой стандарт по умолчанию, используемый RT.Warehouse. Если не указано иное, переменная среда LC\_ALL, LC\_COLLATE или LANG хоста мастера определяет языковой стандарт. Если они не установлены, по умолчанию используется язык C (POSIX). Идентификатор локали состоит из идентификатора языка и идентификатора региона, а также, необязательно, кодировки набора символов.  Например, sv\_SE — это шведский, как говорят в Швеции, en\_US — американский английский, а fr\_CA — французско-канадский. Если для языкового стандарта может быть полезно более одного набора символов, тогда спецификации будут выглядеть следующим образом: en\_US.UTF-8 (спецификация языкового стандарта и кодировка набора символов). В большинстве систем команда locale покажет настройки среды локали, а locale -a покажет список всех доступных локалей. |
| --lc-collate=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для сопоставления (сортировки данных). Порядок сортировки нельзя изменить после инициализации RT.Warehouse, поэтому важно выбрать локаль сопоставления, совместимую с кодировками набора символов, которые вы планируете использовать для своих данных. Существует специальное имя сопоставления C или POSIX (сортировка по порядку байтов в отличие от сортировки по словарю). Параметры сортировки C могут использоваться с любой кодировкой символов. |
| --lc-ctype=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для классификации символов (какие последовательности символов допустимы и как они интерпретируются). Это невозможно изменить после инициализации базы данных RT.Warehouse, поэтому важно выбрать языковой стандарт классификации символов, совместимый с данными, которые вы планируете хранить в RT.Warehouse. |
| --lc-messages=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для вывода сообщений RT.Warehouse. Текущая версия RT.Warehouse не поддерживает несколько языков для выходных сообщений (все сообщения на английском языке), поэтому изменение этого параметра не повлияет. |
| --lc-monetary=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для форматирования денежных сумм. |
| --lc-numeric=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для форматирования чисел. |
| --lc-time=*locale* | Аналогично --locale, но устанавливает языковой стандарт, используемый для форматирования даты и времени. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -m *number* | --max\_connections=*number* | Устанавливает максимальное количество клиентских подключений, разрешённых мастеру. По умолчанию — 250. |
| -O *output\_configuration\_file* | При использовании с параметром -O gpinitsystem не создаёт новый кластер RT.Warehouse, а вместо этого записывает предоставленную информацию о конфигурации кластера в указанный файл output\_configuration\_file. Этот файл определяет элементы и сегменты RT.Warehouse, используя параметры QD\_PRIMARY\_ARRAY, PRIMARY\_ARRAY и MIRROR\_ARRAY, и позже может использоваться с -I *input\_configuration\_file* для инициализации нового кластера. |
| -p *postgresql\_conf\_param\_file* | Опционально. Имя файла, содержащего настройки параметров postgresql.conf, которые вы хотите установить для RT.Warehouse. Эти настройки будут использоваться при инициализации отдельных инстансов мастера и сегмента. Вы также можете установить параметры после инициализации с помощью утилиты gpconfig. |
| -q | Работать в тихом режиме. Вывод команды не отображается на экране, но все равно записывается в файл журнала. |
| -b *size* | --shared\_buffers=*size* | Устанавливает объём памяти, который инстанс сервера RT.Warehouse использует для буферов разделяемой памяти. Вы можете указать размер в килобайтах (КБ), мегабайтах (МБ) или гигабайтах (ГБ). По умолчанию — 125 МБ. |
| -s *standby\_master\_host* | Опционально. Если вы хотите настроить инстанс резервного мастера, укажите имя хоста с помощью этого параметра. Программное обеспечение RT.Warehouse уже должно быть установлено и настроено на этом хосте. |
| -P *standby\_master\_port* | Если вы настраиваете инстанс резервного мастера с помощью -s, укажите его номер порта с помощью этого параметра. Порт по умолчанию такой же, как и главный порт. Чтобы запустить резервный мастер и основной мастер на одном хосте, вы должны использовать этот параметр, чтобы указать другой порт для резервного. Программное обеспечение RT.Warehouse уже должно быть установлено и настроено на резервном хосте. |
| -S *standby\_master\_datadir* | --standby\_dir=*standby\_master\_datadir* | Если вы настраиваете хост резервного мастера с помощью -s, используйте этот параметр, чтобы указать его каталог данных. Если вы настраиваете резервный мастер на том же хосте, что и инстанс мастера, основной мастер и резервный мастер должны иметь отдельные каталоги данных. |
| -e *superuser\_password* | --su\_password=*superuser\_password* | Используйте этот параметр, чтобы указать пароль для учётной записи суперпользователя RT.Warehouse (например, gpadmin). Если этот параметр не указан, учётной записи суперпользователя назначается пароль по умолчанию gparray. Вы можете использовать команду ALTER ROLE, чтобы изменить пароль позже. Рекомендуемые методы безопасности:   * Не используйте параметр пароля по умолчанию для производственных сред. * Измените пароль сразу после установки. |
| --mirror\_mode={group|spread} | Используйте этот параметр, чтобы указать размещение инстансов зеркального сегмента на хостах сегмента. По умолчанию group группирует зеркальные сегменты для всех основных сегментов хоста на одном альтернативном хосте. spread распределяет зеркальные сегменты для основных сегментов на хосте по разным хостам в массиве RT.Warehouse.  Распространение разрешено только в том случае, если количество хостов превышает количество инстансов сегмента на хост. |
| -v | --version | Распечатывает версию gpinitsystem и выходит. |
| -? | --help | Показывает справку об аргументах командной строки gpinitsystem и выходит. |

Формат файла конфигурации инициализации

gpinitsystem требуется файл конфигурации кластера со следующими определёнными параметрами. Пример файла конфигурации инициализации можно найти в $GPHOME/docs/cli\_help/gpconfigs/gpinitsystem\_config.

Чтобы избежать конфликтов портов между RT.Warehouse и другими приложениями, номера портов RT.Warehouse не должны находиться в диапазоне, заданном параметром операционной системы net.ipv4.ip\_local\_port\_range. Например, если net.ipv4.ip\_local\_port\_range = 10000 65535, вы можете установить эти значения для номеров базовых портов RT.Warehouse.

|  |
| --- |
| PORT\_BASE = 6000  MIRROR\_PORT\_BASE = 7000 |

Таблица — Параметры gpinitsystem\_config

| Параметр | Описание |
| --- | --- |
| ARRAY\_NAME | Обязательный. Имя настраиваемого массива. Вы можете использовать любое имя, какое захотите. Заключите имя в кавычки, если имя содержит пробелы. |
| MACHINE\_LIST\_FILE | Опционально. Может использоваться вместо параметра -h. Параметр указывает на файл, содержащий список имён хостов сегментов, составляющих систему RT.Warehouse. Предполагается, что хостом мастера является хост, с которого вы запускаете утилиту, и его не следует включать в этот файл. Если хосты вашего сегмента имеют несколько сетевых интерфейсов, этот файл будет включать все адреса хоста. Укажите абсолютный путь к файлу. |
| SEG\_PREFIX | Обязательный. Параметр указывает префикс, который будет использоваться для именования каталогов данных на инстансах мастера и сегмента. Соглашение об именах каталогов данных в системе RT.Warehouse — SEG\_PREFIXnumber, где number начинается с 0 для инстансов сегмента (мастер всегда -1). Так, например, если вы выберете префикс gpseg, каталог данных вашего инстанса мастера будет называться gpseg-1, а инстансы сегментов будут называться gpseg0, gpseg1, gpseg2, gpseg3 и так далее. |
| PORT\_BASE | Обязательный. Параметр указывает базовое число, по которому вычисляются номера портов основного сегмента. Порт первого основного сегмента на хосте устанавливается как PORT\_BASE, а затем увеличивается на единицу для каждого дополнительного основного сегмента на этом хосте. Допустимые значения от 1 до 65535. |
| DATA\_DIRECTORY | Обязательный. Здесь указываются места хранения данных, в которых утилита будет создавать каталоги данных основных сегментов. Количество местоположений в списке определяет количество основных сегментов, которые будут созданы для каждого физического хоста (если в файле хоста указано несколько адресов хоста, количество сегментов будет равномерно распределено по указанным адресам интерфейса). Можно указать одну и ту же область хранения данных несколько раз, если вы хотите, чтобы каталоги данных создавались в одном месте. Пользователь, который запускает gpinitsystem (например, пользователь gpadmin), должен иметь разрешение на запись в эти каталоги. Например, создать шесть основных сегментов для каждого хоста:   |  | | --- | | declare -a DATA\_DIRECTORY=(/data1/primary /data1/primary  /data1/primary /data2/primary /data2/primary /data2/primary) | |
| MASTER\_HOSTNAME | Обязательный. Имя хоста инстанса мастера. Это имя хоста должно точно совпадать с настроенным именем хоста машины (запустите команду hostname, чтобы определить правильное имя хоста). |
| MASTER\_DIRECTORY | Обязательный. Параметр указывает место, где будет создан каталог данных на хосте мастера. Вы должны убедиться, что у пользователя, запускающего gpinitsystem (например, пользователя gpadmin), есть права на запись в этот каталог. |
| MASTER\_PORT | Обязательный. Номер порта для инстанса мастера. Это номер порта, который пользователи и клиентские соединения будут использовать при доступе к системе RT.Warehouse. |
| TRUSTED\_SHELL | Обязательный. Оболочка, которую утилита gpinitsystem использует для выполнения команд на удалённых хостах. Допустимые значения: ssh. Перед запуском утилиты gpinitsystem необходимо настроить среду доверенного хоста (для этого можно использовать gpssh-exkeys). |
| CHECK\_POINT\_SEGMENTS | Обязательный. Максимальное расстояние между контрольными точками журнала автоматической записи (WAL) в сегментах файла журнала (каждый сегмент обычно составляет 16 мегабайт). Это установит параметр checkpoint\_segments в файле postgresql.conf для каждого инстанса сегмента в системе RT.Warehouse. |
| ENCODING | Обязательный. Используемая кодировка набора символов. Этот набор символов должен быть совместим с используемыми настройками --locale, особенно --lc-collate и --lc-ctype. RT.Warehouse поддерживает те же наборы символов, что и PostgreSQL. |
| DATABASE\_NAME | Опционально. Имя базы данных RT.Warehouse, создаваемой после инициализации системы. Вы всегда можете создать базу данных позже, используя команду CREATE DATABASE или утилиту createdb. |
| MIRROR\_PORT\_BASE | Опционально. Параметр указывает базовый номер, по которому вычисляются номера портов зеркального сегмента. Порт первого зеркального сегмента на хосте устанавливается как MIRROR\_PORT\_BASE, а затем увеличивается на единицу для каждого дополнительного зеркального сегмента на этом хосте. Допустимые значения находятся в диапазоне от 1 до 65535 и не могут конфликтовать с портами, рассчитанными с помощью PORT\_BASE. |
| MIRROR\_DATA\_DIRECTORY | Опционально. Здесь указываются места хранения данных, где утилита будет создавать каталоги данных зеркальных сегментов. Для инстансов зеркального сегмента должно быть объявлено такое же количество каталогов данных, что и для инстансов основного сегмента (см. параметр DATA\_DIRECTORY). Пользователь, который запускает gpinitsystem (например, пользователь gpadmin), должен иметь разрешение на запись в эти каталоги. Например:   |  | | --- | | declare -a MIRROR\_DATA\_DIRECTORY=(/data1/mirror  /data1/mirror /data1/mirror /data2/mirror /data2/mirror  /data2/mirror) | |
| QD\_PRIMARY\_ARRAY, PRIMARY\_ARRAY, MIRROR\_ARRAY | Эти параметры могут быть предоставлены только с использованием входного файла конфигурации с параметром gpinitsystem -I input\_configuration\_file. QD\_PRIMARY\_ARRAY, PRIMARY\_ARRAY и MIRROR\_ARRAY определяют хост мастера RT.Warehouse, а также основной и зеркальный инстансы на хостах сегмента, соответственно, используя формат:   |  | | --- | | host~port~data\_directory/seg\_prefix<segment\_id>~dbid~content\_id |   Мастер RT.Warehouse всегда использует значение -1 для идентификатора сегмента и идентификатора содержимого. Например:   |  | | --- | | QD\_PRIMARY\_ARRAY=127.0.0.1~5432~/gpmaster/gpsne-1~1~-1~0  declare -a PRIMARY\_ARRAY=(  127.0.0.1~40000~/gpdata1/gpsne0~2~0  127.0.0.1~40001~/gpdata2/gpsne1~3~1  )  declare -a MIRROR\_ARRAY=(  127.0.0.1~50000~/gpmirror1/gpsne0~4~0  127.0.0.1~50001~/gpmirror2/gpsne1~5~1  ) |   Вы можете использовать gpinitsystem -O output\_configuration\_file для заполнения QD\_PRIMARY\_ARRAY, PRIMARY\_ARRAY, MIRROR\_ARRAY с использованием хостов, каталогов данных, префикса сегмента и значений базового порта, которые вы предоставляете команде. В целях восстановления вы можете изменить идентификаторы сегмента и контента, чтобы они соответствовали значениям существующей резервной копии RT.Warehouse. |
| HEAP\_CHECKSUM | Рпционально. Этот параметр определяет, использует ли gpinitsystem IP-адреса или имена хостов в файле pg\_hba.conf при обновлении файла с адресами, которые могут подключаться к RT.Warehouse. Значение по умолчанию — 0, т.е. утилита использует IP-адреса при обновлении файла. При инициализации системы RT.Warehouse укажите HBA\_HOSTNAMES = 1, чтобы утилита использовала имена хостов в файле pg\_hba.conf. |

Примеры

Инициализировать массив RT.Warehouse, предоставив файл конфигурации кластера и файл адреса хоста сегмента, и настроить конфигурацию зеркального распределённого отображения (--mirror-mode = spread):

|  |
| --- |
| $ gpinitsystem -c gpinitsystem\_config -h hostfile\_gpinitsystem --mirror-mode=spread |

Инициализировать массив RT.Warehouse и установить удалённый пароль суперпользователя:

|  |
| --- |
| $ gpinitsystem -c gpinitsystem\_config -h hostfile\_gpinitsystem --su-password=mypassword |

Инициализировать массив RT.Warehouse с дополнительным хостом резервного мастера:

|  |
| --- |
| $ gpinitsystem -c gpinitsystem\_config -h hostfile\_gpinitsystem -s host09 |

Вместо инициализации массива RT.Warehouse записать предоставленную конфигурацию в выходной файл. Выходной файл использует параметры QD\_PRIMARY\_ARRAY и PRIMARY\_ARRAY для определения хостов мастера и сегментов:

|  |
| --- |
| $ gpinitsystem -c gpinitsystem\_config -h hostfile\_gpinitsystem --mirror-mode=spread –O cluster\_init.config |

Инициализировать RT.Warehouse, используя входной файл конфигурации (файл, который определяет массив RT.Warehouse), используя параметры QD\_PRIMARY\_ARRAY и PRIMARY\_ARRAY:

|  |
| --- |
| $ gpinitsystem -I cluster\_init.config |

gpload

Выполняет задание загрузки, как определено в файле управления в формате YAML.

Синтаксис

|  |
| --- |
| gpload -f *control\_file* [-l *log\_file*] [-h *hostname*] [-p *port*]  [-U *username*] [-d *database*] [-W] [--gpfdist\_timeout *seconds*]  [--no\_auto\_trans] [[-v | -V] [-q]] [-D]  gpload -?  gpload --version |

Предварительные условия

Клиентская машина, на которой выполняется gpload, должна иметь следующее:

1. Python 2.6.2 или более новую версию, pygresql (интерфейс Python для PostgreSQL) и pyyaml. Обратите внимание, что Python и необходимые библиотеки Python включены в установку сервера RT.Warehouse, поэтому, если на машине, на которой выполняется gpload, установлена RT.Warehouse, отдельная установка Python не требуется.

|  |
| --- |
| **Примечание.**  Загрузчики RT.Warehouse для Windows поддерживают только Python 2.5 (доступен по адресу <https://www.python.org>). |

Программа параллельного распределения файлов gpfdist установлена ​​и находится в вашей $PATH. Данная программа находится в каталоге $GPHOME/bin вашего сервера RT.Warehouse.

Сетевой доступ ко всем хостам в массиве RT.Warehouse (мастер и сегменты) и от них.

Сетевой доступ к хостам, на которых находятся загружаемые данные (ETL-серверы), и от них.

Описание

gpload — это утилита загрузки данных, которая действует как интерфейс для функции параллельной загрузки внешней таблицы RT.Warehouse. Используя спецификацию загрузки, определённую в файле управления в формате YAML, gpload выполняет загрузку, вызывая параллельный файловый сервер RT.Warehouse (gpfdist), создавая определение внешней таблицы на основе определённых исходных данных и выполняя операцию INSERT, UPDATE или MERGE для загрузки исходных данных в целевую таблицу в базе данных.

|  |
| --- |
| **Примечание.**  gpfdist и gpload совместимы только с основной версией RT.Warehouse, в которой они поставляются. Например, утилита gpfdist, установленная с RT.Warehouse 4.x, не может использоваться с RT.Warehouse 5.x или 6.x. |

|  |
| --- |
| **Примечание.**  Операции MERGE и UPDATE не поддерживаются, если имя столбца целевой таблицы является зарезервированным ключевым словом, имеет заглавные буквы или включает любой символ, требующий кавычек ("") для идентификации столбца. |

Операция, включая любые SQL-команды, указанные в SQL-коллекции управляющего файла YAML (см. Формат файла управления), выполняется как одна транзакция, чтобы предотвратить несогласованность данных при выполнении нескольких одновременных операций загрузки в целевой таблице.

Параметры

Таблица — Параметры gpload

| Параметр | Описание |
| --- | --- |
| -f *control\_file* | Обязательный. Файл YAML, содержащий подробные сведения о спецификации загрузки. См. Формат файла управления. |
| --gpfdist\_timeout *seconds* | Устанавливает тайм-аут для программы параллельного распределения файлов gpfdist для отправки ответа. Введите значение от 0 до 30 секунд (значение 0 отключает таймауты). Обратите внимание, что вам может потребоваться увеличить это значение при работе в сетях с высоким трафиком. |
| -l *log\_file* | Указывает, куда записывать файл журнала. По умолчанию ~/gpAdminLogs/gpload\_*YYYYMMDD*. Дополнительные сведения о файле журнала см. Формат файла журнала. |
| --no\_auto\_trans | Укажите --no\_auto\_trans, чтобы отключить обработку операции загрузки как одной транзакции, если вы выполняете одну операцию загрузки в целевой таблице.  По умолчанию gpload обрабатывает каждую операцию загрузки как одну транзакцию, чтобы предотвратить несогласованность данных при выполнении нескольких одновременных операций над целевой таблицей. |
| -q | Работать в тихом режиме. Вывод команды не отображается на экране, но все равно записывается в файл журнала. |
| -D | Проверка наличия ошибок без выполнения загрузки. |
| -v | Показать подробный (verbose) вывод шагов загрузки по мере их выполнения. |
| -V | Показывает очень подробный (very verbose) вывод. |
| -? | Показать справку, затем выйти. |
| --version | Показать версию этой утилиты, затем выйти. |

Таблица — Параметры подключения gpload

| Параметр | Описание |
| --- | --- |
| -d *database* | База данных для загрузки. Если параметр не указан, читает значение из файла управления загрузкой, переменной среды $PGDATABASE или по умолчанию используется текущее имя пользователя системы. |
| -h *hostname* | Задаёт имя хоста машины, на которой работает сервер мастера базы данных RT.Warehouse. Если параметр не указан, читает значение из файла управления загрузкой, переменной среды $PGHOST или по умолчанию localhost. |
| -p *port* | Задаёт TCP-порт, на котором сервер мастера базы данных RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из файла управления загрузкой, переменной среды $PGPORT или по умолчанию 5432. |
| -U *username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из файла управления загрузкой, переменной среды $PGUSER или по умолчанию используется имя текущего пользователя системы. |
| -W | Принудительный ввод пароля. Если параметр не указан, считывает пароль из переменной среды $PGPASSWORD или из файла паролей, указанного в $PGPASSFILE или в ~/.pgpass. Если они не установлены, gpload запросит пароль, даже если -W не указан. |

Формат файла управления

Управляющий файл gpload использует формат документа YAML 1.1, а затем реализует свою собственную схему для определения различных этапов операции загрузки RT.Warehouse. Контрольный файл должен быть действующим документом YAML.

Программа gpload обрабатывает документ управляющего файла по порядку и использует отступы (пробелы) для определения иерархии документа и отношений разделов друг с другом. Использование белого пространства имеет большое значение. Пробелы не должны использоваться просто для форматирования, а табуляции вообще не должны использоваться.

Базовая структура файла управления загрузкой:

|  |
| --- |
| ---  VERSION: 1.0.0.1  DATABASE: db\_name  USER: db\_username  HOST: master\_hostname  PORT: master\_port  GPLOAD:  INPUT:  - SOURCE:  LOCAL\_HOSTNAME:  - hostname\_or\_ip  PORT: http\_port  | PORT\_RANGE: [start\_port\_range, end\_port\_range]  FILE:  - /path/to/input\_file  SSL: true | false  CERTIFICATES\_PATH: /path/to/certificates  - FULLY\_QUALIFIED\_DOMAIN\_NAME: true | false  - COLUMNS:  - field\_name: data\_type  - TRANSFORM: 'transformation'  - TRANSFORM\_CONFIG: 'configuration-file-path'  - MAX\_LINE\_LENGTH: integer  - FORMAT: text | csv  - DELIMITER: 'delimiter\_character'  - ESCAPE: 'escape\_character' | 'OFF'  - NULL\_AS: 'null\_string'  - FORCE\_NOT\_NULL: true | false  - QUOTE: 'csv\_quote\_character'  - HEADER: true | false  - ENCODING: database\_encoding  - ERROR\_LIMIT: integer  - LOG\_ERRORS: true | false  EXTERNAL:  - SCHEMA: schema | '%'  OUTPUT:  - TABLE: schema.table\_name  - MODE: insert | update | merge  - MATCH\_COLUMNS:  - target\_column\_name  - UPDATE\_COLUMNS:  - target\_column\_name  - UPDATE\_CONDITION: 'boolean\_condition'  - MAPPING:  target\_column\_name: source\_column\_name | 'expression'  PRELOAD:  - TRUNCATE: true | false  - REUSE\_TABLES: true | false  - STAGING\_TABLE: external\_table\_name  - FAST\_MATCH: true | false  SQL:  - BEFORE: "sql\_command"  - AFTER: "sql\_command" |

Таблица — Параметры файла управления gpload

| № | Параметр | Описание |
| --- | --- | --- |
| 1 | VERSION | Опционально. Версия схемы файла управления gpload. Текущая версия — 1.0.0.1. |
| 2 | DATABASE | Опционально. Указывает, к какой базе данных в системе RT.Warehouse подключиться. Если параметр не указан, по умолчанию используется $PGDATABASE, если установлено, или текущее имя пользователя системы. Вы также можете указать базу данных в командной строке, используя параметр -d. |
| 3 | USER | Опционально. Указывает, какую роль базы данных использовать для подключения. Если параметр не указан, по умолчанию используется текущий пользователь или $PGUSER, если установлено. Вы также можете указать роль базы данных в командной строке, используя параметр -U.  Если пользователь, запускающий gpload, не является суперпользователем RT.Warehouse, тогда ему должны быть предоставлены соответствующие права для обработки нагрузки. |
| 4 | HOST | Опционально. Задаёт имя хоста мастера RT.Warehouse. Если параметр не указан, по умолчанию используется localhost или $PGHOST, если установлено. Вы также можете указать имя хоста мастера в командной строке, используя параметр -h. |
| 5 | PORT | Опционально. Задаёт порт мастера RT.Warehouse. Если параметр не указан, по умолчанию 5432 или $PGPORT, если установлено. Вы также можете указать порт мастера в командной строке, используя параметр -p. |
| 6 | GPLOAD | Обязательный. Начинает раздел спецификации загрузки. В спецификации GPLOAD должны быть определены разделы INPUT и OUTPUT. |
| 6.1 | INPUT | Обязательный. Определяет расположение и формат загружаемых входных данных. gpload запустит один или несколько инстансов программы распределения файлов gpfdist на текущем хосте и создаст необходимые определения внешних таблиц в RT.Warehouse, которые указывают на исходные данные. Обратите внимание, что хост, с которого вы запускаете gpload, должен быть доступен по сети для всех хостов RT.Warehouse (мастера и сегментов). |
| 6.1.1 | SOURCE | Обязательный. Блок SOURCE спецификации INPUT определяет расположение исходного файла. В разделе INPUT может быть определено более одного блока SOURCE. Каждый определённый блок SOURCE соответствует одному инстансу программы распределения файлов gpfdist, которая будет запущена на локальной машине. Каждый определённый блок SOURCE должен иметь спецификацию FILE. |
| 6.1.1.1 | LOCAL\_HOSTNAME | Опционально. Задаёт имя хоста или IP-адрес локальной машины, на которой работает gpload. Если эта машина сконфигурирована с несколькими сетевыми интерфейсными картами (NIC), вы можете указать имя хоста или IP-адрес каждой отдельной NIC, чтобы разрешить сетевому трафику использовать все NIC одновременно. По умолчанию используется только имя основного хоста или IP-адрес локальной машины. |
| 6.1.1.2 | PORT | Опционально. Задаёт конкретный номер порта, который должна использовать программа распределения файлов gpfdist. Вы также можете указать PORT\_RANGE, чтобы выбрать доступный порт из указанного диапазона. Если определены и PORT, и PORT\_RANGE, приоритет имеет PORT. Если ни PORT, ни PORT\_RANGE не определены, по умолчанию выбирается доступный порт между 8000 и 9000. Если в LOCAL\_HOSTNAME объявлено несколько имён хостов, этот номер порта используется для всех хостов. Эта конфигурация желательна, если вы хотите использовать все сетевые адаптеры для загрузки одного и того же файла или набора файлов в заданном месте каталога. |
| 6.1.1.3 | PORT\_RANGE | Опционально. Может использоваться вместо PORT для предоставления диапазона номеров портов, из которых gpload может выбрать доступный порт для этого инстанса программы рапределения файлов gpfdist. |
| 6.1.1.4 | FILE | Обязательный. Задаёт расположение файла, именованного канала или каталога в локальной файловой системе, который содержит данные для загрузки. Вы можете объявить более одного файла, если данные имеют один и тот же формат во всех указанных файлах. Если файлы сжаты с использованием gzip или bzip2 (имеют расширение файла .gz или .bz2), файлы будут автоматически распакованы (при условии, что на вашем пути есть gunzip или bunzip2). При указании исходных файлов для загрузки вы можете использовать подстановочный знак (\*) или другое сопоставление с образцом в стиле C для обозначения нескольких файлов. Предполагается, что указанные файлы относятся к текущему каталогу, из которого выполняется gpload (или вы можете объявить абсолютный путь). |
| 6.1.1.5 | SSL | Опционально. Указывает использование SSL-шифрования. Если для SSL установлено значение true, gpload запускает сервер gpfdist с параметром --ssl и использует протокол gpfdists://. |
| 6.1.1.6 | CERTIFICATES\_PATH | Требуется, если для SSL значение true; не может быть указан, если SSL имеет значение false или unspecified. Местоположение, указанное в CERTIFICATES\_PATH, должно содержать следующие файлы:   * Файл сертификата сервера server.crt. * Файл закрытого ключа сервера server.key. * Доверенные центры сертификации root.crt.   Корневой каталог (/) не может быть указан как CERTIFICATES\_PATH. |
| 6.1.2 | FULLY\_QUALIFIED\_DOMAIN\_NAME | Опцционально. Указывает, разрешает ли gpload для имён хостов полное доменное имя (FQDN) или локальное имя хоста. Если установлено значение true, для имён разрешается полное доменное имя. Если установлено значение false, разрешается локальное имя хоста. По умолчанию — false. В некоторых ситуациях может потребоваться полное доменное имя. Например, если система RT.Warehouse находится в другом домене, чем приложение ETL, к которому обращается gpload. |
| 6.1.3 | COLUMNS | Опционально. Задаёт схему файла(ов) исходных данных в формате *field\_name*:*data\_type*. Символ DELIMITER в исходном файле разделяет два поля (столбца) значений данных. Строка определяется символом перевода строки (0x0a). Если входные COLUMNS не указаны, то подразумевается схема выходной TABLE, что означает, что исходные данные должны иметь тот же порядок столбцов, количество столбцов и формат данных, что и целевая таблица. Отображение источника и цели по умолчанию основано на совпадении имён столбцов, как определено в этом разделе, и имён столбцов в целевой TABLE. Это сопоставление по умолчанию можно переопределить с помощью раздела MAPPING. |
| 6.1.4 | TRANSFORM | Опционально. Задаёт имя входного преобразования, передаваемого в gpload. |
| 6.1.5 | TRANSFORM\_CONFIG | Требуется, если указан TRANSFORM. Задаёт расположение файла конфигурации преобразования, указанного выше в параметре TRANSFORM. |
| 6.1.6 | MAX\_LINE\_LENGTH | Опционально. Целое число, указывающее максимальную длину строки в данных преобразования XML, передаваемых в gpload. |
| 6.1.7 | FORMAT | Опционально. Задаёт формат файла(ов) исходных данных — обычный текст (TEXT) или формат значений, разделённых запятыми (CSV). Если не указано иное, по умолчанию используется TEXT. |
| 6.1.8 | DELIMITER | Опционально. Задаёт один символ ASCII, разделяющий столбцы в каждой строке данных. По умолчанию в режиме TEXT используется символ табуляции, в режиме CSV — запятая. Вы также можете указать непечатаемый символ ASCII или непечатаемый символ Юникода, например: "\x1B" или "\u001B". Синтаксис escape-строки, E'*character*-*code*', также поддерживается для непечатаемых символов. Символ ASCII или Unicode необходимо заключать в одинарные кавычки. Например: E'\x1B' или E'\u001B'. |
| 6.1.9 | ESCAPE | Задаёт одиночный символ, который используется для escape-последовательностей C (например, \n, \t, \100 и т.д.) И для escape-символов данных, которые в противном случае могли бы быть приняты как разделители строк или столбцов. Обязательно выберите escape-символ, который нигде не используется в ваших фактических данных столбца. Escape-символ по умолчанию \ (обратная косая черта) для файлов в текстовом формате и " (двойная кавычка) для файлов в формате CSV, однако можно указать другой символ для escape-представления. Также можно отключить escape в текст-форматированные файлы, указав в качестве escape-значения значение "OFF" .Это очень полезно для данных, таких как текстовые данные веб-журнала, которые содержат множество встроенных обратных косых черт, которые не предназначены для использования в качестве escape-символов. |
| 6.1.10 | NULL\_AS | Опционально. Задаёт строку, представляющую нулевое значение. По умолчанию в режиме TEXT используется \N (обратная косая черта-N), а в режиме CSV — пустое значение без кавычек. Вы можете предпочесть пустую строку даже в режиме TEXT для случаев, когда вы не хотите отличать нули от пустых строк. Любой элемент данных источника, соответствующий этой строке, будет считаться нулевым значением. |
| 6.1.11 | FORCE\_NOT\_NULL | Опционально. В режиме CSV обрабатывает каждый указанный столбец так, как если бы он был заключен в кавычки и, следовательно, не имел значения NULL. Для нулевой строки по умолчанию в режиме CSV (ничего между двумя разделителями) это приводит к тому, что отсутствующие значения оцениваются как строки нулевой длины. |
| 6.1.11 | QUOTE | Требуется, если FORMAT имеет значение CSV. Задаёт кавычки для режима CSV. По умолчанию используется двойная кавычка ("). |
| 6.1.12 | HEADER | Опционально. Указывает, что первая строка в файле(ах) данных является строкой заголовка (содержит имена столбцов) и не должна включаться в качестве данных для загрузки. При использовании нескольких файлов источников данных все файлы должны иметь строку заголовка. По умолчанию предполагается, что входные файлы не имеют строки заголовка. |
| 6.1.13 | ENCODING | Опционально. Кодировка набора символов исходных данных. Укажите строковую константу (например, 'SQL\_ASCII'), целочисленный номер кодировки или 'DEFAULT', чтобы использовать клиентскую кодировку по умолчанию. Если не указано, используется клиентская кодировка по умолчанию.  **Примечание**. Если вы измените значение ENCODING в существующем файле управления gpload, вы должны вручную удалить все внешние таблицы, которые создавались с использованием предыдущей конфигурации ENCODING. gpload не удаляет и не воссоздает внешние таблицы для использования нового ENCODING, если REUSE\_TABLES имеет значение true. |
| 6.1.14 | ERROR\_LIMIT | Опционально. Включает режим изоляции ошибок одной строки для этой операции загрузки. Если этот параметр включён, входные строки с ошибками формата будут отброшены при условии, что предельное количество ошибок не достигнуто ни в одном инстансе сегмента RT.Warehouse во время обработки ввода. Если предел ошибок не достигнут, все исправные строки будут загружены, а все строки с ошибками будут либо отброшены, либо захвачены как часть информации журнала ошибок. По умолчанию операция загрузки прерывается при первой обнаруженной ошибке. Обратите внимание, что изоляция ошибок одной строки применяется только к строкам данных с ошибками формата; например, лишние или отсутствующие атрибуты, атрибуты неправильного типа данных или недопустимые клиентские последовательности кодирования. Ошибки ограничений, такие как нарушения первичного ключа, все равно будут вызывать прерывание операции загрузки, если они встречаются. |
| 6.1.15 | LOG\_ERRORS | Опционально, если объявлен ERROR\_LIMIT. Значение может быть true или false. Значение по умолчанию — false. Если значение true, строки с ошибками форматирования регистрируются во внутреннем журнале при работе в режиме изоляции ошибок одной строки. Вы можете проверить ошибки форматирования с помощью встроенной SQL-функции RT.Warehouse gp\_read\_error\_log('*table\_name*'). Если при загрузке данных обнаруживаются ошибки форматирования, gpload генерирует предупреждающее сообщение с именем таблицы, которая содержит информацию об ошибке, аналогичную этому сообщению.   |  | | --- | | *timestamp*|WARN|1 bad row, please use GPDB built-in function gp\_read\_error\_log('table-name')  to access the detailed error row |   Если указан LOG\_ERRORS: true, необходимо указать REUSE\_TABLES: true, чтобы ошибки форматирования сохранялись в журналах ошибок RT.Warehouse. Если REUSE\_TABLES: true не указано, информация об ошибке удаляется после операции gpload. Возвращается только сводная информация об ошибках форматирования. Вы можете удалить ошибки форматирования из журналов ошибок с помощью функции RT.Warehouse gp\_truncate\_error\_log().  **Примечание**. Когда gpfdist считывает данные и обнаруживает ошибку форматирования данных, сообщение об ошибке включает номер строки, указывающий местонахождение ошибки форматирования. gpfdist пытается захватить строку, содержащую ошибку. Однако gpfdist может не захватить точную строку для некоторых ошибок форматирования. |
| 6.1.16 | EXTERNAL | Опционально. Определяет схему объектов базы данных внешних таблиц, созданных gpload. По умолчанию используется search\_path RT.Warehouse. |
| 6.1.16.1 | SCHEMA | Требуется при объявлении EXTERNAL. Имя схемы внешней таблицы. Если схема не существует, возвращается ошибка. Если указан % (символ процента), используется схема имени таблицы, заданная параметром TABLE в разделе OUTPUT. Если в имени таблицы не указана схема, используется схема по умолчанию. |
| 6.2 | OUTPUT | Обязательный. Определяет значения целевой таблицы и конечных столбцов данных, которые должны быть загружены в базу данных. |
| 6.2.1 | TABLE | Обязательный. Имя целевой таблицы для загрузки. |
| 6.2.2 | MODE | Опционально. Если не указано иное, по умолчанию используется INSERT. Доступны три режима загрузки:   * INSERT — загружает данные в целевую таблицу, используя следующий метод:  |  | | --- | | INSERT INTO *target\_table* SELECT \* FROM *input\_data*; |  * UPDATE — обновляет UPDATE\_COLUMNS целевой таблицы, в которой строки имеют значения атрибута MATCH\_COLUMNS, равные значениям атрибута входных данных, а необязательный UPDATE\_CONDITION имеет значение true. UPDATE не поддерживается, если имя столбца целевой таблицы является зарезервированным ключевым словом, имеет заглавные буквы или включает любой символ, требующий кавычек ("") для идентификации столбца; * MERGE — вставляет новые строки и обновляет UPDATE\_COLUMNS существующих строк, где значения атрибута FOOBAR равны значениям входных данных, а необязательный MATCH\_COLUMNS равен true. Новые строки идентифицируются, когда значение MATCH\_COLUMNS в исходных данных не имеет соответствующего значения в существующих данных целевой таблицы. В этих случаях вставляется вся строка из исходного файла, а не только столбцы MATCH и UPDATE. Если есть несколько новых значений MATCH\_COLUMNS, которые совпадают, будет вставлена ​​только одна новая строка для этого значения. Используйте UPDATE\_CONDITION, чтобы отфильтровать строки, которые нужно отбросить. MERGE не поддерживается, если имя столбца целевой таблицы является зарезервированным ключевым словом, имеет заглавные буквы или включает любой символ, требующий кавычек ("") для идентификации столбца. |
| 6.2.3 | MATCH\_COLUMNS | Требуется, если MODE имеет значения UPDATE или MERGE. Задаёт столбцы, используемые в качестве условия соединения для обновления. Значение атрибута в указанном целевом столбце(ах) должно быть равно значению соответствующего столбца(ов) исходных данных, чтобы строка в целевой таблице была обновлена. |
| 6.2.4 | UPDATE\_COLUMNS | Требуется, если MODE имеет значения UPDATE или MERGE. Задаёт столбцы для обновления для строк, которые соответствуют критерию MATCH\_COLUMNS и необязательному UPDATE\_CONDITION. |
| 6.2.5 | UPDATE\_CONDITION | Опционально. Задаёт логическое условие (аналогичное тому, которое вы бы объявили в предложении WHERE), которое должно быть выполнено, чтобы строка в целевой таблице была обновлена (или вставлена в случае MERGE). |
| 6.2.6 | MAPPING | Опционально. Если сопоставление указано, оно переопределяет сопоставление столбцов источника и назначения по умолчанию. Отображение источника и назначения по умолчанию основано на совпадении имён столбцов, как определено в разделе исходных COLUMNS, и имён столбцов целевой TABLE. Отображение задаётся как:  *target\_column\_name*: *source\_column\_name*  или  *target\_column\_name*: '*expression*'.  Где *expression* — это любое выражение, которое вы должны указать в списке SELECT запроса, например постоянное значение, ссылка на столбец, вызов оператора, вызов функции и прочее. |
| 7 | PRELOAD | Опционально. Задаёт операции, выполняемые перед операцией загрузки. Сейчас единственная операция предварительной загрузки — TRUNCATE. |
| 7.1 | TRUNCATE | Опционально. Если установлено значение true, gpload удалит все строки в целевой таблице перед её загрузкой. По умолчанию — false. |
| 7.2 | REUSE\_TABLES | Опционально. Если установлено значение true, gpload не будет отбрасывать объекты внешних таблиц и объекты промежуточных таблиц, которые он создаёт. Эти объекты будут повторно использоваться для будущих операций загрузки, использующих те же спецификации загрузки. Это улучшает производительность непрерывных загрузок (текущих небольших загрузок в одну и ту же целевую таблицу). Если задано значение LOG\_ERRORS: true, необходимо указать REUSE\_TABLES: true, чтобы ошибки форматирования сохранялись в журналах ошибок RT.Warehouse. Если REUSE\_TABLES: true не указано, информация об ошибке форматирования удаляется после операции gpload. Если external\_table\_name существует, утилита использует существующую таблицу. Утилита возвращает ошибку, если схема таблицы не соответствует схеме таблицы OUTPUT. |
| 7.3 | STAGING\_TABLE | Опционально. Укажите имя временной внешней таблицы, которая создается во время операции gpload. Внешняя таблица используется gpfdist. REUSE\_TABLES: true также необходимо указать. Если REUSE\_TABLES false или не указано, STAGING\_TABLE игнорируется. По умолчанию gpload создаёт временную внешнюю таблицу со случайно сгенерированным именем. Если external\_table\_name содержит точку (.), gpload возвращает ошибку. Если таблица существует, утилита использует её. Утилита возвращает ошибку, если существующая схема таблицы не соответствует схеме таблицы OUTPUT.  Утилита использует значение SCHEMA в разделе EXTERNAL как схему для external\_table\_name. Если значение SCHEMA равно %, схема для external\_table\_name такая же, как схема целевой таблицы, схема TABLE в разделе OUTPUT. Если SCHEMA не установлен, утилита выполняет поиск таблицы (используя схемы в базе данных search\_path). Если таблица не найдена, в схеме PUBLIC по умолчанию создаётся external\_table\_name. |
| 7.4 | FAST\_MATCH | Опционально. Если установлено значение true, gpload ищет в базе данных соответствующие объекты внешних таблиц только при повторном использовании внешних таблиц. Утилита не проверяет имена столбцов внешней таблицы и типы столбцов в таблице каталога pg\_attribute, чтобы гарантировать, что таблица может использоваться для операции gpload. Установите значение true, чтобы повысить производительность gpload при повторном использовании объектов внешней таблицы, а таблица каталога базы данных pg\_attribute содержит большое количество строк. Утилита возвращает ошибку и завершает работу, если определения столбцов несовместимы.  Значение по умолчанию — false, утилита проверяет имена столбцов определения внешней таблицы и типы столбцов. REUSE\_TABLES: true также необходимо указать. Если REUSE\_TABLES имеет значение false или не указано, а параметр FAST\_MATCH: true указан, gpload возвращает предупреждающее сообщение. |
| 8 | SQL | Опционально. Определяет команды SQL для запуска до и/или после операции загрузки. Вы можете указать несколько команд BEFORE и/или AFTER. Перечислите команды в желаемом порядке выполнения. |
| 8.1 | BEFORE | Опционально. Команда SQL, выполняемая перед началом операции загрузки. Заключите команды в кавычки. |
| 8.2 | AFTER | Опционально. Команда SQL, запускаемая после завершения операции загрузки. Заключите команды в кавычки. |

Формат файла журнала

Файлы журнала, выводимые gpload, имеют следующий формат:

|  |
| --- |
| *timestamp*|*level*|*message* |

Где *timestamp* имеет вид: YYYY-MM-DD HH:MM:SS, *level* — это одно из DEBUG, LOG, INFO, ERROR, а *message* — обычное текстовое сообщение.

Некоторые сообщения INFO, которые могут представлять интерес в файлах журнала (где # соответствует фактическому количеству секунд, единицам данных или ошибочным строкам):

|  |
| --- |
| INFO|running time: #.## seconds  INFO|transferred #.# kB of #.# kB.  INFO|gpload succeeded  INFO|gpload succeeded with warnings  INFO|gpload failed  INFO|1 bad row  INFO|# bad rows |

Примечания

Если имена объектов вашей базы данных были созданы с использованием идентификатора, заключенного в двойные кавычки (идентификатора с разделителями), вы должны указать имя с разделителями в одинарных кавычках в управляющем файле gpload. Например, если вы создаёте таблицу следующим образом:

|  |
| --- |
| CREATE TABLE "MyTable" ("MyColumn" text); |

Ваш файл управления gpload в формате YAML будет ссылаться на приведённые выше имена таблиц и столбцов следующим образом:

|  |
| --- |
| - COLUMNS:  - '"MyColumn"': text  OUTPUT:  - TABLE: public.'"MyTable"' |

Если управляющий файл YAML содержит элемент ERROR\_TABLE, который был доступен в RT.Warehouse 4.3.x, gpload регистрирует предупреждение о том, что ERROR\_TABLE не поддерживается, а ошибки загрузки обрабатываются, как если бы для элементов LOG\_ERRORS и REUSE\_TABLE было установлено значение true. Строки с ошибками форматирования регистрируются во внутреннем журнале при работе в режиме изоляции ошибок одной строки.

Примеры

Запустить задание загрузки, как определено в my\_load.yml:

|  |
| --- |
| gpload -f my\_load.yml |

Пример файла управления загрузкой:

|  |
| --- |
| ---  VERSION: 1.0.0.1  DATABASE: ops  USER: gpadmin  HOST: mdw-1  PORT: 5432  GPLOAD:  INPUT:  - SOURCE:  LOCAL\_HOSTNAME:  - etl1-1  - etl1-2  - etl1-3  - etl1-4  PORT: 8081  FILE:  - /var/load/data/\*  - COLUMNS:  - name: text  - amount: float4  - category: text  - descr: text  - date: date  - FORMAT: text  - DELIMITER: '|'  - ERROR\_LIMIT: 25  - LOG\_ERRORS: true  OUTPUT:  - TABLE: payables.expenses  - MODE: INSERT  PRELOAD:  - REUSE\_TABLES: true  SQL:  - BEFORE: "INSERT INTO audit VALUES('start', current\_timestamp)"  - AFTER: "INSERT INTO audit VALUES('end', current\_timestamp)" |

gplogfilter

Ищет указанные записи в файлах журнала RT.Warehouse.

Синтаксис

|  |
| --- |
| gplogfilter [*timestamp\_options*] [*pattern\_options*]  [*output\_options*] [*input\_options*] [*input\_file*]  gplogfilter --help  gplogfilter --version |

Описание

Утилиту gplogfilter можно использовать для поиска в файле журнала RT.Warehouse записей, соответствующих указанным критериям. Если входной файл не предоставлен, то gplogfilter будет использовать переменную среды $MASTER\_DATA\_DIRECTORY, чтобы найти файл журнала мастера RT.Warehouse в стандартном месте ведения журнала. Для чтения из стандартного ввода используйте тире (-) в качестве имени входного файла. Входные файлы могут быть сжаты с помощью gzip. Во входном файле запись журнала идентифицируется своей меткой времени в формате YYYY-MM-DD [hh:mm[:ss]].

Вы также можете использовать gplogfilter для поиска сразу во всех файлах журнала сегментов, запустив его через утилиту gpssh. Например, чтобы отобразить последние три строки каждого файла журнала сегмента:

|  |
| --- |
| gpssh -f seg\_host\_file  => source /usr/local/greenplum-db/greenplum\_path.sh  => gplogfilter -n 3 /gpdata/\*/pg\_log/gpdb\*.csv |

По умолчанию вывод gplogfilter отправляется на стандартный вывод. Используйте параметр -o, чтобы отправить вывод в файл или каталог. Если вы укажете имя выходного файла, заканчивающееся на .gz, выходной файл будет сжат по умолчанию с использованием максимального сжатия. Если местом назначения вывода является каталог, выходному файлу даётся то же имя, что и входному файлу.

Параметры

Таблица — timestamp параметры gplogfilter

| Параметр | Описание |
| --- | --- |
| -b *datetime* | --begin=*datetime* | Указывает начальную дату и время начала поиска записей журнала в формате YYYY-MM-DD [hh:mm[:ss]].  Если указано время, дату и время следует заключить в одинарные или двойные кавычки. В этом примере дата и время заключаются в одинарные кавычки:   |  | | --- | | gplogfilter -b '2013-05-23 14:33' | |
| -e *datetime* | --end=*datetime* | Указывает конечную дату и время завершения поиска записей журнала в формате YYYY-MM-DD [hh:mm[:ss]].  Если время указано, дату и время необходимо заключить в одинарные или двойные кавычки. В этом примере дата и время заключаются в одинарные кавычки:   |  | | --- | | gplogfilter -e '2013-05-23 14:33' | |
| -d *time* | --duration=*time* | Задаёт продолжительность поиска записей журнала в формате [hh][:mm[:ss]]. При использовании без параметра -b или -e в качестве основы будет использоваться текущее время. |

Таблица 37 — pattern matching параметры gplogfilter

| Параметр | Описание |
| --- | --- |
| -c i [gnore] | r [espect] | --case=i [gnore] | r [espect] | Сопоставление буквенных символов по умолчанию чувствительно к регистру, если только не используется параметр --case=ignore. |
| -C '*string*' | --columns='*string*' | Выбирает определённые столбцы из файла журнала. Задайте нужные столбцы в виде строки номеров столбцов, разделённых запятыми, которые начинаются с 1, где второй столбец 2, третий 3 и т.д. |
| -f '*string*' | --find='*string*' | Находит записи журнала, содержащие указанную строку. |
| -F '*string*' | --nofind='*string*' | Отклоняет записи журнала, содержащие указанную строку. |
| -m *regex* | --match=*regex* | Находит записи журнала, соответствующие указанному регулярному выражению Python. См. <https://docs.python.org/library/re.html> для синтаксиса регулярных выражений Python. |
| -M *regex* | --nomatch=*regex* | Отклоняет записи журнала, соответствующие указанному регулярному выражению Python. См. <https://docs.python.org/library/re.html> для синтаксиса регулярных выражений Python. |
| -t | --trouble | Находит только те записи журнала, в первой строке которых есть ERROR:, FATAL: или PANIC:. |

Таблица 38 — output параметры gplogfilter

| Параметр | Описание |
| --- | --- |
| -n *integer* | --tail=*integer* | Ограничивает вывод последним *integer* найденных подходящих записей журнала. |
| -s *offset* [limit] | --slice=*offset* [limit] | Из списка подходящих записей журнала возвращает ограниченное (*limit*) количество записей, начиная с номера *offset* записи, где *offset* равный нулю (0) обозначает первую запись в наборе результатов, а *offset* любого числа больше нуля отсчитывается от конца набора результатов. |
| -o *output\_file* | --out=*output\_file* | Записывает вывод в указанный файл или каталог вместо STDOUT. |
| -z 0-9 | --zip=0-9 | Сжимает выходной файл до указанного уровня сжатия с помощью gzip, где 0 — отсутствие сжатия, а 9 — максимальное сжатие. Если вы укажете имя выходного файла, заканчивающееся на .gz, выходной файл будет сжат по умолчанию с использованием максимального сжатия. |
| -a | --append | Если выходной файл уже существует, добавляется к файлу, а не перезаписывается. |

Таблица 39 — input параметры gplogfilter

| Параметр | Описание |
| --- | --- |
| input\_file | Имя входного файла(ов) журнала для поиска. Если входной файл не указан, gplogfilter будет использовать переменную среды $MASTER\_DATA\_DIRECTORY для поиска файла журнала мастера RT.Warehouse. Для чтения из стандартного ввода используйте тире (-) в качестве имени входного файла. |
| -u | --unzip | Распаковать входной файл с помощью gunzip. Если имя входного файла заканчивается на .gz, по умолчанию он не будет сжат. |
| --help | Отображать интерактивную справку. |
| --version | Отображать версию этой утилиты. |

Примеры

Отобразить последние три сообщения об ошибках в файле журнала мастера:

|  |
| --- |
| gplogfilter -t -n 3 |

Показать все сообщения журнала в файле журнала мастера с отметкой времени за последние 10 минут:

|  |
| --- |
| gplogfilter -d :10 |

Отображение сообщений журнала в файле журнала мастера, содержащем строку |con6 cmd11|:

|  |
| --- |
| gplogfilter -f '|con6 cmd11|' |

Используя gpssh, запустить gplogfilter на хостах сегмента и найти сообщения журнала в файлах журнала сегмента, содержащих строку con6, и сохраните вывод в файл.

|  |
| --- |
| gpssh -f seg\_hosts\_file -e 'source  /usr/local/greenplum-db/greenplum\_path.sh ; gplogfilter -f  con6 /gpdata/\*/pg\_log/gpdb\*.csv' > seglog.out |

gpmapreduce

Выполняет задания RT.Warehouse MapReduce, как определено в документе спецификации YAML.

Синтаксис

|  |
| --- |
| gpmapreduce -f *config.yaml* [*dbname* [*username*]]  [-k *name=value* | --key *name=value*]  [-h *hostname* | --host *hostname*] [-p *port*| --port *port*]  [-U *username* | --username *username*] [-W] [-v]  gpmapreduce -x | --explain  gpmapreduce -X | --explain-analyze  gpmapreduce -V | --version  gpmapreduce -h | --help |

Предварительные условия

Перед запуском данной программы необходимо следующее:

1. У вас должно быть задание MapReduce, определённое в файле YAML. См. gpmapreduce.yaml для получения дополнительной информации о формате и ключевых словах, поддерживаемых в файле конфигурации RT.Warehouse MapReduce YAML.

Вы должны быть суперпользователем RT.Warehouse, чтобы запускать задания MapReduce, написанные на Perl или Python.

Вы должны быть суперпользователем RT.Warehouse, чтобы запускать задания MapReduce с входными параметрами EXEC и FILE.

Вы должны быть суперпользователем RT.Warehouse, чтобы запускать задания MapReduce с вводом GPFDIST, если пользователю не предоставлены соответствующие права.

Описание

MapReduce— это модель программирования, разработанная Google для обработки и создания больших наборов данных на массиве обычных серверов. RT.Warehouse MapReduce позволяет программистам, знакомым с парадигмой MapReduce, писать функции map и reduce и отправлять их в параллельный маханизм RT.Warehouse для обработки.

gpmapreduce — это программа RT.Warehouse MapReduce. Вы настраиваете задание RT.Warehouse MapReduce через файл конфигурации в формате YAML, который вы передаёте программе для выполнения параллельным механизмом RT.Warehouse. Система RT.Warehouse распределяет входные данные, выполняет программу через набор машин, обрабатывает сбои машин и управляет необходимой связью между нашин.

Параметры

Таблица — Параметры gpmapreduce

| Параметр | Описание |
| --- | --- |
| -f *config.yaml* | Обязательный. Файл YAML, содержащий определения заданий RT.Warehouse MapReduce. См. gpmapreduce.yaml, чтобы узнать о формате и содержании параметров, которые вы указываете в этом файле. |
| -? | --help | Показать справку, затем выйти. |
| -V | --version | Показать информацию о версии, затем выйти. |
| -v | --verbose | Показать подробный вывод. |
| -x | --explain | Не запускать задания MapReduce, а создать план выполнения. |
| -X | --explain-analyze | Запустить задания MapReduce и создать план выполнения и анализа. |
| -k | --keyname=*value* | Установить переменную YAML. Требуется значение. По умолчанию используется key, если имя переменной не указано. |

Таблица — Параметры подключения gpmapreduce

| Параметр | Описание |
| --- | --- |
| -h *host* | --host *host* | Задаёт имя хоста машины, на которой работает сервер мастера RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port *port* | Задает TCP-порт, на котором сервер мастера RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username *username* | Имя роли базы данных для подключения. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется текущее имя пользователя системы. |
| -W | --password | Принудительно ввести пароль. |

Примеры

Запустить задание MapReduce, как определено в my\_mrjob.yaml, и подключиться к базе данных mydatabase:

|  |
| --- |
| gpmapreduce -f my\_mrjob.yaml mydatabase |

gpmapreduce.yaml

Файл конфигурации gpmapreduce.

Синтаксис

|  |
| --- |
| %YAML 1.1  ---  VERSION: 1.0.0.2  DATABASE: *dbname*  USER: *db\_username*  HOST: *master\_hostname*  PORT: *master\_port*  DEFINE:  - INPUT:  NAME: *input\_name*  FILE:  - *hostname:/path/to/file*  GPFDIST:  - *hostname:port/file\_pattern*  TABLE: *table\_name*  QUERY: *SELECT\_statement*  EXEC: *command\_string*  COLUMNS:  - *field\_name data\_type*  FORMAT: TEXT | CSV  DELIMITER: *delimiter\_character*  ESCAPE: *escape\_character*  NULL: *null\_string*  QUOTE: *csv\_quote\_character*  ERROR\_LIMIT: *integer*  ENCODING: *database\_encoding*  - OUTPUT:  NAME: *output\_name*  FILE: *file\_path\_on\_client*  TABLE: *table\_name*  KEYS:  - *column\_name*  MODE: REPLACE | APPEND  - MAP:  NAME: *function\_name*  FUNCTION: *function\_definition*  LANGUAGE: perl | python | c  LIBRARY: */path/filename.so*  PARAMETERS:  - *nametype*  RETURNS:  - *nametype*  OPTIMIZE: STRICT IMMUTABLE  MODE: SINGLE | MULTI  - TRANSITION | CONSOLIDATE | FINALIZE:  NAME: *function\_name*  FUNCTION: *function\_definition*  LANGUAGE: perl | python | c  LIBRARY: */path/filename.so*  PARAMETERS:  - *nametype*  RETURNS:  - *nametype*  OPTIMIZE: STRICT IMMUTABLE  MODE: SINGLE | MULTI  - REDUCE:  NAME: *reduce\_job\_name*  TRANSITION: *transition\_function\_name*  CONSOLIDATE: *consolidate\_function\_name*  FINALIZE: *finalize\_function\_name*  INITIALIZE: *value*  KEYS:  - *key\_name*  - TASK:  NAME: *task\_name*  SOURCE: *input\_name*  MAP: *map\_function\_name*  REDUCE: *reduce\_function\_name*  EXECUTE:  - RUN:  SOURCE: *input\_or\_task\_name*  TARGET: *output\_name*  MAP: *map\_function\_name*  REDUCE: *reduce\_function\_name*... |

Описание

Вы указываете задачи input, map и reduce, и output для программы RT.Warehouse MapReduce gpmapreduce в файле конфигурации в формате YAML. Данный раздел использует имя gpmapreduce.yaml для данного файла; вы можете выбрать собственное имя для файла.

Утилита gpmapreduce обрабатывает файл конфигурации YAML по порядку, используя отступы (пробелы) для определения иерархии документа и отношений между разделами. Использование белого пространства в файле является значительным.

Ключи и значения

Таблица — Параметры gpmapreduce.yaml

| № | Параметр | Описание |
| --- | --- | --- |
| 1 | VERSION | Обязательный. Версия YAML-спецификации RT.Warehouse MapReduce. Текущие поддерживаемые версии: 1.0.0.1, 1.0.0.2 и 1.0.0.3. |
| 2 | DATABASE | Опционально. Указывает, к какой базе данных в RT.Warehouse подключаться. Если параметр не указан, по умолчанию используется база данных по умолчанию или $PGDATABASE, если задано. |
| 3 | USER | Опционально. Указывает, какую роль базы данных использовать для подключения. Если параметр не указан, по умолчанию используется текущий пользователь или $PGUSER, если установлено. Вы должны быть суперпользователем RT.Warehouse, чтобы запускать функции, написанные на Python и Perl. Обычные пользователи базы данных могут запускать функции, написанные на Perl. Вы также должны быть суперпользователем базы данных, чтобы запускать задания MapReduce, которые содержат типы ввода FILE, GPFDIST и EXEC. |
| 4 | HOST | Опционально. Задаёт имя хоста мастера RT.Warehouse. Если параметр не указан, по умолчанию используется localhost или $PGHOST, если установлено. |
| 5 | PORT | Опционально. Задаёт порт мастера RT.Warehouse. Если параметр не указан, по умолчанию 5432 или $PGPORT, если установлено. |
| 6 | DEFINE | Обязательный. Последовательность определений для этого документа MapReduce. В разделе DEFINE должно быть хотя бы одно определение INPUT. |
| 6.1 | INPUT | Обязательный. Определяет входные данные. В каждом документе MapReduce должен быть определен хотя бы один вход. В документе разрешено несколько определений ввода, но каждое определение ввода может указывать только один из этих типов доступа: файл, ссылка на файл gpfdist, таблица в базе данных, команда SQL или команда операционной системы. См. gpfdist для получения информации об этой ссылке. |
| 6.1.1 | NAME | Имя для этого входа. Имена должны быть уникальными по отношению к именам других объектов в этом задании MapReduce (например, функция map, task, функция reduce и имена output). Кроме того, имена не могут конфликтовать с существующими объектами в базе данных (такими как таблицы, функции или представления). |
| 6.1.2 | FILE | Последовательность из одного или нескольких входных файлов в формате: seghostname:/path/to/filename. Вы должны быть суперпользователем RT.Warehouse, чтобы запускать задания MapReduce с вводом FILE. Файл должен находиться на хосте сегмента RT.Warehouse. |
| 6.1.3 | GPFDIST | Последовательность, идентифицирующая один или несколько запущенных файловых серверов gpfdist в формате hostname[:port]/file\_pattern. Чтобы запускать задания MapReduce с вводом GPFDIST, вы должны быть суперпользователем RT.Warehouse. |
| 6.1.4 | TABLE | Имя существующей таблицы в базе данных. |
| 6.1.5 | QUERY | SQL-команда SELECT для запуска в базе данных. |
| 6.1.6 | EXEC | Команда операционной системы для запуска на хостах сегмента RT.Warehouse. По умолчанию команда запускается всеми инстансами сегмента в системе. Например, если у вас есть четыре инстанса сегмента на хост сегмента, команда будет запущена четыре раза на каждом хосте. Чтобы запускать задания MapReduce с входными данными EXEC, вы должны быть суперпользователем RT.Warehouse. |
| 6.1.7 | COLUMNS | Опционально. Столбцы указываются как: column\_name [data\_type]. Если не указан, по умолчанию используется value text. Символ DELIMITER разделяет два поля (столбца) значений данных. Строка определяется символом перевода строки (0x0a). |
| 6.1.8 | FORMAT | Опционально. Задаёт формат данных — текст с разделителями (TEXT) или формат значений, разделённых запятыми (CSV). Если формат данных не указан, по умолчанию используется TEXT. |
| 6.1.9 | DELIMITER | Опционально для входов FILE, GPFDIST и EXEC. Задаёт один символ, разделяющий значения данных. По умолчанию в режиме TEXT используется символ табуляции, в режиме CSV — запятая. Символ-разделитель должен появляться только между любыми двумя полями значений данных. Не ставьте разделитель в начале или в конце строки. |
| 6.1.10 | ESCAPE | Опционально для входов FILE, GPFDIST и EXEC. Задаёт одиночный символ, который используется для escape-последовательностей C (например, \n, \t, \100 и т.д.) И для escape-символов данных, которые в противном случае могли бы быть приняты как разделители строк или столбцов. Обязательно выберите escape-символ, который нигде не используется в ваших фактических данных столбца. Escape-символ по умолчанию \ (обратная косая черта) для файлов в текстовом формате и " (двойная кавычка) для файлов в формате CSV, однако можно указать другой символ для представления escape-символа. Также можно отключить escape, указав значение ‘OFF’ в качестве escape-значения. Это очень полезно для таких данных, как данные веб-журнала в текстовом формате, которые содержат множество встроенных обратных косых черт, которые не предназначены для использования в качестве escape-символов. |
| 6.1.11 | NULL | Опционально для входов FILE, GPFDIST и EXEC. Задаёт строку, представляющую нулевое значение. По умолчанию используется \N в формате TEXT и пустое значение без кавычек в формате CSV. Вы можете предпочесть пустую строку даже в режиме TEXT для случаев, когда вы не хотите отличать нули от пустых строк. Любой элемент входных данных, соответствующий этой строке, будет считаться нулевым значением. |
| 6.1.12 | QUOTE | Опционально для входов FILE, GPFDIST и EXEC. Задаёт кавычки для файлов в формате CSV. По умолчанию используется двойная кавычка ("). В файлах в формате CSV поля значений данных должны быть заключены в двойные кавычки, если они содержат запятые или встроенные новые строки. Поля, содержащие символы двойных кавычек, должны быть заключены в двойные кавычки, а встроенные двойные кавычки должны быть представлены парой последовательных двойных кавычек. Важно всегда открывать и закрывать кавычки правильно для правильного анализа строк данных. |
| 6.1.13 | ERROR\_LIMIT | Если во входных строках есть ошибки формата, они будут отброшены при условии, что предельное количество ошибок не будет достигнуто ни на одном инстансе сегмента RT.Warehouse во время обработки ввода. Если предел ошибок не достигнут, все исправные строки будут обработаны, а все строки с ошибками будут отброшены. |
| 6.1.14 | ENCODING | Кодировка набора символов, используемая для данных. Укажите строковую константу (например, 'SQL\_ASCII'), целое число кодировки или DEFAULT, чтобы использовать клиентскую кодировку по умолчанию. |
| 6.2 | OUTPUT | Опционально. Определяет, куда выводить форматированные данные этого задания MapReduce. Если вывод не определён, по умолчанию используется STDOUT (стандартный вывод клиента). Вы можете отправить вывод в файл на клиентском хосте или в существующую таблицу в базе данных. |
| 6.2.1 | NAME | Имя для этого выхода. Имя вывода по умолчанию — STDOUT. Имена должны быть уникальными по отношению к именам других объектов в этом задании MapReduce (например, функция map, task, функция reduce и имена output). Кроме того, имена не могут конфликтовать с существующими объектами в базе данных (такими как таблицы, функции или представления). |
| 6.2.2 | FILE | Задаёт расположение файла на клиентском компьютере MapReduce для вывода данных в формате: /path/to/filename. |
| 6.2.3 | TABLE | Задаёт имя таблицы в базе данных для вывода данных. Если эта таблица не существует до запуска задания MapReduce, она будет создана с использованием политики распределения, указанной с помощью KEYS. |
| 6.2.4 | KEYS | Опционально для вывода TABLE. Задаёт столбцы, которые будут использоваться в качестве ключа распределения RT.Warehouse. Если задача EXECUTE содержит определение REDUCE, то ключи REDUCE будут использоваться в качестве ключа распределения таблицы по умолчанию. В противном случае первый столбец таблицы будет использоваться как ключ распределения. |
| 6.2.5 | MODE | Опционально для вывода TABLE. Если не указан, по умолчанию таблица создаётся, если она ещё не существует, то выводится ошибка, если она существует. Объявление APPEND добавляет выходные данные в существующую таблицу (при условии, что схема таблицы соответствует выходному формату) без удаления каких-либо существующих данных. Объявление REPLACE удалит таблицу, если она существует, а затем создаст её заново. И APPEND, и REPLACE создадут новую таблицу, если таковая не существует. |
| 6.3 | MAP | Обязательный. Каждая функция MAP принимает данные, структурированные парами (key, value), обрабатывает каждую пару и генерирует ноль или более выходных пар (key, value). Затем инфраструктура RT.Warehouse MapReduce собирает все пары с одним и тем же ключом из всех выходных списков и группирует их вместе. Эти выходные данные затем передаются в задачу REDUCE, которая состоит из TRANSITION | CONSOLIDATE | FINALIZE функции. Существует одна предопределённая функция MAP с именем IDENTITY, которая возвращает пары (key, value) без изменений. Хотя (key, value) являются параметрами по умолчанию, вы можете указать другие прототипы по мере необходимости. |
| 6.4 | TRANSITION | CONSOLIDATE | FINALIZE | TRANSITION, CONSOLIDATE и FINALIZE являются составными частями REDUCE. Функция TRANSITION требуется. Функции CONSOLIDATE и FINALIZE опциональны. По умолчанию все принимают state в качестве первого из своих входных PARAMETERS, но также могут быть определены другие прототипы.  Функция TRANSITION выполняет итерацию по каждому значению данного ключа и накапливает значения в переменной state. Когда функция TRANSITION вызывается для первого значения ключа, state устанавливается на значение, указанное в INITIALIZE задания REDUCE (или значение состояния по умолчанию для типа данных). Переход принимает на вход два аргумента: текущее состояние редукции ключа и следующее значение, которое затем создаёт новое state.  Если указана функция CONSOLIDATE, обработка TRANSITION выполняется на уровне сегмента перед перераспределением ключей по интерконнекту RT.Warehouse для окончательного агрегирования (двухфазное агрегирование). Перераспределяется только результирующее значение state для данного ключа, что приводит к снижению трафика интерконнекта и большему параллелизму. CONSOLIDATE обрабатывается как TRANSITION, за исключением того, что вместо (state + value) => state это (state + state) => state.  Если функция указана FINALIZE, она принимает окончательное state, созданное CONSOLIDATE (если есть) или TRANSITION и выполняет окончательную обработку перед выдачей окончательного результата. Функции TRANSITION и CONSOLIDATE не могут возвращать набор значений. Если вам нужно задание REDUCE для возврата набора, то FINALIZE необходимо для преобразования конечного состояния в набор выходных значений. |
| 6.4.1 | NAME | Обязательный. Имя функции. Имена должны быть уникальными по отношению к именам других объектов в этом задании MapReduce (например, имена функций, задач, ввода и вывода). Вы также можете указать имя функции, встроенной в RT.Warehouse. При использовании встроенной функции не указывайте LANGUAGE или FUNCTION. |
| 6.4.2 | FUNCTION | Опционально. Задаёт полное тело функции на указанном LANGUAGE. Если FUNCTION не указано, то используется встроенная функция базы данных, соответствующая NAME. |
| 6.4.3 | LANGUAGE | Требуется при использовании FUNCTION. Задаёт язык реализации, используемый для интерпретации функции. В этом выпуске есть языковая поддержка perl, python и C. При вызове встроенной функции базы данных не следует указывать LANGUAGE. |
| 6.4.4 | LIBRARY | Требуется, если LANGUAGE имеет значение C (не разрешено для других функций языка). Чтобы использовать этот атрибут, VERSION должна быть 1.0.0.2. Указанный файл библиотеки должен быть установлен до запуска задания MapReduce, и он должен существовать в одном месте файловой системы на всех хостах RT.Warehouse (мастере и сегментах). |
| 6.4.5 | PARAMETERS | Опционально. Входные параметры функции. Тип по умолчанию — text.   * MAP по умолчанию — key text, value text; * TRANSITION по умолчанию — state text, value text; * CONSOLIDATE по умолчанию — state1 text, state2 text (должен иметь ровно два входных параметра одного типа данных); * FINALIZE по умолчанию — state text (только один параметр). |
| 6.4.6 | RETURNS | Опционально. Тип возврата по умолчанию — text.   * MAP по умолчанию — key text, value text; * TRANSITION по умолчанию — state text (только одно возвращаемое значение); * CONSOLIDATE по умолчанию — state text (только одно возвращаемое значение); * FINALIZE по умолчанию — value text. |
| 6.4.7 | OPTIMIZE | Опциональные параметры оптимизации для функции:   * STRICT — на функцию не влияют значения NULL; * IMMUTABLE — функция всегда будет возвращать одно и то же значение для заданного входа. |
| 6.4.8 | MODE | Опционально. Задаёт количество строк, возвращаемых функцией:   * MULTI — возвращает 0 или более строк на входную запись. Возвращаемое значение функции должно быть массивом возвращаемых строк, или функция должна быть написана как итератор с использованием yield в Python или return\_next в Perl. MULTI — это режим по умолчанию для функций MAP и FINALIZE. * SINGLE — возвращает ровно одну строку для каждой входной записи. SINGLE — единственный режим, поддерживаемый функциями TRANSITION и CONSOLIDATE. При использовании с функциями MAP и FINALIZE режим SINGLE может обеспечить небольшое улучшение производительности. |
| 6.5 | REDUCE | Обязательный. Определение REDUCE называет функции TRANSITION | CONSOLIDATE | FINALIZE, которые включают сокращение пар (key, value) до окончательного набора результатов. Есть также несколько предопределённых заданий REDUCE, которые вы можете выполнить, и все они работают над столбцом с именем value:   * IDENTITY — возвращает пары (key, value) без изменений; * SUM — вычисляет сумму числовых данных; * AVG — вычисляет среднее числовых данных; * COUNT — подсчитывает количество входных данных; * MIN — вычисляет минимальное значение числовых данных; * MAX — вычисляет максимальное значение числовых данных. |
| 6.5.1 | NAME | Обязательный. Имя данного задания REDUCE. Имена должны быть уникальными по отношению к именам других объектов в этом задании MapReduce (имена функций, задач, входов и выходов). Кроме того, имена не могут конфликтовать с существующими объектами в базе данных (такими как таблицы, функции или представления). |
| 6.5.2 | TRANSITION | Обязательный. Имя функции TRANSITION. |
| 6.5.3 | CONSOLIDATE | Опционально. Имя функции CONSOLIDATE. |
| 6.5.4 | FINALIZE | Опционально. Имя функции FINALIZE. |
| 6.5.5 | INITIALIZE | Опционально для типов данных text и float. Обязателен для всех остальных типов данных. Значение по умолчанию для text — ''. Значение по умолчанию для float — 0.0. Устанавливает значение начального state функции TRANSITION. |
| 6.5.6 | KEYS | Опционально. По умолчанию [key, \*]. При использовании сокращения по нескольким столбцам может потребоваться указать, какие столбцы являются ключами, а какие — столбцами значений. По умолчанию все входные столбцы, которые не передаются в функцию TRANSITION, являются ключевыми столбцами, а столбец с именем key всегда является ключевым столбцом, даже если он передаётся в функцию TRANSITION. Специальный индикатор \* указывает на все столбцы, не переданные в функцию TRANSITION. Если этого индикатора нет в списке ключей, то все несовпадающие столбцы отбрасываются. |
| 6.6 | TASK | Опционально. TASK определяет полный сквозной этап INPUT/MAP/REDUCE в конвейере заданий RT.Warehouse MapReduce. Он похож на EXECUTE, за исключением того, что не выполняется сразу. Объект задачи может называться INPUT для дальнейших этапов обработки. |
| 6.6.1 | NAME | Обязательный. Название этой задачи. Имена должны быть уникальными по отношению к именам других объектов в этом задании MapReduce (например, функция map, функция reduce, имена input и output). Кроме того, имена не могут конфликтовать с существующими объектами в базе данных (такими как таблицы, функции или представления). |
| 6.6.2 | SOURCE | Имя INPUT или другой TASK. |
| 6.6.3 | MAP | Опционально. Имя функции MAP. Если параметр не указан, по умолчанию используется IDENTITY. |
| 6.6.4 | REDUCE | Опционально. Имя функции REDUCE. Если параметр не указан, по умолчанию используется IDENTITY. |
| 7 | EXECUTE | Обязательный. EXECUTE определяет заключительный этап INPUT/MAP/ REDUCE в конвейере задания RT.Warehouse MapReduce. |
| 7.1 | RUN | — |
| 7.1.1 | SOURCE | Обязательный. Имя INPUT или TASK. |
| 7.1.2 | TARGET | Опционально. Имя OUTPUT. Выход по умолчанию — STDOUT. |
| 7.1.3 | MAP | Опционально. Имя функции MAP. Если параметр не указан, по умолчанию используется IDENTITY. |
| 7.1.4 | REDUCE | Опционально. Имя функции REDUCE. По умолчанию — IDENTITY. |

gpmovemirrors

Перемещает инстансы зеркального сегмента в новое место.

Синтаксис

|  |
| --- |
| gpmovemirrors -i *move\_config\_file* [-d *master\_data\_directory*]  [-l *logfile\_directory*]  [-B *parallel\_processes*] [-v]  gpmovemirrors -?  gpmovemirrors --version |

Описание

Утилита gpmovemirrors перемещает инстансы зеркальных сегментов в новые места. Вы можете переместить зеркала в новые места, чтобы оптимизировать распределение или хранение данных.

Перед перемещением сегментов утилита проверяет, являются ли они зеркалами, и что соответствующие им основные сегменты включены и находятся в режиме синхронизации или повторной синхронизации.

По умолчанию утилита запросит у вас расположение файловой системы, куда она переместит каталоги данных зеркальных сегментов.

Вы должны убедиться, что пользователь, запускающий gpmovemirrors (пользователь gpadmin), имеет права на запись в указанные местоположения каталога данных. Вы можете создать эти каталоги на хостах сегментов и chown их соответствующему пользователю перед запуском gpmovemirrors.

Параметры

Таблица — Параметры gpmovemirrors

| Параметр | Описание |
| --- | --- |
| -B *parallel\_processes* | Количество сегментов зеркала, перемещаемых параллельно. Если параметр не указан, утилита запустит до 4 параллельных процессов в зависимости от того, сколько инстансов зеркального сегмента необходимо переместить. |
| -d *master\_data\_directory* | Каталог основных данных. Если параметр не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -i *move\_config\_file* | Файл конфигурации, содержащий информацию о том, какие зеркальные сегменты нужно переместить и куда их переместить. Для каждого основного сегмента в системе должен быть указан один зеркальный сегмент. Каждая строка внутри файла конфигурации имеет следующий формат (согласно атрибутам в таблице каталога gp\_segment\_configuration):   |  | | --- | | *contentID*|*address*|*port*|*data\_dir* *new\_address*|*port*|*data\_dir* |   Где contentID — это идентификатор содержимого инстанса сегмента, address — имя хоста или IP-адрес хоста сегмента, port — это порт связи, а data\_dir — это каталог данных инстанса сегмента. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -v | Устанавливает подробный (verbose) вывод журнала. |
| --version | Отображает версию этой утилиты. |
| -? | Отображает интерактивную справку. |

Примеры

Переместить зеркала из существующей системы RT.Warehouse на другой набор хостов:

|  |
| --- |
| $ gpmovemirrors -i move\_config\_file |

Где move\_config\_file выглядит примерно так:

|  |
| --- |
| 1|sdw2|50001|/data2/mirror/gpseg1 sdw3|50001|/data/mirror/gpseg1  2|sdw2|50001|/data2/mirror/gpseg2 sdw4|50001|/data/mirror/gpseg2  3|sdw3|50001|/data2/mirror/gpseg3 sdw1|50001|/data/mirror/gpseg3 |

gpperfmon\_install

Устанавливает базу данных gpperfmon и дополнительно включает агенты сбора данных.

Синтаксис

|  |
| --- |
| gpperfmon\_install --port *gpdb\_port*  [--enable --password *gpmon\_password* [--pgpass *path\_to\_file*]]  [--verbose]  gpperfmon\_install --help | -h | -? |

Описание

Утилита gpperfmon\_install автоматизирует шаги, необходимые для включения агентов сбора данных. Для запуска этой утилиты вы должны быть системным пользователем RT.Warehouse (gpadmin). Параметр --port обязателен. При использовании параметра --enable также требуется параметр --password. Используйте параметр --port, чтобы указать порт инстанса мастера RT.Warehouse. При использовании параметра ‑‑enable необходимо перезапустить RT.Warehouse после завершения работы утилиты.

При запуске без параметра --enable утилита просто создаёт базу данных gpperfmon (база данных, используемая для хранения системных метрик, собранных агентами сбора данных). При запуске с параметром --enable утилита также выполняет следующие дополнительные задачи, необходимые для включения агентов сбора данных мониторинга производительности:

1. Создаёт роль суперпользователя gpmon в RT.Warehouse. Агентам сбора данных эта роль требуется для подключения к базе данных и записи своих данных. Роль суперпользователя gpmon по умолчанию использует аутентификацию с паролем с шифрованием MD5. Используйте параметр --password, чтобы установить пароль суперпользователя gpmon.

Обновляет файл $MASTER\_DATA\_DIRECTORY/pg\_hba.conf. Утилита добавляет эти строки в файл аутентификации хостов (pg\_hba.conf):

|  |
| --- |
| local gpperfmon gpmon md5  host all gpmon 127.0.0.1/28 md5  host all gpmon ::1/128 md5 |

Вторая и третья строки, записи host, предоставляют gpmon доступ ко всем базам данных RT.Warehouse.

|  |  |  |
| --- | --- | --- |
| **Примечание.**  Может потребоваться отредактировать строки в файле pg\_hba.conf после запуска утилиты gpperfmon\_install, чтобы ограничить доступ роли gpmon к базам данных или изменить метод аутентификации. После редактирования файла запустите gpstop -u, чтобы перезагрузить файл в RT.Warehouse.   1. Чтобы ограничить доступ gpmon только к базе данных gpperfmon, отредактируйте записи host в файле pg\_hba.conf. Для пользователя gpmon измените второе поле с all на gpperfmon:  |  | | --- | | local gpperfmon gpmon md5  host gpperfmon gpmon 127.0.0.1/28 md5  host gpperfmon gpmon ::1/128 md5 |   Утилита gpperfmon\_install использует метод аутентификации MD5 по умолчанию. RT.Warehouse может быть дополнительно сконфигурирована для использования хэш-алгоритма SHA-256 для вычисления хэшей паролей, сохранённых в системном каталоге. Это несовместимо с методом аутентификации подлинности MD5, который ожидает хэш-код MD5 или пароль в виде открытого текста в системном каталоге. По этой причине, если вы включили хэш-алгоритм SHA-256 в базе данных, вы должны отредактировать файл pg\_hba.conf после запуска утилиты gpperfmon\_install. Для записей host измените метод аутентификации для роли gpmon с md5 на password:   |  | | --- | | local gpperfmon gpmon md5  host all gpmon 127.0.0.1/28 password  host all gpmon ::1/128 password |   Метод аутентификации password представляет для аутентификации пароль пользователя в виде открытого текста и не должен использоваться в ненадежной сети. |

Обновляет файл паролей (.pgpass). Чтобы позволить агентам сбора данных подключаться в качестве роли gpmon без запроса пароля, у вас должен быть файл паролей, в котором есть запись для пользователя gpmon. Утилита добавляет следующую запись в ваш файл паролей (если файл не существует, утилита создаст его):

|  |
| --- |
| \*:5432:gpperfmon:gpmon:*gpmon\_password* |

Если ваш файл паролей находится не в папке по умолчанию (~/.pgpass), используйте параметр --pgpass, чтобы указать местоположение файла.

Устанавливает параметры конфигурации сервера для базы данных gpperfmon. Чтобы агенты сбора данных начали сбор данных, должны быть включены следующие параметры. Утилита устанавливает следующие параметры в файлах конфигурации postgresql.conf RT.Warehouse:

gp\_enable\_gpperfmon=on (во всех файлах postgresql.conf);

gpperfmon\_port=8888 (во всех файлах postgresql.conf);

gp\_external\_enable\_exec=on (в файле мастера postgresql.conf).

Агенты сбора данных можно настроить, задав параметры в файле конфигурации gpperfmon.conf. Подробнее см. Конфигурация агента сбора данных.

Параметры

Таблица — Параметры gpperfmon\_install

| Параметр | Описание |
| --- | --- |
| --enable | Помимо создания базы данных gpperfmon, выполняет дополнительные шаги, необходимые для включения агентов сбора данных. Если указан параметр --enable, утилита также создаст и настроит учётную запись суперпользователя gpmon и установит параметры конфигурации сервера gpperfmon в файлах postgresql.conf. |
| --password *gpmon\_password* | Обязательно, если указан --enable. Устанавливает пароль суперпользователя gpmon. Недопустимо, если не указан ‑‑enable. |
| --port *gpdb\_port* | Обязательный. Задаёт порт подключения сервера мастера RT.Warehouse. |
| --pgpass *path\_to\_file* | Опционально, если указан --enable. Если файл паролей находится не в местоположении по умолчанию ~/.pgpass, указывает местоположение файла паролей. |
| --verbose | Задаёт подробный уровень ведения журнала. |
| --help | -h | -? | Отображает интерактивную справку. |

Конфигурация агента сбора данных

В файле $MASTER\_DATA\_DIRECTORY/gpperfmon/conf/gpperfmon.conf хранятся параметры конфигурации для агентов сбора данных. Чтобы изменения конфигурации этих параметров вступили в силу, необходимо сохранить файл gpperfmon.conf, а затем перезапустить сервер RT.Warehouse (gpstop -r).

Файл gpperfmon.conf содержит следующие параметры конфигурации.

Таблица — Параметры gpperfmon.conf

| Параметр | Описание |
| --- | --- |
| log\_location | Задаёт расположение каталога для файлов журнала gpperfmon. По умолчанию это $MASTER\_DATA\_DIRECTORY/gpperfmon/logs. |
| min\_query\_time | Задаёт минимальное время выполнения запроса в секундах для сбора статистики. Все запросы, которые выполняются дольше этого значения, регистрируются в таблице query\_history. Для запросов с более коротким временем выполнения исторические данные не собираются. По умолчанию — 20 секунд.  Если вы знаете, что хотите собирать данные для всех запросов, вы можете установить для этого параметра низкое значение. Однако установка минимального времени выполнения запроса равным нулю позволяет собирать данные даже для тривиальных запросов и может создать большой объём данных, которые могут оказаться бесполезными. |
| max\_log\_size | Этот параметр не включён в gpperfmon.conf, но его можно добавить в этот файл.  Чтобы файлы журнала не увеличивались до чрезмерного размера, вы можете добавить параметр max\_log\_size в gpperfmon.conf. Значение этого параметра измеряется в байтах.  Например:   |  | | --- | | max\_log\_size = 10485760 |   С данной настройкой файлы журнала накапливаются до 10 МБ, прежде чем система перейдёт к новому файлу журнала. |
| partition\_age | Количество месяцев, в течение которых данные статистики gpperfmon будут храниться. Значение по умолчанию — 0, что означает, что разделы никогда не удаляются автоматически. |
| quantum | Задаёт время в секундах между обновлениями от агентов сбора данных на всех сегментах. Допустимые значения: 10, 15, 20, 30 и 60. По умолчанию — 15 секунд.  Если вы предпочитаете менее детальное представление о производительности или хотите собирать и анализировать минимальные объёмы данных для системных метрик, выберите более высокий уровень. Чтобы собирать данные чаще, выберите меньшее значение. |
| harvest\_interval | Время в секундах между сбором данных. Сбор данных перемещает последние данные из внешних (\_tail) таблиц gpperfmon в соответствующие файлы истории. По умолчанию — 120. Минимальное значение — 30. |
| ignore\_qexec\_packet | Не рекомендуется. Если установлено значение true, агенты сбора данных не собирают данные о производительности в базы данных gpperfmon таблицах queries\_\*: rows\_out, cpu\_elapsed, cpu\_currpct, skew\_cpu и skew\_rows. Значение по умолчанию, true, уменьшает объём памяти, потребляемой процессом gpmmon. Установите для этого параметра значение false, если вам требуются эти дополнительные данные о производительности. |
| smdw\_aliases | Этот параметр позволяет указать дополнительные имена хостов для резервного мастера. Например, если у резервного мастера есть две сетевые карты, вы можете ввести:   |  | | --- | | smdw\_aliases=smdw-1,smdw-2 | |

Примечания

База данных gpperfmon требует роли gpmon. После создания базы данных gpperfmon и роли gpmon вы можете изменить пароль для роли gpmon:

1. Войдите в RT.Warehouse как суперпользователь и измените пароль gpmon с помощью команды ALTER ROLE.

|  |
| --- |
| # ALTER ROLE gpmon WITH PASSWORD '*new*\_*password*' ; |

Обновите пароль в файле .pgpass. Расположение файла по умолчанию — домашний каталог gpadmin (~/.pgpass). Файл .pgpass содержит строку с паролем gpmon.

|  |
| --- |
| \*:5432:gpperfmon:gpmon:*new\_password* |

Система мониторинга gpperfmon требует инициализации после запуска. Информация о мониторинге появляется через несколько минут, а не сразу после установки и запуска системы gpperfmon.

Примеры

Создать только базу данных gpperfmon:

|  |
| --- |
| $ su - gpadmin  $ gpperfmon\_install --port 5432 |

Создать базу данных gpperfmon, создать суперпользователя gpmon и включить агенты сбора данных:

|  |
| --- |
| $ su - gpadmin  $ gpperfmon\_install --enable --password changeme --port 5432  $ gpstop -r |

gppkg

Устанавливает расширения RT.Warehouse в формате .gppkg, такие как PL/Java, PL/R и MADlib, а также их зависимости во всем кластере.

Синтаксис

|  |
| --- |
| gppkg [-i *package* | -u *package* | -r *name-version* | -c]  [-d *master\_data\_directory*] [-a] [-v]  gppkg --migrate *GPHOME\_old GPHOME\_new* [-a] [-v]  gppkg [-q | --query] *query\_option*  gppkg -? | --help | -h  gppkg --version |

Описание

Утилита RT.Warehouse Package Manager (gppkg) устанавливает расширения RT.Warehouse вместе с любыми зависимостями на всех хостах в кластере. Она также автоматически установит расширения на новые хосты в случае расширения системы и восстановления сегмента.

|  |
| --- |
| **Примечание.**  После серьёзного обновления RT.Warehouse необходимо снова загрузить и установить все расширения gppkg. |

Параметры

Таблица — Параметры gppkg

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -c | --clean | Согласовывает состояние пакета кластера с состоянием хоста мастера. Запуск этого параметра после неудачной или частичной установки/удаления гарантирует, что состояние установки пакета будет согласованным во всем кластере. |
| -d *master\_data\_directory* | Каталог данных мастера. Если параметр не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -i *package* | --install=*package* | Устанавливает данный пакет. Это включает в себя все шаги до и после установки и установку любых зависимостей. |
| --migrate *GPHOME\_old GPHOME\_new* | Переносит пакеты из отдельного $GPHOME. Переносит пакеты из одной версии RT.Warehouse в другую.  Например:   |  | | --- | | gppkg --migrate /usr/local/greenplum-db-<old-version> /usr/local/greenplum-db-<new-version> |   При переносе пакетов должны быть выполнены следующие требования:   * По крайней мере, должен быть запущен инстанс мастера целевой RT.Warehouse (инстанс, установленный в *GPHOME\_new*). Перед запуском команды gppkg запустите мастер RT.Warehouse с помощью команды gpstart -m. * Запустите утилиту gppkg из установки *GPHOME\_new* (каталог установки для миграции). |
| -q | --query *query\_option* | Предоставляет информацию, указанную в query\_option, об установленных пакетах. За один раз можно указать только один *query\_option*. Ниже перечислены возможные значения параметров для query\_option, где <package\_file> — имя пакета:   * <package\_file> — установлен ли указанный пакет; * --info <package\_file> — имя, версия и другая информация об указанном пакете; * --list <package\_file> — содержимое файла указанного пакета; * --all — список всех установленных пакетов. |
| -r *name-version* | --remove=*name-version* | Удаляет указанный пакет. |
| -u *package* | --update=*package* | Обновляет данный пакет.  **Внимание**. Процесс обновления пакета включает удаление всех предыдущих версий системных объектов, связанных с пакетом. Например, удаляются предыдущие версии общих библиотек. После процесса обновления функция базы данных завершится ошибкой при её вызове, если функция ссылается на файл пакета, который был удалён. |
| --version | Отображает версию этой утилиты. |
| -v | --verbose | Задаёт подробный уровень ведения журнала. |
| -? | -h | --help | Отображает интерактивную справку. |

gprecoverseg

Восстанавливает основной или зеркальный инстанс сегмента, который был помечен как отключённый (если зеркалирование включено).

Синтаксис

|  |
| --- |
| gprecoverseg [-p *new\_recover\_host*[,...]] | -i *recover\_config\_file*] [-d *master\_data\_directory*]  [-B *parallel\_processes*] [-F [-s]] [-a] [-q]  [--no-progress] [-l *logfile\_directory*]  gprecoverseg -r  gprecoverseg -o *output\_recover\_config\_file*  [-p *new\_recover\_host*[,...]]  gprecoverseg -? | --help  gprecoverseg --version |

Описание

В системе с включёнными зеркалами утилита gprecoverseg реактивирует отказавший инстанс сегмента и идентифицирует изменённые файлы базы данных, требующие ресинхронизации. После того, как gprecoverseg завершит этот процесс, система перейдёт в режим ресинхронизации, пока восстановленный сегмент не будет обновлён. Система активна и полностью работоспособна во время ресинхронизации.

Во время инкрементного восстановления (параметр -F не ​​указан), если gprecoverseg обнаруживает инстанс сегмента с отключённым зеркалированием в системе с включёнными зеркалами, утилита сообщает, что зеркалирование отключено для сегмента, и не пытается восстановить этот инстанс сегмента, и продолжает процесс восстановления.

Инстанс сегмента может выйти из строя по нескольким причинам, например, из-за сбоя хоста, сбоя сети или сбоя диска. Когда инстанс сегмента выходит из строя, его статус отмечается как отключённый в системном каталоге RT.Warehouse, а его зеркало активируется в режиме отслеживания изменений. Чтобы снова запустить отказавший инстанс сегмента, необходимо сначала исправить проблему, из-за которой произошёл сбой, а затем восстановить инстанс сегмента в RT.Warehouse с помощью gprecoverseg.

Для восстановления сегмента с помощью gprecoverseg необходимо наличие активного зеркала для восстановления. Для систем, в которых не включено зеркалирование, или в случае двойного сбоя (одновременное отключение основного и зеркального сегмента), необходимо вручную предпринять шаги для восстановления отказавших инстансов сегмента, а затем выполнить перезагрузку системы, чтобы сегменты снова были работоспособны Например, эта команда перезапускает систему.

|  |
| --- |
| gpstop -r |

По умолчанию отказавший сегмент восстанавливается на месте, что означает, что система возвращает сегмент в работоспособный режим на том же хосте и в том же месте каталога данных, на котором он был изначально настроен. В этом случае используйте следующий формат файла конфигурации восстановления (с помощью -i).

|  |
| --- |
| <failed\_host\_address>|<port>|<data\_directory> |

В некоторых случаях это может быть невозможно (например, если хост был физически повреждён и не может быть восстановлен). В этой ситуации gprecoverseg позволяет вам восстанавливать отказавшие сегменты на новый хост (используя -p), в альтернативном местоположении каталога данных на ваших оставшихся живых хостах сегментов (используя -s) или предоставляя файл конфигурации восстановления (используя -i) в следующем формате. Слово SPACE указывает расположение необходимого места. Не добавляйте дополнительных пробелов.

|  |
| --- |
| <failed\_host\_address>|<port>|<data\_directory>SPACE  <recovery\_host\_address>|<port>|<data\_directory> |

Таблица системного каталога gp\_segment\_configuration может помочь вам определить текущую конфигурацию сегмента, чтобы вы могли спланировать конфигурацию восстановления зеркала. Например, выполните следующий запрос:

|  |
| --- |
| =# SELECT dbid, content, address, port, datadir  FROM gp\_segment\_configuration  ORDER BY dbid; |

На новом восстановленном хосте сегмента должен быть предварительно установлено программное обеспечение RT.Warehouse, а также он должен быть настроен точно так же, как существующие хосты сегмента. Запасной каталог данных должен существовать на всех хостах текущего настроенного сегмента и иметь достаточно места на диске для размещения отказавших сегментов.

В процессе восстановления сегмент снова помечается как активный в системном каталоге RT.Warehouse, а затем запускается процесс ресинхронизации, чтобы актуализировать состояние транзакций сегмента с учётом последних изменений. Система активна и доступна во время ресинхронизации. Чтобы проверить статус выполнения процесса ресинхронизации:

|  |
| --- |
| gpstate -m |

Параметры

Таблица — Параметры gprecoverseg

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -B *parallel\_processes* | Количество сегментов для параллельного восстановления. Если не указано иное, утилита запустит до 16 параллельных процессов в зависимости от того, сколько инстансов сегментов необходимо восстановить. |
| -d *master\_data\_directory* | Опционально. Каталог данных хоста мастера. Если параметр не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -F | Опционально. Выполните полную копию (full recovery) инстанса активного сегмента, чтобы восстановить отказавший сегмент. По умолчанию копируются только те инкрементные изменения, которые произошли во время остановки сегмента.  Для больших баз данных полное восстановление может занять много времени, поэтому gprecoverseg отображает текущую оценку хода завершения копирования для каждого сегмента. Ход выполнения для каждого сегмента обновляется раз в секунду с использованием управляющих кодов ANSI для обновления строки для каждого сегмента на месте. Если вы перенаправляете вывод gprecoverseg в файл или если escape-коды ANSI не работают правильно на вашем терминале, вы можете включить параметр -s в командную строку, чтобы Не включать escape-коды ANSI. Это выводит новую строку один раз в секунду для каждого сегмента. Включите параметр --no-progress, чтобы полностью отключить отчёты о ходе выполнения.  **Внимание**. Полное восстановление удаляет каталог данных инстанса сегмента, вышедшего из строя, перед копированием данных из активного (текущего основного) инстанса сегмента. Перед выполнением полного восстановления убедитесь, что сбой сегмента не привёл к повреждению данных и что все проблемы с диском сегмента хоста устранены.  Кроме того, для полного восстановления утилита не восстанавливает пользовательские файлы, которые хранятся в каталоге данных инстанса сегмента, даже если пользовательские файлы также находятся в активном инстансе сегмента. Вы должны восстановить пользовательские файлы вручную. Например, при использовании протокола gpfdists (gpfdist с SSL-шифрованием) для управления внешними данными, файлы сертификатов клиента требуются в каталоге инстанса сегмента $PGDATA/gpfdists. Эти файлы не восстанавливаются. |
| -i *recover\_config\_file* | Задаёт имя файла с подробностями о неисправных сегментах, которые необходимо восстановить. Каждая строка в файле имеет следующий формат. Слово SPACE указывает расположение необходимого пробела. Не добавляйте дополнительные пробелы.   |  | | --- | | <failed\_host\_address>|<port>|<data\_directory>SPACE  <recovery\_host\_address>|<port>|<data\_directory> |   **Комментарии**  Строки, начинающиеся с символа #, рассматриваются как комментарии и игнорируются.  **Сегменты для восстановления**  Каждая строка после первой указывает сегмент для восстановления. Эта строка может иметь один из двух форматов. В случае восстановления на месте введите в строке одну группу полей, разделённых двоеточиями. Например:   |  | | --- | | failedAddress|failedPort|failedDataDirectory |   Для восстановления в новое место введите две группы полей, разделённых пробелом в строке. Требуемый пробел обозначается SPACE. Не добавляйте дополнительных пробелов.   |  | | --- | | failedAddress|failedPort|failedDataDirectorySPACEnewAddress|  newPort|newDataDirectory |   **Примеры**  Восстановление одного зеркала на прежнее место.   |  | | --- | | sdw1-1|50001|/data1/mirror/gpseg16 |   Восстановление одного зеркала на новый хост.   |  | | --- | | sdw1-1|50001|/data1/mirror/gpseg16SPACEsdw4-1|50001|/data1/recover1/gpseg16 |   **Получение образца файла**  Вы можете использовать параметр -o для вывода образца файла конфигурации восстановления, который будет использоваться в качестве отправной точки. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -o *output\_recover\_config\_file* | Задаёт имя файла и расположение для вывода образца файла конфигурации восстановления. В выходном файле перечислены не рабочие в настоящее время сегменты и их расположение для восстановления по умолчанию в формате, который требуется параметром -i. Используйте вместе с параметром -p, чтобы вывести образец файла для восстановления на другом хосте. При необходимости этот файл можно отредактировать, чтобы указать альтернативные места для восстановления. |
| -p *new\_recover\_host*[,...] | Задаёт запасной хост вне текущего настроенного массива RT.Warehouse, на котором нужно восстановить вышедшие из строя сегменты. В случае нескольких отказавших хостов сегмента вы можете указать список, разделённый запятыми. На запасном хосте должно быть установлено и настроено программное обеспечение RT.Warehouse, а также должна быть установлена такая же конфигурация оборудования и ОС, как и у хостов текущего сегмента (та же версия ОС, локали, учётная запись пользователя gpadmin, созданные каталоги данных, обмен ключами ssh, количество сетевых интерфейсов, соглашение об именах сетевых интерфейсов и т.д.). |
| -q | Работать в тихом режиме. Вывод команды не отображается на экране, но всё равно записывается в файл журнала. |
| -r | После восстановления сегмента инстансы сегмента не могут быть возвращены в предпочтительную роль, которую они получили во время инициализации системы. Это может оставить систему в потенциально несбалансированном состоянии, поскольку некоторые хосты сегментов могут иметь больше активных сегментов, чем оптимально для максимальной производительности системы. Этот параметр перебалансирует основные и зеркальные сегменты, возвращая им их предпочтительные роли. Все сегменты должны быть действительными и синхронизированными перед запуском gprecoverseg -r. Если есть какие-либо незавершённые запросы, они будут отменены и откачены назад. |
| -s | Показывать прогресс pg\_basebackup последовательно, а не in-place. Полезно при записи в файл или если tty не поддерживает escape-последовательности. По умолчанию прогресс отображается in-place. |
| --no-progress | Не отображать отчёты о ходе выполнения от утилиты pg\_basebackup. По умолчанию отображается прогресс базового резервного копирования. |
| -v | Устанавливает подробный (verbose) вывод журнала. |
| --version | Отображает версию этой утилиты. |
| -? (help) | Отображает интерактивную справку. |

Примеры

Восстановить все отказавшие инстансы сегмента на месте:

|  |
| --- |
| $ gprecoverseg |

Перебалансировать свою систему RT.Warehouse после восстановления, вернув все сегменты к их предпочтительной роли. Сначала убедиться, что все сегменты включены и синхронизированы.

|  |
| --- |
| $ gpstate -m  $ gprecoverseg -r |

Восстановить все вышедшие из строя инстансы сегмента на вновь настроенный хост резервного сегмента:

|  |
| --- |
| $ gprecoverseg -i recover\_config\_file |

Вывести файл конфигурации восстановления по умолчанию:

|  |
| --- |
| $ gprecoverseg -o /home/gpadmin/recover\_config\_file |

gpreload

Перезагружает данные таблицы RT.Warehouse, сортируя данные по указанным столбцам.

Синтаксис

|  |
| --- |
| gpreload -d *database* [-p *port*] {-t | --table-file} *path\_to\_file* [-a]  gpreload -h  gpreload --version |

Описание

Утилита gpreload перезагружает данные таблицы с отсортированными данными столбцов. Для таблиц, которые были созданы с параметром хранения таблиц appendoptimized=TRUE и включённым сжатием, перезагрузка данных с отсортированными данными может улучшить сжатие таблицы. Вы указываете список таблиц, которые нужно перезагрузить, и столбцы таблицы, которые нужно отсортировать, в текстовом файле.

Сжатие улучшается за счёт сортировки данных, когда данные в столбце имеют относительно небольшое количество различных значений по сравнению с общим количеством строк.

Для перезагружаемой таблицы порядок сортировки столбцов может повлиять на сжатие. Столбцы с наименьшим количеством различных значений должны быть перечислены первыми. Например, если указать штат, а затем город, сжатие будет лучше, чем если указать город и штат.

|  |
| --- |
| public.cust\_table: state, city  public.cust\_table: city, state |

Для получения информации о формате файла, используемого с gpreload, см. параметр --table-file.

Примечания

Чтобы повысить производительность перезагрузки, перед перезагрузкой данных необходимо удалить индексы перезагружаемых таблиц.

Выполнение команды ANALYZE после перезагрузки данных таблицы может потребовать производительности из-за изменения в распределении данных перезагруженных данных.

Для каждой таблицы утилита копирует данные таблицы во временную таблицу, обрезает существующие данные таблицы и вставляет данные из временной таблицы в таблицу в указанном порядке сортировки. Каждая перезагрузка таблицы выполняется за одну транзакцию.

Для партиционированной таблицы вы можете перезагрузить данные дочерней leaf-партиции. Однако данные вставляются из корневой партиции таблицы, которая получает блокировку ROW EXCLUSIVE для всей таблицы.

Параметры

Таблица — Параметры gpreload

| Параметр | Описание |
| --- | --- |
| -a | Опционально. Если параметр указан, утилита gpreload не запрашивает подтверждения у пользователя. |
| -d *database* | База данных, содержащая таблицы для перезагрузки. Утилита gpreload подключается к базе данных как пользователь, запускающий утилиту. |
| -p *port* | Порт мастера RT.Warehouse. Если параметр не указан, используется значение переменной среды PGPORT. Если значение недоступно, возвращается ошибка. |
| {-t | --table-file } *path\_to\_file* | Местоположение и имя файла, содержащего список имён таблиц с указанием схемы для перезагрузки и имена столбцов для изменения порядка из RT.Warehouse. Поддерживаются только определённые пользовательские таблицы. Представления или таблицы системного каталога не поддерживаются.  Если для таблицы, указанной в файле, определены индексы, gpreload предложит продолжить.  В каждой строке указывается имя таблицы и список столбцов для сортировки. Данный формат требуется для каждой строки в файле:   |  | | --- | | schema.table\_name: column [desc] [, column2 [desc] ... ] |   За именем таблицы следует двоеточие (:), а затем хотя бы одно имя столбца. Если вы указываете более одного столбца, разделяйте имена столбцов запятыми. Столбцы сортируются по возрастанию. Укажите ключевое слово desc после имени столбца, чтобы отсортировать столбец в порядке убывания. Подстановочные символы не поддерживаются. Если в файле есть ошибки, gpreload сообщает о первой ошибке и завершает работу. Никакие данные не перезагружаются. В следующем примере перезагружаются три таблицы:   |  | | --- | | public.clients: region, state, rep\_id desc  public.merchants: region, state  test.lineitem: group, assy, whse |   В первой таблице public.clients данные в столбце rep\_id отсортированы по убыванию. Данные в других столбцах отсортированы по возрастанию. |
| --version | Отображает версию этой утилиты. |
| -? | Отображает интерактивную справку. |

Примеры

В этом примере команда перезагружает таблицы в базе данных mytest, перечисленные в файле data-tables.txt.

|  |
| --- |
| gpreload -d mytest --table-file data-tables.txt |

gprestore

Восстановите резервную копию RT.Warehouse, созданную с помощью утилиты gpbackup. По умолчанию gprestore использует резервные копии файлов метаданных и файлов DDL, расположенных в каталоге данных хоста мастера RT.Warehouse, с табличными данными, хранящимися локально на хостах сегментов в файлах данных CSV.

Синтаксис

|  |
| --- |
| gprestore --timestamp YYYYMMDDHHMMSS  [--backup-dir directory]  [--create-db]  [--debug]  [--exclude-schema schema\_name]  [--exclude-table schema.table]  [--exclude-table-file file\_name]  [--include-schema schema\_name]  [--include-table schema.table]  [--include-table-file file\_name]  [--data-only | --metadata-only]  [--jobs int]  [--on-error-continue]  [--plugin-config config\_file\_location]  [--quiet]  [--redirect-db database\_name]  [--verbose]  [--version]  [--with-globals]  [--with-stats]  gprestore --help |

Описание

Чтобы использовать gprestore для восстановления из набора резервных копий, необходимо включить параметр --timestamp, чтобы указать точное значение метки времени (YYYYMMDDHHMMSS) восстанавливаемого набора резервных копий. Если вы указали настраиваемый --backup-dir для объединения файлов резервных копий, включите тот же параметр --backup-dir с gprestore, чтобы найти файлы резервных копий.

Если указанная вами резервная копия является инкрементной, вам понадобится полный набор файлов резервных копий (полная резервная копия и все необходимые инкрементные резервные копии). gprestore гарантирует, что полный набор резервных копий доступен перед попыткой восстановления резервной копии.

|  |
| --- |
| **Внимание.**  Для наборов инкрементных резервных копий резервные копии должны быть на одном устройстве. Например, весь набор резервных копий должен находиться в системе Data Domain. |

При восстановлении из набора резервных копий gprestore выполняет восстановление в базу данных с тем же именем, что и имя, указанное при создании набора резервных копий. Если целевая база данных существует и восстанавливаемая таблица существует в базе данных, операция восстановления не выполняется. Включите параметр --create-db, если целевая база данных не существует в кластере. При желании вы можете восстановить набор резервных копий в другую базу данных с помощью параметра --redirect-db.

При восстановлении набора резервных копий, который содержит данные из некоторых leaf-партиций партиционированных таблиц, партиционированная таблица восстанавливается вместе с данными для leaf-партиций. Например, вы создаёте резервную копию с параметром gpbackup --include-table-file, и в текстовом файле перечислены некоторые leaf-партиции партиционированной таблицы. При восстановлении резервной копии создаётся партиционированная таблица и восстанавливаются данные только для leaf-партиций, перечисленных в файле.

Системные объекты RT.Warehouse автоматически включаются в набор резервных копий gpbackup, но эти объекты восстанавливаются только в том случае, если вы включили параметр --with-globals в gprestore. Точно так же, если вы создали резервную копию статистики плана запроса с помощью параметра --with-stats, вы можете восстановить эту статистику, указав --with-stats для gprestore. По умолчанию восстанавливаются только объекты базы данных из резервного набора.

Производительность операций восстановления можно повысить, создав несколько параллельных соединений для восстановления табличных данных и метаданных. По умолчанию gprestore использует 1 соединение, но вы можете увеличить это число с помощью параметра --jobs для больших операций восстановления.

Когда операция восстановления завершается, gprestore возвращает код состояния.

gprestore может отправлять уведомления о статусе по электронной почте после завершения операции резервного копирования. Вы указываете, когда утилита отправляет почту, и получателей электронной почты в файле конфигурации.

|  |
| --- |
| **Примечание.**  Эта утилита использует соединения защищенной оболочки (SSH) между системами для выполнения своих задач. В больших развёртываниях RT.Warehouse, облачных развёртываниях или развёртываниях с большим количеством сегментов на хост эта утилита может превышать максимальный порог хоста для неаутентифицированных соединений. Рассмотрите возможность обновления параметра конфигурации SSH MaxStartups, чтобы увеличить этот лимит. |

Параметры

Таблица — Параметры gprestore

| Параметр | Описание |
| --- | --- |
| --timestamp *YYYYMMDDHHMMSS* | Обязательный. Задаёт метку времени для восстанавливаемого набора резервных копий gpbackup. По умолчанию gprestore пытается найти файлы метаданных для отметки времени на хосте мастера RT.Warehouse в каталоге $MASTER\_DATA\_DIRECTORY/backups/YYYYMMDD/YYYYMMDDhhmmss/ и файлы данных CSV в каталоге хоста каждого сегмента <seg\_dir>/backups/YYYYMYMDMss/YYYYMYMDSS. |
| --backup-dir *directory* | Опциональный. Источник всех файлов резервных копий (файлов метаданных и файлов данных) из указанного каталога. Вы должны указать каталог как абсолютный путь (не относительный). Если вы не укажете этот параметр, gprestore попытается найти файлы метаданных для метки времени на хосте мастера RT.Warehouse в каталоге $MASTER\_DATA\_DIRECTORY/backups/YYYYMMDD/YYYYMMDDhhmmss/. Файлы данных CSV должны быть доступны в каталоге каждого сегмента <seg\_dir>/backups/YYYYMMDD/YYYYMMDDhhmmss/. Включите этот параметр при указании настраиваемого каталога резервного копирования с помощью gpbackup. Этот параметр нельзя комбинировать с параметром --plugin-config. |
| --create-db | Опциональный. Создаёт базу данных перед восстановлением метаданных объекта базы данных. База данных создаётся путём клонирования пустой стандартной системной базы данных template0. |
| --data-only | Опциональный. Восстанавливает данные таблиц из резервной копии, созданной с помощью утилиты gpbackup, без создания таблиц базы данных. Этот параметр предполагает, что таблицы существуют в целевой базе данных. Чтобы восстановить данные для определённого набора таблиц из набора резервных копий, вы можете указать параметр для включения таблиц или схем или исключения таблиц или схем. Укажите параметр --with-stats, чтобы восстановить статистику таблиц из резервной копии. Набор резервных копий должен содержать данные таблицы, которые необходимо восстановить. Например, резервная копия, созданная с параметром gpbackup --metadata-only, не содержит табличных данных. Чтобы восстановить только таблицы базы данных без восстановления данных таблицы, см. параметр --metadata-only. |
| --debug | Опциональный. Отображает подробные сообщения и сообщения журнала отладки во время операции восстановления. |
| --exclude-schema *schema\_name* | Опциональный. Задаёт схему базы данных, которую нужно исключить из операции восстановления. Вы можете указать этот параметр несколько раз, чтобы исключить несколько схем. Вы не можете комбинировать этот параметр с параметром --include-schema или параметром фильтрации таблиц, таким как --include-table. |
| --exclude-table *schema.table* | Опциональный. Задаёт таблицу, которую нужно исключить из операции восстановления. Таблица должна быть в формате <schema-name>.<table-name>. Если в имени таблицы или схемы используется какой-либо символ, кроме строчной буквы, числа или символа подчёркивания, то вы должны заключить это имя в двойные кавычки. Вы можете указать эту опцию несколько раз. Если таблицы нет в резервном наборе, операция восстановления завершится неудачно. Вы не можете указать leaf-партицию партиционированной таблицы. Вы не можете комбинировать этот параметр с параметром --exclude-schema или другим параметром фильтрации таблицы, например --include-table. |
| --exclude-table-file *file\_name* | Опциональный. Задаёт текстовый файл, содержащий список таблиц, которые следует исключить из операции восстановления. Каждая строка в текстовом файле должна определять одну таблицу в формате <schema-name>.<table-name>. Файл не должен содержать завершающих строк. Если в имени таблицы или схемы используется какой-либо символ, кроме строчной буквы, числа или символа подчёркивания, то вы должны заключить это имя в двойные кавычки. Если таблицы нет в резервном наборе, операция восстановления завершится ошибкой. Вы не можете указать leaf-партицию партиционированной таблицы. Вы не можете комбинировать этот параметр с параметром --exclude-schema или другим параметром фильтрации таблицы, например --include-table. |
| --include-schema *schema\_name* | Опциональный. Задаёт схему базы данных для восстановления. Вы можете указать этот параметр несколько раз, чтобы включить несколько схем. Если вы укажете этот параметр, все указанные вами схемы должны быть доступны в резервном наборе. Любые схемы, которые не включены в последующие параметры --include-schema, исключаются из операции восстановления. Если схема, которую вы указываете для включения, существует в базе данных, утилита выдаёт ошибку и продолжает операцию. Утилита не работает, если восстанавливаемая таблица существует в базе данных. Вы не можете использовать эту опцию, если объекты в наборе резервных копий зависят от нескольких схем. |
| --include-table *schema.table* | Опциональный. Задаёт таблицу для восстановления. Таблица должна быть в формате <schema-name>.<table-name>. Если в имени таблицы или схемы используется какой-либо символ, кроме строчной буквы, числа или символа подчёркивания, то вы должны заключить это имя в двойные кавычки. Вы можете указать эту опцию несколько раз. Вы не можете указать leaf-партицию партиционированной таблицы. Вы также можете указать полное имя последовательности или представления. Если вы укажете эту опцию, утилита не будет автоматически восстанавливать зависимые объекты. Вы также должны явно указать требуемые зависимые объекты. Например, если вы восстанавливаете представление, вы должны также восстановить таблицы, которые оно использует. Если вы восстанавливаете таблицу, в которой используется последовательность, вы также должны восстановить последовательность. Зависимые объекты должны существовать в резервном наборе. Этот параметр нельзя комбинировать с параметром фильтрации схемы, например --include-schema, или другим параметром фильтрации таблиц, например --exclude-table-file. |
| --include-table-file *file\_name* | Опциональный. Задаёт текстовый файл, содержащий список таблиц для восстановления. Каждая строка в текстовом файле должна определять одну таблицу в формате <schema-name>.<table-name>. Файл не должен содержать завершающих строк. Если в имени таблицы или схемы используется любой символ, кроме строчной буквы, числа или символа подчёркивания, то вы должны заключить это имя в двойные кавычки. Любые таблицы, не перечисленные в этом файле, исключаются из операции восстановления. Вы не можете указать leaf-партицию партиционированной таблицы. Вы также можете указать полное имя последовательности или представления. Если вы укажете эту опцию, утилита не будет автоматически восстанавливать зависимые объекты. Вы также должны явно указать требуемые зависимые объекты. Например, если вы восстанавливаете представление, вы также должны указать таблицы, которые оно использует. Если вы указываете таблицу, в которой используется последовательность, вы также должны указать последовательность. Зависимые объекты должны существовать в резервном наборе. Если вы используете параметр --include-table-file, gprestore не создаёт роли и не устанавливает владельца таблиц. Утилита восстанавливает индексы и правила таблиц. Триггеры также восстанавливаются, но не поддерживаются в RT.Warehouse. |
| --jobs *int* | Опциональный. Задаёт количество параллельных подключений для использования при восстановлении табличных данных и метаданных. По умолчанию gprestore использует 1 соединение. Увеличение этого числа может повысить скорость восстановления данных.  **Примечание**. Если вы использовали параметр gpbackup --single-data-file для объединения резервных копий таблиц в один файл для каждого сегмента, вы не можете установить для параметра --jobs значение выше 1 для выполнения операции параллельного восстановления. |
| --metadata-only | Опциональный. Создаёт таблицы базы данных из резервной копии, созданной с помощью утилиты gpbackup, но не восстанавливает данные таблицы. Этот параметр предполагает, что таблицы не существуют в целевой базе данных. Чтобы создать определённый набор таблиц из резервного набора, вы можете указать параметр для включения таблиц или схем или исключения таблиц или схем. Укажите параметр --with-globals, чтобы восстановить системные объекты RT.Warehouse. Набор резервных копий должен содержать DDL для таблиц, подлежащих восстановлению. Например, резервная копия, созданная с параметром gpbackup --data-only, не содержит DDL для таблиц. Чтобы восстановить данные таблицы после создания таблиц базы данных, см. параметр --data-only. |
| --on-error-continue | Опциональный. Укажите этот параметр, чтобы продолжить операцию восстановления в случае возникновения ошибки SQL при создании метаданных базы данных (таких как таблицы, роли или функции) или при восстановлении данных. Если возникает ошибка другого типа, утилита закрывается. Утилита отображает сводку ошибок и записывает информацию об ошибках в файл журнала gprestore и продолжает операцию восстановления. По умолчанию выход выполняется при первой ошибке. |
| --plugin-config *config-file\_location* | Укажите расположение файла конфигурации подключаемого модуля gpbackup, текстового файла в формате YAML. Файл содержит информацию о конфигурации для приложения подключаемого модуля, которое gprestore использует во время операции восстановления. Если вы укажете параметр --plugin-config при резервном копировании базы данных, вы должны указать эту опцию с информацией о конфигурации для соответствующего приложения подключаемого модуля при восстановлении базы данных из резервной копии. Этот параметр нельзя комбинировать с параметром --backup-dir. |
| --quiet | Опциональный. Подавить все сообщения журнала без предупреждений и ошибок. |
| --redirect-db *database\_name* | Опциональный. Восстановление в указанную database\_name вместо базы данных, для которой была создана резервная копия. |
| --verbose | Опциональный. Отображает подробные сообщения журнала во время операции восстановления. |
| --version | Опциональный. Распечатывает номер версии и завершается. |
| --with-globals | Опциональный. Восстанавливает системные объекты RT.Warehouse из набора резервных копий в дополнение к объектам базы данных. |
| --with-stats | Опциональный. Восстанавливает статистику плана запроса из резервного набора. |
| --help | Отображает интерактивную справку. |

Коды возврата

Один из этих кодов возвращается после завершения работы gprestore:

0 — восстановление выполнено без проблем;

1 — восстановление завершено с нефатальными ошибками;

2 — восстановление завершилось фатальной ошибкой.

Примеры

Создать базу данных demo и восстановить все схемы и таблицы в резервном наборе для указанной временной метки:

|  |
| --- |
| $ dropdb demo  $ gprestore --timestamp 20171103152558 --create-db |

Восстановить набор резервных копий в базе данных demo2 вместо базы данных demo, для которой была создана резервная копия:

|  |
| --- |
| $ createdb demo2  $ gprestore --timestamp 20171103152558 --redirect-db demo2 |

Восстановить глобальные метаданные RT.Warehouse и статистику плана запроса в дополнение к объектам базы данных:

|  |
| --- |
| $ gprestore --timestamp 20171103152558 --create-db --with-globals --with-stats |

Восстановить, используя файлы резервных копий, которые были созданы в каталоге /home/gpadmin/backup, создав 8 параллельных подключений:

|  |
| --- |
| $ gprestore --backup-dir /home/gpadmin/backups/ --timestamp 20171103153156 --create-db --jobs 8 |

Восстановить только схему wikipedia, включённую в набор резервных копий:

|  |
| --- |
| $ dropdb demo  $ gprestore --include-schema wikipedia --backup-dir /home/gpadmin/backups/ --timestamp 20171103153156 --create-db |

Если вы выполняете восстановление из набора инкрементных резервных копий, все необходимые файлы в наборе резервных копий должны быть доступны для gprestore. Например, следующие ключи отметок времени указывают набор добавочных резервных копий. 20170514054532 — это полная резервная копия, а остальные — инкрементные.

|  |
| --- |
| 20170514054532 (full backup)  20170714095512  20170914081205  20171114064330  20180114051246 |

Следующая команда gprestore указывает метку времени 20121114064330. Для выполнения восстановления должна быть доступна инкрементная резервная копия с метками времени 20120714095512 и 20120914081205 и полная резервная копия.

|  |
| --- |
| gprestore --timestamp 20121114064330 --redirect-db mystest --create-db |

gpscp

Копирует файлы сразу между несколькими хостами.

Синтаксис

|  |
| --- |
| gpscp { -f *hostfile\_gpssh* | -h *hostname* [-h *hostname* ...] }  [-J *character*] [-v] [[*user@*]*hostname*:]*file\_to\_copy* [...]  [[*user*@]*hostname*:]*copy\_to\_path*  gpscp -?  gpscp --version |

Описание

Утилита gpscp позволяет копировать один или несколько файлов с указанных хостов на другие указанные хосты одной командой, используя SCP (безопасное копирование, secure copy). Например, вы можете скопировать файл с хоста мастера RT.Warehouse на все хосты сегментов одновременно.

Чтобы указать хосты, участвующие в сеансе SCP, используйте параметр ‑f, чтобы указать файл, содержащий список имён хостов, или используйте параметр -h, чтобы указать имена отдельных хостов в командной строке. Требуется хотя бы одно имя хоста (-h) или файл хоста (‑f). Параметр -J позволяет вам указать один символ для замены hostname в copy from и copy to строки назначения. Если -J не указан, символ подстановки по умолчанию — знак равенства (=). Например, следующая команда скопирует .bashrc с локального хоста в /home/gpadmin на всех хостах, указанных в hostfile\_gpssh:

|  |
| --- |
| gpscp -f hostfile\_gpssh .bashrc =:/home/gpadmin |

Если имя пользователя не указано в списке хостов или с *user*@ в пути к файлу, gpscp скопирует файлы как текущий авторизованный пользователь. Чтобы определить текущего пользователя, вошедшего в систему, выполните команду whoami. По умолчанию gpscp переходит к $HOME пользователя сеанса на удалённых хостах после входа в систему. Чтобы файл был скопирован в правильное место на удалённых хостах, рекомендуется использовать абсолютные пути.

Перед использованием gpscp вы должны настроить доверенный доступ между хостами, участвующими в сеансе SCP. Вы можете использовать утилиту gpssh-exkeys для обновления известных файлов хоста и обмена открытыми ключами между хостами, если вы этого еще не сделали.

Параметры

Таблица — Параметры gpscp

| Параметр | Описание |
| --- | --- |
| -f *hostfile\_gpssh* | Задаёт имя файла, содержащего список хостов, которые будут участвовать в этом сеансе SCP. Синтаксис файла хоста — по одному хосту в строке следующим образом:   |  | | --- | | <*hostname*> | |
| -h *hostname* | Задаёт одно имя хоста, который будет участвовать в этом сеансе SCP. Вы можете использовать параметр -h несколько раз, чтобы указать несколько имён хостов. |
| -J *character* | Параметр -J позволяет указать один символ для замены имени хоста в строках назначения и копировании из них. Если -J не указан, символ подстановки по умолчанию — знак равенства (=). |
| -v | Опционально. Сообщает дополнительные сообщения в дополнение к выходным данным команды SCP. |
| *file\_to\_copy* | Обязательный. Имя файла (или абсолютный путь) к файлу, который вы хотите скопировать на другие хосты (или места расположения файлов). Это может быть файл на локальном хосте или на другом названном хосте. |
| *copy\_to\_path* | Обязательный. Путь, по которому вы хотите скопировать файл(ы) на указанные хосты. Если абсолютный путь не используется, файл будет скопирован относительно $HOME пользователя сеанса. Вы также можете использовать знак равенства '=' (или другой символ, указанный вами с параметром -J) вместо hostname. Затем это будет заменено на каждое имя хоста, указанное в предоставленном файле хоста (-f) или с параметром -h. |
| -? | Отображает интерактивную справку. |
| --version | Отображает версию этой утилиты. |

Примеры

Скопировать файл с именем installer.tar на все хосты в файл hostfile\_gpssh.

|  |
| --- |
| gpscp -f hostfile\_gpssh installer.tar =:/ |

Скопировать файл с именем myfuncs.so в указанное место на хостах с именами sdw1 и sdw2:

|  |
| --- |
| gpscp -h sdw1 -h sdw2 myfuncs.so =:/usr/local/greenplum-db/lib |

gpssh

Обеспечивает SSH-доступ сразу к нескольким хостам.

Синтаксис

|  |
| --- |
| gpssh { -f *hostfile\_gpssh* | - h *hostname* [-h *hostname* ...] } [-s] [-e]  [-d *seconds*] [-t *multiplier*] [-v]  [*bash\_command*]  gpssh -?  gpssh --version |

Описание

Утилита gpssh позволяет запускать команды bash-оболочки на нескольких хостах одновременно, используя SSH (secure shell). Вы можете выполнить одну команду, указав её в командной строке, или перевести команду для входа в интерактивный сеанс командной строки.

Чтобы указать хосты, участвующие в сеансе SSH, используйте параметр ‑f, чтобы указать файл, содержащий список имён хостов, или используйте параметр -h, чтобы указать имена отдельных хостов в командной строке. Требуется хотя бы одно имя хоста (-h) или файл хоста (‑f). Обратите внимание, что текущий хост не включён в сеанс по умолчанию, чтобы включить локальный хост, вы должны явно объявить его в списке хостов, участвующих в сеансе.

Перед использованием gpssh необходимо настроить доверенный доступ между хостами, участвующими в сеансе SSH. Вы можете использовать утилиту gpssh-exkeys для обновления известных файлов хоста и обмена открытыми ключами между хостами, если вы этого ещё не сделали.

Если вы не укажете команду в командной строке, gpssh перейдёт в интерактивный режим. В командной строке gpssh (=>) вы можете ввести команду, как в обычной командной строке терминала bash, и команда будет выполняться на всех хостах, участвующих в сеансе. Чтобы завершить интерактивный сеанс, нажмите CTRL + D на клавиатуре или введите exit или quit.

Если имя пользователя не указано в файле хоста, gpssh будет выполнять команды от имени текущего вошедшего в систему пользователя. Чтобы определить текущего пользователя, вошедшего в систему, выполните команду whoami. По умолчанию gpssh переходит в $HOME пользователя сеанса на удалённых хостах после входа в систему. Чтобы команды выполнялись правильно на всех удалённых хостах, вы всегда должны вводить абсолютные пути.

Если вы столкнулись с проблемами тайм-аута сети при использовании gpssh, вы можете использовать параметры -d и -t или установить параметры в файле gpssh.conf для управления временем, которое gpssh использует при проверке начального ssh-соединения. Для получения информации о файле конфигурации см. Файл конфигурации gpssh.

Параметры

Таблица — Параметры gpssh

| Параметр | Описание |
| --- | --- |
| *bash\_command* | Команда оболочки bash, выполняемая на всех хостах, участвующих в этом сеансе (необязательно заключена в кавычки). Если не указан, gpssh запускает интерактивный сеанс. |
| -d *seconds* | Опционально. Задаёт время ожидания в секундах перед началом взаимодействия gpssh с ssh. По умолчанию — 0.05. Этот параметр переопределяет значение delaybeforesend, указанное в файле конфигурации gpssh.conf.  Увеличение этого значения может вызвать длительное время ожидания во время запуска gpssh. |
| -e | Опционально. Отображает команды, переданные каждому хосту, и их результат при работе в неинтерактивном режиме. |
| -f *hostfile\_gpssh* | Задаёт имя файла, содержащего список хостов, которые будут участвовать в этом сеансе SSH. Синтаксис файла хоста — по одному хосту на строку. |
| -h *hostname* | Задаёт одно имя хоста, который будет участвовать в этом сеансе SSH. Вы можете использовать параметр -h несколько раз, чтобы указать несколько имён хостов. |
| -s | Опционально. Если параметр указан, перед выполнением любых команд на целевом хосте, gpssh отправляет файл greenplum\_path.sh в каталог, указанный в переменной среды $GPHOME. Этот параметр действителен как для интерактивного режима, так и для однокомандного режима. |
| -t *multiplier* | Опционально. Десятичное число больше 0 (нуля), которое является множителем для тайм-аута, который gpssh использует при проверке валидации ssh. По умолчанию — 1. Этот параметр переопределяет значение prompt\_validation\_timeout, указанное в файле конфигурации gpssh.conf. Увеличение этого значения имеет небольшое влияние во время запуска gpssh. |
| -v | Опционально. Сообщает дополнительные сообщения в дополнение к выходным данным команды при работе в неинтерактивном режиме. |
| --version | Отображает версию этой утилиты. |
| -? (help) | Отображает интерактивную справку. |

Файл конфигурации gpssh

Файл gpssh.conf содержит параметры, позволяющие настроить время, которое gpssh использует при проверке начального ssh-соединения. Эти параметры влияют на сетевое соединение до того, как сеанс gpssh выполнит команды с ssh. Расположение файла определяется переменной среды MASTER\_DATA\_DIRECTORY. Если переменная среды не определена или файл gpssh.conf не существует, gpssh использует значения по умолчанию или значения, установленные с помощью параметров -d и -t.

Файл gpssh.conf — это текстовый файл, состоящий из раздела [gpssh] и параметров. В строке # (знак решётки) обозначает начало комментария. Это пример файла gpssh.conf.

|  |
| --- |
| [gpssh]  delaybeforesend = 0.05  prompt\_validation\_timeout = 1.0  sync\_retries = 5 |

Таблица — Параметры gpssh.conf

| Параметр | Описание |
| --- | --- |
| delaybeforesend = *seconds* | Задаёт время ожидания в секундах до начала взаимодействия gpssh с ssh. По умолчанию — 0.05. Увеличение этого значения может вызвать длительное время ожидания при запуске gpssh. Параметр -d переопределяет этот параметр. |
| prompt\_validation\_timeout = *multiplier* | Десятичное число больше 0 (нуля), которое является множителем для тайм-аута, который gpssh использует при проверке приглашения ssh. Увеличение этого значения оказывает небольшое влияние на запуск gpssh. По умолчанию — 1. Параметр -t переопределяет этот параметр. |
| sync\_retries = *attempts* | Неотрицательное целое число, указывающее максимальное количество попыток gpssh подключения к удалённому хосту RT.Warehouse. По умолчанию — 3. Если значение равно 0, gpssh возвращает ошибку, если первоначальная попытка подключения не удалась. Увеличение количества попыток также увеличивает время между повторами попыток. Этот параметр нельзя настроить с помощью параметра командной строки. Параметр -t также влияет на время между повторами попыток. Увеличение этого значения может компенсировать низкую производительность сети или проблемы с производительностью хоста сегмента, такие как высокая загрузка ЦПУ или ввода-вывода. Однако, если соединение не может быть установлено, увеличенное значение также увеличивает задержку при возврате ошибки. |

Примеры

Запустить интерактивный групповой сеанс SSH со всеми хостами, перечисленными в файле hostfile\_gpssh:

|  |
| --- |
| $ gpssh -f hostfile\_gpssh |

В интерактивной командной строке gpssh запустить shell-команду на всех хостах, участвующих в этом сеансе.

|  |
| --- |
| => ls -a /data/primary/\* |

Выйти из интерактивного сеанса:

|  |
| --- |
| => exit  => quit |

Запустить неинтерактивный групповой сеанс SSH с хостами с именами sdw1 и sdw2 и передать gpssh файл, содержащий несколько команд с именем command\_file:

|  |
| --- |
| $ gpssh -h sdw1 -h sdw2 -v -e < command\_file |

Выполнить одиночные команды в неинтерактивном режиме на хостах sdw2 и localhost:

|  |
| --- |
| $ gpssh -h sdw2 -h localhost -v -e 'ls -a /data/primary/\*'  $ gpssh -h sdw2 -h localhost -v -e 'echo $GPHOME'  $ gpssh -h sdw2 -h localhost -v -e 'ls -1 | wc -l' |

gpssh-exkeys

Обменивается открытыми SSH-ключами между хостами.

Синтаксис

|  |
| --- |
| gpssh-exkeys -f *hostfile\_exkeys* | -h *hostname* [-h *hostname* ...]  gpssh-exkeys -e *hostfile\_exkeys* -x *hostfile\_gpexpand*  gpssh-exkeys -?  gpssh-exkeys --version |

Описание

Утилита gpssh-exkeys обменивается SSH-ключами между указанными именами хостов (или адресами хостов). Это позволяет устанавливать SSH-соединения между хостами RT.Warehouse и сетевыми интерфейсами без запроса пароля. Утилита используется для первоначальной подготовки системы RT.Warehouse для доступа по SSH без пароля, а также для подготовки дополнительных хостов для доступа по SSH без пароля при расширении системы RT.Warehouse.

Ключи обмениваются как текущий авторизованный пользователь. Вы запускаете утилиту на хосте мастера как пользователь gpadmin (пользователь, которому назначена ваша установка RT.Warehouse). Утилиты управления RT.Warehouse требуют, чтобы пользователь gpadmin был создан на всех хостах в системе RT.Warehouse, а утилиты должны иметь возможность подключаться от имени этого пользователя ко всем хостам без запроса пароля.

Вы также можете использовать gpssh-exkeys для включения SSH без пароля для дополнительных пользователей, например, root.

Утилита gpssh-exkeys имеет следующие предварительные требования:

1. Пользователь должен иметь учётную запись на мастере, резервном мастере и каждом хосте сегмента в кластере RT.Warehouse.

У пользователя должна быть установлена ​​пара SSH-ключей id\_rsa на хосте мастера.

Пользователь должен иметь возможность подключаться по SSH от хоста мастера к любой другой хост-машине без ввода пароля ("1-n passwordless SSH").

Вы можете включить "1-n passwordless SSH" с помощью команды ssh-copy-id, чтобы добавить открытый ключ пользователя в файл authorized\_keys каждого хоста. Утилита gpssh-exkeys включает "n-n passwordless SSH", что позволяет пользователю подключаться по SSH с любого хоста к любому другому хосту в кластере без пароля.

Чтобы указать хосты, участвующие в обмене ключами SSH, используйте параметр -f, чтобы указать файл, содержащий список имён хостов (рекомендуется), или используйте параметр -h, чтобы указать имена отдельных хостов в командной строке. Требуется хотя бы одно имя хоста (‑h) или файл хоста (-f). Обратите внимание, что локальный хост включён в обмен ключами по умолчанию.

Чтобы указать новые хосты расширения, которые будут добавлены в существующую систему RT.Warehouse, используйте параметры -e и ‑x. Параметр -e указывает файл, содержащий список существующих хостов в системе, которые постоянно обменивались ключами SSH. Параметр -x указывает файл, содержащий список новых хостов, которые должны участвовать в обмене ключами SSH.

Утилита gpssh-exkeys выполняет обмен ключами, используя следующие шаги:

1. Добавляет открытый ключ пользователя в собственный файл authorized\_keys пользователя на текущем хосте.

Обновляет файл known\_hosts текущего пользователя с помощью ключа хоста каждого хоста, указанного с помощью параметров ‑h, ‑f, -e и -x.

Подключается к каждому хосту с помощью ssh и получает файлы authorized\_keys, known\_hosts и id\_rsa.pub пользователя.

Добавляет ключи из файлов id\_rsa.pub, полученных от каждого хоста, в файл authorized\_keys текущего пользователя.

Обновляет файлы authorized\_keys, known\_hosts и id\_rsa.pub на всех хостах, добавляя информацию о новом хосте (если есть).

Параметры

Таблица — Параметры gpssh-exkeys

| Параметр | Описание |
| --- | --- |
| -e *hostfile\_exkeys* | При расширении системы это имя и расположение файла, содержащего все настроенные имена хостов и адреса хостов (имена интерфейсов) для каждого хоста в вашей текущей системе RT.Warehouse (мастера, резервного мастера и сегментов), по одному имени в строке без пустых строк или лишних пробелов. Хосты, указанные в этом файле, не могут быть указаны в файле хоста, используемом с -x. |
| -f *hostfile\_exkeys* | Задаёт имя и расположение файла, содержащего все настроенные имена хостов и адреса хостов (имена интерфейсов) для каждого хоста в вашей системе RT.Warehouse (мастера, резервного мастера и сегментов), по одному имени в строке без пустых строк или лишних пробелов. |
| -h *hostname* | Задаёт одно имя хоста (или адрес хоста), которое будет участвовать в обмене ключами SSH. Вы можете использовать параметр -h несколько раз, чтобы указать несколько имён хостов и адресов хостов. |
| --version | Отображает версию этой утилиты. |
| -x *hostfile\_gpexpand* | При расширении системы это имя и расположение файла, содержащего все настроенные имена хостов и адреса хостов (имена интерфейсов) для каждого нового хоста сегмента, который вы добавляете в свою систему RT.Warehouse, по одному имени в строке без пустых строк или лишних пробелов. Хосты, указанные в этом файле, не могут быть указаны в файле хоста, используемом с -e. |
| -? | Отображает интерактивную справку. |

Примеры

Обменяться ключами SSH между всеми именами хостов и адресами, перечисленными в файле hostfile\_exkeys:

|  |
| --- |
| $ gpssh-exkeys -f hostfile\_exkeys |

Обменяться ключами SSH между хостами sdw1, sdw2 и sdw3:

|  |
| --- |
| $ gpssh-exkeys -h sdw1 -h sdw2 -h sdw3 |

Обменяться ключами SSH между существующими хостами sdw1, sdw2 и sdw3 и новыми хостами sdw4 и sdw5 в рамках операции расширения системы:

|  |
| --- |
| $ cat hostfile\_exkeys  mdw  mdw-1  mdw-2  smdw  smdw-1  smdw-2  sdw1  sdw1-1  sdw1-2  sdw2  sdw2-1  sdw2-2  sdw3  sdw3-1  sdw3-2  $ cat hostfile\_gpexpand  sdw4  sdw4-1  sdw4-2  sdw5  sdw5-1  sdw5-2  $ gpssh-exkeys -e hostfile\_exkeys -x hostfile\_gpexpand |

gpstart

Запускает систему RT.Warehouse.

Синтаксис

|  |
| --- |
| gpstart [-d *master\_data\_directory*] [-B *parallel\_processes*] [-R]  [-m] [-y] [-a] [-t *timeout\_seconds*] [-l *logfile\_directory*]  [--skip-heap-checksum-validation]  [-v | -q]  gpstart -? | -h | --help  gpstart --version |

Описание

Утилита gpstart используется для запуска процессов сервера RT.Warehouse. Когда вы запускаете систему RT.Warehouse, вы фактически запускаете сразу несколько процессов слушателя сервера баз данных postgres (мастер и все инстансы сегментов). Утилита gpstart обрабатывает запуск отдельных инстансов. Каждый инстанс запускается параллельно.

При первом запуске администратор запускает утилиту gpstart, утилита создает файл кэша хостов с именем .gphostcache в домашнем каталоге пользователя. Впоследствии утилита использует этот список хостов для более эффективного запуска системы. Если в систему добавляются новые хосты, вы должны вручную удалить этот файл из домашнего каталога пользователя gpadmin. Утилита создаст новый файл кэша хостов при следующем запуске.

Как часть процесса запуска, утилита проверяет согласованность настройки контрольной суммы кучи (heap) между инстансом мастера RT.Warehouse и инстансами сегмента, включённых или отключённых на всех инстансах. Если контрольная сумма heap в инстансах различается, возвращается ошибка, и RT.Warehouse не запускается. Проверка может быть отключена, указав параметр --skip-heap-checkum-validation.

|  |
| --- |
| **Примечание.**  Прежде чем вы сможете запустить систему RT.Warehouse, вы должны инициализировать систему с помощью gpinitsystem. Включение или отключение контрольных сумм heap устанавливается при инициализации системы и не может быть изменено после инициализации. |

Параметры

Таблица — Параметры gpstart

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать у пользователя подтверждение. |
| -B *parallel\_processes* | Количество сегментов для параллельного запуска. Если параметр не указан, утилита запустит до 64 параллельных процессов в зависимости от того, сколько инстансов сегмента ей нужно запустить. |
| -d *master\_data\_directory* | Опционально. Каталог данных хоста мастера. Если параметр не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -m | Опционально. Запускает только инстанс мастера, что может быть полезно для задач обслуживания. Этот режим позволяет подключаться к мастеру только в режиме утилит. Например:   |  | | --- | | PGOPTIONS='-c gp\_session\_role=utility' psql |   Согласованность настройки контрольной суммы кучи (heap) на инстансе мастера и инстансах сегмента не проверяется. |
| -q | Работать в тихом режиме. Вывод команды не отображается на экране, но все равно записывается в файл журнала. |
| -R | Запускает RT.Warehouse в ограниченном режиме (подключение разрешено только суперпользователям базы данных). |
| --skip-heap-checksum-validation | Во время запуска утилита не проверяет согласованность настройки контрольной суммы кучи (heap) для инстанса мастера RT.Warehouse и инстансов сегмента. По умолчанию гарантируется, что настройка контрольной суммы heap одинакова для всех инстансов, включённых или отключённых.  **Внимание**. Запуск RT.Warehouse без этой проверки может привести к потере данных. Используйте этот параметр для запуска RT.Warehouse только тогда, когда необходимо игнорировать ошибки проверки контрольной суммы кучи для восстановления данных или устранения ошибок. |
| -t *timeout\_seconds* | Задаёт тайм-аут в секундах для ожидания запуска инстанса сегмента. Если инстанс сегмента был аварийно завершён (например, из-за сбоя питания или прерывания процесса слушателя базы данных postgres), запуск может занять больше времени из-за процесса восстановления и проверки базы данных. Если параметр не указан, время ожидания по умолчанию составляет 60 секунд. |
| -v | Отображает подробные сообщения о состоянии, прогрессе и ошибках, выводимые утилитой. |
| -y | Опционально. Не запускать хост резервного мастера. По умолчанию запускается хост резервного мастера и процесс синхронизации. |
| -? | -h | --help | Отображает интерактивную справку. |
| --version | Отображает версию этой утилиты. |

Примеры

Запустить систему RT.Warehouse:

|  |
| --- |
| gpstart |

Запустить систему RT.Warehouse в ограниченном режиме (разрешить соединения только суперпользователя):

|  |
| --- |
| gpstart -R |

Запустить только инстанс мастера RT.Warehouse и подключиться в режиме утилит:

|  |
| --- |
| gpstart -m PGOPTIONS='-c gp\_session\_role=utility' psql |

gpstate

Показывает состояние работающей системы RT.Warehouse.

Синтаксис

|  |
| --- |
| gpstate [-d *master\_data\_directory*] [-B *parallel\_processes*]  [-s | -b | -Q | -e] [-m | -c] [-p] [-i] [-f] [-v | -q] | -x  [-l *log\_directory*]  gpstate -? | -h | --help |

Описание

Утилита gpstate отображает информацию о запущенном инстансе RT.Warehouse. Существует дополнительная информация, которую вы можете захотеть узнать о системе RT.Warehouse, поскольку она состоит из нескольких инстансов (сегментов) базы данных PostgreSQL, охватывающих несколько машин. Утилита gpstate предоставляет дополнительную информацию о состоянии системы RT.Warehouse, например:

1. Какие сегменты не работают.

Информация о конфигурации мастера и сегмента (хосты, каталоги данных и т.д.).

Порты, используемые системой.

Отображение основных сегментов на соответствующие им зеркальные сегменты.

Параметры

Таблица — Параметры gpstate

| Параметр | Описание |
| --- | --- |
| -b | Опционально. Отображает краткое описание состояния системы RT.Warehouse. Это вариант по умолчанию. |
| -B *parallel\_processes* | Количество сегментов для параллельной проверки. Если параметр не указан, утилита запустит до 60 параллельных процессов в зависимости от того, сколько инстансов сегментов необходимо проверить. |
| -c | Опционально. Отображает маппинг основных сегментов на соответствующие им зеркальные сегменты. |
| -d *master\_data\_directory* | Опционально. Каталог данных мастера. Если не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| -e | Показать подробную информацию о парах основных и зеркальных сегментов, которые имеют потенциальные проблемы, такие как:   * Активный сегмент работает в режиме отслеживания изменений, что означает, что сегмент не работает; * Активный сегмент находится в режиме ресинхронизации, что означает, что он отлавливает изменения в зеркале; * Сегмент не находится в своей предпочтительной роли, например сегмент, который был основным во время инициализации системы, теперь действует как зеркало, что означает, что у вас может быть один или несколько хостов сегментов с несбалансированной нагрузкой обработки. |
| -f | Отображает сведения о хосте резервного мастера, если он настроен. |
| -i | Отображает информацию о версии программного обеспечения RT.Warehouse для каждого инстанса. |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -m | Опционально. Перечисляет инстансы зеркальных сегментов в системе, их текущую роль и состояние синхронизации. |
| -p | Перечисляет номера портов, используемые в системе RT.Warehouse. |
| -q | Опционально. Работать в тихом режиме. За исключением предупреждающих сообщений, вывод команды не отображается на экране. Однако эта информация по-прежнему записывается в файл журнала. |
| -Q | Опционально. Проверяет статус сегмента в системном каталоге на хосте мастера. Не запрашивает статус сегментов. |
| -s | Опционально. Отображает подробную информацию о состоянии системы RT.Warehouse. |
| -v | Опционально. Отображает сообщения об ошибках и выводит подробную информацию о состоянии и прогрессе. |
| -x | Опционально. Отображает подробную информацию о ходе и состоянии расширения системы RT.Warehouse. |
| -? | -h | --help | Отображает интерактивную справку. |

Определения выходных полей

Следующие поля вывода сообщаются gpstate -s для мастера:

Таблица — Выходные данные gpstate –s для мастера

| Выходные данные | Описание |
| --- | --- |
| Master host | Имя хоста мастера. |
| Master postgres process ID | PID процесса слушателя базы данных мастера. |
| Master data directory | Расположение в файловой системе каталога данных мастера. |
| Master port | Порт процесса слушателя базы данных postgres мастера. |
| Master current role | dispatch — обычный режим работы.  utility — режим обслуживания. |
| Greenplum array configuration type | standard — одна сетевая карта (NIC) на хост.  multi-home — несколько сетевых карт (NIC) на хост. |
| Greenplum initsystem version | Версия RT.Warehouse при первой инициализации системы. |
| Greenplum current version | Текущая версия RT.Warehouse. |
| Postgres version | Версия PostgreSQL, на которой основана RT.Warehouse. |
| Greenplum mirroring status | Физическое зеркалирование или нет. |
| Master standby | Имя хоста резервного мастера. |
| Standby master state | Статус резервного мастера: активный (active) или пассивный (passive). |

gpstate -s сообщает следующие поля вывода для каждого сегмента:

Таблица — Выходные данные gpstate –s для сегментов

| Выходные данные | Описание |
| --- | --- |
| Hostname | Системное имя хоста. |
| Address | Сетевой адрес имени хоста (имя сетевой карты (NIC)). |
| Datadir | Расположение в файловой системе каталога данных сегмента. |
| Port | Номер порта процесса слушателя базы данных postgres. |
| Current Role | Текущая роль сегмента: зеркало (mirror) или основной (primary). |
| Preferred Role | Роль во время инициализации системы: зеркало (mirror) или основной (primary). |
| Mirror Status | Статус пары основного и зеркального сегментов:   * synchronized — данные актуальны на обоих. * resynchronization — данные в настоящее время копируются из одного в другой. * change tracking — сегмент не работает, а активный сегмент регистрирует изменения. |
| Change tracking data size | В режиме change tracking размер файла журнала изменений (может увеличиваться и уменьшаться по мере применения сжатия). |
| Estimated total data to synchronize | В режиме resynchronization предполагаемый размер данных, оставшихся для синхронизации. |
| Data synchronized | В режиме resynchronization предполагаемый размер данных, которые уже были синхронизированы. |
| Estimated resync progress with mirror | В режиме resynchronization предполагаемый процент завершения. |
| Estimated resync end time | В режиме resynchronization расчётное время для завершения. |
| File postmaster.pid | Статус файла блокировки postmaster.pid: found или missing. |
| PID from postmaster.pid file | PID, найденный в файле postmaster.pid. |
| Lock files in /tmp | Файл блокировки порта сегмента для его процесса postgres создаётся в /tmp (файл удаляется, когда сегмент завершает работу). |
| Active PID | Идентификатор активного процесса сегмента. |
| Master reports status as | Статус сегмента, как указано в системном каталоге: up или down. |
| Database status | Статус RT.Warehouse для входящих запросов: up, down, или suspended. Состояние suspended означает, что активность базы данных временно приостанавливается, пока сегмент переходит из одного состояния в другое. |

Следующие поля вывода сообщаются gpstate -f для статуса резервного мастера:

Таблица — Выходные данные gpstate –f для резервного мастера

| Выходные данные | Описание |
| --- | --- |
| Standby address | Имя хоста резервного мастера. |
| Standby data dir | Расположение в файловой системе каталога данных резервного мастера. |
| Standby port | Порт процесса прослушивателя базы данных postgres резервного мастера. |
| Standby PID | ID процесса резервного мастера. |
| Standby status | Состояние резервного мастера: standby host passive. |
| WAL Sender State | Состояние потоковой передачи журнала предзаписи (WAL): streaming, startup, backup, catchup. |
| Sync state | Состояние синхронизации отправителя WAL: sync. |
| Sent Location | Местоположение отправленной записи журнала транзакций отправителя WAL (xlog). |
| Flush Location | Место сброса записи xlog приёмника WAL. |
| Replay Location | Резервное место воспроизведения записи xlog. |

Примеры

Показать подробную информацию о состоянии системы RT.Warehouse:

|  |
| --- |
| gpstate -s |

Сделать быструю проверку неработающих сегментов в системном каталоге хоста мастера:

|  |
| --- |
| gpstate -Q |

Показать информацию об инстансах зеркального сегмента:

|  |
| --- |
| gpstate -m |

Показать информацию о конфигурации резервного мастера:

|  |
| --- |
| gpstate -f |

Отобразить информацию о версии программного обеспечения RT.Warehouse:

|  |
| --- |
| gpstate -i |

gpstop

Останавливает или перезапускает систему RT.Warehouse.

Синтаксис

|  |
| --- |
| gpstop [-d *master\_data\_directory*] [-B *parallel\_processes*]  [-M smart | fast | immediate] [-t *timeout\_seconds*] [-r] [-y] [-a]  [-l *logfile\_directory*] [-v | -q]  gpstop -m [-d *master\_data\_directory*] [-y] [-l *logfile\_directory*] [-v | -q]  gpstop -u [-d *master\_data\_directory*] [-l *logfile\_directory*] [-v | -q]    gpstop --host host\_name [-d *master\_data\_directory*] [-l *logfile\_directory*]  [-t *timeout\_seconds*] [-a] [-v | -q]  gpstop --version  gpstop -? | -h | --help |

Описание

Утилита gpstop используется для остановки серверов баз данных, входящих в систему RT.Warehouse. Когда вы останавливаете систему RT.Warehouse, вы фактически останавливаете сразу несколько процессов сервера баз данных postgres (мастер и все инстансы сегментов). Утилита gpstop обрабатывает завершение работы отдельных инстансов. Каждый инстансов выключается параллельно.

Режим выключения по умолчанию (-M smart) ожидает завершения текущих клиентских подключений перед завершением выключения. Если какие-либо соединения остаются открытыми по истечении периода тайм-аута или если вы прерываете соединение с помощью CTRL-C, gpstop выводит список открытых соединений и подсказывает, следует ли продолжать ожидание завершения соединений или выполнить быстрое или немедленное завершение работы. Период ожидания по умолчанию составляет 120 секунд и может быть изменён с помощью параметра -t *timeout\_seconds*.

Укажите -M fast режим выключения, чтобы откатить все выполняющиеся транзакции и разорвать любые соединения перед завершением работы.

С параметром -u утилита загружает изменения, внесённые в файл мастера pg\_hba.conf или в параметры конфигурации среды выполнения в файле мастера postgresql.conf, без прерывания обслуживания. Обратите внимание, что любые активные сеансы не будут принимать изменения, пока они не подключатся к базе данных.

Параметры

Таблица — Параметры gpstop

| Параметр | Описание |
| --- | --- |
| -a | Не запрашивать подтверждение у пользователя. |
| -B *parallel\_processes* | Количество сегментов, которые нужно остановить параллельно. Если параметр не указан, утилита запустит до 64 параллельных процессов в зависимости от того, сколько инстансов сегмента ей необходимо остановить. |
| -d *master\_data\_directory* | Опционально. Каталог данных хоста мастера. Если не указан, будет использоваться значение, установленное для $MASTER\_DATA\_DIRECTORY. |
| --host *host\_name* | Утилита завершает работу инстансов сегмента RT.Warehouse на указанном хосте, чтобы разрешить обслуживание хоста. Каждый инстанс основного сегмента на хосте отключается, а связанный инстанс зеркального сегмента повышается до основного сегмента, если зеркальный сегмент находится на другом хосте. Инстансы зеркальных сегментов на хосте выключаются.  Инстансы сегментов не выключаются, и утилита возвращает ошибку в следующих случаях:   * Зеркалирование сегментов не включено для системы. * Мастер или резервный мастер активны на хосте. * Инстанс основного сегмента и его зеркало активны на хосте.   Этот параметр нельзя указывать с параметрами -m, -r, -u или ‑y.  **Примечание**. Утилита gprecoverseg восстанавливает инстансы сегментов. Запуск команды gprecoverseg запускает сегменты как зеркала, а затем возвращает сегменты в их предпочтительную роль (основные сегменты). |
| -l *logfile\_directory* | Каталог для записи файла журнала. По умолчанию ~/gpAdminLogs. |
| -m | Опционально. Завершает работу инстанса мастера RT.Warehouse, запущенного в режиме обслуживания. |
| -M fast | Быстрое выключение. Любые текущие транзакции прерываются и откатываются. |
| -M immediate | Немедленное отключение. Любые выполняющиеся транзакции прерываются. Этот режим завершает все процессы postgres, не позволяя серверу базы данных завершить обработку транзакции или очистить любые временные или находящиеся в процессе рабочие файлы. |
| -M smart | Умное выключение. Это режим выключения по умолчанию. gpstop ожидает отключения активных пользовательских подключений и затем завершает работу. Если какие-либо пользовательские подключения остаются открытыми после периода тайм-аута (или если вы прерываете их нажатием CTRL-C), gpstop выводит список открытых пользовательских подключений и предлагает продолжить ожидание завершения подключений или выполнить быстрое или немедленное завершение работы. |
| -q | Работа в тихом режиме. Вывод команды не отображается на экране, но все равно записывается в файл журнала. |
| -r | Перезагрузить после завершения выключения. |
| -t *timeout\_seconds* | Задаёт порог тайм-аута (в секундах) для ожидания завершения работы инстанса сегмента. Если инстанс сегмента не завершает работу в течение указанного количества секунд, gpstop отображает сообщение, указывающее, что один или несколько сегментов всё ещё находятся в процессе завершения работы и что вы не можете перезапустить RT.Warehouse, пока инстансы сегмента не будут остановлены. Этот параметр полезен в ситуациях, когда выполняется gpstop и есть очень большие транзакции, которые необходимо откатить. Для отката таких крупных транзакций может потребоваться более минуты, а время ожидания по умолчанию превышает 120 секунд. |
| -u | Этот параметр перезагружает файлы pg\_hba.conf мастера и сегментов, а также параметры времени выполнения файлов postgresql.conf, но не завершает работу массива RT.Warehouse. Используйте этот параметр, чтобы активировать новые настройки конфигурации после редактирования postgresql.conf или pg\_hba.conf. Обратите внимание, что это относится только к параметрам конфигурации, которые определены как параметры времени выполнения. |
| -v | Отображает подробные сообщения о состоянии, прогрессе и ошибках, выводимые утилитой. |
| -y | Не останавливает процесс резервного мастера. По умолчанию резервный мастер останавливается. |
| -? | -h | --help | Отображает интерактивную справку. |
| --version | Отображает версию этой утилиты. |

Примеры

Остановить систему RT.Warehouse в интеллектуальном режиме:

|  |
| --- |
| gpstop |

Остановить систему RT.Warehouse в быстром режиме:

|  |
| --- |
| gpstop -M fast |

Остановить все инстансы сегментов, а затем перезапустить систему:

|  |
| --- |
| gpstop -r |

Остановить инстанс мастера, запущенный в режиме обслуживания:

|  |
| --- |
| gpstop -m |

После внесения изменений в конфигурацию перезагрузить файлы postgresql.conf и pg\_hba.conf, но не завершать работу массива RT.Warehouse:

|  |
| --- |
| gpstop -u |

gpsys1

Отображает информацию о вашей операционной системе.

Синтаксис

|  |
| --- |
| gpsys1 [ -a | -m | -p ]  gpsys1 -?  gpsys1 --version |

Описание

gpsys1 отображает платформу и установленную память (в байтах) текущего хоста. Например:

|  |
| --- |
| linux 1073741824 |

Параметры

Таблица — Параметры gpsys1

| Параметр | Описание |
| --- | --- |
| -a | Показывает информацию о платформе и памяти для текущего хоста. Это значение по умолчанию. |
| -m | Показывает только установленную системную память в байтах. |
| -p | Показывает только платформу ОС. Платформа может быть linux, darwin или sunos5. |
| -? | Отображает интерактивную справку. |
| --version | Отображает версию этой утилиты. |

Примеры

Показать информацию о текущей операционной системе хоста:

|  |
| --- |
| gpsys1 |

pg\_config

Получает информацию об установленной версии RT.Warehouse.

Синтаксис

|  |
| --- |
| pg\_config [*option* ...]  pg\_config -? | --help  pg\_config --version |

Описание

Утилита pg\_config распечатывает параметры конфигурации текущей установленной версии RT.Warehouse. Она предназначена, например, для использования программными пакетами, которые хотят взаимодействовать с RT.Warehouse, чтобы облегчить поиск необходимых файлов заголовков и библиотек. Обратите внимание, что информация, распечатанная с помощью pg\_config, только о мастере RT.Warehouse.

Если указано более одного параметра, информация печатается в указанном порядке, по одному элементу в строке. Если параметры не указаны, вся доступная информация печатается с этикетками.

Параметры

Таблица — Параметры pg\_config

| Параметр | Описание |
| --- | --- |
| --bindir | Выводит расположение исполняемых файлов пользователя. Используйте этот параметр, например, чтобы найти программу psql. Обычно это то же место, где находится программа pg\_config. |
| --docdir | Выводит расположение файлов документации. |
| --includedir | Выводит расположение файлов C-заголовков клиентских интерфейсов. |
| --pkgincludedir | Выводит расположение других файлов C-заголовков. |
| --includedir-server | Выводит расположение файлов C-заголовков для программирования сервера. |
| --libdir | Выводит расположение библиотек объектного кода. |
| --pkglibdir | Выводит расположение динамически загружаемых модулей или место, где сервер будет их искать. (Другие файлы данных, зависящие от архитектуры, также могут быть установлены в этом каталоге.) |
| --localedir | Выводит расположение файлов поддержки локали. |
| --mandir | Выводит расположение страниц руководства. |
| --sharedir | Выводит расположение файлов поддержки, не зависящих от архитектуры. |
| --sysconfdir | Выводит расположение общесистемных файлов конфигурации. |
| --pgxs | Выводит расположение make-файлов расширения. |
| --configure | Выводит параметры, которые были заданы скрипту настройки при настройке RT.Warehouse для сборки. |
| --cc | Выводит значение переменной CC, которая использовалась для сборки RT.Warehouse. Параметр показывает используемый компилятор C. |
| --cppflags | Выводит значение переменной CPPFLAGS, которая использовалась для сборки RT.Warehouse. Параметр показывает переключатели компилятора C, необходимые во время предварительной обработки. |
| --cflags | Выводит значение переменной CFLAGS, которая использовалась для сборки RT.Warehouse. Параметр показывает переключатели компилятора C. |
| --cflags\_sl | Выводит значение переменной CFLAGS\_SL, которая использовалась для сборки RT.Warehouse. Параметр показывает дополнительные переключатели компилятора C, используемые для создания общих библиотек. |
| --ldflags | Выводит значение переменной LDFLAGS, которая использовалась для сборки RT.Warehouse. Параметр показывает переключатели компоновщика. |
| --ldflags\_ex | Выводит значение переменной LDFLAGS\_EX, которая использовалась для сборки RT.Warehouse. Параметр показывает переключатели компоновщика, которые использовались только для сборки исполняемых файлов. |
| --ldflags\_sl | Выводит значение переменной LDFLAGS\_SL, которая использовалась для сборки RT.Warehouse. Параметр показывает переключатели компоновщика, используемые только для создания общих библиотек. |
| --libs | Выводит значение переменной LIBS, которая использовалась для сборки RT.Warehouse. Обычно параметр содержит ключи -l для внешних библиотек, связанных с RT.Warehouse. |
| --version | Выводит версию RT.Warehouse. |

Примеры

Чтобы воспроизвести конфигурацию сборки текущей установки RT.Warehouse, выполните следующую команду:

|  |
| --- |
| eval ./configure 'pg\_config --configure' |

Вывод pg\_config --configure содержит кавычки оболочки, поэтому аргументы с пробелами представлены правильно. Следовательно, для получения правильных результатов необходимо использовать eval.

pg\_dump

Извлекает базу данных в отдельный файл скрипта или другой архивный файл.

Синтаксис

|  |
| --- |
| pg\_dump [*connection-option* ...] [*dump\_option* ...] [*dbname*]  pg\_dump -? | --help  pg\_dump -V | --version |

Описание

pg\_dump — это стандартная утилита PostgreSQL для резервного копирования базы данных, которая также поддерживается в RT.Warehouse. Создаёт единый (непараллельный) файл дампа. Для регулярного резервного копирования RT.Warehouse лучше использовать утилиту резервного копирования RT.Warehouse gpbackup для обеспечения максимальной производительности.

Используйте pg\_dump, если вы переносите свои данные в систему базы данных другого вендора или в другую систему RT.Warehouse с другой конфигурацией сегментов (например, если система, в которую вы мигрируете, имеет большее или меньшее количество инстансов сегментов). Для восстановления вы должны использовать соответствующую утилиту pg\_restore (если файл дампа находится в формате архива) или вы можете использовать клиентскую программу, такую ​​как psql (если файл дампа находится в текстовом формате).

Поскольку pg\_dump совместима с обычной PostgreSQL, её можно использовать для переноса данных в RT.Warehouse. Утилита pg\_dump в RT.Warehouse очень похожа на утилиту pg\_dump PostgreSQL, со следующими исключениями и ограничениями:

1. При использовании pg\_dump для резервного копирования базы данных RT.Warehouse имейте в виду, что операция дампа может занять много времени (несколько часов) для очень больших баз данных. Кроме того, вы должны убедиться, что у вас достаточно места на диске для создания файла дампа.

Если вы переносите данные из одной системы RT.Warehouse в другую, используйте параметр командной строки --gp-syntax, чтобы включить предложение DISTRIBUTED BY в операторы CREATE TABLE. Это гарантирует, что данные таблицы RT.Warehouse будут распределены с правильными столбцами ключа распределения после восстановления.

pg\_dump делает последовательные резервные копии, даже если Одновременно с этим база данных используется. pg\_dump не блокирует доступ к базе данных других пользователей (на чтение или запись).

При использовании с одним из форматов архивных файлов и в сочетании с pg\_restore, pg\_dump предоставляет гибкий механизм архивирования и передачи. pg\_dump можно использовать для резервного копирования всей базы данных, затем pg\_restore можно использовать для проверки архива и/или выбора частей базы данных для восстановления. Наиболее гибкими форматами выходных файлов являются формат *custom* (-Fc) и формат *directory* (-Fd). Они позволяют выбирать и переупорядочивать все заархивированные элементы, поддерживают параллельное восстановление и сжаты по умолчанию. Формат *directory* — единственный формат, поддерживающий параллельные дампы.

|  |
| --- |
| **Примечание.**  Параметр --ignore-version устарел и будет удален в следующем выпуске. |

Параметры

Таблица — dbname параметры pg\_dump

| Параметр | Описание |
| --- | --- |
| dbname | Задаёт имя базы данных для дампа. Если параметр не указан, используется переменная среды PGDATABASE. Если она не установлена, используется имя пользователя, указанное для соединения. |

Таблица — dump параметры pg\_dump

| Параметр | Описание |
| --- | --- |
| -a | --data-only | Выгружает только данные, а не схему (определения данных). Выгружаются данные таблицы и значения последовательности.  Этот параметр аналогичен, но по историческим причинам не идентичен, указанию параметра --section=data. |
| -b | --blobs | Включает большие объекты в дамп. Это поведение по умолчанию, за исключением случаев, когда указаны параметры --schema, --table или --schema-only. Параметр -b полезен только при добавлении больших объектов в дампы, где была запрошена конкретная схема или таблица. Обратите внимание, что большие бинарные объекты (blobs) считаются данными и поэтому будут включены, когда используется --data-only, но не когда --schema-only.  **Примечание**. RT.Warehouse не поддерживает [функцию больших объектов](https://www.postgresql.org/docs/9.4/largeobjects.html) PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| -c | --clean | Добавляет команды в текстовый файл вывода для очистки (удаления) объектов базы данных перед выводом команд для их создания. (Восстановление может привести к появлению сообщений о некоторых безобидных ошибках, если какие-либо объекты не присутствуют в целевой базе данных.) Обратите внимание, что объекты не удаляются до начала операции дампа, а команды DROP добавляются в выходные файлы дампа DDL, чтобы при использовании этих файлов для восстановления команды DROP запускались до команд CREATE. Этот параметр имеет смысл использовать только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -C | --create | Начнает вывод с команды для создания самой базы данных и повторного подключения к созданной базе данных. (Используя скрипт этой формы, не имеет значения, к какой базе данных в целевой установке вы подключаетесь перед запуском скрипта.) Если также указан параметр --clean, скрипт удаляет и воссоздает целевую базу данных перед повторным подключением к ней. Этот параметр имеет смысл использовать только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -E *encoding* | --encoding=*encoding* | Создаёт дамп в указанной кодировке набора символов. По умолчанию дамп создается в кодировке базы данных. (Другой способ получить тот же результат — установить для переменной среды PGCLIENTENCODING желаемую кодировку дампа.) |
| -f *file* | --file=*file* | Отправляет вывод в указанный файл. Этот параметр можно не указывать для форматов вывода на основе файлов, и в этом случае используется стандартный вывод. Однако он должен быть указан для формата вывода в каталог, где указывается целевой каталог вместо файла. В этом случае pg\_dump создаёт каталог, который не должен существовать раньше. |
| -F p|c|d|t | --format=plain|custom|directory|tar | Выбирает формат вывода. Формат может быть одним из следующих:   * p | plain — выводит текстовый файл SQL-скрипта (по умолчанию). * c | custom — выводит собственный архив, пригодный в качестве ввода для pg\_restore. Вместе с выходным форматом каталога (directory) это наиболее гибкий выходной формат, поскольку он позволяет вручную выбирать и переупорядочивать заархивированные элементы во время восстановления. По умолчанию этот формат сжимается, а также поддерживает параллельные дампы. * d | directory — выводит архив в формате каталога, подходящий в качестве ввода для pg\_restore. Будет создан каталог с одним файлом для каждой выгружаемой таблицы и blob-объекта, а также так называемый файл Table of Contents, описывающий выгружаемые объекты в машиночитаемом формате, который может читать pg\_restore. Архивом формата каталогов можно управлять стандартными инструментами Unix; например, файлы в несжатом архиве можно сжать с помощью инструмента gzip. По умолчанию данный формат имеет сжатие. * t | tar — выводит архив в формате tar, подходящий в качестве ввода для pg\_restore. Формат tar совместим с форматом каталога; при извлечении tar-архива получается валидный архив в формате каталога. Однако формат tar не поддерживает сжатие. Кроме того, при использовании формата tar относительный порядок элементов данных таблицы не может быть изменён во время восстановления. |
| -i | --ignore-version | **Примечание**. Эта опция устарела и будет удалена в следующем выпуске.  Игнорировать несоответствие версий между pg\_dump и сервером базы данных. pg\_dump может выгружать данные с серверов, на которых запущены предыдущие версии RT.Warehouse (или PostgreSQL), но очень старые версии могут больше не поддерживаться. Используйте этот параметр, если вам нужно отменить проверку версии. |
| -j *njobs* | --jobs=*njobs* | Запускает дамп параллельно, одновременно выгружая *njobs* таблиц. Этот параметр сокращает время дампа, но также увеличивает нагрузку на сервер базы данных. Вы можете использовать этот параметр только с форматом вывода каталога (directory), потому что это единственный формат вывода, в котором несколько процессов могут записывать свои данные одновременно.  **Примечание**. Параллельные дампы с использованием pg\_dump распараллеливаются только на хосте диспетчера запросов (мастере), а не на хостах исполнителя (сегмента) запроса, как в случае использования gpbackup.  pg\_dump откроет *njobs* + 1 соединений с базой данных, поэтому убедитесь, что ваш параметр max\_connections достаточно высок, чтобы вместить все соединения.  Запрос исключительных блокировок объектов базы данных при запуске параллельного дампа может привести к сбою дампа. Причина в том, что процесс мастера pg\_dump запрашивает общие блокировки на объектах, которые рабочие процессы собираются сбрасывать позже, чтобы гарантировать, что никто не удалит их и не заставит их уйти во время выполнения дампа. Если другой клиент затем запрашивает исключительную блокировку для таблицы, эта блокировка не будет предоставлена, но будет поставлена ​​в очередь в ожидании освобождения общей блокировки процесса мастера. Следовательно, любой другой доступ к таблице также не будет предоставлен и будет помещён в очередь после запроса исключительной блокировки. Это включает в себя рабочий процесс, пытающийся выгрузить таблицу. Без всяких мер предосторожности это была бы классическая тупиковая ситуация. Чтобы обнаружить этот конфликт, рабочий процесс pg\_dump запрашивает другую разделяемую блокировку с помощью параметра NOWAIT. Если рабочему процессу не предоставлена ​​эта разделяемая блокировка, значит, кто-то еще должен был запросить исключительную блокировку тем временем, и нет возможности продолжить создание дампа, поэтому у pg\_dump нет другого выбора, кроме как прервать создание дампа.  Для согласованного резервного копирования сервер базы данных должен поддерживать синхронизированные снапшоты — функцию, которая была представлена ​​в RT.Warehouse 6.0. С помощью этой функции клиенты базы данных могут гарантировать, что они видят один и тот же набор данных, даже если они используют разные соединения. pg\_dump -j использует несколько соединений с базой данных; он подключается к базе данных один раз с процессом мастера и ещё раз для каждого рабочего задания. Без функции синхронизированного снапшота разные рабочие задания не будут гарантированно видеть одни и те же данные в каждом соединении, что может привести к несогласованному резервному копированию.  Если вы хотите запустить параллельный дамп сервера до версии 6.0, вам необходимо убедиться, что содержимое базы данных не меняется с момента подключения мастера к базе данных до момента подключения последнего рабочего задания к базе данных. Самый простой способ сделать это — остановить любые процессы изменения данных (DDL и DML), обращающиеся к базе данных, перед запуском резервного копирования. Вам также необходимо указать параметр --no-synchronized-snapshots при запуске pg\_dump -j для сервера RT.Warehouse до версии 6.0. |
| -n *schema* | --schema=*schema* | Выгружает только схемы, соответствующие шаблону схемы; выбирается как сама схема, так и все содержащиеся в ней объекты. Если этот параметр не указан, все несистемные схемы в целевой базе данных будут сброшены. Можно выбрать несколько схем, написав несколько ключей -n. Кроме того, параметр схемы интерпретируется как шаблон в соответствии с теми же правилами, которые используются командами psql \d, поэтому несколько схем также можно выбрать, написав подстановочные символы в шаблоне. При использовании подстановочных символов будьте осторожны при необходимости заключать образец в кавычки, чтобы оболочка не расширяла подстановочные знаки.  **Примечание**. Если указан -n, pg\_dump не пытается сбросить дамп любых других объектов базы данных, от которых может зависеть выбранная схема (схемы). Следовательно, нет гарантии, что результаты дампа конкретной схемы могут быть успешно восстановлены сами по себе в чистую базу данных.  **Примечание**. Объекты, не относящиеся к схеме, такие как blob- объекты, не выгружаются, если указан параметр -n. Вы можете добавить blob-объекты обратно в дамп с помощью --blobs. |
| -N *schema* | --exclude-schema=*schema* | Не выгружает схемы, соответствующие шаблону схемы. Шаблон интерпретируется по тем же правилам, что и -n. -N можно указывать более одного раза, чтобы исключить схемы, соответствующие любому из нескольких шаблонов. Когда заданы оба параметра -n и -N, происходит сброс только тех схем, которые соответствуют хотя бы одному переключателю ‑n, но не соответствуют переключателям -N. Если -N отображается без -n, то схемы, соответствующие -N, исключаются из обычного дампа. |
| -o | --oids | Дамп идентификаторов объектов (OID) как часть данных для каждой таблицы. Использование этого параметра не рекомендуется для файлов, которые должны быть восстановлены в RT.Warehouse. |
| -O | --no-owner | Не выводить команды для установки владения объектами в соответствии с исходной базой данных. По умолчанию pg\_dump выдаёт операторы ALTER OWNER или SET SESSION AUTHORIZATION, чтобы установить владельца созданных объектов базы данных. Эти операторы завершатся ошибкой при запуске скрипта, если он не запущен суперпользователем (или тем же пользователем, которому принадлежат все объекты в скрипте). Чтобы создать скрипт, который может быть восстановлен любым пользователем, но предоставит этому пользователю право владения всеми объектами, укажите -O. Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -s | --schema-only | Выгружает только определения объекта (схему), но не данные.  Этот параметр противоположен --data-only. Это похоже на указание --section=pre-data --section=post-data, но по историческим причинам не идентично ему.  (Не путайте его с параметром --schema, в котором слово «схема» используется в другом значении.)  Чтобы исключить данные таблицы только для подмножества таблиц в базе данных, см. --exclude-table-data. |
| -S *username* | --superuser=*username* | Указывает имя пользователя суперпользователя, которое будет использоваться при отключении триггеров. Это актуально, только если используется --disable-triggers. Лучше оставить это без внимания и вместо этого запустить полученный скрипт от имени суперпользователя.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| -t *table* | --table=*table* | Выгружает только таблицы (или представления, или последовательности, или сторонние таблицы), соответствующие шаблону таблицы. Укажите таблицу в формате schema.table.  Можно выбрать несколько таблиц, написав несколько ключей ‑t. Кроме того, параметр *table* интерпретируется как шаблон в соответствии с теми же правилами, которые используются командами psql\d, поэтому несколько таблиц также можно выбрать, записав в шаблон подстановочные символы. При использовании подстановочных символов будьте осторожны при необходимости заключать образец в кавычки, чтобы оболочка не расширяла подстановочные знаки. Ключи -n и -N не действуют, когда используется -t, потому что таблицы, выбранные -t, будут сброшены независимо от этих переключателей, а объекты, не являющиеся таблицами, не будут сброшены.  **Примечание**. Если указан -t, pg\_dump не пытается сбросить дамп любых других объектов базы данных, от которых может зависеть выбранная таблица (таблицы). Следовательно, нет гарантии, что результаты дампа конкретной таблицы могут быть успешно восстановлены сами по себе в чистую базу данных.  Кроме того, -t нельзя использовать для указания раздела дочерней таблицы. Чтобы создать дамп партиционированной таблицы, вы должны указать имя родительской таблицы. |
| -T *table* | --exclude-table=*table* | Не выгружает таблицы, соответствующие шаблону таблицы. Шаблон интерпретируется по тем же правилам, что и для -t. -T можно указывать более одного раза, чтобы исключить таблицы, соответствующие любому из нескольких шаблонов. Когда заданы оба параметра -t и -T, происходит сброс только тех таблиц, которые соответствуют хотя бы одному переключателю -t, но не соответствуют параметрам -T. Если -T отображается без -t, то таблицы, соответствующие -T, исключаются из обычного дампа. |
| -v | --verbose | Задаёт подробный режим. pg\_dump будет выводить подробные комментарии к объекту и время запуска/остановки в файл дампа, а также сообщения о ходе выполнения до стандартной ошибки. |
| -V | --version | Распечатывает версию pg\_dump и выходит. |
| -x | --no-privileges | --no-acl | Запрещает сброс прав доступа (команды GRANT/REVOKE). |
| -Z 0..9 | --compress=0..9 | Указывает используемый уровень сжатия. Ноль означает отсутствие сжатия. Для пользовательского формата архива параметр определяет сжатие отдельных табличных данных сегментов, и по умолчанию сжатие выполняется на умеренном уровне.  Для вывода обычного текста установка ненулевого уровня сжатия приводит к сжатию всего выходного файла, как если бы он был передан через gzip. Но по умолчанию сжатие не установлено. Формат tar-архива в настоящее время вообще не поддерживает сжатие. |
| --binary-upgrade | Этот параметр предназначен для использования утилит обновления in-place. Его использование для других целей не рекомендуется и не поддерживается. Поведение параметра может измениться в будущих выпусках без предварительного уведомления. |
| --column-inserts | --attribute-inserts | Выгружает данные как команды INSERT с явными именами столбцов (INSERT INTOtable (column, ...) VALUES ...). Это сделает восстановление очень медленным; в основном это полезно для создания дампов, которые могут быть загружены в базы данных, не основанные на PostgreSQL. Однако, поскольку этот параметр генерирует отдельную команду для каждой строки, ошибка при перезагрузке строки приводит к потере только этой строки, а не всего содержимого таблицы. |
| --disable-dollar-quotin | Этот параметр отключает использование долларовых кавычек для тел функций и заставляет их заключать в кавычки с использованием стандартного строкового синтаксиса SQL. |
| --disable-triggers | Этот параметр актуален только при создании дампа только с данными. Он предписывает pg\_dump включать команды для временного отключения триггеров в целевых таблицах на время перезагрузки данных. Используйте этот параметр, если у вас есть триггеры для таблиц, которые вы не хотите вызывать во время перезагрузки данных. Команды, выдаваемые для --disable-triggers, должны выполняться от имени суперпользователя. Таким образом, вы также должны указать имя суперпользователя с -S или, желательно, быть осторожным, чтобы запустить получившийся скрипт как суперпользователь. Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| --exclude-table-data=*table* | Не выгружает данные для любых таблиц, соответствующих шаблону таблицы. Шаблон интерпретируется по тем же правилам, что и для -t. --exclude-table-data можно указывать более одного раза, чтобы исключить таблицы, соответствующие любому из нескольких шаблонов. Этот параметр полезен, когда вам нужно определение конкретной таблицы, даже если вам не нужны данные в ней.  Чтобы исключить данные для всех таблиц в базе данных, см. --schema-only. |
| --if-exists | Использует условные команды (например, добавление предложения IF EXISTS) при очистке объектов базы данных. Этот параметр недействителен, если также не указан --clean. |
| --inserts | Выгружает данные как команды INSERT (а не COPY). Это сделает восстановление очень медленным; в основном это полезно для создания дампов, которые могут быть загружены в базы данных, не основанные на PostgreSQL. Однако, поскольку этот параметр генерирует отдельную команду для каждой строки, ошибка при перезагрузке строки приводит к потере только этой строки, а не всего содержимого таблицы. Обратите внимание, что восстановление может полностью завершиться неудачно, если вы изменили порядок столбцов. Параметр --column-inserts безопасен от изменений порядка столбцов, хотя и медленнее. |
| --lock-wait-timeout=*timeout* | Не ждать бесконечно, чтобы получить блокировки разделяемой таблицы в начале дампа. Вместо этого выполняет сбой, если не удалось заблокировать таблицу в течение указанного *timeout*. Указывает время ожидания в миллисекундах. |
| --no-security-labels | Не дампить защитные наклейки. |
| --no-synchronized-snapshots | Этот параметр позволяет запускать pg\_dump -j для сервера RT.Warehouse до 6.0; см. документацию по параметру -j для получения более подробной информации. |
| --no-tablespaces | Не выводиn команды для выбора табличных пространств. С этим параметром все объекты будут созданы в том табличном пространстве, которое используется по умолчанию во время восстановления.  Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| --no-unlogged-table-data | Не выгружает содержимое незарегистрированных таблиц. Этот параметр не влияет на то, сбрасываются ли определения таблиц (схемы); он только подавляет сброс данных таблицы. Данные в незарегистрированных таблицах всегда исключаются при выгрузке с резервного сервера. |
| --quote-all-identifiers | Принудительное цитирование всех идентификаторов. Этот параметр рекомендуется при выгрузке базы данных с сервера, основная версия RT.Warehouse которого отличается от версии pg\_dump, или когда выходные данные предназначены для загрузки на сервер другой основной версии. По умолчанию pg\_dump цитирует только идентификаторы, которые являются зарезервированными словами в его собственной основной версии. Иногда это приводит к проблемам совместимости при работе с серверами других версий, которые могут иметь несколько другие наборы зарезервированных слов. Использование --quote-all-identifiers предотвращает такие проблемы за счёт более трудного для чтения скрипта дампа. |
| --section=*sectionname* | Выгружает только названную партицию. Имя партиции может быть pre-data, data или post-data. Этот параметр можно указывать несколько раз, чтобы выбрать несколько партиций. По умолчанию дампятся все партиции.  Партиция data содержит фактические данные таблицы и значения последовательности. Элементы post-data включают определения индексов, триггеров, правил и ограничений, кроме проверенных ограничений. Элементы pre-data включают все другие элементы определения данных. |
| --serializable-deferrable | Использует сериализуемую транзакцию для дампа, чтобы гарантировать, что используемый снапшот совместим с более поздними состояниями базы данных; но делает это, дождавшись точки в потоке транзакций, в которой не может быть никаких аномалий, чтобы не было риска сбоя дампа или отката других транзакций с помощью serialization\_failure.  Этот вариант невыгоден для дампа, который предназначен только для аварийного восстановления. Он может быть полезнен для дампа, используемого для загрузки копии базы данных для отчётов или загрузки общей копии только для чтения, в то время как исходная база данных продолжает обновляться. Без этого дамп может отражать состояние, которое не согласуется с каким-либо последовательным выполнением транзакций, в конечном итоге совершённых. Например, если используются методы пакетной обработки, пакет может отображаться как закрытый в дампе, но не отображаются все элементы, входящие в пакет.  Этот параметр не будет иметь никакого значения, если при запуске pg\_dump нет активных транзакций чтения-записи. Если транзакции чтения-записи активны, начало дампа может быть отложено на неопределённый период времени. После запуска производительность с переключателем или без него такая же.  **Примечание**. Поскольку RT.Warehouse не поддерживает сериализуемые транзакции, параметр --serializable-deferrable не действует в RT.Warehouse. |
| --use-set-session-authorization | Выводит стандартные SQL-команды SET SESSION AUTHORIZATION вместо команд ALTER OWNER, чтобы определить владельца объекта. Это делает дамп более совместимым со стандартами, но в зависимости от истории объектов в дампе может не восстанавливаться должным образом. Дамп с использованием SET SESSION AUTHORIZATION потребует прав суперпользователя для правильного восстановления, тогда как ALTER OWNER требует меньших прав. |
| --gp-syntax | --no-gp-syntax | Используйте --gp-syntax для вывода синтаксиса RT.Warehouse в операторах CREATE TABLE. Это позволяет сбрасывать политику распределения (условия DISTRIBUTED BY или DISTRIBUTED RANDOMLY) таблицы RT.Warehouse, что полезно для восстановления в других системах RT.Warehouse. По умолчанию синтаксис RT.Warehouse включается при подключении к системе RT.Warehouse и исключается при подключении к обычной системе PostgreSQL. |
| --function-oids *oids* | Дамп функции (функций), указанной в списке идентификаторов объектов *oids*.  **Примечание**. Этот параметр предназначен исключительно для использования другими утилитами администрирования; его использование для каких-либо других целей не рекомендуется и не поддерживается. Поведение параметра может измениться в будущих выпусках без предварительного уведомления. |
| --relation-oids *oids* | Выгружает отношения, указанные в списке идентификаторов объектов *oids*.  **Примечание**. Этот параметр предназначен исключительно для использования другими утилитами администрирования; его использование для каких-либо других целей не рекомендуется и не поддерживается. Поведение опции может измениться в будущих выпусках без предварительного уведомления. |
| -? | --help | Показывает справку об аргументах командной строки pg\_dump и выходит. |

Таблица 64 — connection параметры pg\_dump

| Параметр | Описание |
| --- | --- |
| -d *dbname* | --dbname=*dbname* | Задаёт имя базы данных для подключения. Этот параметр эквивалентен указанию dbname в качестве первого аргумента командной строки, не являющегося параметром.  Если этот параметр содержит знак = или начинается с допустимого префикса URI (postgresql:// или postgres://), он рассматривается как строка conninfo. Для получения дополнительной информации см. [Строки подключения](https://www.postgresql.org/docs/9.4/libpq-connect.html#LIBPQ-CONNSTRING) в документации PostgreSQL. |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает сервер мастера базы данных RT.Warehouse. Если не указан, считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port=*port* | Порт TCP, на котором сервер мастера базы данных RT.Warehouse ожидает подключений. Если не указан, считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если не указан, считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -W | --password | Принудительно ввести пароль. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя для ввода пароля. |
| --role=*rolename* | Задаёт имя роли, которая будет использоваться для создания дампа. Этот параметр заставляет pg\_dump выдавать команду SET ROLE *rolename* после подключения к базе данных. Это полезно, когда аутентифицированный пользователь (указанный параметром -U) не имеет прав, необходимых для pg\_dump, но может переключиться на роль с необходимыми правами. Некоторые установки имеют политику, запрещающую вход непосредственно в систему как суперпользователь, и использование этого параметра позволяет делать дамп без нарушения политики. |

Примечания

Если выбран дамп только данных и используется параметр --disable-triggers, pg\_dump выдаёт команды для отключения триггеров в пользовательских таблицах перед вставкой данных и команды для их повторного включения после того, как данные были вставлены. Если восстановление остановлено посередине, системные каталоги могут остаться в неправильном состоянии.

Файл дампа, созданный pg\_dump, не содержит статистики, используемой оптимизатором для принятия решений по планированию запросов. Таким образом, после восстановления из файла дампа рекомендуется запустить ANALYZE, чтобы обеспечить оптимальную производительность.

Активность pg\_dump в базе данных обычно собирается сборщиком статистики. Если это нежелательно, вы можете установить для параметра track\_counts значение false с помощью PGOPTIONS или команды ALTER USER.

Поскольку pg\_dump может использоваться для передачи данных в более новые версии RT.Warehouse, можно ожидать, что вывод pg\_dump будет загружен в версии RT.Warehouse более новые, чем версия pg\_dump. pg\_dump также может выполнять дамп из версий RT.Warehouse, более старых, чем её собственная версия. Однако pg\_dump не может выполнять дамп из версий RT.Warehouse новее, чем её собственная основная версия; он откажется даже от попытки, а не рискнёт сделать недопустимый дамп. Кроме того, не гарантируется, что вывод pg\_dump может быть загружен на сервер более старой основной версии — даже если дамп был взят с сервера этой версии. Загрузка файла дампа на старый сервер может потребовать ручного редактирования файла дампа, чтобы удалить синтаксис, не понятый старому серверу. В случаях перекрёстных версий рекомендуется использовать параметр --quote-all-identifiers, так как он может предотвратить проблемы, возникающие из-за изменения списков зарезервированных слов в разных версиях RT.Warehouse.

Примеры

Выгрузить базу данных mydb в файл SQL-скрипта:

|  |
| --- |
| pg\_dump mydb > db.sql |

Чтобы перезагрузить такой скрипт в (только что созданную) базу данных с именем newdb:

|  |
| --- |
| psql -d newdb -f db.sql |

Сделать дамп RT.Warehouse в формате tar и включить информацию о политике распределения:

|  |
| --- |
| pg\_dump -Ft --gp-syntax mydb > db.tar |

Чтобы выгрузить базу данных в архивный файл произвольного формата:

|  |
| --- |
| pg\_dump -Fc mydb > db.dump |

Чтобы выгрузить базу данных в архив в формате каталога:

|  |
| --- |
| pg\_dump -Fd mydb -f dumpdir |

Чтобы выгрузить базу данных в архив формата каталога параллельно с 5 рабочими заданиями:

|  |
| --- |
| pg\_dump -Fd mydb -j 5 -f dumpdir |

Чтобы перезагрузить архивный файл в (только что созданную) базу данных с именем newdb:

|  |
| --- |
| pg\_restore -d newdb db.dump |

Чтобы выгрузить одну таблицу с именем mytab:

|  |
| --- |
| pg\_dump -t mytab mydb > db.sql |

Чтобы указать имя в верхнем или смешанном регистре в -t и связанных переключателях, вам нужно заключить имя в двойные кавычки; иначе он будет сложен в нижний регистр. Но двойные кавычки — это особенность оболочки, поэтому они, в свою очередь, должны быть заключены в кавычки. Таким образом, чтобы выгрузить одну таблицу со смешанным именем, вам понадобится что-то вроде:

|  |
| --- |
| pg\_dump -t '"MixedCaseName"' mydb > mytab.sql |

pg\_dumpall

Извлекает все базы данных в системе RT.Warehouse в один файл скрипта или другой архивный файл.

Синтаксис

|  |
| --- |
| pg\_dumpall [*connection-option* ...] [*dump\_option* ...]  pg\_dumpall -? | --help  pg\_dumpall -V | --version |

Описание

pg\_dumpall — это стандартная утилита PostgreSQL для резервного копирования всех баз данных в инстансе RT.Warehouse (или PostgreSQL), которая также поддерживается в RT.Warehouse. Она создаёт единый (непараллельный) файл дампа. Для регулярного резервного копирования RT.Warehouse лучше использовать утилиту резервного копирования RT.Warehouse gpbackup для достижения наилучшей производительности.

pg\_dumpall создаёт один файл скрипта, содержащий SQL-команды, которые можно использовать в качестве входных данных для psql для восстановления баз данных. Это делается путём вызова pg\_dump для каждой базы данных. pg\_dumpall также выгружает глобальные объекты, общие для всех баз данных (pg\_dump не сохраняет эти объекты). В настоящее время в них входит информация о пользователях и группах базы данных, а также разрешениях на доступ, которые применяются к базам данных в целом.

Поскольку pg\_dumpall считывает таблицы из всех баз данных, вам скорее всего придётся подключиться как суперпользователь базы данных, чтобы создать полный дамп. Также вам потребуются привилегии суперпользователя для выполнения сохраненного скрипта, чтобы иметь возможность добавлять пользователей и группы, а также создавать базы данных.

Скрипт SQL будет записан в стандартный вывод. Используйте [-f | --file] или операторы оболочки, чтобы перенаправить его в файл.

pg\_dumpall требуется несколько раз подключиться к серверу мастера RT.Warehouse (один раз для каждой базы данных). Если вы используете аутентификацию по паролю, вероятно, каждый раз будет запрашивать пароль. В таких случаях удобно иметь файл ~/.pgpass.

|  |
| --- |
| **Примечание.**  Параметр --ignore-version устарел и будет удалён в следующем выпуске. |

Параметры

Таблица — dump параметры pg\_dumpall

| Параметр | Описание |
| --- | --- |
| -a | --data-only | Выгружает только данные, а не схему (определения данных). Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -c | --clean | Выводит команды для очистки (удаления) объектов базы данных до (команд для) их создания. Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -f filename | --file=filename | Отправляет вывод в указанный файл. |
| -g | --globals-only | Дамп только глобальных объектов (ролей и табличных пространств), без баз данных. |
| -i | --ignore-version | **Примечание**. Этот параметр устарел и будет удалён в следующем выпуске.  Игнорирует несоответствие версий между pg\_dump и сервером базы данных. pg\_dump может выгружать данные с серверов, на которых запущены предыдущие версии RT.Warehouse (или PostgreSQL), но очень старые версии могут больше не поддерживаться. Используйте этот параметр, если вам нужно отменить проверку версии. |
| -o | --oids | Дамп идентификаторов объектов (OID) как часть данных для каждой таблицы. Использование этого параметра не рекомендуется для файлов, которые должны быть восстановлены в RT.Warehouse. |
| -O | --no-owner | Не выводит команды для установки владения объектами в соответствии с исходной базой данных. По умолчанию pg\_dump выдаёт операторы ALTER OWNER или SET SESSION AUTHORIZATION, чтобы установить владельца созданных объектов базы данных. Эти операторы завершатся ошибкой при запуске скрипта, если он не запущен суперпользователем (или тем же пользователем, которому принадлежат все объекты в скрипте). Чтобы создать скрипт, который может быть восстановлен любым пользователем, но предоставит этому пользователю право владения всеми объектами, укажите -O. Этот параметр имеет смысл только для обычного текстового формата. Для форматов архивов вы можете указать параметр при вызове pg\_restore. |
| -r | --roles-only | Дампит только роли, а не базы данных или табличные пространства. |
| -s | --schema-only | Выгружает только определения объекта (схему), но не данные. |
| -S *username* | --superuser=*username* | Указывает имя пользователя суперпользователя, которое будет использоваться при отключении триггеров. Это актуально, только если используется --disable-triggers. Лучше оставить это без внимания и вместо этого запустить полученный скрипт от имени суперпользователя.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| -t | --tablespaces-only | Дампит только табличные пространства, а не базы данных или роли. |
| -v | --verbose | Задаёт подробный режим. pg\_dump будет выводить подробные комментарии к объекту и время запуска/остановки в файл дампа, а также сообщения о ходе выполнения до стандартной ошибки. |
| -V | --version | Распечатывает версию pg\_dumpall и выходит. |
| -x | --no-privileges | --no-acl | Запрещает сброс прав доступа (команды GRANT/REVOKE). |
| --binary-upgrade | Этот параметр предназначен для использования утилит обновления in-place. Его использование для других целей не рекомендуется и не поддерживается. Поведение параметра может измениться в будущих выпусках без предварительного уведомления. |
| --column-inserts | --attribute-inserts | Выводит данные в виде команд INSERT с явными именами столбцов (INSERT INTO table (column, ...) VALUES ...). Это сделает восстановление очень медленным; в основном это полезно для создания дампов, которые могут быть загружены в базы данных, не основанные на PostgreSQL. Кроме того, поскольку этот параметр генерирует отдельную команду для каждой строки, ошибка при перезагрузке строки приводит к потере только этой строки, а не всего содержимого таблицы. |
| --disable-dollar-quoting | Этот параметр отключает использование долларовых кавычек для тел функций и заставляет их заключать в кавычки с использованием стандартного строкового синтаксиса SQL. |
| --disable-triggers | Этот параметр актуален только при создании дампа только с данными. Он указывает pg\_dumpall включать команды для временного отключения триггеров в целевых таблицах на время перезагрузки данных. Используйте его, если у вас есть триггеры для таблиц, которые вы не хотите вызывать во время перезагрузки данных. Команды, выдаваемые для --disable-triggers, должны выполняться от имени суперпользователя. Таким образом, вы также должны указать имя суперпользователя с -S или, желательно, быть осторожным, чтобы запустить получившийся скрипт как суперпользователь.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| --inserts | Выгружает данные как команды INSERT (а не COPY). Это сделает восстановление очень медленным; в основном это полезно для создания дампов, которые могут быть загружены в базы данных, не основанные на PostgreSQL. Кроме того, поскольку этот параметр генерирует отдельную команду для каждой строки, ошибка при перезагрузке строки приводит к потере только этой строки, а не всего содержимого таблицы. Обратите внимание, что восстановление может полностью завершиться неудачно, если вы изменили порядок столбцов. Параметр --column-inserts безопасен от изменений порядка столбцов, хотя и медленнее. |
| --lock-wait-timeout=*timeout* | Не ждать бесконечно, чтобы получить блокировки разделяемой таблицы в начале дампа. Вместо этого произойдёт сбой, если не удалось заблокировать таблицу в течение указанного времени ожидания. Тайм-аут может быть указан в любом из форматов, принимаемых SET statement\_timeout. Допустимые значения зависят от версии сервера, с которого выполняется дамп, но все версии RT.Warehouse принимают целое число миллисекунд. |
| --no-security-labels | Не сбрасывать защитные наклейки. |
| --no-tablespaces | Не выводите команды для выбора табличных пространств. С этой опцией все объекты будут созданы в том табличном пространстве, которое используется по умолчанию во время восстановления. |
| --no-unlogged-table-data | Не выгружайте содержимое незарегистрированных таблиц. Эта опция не влияет на то, сбрасываются ли определения таблиц (схемы); он только подавляет сброс данных таблицы. |
| --quote-all-identifiers | Принудительное цитирование всех идентификаторов. Этот вариант рекомендуется при выгрузке базы данных с сервера, основная версия RT.Warehouse которого отличается от версии pg\_dumpall, или когда выходные данные предназначены для загрузки на сервер другой основной версии. По умолчанию pg\_dumpall цитирует только те идентификаторы, которые являются зарезервированными словами в его собственной основной версии. Иногда это приводит к проблемам совместимости при работе с серверами других версий, которые могут иметь несколько другие наборы зарезервированных слов. Использование --quote-all-identifiers предотвращает такие проблемы за счёт более трудного для чтения сценария дампа. |
| --resource-queues | Дамп определений очереди ресурсов. |
| --resource-groups | Дамп определений групп ресурсов. |
| --use-set-session-authorization | Выведите стандартные команды SET SESSION AUTHORIZATION вместо команд ALTER OWNER, чтобы определить владельца объекта. Это делает дамп более совместимым со стандартами, но в зависимости от истории объектов в дампе может не восстанавливаться должным образом. Дамп с использованием SET SESSION AUTHORIZATION потребует прав суперпользователя для правильного восстановления, тогда как ALTER OWNER требует меньших прав. |
| --gp-syntax | Выведите синтаксис базы данных RT.Warehouse в операторах CREATE TABLE. Это позволяет сбрасывать политику распространения (условия DISTRIBUTED BY или DISTRIBUTED RANDOMLY) таблицы базы данных RT.Warehouse, что полезно для восстановления в других системах баз данных RT.Warehouse. |
| --no-gp-syntax | Не выводите предложения о распределении таблиц в операторах CREATE TABLE. |
| -? | --help | Показать справку об аргументах командной строки pg\_dumpall и выйти. |

Таблица 66 — connection параметры pg\_dump

| Параметр | Описание |
| --- | --- |
| -d *connstr* | --dbname=*connstr* | Задает параметры, используемые для подключения к серверу, в виде строки подключения. Для получения дополнительной информации см. Строки подключения в документации PostgreSQL.  Параметр называется --dbname для согласованности с другими клиентскими приложениями, но поскольку pg\_dumpall необходимо подключаться ко многим базам данных, имя базы данных в строке подключения будет проигнорировано. Используйте параметр -l, чтобы указать имя базы данных, используемой для создания дампа глобальных объектов, и узнать, какие другие базы данных следует сбросить. |
| -h *host* | --host=*host* | Имя хоста компьютера, на котором работает главный сервер базы данных RT.Warehouse. Если не указано, считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -l *dbname* | --database=*dbname* | Задает имя базы данных, к которой нужно подключиться для создания дампа глобальных объектов. Если не указано, используется база данных postgres. Если база данных postgres не существует, используется база данных template1. |
| -p *port* | --port=*port* | Порт TCP, на котором главный сервер базы данных RT.Warehouse прослушивает соединения. Если не указано, считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username= *username* | Имя роли базы данных для подключения. Если не указано, считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Никогда не запрашивайте пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими средствами, такими как файл .pgpass, попытка подключения не удастся. Эта опция может быть полезна в пакетных заданиях и сценариях, где нет пользователя для ввода пароля. |
| -W | --password | Принудительно ввести пароль. |
| --role=*rolename* | Задает имя роли, которая будет использоваться для создания дампа. Эта опция заставляет pg\_dumpall выдавать команду SET ROLE rolename после подключения к базе данных. Это полезно, когда аутентифицированный пользователь (указанный параметром -U) не имеет прав, необходимых для pg\_dumpall, но может переключиться на роль с необходимыми правами. Некоторые установки имеют политику, запрещающую вход непосредственно в систему как суперпользователь, и использование этой опции позволяет делать дамп без нарушения политики. |

Примечания

Поскольку pg\_dumpall вызывает внутри себя pg\_dump, некоторые диагностические сообщения будут относиться к pg\_dump.

После восстановления целесообразно запустить ANALYZE для каждой базы данных, чтобы у планировщика запросов была полезная статистика. Вы также можете запустить Vacuumdb -a -z для анализа всех баз данных.

pg\_dumpall требует, чтобы все необходимые каталоги табличных пространств существовали до восстановления; в противном случае создание базы данных не удастся для баз данных в местах, отличных от заданных по умолчанию.

Примеры

Чтобы выгрузить все базы данных:

|  |
| --- |
| pg\_dumpall > db.out |

Чтобы перезагрузить базы данных из этого файла, вы можете использовать:

|  |
| --- |
| psql template1 -f db.out |

Чтобы сбрасывать только глобальные объекты (включая очереди ресурсов):

|  |
| --- |
| pg\_dumpall -g --resource-queues |

pg\_restore

Восстанавливает базу данных из архивного файла, созданного pg\_dump.

Синтаксис

|  |
| --- |
| pg\_restore [*connection-option* ...] [*restore\_option* ...] *filename*  pg\_restore -? | --help  pg\_restore -V | --version |

Описание

pg\_restore — это утилита для восстановления базы данных из архива, созданного pg\_dump в любом формате кроме текстового. Она выполняет команды, необходимые для восстановления базы данных в том состоянии, в котором она находилась на момент сохранения. Файлы архивов позволяют pg\_restore выбирать, какие данные необходимо восстановить, или даже изменять порядок объектов перед восстановлением.

pg\_restore может работать в двух режимах. Если указано имя базы данных, содержимое архива восстанавливается непосредственно в базу данных. В противном случае создаётся скрипт с SQL-командами, необходимыми для восстановления базы данных, и записывается в файл или на стандартный вывод. Содержание созданного скрипта эквивалентно выводу pg\_dump в формате текста. Поэтому некоторые параметры, управляющие выводом, аналогичны параметрам pg\_dump.

pg\_restore не может восстановить информацию, которой нет в архивном файле. Например, если архив был создан с использованием параметра «выгрузить данные как команды INSERT», pg\_restore не сможет загрузить данные с помощью операторов COPY.

|  |
| --- |
| **Примечание.**  Параметр --ignore-version устарел и будет удалён в следующем выпуске. |

Параметры

Таблица 67 — filename параметры pg\_restore

| Параметр | Описание |
| --- | --- |
| filename | Указывает расположение файла архива (или каталога — для архива в формате каталога), который нужно восстановить. Если параметр не указан, используется стандартный ввод. |

Таблица 68 — restore параметры pg\_restore

| Параметр | Описание |
| --- | --- |
| -a | --data-only | Восстанавливает только данные, без схемы (определений данных). Данные таблицы и значения последовательности восстанавливаются, если они есть в архиве.  Этот параметр аналогичен указанию --section=data, но по историческим причинам не идентичен. |
| -c | --clean | Очищает (drop) объекты базы данных перед их восстановлением. (Это может привести к появлению некоторых безобидных сообщений об ошибках, если какие-либо объекты не присутствуют в целевой базе данных.) |
| -C | --create | Создаёт базу данных перед восстановлением в неё. Если также указан параметр --clean, удаляет и заново создаёт целевую базу данных перед подключением к ней.  Когда используется этот параметр, база данных, заданная параметром -d, используется только для выполнения начальных команд DROP DATABASE и CREATE DATABASE. Все данные восстанавливаются в базу данных, имя которой записано в архив. |
| -d *dbname* | --dbname=*dbname* | Подключается к заданной базе данных и производит восстановление данных непосредственно в эту базу данных. Эта утилита, как и большинство других утилит RT.Warehouse, также использует переменные среды, поддерживаемые libpq. Однако она не читает PGDATABASE, если имя базы данных не указано. |
| -e | --exit-on-error | Завершает работу, если при отправке SQL-команд в базу данных возникла ошибка. По умолчанию восстановление продолжается, а по его завершению отображается количество возникших ошибок. |
| -f *outfilename* | --file=*outfilename* | Задаёт выходной файл для сгенерированного скрипта или для списка объектов, полученного при использовании -l. По умолчанию используется стандартный вывод. |
| -F c|d|t | --format={custom | directory | tar} | Задаёт формат архива, созданного pg\_dump. Формат указывать не обязательно, поскольку pg\_restore определяет формат автоматически. Если формат задаётся, то используется одно из следующих значений:   * custom — архив сохранён в специальном формате pg\_dump; * directory — архив сохранён в формате каталога; * tar — архив сохранён в tar-формате. |
| -i | --ignore-version | **Примечание**. Этот параметр устарел и будет удалён в следующем выпуске.  Игнорирует проверки версии базы данных. |
| -I *index* | --index=*index* | Восстанавливает определение только заданного индекса. |
| -j | --number-of-jobs | --jobs=*number-of-jobs* | Запускает наиболее трудоёмкие операции pg\_restore (те, которые загружают данные, создают индексы или создают ограничения), используя несколько параллельных заданий. Этот вариант может значительно сократить время восстановления большой базы данных на сервере, работающем на многопроцессорной машине.  Каждое задание представляет собой отдельный процесс или поток, в зависимости от операционной системы, и использует отдельное соединение с сервером.  Оптимальное значение для этого параметра зависит от аппаратной конфигурации сервера, клиента и сети. Также имеют значение количество процессорных ядер и конфигурацию дискового хранилища. В качестве начального значения можно выбрать количество процессорных ядер на сервере, а при увеличении данного значения восстановление может происходить быстрее. Слишком высокие значения приведут к снижению производительности из-за перегрузки.  Этот параметр поддерживает только с архивом в формате custom. Входной файл должен быть обычным файлом (не, например, каналом ввода). Этот параметр игнорируется при генерации скрипта (т.е. когда отсутствует прямое подключение к серверу базы данных). Кроме того, несколько заданий нельзя использовать вместе с параметром --single-transaction. |
| -l | --list | Выводит содержимое архива. Выходные данные этой операции можно использовать в качестве ввода для параметра -L для ограничения и изменения порядка восстанавливаемых объектов. |
| -L *list-file* | --use-list=*list-file* | Восстанавливает только объекты, которые перечислены в *list-file* и в том порядке, в котором они представлены в файле. Обратите внимание, что если параметры фильтрации, такие как -n или -t, используются с -L, они дополнительно ограничивают восстанавливаемые объекты.  *list-file* обычно представляет собой отредактированный результат вывода предыдущей операции -l. Строки файла можно перемещать или удалять, а также можно закомментировать, поставив точку с запятой (;) в начале строки. |
| -n *schema* | --schema=*schema* | Восстанавливает только те объекты, которые находятся в указанной схеме. Этот параметр можно комбинировать с параметром -t для восстановления только определённой таблицы. |
| -O | --no-owner | Не генерирует команды, восстанавливающие владение объектами в соответствии с исходной базой данных. По умолчанию pg\_restore генерирует команды ALTER OWNER или SET SESSION AUTHORIZATION, чтобы восстановить исходных владельцев создаваемых объектов схемы. Эти команды завершатся ошибкой, если начальное соединение с базой данных не будет выполнено суперпользователем (или пользователем, которому принадлежат все объекты в скрипте). Чтобы получить скрипт, который сможет восстановить любой подключающийся пользователь (но при этом он станет владельцем всех созданных объектов), используется -O. |
| -P '*function-name*(*argtype* [, ...])' | --function='*function-name*(*argtype* [, ...])' | Восстановливает только указанную функцию. Имя функции должно быть заключено в кавычки. Важно указать имя функции и аргументы точно так, как они записаны в оглавлении файла дампа (как показано параметром --list). |
| -s | --schema-only | Восстанавливает только схему (определения данных), без данных, в том объёме, в котором схемы присутствуют в архиве.  Этот параметр противоположен по действию параметру --data-only. Этот параметр похож на указание --section=pre-data --section=post-data, но по историческим причинам не идентичен ему.  (Не путайте этот параметр с параметром --schema, в котором слово «схема» используется в другом значении.) |
| -S *username* | --superuser=*username* | Задаёт имя суперпользователя, которое будет использоваться для отключения триггеров. Использование этого параметра актуально, только если используется --disable-triggers.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| -t *table* | --table=*table* | Восстанавливает определение и/или данные только указанной таблицы. Несколько таблиц могут быть указаны с помощью нескольких ключей -t. Этот параметр можно комбинировать с параметром -n, чтобы указать схему. |
| -T *trigger* | --trigger=*trigger* | Восстановливает только указанный триггер.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| -v | --verbose | Включает режим подробных сообщений. |
| -V | --version | Выводит версию pg\_restore и завершает работу. |
| -x | --no-privileges | --no-acl | Не восстанавливает права доступа (т.е. не выполняет команды GRANT/REVOKE). |
| -1 | --single-transaction | Выполняет восстановление в рамках одной транзакции. Этот параметр подразумевает, что либо все команды будут выполнены успешно, либо никакие изменения не будут применены. |
| --disable-triggers | Этот параметр актуален только при выполнении восстановления только данных. При его использовании pg\_restore выполняет команды для отключения триггеров в целевых таблицах на время загрузки данных. Используйте этот параметр, если у вас имеются триггеры для таблиц, которые вы не хотите вызывать во время загрузки данных. Команды, генерируемые --disable-triggers, должны выполняться от имени суперпользователя. Поэтому вы также должны указать имя суперпользователя с помощью параметра -S или, что предпочтительней, запустить pg\_restore как суперпользователь.  **Примечание**. RT.Warehouse не поддерживает пользовательские триггеры. |
| --no-data-for-failed-tables | По умолчанию данные таблицы восстанавливаются, даже если команда создания таблицы завершилась с ошибкой (например, потому что таблица уже существует). При выборе этого параметра данные для такой таблицы не восстанавливаются. Такое поведение полезно, когда целевая таблица уже содержит необходимые данные. Указание этого параметра предотвращает загрузку повторяющихся или устаревших данных.  Этот параметр эффективен только при восстановлении непосредственно в базу данных, но не при создании вывода в формате SQL-скрипта. |
| --no-security-labels | Не генерирует команды, восстанавливающие метки безопасности, даже если они содержатся в архиве. |
| --no-tablespaces | Не генерирует команды для указания табличных пространств. С данным параметром все объекты будут созданы в том табличном пространстве, которое используется по умолчанию во время восстановления. |
| --section=*sectionname* | Восстановливает только указанный раздел. В качестве *sectionname* можно указать следующие значения:   * data — собственно данные таблиц и определения больших объектов; * post-data — определения индексов, триггеров, правил и ограничений (кроме отдельно проверяемых); * pre-data — остальные определения.   Этот параметр можно указывать несколько раз, чтобы выбрать несколько разделов.  По умолчанию восстанавливаются все разделы. |
| --use-set-session-authorization | Генерирует стандартизированные команды SET SESSION AUTHORIZATION вместо команд ALTER OWNER, чтобы назначить владельца объекта. Использование этого параметра делает дамп более стандартизированным, но в может не восстановиться корректно в зависимости от истории объектов в дампе. |
| -? | --help | Показывает справку об аргументах командной строки pg\_restore и завершается. |

Таблица 69 — connection параметры pg\_restore

| Параметр | Описание |
| --- | --- |
| -h *host* | --host *host* | Имя хоста машины, на которо работает сервер мастера RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost. |
| -p *port* | --port *port* | TCP-порт, на котором сервер мастера RT.Warehouse принимает подключения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию используется 5432. |
| -U *username* | --username *username* | Имя системной роли, используемое для подключения к базе данных. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -w | --no-password | Не выводить запрос на ввод пароля. Если сервер требует аутентификацию по паролю, а пароль недоступен с помощью других средств, таких как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который вводит пароль. |
| -W | --password | Принудительно запрашивать пароль перед подключением к базе данных. |
| --role=*rolename* | Задаёт имя роли, которая будет использоваться для восстановления. При использовании этого параметра pg\_restore будет генерировать команду SET ROLE *rolename* после подключения к базе данных. Это полезно, когда аутентифицированный пользователь (указанный параметром -U) не имеет прав, необходимых для pg\_restore, но может переключиться на роль с необходимыми правами. Некоторые установки имеют политику, запрещающую вход непосредственно в систему как суперпользователь, и использование этого параметра позволяет выполнять восстановление без нарушения политики. |

Примечания

Если в вашей установке база данных template1 имеет какие-либо локальные дополнения, убедитесь, что pg\_restore осуществляет вывод в действительно пустую базу данных; в противном случае вы, вероятно, получите ошибки из-за повторяющихся определений добавленных объектов. Чтобы создать пустую базу данных без каких-либо локальных дополнений, используйте template0, а не template1, например:

|  |
| --- |
| CREATE DATABASE foo WITH TEMPLATE template0; |

При восстановлении данных в уже существующие таблицы и использовании параметра --disable-triggers, pg\_restore генерирует команды для отключения триггеров в пользовательских таблицах перед вставкой данных, а затем генерирует команды для их повторного включения после того, как данные были вставлены. Если восстановление прервётся в процессе, системные каталоги могут остаться в некорректном состоянии.

См. также pg\_dump для получения подробной информации об её ограничениях.

После восстановления целесообразно запустить ANALYZE для каждой восстановленной таблицы, чтобы планировщик запросов получил актуальную статистику.

Примеры

Предположим, мы выгрузили базу данных с именем mydb в файл дампа формата custom:

|  |
| --- |
| pg\_dump -Fc mydb > db.dump |

Чтобы удалить базу данных и воссоздать её из дампа:

|  |
| --- |
| dropdb mydb  pg\_restore -C -d template1 db.dump |

Чтобы перезагрузить дамп в новую базу данных с именем newdb. Обратите внимание, что параметр -C не используется в примере, вместо этого мы подключаемся напрямую к базе данных, в которую нужно восстановить. Также обратите внимание, что мы клонируем новую базу данных из template0, а не из template1, чтобы убедиться, что она изначально пуста:

|  |
| --- |
| createdb -T template0 newdb  pg\_restore -d newdb db.dump |

Чтобы изменить порядок объектов базы данных, сначала необходимо выгрузить оглавление архива:

|  |
| --- |
| pg\_restore -l db.dump > db.list |

Файл оглавления состоит из заголовка и одной строки для каждого элемента, например,

|  |
| --- |
| ; Archive created at Mon Sep 14 13:55:39 2009  ; dbname: DBDEMOS  ; TOC Entries: 81  ; Compression: 9  ; Dump Version: 1.10-0  ; Format: CUSTOM  ; Integer: 4 bytes  ; Offset: 8 bytes  ; Dumped from database version: 8.3.5  ; Dumped by pg\_dump version: 8.3.8  ;  ; Selected TOC Entries:  ;  3; 2615 2200 SCHEMA - public pasha  1861; 0 0 COMMENT - SCHEMA public pasha  1862; 0 0 ACL - public pasha  317; 1247 17715 TYPE public composite pasha  319; 1247 25899 DOMAIN public domain0 pasha2 |

Комментарии начинаются с точки с запятой (;), а числа в начале строк обозначают внутренний идентификатор архива, присвоенный каждому элементу архива. Строки в файле можно закомментировать, удалить и изменить порядок. Например:

|  |
| --- |
| 10; 145433 TABLE map\_resolutions postgres  ;2; 145344 TABLE species postgres  ;4; 145359 TABLE nt\_header postgres  6; 145402 TABLE species\_records postgres  ;8; 145416 TABLE ss\_old postgres |

Таким образом данный файл может использоваться в качестве входных данных для pg\_restore для восстановления только элементов 10 и 6 в указанном порядке:

|  |
| --- |
| pg\_restore -L db.list db.dump |

pgbouncer

Управляет пулами соединений с базой данных.

Синтаксис

|  |
| --- |
| pgbouncer [OPTION ...] pgbouncer.ini  OPTION  [ -d | --daemon ]  [ -R | --restart ]  [ -q | --quiet ]  [ -v | --verbose ]  [ {-u | --user}=*username* ]  pgbouncer [ -V | --version ] | [ -h | --help ] |

Описание

PgBouncer — это менеджер пула соединений для RT.Warehouse и PostgreSQL. PgBouncer поддерживает пул соединений для каждого пользователя базы данных и комбинации баз данных. PgBouncer либо создаёт новое соединение с базой данных для клиента, либо повторно использует имеющееся соединение для того же пользователя и базы данных. Когда клиент отключается, PgBouncer возвращает соединение в пул для повторного использования.

PgBouncer поддерживает стандартный интерфейс подключения, используемый PostgreSQL и RT.Warehouse. Клиентское приложение RT.Warehouse (например, psql) должно подключиться к хосту и порту, на котором работает PgBouncer, а не напрямую к хосту и порту мастера RT.Warehouse.

Настройка PgBouncer и его доступ к RT.Warehouse осуществляется через файл конфигурации. При запуске команды pgbouncer необходимо указать имя файла конфигурации, обычно pgbouncer.ini. Этот файл содержит информацию о местонахождении RT.Warehouse. Файл pgbouncer.ini также определяет процесс, пул соединений, авторизованных пользователей и конфигурацию аутентификации для PgBouncer, помимо прочих параметров конфигурации.

По умолчанию процесс pgbouncer выполняется как приоритетный процесс. При желании вы можете запустить pgbouncer как фоновый (демон) процесс, используя параметр -d.

Процесс pgbouncer принадлежит пользователю операционной системы, который запускает этот процесс. При желании вы можете указать другое имя пользователя, под которым будет запускаться pgbouncer.

PgBouncer включает административную консоль, подобную psql. Авторизованные пользователи могут подключаться к виртуальной базе данных для мониторинга и управления PgBouncer. Вы можете управлять процессом демона PgBouncer через административную консоль. Вы также можете использовать консоль для обновления и перезагрузки конфигурации PgBouncer во время его работы без остановки и перезапуска процесса.

Для получения дополнительной информации о PgBouncer обратитесь к [PgBouncer FAQ](https://pgbouncer.github.io/faq.html).

Параметры

Таблица — Параметры pgbouncer

| Параметр | Описание |
| --- | --- |
| -d | --daemon | Запускает PgBouncer как демон (фоновый процесс). По умолчанию PgBouncer запускается как процесс переднего плана.  При запуске в качестве демона PgBouncer отображает сообщения запуска на стандартный вывод. Чтобы данные сообщения не отображались, используйте параметр -q при запуске PgBouncer.  Чтобы остановить процесс PgBouncer, который был запущен как демон, введите команду SHUTDOWN из административной консоли PgBouncer. |
| -R | --restart | Перезапускает PgBouncer, используя указанные аргументы командной строки. Соединения с базами данных (кроме TLS) поддерживаются во время перезапуска. Соединения TLS не поддерживаются, они сбрасываются.  Чтобы перезапустить PgBouncer как демон, укажите параметры -Rd.  **Примечание**. Перезагрузка возможна только в том случае, если операционная система поддерживает сокеты Unix и конфигурация unix\_socket\_dir PgBouncer не отключена. |
| -q | --quiet | Тихий режим. Не выводит сообщения в stdout. |
| -v | --verbose | Увеличивает уровень детализации сообщений. Параметр можно указывать несколько раз. |
| {-u | --user}=*username* | Переключается на указанного пользователя при запуске процесса PgBouncer. |
| -V | --version | Выводит версию и завершается. |
| -h | --help | Выводит справку по команде и завершается. |

pgbouncer.ini

Файл конфигурации PgBouncer.

Синтаксис

|  |
| --- |
| [databases]  db = ...  [pgbouncer]  ...  [users]  ... |

Описание

В файле конфигурации pgbouncer.ini указываются параметры конфигурации PgBouncer и определяются параметры конфигурации для конкретного пользователя.

Файл конфигурации PgBouncer (обычно называется pgbouncer.ini) имеет формат .ini. Файлы в формате .ini состоят из разделов, параметров и значений. Имена разделов заключаются в квадратные скобки, например, [section\_name]. Параметры и значения указываются в формате *key*=*value*. Строки, начинающиеся с точки с запятой (;) или знака решётки (#), считаются строками комментариев и игнорируются.

Файл конфигурации PgBouncer может содержать директивы %include, которые определяют, что необходимо прочитать и обработать дополнительный файл конфигурации. Директивы включения позволяет разделить файл конфигурации на отдельные части. Например:

|  |
| --- |
| %include *filename* |

Если указанный файл не задаётся абсолютным путём, его расположение воспринимается относительно текущего рабочего каталога.

Файл конфигурации PgBouncer включает следующие разделы, подробно описанные ниже:

1. Раздел [databases].

Раздел [pgbouncer].

Раздел [users].

Раздел [databases]

Раздел [databases] содержит пары *key*=*value*, где *key* — это имя базы данных, а *value* — это строка подключения для libpq в виде пары *key*=*value*.

Имя базы данных может содержать символы [0-9A-Za-z\_.-] без кавычек. Имена, содержащие другие символы, должны быть заключены в стандартные кавычки идентификаторов SQL:

Имена заключены в двойные кавычки ("").

Двойные кавычки в идентификаторе представлены двумя последовательными двойными кавычками.

Имя базы данных \* — это резервная база данных. PgBouncer использует значение этого ключа как строку подключения для запрошенной базы данных. Автоматически созданные записи базы данных, подобные этим, очищаются, если они остаются бездействующими дольше времени, указанного в параметре autodb\_idle\_timeout.

Параметры подключения к базе данных

Следующие параметры могут быть включены в *value*, чтобы указать расположение базы данных.

Таблица — Параметры подключения к базе данных

| Параметр | Описание |
| --- | --- |
| dbname | Имя целевой базы данных.  По умолчанию — указанное клиентом имя базы данных. |
| host | Имя или IP-адрес хоста мастера RT.Warehouse. Имена хостов разрешаются в момент подключения. Если DNS возвращает несколько результатов, они используются циклически. Результат DNS кэшируется, а параметр dns\_max\_ttl определяет, когда истекает срок действия записи в кэше.  По умолчанию — не установлено, соединение осуществляется через сокет Unix. |
| port | Порт мастера RT.Warehouse.  По умолчанию — 5432 |
| user, password | Если задан user=, все подключения к целевой базе данных инициируются от имени указанного пользователя, в результате чего создается один пул подключений для базы данных.  Если параметр user= не установлен, PgBouncer пытается войти в целевую базу данных с именем пользователя, переданным клиентом. В этой ситуации будет отдельный пул для каждого пользователя, который подключается к базе данных. |
| auth\_user | Если указан параметр auth\_user, любой пользователь, который не указан в auth\_file, аутентифицируется путём запроса к представлению базы данных pg\_shadow. PgBouncer выполняет этот запрос как пользователь auth\_user базы данных RT.Warehouse. Пароль auth\_user должен быть установлен в auth\_file. |
| client\_encoding | Запрашивает у сервера использование указанной клиентской кодировки (client\_encoding). |
| datestyle | Запрашивает у сервера использование указанного стиля даты (datestyle). |
| timezone | Запрашивает у сервера использование указанного часового пояса (timezone). |

Параметры конфигурации пула

Используйте следующие параметры для конфигурации пула для конкретной базы данных.

Таблица — Параметры конфигурации пула

| Параметр | Описание |
| --- | --- |
| pool\_size | Устанавливает максимальный размер пулов для этой базы данных. Если параметр не установлен, используется значение default\_pool\_size. |
| connect\_query | Запрос, который должен быть выполнен после установления соединения, но перед тем, как разрешить использование соединения любым клиентам. Если запрос вызывает ошибки, они регистрируются, но в противном случае игнорируются. |
| pool\_mode | Устанавливает режим пула для этой базы данных. Если параметр не установлен, используется режим pool\_mode по умолчанию. |
| max\_db\_connections | Устанавливает максимальное количество соединений PgBouncer для этой базы данных. Общее количество подключений для всех пулов для этой базы данных не будет превышать это значение. |

Раздел [pgbouncer]

Параметры общей настройки

Таблица — Параметры общей настройки

| Параметр | Описание |
| --- | --- |
| logfile | Расположение файла журнала. Файл журнала остаётся открытым. После ротации журнала выполните kill -HUP pgbouncer или запустите RELOAD; в административной консоли PgBouncer.  По умолчанию — не задан. |
| pidfile | Имя pid-файла. Без указания pidfile вы не можете запустить PgBouncer как фоновый (демон) процесс.  По умолчанию — не задан. |
| listen\_addr | Список адресов интерфейсов, на которых PgBouncer прослушивает TCP-соединения. Вы также можете указать \*, что означает прослушивание всех интерфейсов. Если параметр не установлен, разрешены только соединения через сокеты Unix.  Указывать адреса можно числами (IPv4/IPv6) или по имени.  По умолчанию — не задан. |
| listen\_port | Порт, который слушает PgBouncer. Задаётся для сокетов TCP и для сокетов Unix.  По умолчанию — 6432. |
| unix\_socket\_dir | Указывает расположение сокетов Unix. Применяется как к прослушивающим сокетам, так и к подлючениям к серверам. Если задана пустая строка, сокеты Unix отключены. Требуется для онлайн-перезагрузки (параметр -R).  По умолчанию — /tmp. |
| unix\_socket\_mode | Режим файловой системы для сокета Unix.  По умолчанию — 0777. |
| unix\_socket\_group | Имя группы для сокета Unix.  По умолчанию — не задан. |
| user | Если параметр указан, указывает пользователя Unix, на которого необходимо переключиться после запуска. Этот параметр работает, только если PgBouncer запускается от имени root или PgBouncer уже запущен от имени указанного пользователя.  По умолчанию — не задан. |
| auth\_file | Имя файла, содержащего имена пользователей и пароли для загрузки. Формат файла такой же, как у файла pg\_auth RT.Warehouse.  По умолчанию — не задан. |
| auth\_hba\_file | Файл конфигурации HBA, который используется в том случае, когда auth\_type равен hba. См. [формат файла HBA](https://pgbouncer.github.io/config.html#hba-file-format) в документации PgBouncer для получения информации о поддержке PgBouncer формата файла аутентификации HBA.  По умолчанию — не задан. |
| auth\_type | Определяет, как аутентифицировать пользователей:   * pam — использует PAM (Pluggable Authentication Modules, Подключаемые модули аутентификации) для аутентификации пользователей. Файл auth\_file игнорируется. Этот метод несовместим с базами данных, использующими параметр auth\_user. Именем службы, сообщаемым для PAM, является pgbouncer. PAM не поддерживается в файле конфигурации HBA. * hba — фактический тип аутентификации загружается из auth\_hba\_file. Этот параметр позволяет разным методам аутентификации использовать разные пути доступа. * cert — клиенты должны подключаться по соединению TLS, используя действующий сертификат клиента. Имя пользователя клиента берется из поля CommonName в сертификате. * md5 — использует проверку пароля на основе MD5. auth\_file может содержать пароли как с шифрованием MD5, так и пароли в виде обычного текста (открытые). Это метод аутентификации является методом по умолчанию. * plain — пароль в открытом виде отправляется по сети. Метод не рекомендуется к использованию. * trust — аутентификация не выполняется. Но имя пользователя должно присутствовать в auth\_file. * any — подобен методу trust, но указанное имя пользователя игнорируется. Требует, чтобы для всех баз данных было настроено подключение заданного пользователя. Кроме того, база данных консоли позволяет любому пользователю войти в систему как администратор. |
| auth\_query | Запрос на извлечение пароля пользователя из базы данных. Если пользователь не существует в auth\_file и запись базы данных включает auth\_user, этот запрос запускается в базе данных как auth\_user для поиска пользователя.  Обратите внимание, что запрос выполняется внутри целевой базы данных, поэтому, если функция используется, её необходимо установить в каждую базу данных.  По умолчанию — SELECT usename, passwd FROM pg\_shadow WHERE usename=$1. |
| auth\_user | Если параметр auth\_user указан, любой пользователь, не указанный в auth\_file, может аутентифицироваться с помощью запроса auth\_query из представления базы данных pg\_shadow. PgBouncer выполняет этот запрос как пользователь RT.Warehouse auth\_user. Пароль auth\_user должен быть установлен в auth\_file.  Для прямого доступа к pg\_shadow требуются права администратора RT.Warehouse. Предпочтительно, чтобы пользователь, не имеющий прав администратора, вместо этого вызывал функцию SECURITY DEFINER. |
| pool\_mode | Указывает, когда соединение с сервером может быть повторно использовано другими клиентами:   * session — подключение возвращается в пул, когда клиент отключается. По умолчанию. * transaction — после завершения транзакции соединение возвращается в пул. * statement — соединение возвращается в пул после завершения текущего запроса. В этом режиме запрещены длинные транзакции с несколькими операторами. |
| max\_client\_conn | Максимальное разрешённое количество клиентских подключений. При увеличении следует также увеличить пределы дескрипторов файлов. Фактическое количество используемых файловых дескрипторов будет превышать max\_client\_conn. Теоретический максимум, используемый, когда каждый пользователь подключается к серверу под своим именем пользователя, составляет:   |  | | --- | | max\_client\_conn + (max\_pool\_size \* total\_databases \* total\_users) |   Если пользователь базы данных указан в строке подключения, то все пользователи подключаются под одним и тем же именем пользователя. Тогда теоретический максимум подключений составляет:   |  | | --- | | max\_client\_conn + (max\_pool\_size \* total\_databases) |   Теоретический максимум никогда не должен быть достигнут, если только кто-то специально не создаёт для него нагрузку. Тем не менее, это означает, что вы должны установить для количества файловых дескрипторов безопасное высокое значение. Найдите ulimit в документации по вашей операционной системе.  По умолчанию — 100. |
| default\_pool\_size | Возможное число подключений к серверу для пары пользователь/база данных. Значение можно переопределить в конфигурации для каждой базы данных.  По умолчанию — 20. |
| min\_pool\_size | Добавляет больше серверных подключений к пулу, если оно меньше этого числа. Это улучшает поведение, когда обычная нагрузка падает, а затем внезапно повышается после периода полного бездействия.  По умолчанию — 0 (отключено). |
| reserve\_pool\_size | Количество дополнительных подключений для пула. При значении 0 резерв отсутствует.  По умолчанию — 0 (отключено). |
| reserve\_pool\_timeout | Если клиент не обслуживается в течение указанного количества секунд, PgBouncer разрешает использование дополнительных подключений из резервного пула. Значение 0 отключает эту функцию.  По умолчанию — 5.0. |
| max\_db\_connections | Максимальное количество подключений к базе данных. Когда достигается заданный предел, закрытие клиентского подключения в одном пуле не позволяет сразу установить подключение к серверу для другого пула, потому что подключение к серверу для первого пула все ещё открыто. После закрытия соединения с сервером (по тайм-ауту неактивности) новое соединение с сервером будет открыто для ожидающего пула.  По умолчанию — unlimited. |
| max\_user\_connections | Максимальное количество подключений на пользователя. Когда достигается заданный предел, закрытие клиентского соединения с одним пулом не позволяет сразу установить соединение для другого пула, потому что соединение для первого пула все ещё открыто. После закрытия соединения для первого пула (по тайм-ауту неактивности) новое серверное соединение будет открыто для ожидающего пула. |
| server\_round\_robin | По умолчанию PgBouncer повторно использует подключения к серверу в порядке LIFO (last-in, first-out), так что наибольшую нагрузку получают несколько последних соединений. Это обеспечивает наилучшую производительность, когда базу данных обслуживает один сервер. Но если за IP-адресом базы данных имеется балансировщик TCP-соединений, то лучше, если PgBouncer также будет использовать соединения таким образом для достижения равномерной нагрузки.  По умолчанию — 0. |
| ignore\_startup\_parameters | По умолчанию PgBouncer принимает только параметры, которые он может отслеживать в стартовых пакетах: client\_encoding, datestyle, timezone и standard\_conforming\_strings.  Все остальные параметры вызывают ошибку. Чтобы принимались другие параметры, укажите их здесь, чтобы PgBouncer мог их игнорировать.  По умолчанию — empty. |
| disable\_pqexec | Отключает протокол простых запросов (PQexec, Simple Query protocol). В отличие от протокола расширенных запросов (Extended Query protocol), протокол простых запросов позволяет использовать несколько запросов в одном пакете, что позволяет использовать некоторые классы атак с использованием SQL-инъекций. Отключение протокола может повысить безопасность. Это означает, что будут иметь возможность работать только клиенты, которые используют исключительно протокол расширенных запросов.  По умолчанию — 0. |
| application\_name\_add\_host | Добавляе адрес хоста клиента и его порт в параметр имени приложения, задаваемый при установлении соединения. Это помогает определить источник некорректных запросов. Параметр перезаписывается без обнаружения, если приложение выполняет команду SET application\_name после подключения.  По умолчанию — 0. |
| conffile | Показывает расположение текущего файла конфигурации. Изменение этого параметра приведёт к тому, что PgBouncer будет использовать другой файл конфигурации для следующих команд RELOAD/SIGHUP.  По умолчанию — файл из командной строки |
| service\_name | Используется при регистрации службы win32.  По умолчанию — pgbouncer. |
| job\_name | Алиас для service\_name. |

Параметры журнала

Таблица — Параметры журнала

| Параметр | Описание |
| --- | --- |
| syslog | Включает и выключает запись в syslog.  По умолчанию — 0. |
| syslog\_ident | Имя, с которым события передаются в syslog.  По умолчанию — pgbouncer. |
| syslog\_facility | Субъект, который будет указываться в событиях, отправляемых в syslog. Возможные значения: auth, authpriv, daemon, user, local0-7.  По умолчанию — daemon. |
| log\_connections | Зарегистрирует в журнале успешные входы в систему.  По умолчанию — 1. |
| log\_disconnections | Регистрирует в журнале отключения с указанием причин.  По умолчанию — 1. |
| log\_pooler\_errors | Регистрирует сообщения об ошибках, которые пул отправляет клиентам.  По умолчанию — 1. |
| stats\_period | Указывает интервал записи агрегированной статистики в журнал.  По умолчанию — 60. |

Параметры управления доступом к консоли

Таблица — Параметры управления доступом к консоли

| Параметр | Описание |
| --- | --- |
| admin\_users | Разделённый запятыми список пользователей базы данных, которым разрешено подключаться и выполнять все команды в консоли администрирования PgBouncer. Игнорируется, когда auth\_type=any, и в этом случае любой пользователь может подключаться в качестве администратора.  По умолчанию — empty. |
| stats\_users | Разделённый запятыми список пользователей базы данных, которым разрешено подключаться и выполнять запросы только на чтения на консоли. Сюда входят все команды SHOW, кроме SHOW FDS.  По умолчанию — empty. |

Параметры проверки активности соединений, таймаутов

Таблица — Параметры проверки активности соединений, таймаутов

| Параметр | Описание |
| --- | --- |
| server\_reset\_query | Запрос, направляемый на сервер при освобождении соединения перед тем, как оно станет доступным для других клиентов. В этот момент никакая транзакция не выполняется, поэтому запрос не должен включать команды ABORT или ROLLBACK.  Запрос должен очистить все изменения, внесённые в сеанс базы данных, чтобы следующий клиент получил соединение в чётко определённом состоянии. По умолчанию установленная команда DISCARD ALL очищает все, но не оставляет следующему клиенту предварительно кэшированного состояния.  Когда используется пул транзакций, server\_reset\_query должен быть пустым, поскольку клиенты не должны использовать какие-либо свойства сеанса. Если клиенты действительно используют свойства сеанса, они будут прерваны, потому что пул транзакций не гарантирует, что следующий запрос будет выполняться в том же соединении.  По умолчанию — DISCARD ALL; |
| server\_reset\_query\_always | Определяет, следует ли запускать server\_reset\_query во всех режимах пула. Когда этот параметр выключен (по умолчанию), server\_reset\_query будет запускаться только в пулах, которые находятся в режиме пула сеансов. Соединения в режиме пула транзакций не должны требовать запроса сброса состояния.  По умолчанию — 0. |
| server\_check\_delay | Определяет, как долго сохранять освобожденные соединения доступными для повторного использования без выполнения запросов проверки работоспособности. Если 0, то запрос выполняется всегда.  По умолчанию — 30.0. |
| server\_check\_query | Простой бездействующий запрос для проверки сохранения соединения с сервером.  Если строка пуста, проверка соединения отключена.  По умолчанию — SELECT 1; |
| server\_lifetime | pgbouncer будет закрывать неиспользуемое серверное подключение, существующее дольше заданного времени (в секундах). Значение 0 означает, что соединение будет использоваться только один раз, а затем будет закрыто.  По умолчанию — 3600.0. |
| server\_idle\_timeout | Если соединение с сервером простаивает больше указанного количества секунд, оно разрывается. Если этот параметр установлен на 0, тайм-аут отключён.  По умолчанию — 600.0. |
| server\_connect\_timeout | Если соединение и вход в систему не завершатся за указанное количество секунд, соединение будет закрыто.  По умолчанию — 15.0. |
| server\_login\_retry | Если вход в систему завершается неудачно из-за сбоя в connect() или аутентификации, pgbouncer будет ждать указанное количество секунд перед повторной попыткой подключения.  По умолчанию — 15.0. |
| client\_login\_timeout | Если клиент подключается, но не может войти в систему за указанное количество секунд, он отключается. Требуется в основном, чтобы «мёртвые» подключения не задерживали операцию SUSPEND и, как следствие, перезагрузку «на лету».  По умолчанию — 60.0. |
| autodb\_idle\_timeout | Если пулы баз данных, созданные автоматически (через \*), не использовались указанное количество секунд, они освобождаются. Их статистика также сбрасывается.  По умолчанию — 3600.0. |
| dns\_max\_ttl | Определяет, как долго в секундах кэшировать запросы DNS. Если поиск в DNS возвращает несколько ответов, PgBouncer будет использовать их по очереди. Фактическое значение TTL DNS игнорируется.  По умолчанию — 15.0. |
| dns\_nxdomain\_ttl | Определяет, как долго можно кэшировать ошибки и запросы DNS NXDOMAIN в секундах.  По умолчанию — 15.0. |
| dns\_zone\_check\_period | Определяет интервал проверки, изменений серийных номеров зон.  PgBouncer может собрать список DNS-зон из имён хостов (все после первой точки), а затем периодически проверять, не изменились ли серийные номера зон. Если изменения обнаружены, все имена хостов в этой зоне ищутся снова. Если какой-либо IP-адрес хоста изменяется, его соединения становятся недействительными.  Работает только с библиотекой UDNS и c-ares (--with-udns или --with-cares для настройки).  По умолчанию — 0,0 (disabled). |

Параметры TLS

Таблица — Параметры TLS

| Параметр | Описание |
| --- | --- |
| client\_tls\_sslmode | Режим TLS, выбираемый для подключений от клиентов. По умолчанию соединения TLS отключены. Если параметр указан, client\_tls\_key\_file и client\_tls\_cert\_file также должны быть настроены для установки ключа и сертификата, которые PgBouncer использует для приёма клиентских подключений.   * disable — обычный TCP. Если клиент запрашивает TLS, он игнорируется. Режим по умолчанию. * allow — если клиент запрашивает TLS, он используется. В противном случае используется обычный TCP. Если клиент использует клиентский сертификат, он не проверяется. * prefer — то же, что и allow. * require — клиент должен использовать TLS. В противном случае клиентское соединение отклоняется. Если клиент использует клиентский сертификат, он не проверяется. * verify-ca — клиент должен использовать TLS с действующим клиентским сертификатом. * verify-full — то же, что verify-ca. |
| client\_tls\_key\_file | Закрытый ключ для PgBouncer для приёма клиентских подключений.  По умолчанию — не указан. |
| client\_tls\_cert\_file | Файл корневого сертификата для проверки сертификатов клиентов.  По умолчанию — не указан. |
| client\_tls\_ca\_file | Корневой сертификат для проверки сертификатов клиентов.  По умолчанию — не указан. |
| client\_tls\_protocols | Какие версии протокола TLS разрешены.  Допустимые значения: tlsv1.0, tlsv1.1, tlsv1.2.  Краткие обозначения: all (tlsv1.0, tlsv1.1, tlsv1.2), secure (tlsv1.2), legacy (all).  По умолчанию — all. |
| client\_tls\_ciphers | По умолчаниюt — fast. |
| client\_tls\_ecdhcurve | Имя эллиптической кривой для обмена ключами ECDH.  Допустимые значения: none (DH отключён), auto (256-битный ECDH), curve name.  По умолчанию — auto. |
| client\_tls\_dheparams | Тип обмена ключами DHE.  Допустимые значения: none (DH отключён), auto (2048-битный DH), legacy (1024-битный DH).  По умолчанию — auto. |
| server\_tls\_sslmode | Режим TLS для подключения к серверам RT.Warehouse и PostgreSQL. По умолчанию соединения TLS отключены.   * disabled — обычный TCP. TLS не запрашивается с сервера. Режим по умолчанию. * allow — если сервер отвергает обычный, попробуйте TLS. * prefer — сначала всегда запрашивается соединение TLS. Когда в соединении отказано, используется простой TPC. Сертификат сервера не проверяется. * require — соединение должно использовать TLS. Если сервер отклоняет это, простой TCP использоваться не будет. Сертификат сервера не проверяется. * verify-ca — соединение должно использовать TLS, а сертификат сервера должен быть действительным в соответствии с server\_tls\_ca\_file. Имя хоста сервера не проверяется сертификатом. * verify-full — соединение должно использовать TLS, а сертификат сервера должен быть действительным в соответствии с server\_tls\_ca\_file. Имя хоста сервера должно совпадать с именем хоста в сертификате. |
| server\_tls\_ca\_file | Путь к файлу корневого сертификата, который используется для проверки сертификатов сервера RT.Warehouse и PostgreSQL.  По умолчанию — не установлено. |
| server\_tls\_key\_file | Закрытый ключ для PgBouncer для аутентификации на сервере RT.Warehouse или PostgreSQL.  По умолчанию — не установлено. |
| server\_tls\_cert\_file | Сертификат на закрытый ключ. Серверы RT.Warehouse или PostgreSQL могут его проверить.  По умолчанию — не установлено. |
| server\_tls\_protocols | Определяет, какие версии протокола TLS разрешены.  Допустимые значения: tlsv1.0, tlsv1.1, tlsv1.2.  Краткие обозначения: all (tlsv1.0, tlsv1.1, tlsv1.2); secure (tlsv1.2); legacy (all).  По умолчанию — all. |
| server\_tls\_ciphers | По умолчанию — fast. |

Параметры опасных таймаутов

Установка следующих таймаутов может вызвать непредвиденные ошибки.

Таблица — Параметры опасных таймаутов

| Параметр | Описание |
| --- | --- |
| query\_timeout | Запросы, выполняющиеся дольше указанного времени (секунд), отменяются. Этот параметр следует использовать только с немного меньшим значением чем в statement\_timeout на стороне сервера, чтобы улавливать запросы с проблемами сети.  По умолчанию — 0.0 (disabled). |
| query\_wait\_timeout | Максимальное время в секундах, в течение которого запросы могут ожидать выполнения. Если запросу не назначено соединение в течение этого времени, клиент отключается. Это используется для предотвращения перехвата соединений неотвечающими серверами.  По умолчанию — 120. |
| client\_idle\_timeout | Клиентские соединения, бездействующие дольше этого количества секунд, закрываются. Это значение должно быть больше, чем таймаут подключения на стороне клиента, и использоваться только для сетевых проблем.  По умолчанию — 0.0 (disabled). |
| idle\_transaction\_timeout | Если клиент находился в состоянии «простоя в транзакции» дольше этого времени (секунд), он отключается.  По умолчанию — 0.0 (disabled). |

Низкоуровневые параметры сети

Таблица — Низкоуровневые параметры сети

| Параметр | Описание |
| --- | --- |
| pkt\_buf | Размер внутреннего буфера для пакетов. Влияет на размер отправляемых TCP-пакетов и общее использование памяти. Фактически пакеты libpq могут быть больше этого размера, поэтому нет необходимости устанавливать значение большим.  По умолчанию — 4096. |
| max\_packet\_size | Максимальный размер пакетов, которые принимает PgBouncer. Один пакет — это либо один запрос, либо одна строка набора результатов. Полный набор результатов может быть больше.  По умолчанию — 2147483647. |
| listen\_backlog | Параметр отставания для системного вызова listen(2). Он определяет, сколько новых безответных попыток подключения хранится в очереди. Когда очередь заполнена, дальнейшие попытки подключения сбрасываются.  По умолчанию — 128. |
| sbuf\_loopcnt | Определяет, сколько раз обрабатывать данные по одному соединению, прежде чем перейти к другому. Без этого ограничения одно соединение с большим набором результатов может задержать PgBouncer на долгое время. Один цикл обрабатывает один объём данных pkt\_buf. 0 означает отсутствие ограничений.  По умолчанию — 5. |
| suspend\_timeout | Определяет, сколько секунд ждать сброса буфера во время выполнения SUSPEND или перезапуска (-R). Соединение разрывается, если сбросить не удалось.  По умолчанию — 10. |
| tcp\_defer\_accept | Подробнее об этом и других параметрах TCP см. справочную страницу tcp(7).  По умолчанию — 45 в Linux, иначе — 0. |
| tcp\_socket\_buffer | По умолчанию — не установлено. |
| tcp\_keepalive | Включает базовую поддержку активности с настройками ОС по умолчанию.  В Linux системные значения по умолчанию: tcp\_keepidle=7200, tcp\_keepintvl=75, tcp\_keepcnt=9.  По умолчанию — 1. |
| tcp\_keepcnt | По умолчанию — не установлено. |
| tcp\_keepidle | По умолчанию — не установлено. |
| tcp\_keepintvl | По умолчанию — не установлено. |

Раздел [user]

Этот раздел содержит пары *key=value*, где *key* — это имя пользователя, а *value* — это строка подключения для libpq в виде пары *key*=*value*.

Параметры пула

Таблица — Параметры пула

| Параметр | Описание |
| --- | --- |
| pool\_mode | Устанавливает режим пула для всех подключений указанного пользователя. Если параметр не задан, используется значение базы данных или pool\_mode по умолчанию. |

Примеры файлов конфигурации

Минимальная конфигурация

|  |
| --- |
| [databases]  postgres = host=127.0.0.1 dbname=postgres auth\_user=gpadmin  [pgbouncer]  pool\_mode = session  listen\_port = 6543  listen\_addr = 127.0.0.1  auth\_type = md5  auth\_file = users.txt  logfile = pgbouncer.log  pidfile = pgbouncer.pid  admin\_users = someuser  stats\_users = stat\_collector |

Использовать параметры подключения, переданные клиентом:

|  |
| --- |
| [databases]  \* =  [pgbouncer]  listen\_port = 6543  listen\_addr = 0.0.0.0  auth\_type = trust  auth\_file = bouncer/users.txt  logfile = pgbouncer.log  pidfile = pgbouncer.pid  ignore\_startup\_parameters=options |

Настройки базы данных по умолчанию

|  |
| --- |
| [databases]  ; foodb over unix socket  foodb =  ; redirect bardb to bazdb on localhost  bardb = host=127.0.0.1 dbname=bazdb  ; access to destination database will go with single user  forcedb = host=127.0.0.1 port=300 user=baz password=foo client\_encoding=UNICODE datestyle=ISO |

pgbouncer-admin

Консоль администрирования PgBouncer.

Синтаксис

|  |
| --- |
| psql -p *port* pgbouncer |

Описание

Консоль администрирования PgBouncer доступна через psql. Чтобы войти в консоль, необходимо подключиться к порту PgBouncer и виртуальной базе данных с именем pgbouncer.

Пользователи, перечисленные в параметрах admin\_users и stats\_users конфигурации pgbouncer.ini, имеют права входить в консоль администрирования PgBouncer.

Вы можете управлять соединениями между PgBouncer и RT.Warehouse с консоли. Вы также можете установить параметры конфигурации PgBouncer.

Параметры

Таблица — Параметры pgbouncer-admin

| Параметр | Описание |
| --- | --- |
| -p *port* | Номер порта PgBouncer. |

Синтаксис команды

|  |
| --- |
| pgbouncer=# SHOW help;  NOTICE: Console usage  DETAIL:  SHOW HELP|CONFIG|DATABASES|POOLS|CLIENTS|SERVERS|VERSION  SHOW FDS|SOCKETS|ACTIVE\_SOCKETS|LISTS|MEM  SHOW DNS\_HOSTS|DNS\_ZONES  SHOW STATS|STATS\_TOTALS|STATS\_AVERAGES  SET key = arg  RELOAD  PAUSE [<db>]  RESUME [<db>]  DISABLE <db>  ENABLE <db>  KILL <db>  SUSPEND  SHUTDOWN |

Команды администрирования

Следующие административные команды PgBouncer управляют запущенным процессом pgbouncer.

Таблица — Команды администрирования

| Команда | Описание |
| --- | --- |
| PAUSE [*db*] | Если база данных не указана, PgBouncer пытается отключиться от всех серверов, сначала ожидая завершения всех запросов. Команда не выполнится, пока все запросы не будут завершены. Эта команда должна использоваться для подготовки к перезапуску базы данных.  Если указано имя базы данных, PgBouncer приостанавливает только эту базу данных.  Если вы запустите команду PAUSE *db*, а затем команду PAUSE, чтобы приостановить все базы данных, вы должны выполнить две команды RESUME, одну для всех баз данных и одну для указанной базы данных. |
| SUSPEND | Все буферы сокетов очищаются, и PgBouncer перестаёт прослушивать данные на них. Команда не выполнится, пока все буферы не очистятся. Эта команда должна использоваться для перезагрузки pgbouncer «на лету».. |
| RESUME [*db*] | Останавливает работу после предыдущей команды PAUSE или SUSPEND.  Если база данных была указана для команды PAUSE, база данных также должна быть указана с помощью команды RESUME.  После приостановки всех баз данных с помощью команды PAUSE возобновление одной базы данных с помощью RESUME **db** не поддерживается. |
| DISABLE *db* | Отклоняет все новые клиентские подключения к базе данных. |
| ENABLE *db* | Разрешает новые клиентские подключения к базе данных. |
| KILL *db* | Немедленно отключает все клиентские и серверные подключения к указанной базе данных. |
| SHUTDOWN | Останавливает процесс PgBouncer. Чтобы выйти из сеанса командной строки psql, введите \q. |
| RELOAD | Процесс PgBouncer перезагружает текущий файл конфигурации и обновляет изменяемые настройки. |
| SET *key*=*value* | Отменяет указанный параметр конфигурации. См. команду SHOW CONFIG;. |

Команда SHOW

Категории команды SHOW отображают различные типы информации PgBouncer. Вы можете указать одну из следующих категорий:

1. ACTIVE\_SOCKETS.

CLIENTS.

CONFIG.

DATABASES.

DNS\_HOSTS.

DNS\_ZONES.

FDS.

LISTS.

MEM.

POOLS.

SERVERS.

STATS.

STATS\_AVERAGES.

STATS\_TOTALS.

USERS.

VERSION.

ACTIVE\_SOCKETS

Таблица — Информация об активном сокете

| Столбец | Описание |
| --- | --- |
| type | S — для сервера, C — для клиента. |
| user | Имя пользователя, которое pgbouncer использует для подключения к серверу. |
| database | Имя базы данных. |
| state | Состояние подключения к серверу: active, used или idle. |
| addr | IP-адрес сервера PostgreSQL. |
| port | Порт сервера PostgreSQL. |
| local\_addr | Исходный адрес подключения на локальной машине. |
| local\_port | Исходный порт подключения на локальной машине. |
| connect\_time | Время установления подключения. |
| request\_time | Время выдачи последнего запроса. |
| wait | Время ожидания. |
| wait\_us | Время ожидания (микросекунды). |
| ptr | Адрес внутреннего объекта для данного соединения. Используется как уникальный идентификатор. |
| link | Адрес клиентского подключения, с которым связан сервер. |
| remote\_pid | Идентификатор обслуживающего серверного процесса (PID) |
| tls | Информация о TLS-подключении. |
| recv\_pos | Позиция приёма в буфере ввода-вывода. |
| pkt\_pos | Парсинг позиции в буфере ввода-вывода. |
| pkt\_remain | Количество пакетов, оставшихся в сокете. |
| send\_pos | Позиция отправки в пакете. |
| send\_remain | Общая длина пакета, оставшегося для отправки. |
| pkt\_avail | Объём буфера ввода-вывода, оставшийся для анализа. |
| send\_avail | Количество буфера ввода-вывода, оставшегося для отправки. |

CLIENTS

Таблица — Клиенты

| Столбец | Описание |
| --- | --- |
| type | C — для клиента. |
| user | Пользователь, подключённый со стороны клиента. |
| database | Имя базы данных. |
| state | Состояние клиентского соединения: active, used, waiting или idle. |
| addr | IP-адрес клиента или unix для подключения через сокет. |
| port | Порт, к которому подключён клиент. |
| local\_addr | Конечный адрес подключения на локальной машине. |
| local\_port | Конечный порт подключения на локальной машине.. |
| connect\_time | Время установления подключения. |
| request\_time | Время последнего запроса клиента. |
| wait | Текущая длительность ожидания (в секундах). |
| wait\_us | Дробная часть текущей длительности ожидания (в микросекундах). |
| ptr | Адрес внутреннего объекта для данного подключения. Используется как уникальный идентификатор. |
| link | Адрес подключения к серверу, с которым связан клиент. |
| remote\_pid | Идентификатор процесса (PID), в случае, если клиент подключается через сокет UNIX и ОС может выдать этот идентификатор. |
| tls | Клиентский контекст TLS. |

CONFIG

Список текущих настроек параметров PgBouncer.

Таблица — Конфигурация

| Столбец | Описание |
| --- | --- |
| key | Имя переменной конфигурации. |
| value | Значение переменной конфигурации. |
| changeable | Либо yes, либо no. Показывает, можно ли изменить переменную во время работы. Если no, переменную можно изменить только во время перезагрузки. |

DATABASES

Таблица — Базы данных

| Столбец | Описание |
| --- | --- |
| name | Имя настроенной записи базы данных. |
| host | Хост, к которому подключается pgbouncer. |
| port | Порт, к которому подключается pgbouncer. |
| database | Фактическое имя базы данных, к которой подключается pgbouncer. |
| force\_user | Когда пользователь указан в строке соединения, подключение между pgbouncer и сервером базы данных должно устанавливаться от его имени, вне зависимости от пользователя на стороне клиента. |
| pool\_size | Максимальное количество подключений к серверу. |
| reserve\_pool | Количество дополнительных подключений, которые могут быть созданы, если размер пула достигнет pool\_size. |
| pool\_mode | Переопределение pool\_mode для базы данных либо NULL, если должен использоваться режим по умолчанию. |
| max\_connections | Максимальное количество подключений для всех пулов для этой базы данных. |
| current\_connections | Общее количество подключений для всех пулов для этой базы данных. |
| paused | 1 — если база данных находится в состоянии паузы, 0 — иначе. |
| disabled | 1 — если база данных находится в отключённом состоянии, 0 — иначе. |

DNS\_HOSTS

Таблица — Хосты DNS в кэше

| Столбец | Описание |
| --- | --- |
| hostname | Имя хоста. |
| ttl | Сколько секунд до следующего поиска. |
| addrs | Список адресов, разделённых запятыми. |

DNS\_ZONES

Таблица — Зоны DNS в кэше

| Столбец | Описание |
| --- | --- |
| zonename | Название зоны. |
| serial | Текущий серийный номер DNS. |
| count | Имена хостов, принадлежащие этой зоне. |

FDS

SHOW FDS — это внутренняя команда, используемая для перезапуска через Интернет, например, при обновлении до новой версии PgBouncer. Он отображает список используемых файловых дескрипторов и их внутреннее состояние. Эта команда блокирует внутренний цикл событий, поэтому её не следует выполнять, пока используется PgBouncer.

При подключении пользователя с именем pgbouncer через сокет Unix из процесса с UID, совпадающим с UID текущего процесса, ему передаются реальные файловые дескрипторы.

Таблица — FDS

| Столбец | Описание |
| --- | --- |
| fd | Числовое значение файлового дескриптора. |
| task | Предназначение. Одно из значений — pooler, client или server. |
| user | Пользователь соединения, использующий указанный файловый дескриптор. |
| database | База данных подключения, использующий указанный файловый дескриптор. |
| addr | IP-адрес соединения, использующий указанный файловый дескриптор; unix, если используется сокет Unix. |
| port | Порт, используемый соединением с использованием файлового дескриптора. |
| cancel | Ключ отмены для этого подключения. |
| link | Файловый дескриптор ответной стороны сервера/клиента. NULL, если подключение простаивает. |
| client\_encoding | Кодировка, используемая для базы данных. |
| std\_strings | Контролирует, обрабатывают ли обычные строковые литералы ('...') обратную косую черту буквально, как указано в стандарте SQL. |
| datestyle | Формат отображения значений даты и времени. |
| timezone | Часовой пояс для интерпретации и отображения отметок времени. |
| password | Пароль auth\_user. |

LISTS

Показывает следующую статистику PgBouncer в двух столбцах: метку элемента и значение.

Таблица — Количество элементов PgBouncer

| Элемент | Описание |
| --- | --- |
| databases | Количество баз данных. |
| users | Количество пользователей. |
| pools | Количество пулов. |
| free\_clients | Количество свободных клиентов. |
| used\_clients | Количество занятых клиентов. |
| login\_clients | Количество клиентов в статусе login. |
| free\_servers | Количество свободных серверов. |
| used\_servers | Количество занятых серверов. |
| dns\_names | Количество DNS-имён. |
| dns\_zones | Количество зон DNS. |
| dns\_queries | Количество DNS-запросов. |
| dns\_pending | Количество запросов DNS «на лету». |

MEM

Показывает информацию о кэш-памяти для этих кэшей PgBouncer:

user\_cache;

db\_cache;

pool\_cache;

server\_cache;

client\_cache;

iobuf\_cache.

Таблица — Кэш in-memory

| Столбец | Описание |
| --- | --- |
| name | Имя кэша. |
| size | Размер одного слота в кэше. |
| used | Количество используемых слотов в кэше. |
| free | Количество доступных слотов в кэше. |
| memtotal | Общее количество байтов, используемых кэшем. |

POOLS

Новая запись в пуле создаётся для каждой пары (база данных, пользователь).

Таблица — Пулы

| Столбец | Описание |
| --- | --- |
| database | Имя базы данных. |
| user | Имя пользователя. |
| cl\_active | Клиентские подключения, которые связаны с сервером и могут обрабатывать запросы. |
| cl\_waiting | Клиентские подключения, которые отправили запросы, но ещё не получили подключение к серверу. |
| sv\_active | Число подключений к серверу, связанных с клиентом. |
| sv\_idle | Число неиспользуемых серверных соединений, которые можно сразу использовать для клиентских запросов. |
| sv\_used | Число серверных соединений, которые простаивают дольше server\_check\_delay. Запрос server\_check\_query должен быть запущен на них, прежде чем их можно будет использовать. |
| sv\_tested | Число подключений к серверу, для которых в данный момент выполняются запросы server\_reset\_query или server\_check\_query. |
| sv\_login | Число серверных соединений, которые в данный момент находятся в процессе входа. |
| maxwait | Время ожидания первого (самого старого) клиента в очереди в секундах. Если он начинает увеличиваться, текущий пул серверов не обрабатывает запросы достаточно быстро. Причина может быть в перегрузке сервера или слишком маленьком значении pool\_size. |
| maxwait\_us | Дробная часть максимального времени ожидания (в микросекундах). |
| pool\_mode | Действующий режим пула. |

SERVERS

Таблица — Серверы

| Столбец | Описание |
| --- | --- |
| type | S — для сервера. |
| user | ID пользователя, который pgbouncer использует для подключения к серверу. |
| database | Имя базы данных. |
| state | Состояние подключения к серверу pgbouncer: active, used или idle. |
| addr | IP-адрес сервера RT.Warehouse или PostgreSQL. |
| port | Порт сервера RT.Warehouse или PostgreSQL. |
| local\_addr | Исходный адрес подключения на локальной машине. |
| local\_port | Исходный порт подключения на локальной машине. |
| connect\_time | Время установления подключения. |
| request\_time | Время выдачи последнего запроса. |
| wait | Текущая длительность ожидания (в секундах). |
| wait\_us | Дробная часть текущей длительности ожидания (в микросекундах). |
| ptr | Адрес внутреннего объекта для этого подключения. Используется как уникальный идентификатор. |
| link | Адрес клиентского соединения, с которым связан сервер. |
| remote\_pid | Идентификатор обслуживающего серверного процесса (PID). Если соединение осуществляется через сокет Unix и ОС поддерживает получение информации об идентификаторе процесса, это идентификатор ОС. В противном случае он извлекается из пакета для отмены, отправленного сервером, который должен быть PID, если сервером является PostgreSQL, но это будет случайное число, если сервером является другой PgBouncer. |
| tls | Информация о TLS-подключении. |

STATS

Показывает статистику.

Таблица — Статистика

| Столбец | Описание |
| --- | --- |
| database | База данных, для которой представлена статистика. |
| total\_xact\_count | Общее количество транзакций SQL, объединённых PgBouncer. |
| total\_query\_count | Общее количество SQL-запросов, объединённых PgBouncer. |
| total\_received | Общий объём сетевого трафика (в байтах), который получил pgbouncer. |
| total\_sent | Общий объём сетевого трафика (в байтах), который передал pgbouncer. |
| total\_xact\_time | Общее количество микросекунд, потраченных PgBouncer при подключении к RT.Warehouse в транзакции, либо в режиме ожидания в транзакции, либо при выполнении запросов. |
| total\_query\_time | Общее количество микросекунд, потраченных pgbouncer при активном подключении к серверу базы данных. |
| total\_wait\_time | Время, затраченное (в микросекундах) клиентами на ожидание сервера. |
| avg\_xact\_count | Среднее количество транзакций SQL, объединённых PgBouncer. |
| avg\_query\_count | Среднее количество запросов в секунду за последний статистический период. |
| avg\_recv | Среднее количество полученных (от клиентов) байт в секунду. |
| avg\_sent | Среднее количество отправленных (клиентам) байт в секунду. |
| avg\_xact\_time | Средняя продолжительность транзакции в микросекундах. |
| avg\_query\_time | Средняя продолжительность запроса в микросекундах. |
| avg\_wait\_time | Время, затраченное клиентами на ожидание сервера в микросекундах (в среднем в секунду). |

STATS\_AVERAGES

Подмножество SHOW STATS, показывающее средние значения для выбранной статистики.

STATS\_TOTALS

Подмножество SHOW STATS, показывающее общие значения для выбранной статистики.

USERS

Таблица — Пользователи

| Столбец | Описание |
| --- | --- |
| name | Имя пользователя. |
| pool\_mode | Пользовательское переопределение pool\_mode или NULL, если вместо него будет использоваться значение по умолчанию. |

VERSION

Показывает информацию о версии PgBouncer.

|  |
| --- |
| **Примечание.**  Эта справочная документация основана на документации PgBouncer 1.8.1. |

psql

Интерактивный интерфейс командной строки для RT.Warehouse.

Синтаксис

|  |
| --- |
| psql [*option ...*] [*dbname* [*username*]] |

Описание

psql — это терминальный интерфейс для RT.Warehouse. Он позволяет вводить запросы в интерактивном режиме, отправлять их в RT.Warehouse и просматривать результаты запросов. Как вариант, ввод может быть из файла. Кроме того, он предоставляет ряд метакоманд и различные функции, подобные оболочке, для облегчения написания скриптов и автоматизации широкого спектра задач.

Параметры

Таблица — Параметры psql

| Параметр | Описание |
| --- | --- |
| -a | --echo-all | Вывод всех непустых строк ввода в стандартный вывод по мере их чтения. (Это не относится к строкам, читаемым в интерактивном режиме.) Это эквивалентно установке переменной ECHO в all. |
| -A | --no-align | Переключение в режим невыровненного вывода. (Режим вывода по умолчанию выровнен.). |
| -c '*command*' | --command='*command*' | Указывает, что psql должен выполнить указанную командную строку, а затем завершиться. Это полезно в скриптах оболочки. Команда должна быть либо командной строкой, которая полностью интерпретируется сервером, либо одной командой с обратной косой чертой. Таким образом, с этим параметром нельзя смешивать метакоманды SQL и psql. Для этого вы можете передать строку в psql, например:   |  | | --- | | echo '\x \\ SELECT \* FROM foo;' | psql |   (\\ — разделитель метакоманды)  Если командная строка содержит несколько SQL-команд, они обрабатываются в одной транзакции, только если в строку не включены явные команды BEGIN/COMMIT, разделяющие её на несколько транзакций. Это отличается от поведения, когда та же строка подаётся на стандартный ввод psql. Кроме того, возвращается только результат последней SQL-команды. |
| -d *dbname* | --dbname=*dbname* | Задаёт имя базы данных для подключения. Это эквивалентно указанию *dbname* в качестве первого аргумента командной строки, не являющегося параметром.  Если этот параметр содержит знак = или начинается с допустимого префикса URI (postgresql:// или postgres://), он рассматривается как строка conninfo. Для получения дополнительной информации см. [Строки подключения](https://www.postgresql.org/docs/9.4/libpq-connect.html#LIBPQ-CONNSTRING) в документации PostgreSQL. |
| -e | --echo-queries | Копирует все SQLкоманды, отправленные на сервер, также на стандартный вывод. |
| -E | --echo-hidden | Отображает фактические запросы, генерируемые \d и другими командами с обратной косой чертой. Вы можете использовать это для изучения внутренних операций psql. Это эквивалентно установке значения on переменной ECHO\_HIDDEN. |
| -f *filename* | --file=*filename* | Использует *filename* в качестве источника команд вместо интерактивного чтения команд. После обработки файла psql завершает работу. Это во многом эквивалентно метакоманде \i.  Если имя файла - (дефис), то стандартный ввод читается до индикации EOF или метакоманды \q. Обратите внимание, однако, что Readline в этом случае не используется (как если бы был указан -n).  Использование этого параметра немного отличается от записи psql < *filename*. В общем, оба варианта будут делать то, что вы ожидаете, но использование -f включает некоторые полезные функции, такие как сообщения об ошибках с номерами строк. Также существует небольшая вероятность того, что использование этого параметра снизит затраты на запуск. С другой стороны, вариант, использующий перенаправление ввода оболочки (теоретически), гарантированно даст точно такой же результат, который вы получили бы, если бы вводили все вручную. |
| -F *separator* | --field-separator=*separator* | Использует указанный разделитель в качестве разделителя полей для невыровненного вывода. |
| -H | --html | Переключает в режим вывода HTML. |
| -l | --list | Выводит список всех доступных баз данных и завершается. Другие параметры, не связанные с соединением, игнорируются. |
| -L *filename* | --log-file=*filename* | В дополнение к обычному выводу, записывает вывод результатов всех запросов в указанный файл. |
| -n | --no-readline | Отключает использование Readline для редактирования командной строки и использования истории команд. Может быть полезно для выключения расширенных действий клавиши табуляции при вырезании и вставке. |
| -o *filename* | --output=*filename* | Записывает вывод результатов всех запросов в указанный файл. |
| -P *assignment* | --pset=*assignment* | Позволяет указать параметры печати в стиле \pset в командной строке. Обратите внимание, что здесь вы должны разделять имя и значение знаком равенства вместо пробела. Таким образом, чтобы установить формат вывода в LaTeX, вы можете написать -P format=latex. |
| -q | --quiet | Указывает, что psql должен работать без вывода дополнительных сообщений. По умолчанию он выводит приветственные сообщения и различные информационные сообщения. Если используется этот параметр, ничего из этого не происходит. Это полезно использовать с параметром -c. Это эквивалентно установке значения on переменной QUIET. |
| -R *separator* | --record-separator=*separator* | Использует указанный разделитель в качестве разделителя записей для невыровненного режима вывода. |
| -s | --single-step | Запуск в пошаговом режиме. Это означает, что пользователю предлагается перед отправкой каждой команды на сервер возможность отменить выполнение. Используйте это для отладки скриптов. |
| -S | --single-line | Запуск в однострочном режиме, где новая строка завершает команду SQL, как и точка с запятой. |
| -t | --tuples-only | Отключает вывод имён столбцов и результирующей строки с количеством выбранных записей. Эта команда эквивалентна \pset tuples\_only и предоставляется для удобства. |
| -T *table\_options* | --table-attr= *table\_options* | Позволяет указать параметры, которые будут помещены в тег таблицы HTML. Подробности смотрите в \pset. |
| -v *assignment* | --set=*assignment* | --variable= *assignment* | Выполняет присвоение переменной, как метакоманда \set. Обратите внимание, что вы должны разделить имя и значение, если они есть, знаком равенства в командной строке. Чтобы отключить переменную, не ставьте знак равенства. Чтобы установить переменную с пустым значением, используйте знак равенства, но опустите значение. Эти присваивания выполняются на очень ранней стадии запуска, поэтому переменные, зарезервированные для внутренних целей, могут быть перезаписаны позже. |
| -V | --version | Выводит версию psql и завершает работу. |
| -x | --expanded | Включает режим развёрнутого вывода таблицы. |
| -X | --no-psqlrc | Не читать стартовые файлы (ни общесистемный файл psqlrc, ни пользовательский файл ~/.psqlrc). |
| -z | --field-separator-zero | Устанавливает нулевой байт в качестве разделителя полей для невыровненного режима вывода. |
| -0 | --record-separator-zero | Установить нулевой байт в качестве разделителя записей для невыровненного режима вывода. Это полезно для взаимодействия, например, с xargs -0. |
| -1 | --single-transaction | Когда psql выполняет скрипт, добавление этого параметра обёртывает BEGIN/COMMIT вокруг скрипта, чтобы выполнить его как одну транзакцию. Это гарантирует, что либо все команды будут выполнены успешно, либо никакие изменения не будут применены.  Если сам скрипт использует BEGIN, COMMIT или ROLLBACK, этот параметр не даст желаемых результатов. Кроме того, если скрипт содержит какую-либо команду, которая не может быть выполнена внутри блока транзакции, указание этого параметра приведет к сбою этой команды (и, следовательно, всей транзакции). |
| -? | --help | Показывает справку об аргументах командной строки psql и завершает работу. |

Таблица — Параметры подключения psql

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Имя хоста машины, на которой работает сервер мастера RT.Warehouse. Если параметр не указан, значение считывается из переменной среды PGHOST или по умолчанию используется localhost.  При запуске psql на хосте мастера, если значение хоста начинается с косой черты, он используется как каталог для сокета UNIX-домена. |
| -p *port* | --port=*port* | Порт TCP, на котором сервер мастера RT.Warehouse прослушивает соединения. Если параметр не указан, значение считывается из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя системной роли, используемое для подключения к базе данных. Если параметр не указан, значение считывается из переменной среды PGUSER или по умолчанию используется имя текущей системной роли. |
| -W | --password | Принудительно запрашивать пароль перед подключением к базе данных. psql автоматически запрашивает пароль всякий раз, когда сервер запрашивает аутентификацию по паролю. Однако в настоящее время обнаружение запроса пароля не является полностью надёжным, поэтому этот параметр принудительно выводит запрос. Если запрос пароля не выдаётся, а сервер требует аутентификации по паролю, попытка подключения не удастся. |
| -w --no-password | Не выдавать запрос на ввод пароля. Если сервер требует аутентификации по паролю, а пароль нельзя получить из других источников, например из файла .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который вводит пароль.  **Примечание**. Этот параметр действует на протяжении всей сессии, поэтому он влияет на использование метакоманды \connect, а также на первую попытку подключения. |

Статус завершения

Если psql завершилась нормально, она возвращает командной оболочке 0; 1 — если произошла фатальная ошибка (нехватка памяти, файл не найден); 2 — если соединение с сервером было неудачным и сеанс был не интерактивным; 3 — если произошла ошибка в скрипте и в установленной ​​переменной ON\_ERROR\_STOP.

Применение

Подключение к базе данных

psql — это клиентское приложение для RT.Warehouse. Для подключения к базе данных вам необходимо знать имя целевой базы данных, имя хоста и номер порта сервера мастера RT.Warehouse, а также имя пользователя базы данных, от имени которого вы хотите подключиться. Эти свойства можно задать через аргументы командной строки, а именно -d, -h, -p и -U соответственно. Если найден аргумент, не принадлежащий ни одному параметру, он будет интерпретирован как имя базы данных (или имя пользователя, если имя базы данных уже указано). Не все эти параметры задавать обязательно; у них есть разумные значения по умолчанию. Если вы опустите имя хоста, psql будет подключаться через сокет UNIX-домена к серверу мастера на локальном хосте или к localhost через TCP/IP на машинах, у которых нет сокетов UNIX-домена. Номер порта мастера по умолчанию — 5432. Если вы используете другой порт для мастера, вы должны указать порт. Имя пользователя базы данных по умолчанию — это имя пользователя вашей операционной системы, как и имя базы данных, по умолчанию. Обратите внимание, что вы не можете просто подключиться к любой базе данных под любым именем пользователя. Узнать о ваших правах доступа можно у администратора базы данных.

Если значения по умолчанию некорректные, вы можете сэкономить время на вводе параметров подключения, установив для любого из них или всех параметров в переменные среды PGAPPNAME, PGDATABASE, PGHOST, PGPORT и PGUSER соответствующие значения.

Также удобно иметь файл ~/.pgpass, чтобы избежать регулярного ввода паролей. Этот файл должен находиться в вашем домашнем каталоге и содержать строки следующего формата:

|  |
| --- |
| *hostname*:*port*:*database*:*username*:*password* |

Разрешения на .pgpass должны запрещать любой доступ всем или группе (например: chmod 0600 ~/.pgpass). Если разрешения менее строгие, файл будет проигнорирован. (Однако в настоящее время права доступа к файлам не проверяются на клиентах Microsoft Windows.)

Альтернативный способ указать параметры подключения — это строка conninfo или URI, который используется вместо имени базы данных. Этот механизм даёт расширенные возможности для управления подключением. Например:

|  |
| --- |
| $ psql "service=myservice sslmode=require"  $ psql postgresql://gpmaster:5433/mydb?sslmode=require |

Вы также можете использовать LDAP для получения параметров подключения, как описано в [Получение параметров подключения LDAP](https://www.postgresql.org/docs/9.4/libpq-ldap.html) в документации PostgreSQL. См. [Ключевые слова параметров](https://www.postgresql.org/docs/9.4/libpq-connect.html#LIBPQ-PARAMKEYWORDS) в документации PostgreSQL для получения дополнительной информации обо всех доступных вариантах подключения.

Если соединение не может быть установлено по какой-либо причине (недостаточные права, сервер не работает и т.д.), psql вернёт ошибку и завершится.

Если хотя бы один из стандартного ввода или стандартного вывода является терминалом, то psql устанавливает для клиентской кодировки значение auto, которое определит соответствующую клиентскую кодировку из локальных настроек (переменная среды LC\_CTYPE в системах Unix). Если это не сработает так, как ожидалось, кодировку клиента можно переопределить с помощью переменной среды PGCLIENTENCODING.

Ввод SQL-команд

При нормальной работе psql предоставляет подсказку с именем базы данных, к которой в настоящее время подключён psql, за которой следует строка => для обычного пользователя или =# для суперпользователя. Например:

|  |
| --- |
| testdb=>  testdb=# |

В командной строке пользователь может вводить команды SQL. Обычно введённые строки отправляются на сервер, когда встречается точка с запятой, завершающая команду. Конец строки не завершает команду. Это позволяет разбивать команды на несколько строк для лучшего понимания. Если команда была отправлена ​​и выполнена без ошибок, результат команды отобразится на экране.

Если к базе данных, которая не приведена в соответствие [шаблону безопасного использования схем](https://www.postgresql.org/docs/9.4/ddl-schemas.html#DDL-SCHEMAS-PATTERNS), имеют доступ недоверенные пользователи, начинайте сеанс с удаления доступных им для записи схем из пути поиска (search\_path). Для этого можно добавить options=-csearch\_path= в строку подключения или выполнить команду SELECT pg\_catalog.set\_config('search\_path', '', false) перед другими SQL-командами. Это касается не только psql, но и любых других интерфейсов для выполнения произвольных SQL-команд.

Метакоманды

Все, что вы вводите в psql, которое начинается с обратной косой черты и не берётся в кавычки, является метакомандой psql, которая обрабатывается самим psql. Эти команды помогают сделать psql более полезным для администрирования или написания скриптов.

Формат команды psql следующий: обратная косая черта (\), сразу за которой следует команда, а затем любые аргументы. Аргументы отделяются от команды и друг от друга любым количеством пробелов.

Чтобы включить пробел в значение аргумента, вы можете заключить его в одинарные кавычки. Чтобы включить одинарную кавычку в значение аргумента, напишите две одинарные кавычки в тексте в одинарных кавычках. Кроме того, все, что содержится в одинарных кавычках, подлежит замене в стиле языка C: для \n (новая строка), \t (табуляция), \b (пробел), \r (возврат каретки), \f (подача страницы), \digits (восьмеричное число) и \xdigits (шестнадцатеричное число). Обратная косая черта перед любым другим символом в тексте в одинарных кавычках заключает в кавычки этот единственный символ, каким бы он ни был.

Внутри аргумента текст, заключённый в обратные кавычки (`), воспринимается как командная строка, которая передается в оболочку. Вывод команды (с удалённым завершающим символом новой строки) заменяет текст в обратных кавычках.

Если в аргументе появляется двоеточие (:) без кавычек, за которым следует имя переменной psql, оно заменяется значением переменной, как описано в [Интерполяция SQL](https://docs.greenplum.org/6-0/utility_guide/ref/psql.html#topic1__section14).

Некоторые команды принимают в качестве аргумента идентификатор SQL (например, имя таблицы). Эти аргументы следуют синтаксическим правилам SQL: буквы без кавычек переводятся в нижний регистр, а двойные кавычки (") защищают буквы от преобразования регистра и позволяют включать пробелы в идентификатор. В двойных кавычках парные двойные кавычки сокращаются до одинарных двойных кавычек в получившееся имя. Например, FOO"BAR"BAZ интерпретируется как fooBARbaz, а "A weird"" name" становится A weird" name.

Синтаксический анализ аргументов останавливается, когда возникает другая обратная косая черта без кавычек. Это считается началом новой метакоманды. Специальная последовательность \\ (две обратные косые черты) отмечает конец аргументов и продолжает анализ SQL-команд, если они есть. Таким образом, SQL-команды и psql можно свободно смешивать в одной строке. Но в любом случае аргументы метакоманды не могут продолжаться за пределами конца строки.

Определены следующие метакоманды:

Таблица — Метакоманды

| Мета-команда | Описание |
| --- | --- |
| \a | Если текущий формат вывода таблицы невыровненный, он переключается на выровненный режим. Если текущий режим выровненный, то устанавливается невыровненный. Эта команда сохранена для обратной совместимости. См. \pset для более общего решения. |
| \c | \connect [*dbname* [*username*] [*host*] [*port*]] | *conninfo* | Устанавливает новое соединение с RT.Warehouse. Используемые параметры соединения могут быть указаны либо с использованием позиционного синтаксиса, либо с использованием строк подключения *conninfo*, как подробно описано в [Строках подключения libpq](https://www.postgresql.org/docs/9.4/libpq-connect.html#LIBPQ-CONNSTRING).  Если команда опускает имя базы данных, пользователя, хост или порт, новое соединение может повторно использовать значения из предыдущего соединения. По умолчанию повторно используются значения из предыдущего соединения, за исключением обработки строки *conninfo*. Передача первого аргумента -reuse-previous=on или -reuse-previous=off отменяет это значение по умолчанию. Если команда не задаёт и не использует повторно конкретный параметр, используется значение по умолчанию libpq. Указание любого из *dbname*, *username*, *host* или *port* как есть — эквивалентно пропуску этого параметра.  Если новое соединение установлено успешно, предыдущее соединение закрывается. Если попытка подключения не удалась, предыдущее подключение будет сохранено, только если psql находится в интерактивном режиме. При выполнении скрипта неинтерактивно, обработка немедленно прекращается с ошибкой. Это различное поведение было выбрано для удобства пользователя для защиты от опечаток и для защиты от случайных действий скриптов некорректно выбранной базе данных.  Примеры:   |  | | --- | | => \c mydb myuser host.dom 6432  => \c service=foo  => \c "host=localhost port=5432 dbname=mydb connect\_timeout=10 sslmode=disable"  => \c postgresql://tom@localhost/mydb?application\_name=myapp | |
| \C [*title*] | Устанавливает заголовок всех таблиц, которые выводятся в результате запроса, или отменяет любой такой заголовок. Эта команда эквивалентна \pset title. |
| \cd [*directory*] | Изменяет текущий рабочий каталог. Без аргументов устанавливается домашний каталог текущего пользователя. Чтобы распечатать текущий рабочий каталог, используйте \!pwd. |
| \conninfo | Отображает информацию о текущем соединении, включая имя базы данных, имя пользователя, тип соединения (сокет домена UNIX, TCP/IP и т.д.), хост и порт. |
| \copy {*table* [(*column\_list*)] | (*query*)} {from | to} {'*filename*' | program '*command*' | stdin | stdout | pstdin | pstdout} [with] (*option* [, ...]) ] | Выполняет внешнюю (клиентскую) копию. Это операция, которая запускает SQL-команду COPY, но вместо того, чтобы сервер считывал или записывал указанный файл, psql считывает или записывает файл и пересылает данные между сервером и локальной файловой системой. Это означает, что доступ к файлам и привилегии принадлежат локальному пользователю, а не серверу, и привилегии суперпользователя SQL не требуются.  Когда указана program, psql выполняет *command*, а данные от или к *command* маршрутизируются между сервером и клиентом. Это означает, что привилегии выполнения принадлежат локальному пользователю, а не серверу, и привилегии суперпользователя SQL не требуются.  \copy ... from stdin | to stdout читает/записывает на основе ввода и вывода команды соответственно. Все строки читаются из того же источника, который выдал команду, до тех пор, пока не встретится \. или поток не достигает EOF. Вывод отправляется в то же место, что и вывод команды. Для чтения/записи стандартного ввода или вывода psql, используйте pstdin или pstdout. Этот параметр полезен для заполнения таблиц прямо в файл SQL-скрипта.  Синтаксис команды аналогичен синтаксису SQL-команды COPY, а *option* должен указывать на один из параметров SQL-команды COPY. Обратите внимание, что из-за этого к команде \copy применяются особые правила синтаксического анализа. В частности, не применяются правила подстановки переменных и экранирование обратной косой черты.  Эта операция не так эффективна, как SQL-команда COPY, поскольку все данные должны проходить через соединение клиент-сервер. |
| \copyright | Показывает авторские права и условия распространения PostgreSQL, на котором основана RT.Warehouse. |
| \d [*relation\_pattern*] | \d+ [*relation\_pattern*] | \dS [*relation\_pattern*] | Для каждого отношения (таблица, внешняя таблица, представление, индекс, последовательность или сторонняя таблица) или составного типа, соответствующего *relation\_pattern*, показывает все столбцы, их типы, табличное пространство (если не по умолчанию) и любые специфические атрибуты, такие как NOT NULL или по умолчанию. Также выводятся связанные индексы, ограничения, правила и триггеры. Для сторонних таблиц также отображается связанный сторонний сервер.   * Для некоторых типов отношений \d выводит дополнительную информацию для каждого столбца: значения столбцов для последовательностей, индексированные выражения для индексов и параметры оболочки сторонних данных для сторонних таблиц. * Форма команды \d+ идентична, за исключением того, что выводится дополнительная информация: отображаются любые комментарии, связанные со столбцами таблицы, а также наличие OID в таблице, определение представления, если отношение является представлением.   Для многораздельных таблиц команда \d или \d+, указанная с корневой таблицей разделов или дочерней таблицей разделов, отображает информацию о таблице, включая ключи разделов, на текущем уровне таблицы разделов. Команда \d+ также отображает непосредственные дочерние разделы таблицы и показывает, является ли дочерний раздел внешней таблицей или обычной таблицей.  Для таблиц, оптимизированных для добавления, и таблиц, ориентированных на столбцы, \d+ отображает параметры хранения для таблицы. Для таблиц, оптимизированных для добавления, параметры отображаются для таблицы. Для таблиц, ориентированных на столбцы, параметры хранения отображаются для каждого столбца.   * По умолчанию отображаются только объекты, созданные пользователем; предоставить шаблон или модификатор S для включения системных объектов.   **Примечание**. Если \d используется без аргумента шаблона, он эквивалентен \dtvsE, который покажет список всех видимых таблиц, представлений, последовательностей и сторонних таблиц. |
| \da[S] [*aggregate\_pattern*] | Выводит перечень агрегатных функций вместе с типами данных, с которыми они работают. Если указан *aggregate\_pattern*, выводятся только агрегаты, имена которых соответствуют указанному шаблону. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S. |
| \db[+] [*tablespace\_pattern*] | Выводит перечень всех доступных табличных пространств и их соответствующих путей. Если указан *tablespace\_pattern*, выводятся только табличные пространства, имена которых соответствуют указанному шаблону. Если к имени команды добавлен +, для каждого объекта дополнительно выводятся связанные разрешения. |
| \dc[S+] [*conversion\_pattern*] | Выводит перечень преобразований между кодировками наборов символов. Если указан *conversion\_pattern*, выводятся только преобразования, имена которых соответствуют указанному шаблону. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S. Если к имени команды добавлен +, для каждого объекта дополнительно выводятся соответствующие описания. |
| \dC[+] [*pattern*] | Выводит перечень приведений типов. Если указан *pattern*, выводятся только те приведения, исходный или целевой типы которых соответствуют указанному шаблону. Если к имени команды добавлен +, для каждого объекта дополнительно выводятся соответствующие описания. |
| \dd[S] [*pattern*] | Выводит описания объектов следующих типов: constraint, operator class, operator family, rule и trigger. Описания остальных объектов можно посмотреть соответствующими командами с обратной косой чертой для этих типов объектов.  \dd выводит описания объектов, соответствующих *pattern*, или доступных объектов указанных типов, если аргумент не задан. Но в любом случае выводятся только объекты, имеющие описание. По умолчанию отображаются только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S.  Описания объектов могут быть созданы с помощью SQL-команды COMMENT. |
| \ddp [*pattern*] | Выводит перечень прав доступа по умолчанию. Выводится запись для каждой роли (и схемы, если применимо), для которой настройки привилегий по умолчанию были изменены по сравнению со встроенными значениями по умолчанию. Если указан *pattern*, выводятся только записи, имя роли или имя схемы которых соответствует указанному шаблону.  Команда ALTER DEFAULT PRIVILEGES используется для установки прав доступа по умолчанию. |
| \dD[S+] [*domain\_pattern*] | Выводит перечень доменов. Если указан *domain\_pattern*, выводятся только домены, имена которых соответствуют указанному шаблону. По умолчанию отображаются только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S. Если к имени команды добавлен +, каждый объект выводится с соответствующими разрешениями и описанием. |
| \dEistPv[S+] [*external\_table* | *index* | *sequence* | *table* | *parent table* | *view*] | Имя данной команды не настоящее: буквы E, i, s, t, P и v обозначают соответственно внешнюю таблицу, индекс, последовательность, таблицу, родительскую таблицу и представление. Вы можете указать любую букву или все эти буквы в любом порядке, чтобы получить список объектов этих типов. Например, \dit выведет перечень индексов и таблиц. Если к имени команды добавляется +, каждый объект выводится с указанием его физического размера на диске и соответствующего описания, если таковое имеется. Если указан шаблон, выводятся только объекты, имена которых соответствуют этому шаблону. По умолчанию отображаются только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S. |
| \des[+] [*foreign\_server\_pattern*] | Выводит перечень сторонних серверов. Если указан *foreign\_server\_pattern*, выводятся только те серверы, имя которых соответствует этому шаблону. Если используется форма \des+, отображается полное описание каждого сервера, включая ACL сервера, тип, версию, параметры и описание. |
| \det[+] [*foreign\_table\_pattern*] | Выводит перечень всех сторонних таблиц. Если указан *foreign\_table\_pattern*, выводятся только записи, имя таблицы или имя схемы которых соответствуют этому шаблону. Если используется форма \det+, также выводятся общие параметры и описание сторонней таблицы. |
| \deu[+] [*user\_mapping\_pattern*] | Выводит перечень сопоставлений пользователей. Если указан *user\_mapping\_pattern*, выводятся только те сопоставления, имена пользователей которых соответствуют указанному шаблону. Если используется форма \deu+, дополнительно выводится информация о каждом сопоставлении.  **Предупреждение**. \deu+ может также отображать имя пользователя и пароль удалённого пользователя, поэтому следует проявлять осторожность, чтобы не разглашать их. |
| \dew[+] [*foreign\_data\_wrapper\_pattern*] | Выводит перечень оболочек сторонних данных. Если указан *foreign\_data\_wrapper\_pattern*, выводятся только те оболочки сторонних данных, имя которых соответствует указанному шаблону. Если используется форма \dew+, дополнительно выводятся ACL, параметры и описание оболочки сторонних данных. |
| \df[antwS+] [*function\_pattern*] | Выводит перечень функций с типами данных их результатов, аргументов и классификацией: agg (агрегат), normal, trigger или window. Чтобы отображать только функции определённого типа(ов), добавьте в команду соответствующие буквы a, n, t или w. Если указан *function\_pattern*, выводятся только функции, имена которых соответствуют этому шаблону. Если используется форма \df+, дополнительно выводится информация о каждой функции, включая изменчивость, классификация по безопасности, язык, исходный код и описание. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо задать шаблон или добавить модификатор S. |
| \dF[+] [*pattern*] | Выводит перечень конфигураций текстового поиска. Если указан *pattern*, выводятся только конфигурации, имена которых соответствуют этому шаблону. Если используется форма \dF+, выводится полное описание каждой конфигурации, включая базовый синтаксический анализатор текстового поиска и список словарей для каждого типа фрагмента синтаксического анализатора. |
| \dFd[+] [*pattern*] | Выводит перечень словарей текстового поиска. Если указан *pattern*, выводятся только словари, имена которых соответствуют этому шаблону. Если используется форма \dFd+, дополнительно выводится информация о каждом выбранном словаре, включая базовый шаблон текстового поиска и значения параметров. |
| \dFp[+] [*pattern*] | Выводит перечень парсеров текстового поиска. Если указан *pattern*, выводятся только синтаксические анализаторы, имена которых соответствуют этому шаблону. Если используется форма \dFp+, выводится полное описание каждого синтаксического анализатора, включая основные функции и список распознанных типов фрагментов. |
| \dFt[+] [*pattern*] | Выводит перечень шаблонов текстового поиска. Если указан *pattern*, выводятся только шаблоны, имена которых соответствуют этому шаблону. Если используется форма \dFt+, дополнительно выводится информация о каждом шаблоне, включая имена основных функций. |
| \dg[+] [*role\_pattern*] | Выводит перечень ролей базы данных. (Поскольку понятия «пользователи» и «группы» были объединены в «роли», эта команда теперь эквивалентна \du.) Если указан *role\_pattern*, выводятся только те роли, имена которых соответствуют этому шаблону. Если используется форма \dg+, дополнительно выводится информация о каждой роли; в настоящее время это комментарий для каждой роли. |
| \dl | Это алиас для \lo\_list, который выводит перечень больших объектов.  **Примечание**. RT.Warehouse не поддерживает функцию больших объектов PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| \dL[S+] [*pattern*] | Выводит перечень процедурных языков. Если указан *pattern*, выводятся только языки, имена которых соответствуют этому шаблону. По умолчанию выводятся только языки, созданные пользователем; для включения системных объектов необходимо добавить модификатор S. Если к имени команды добавляется +, каждый язык выводится с указанием его обработчика вызовов, валидатора, прав доступа и информации о том, является ли он системным объектом. |
| \dn[S+] [*schema\_pattern*] | Выводит перечень всех доступных схем (пространств имен). Если указан *schema\_pattern*, выводятся только схемы, имена которых соответствуют этому шаблону. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо указать шаблон или добавить модификатор S. Если к имени команды добавлен +, каждый объект выводится с соответствующими разрешениями и описанием, если таковое имеется. |
| \do[S] [*operator\_pattern*] | Выводит перечень доступных операторов с их операндами и типами возвращаемых значений. Если указан *operator\_pattern*, выводятся только операторы, имена которых соответствуют этому шаблону. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо указать шаблон или добавить модификатор S. |
| \dO[S+] [*pattern*] | Выводит перечень правил сортировки. Если указан *pattern*, выводятся только правила, имена которых соответствуют шаблону. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо указать шаблон или добавить модификатор S. Если к имени команды добавлен +, каждое правило сортировки выводится с соответствующим описанием, если таковое имеется. Обратите внимание, что выводятся только правила, которые можно использовать с кодировкой текущей базы данных, поэтому результаты могут отличаться в разных базах данных одной и той же установки. |
| \dp [*relation\_pattern\_to\_show\_privileges*] | Выводит перечень таблиц, представлений и последовательностей с соответствующими правами доступа. Если указан *relation\_pattern\_to\_show\_privileges*, выводятся только таблицы, представления и последовательности, имена которых соответствуют этому шаблону. Команды GRANT и REVOKE используются для установки прав доступа. |
| \drds [*role-pattern* [*database-pattern*]] | Выводит перечень специфических параметров конфигурации. Данные параметры могут быть специфическими для роли, специфическими для базы данных, или обеих. *role-pattern* и *database-pattern* используются для выбора конкретных ролей и баз данных для вывода в перечень соответственно. Если шаблон не указан или указано \*, все параметры конфигурации, в том числе не относящиеся к ролям или базам данных.  Команды ALTER ROLE и ALTER DATABASE используются для определения параметров конфигурации, специфических для роли или базы данных. |
| \dT[S+] [*datatype\_pattern*] | Выводит перечень типов данных. Если указан *datatype\_pattern*, выводятся только типы, имена которых соответствуют этому шаблону. Если к имени команды добавлен +, каждый тип выводится с его внутренним именем и размером, его допустимыми значениями, если это тип enum, и соответствующими правами доступа. По умолчанию выводятся только объекты, созданные пользователем; для включения системных объектов необходимо указать шаблон или добавить модификатор S. |
| \du[+] [*role\_pattern*] | Выводит перечень ролей базы данных. (Поскольку понятия «пользователи» и «группы» были объединены в «роли», эта команда теперь эквивалентна \dg.) Если указан *role\_pattern*, выводятся только те роли, имена которых соответствуют этому шаблону. Если используется форма \du+, дополнительно выводится информация о каждой роли; в настоящее время это комментарий для каждой роли. |
| \dx[+] [*extension\_pattern*] | Выводит перечень установленных расширений. Если указан *extension\_pattern*, выводятся только те расширения, имена которых соответствуют этому шаблону. Если используется форма \dx+, выводятся все объекты, принадлежащие каждому соответствующему расширению. |
| \dy[+] [*pattern*] | Выводит перечень триггеров событий. Если указан *pattern*, выводятся только те триггеры, имена которых соответствуют шаблону. Если к имени команды добавлен +, каждый объект выводится вместе с соответствующим описанием.  **Примечание**. База данных RT.Warehouse не поддерживает пользовательские триггеры. |
| \e | \edit [*filename*] [*line\_number*] | Если указано *filename*, файл редактируется; после выхода из редактора его содержимое копируется обратно в буфер текущего запроса. Если *filename* не указано, буфер текущего запроса копируется во временный файл, который затем редактируется таким же образом.  Новое содержимое буфера запроса повторно анализируется в соответствии с обычными правилами psql, т.е. весь буфер обрабатывается как одна строка. (Т.е. вы не можете выполнять скрипты таким образом. Используйте для этого \i.) Это также означает, что если запрос заканчивается или содержит точку с запятой, он немедленно выполняется. В других случаях он просто будет ждать в буфере запроса; введите точку с запятой или \g, чтобы передать его, или \r, чтобы отменить.  Если указан номер строки, psql поместит курсор на указанную строку файла или буфера запроса. Обратите внимание: если задан единственный аргумент, состоящий из цифр, psql предполагает, что это номер строки, а не имя файла.  См. Переменные окружения для получения информации о настройке и настройке вашего редактора. |
| \echo text [ ... ] | Выводит аргументы в стандартный вывод, разделённые одним пробелом и с добавленным в конце переводом строки. Это может быть полезно для вывода скриптов. Если первым аргументом является -n без кавычек, то перевод строки в конце не ставится.  **Примечание**. Если вы используете команду \o для перенаправления вывода запроса, вы можете использовать \qecho вместо этой команды. |
| \ef [*function\_description* [*line\_number*]] | Эта команда извлекает из базы определение заданной функции в форме команды CREATE OR REPLACE FUNCTION и открывает его на редактирование. Редактирование осуществляется таким же образом, как и при выполнении \edit. После выхода из редактора изменённая команда остаётся в буфере запроса; введите точку с запятой или \g, чтобы отправить её серверу, либо \r, чтобы отменить.  Для функции может быть задано только имя или имя и аргументы, например foo(integer, text). Типы аргументов необходимы, если существует более чем одна функция с тем же именем.  Если функция не указана, для редактирования предоставляется пустой шаблон CREATE FUNCTION.  Если указан номер строки, psql поместит курсор в указанную строку тела функции. (Обратите внимание, что тело функции обычно не начинается с первой строки файла.)  См. Переменные окружения для получения информации о настройке и настройке вашего редактора. |
| \encoding [*encoding*] | Устанавливает кодировку набора символов клиента. Без аргумента эта команда показывает текущую кодировку. |
| \f [*field\_separator\_string*] | Устанавливает разделитель полей для невыровненного вывода запроса. По умолчанию используется вертикальная черта (|). См. также \pset для общего способа установки параметров вывода. |
| \g [*filename*]  \g [ | *command* ] | Отправляет содержимое буфера текущего запроса на сервер и, при необходимости, сохраняет выходные данные запроса в *filename* или направляет выходные данные в *command*. Файл или команда записываются только в том случае, если запрос успешно возвращает 0 или более строк, а не в случае сбоя запроса или SQL-команды, не возвращающей данные.  \g без аргумента по сути эквивалентен точке с запятой. \g с аргументом — это одноразовая альтернатива команде \o. |
| \gset [*prefix*] | Отправляет буфер текущего запроса на сервер для выполнения и сохраняет результат запроса в переменных psql. Выполняемый запрос должен возвращать ровно одну строку. Каждый столбец строки сохраняется в отдельной переменной, названной так же, как столбец. Например:   |  | | --- | | => SELECT 'hello' AS var1, 10 AS var2;  -> \gset  => \echo :var1 :var2  hello 10 |   Если вы указываете *prefix*, то он добавляется в начале к именам переменных:   |  | | --- | | => SELECT 'hello' AS var1, 10 AS var2;  -> \gset result\_  => \echo :result\_var1 :result\_var2  hello 10 |   Если результат столбца NULL, то вместо присвоения значения соответствующая переменная удаляется.  Если запрос завершается ошибкой или не возвращает одну строку, то никакие переменные не меняются. |
| \h | \help [*sql\_command*] | Выводит справку по синтаксису указанной *sql\_command*. Если команда не указана, psql выведет список всех команд, для которых доступна справка по синтаксису. Если в качестве команды указана звёздочка (\*), то отображается справка по синтаксису для всех SQL-команд. Для упрощения ввода команды, состоящей из нескольких слов, можно не заключать её в кавычки. |
| \H | \html | Включает вывод запросов в формате HTML. Если формат HTML уже включён, происходит переключение обратно на выровненный формат. Эта команда предназначена для совместимости и удобства, но о настройке других параметров вывода см. \pset. |
| \i | \include *filename* | Читает ввод из файла *filename* и выполняет его, как если бы он был набран на клавиатуре.  Если имя *filename* задано как - (дефис), то стандартный ввод читается до индикации EOF или метакоманды \q. Это можно использовать, чтобы совмещать интерактивный ввод с вводом из файлов. Обратите внимание, что поведение Readline будет использоваться, только если оно активно на самом внешнем уровне.  Если вы хотите видеть строки на экране по мере их чтения, вы должны установить для переменной ECHO значение all. |
| \ir | \include\_relative *filename* | Команда \ir похожа на \i, но по-разному интерпретирует относительные имена файлов. При выполнении в интерактивном режиме обе команды ведут себя идентично. Однако при вызове из скрипта \ir интерпретирует имена файлов относительно каталога, в котором расположен скрипт, а не текущего рабочего каталога. |
| \l[+] | \list[+] [*pattern*] | Выводит перечень баз данных на сервере и показывает их имена, владельцев, кодировки набора символов и права доступа. Если указан *pattern*, выводятся только базы данных, имена которых соответствуют этому шаблону. Если к имени команды добавлен +, также отображаются размеры базы данных, табличные пространства по умолчанию и описания. (Информация о размере доступна только для баз данных, к которым может подключиться текущий пользователь.) |
| \lo\_export *loid filename* | Читает большой объект с указанным OID *loid* из базы данных и записывает его в *filename*. Обратите внимание, что это немного отличается от серверной функции lo\_export, которая действует с правами пользователя, от имени которого работает сервер базы данных, и в файловой системе сервера. Используйте \lo\_list, чтобы узнать OID большого объекта.  **Примечание**. RT.Warehouse не поддерживает функцию больших объектов PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| \lo\_import *large\_object\_filename* [*comment*] | Сохраняет файл в большом объекте. При желании он связывает данный комментарий с объектом. Пример:   |  | | --- | | mydb=> \lo\_import '/home/gpadmin/pictures/photo.xcf' 'a  picture of me'  lo\_import 152801 |   Ответ указывает, что большой объект получил идентификатор объекта (OID) 152801, который может быть использован для доступа к вновь созданному объекту в будущем. По этой причине рекомендуется всегда связывать понятный комментарий с каждым объектом. Затем их можно увидеть с помощью команды \lo\_list. Обратите внимание, что эта команда немного отличается от команды lo\_import на стороне сервера, поскольку она действует как локальный пользователь в локальной файловой системе, а не как пользователь сервера в файловой системе.  **Примечание**. RT.Warehouse не поддерживает функцию больших объектов PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| \lo\_list | Выводит перечень всех больших объектов, хранящихся в настоящее время в базе данных, вместе с любыми комментариями к ним.  **Примечание**. RT.Warehouse не поддерживает функцию больших объектов PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| \lo\_unlink *largeobject\_oid* | Удаляет из базы данных большой объект указанного OID. Используйте \lo\_list, чтобы узнать OID большого объекта.  **Примечание**. RT.Warehouse не поддерживает функцию больших объектов PostgreSQL для потоковой передачи пользовательских данных, которые хранятся в структурах больших объектов. |
| \o | \out [ *filename* ]  \o | \out [ | *command* ] | Сохраняет результаты запроса в файл *filename* или передаёт результаты команде оболочки *command*. Если аргумент не указан, вывод запроса перенаправляется на стандартный вывод. Результаты запроса включают все таблицы, ответы на команды и уведомления, полученные от сервера базы данных, а также вывод различных команд с обратной косой чертой, которые обращаются к базе данных (например, \d), но не сообщения об ошибках. Чтобы совмещать вывод текста между результатами запроса, используйте \qecho. |
| \p | Печатает содержимое буфера текущего запроса в стандартный вывод. |
| \password [*username*] | Изменяет пароль указанного пользователя (по умолчанию текущего пользователя). Эта команда запрашивает новый пароль, шифрует его и отправляет на сервер как команду ALTER ROLE. Это гарантирует, что новый пароль не появится в открытом виде в истории команд, журнале сервера или где-либо ещё. |
| \prompt [ *text* ] *name* | Предлагает пользователю ввести значение, которое назначается переменной *name*. Можно указать текст подсказки *text*. (Для подсказок, состоящих из нескольких слов, заключите текст в одинарные кавычки.)  По умолчанию \prompt использует терминал для ввода и вывода. Однако, если используется параметр командной строки -f, \prompt использует стандартный ввод и стандартный вывод. |
| \pset [*print\_option* [*value*]] | Эта команда устанавливает параметры, влияющие на вывод таблиц результатов запроса. *print\_option* описывает, какой параметр должен быть установлен. Семантика *value* варьируется в зависимости от выбранного параметра. Для некоторых параметров пропуск *value* приводит к тому, что значение параметра будет переключено или сброшено, как описано в разделе о конкретном параметре. Если такое поведение не упоминается, то пропуск *value* просто приводит к отображению текущего значения параметра.  \pset без аргументов отображает текущее состояние всех параметров команды.  Настраиваемые параметры:   * border — *value* должно быть числом. В целом, чем больше это число, тем больше границ и линий будет в таблицах, но детали зависят от формата. В формате HTML заданное значение напрямую отображается в атрибут border=.... Для большинства других форматов имеют смысл только значения 0 (нет границы), 1 (внутренние разделительные линии) и 2 (граница таблицы). Форматы latex и latex-longtable дополнительно поддерживают значение 3, добавляющее разделительные линии между строками данных. * columns — устанавливает максимальную ширину для формата wrapped, а также ограничение по ширине, свыше которого будет требоваться постраничник или произойдёт переключение в вертикальное отображение при режиме expanded auto. При значении 0 (по умолчанию) максимальная ширина управляется переменной среды COLUMNS или шириной экрана, если COLUMNS не установлена. Кроме того, если columns равно нулю, то формат wrapped влияет только на вывод на экран. Если columns не равно 0, то это также влияет на вывод в файл или в другую команду через канал.   После установки целевой ширины используйте команду \pset format wrapped, чтобы включить формат wrapped.   * expanded | x — указанное *value* допускает варианты on или off, которые включают или отключают развёрнутый режим, либо auto. Если *value* опущено, команда включает/отключает режим. Когда развёрнутый режим включён, результаты запроса выводятся в две колонки: имя столбца в левой, данные в правой. Этот режим полезен, если данные не помещаются на экране в обычном «горизонтальном» режиме. При выборе auto развёрнутый режим используется, когда результат запроса содержит несколько столбцов и по ширине не умещается на экране; в противном случае используется обычный режим. Режим auto распространяется только на форматы aligned и wrapped. С другими форматами он всегда равнозначен отключённому состоянию. * fieldsep — устанавливает разделитель полей для невыровненного режима вывода запросов. Таким образом, можно формировать вывод, в котором значения будут разделены, например, табуляцией. Это может быть предпочтительным для использования в других программах. Для установки символа табуляции в качестве разделителя полей введите \pset fieldsep '\t'. По умолчанию используется вертикальная черта ('|'). * fieldsep\_zero — устанавливает разделитель полей для невыровненного режима вывода в нулевой байт. * footer — возможны два варианта *value*: on или off, которые включают или отключают вывод результирующей строки с количеством выбранных записей (*n* строк). Если *value* не указано, то команда переключает текущее значение в on или off. * format - устанавливает один из форматов вывода: unaligned, aligned, html, latex (использует tabular), latex-longtable, troff-ms, or wrapped. Допускаются уникальные сокращения.   в формате unaligned все столбцы выводятся в одной строке и отделяются друг от друга активным разделителем полей. Это полезно для получения вывода, который будут читать другие программы, например, для вывода данных с разделителем Tab или через запятую.  формат aligned — это стандартный, удобочитаемый, красиво отформатированный текстовый вывод; это значение по умолчанию.  форматы html, latex, latex-longtable и troff-ms выводят таблицы, которые предназначены для включения в документы с помощью соответствующего языка разметки. Они не являются полноценными документами! Это может быть не критично для HTML, но для LaTeX обязателен документ-контейнер. Формат latex использует среду LaTeX tabular, а формат latex-longtable требует наличия пакетов longtable и booktabs.  формат wrapped аналогичен aligned, но переносит длинные значения столбцов на новые строки, чтобы общий вывод поместился в заданную ширину. Задание ширины вывода описано в параметре columns. Обратите внимание, что psql не будет пытаться переносить на новые строки заголовки столбцов. Поэтому формат wrapped работает так же, как aligned если общая ширина, требуемая для всех заголовков столбцов, превышает установленную максимальную ширину.   * linestyle [unicode | ascii | old-ascii] — задаёт стиль отрисовки линий границы: ascii, old-ascii или unicode. Допускается сокращение слова до уникального значения. (Это значит, что одной буквы будет достаточно.) Значение по умолчанию: ascii. Этот параметр действует только в форматах aligned и wrapped.   ascii — стиль использует обычные символы ASCII. Символы новой строки в данных показываются с использованием символа + в правом поле. Когда при формате wrapped происходит перенос данных на новую строку (без символа новой строки), ставится точка (.) в правом поле первой строки и точка в левом поле следующей строки.  old-ascii — стиль использует обычные символы ASCII в стиле PostgreSQL 8.4 и раньше. Символы новой строки в данных отображаются, используя символ : вместо левого разделителя полей. Когда происходит перенос данных на новую строку без символа новой строки, символ ; используется вместо левого разделителя полей.  unicode — стиль использует символы Юникода для рисования линий. Символы новой строки в данных показываются с использованием символа возврата каретки в правом поле. Когда при формате wrapped происходит перенос данных на новую строку (без символа новой строки), ставится символ многоточия в правом поле первой строки и в левом поле следующей строки.  Когда значение border больше нуля, параметр linestyle также определяет символы, которыми будут рисоваться границы. Обычные символы ASCII работают везде, но символы Юникода смотрятся лучше на терминалах, распознающих их.   * null 'string' — устанавливает строку, которая будет напечатана вместо значения null. По умолчанию не печатается ничего, что можно ошибочно принять за пустую строку. Например, можно было бы предпочесть \pset null '(null)'. * numericlocale — если задаётся *value*, возможны два варианта: on или off, которые включают или отключают отображение специфичного для локали символа, разделяющего группы цифр левее десятичной точки. Если *value* не указано, то команда переключает вывод чисел с локализованного на обычный и обратно. * pager — управляет использованием постраничника для просмотра результатов запросов и справочной информации psql. Если установлена переменная среды PAGER, вывод передаётся указанной программе. В противном случае используется платформозависимая программа по умолчанию (например, more). Если pager имеет значение off, программа постраничного просмотра (постраничник) не используется. Если pager имеет значение on, эта программа используется при необходимости, т.е. когда вывод на терминал не помещается на экране. Параметр pager также может иметь значение always, при этом постраничник будет использоваться всегда, независимо от того, помещается вывод на экран терминала или нет. Команда \pset pager без указания *value* переключает варианты on и off. * recordsep — устанавливает разделитель записей (строк) для невыровненного режима вывода. По умолчанию используется символ новой строки. * recordsep\_zero — устанавливает разделитель записей для невыровненного режима вывода в нулевой байт. * tableattr | T [*text*] — устанавливает атрибуты, которые будут помещены в тег table, при формате вывода HTML. Например, это может быть cellpadding или border. Заметьте, что, вероятно, не нужно здесь задавать border, так как для этого уже есть \pset border. Если *value* не задано, атрибуты таблицы удаляются. В формате latex-longtable этот параметр контролирует пропорциональную ширину каждого столбца, данные которого выровнены по левому краю. Он указывается как список разделённых пробелами значений, например '0.2 0.2 0.6'. Для столбцов, которым не хватает значений, используется последнее из заданных. * title [*text*] — устанавливает заголовок таблицы для любых впоследствии выводимых таблиц. Это можно использовать для задания описательных тегов при формировании вывода. Если *value* не задано, заголовок таблицы удаляется. * tuples\_only | t [*novalue* | *on* | *off*] — если указано *value*, оно должно быть on или off, которые включают или отключают режим вывода только кортежей. Если *value* не указано, то команда переключает с режима вывода только кортежей на обычный режим и обратно. Обычный вывод включает в себя дополнительную информацию, такую как заголовки столбцов и различные колонтитулы. В режиме вывода только кортежей отображаются только фактические табличные данные. Команда \t эквивалентна \psettuples\_only и предоставляется для удобства.   Для \pset есть различные команды быстрого доступа. См. \a, \C, \f, \H, \t, \T и \x. |
| \q | \quit | Выход из psql. При использовании в скрипте прекращается только выполнение этого скрипта. |
| \qecho *text* [ ... ] | Эта команда идентична \echo, за исключением того, что вывод будет записан в канал вывода запроса, как установлено \o. |
| \r | \reset | Сбрасывает (очищает) буфер запроса. |
| \s [*filename*] | Записывает историю командной строки psql в *filename*. Если *filename* не указано, история записывается на стандартный вывод (с использованием постраничника, когда уместно). Эта команда недоступна, если psql был собран без поддержки Readline. |
| \set [*name* [*value* [ ... ]]] | Устанавливает переменной psql *name* указанное значение *value* или, если указано более одного значения, объединению их всех. Если указан только один аргумент, значением переменной становится пустая строка. Для сброса переменной используйте команду \unset.  \set без аргументов выводит имена и значения всех текущих переменных psql.  Имена переменных могут содержать буквы, цифры и знаки подчёркивания. Подробнее см. Переменные. Имена переменных чувствительны к регистру.  Хотя вы можете установить любую переменную, какую захотите, psql рассматривает несколько переменных как особые. Они задокументированы в теме о переменных.  Эта команда не связана с SQL-командой SET. |
| \setenv *name* [ *value* ] | Задаёт для имени *name* переменной среды значение *value* или, если *value* не указано, удаляет переменную среды. Пример:   |  | | --- | | testdb=> \setenv PAGER less  testdb=> \setenv LESS -imx4F | |
| \sf[+] *function\_description* | Эта команда извлекает из базы и выводит определение заданной функции в форме команды CREATE OR REPLACE FUNCTION. Определение выводится в текущий канал вывода запроса, как установлено \o.  Для функции может быть задано только имя или имя и аргументы, например foo(integer, text). Типы аргументов необходимы, если существует более чем одна функция с тем же именем.  Если к имени команды добавляется +, то выходные строки нумеруются, причём первой строкой тела функции является строка 1. |
| \t [*novalue* | *on* | *off*] | Включает/выключает отображение имён столбцов и результирующей строки с количеством выбранных записей для запросов. Значения on и off устанавливают отображение кортежей независимо от текущего значения. Эта команда эквивалентна \pset tuples\_only и предоставляется для удобства. |
| \T *table\_options* | Задаёт атрибуты, которые будут помещены в тег table в формате вывода HTML. Эта команда эквивалентна \pset tableattr *table\_options*. |
| \timing [*novalue* | *on* | *off*] | Без параметра переключает отображение выполнения каждого оператора SQL в миллисекундах на противоположное. Значения on и off устанавливают отображение времени независимо от текущей настройки. |
| \unset *name* | Сбрасывает (удаляет) *name* переменной psql. |
| \w | \write *filename*  \w | \write | *command* | Выводит буфер текущего запроса в файл *filename* или передаёт его команде оболочки *command*. |
| \watch [*seconds*] | Многократно выполняет текущий запрос в буфере (как \g), пока не будет прервана или не возникнет ошибка. Аргумент *seconds* задаёт количество секунд ожидания между выполнениями запроса (по умолчанию — 2). |
| \x [ *on* | *off* | *auto* ] | Устанавливает или переключает режим развёрнутого вывода таблицы. Это эквивалентно \pset expanded. |
| \z [*pattern*] | Выводит перечень таблиц, представлений и последовательностей с их правами доступа. Если указан *pattern*, выводятся только таблицы, представления и последовательности, имена которых соответствуют ему.  Это псевдоним для \dp. |
| \! [*command*] | Без аргументов запускает подчинённую оболочку; когда эта оболочка завершается, psql продолжает работу. Если добавлен аргумент, запускает команду оболочки *command*. Аргументы далее не интерпретируются; оболочка увидит их как есть. В частности, не применяются правила замены переменных и экранирование обратной косой черты. |
| \? | Показывает справочную информацию о командах psql с обратной косой чертой. |

Паттерны

Различные команды \d принимают параметр *шаблона*, чтобы указать имя (имена) объекта для отображения. В простейшем случае шаблон — это просто точное имя объекта. Символы в шаблоне обычно переводятся в нижний регистр, как и в именах SQL-объектов; например, \dt FOO выведет таблицу с именем foo. Как и в именах SQL, размещение шаблона в двойных кавычках предотвратить перевод в нижний регистр. Для включения символа двойной кавычки в шаблон используются два символа двойных кавычек подряд внутри шаблона в двойных кавычках. Опять же, это соответствует правилам для SQL-идентификаторов. Например, \dt "FOO""BAR" отобразит таблицу с именем FOO"BAR (не foo"bar). В отличие от обычных правил для SQL-имён, вы можете заключить в двойные кавычки только часть шаблона, например, \dt FOO"FOO"BAR выведет таблицу с именем fooFOObar.

Если в *шаблоне* указано \*, это соответствует любой последовательности символов (включая отсутствие символов), а указание ? в качестве *шаблона* соответствует любому одному символу. (Эта нотация сопоставима с шаблонами имён файлов оболочки UNIX.) Например, \dt int\* выводит все таблицы, имена которых начинаются с int. Но в двойных кавычках \* и ? теряют своё специальное значение и становятся обычными символами.

Шаблон, содержащий точку (.), интерпретируется как шаблон имени схемы, за которым следует шаблон имени объекта. Например, \dt foo\*.bar\* отображает все таблицы, имена которых включают bar, и расположенные в схемах, имена которых начинаются с foo. Шаблону, не содержащему точку, могут соответствовать только объекты текущей схемы. Опять же, точка внутри двойных кавычек теряет своё специальное значение.

Опытные пользователи могут использовать нотации регулярных выражений. Все специальные символы регулярных выражений работают, как указано в документации [PostgreSQL по регулярным выражениям](https://www.postgresql.org/docs/9.4/functions-matching.html#FUNCTIONS-POSIX-REGEXP), за исключением: ., которая используется как разделитель, как упоминалось выше; \* , которая переводится в нотацию регулярных выражений; \* и ? соответствуют .. Вы можете при необходимости эмулировать эти символы шаблона, указав ? для эмуляции ., (R+|) для эмуляции R\* или (R|) для эмуляции R?. Помните, что шаблон должен совпадать с именем целиком, в отличие от обычной интерпретации регулярных выражений. Укажите \* в начале и/или в конце, если вы не хотите, чтобы шаблон был привязан. Обратите внимание, что внутри двойных кавычек, все специальные символы регулярных выражений теряют своё специальное значение и соответствуют сами себе. Также, специальные символы регулярных выражений не действуют в шаблонах для имён операторов (т. е. в аргументе команды \do).

Всякий раз, когда параметр *шаблона* полностью опущен, команды \d отображают все объекты, которые видны в текущем пути поиска схемы — это эквивалентно использованию в качестве *шаблона* \*. Чтобы увидеть все объекты в базе данных, независимо от видимости, используйте в качестве *шаблона* \*.\*.

Расширенные возможности

Переменные

psql предоставляет возможности подстановки переменных подобные тем, что используются в командных оболочках Unix. Переменные представляют собой пары имя/значение, где значением может быть любая строка любой длины. Имя должно состоять из букв (включая нелатинские буквы), цифр и знаков подчёркивания.

Чтобы установить переменную, используйте метакоманду psql \set. Например:

|  |
| --- |
| testdb=> \set foo bar |

Присваивает переменной foo значение bar. Чтобы получить значение переменной, нужно поставить двоеточие перед её именем, например:

|  |
| --- |
| testdb=> \echo :foo  bar |

Это работает как в обычных SQL-командах, так и в метакомандах; подробнее в [Интерполяция SQL](https://docs.greenplum.org/6-0/utility_guide/ref/psql.html#topic1__section14).

Если вы вызываете \set без второго аргумента, переменная устанавливается с пустой строкой в ​​качестве значения. Чтобы сбросить (т.е. удалить) переменную, используйте команду \unset. Чтобы показать значения всех переменных, вызовите \set без аргументов.

|  |
| --- |
| **Примечание.**  Аргументы \set подчиняются тем же правилам замены, что и другие команды. Таким образом, вы можете создавать интересные ссылки, такие как \set :foo 'something', и получать 'мягкие ссылки' в Perl или 'переменные переменные' в PHP. К сожалению, с этими конструкциями ничего полезного сделать нельзя. С другой стороны, \set bar :foo — вполне допустимый способ скопировать переменную. |

Некоторые из этих переменных обрабатываются psql особым образом. Они представляют собой определённые настройки параметров, которые можно изменить во время выполнения, изменяя значение переменной, или в некоторых случаях представляют изменяемое состояние psql. Хотя вы можете использовать эти переменные для других целей, это не рекомендуется, поскольку поведение программы может очень быстро стать очень странным. По соглашению, имена всех специально обработанных переменных состоят из заглавных букв ASCII (и, возможно, цифр и знаков подчёркивания). Чтобы обеспечить максимальную совместимость в будущем, избегайте использования таких имён переменных в своих целях. Список всех специальных переменных следует ниже.

Таблица — Переменные

| Переменная | Описание |
| --- | --- |
| AUTOCOMMIT | При значении on (по умолчанию) после каждой успешно выполненной команды выполняется фиксация изменений. Чтобы отложить фиксацию изменений в этом режиме, нужно выполнить SQL-команду BEGIN или START TRANSACTION. При значении off или если переменная не определена, фиксация изменений не происходит до тех пор, пока явно не выполнена команда COMMIT или END. При значении off неявно выполняется BEGIN непосредственно перед любой командой, за исключением случаев когда: команда уже в транзакционном блоке; перед самой командой BEGIN или другой командой управления транзакциями; перед командой, которая не может выполняться внутри транзакционного блока (например, VACUUM).  Если режим autocommit отключён, необходимо явно откатывать изменения в неуспешных транзакциях, выполняя команду ABORT или ROLLBACK. Также имейте в виду, что при выходе из сессии без фиксации изменений несохранённые изменения будут потеряны.  Включённый режим autocommit является традиционным для PostgreSQL, а выключенный режим ближе к спецификации SQL. Если вы предпочитаете отключить режим autocommit, это можно сделать в общесистемном файле psqlrc или в персональном файле ~/.psqlrc. |
| COMP\_KEYWORD\_CASE | Определяет, какой регистр букв будет использован при автоматическом завершении ключевых слов SQL. Если установлено lower или upper, будет использоваться нижний или верхний регистр соответственно. Если установлено preserve-lower или preserve-upper (по умолчанию), то завершаемое слово будет в том же регистре, что и уже введённое начало слова, но последующие слова, завершаемые полностью, будут в нижнем или верхнем регистре соответственно. |
| DBNAME | Имя базы данных, к которой вы в данный момент подключены. Он устанавливается каждый раз, когда вы подключаетесь к базе данных (включая запуск программы), но эту переменную можно изменить или сбросить. |
| ECHO | Если установлено значение all, все непустые строки ввода выводятся на стандартный вывод по мере чтения. (Это не относится к строкам, читаемым в интерактивном режиме.) Чтобы выбрать это поведение при запуске программы, используйте ключ -a. Со значением queries psql выдаёт каждый запрос, отправляемый серверу, в стандартный вывод. Этому значению соответствует ключ -e. |
| ECHO\_HIDDEN | Когда для этой переменной установлено значение on и метакоманда запрашивает базу данных, сначала выводится текст нижележащего запроса. Эта функция поможет вам изучить внутреннее устройство RT.Warehouse и реализовывать похожую функциональность в своих программах. (Чтобы выбрать это поведение при запуске программы, используйте ключ -E.) Если вы установите для переменной значение noexec, запросы просто отображаются, но фактически не отправляются на сервер и не выполняются. |
| ENCODING | Текущая кодировка набора символов на стороне клиента. |
| FETCH\_COUNT | Если для этой переменной задано целочисленное значение больше нуля, результаты запросов SELECT извлекаются и отображаются группами с заданным количеством строк, в отличие от поведения по умолчанию, когда перед отображением результирующий набор накапливается целиком. Это позволяет использовать ограниченный размер памяти независимо от размера выборки. При включении этой функциональности обычно используются значения от 100 до 1000. Имейте в виду, что запрос может завершиться ошибкой после отображения некоторого количества строк.  Хотя можно использовать любой формат вывода, формат по умолчанию aligned как правило выглядит хуже, потому что каждая группа по FETCH\_COUNT строк форматируется отдельно, что может привести к разной ширине столбцов в разных группах. Остальные форматы вывода работают лучше. |
| HISTCONTROL | Если переменная имеет значение ignorespace, строки, начинающиеся с пробела, не сохраняются в истории. Если она имеет значение ignoredups, в историю не добавляются строки, которые в ней уже есть. Значение ignoreboth объединяет эти два варианта. Со значением none (по умолчанию) в истории сохраняются все строки, считываемые в интерактивном режиме. |
| HISTFILE | Имя файла, который будет использоваться для хранения списка истории команд. Значение по умолчанию — ~/psql\_history. Например, установив:   |  | | --- | | \set HISTFILE ~/.psql\_history- :DBNAME |   в ~/.psqlrc, psql будет вести отдельный файл истории для каждой базы данных. |
| HISTSIZE | Максимальное количество команд, сохраняемых в истории команд. Значение по умолчанию — 500. |
| HOST | Хост сервера базы данных, к которому вы в настоящее время подключены. Он устанавливается каждый раз, когда вы подключаетесь к базе данных (включая запуск программы), но эту переменную можно изменить или сбросить. |
| IGNOREEOF | Если значение не установлено, отправка символа конца файла EOF (обычно CTRL + D) в интерактивном сеансе psql завершит работу приложения. Если установлено числовое значение, такое количество символов EOF игнорируется до завершения работы приложения. Если переменная установлена, но не имеет числового значения, значение по умолчанию — 10. |
| LASTOID | Содержит значение последнего OID, полученного командой INSERT или \lo\_import. Корректное значение переменной гарантируется до тех пор, пока не будет отображён результат следующей SQL-команды. |
| ON\_ERROR\_ROLLBACK | При значении on, если команда в блоке транзакции выдаёт ошибку, ошибка игнорируется и транзакция продолжается. Со значением interactive такие ошибки игнорируются только в интерактивных сеансах, но не в скриптах. Со значением off (по умолчанию) команда в блоке транзакции, выдающая ошибку, прерывает всю транзакцию. Для реализации режима отката транзакции за вас неявно выполняется команда SAVEPOINT непосредственно перед каждой командой в блоке транзакции, а в случае ошибки команды происходит откат к этой точке сохранения. |
| ON\_ERROR\_STOP | По умолчанию, после возникновения ошибки обработка команд продолжается. Если эта переменная установлена в значение on, обработка команд будет немедленно прекращена. В интерактивном режиме psql вернётся в командную строку; иначе psql прекратит работу с кодом возврата 3, чтобы отличить этот случай от фатальных ошибок, для которых используется код возврата 1. В любом случае выполнение всех запущенных скриптов (высокоуровневый скрипт и любые другие, которые он мог запустить) будет немедленно прекращено. Если высокоуровневая командная строка содержит несколько SQL-команд, выполнение завершится на текущей команде. |
| PORT | Порт сервера базы данных, к которому вы в данный момент подключены. Он устанавливается каждый раз, когда вы подключаетесь к базе данных (включая запуск программы), но эту переменную можно изменить или сбросить. |
| PROMPT1  PROMPT2  PROMPT3 | Они определяют, как должны выглядеть запросы psql. См. Подсказки. |
| QUIET | Установка значения on эквивалента параметру командной строки -q. Это, вероятно, не слишком полезно в интерактивном режиме. |
| SINGLELINE | Установка значения on эквивалентна параметру командной строки -S. |
| SINGLESTEP | Установка значения on эквивалентна параметру командной строки -s. |
| USER | Пользователь базы данных, с которым вы в настоящее время подключены к базе данных. Он устанавливается каждый раз, когда вы подключаетесь к базе данных (включая запуск программы), но эту переменную можно изменить или сбросить. |
| VERBOSITY | Этой переменной можно присвоить значения default, verbose, terse или sqlstate для изменения уровня детализации в сообщениях об ошибках. |

SQL-интерполяция

Ключевой особенностью переменных psql является то, что вы можете заменять («интерполировать») их в обычные операторы SQL, а также в аргументы метакоманд. Кроме того, psql предоставляет средства для корректного использования кавычек для значений переменных, используемых в качестве литералов и идентификаторов SQL. Чтобы подставить значение без кавычек, нужно добавить перед именем переменной двоеточие (:). Например:

|  |
| --- |
| testdb=> \set foo 'my\_table'  testdb=> SELECT \* FROM :foo; |

запросит таблицу my\_table. Обратите внимание, что это может быть небезопасно: значение переменной копируется буквально, поэтому оно может содержать непарные кавычки или даже метакоманды. Вы должны убедиться, что это имеет смысл.

Когда значение должно использоваться в качестве литерала или идентификатора SQL, безопаснее всего сделать так, чтобы оно было заключено в кавычки. Если значение переменной используется как SQL литерал, то после двоеточия нужно написать имя переменной в одинарных кавычках. Если значение переменной используется как SQL идентификатор, то после двоеточия нужно написать имя переменной в двойных кавычках. Эти конструкции корректно работают с кавычками и другими специальными символами, которые могут содержаться в значении переменной. Предыдущий пример безопаснее было бы написать так:

|  |
| --- |
| testdb=> \set foo 'my\_table'  testdb=> SELECT \* FROM :"foo"; |

Интерполяция переменных не будет выполняться, если SQL литералы или идентификаторы заключены в кавычки. Поэтому конструкция ':foo' не превратится во взятое в кавычки значение переменной (и это было бы небезопасно, если бы работало, так как обработка кавычек внутри значения переменной была бы некорректной).

Одним из примеров использования этого механизма является копирование содержимого файла в столбец таблицы. Сначала загрузите файл в переменную, а затем интерполируйте значение переменной как строку в кавычках:

|  |
| --- |
| testdb=> \set content `cat my\_file.txt`  testdb=> INSERT INTO my\_table VALUES (:'content'); |

(Обратите внимание, что это всё равно не будет работать, если my\_file.txt содержит байты NUL. psql не поддерживает встроенные байты NUL в значения переменных.)

Так как двоеточие может легально присутствовать в SQL-командах, попытка интерполяции (например для :name, :'name' или :"name") не выполняется, если переменная не установлена. В любом случае можно экранировать двоеточие с помощью обратной косой черты, чтобы предотвратить подстановку.

Использование двоеточия для переменных является стандартом SQL для встраиваемых языков запросов, таких как ECPG. Использование двоеточия для срезов массивов и приведения типов является расширениями PostgreSQL, что иногда может конфликтовать со стандартным использованием. Использование двоеточия и кавычек для экранирования значения переменной при подстановке в качестве SQL литерала или идентификатора — это расширение psql.

Подсказки

Вы можете настроить подсказки psql по своему усмотрению. Три переменные PROMPT1, PROMPT2 и PROMPT3 содержат строки и специальные escape-последовательности, которые задают внешний вид подсказок. PROMPT1 — это обычная подсказка, которое выдаётся, когда psql ожидает ввода новой команды. PROMPT2 выдаётся, когда во время ввода команды ожидается дополнительный ввод, например, потому что команда не была завершена точкой с запятой или не были закрыты кавячки. PROMPT3 выдаётся, когда вы запускаете SQL-команду COPY FROM STDIN и вам нужно ввести значение новой строки в терминале.

Значения этих переменных выводятся буквально, за исключением случаев, когда в них встречается знак процента (%). В зависимости от следующего символа будет подставляться определённый текст. Существуют следующие подстановки:

Таблица — Подстановки

| Подстановка | Описание |
| --- | --- |
| %M | Полное имя хоста (с доменным именем) сервера базы данных или [local], если соединение осуществляется через сокет домена UNIX, или [local:/dir/name], если при компиляции был изменён путь Unix-сокета по умолчанию. |
| %m | Имя хоста сервера базы данных, усечённое до первой точки, или [local], если соединение осуществляется через сокет домена UNIX. |
| %> | Номер порта, который прослушивает сервер базы данных. |
| %n | Имя пользователя базы данных для текущей сессии. (Это значение может меняться в течение сессии в результате выполнения команды SET SESSION AUTHORIZATION.) |
| %/ | Имя текущей базы данных. |
| %~ | Похоже на %/, но выводит тильду ~, если текущая база данных совпадает с базой данных по умолчанию. |
| %# | Если пользователь текущей сессии является суперпользователем базы данных, то выводит #, иначе >. (Это значение может меняться в течение сессии в результате выполнения команды SET SESSION AUTHORIZATION.) |
| %R | В подсказке 1 это обычно символ =, но @, если сеанс находится в неактивной ветви блока условия, либо ^ в однострочном режиме, либо !, если сеанс не подключён к базе данных (что возможно при ошибке \connect). В подсказке 2 %R заменяется символом, показывающим, почему psql ожидает дополнительный ввод: -, если команда просто ещё не была завершена, но \*, если не завершён комментарий /\* ... \*/; апостроф, если не завершена строка в апострофах; кавычки, если не завершён идентификатор в кавычках; знак доллара, если не завершена строка в долларах; либо (, если после открывающей скобки не хватает закрывающей. В подсказке 3 %R не выдаёт ничего. |
| %x | Статус транзакции: пустая строка, если не в транзакционном блоке; \*, когда в транзакционном блоке; !, когда в транзакционном блоке, в котором произошла ошибка и ?, когда состояние транзакции не определено (например, нет подключения к базе данных). |
| %digits | Подставляется символ с указанным восьмеричным кодом. |
| %:name: | Значение имени (name) переменной psql. Подробнее см. Переменные. |
| %`command` | Подставляется вывод команды (command), как и в обычной подстановке с обратными апострофами. |
| %[ ... %] | Подсказки могут содержать символы управления терминалом, которые, например, изменяют цвет, фон или стиль текста подсказки или меняют заголовок окна терминала. Чтобы редактирование строки работало правильно, эти непечатаемые управляющие символы должны быть обозначены как невидимые, заключив их в %[ and %]. В подсказке может появиться несколько таких пар. Например,   |  | | --- | | testdb=> \set PROMPT1 '%[%033[1;33;40m%]%n@%/%R%[%033[0m%]%#' |   приводит к появлению выделенного жирным шрифтом (1;) подсказки жёлтогй на черном (33;40) на терминалах, совместимых с VT100 и поддерживающих цвет. Чтобы вставить знак процента в подсказку, напишите %%. Подсказки по умолчанию: '%/%R%# ' для PROMPT1 и PROMPT2, '>> ' для PROMPT3. |

Редактирование командной строки

psql использует библиотеку readline для удобного редактирования командной строки. История команд автоматически сохраняется при выходе из psql и перезагружается при запуске psql. Также поддерживается завершение по табуляции, хотя логика завершения не претендует на роль анализатора SQL. Запросы, генерируемые с помощью клавиши Tab, также могут конфликтовать с другими командами SQL, например, SET TRANSACTION ISOLATION LEVEL. Если по какой-то причине вам не нравится завершение по табуляции, вы можете отключить его в файле .inputrc в своём домашнем каталоге:

|  |
| --- |
| $if psql  set disable-completion on  $endif |

Переменные окружения

Таблица — Переменные окружения

| Параметр | Описание |
| --- | --- |
| COLUMNS | Если \pset columns равно нулю, управляет шириной формата вывода wrapped, а также определяет, нужно ли использовать постраничник и нужно ли переключаться в вертикальный формат в режиме expanded auto. |
| PAGER | Если результат запроса не помещается на экране, он пропускается через эту программу. Обычно это more или less. От использования постраничника можно отказаться, присвоив переменной PSQL\_PAGER или PAGER пустую строку, либо изменив соответствующие параметры с помощью команды \pset. |
| PGDATABASE | — |
| PGHOST | — |
| PGPORT | — |
| PGUSER | Параметры подключения по умолчанию. |
| PSQL\_EDITOR | — |
| EDITOR | — |
| VISUAL | Редактор, используемый командами \e, \ef. Эти переменные рассматриваются в том же порядке; в силу вступает первое установленное значение. Если ни одна из переменных не установлена, по умолчанию в Unix-системах используется vi, а в Windows — notepad.exe. |
| PSQL\_EDITOR\_LINENUMBER\_ARG | Если в командах \e, \ef указан номер строки, эта переменная задаёт аргумент командной строки, с которым номер строки может быть передан в редактор. Например, для редакторов Emacs и vi это знак плюс. Добавьте в конец значения пробел, если он требуется для отделения имени аргумента от номера строки. Примеры:   |  | | --- | | PSQL\_EDITOR\_LINENUMBER\_ARG='+'  PSQL\_EDITOR\_LINENUMBER\_ARG='--line ' |   Значение по умолчанию + в Unix-подобных системах (соответствует редактору по умолчанию vi и многим другим распространённым редакторам). На платформе Windows нет значения по умолчанию. |
| PSQL\_HISTORY | Альтернативное расположение файла истории команд. Допускается использование тильды (~). |
| PSQLRC | Альтернативное расположение файла .psqlrc пользователя. Допускается использование тильды (~). |
| SHELL | Команда операционной системы, выполняемая метакомандой \!. |
| TMPDIR | Каталог для хранения временных файлов. По умолчанию это /tmp. |

Файлы

Таблица — Файлы

| Файл | Описание |
| --- | --- |
| psqlrc и ~/.psqlrc | Если не указан параметр -X или -c, psql пытается считать и выполнить команды из общесистемного файла запуска (psqlrc), а затем из персонального файла запуска пользователя (~/.psqlrc) после подключения к базе данных, но перед подключением обычных команд. Эти файлы можно использовать для настройки клиента и/или сервера, обычно с помощью команд \set и SET.  Общесистемный файл запуска называется psqlrc и ищется в каталоге установки "system configuration", который наиболее надежно определяется путем выполнения pg\_config --sysconfdir. По умолчанию этот каталог будет расположен ../etc/ относительно каталога, содержащего исполняемые файлы RT.Warehouse. Имя этого каталога можно задать явно с помощью переменной среды PGSYSCONFDIR.  Персональный файл запуска пользователя называется .psqlrc и ищется в домашнем каталоге вызывающего пользователя. В Windows, в которой отсутствует такая концепция, персональный файл запуска называется %APPDATA%\postgresql\psqlrc.conf. Расположение персонального файла запуска пользователя можно указать явно с помощью переменной среды PSQLRC.  Как общесистемный файл запуска, так и персональный файл запуска пользователя можно привязать к конкретной версии psql, добавив к имени файла дефис и основной или дополнительный номер релиза PostgreSQL, например ~/.psqlrc-9.4. При наличии нескольких файлов, файл с более детальным номером версии будет иметь предпочтение. |
| .psql\_history | История командной строки хранится в файле ~/.psql\_history или %APPDATA%\postgresql\psql\_history в Windows.  Расположение файла истории можно указать явно через переменную среды PSQL\_HISTORY. |

Примечания

psql лучше всего работает с серверами той же или более старой основной версии. Метакоманды особенно часто не срабатывают, если сервер имеет более новую версию, чем сам psql. Однако метакоманды семейства \d должны работать со старыми версиями серверов, но не обязательно с серверами новее, чем сам psql. Общая функциональность запуска SQL-команд и отображения результатов запросов также должна работать на серверах с более новой основной версией, но это не гарантируется во всех случаях.

Если вы хотите использовать psql для подключения к нескольким серверам разных основных версий, рекомендуется использовать последнюю версию psql. Также можно собрать копии psql от каждой основной версии и использовать ту, которая соответствует версии сервера. Но на практике в этих дополнительных сложностях нет необходимости.

Примечания для пользователей Windows

psql построен как консольное приложение. Поскольку в Windows консольные окна используют кодировку символов отличную от той, что используется для остальной системы, нужно проявить особую осторожность при использовании 8-битных символов. Если psql обнаружит проблемную кодовую страницу консоли, он предупредит вас при запуске. Чтобы изменить кодовую страницу консоли, необходимы две вещи:

Установите кодовую страницу, введя:

|  |
| --- |
| cmd.exe /c chcp 1252 |

1252 — это кодировка латинского алфавита, используемая Microsoft Windows для английского и некоторых других западных языков. Если вы используете Cygwin, вы можете поместить эту команду в /etc/profile.

Установите для консольного шрифта значение Lucida Console, поскольку растровый шрифт не работает с кодовой страницей ANSI.

Примеры

Запустить psql в интерактивном режиме:

|  |
| --- |
| psql -p 54321 -U sally mydatabase |

В интерактивном режиме psql распределить команду по нескольким строкам ввода. Обратите внимание на изменение подсказки:

|  |
| --- |
| testdb=> CREATE TABLE my\_table (  testdb(> first integer not null default 0,  testdb(> second text)  testdb-> ;  CREATE TABLE |

Посмотрите определение таблицы:

|  |
| --- |
| testdb=> \d my\_table  Table "my\_table"  Attribute | Type | Modifier  -----------+---------+--------------------  first | integer | not null default 0  second | text | |

Запустить psql в неинтерактивном режиме, передав файл, содержащий SQL-команды:

|  |
| --- |
| psql -f /home/gpadmin/test/myscript.sql |

pxf

Управляет конфигурацией PXF и инстансом службы PXF на локальном хосте RT.Warehouse.

Синтаксис

|  |
| --- |
| pxf <command> [<option>] |

Где <command> это:

|  |
| --- |
| cluster  help  init  reset  restart  start  status  stop  sync  version |

Описание

Утилита pxf управляет конфигурацией PXF и инстансом службы PXF на локальном хосте RT.Warehouse.

Вы можете инициализировать или сбросить PXF на мастере, резервном мастере или определённом хосте сегмента. Вы также можете синхронизировать конфигурацию PXF от мастера к этим хостам.

Вы можете запустить, остановить или перезапустить инстанс службы PXF на определённом хосте сегмента или отобразить состояние инстанса службы PXF, запущенного на хосте сегмента.

(Используйте команду pxf cluster для инициализации или сброса PXF на всех хостах, синхронизации конфигурации PXF с кластером RT.Warehouse или для запуска, остановки или отображения состояния инстанса службы PXF на всех хостах сегмента в кластере.)

Команды

Таблица — Команды pxf

| Команда | Описание |
| --- | --- |
| cluster | Управление конфигурацией PXF и инстансом службы PXF на всех хостах RT.Warehouse. См. pxf cluster. |
| help | Отображение справочного сообщения утилиты управления pxf и дальнейшее завершение. |
| init | Инициализация инстанса службы PXF на хосте. Когда вы инициализируете PXF, вы должны идентифицировать каталог конфигурации пользователя PXF через переменную среды с именем $PXF\_CONF. Если вы не установите $PXF\_CONF перед инициализацией PXF, PXF предложит вам принять или отклонить каталог конфигурации пользователя по умолчанию, $HOME/pxf, во время процесса инициализации. См. Параметры. |
| reset | Сброс инстанса службы PXF, запущенный на хосте. Сброс удаляет файлы и каталоги среды выполнения PXF и возвращает PXF в неинициализированное состояние. Перед сбросом PXF на хосте необходимо остановить инстанс службы PXF, работающий на хосте сегмента. |
| restart | Перезапуск инстанса службы PXF, запущенного на хосте сегмента. |
| start | Запуск инстанса службы PXF на хосте сегмента. |
| status | Отображение состояния инстанса службы PXF, запущенного на хосте сегмента. |
| stop | Остановка инстанса службы PXF, работающего на хосте сегмента. |
| sync | Синхронизация конфигурации PXF от мастера к определённому резервному мастеру или хосту сегмента RT.Warehouse. Вы должны запустить pxf sync на хосте мастера. См. Параметры. |
| version | Отображение версии PXF и дальнейшее завершение. |

Параметры

Команда pxf init принимает следующие параметры.

Таблица — Параметры pxf int

| Параметр | Описание |
| --- | --- |
| -y | Не запрашивать, использует расположение каталога по умолчанию $PXF\_CONF, если переменная среды не установлена. |

Команда pxf reset принимает следующие параметры.

Таблица — Параметры pxf reset

| Параметр | Описание |
| --- | --- |
| -f | –force | Не запрашивать перед сбросом инстанса службы PXF; сбросить без вмешательства пользователя. |

Команда pxf sync, выполняемая на хосте мастера RT.Warehouse, принимает следующие параметры.

Таблица — Параметры pxf sync

| Параметр | Описание |
| --- | --- |
| <gphost> | Хост RT.Warehouse, с которым нужно синхронизировать конфигурацию PXF. <gphost> должен идентифицировать хост резервного мастера или хост сегмента. |

Примеры

Запустить инстанс службы PXF на хосте локального сегмента:

|  |
| --- |
| $ $GPHOME/pxf/bin/pxf start |

pxf cluster

Управляет конфигурацией PXF и инстансом службы PXF на всех хостах RT.Warehouse.

Синтаксис

|  |
| --- |
| pxf cluster <command> |

Где <command> это:

|  |
| --- |
| help  init  reset  start  status  stop  sync |

Описание

Команда утилиты pxf cluster управляет PXF на хостах мастера, резервного мастера и всех сегментов RT.Warehouse. Вы можете использовать эту утилиту для следующих действий:

1. Инициализация конфигурации PXF на всех хостах кластера RT.Warehouse.

Сброс инстанса службы PXF на всех хостах в неинициализированное состояние.

Запуск и остановка инстанса службы PXF на всех хостах сегментов.

Отображение состояния инстанса службы PXF на всех хостах сегментов.

Синхронизация конфигурации PXF хоста мастера RT.Warehouse с резервным мастером и со всеми хостами сегментов.

Для pxf cluster требуется работающий кластер RT.Warehouse. Вы должны запустить эту утилиту на хосте мастера RT.Warehouse.

(Если вы хотите управлять инстансом службы PXF на определённом хосте сегмента, используйте утилиту pxf. См. pxf.)

Команды

Таблица — Команды pxf cluster

| Команда | Описание |
| --- | --- |
| help | Отображение справочного сообщения pxf cluster и дальнейшее завершение. |
| init | Инициализация инстанса службы PXF на мастере, резервном мастере и на всех хостах сегментов. При инициализации PXF в кластере RT.Warehouse необходимо определить каталог конфигурации пользователя PXF с помощью переменной среды с именем $PXF\_CONF. Если вы не установите $PXF\_CONF перед инициализацией PXF, PXF вернёт ошибку. |
| reset | Сброс инстанса службы PXF на мастере, резервном мастере и на всех хостах сегментов. Сброс удаляет файлы и каталоги среды выполнения PXF и возвращает PXF в неинициализированное состояние. Перед сбросом PXF в кластере RT.Warehouse необходимо остановить инстанс службы PXF, работающий на каждом хосте сегмента. |
| start | Запуск инстанса службы PXF на всех хостах сегментов. |
| status | Отображение состояния инстанса службы PXF на всех хостах сегментов. |
| stop | Остановка инстанса службы PXF на всех хостах сегментов. |
| sync | Синхронизация конфигурации PXF от мастера к резервному мастеру и ко всем хостам сегментов RT.Warehouse. Если вы обновили конфигурацию пользователя PXF или добавили файлы JAR, вы также должны перезапустить PXF после синхронизации конфигурации PXF. |

Примеры

Остановить инстанс службы PXF на всех хостах сегментов:

|  |
| --- |
| $ $GPHOME/pxf/bin/pxf cluster stop |

reindexdb

Перестраивает индексы в базе данных.

Синтаксис

|  |
| --- |
| reindexdb [*connection-option ...*] [--table | -t *table* ]  [--index | -i *index* ] [*dbname*]  reindexdb [*connection-option ...*] --all | -a  reindexdb [*connection-option ...*] --system | -s [*dbname*]  reindexdb -? | --help  reindexdb -V | --version |

Описание

reindexdb — это утилита для восстановления индексов в RT.Warehouse.

reindexdb — это оболочка для SQL-команды REINDEX. Эффективной разницы между переиндексированием баз данных с помощью этой утилиты и других методов доступа к серверу нет.

Параметры

Таблица — Параметры reindexdb

| Параметр | Описание |
| --- | --- |
| -a | --all | Переиндексация всех баз данных. |
| [-d] *dbname* | [--dbname=]*dbname* | Задаёт имя переиндексируемой базы данных. Если данный параметр не указан и -all не используется, имя базы данных считывается из переменной среды PGDATABASE. Если параметр не установлен, используется имя пользователя, указанное для соединения. |
| -e | --echo | Вызов команд, которые создаёт и отправляет на сервер reindexdb. |
| -i *index* | --index=*index* | Пересоздать только индекс. |
| -q | --quiet | Не отображать ответ. |
| -s | --system | Переиндексировать системные каталоги. |
| -t *table* | --table=*table* | Переидексировать только таблицу. Несколько таблиц можно переиндексировать, написав несколько ключей -t. |
| -V | --version | Вывод версии reindexdb и выход. |
| -? | --help | Вывод справки об аргументах командной строки reindexdb и выход. |

Таблица — Параметры подключения reindexdb

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Задаёт имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, данные считываются из переменной среды PGHOST или по умолчанию используется localhost. |
| -p port | --port=port | Задаёт TCP-порт, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, данные считываются из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, данные считываются из переменной среды PGUSER или по умолчанию используется текущее имя пользователя системы. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими способами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в batch-заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |
| --maintenance-db=*dbname* | Задает имя базы данных для подключения, чтобы определить, какие другие базы данных следует переиндексировать. Если параметр не указан, будет использоваться база данных postgres, а если она не существует, будет использоваться template1. |

Примечания

reindexdb может потребоваться несколько раз подключиться к мастер-серверу, каждый раз запрашивая пароль. В таких случаях удобно иметь файл ~/.pgpass.

Примеры

Переиндексировать базу данных mydb:

|  |
| --- |
| reindexdb mydb |

Переиндексировать таблицу foo и индекс bar в базе данных с именем abcd:

|  |
| --- |
| reindexdb --table foo --index bar abcd |

vacuumdb

Сборщик мусора и анализ базы данных.

Синтаксис

|  |
| --- |
| vacuumdb [*connection-option...*] [--full | -f] [--freeze | -F] [--verbose | -v]  [--analyze | -z] [--analyze-only | -Z] [--table | -t *table* [( *column* [,...] )] ] [*dbname*]  vacuumdb [*connection-option...*] [--all | -a] [--full | -f] [-F]  [--verbose | -v] [--analyze | -z]  [--analyze-only | -Z]  vacuumdb -? | --help  vacuumdb -V | --version |

Описание

vacuumdb — это утилита для очистки базы данных RT.Warehouse. vacuumdb также будет генерировать внутреннюю статистику, используемую оптимизатором запросов к RT.Warehouse.

vacuumdb — это оболочка для SQL-команды VACUUM. Эффективной разницы между очисткой баз данных с помощью этой утилиты и других методов доступа к серверу нет.

Параметры

Таблица — Параметры vacuumdb

| Параметр | Описание |
| --- | --- |
| -a | --all | Удаление всех баз данных. |
| [-d] *dbname* | [--dbname=]*dbname* | Имя базы данных для очистки. Если параметр не указан и -a (или --all) не используется, имя базы данных считывается из переменной среды PGDATABASE. Если он не установлен, используется имя пользователя, указанное для соединения. |
| -e | --echo | Вызов команд, которые создаёт и отправляет на сервер reindexdb. |
| -f | --full | Выбор полной очистки, которая может освободить больше места, но занимает гораздо больше времени и блокирует таблицу.  **Предупреждение**. VACUUM FULL не рекомендуется в RT.Warehouse. |
| -F | --freeze | Заморозить информацию о транзакции строки. |
| -q | --quiet | Не отображать ответ. |
| -t *table* [(*column*)] | --table= *table* [(*column*)] | Очистка или анализ только выбранной таблицы. Имена столбцов можно указывать только вместе с параметрами --analyze или --analyze-all. Несколько таблиц можно очистить, написав несколько ключей -t. Если вы указываете столбцы, вам, вероятно, придётся избегать скобок в оболочке. |
| -v | --verbose | Вывод подробной информации во время обработки. |
| -z | --analyze | Сбор статистики для использования планировщиком запросов. |
| -Z | --analyze-only | Вычисление статистики только для использования планировщиком запросов (без вакуума). |
| -V | --version | Вывод версии vacuumdb и выход. |
| -? | --help | Вывод справки об аргументах командной строки vacuumdb и выход. |

Таблица — Параметры подключения vacuumdb

| Параметр | Описание |
| --- | --- |
| -h *host* | --host=*host* | Задаёт имя хоста машины, на которой работает мастер-сервер RT.Warehouse. Если параметр не указан, данные считываются из переменной среды PGHOST или по умолчанию используется localhost. |
| -p port | --port=port | Задаёт TCP-порт, на котором мастер-сервер RT.Warehouse прослушивает соединения. Если параметр не указан, данные считываются из переменной среды PGPORT или по умолчанию 5432. |
| -U *username* | --username=*username* | Имя роли базы данных для подключения. Если параметр не указан, данные считываются из переменной среды PGUSER или по умолчанию используется текущее имя пользователя системы. |
| -w | --no-password | Никогда не запрашивать пароль. Если сервер требует аутентификации по паролю, а пароль недоступен другими способами, такими как файл .pgpass, попытка подключения не удастся. Этот параметр может быть полезен в batch-заданиях и скриптах, где нет пользователя, который мог бы ввести пароль. |
| -W | --password | Принудительный ввод пароля. |
| --maintenance-db=*dbname* | Задает имя базы данных для подключения, чтобы определить, какие другие базы данных следует переиндексировать. Если параметр не указан, будет использоваться база данных postgres, а если она не существует, будет использоваться template1. |

Примечания

vacuumdb может потребоваться несколько раз подключиться к мастер-серверу, каждый раз запрашивая пароль. В таких случаях удобно иметь файл ~/.pgpass.

Примеры

Очистить базу данных test:

|  |
| --- |
| vacuumdb test |

Очистить и проанализировать базу данных с именем bigdb:

|  |
| --- |
| vacuumdb --analyze bigdb |

Чтобы очистить одну таблицу foo в базе данных с именем mydb и проанализировать отдельный столбец bar таблицы. Обратите внимание на кавычки вокруг имён таблиц и столбцов, чтобы избежать скобок в оболочке:

|  |
| --- |
| vacuumdb --analyze --verbose --table 'foo(bar)' mydb |