

Авторы: Егор Рогов, Павел Лузанов



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

# Темы

Понятие логической резервной копии

Копирование и восстановление отдельных таблиц

Копирование и восстановление баз данных

Копирование и восстановление кластера

# Логическая копия

Команды SQL для создания объектов и наполнения данными

+ можно сделать копию отдельного объекта или отдельной базы
+ можно восстановиться на другой версии или архитектуре (не требуется двоичная совместимость)
− невысокая скорость работы
− восстановление только на момент создания резервной копии



Логическое резервирование — набор команд SQL, восстанавливающих кластер (или базу данных, или отдельную таблицу) с нуля: создаются необходимые объекты и наполняются данными.

Команды можно выполнить на другой версии СУБД (при наличии совместимости на уровне команд) или на другой платформе и архитектуре (не требуется двоичная совместимость).

В частности, логическую резервную копию можно использовать для долговременного хранения: ее можно будет восстановить и после обновления сервера на новую версию.

Однако для большой базы команды могут выполняться очень долго. Восстановить систему из логической копии можно ровно на момент начала резервного копирования.

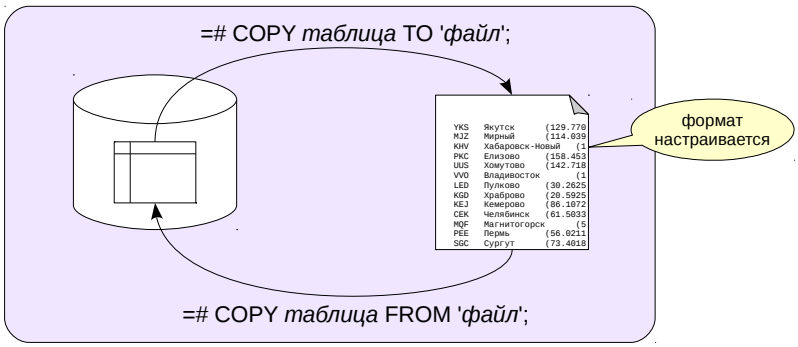
https://postgrespro.ru/docs/postgresql/10/backup-dump.html

# Копия таблицы

Серверный вариант: SQL-команда COPY

Клиентский вариант: команда psql \copy

# SQL-команда COPY




файл в ФС сервера и доступен владельцу экземпляра PostgreSQL
можно ограничить столбцы (или использовать произвольный запрос)
при восстановлении строки добавляются к имеющимся в таблице



Если требуется сохранить только содержимое одной таблицы, можно воспользоваться командой COPY.

Команда позволяет записать таблицу (или часть столбцов таблицы, или даже результат произвольного запроса) либо в файл, либо на консоль, либо на вход какой-либо программе. При этом можно указать ряд параметров, таких как формат (текстовый, CSV или двоичный), разделитель полей, текстовое представление NULL и т. п.

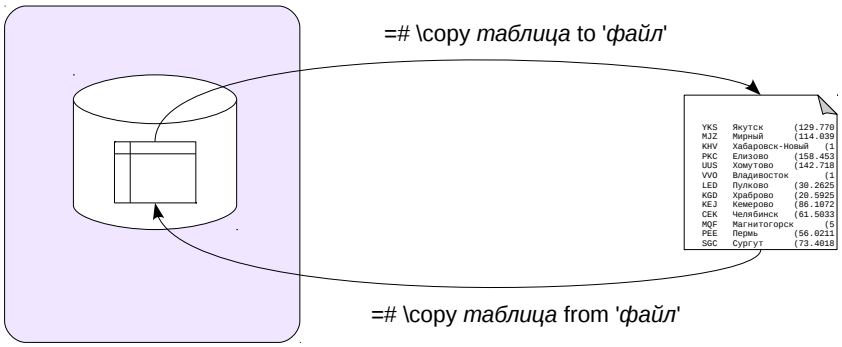
Другой вариант команды, наоборот, считывает из файла или из консоли строки с полями и записывает их в таблицу. Таблица при этом не очищается, новые строки добавляются к уже существующим.

Команда COPY работает существенно быстрее, чем аналогичные команды INSERT — клиенту не нужно много раз обращаться к серверу, а серверу не нужно много раз анализировать команды.

Тонкость: при выполнении команды COPY FROM не применяются правила (rules), хотя ограничения целостности и триггеры выполняются.

https://postgrespro.ru/docs/postgresql/10/sql-copy

# Команда psql \copy




файл в ФС клиента и доступен пользователю ОС, запустившему psql
происходит пересылка данных между клиентом и сервером
синтаксис и возможности аналогичны команде COPY



В psql существует клиентский вариант команды COPY с аналогичным синтаксисом.

В отличие от серверного варианта COPY, который является командой SQL, клиентский вариант — это команда psql.

Указание имени файла в команде SQL соответствует файлу на сервере БД. У пользователя, под которым работает PostgreSQL (обычно postgres), должен быть доступ к этому файлу.

В клиентском же варианте обращение к файлу происходит на клиенте, а на сервер передается только содержимое.

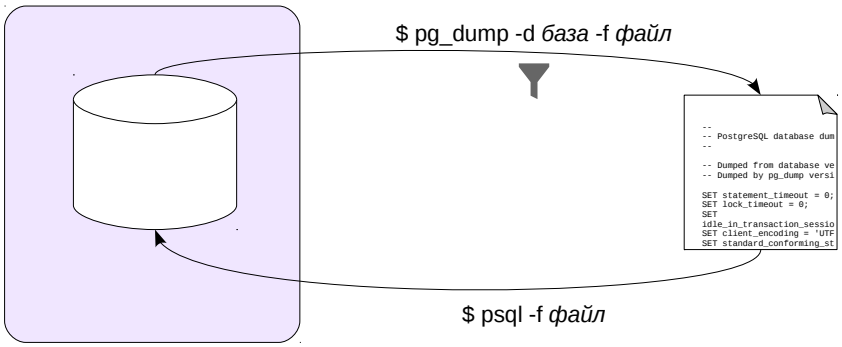
https://postgrespro.ru/docs/postgresql/10/app-psql

# Копия базы данных

Утилита pg_dump

Возможные форматы выгрузки

# Копия базы данных




формат: команды SQL

при выгрузке можно выбрать отдельные объекты базы данных

новая база должна быть создана из шаблона template0

заранее должны быть созданы роли и табличные пространства

после загрузки имеет смысл выполнить ANALYZE



Для создания полноценной резервной копии базы данных используется утилита pg_dump.

Если не указать имя файла (-f, --file), то утилита выведет результат на консоль. А результатом является скрипт, предназначенный для psql, который содержит команды для создания необходимых объектов и наполнения их данными.

Дополнительными ключами утилиты можно ограничить набор объектов: выбрать указанные таблицы, или все объекты в указанных схемах, или наложить другие фильтры.

Чтобы восстановить объекты из резервной копии, достаточно выполнить полученный скрипт в psql.

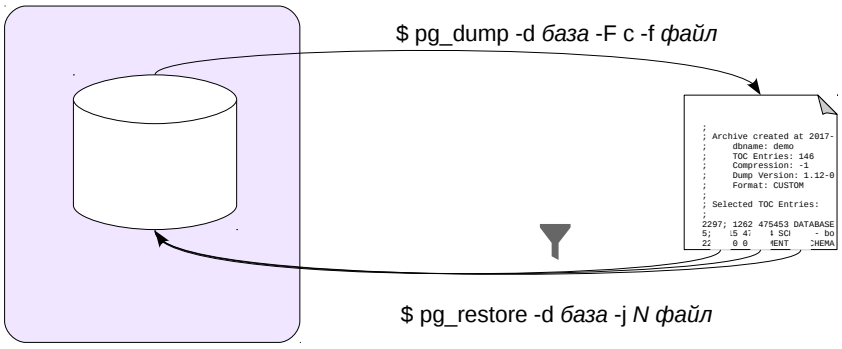
Следует иметь в виду, что базу данных для восстановления надо создавать из шаблона template0, так как все изменения, сделанные в template1, также попадут в резервную копию.

Кроме того, заранее должны быть созданы необходимые роли и табличные пространства. Поскольку эти объекты не относятся к конкретной БД, они не будет выгружены в резервную копию.

После восстановления базы имеет смысл выполнить команду ANALYZE: она соберет статистику, необходимую оптимизатору для планирования запросов.

https://postgrespro.ru/docs/postgresql/10/app-pgdump

# Формат custom




внутренний формат с оглавлением

отдельные объекты базы данных можно выбрать на этапе восстановления

возможна загрузка в несколько параллельных потоков



Утилита pg_dump позволяет указать формат резервной копии. По умолчанию это plain — простые команды для psql.

Формат custom (-F c, --format=custom) создает резервную копию в специальном формате, содержащем не только объекты, но и оглавление. Наличие оглавления позволяет выбирать объекты для восстановления не при создании копии, а непосредственно при восстановлении.

Файл формата custom по умолчанию сжат.

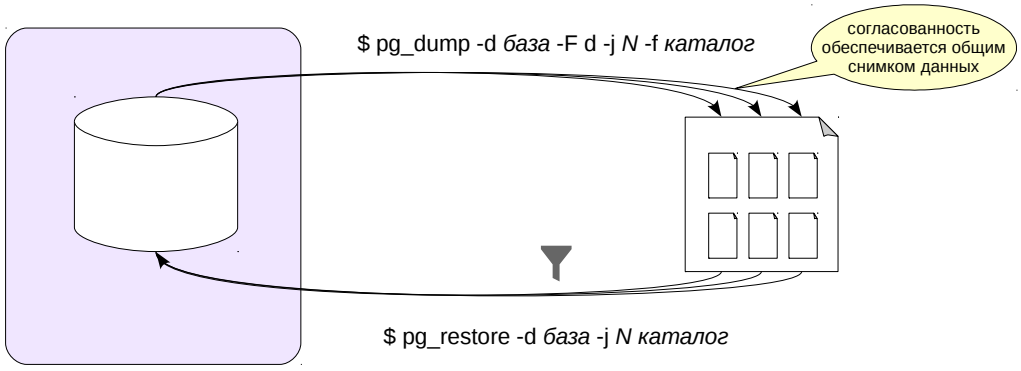
Для восстановления потребуется другая утилита — pg_restore. Она читает файл и преобразует его в команды psql. Если не указать явно имя базы данных (в ключе -d), то команды будут выведены на консоль. Если же база данных указана — утилита соединится с этой БД и выполнит команды без участия psql.

Чтобы восстановить только часть объектов, можно воспользоваться одним из двух подходов. Во-первых, можно ограничить объекты аналогично тому, как они ограничиваются в pg_dump. Вообще, утилита pg_restore понимает многие параметры из репертуара pg_dump.

Во-вторых, можно получить из оглавления список объектов, содержащихся в резервной копии (ключ --list). Затем этот список можно отредактировать вручную, удалив ненужное и передать измененный список на вход pg_restore (ключ --use-list).

https://postgrespro.ru/docs/postgresql/10/app-pgrestore.html

# Формат directory




каталог с оглавлением и отдельными файлами на каждый объект базы
отдельные объекты базы данных можно выбрать на этапе восстановления
и выгрузка, и загрузка возможны в несколько параллельных потоков



Еще один формат резервной копии — directory. В таком случае будет создан не один файл, а каталог, содержащий объекты и оглавление. По умолчанию файлы внутри каталога будут сжаты.

Преимущество перед форматом custom состоит в том, что такая резервная копия может создаваться параллельно в несколько потоков (количество указывается в ключе -j, --jobs).

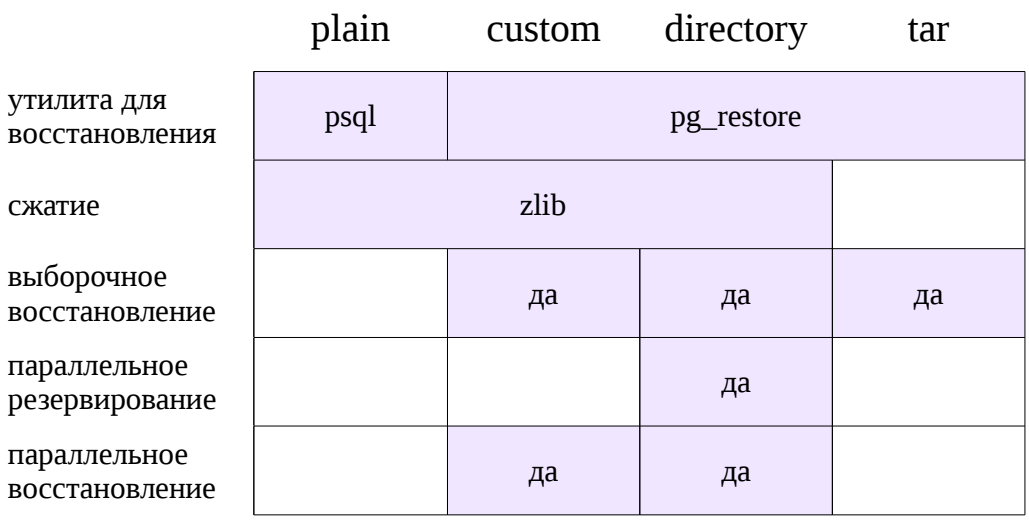
Разумеется, несмотря на параллельное выполнение, копия будет содержать согласованные данные. Это обеспечивается общим снимком данных для всех параллельно работающих процессов.

https://postgrespro.ru/docs/postgresql/10/functions-admin.html#FUNCTIONS-SNAPSHOT-SYNCHRONIZATION

Восстановление также возможно в несколько потоков (это работает и для формата custom).

В остальном возможности по работе с форматом directory не отличается от ранее рассмотренных: поддерживаются те же ключи и подходы.

# Сравнение форматов





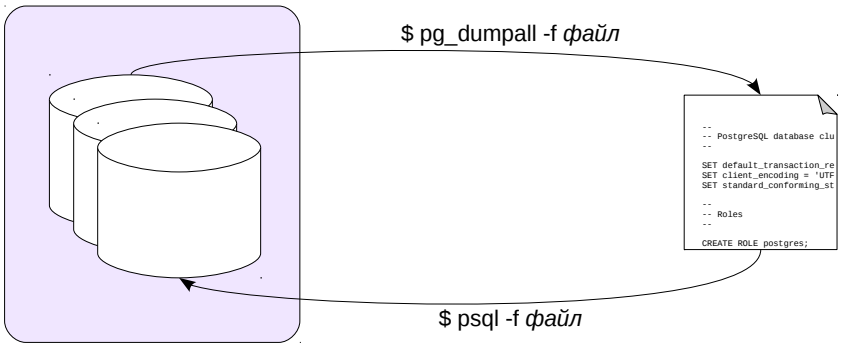

В приведенной таблице разные форматы сравниваются с точки зрения предоставляемых ими возможностей.

Отметим, что имеется и четвертый формат — tar. Он не рассматривался, так как не привносит ничего нового и не дает преимуществ перед другими форматами. Фактически он соответствует созданию tar-файла из каталога в формате directory, но не поддерживает сжатие и параллелизм.

# Копия кластера БД

Утилита pg_dumpall

# Копия кластера БД




формат: команды SQL

выгружает весь кластер, включая роли и табличные пространства

пользователь должен иметь доступ ко всем объектам кластера

не поддерживает параллельную выгрузку



Чтобы создать резервную копию всего кластера, включая роли и табличные пространства, можно воспользоваться утилитой pg_dumpall.

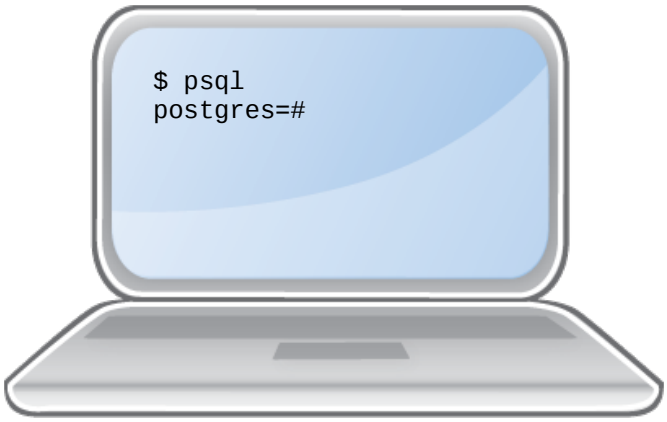
Поскольку pg_dumpall требуется доступ ко всем объектам всех БД, имеет смысл запускать ее от имени суперпользователя. Утилита по очереди подключается к каждой БД кластера и выгружает информацию с помощью pg_dump. Кроме того, она сохраняет и данные, относящиеся к кластеру в целом.

Чтобы начать работу, утилите требуется подключиться хотя бы к какой-то базе данных. По умолчанию выбирается postgres или template1, но можно указать и другую.

Результатом работы pg_dumpall является скрипт для psql. Другие форматы не поддерживаются. Это означает, что pg_dumpall не поддерживает параллельную выгрузку данных, что может оказаться проблемой при больших объемах данных. В таком случае можно воспользоваться ключом --globals-only, чтобы выгрузить только роли и табличные пространства, а сами базы данных выгрузить отдельно с помощью pg_dump в параллельном режиме.

https://postgrespro.ru/docs/postgresql/10/app-pg-dumpall

# Демонстрация



# Итоги

Логическое резервирование позволяет сделать копию всего кластера, базы данных или отдельных объектов

Хорошо подходит

- для данных небольшого объема
- для длительного хранения, за время которого меняется версия сервера
- для миграции на другую платформу

Плохо подходит

- для восстановления после сбоя с минимальной потерей данных

# Практика

1. На первом сервере создайте несколько баз данных.
   В них создайте различные объекты (например, таблицы, представления, индексы).
2. Сделайте копию только глобальных объектов кластера с помощью утилиты pg_dumpall.
3. Сделайте копии каждой базы данных кластера с помощью утилиты pg_dump в параллельном режиме.
4. Полностью восстановите кластер на другом сервере, используя созданные резервные копии.
5. Попробуйте подобрать такие данные и параметры команды COPY, чтобы созданную копию таблицы невозможно было загрузить.