

Авторы: Алексей Береснев, Илья Баштанов, Павел Толмачев



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

# Темы

Резервное копирование

Локальная работа pg_probackup

# Резервное копирование

Угрозы и требования

Характеристики

Общая классификация

Инструменты PostgreSQL

Возможности pg_probackup

# Угрозы и требования

Резервное копирование необходимо для отражения угроз

- отказ оборудования и сбои программного обеспечения
- порча данных вследствие человеческих ошибок
- негативное воздействие окружающей среды
- криминальная активность
- и другие

Обычные требования к резервному копированию

- небольшое время резервного копирования
- минимизация места для хранения копий
- быстрое восстановление из резервной копии



Любая реальная система рано или поздно столкнется с отказом, связанным с исчезновением или повреждением данных. Резервное копирование позволяет снизить риск потери данных.

Основные причины, приводящие к возможной потере данных:

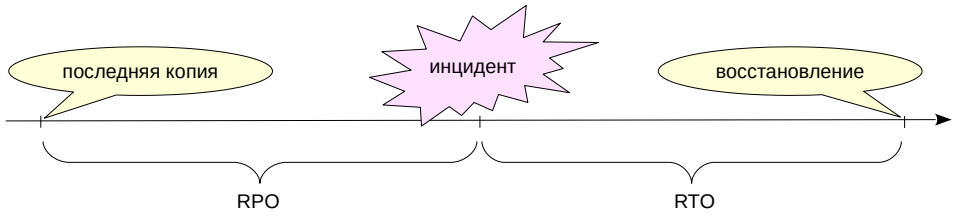
- природные и антропогенные катастрофы;
- отказы и сбои оборудования;
- ошибки в программном обеспечении;
- намеренная порча или кража оборудования;
- взлом или компрометация компьютерных систем;
- человеческие ошибки.

В зависимости от ценности защищаемой информации должна быть построена более или менее сложная система резервного копирования, которое обычно производится по плану. Чтобы обеспечить максимальную защиту данных, мероприятия, связанные с резервным копированием должны включать:

- план резервного копирования;
- процедуры проверки целостности и исправности копий;
- систему хранения резервных копий в надлежащих условиях;
- план восстановления системы с помощью резервных копий;
- средства защиты системы резервного копирования и самих копий.

Желательно обеспечить минимальное время копирования и восстановления и ограничить пространство для хранения копий.

# Характеристики




Идеальные характеристики

| | |
|---|---|
| отношение времени работы системы к общему времени | → 100 % |
| RTO — целевое время восстановления | → 0 |
| RPO — целевая точка восстановления | → 0 |



Идеальная система никогда не простаивает и отношение времени продуктивной работы системы к общему времени ее эксплуатации равно 100%.

Однако в реальных системах инциденты, требующие восстановления данных из резервных копий, рано или поздно происходят.

Оценка максимального периода времени, за который данные могут быть потеряны вследствие инцидента, связана с периодичностью и способами выполнения резервного копирования. Такая оценка выражается показателем Recovery Point Objective (RPO).

После обнаружения инцидента требуется время для восстановления системы. Показатель Recovery Time Objective (RTO) указывает целевое время, по истечении которого система восстанавливается для обслуживания клиентов.

В сложных системах процесс восстановления системы проходит через несколько промежуточных стадий, что, например, может быть вызвано восстановлением из резервных копий нескольких различных экземпляров БД.

# Общая классификация

По вовлеченности администратора

- неконтролируемое (unattended)
- контролируемое (attended)

По плановости

- незапланированное (non-scheduled)
- календарное (scheduled)

По полноте резервирования данных

- полное (full)
- инкрементальное (incremental)



Принятая в системах резервного копирования классификация подразделяет резервное копирование по следующим критериям:

- по степени вовлеченности администратора в процесс резервного копирования на требующее контроля и автоматическое неконтролируемое копирование;
- является ли резервное копирование незапланированным, или же оно выполняется в соответствии с календарным расписанием;
- помещаются ли в резервную копию все данные (возможно, в сжатом виде), или же в конкретную копию помещаются лишь данные, измененные с момента предыдущего резервного копирования.

Незапланированное резервное копирование осуществляется с помощью команды администратора (attended non-scheduled backup).

Однако чаще всего заранее создается календарный план резервного копирования, в котором фиксируется периодичность автоматического выполнения полного (full) или инкрементального (incremental) резервного копирования. Другое название инкрементальных копий — разностные копии (в конкретных системах резервного копирования понятия инкрементальных и разностных копий могут отличаться). Инкрементальные копии содержат данные, измененные с момента последнего резервного копирования, поэтому размер такой копии меньше, чем у полной.

СУБД вносят свою дополнительную классификацию в силу специфики.

# Инструменты PostgreSQL

COPY, pg_dump, pg_dumpall

- высокое RPO

pg_basebackup

- нет каталога и хранилища резервных копий
- нет политик хранения и удержания копий
- нет инкрементального копирования
- невозможна параллельная работа



Собственные инструменты PostgreSQL позволяют выполнять как логическое копирование всего кластера или отдельных объектов, так и физическое копирование кластера. Эти инструменты подробно рассмотрены в курсе DBA3.

Однако имеющиеся в PostgreSQL инструменты неудобны для промышленного использования.

Средства логического резервирования не удовлетворяют поставленным требованиям из-за крайне высокого RPO.

Средства физического резервирования не предоставляют возможности ведения каталога резервных копий, который содержал бы информацию о копиях (метаданные). Например, в каталоге резервных копий можно хранить тип резервной копии (полная или инкрементальная), время и способ ее создания. Такой каталог позволяет выполнять проверку исправности резервных копий, а также упрощать и ускорять восстановление данных.

Другой недостаток штатных инструментов — отсутствие политик, определяющих правила хранения копий и их последующего удаления.

Кроме того, производительность штатной утилиты pg_basebackup из-за ее однопоточности явно недостаточна для резервирования больших объемов данных.

# Возможности pg_probackup

Централизованный каталог копий и политики хранения

Резервирование на локальный или удаленный сервер

Постраничное инкрементальное копирование

Архивирование WAL

Восстановление выбранных баз данных

Параллельное резервирование и восстановление

Сжатие данных и поддержка CFS

Резервирование файлов вне каталога данных

Поддержка облачных хранилищ S3



Утилита pg_probackup — система резервного копирования и восстановления корпоративного уровня, разработанная компанией Postgres Professional.

Утилита работает с централизованным каталогом копий и поддерживает инкрементальное копирование и восстановление.

При локальной работе каталог резервных копий размещается на том же сервере, что и копируемые экземпляры. Для удаленного доступа к каталогу резервных копий используется SSH.

Наличие хранилища копий позволяет выполнять сложные операции, такие как слияние копий, и реализовывать политики их хранения. Имея каталог копий, можно проверить целостность данных и возможность восстановления на заданный момент времени (PITR, Point In Time Recovery), не выполняя само восстановление.

Утилита поддерживает параллельное резервирование и восстановление, может выполнять роль архиватора журнала предзаписи.

Другие важные возможности pg_probackup: восстановление из копии только заданных баз данных, поддержка сжатия данных и файловой системы CFS (Compressed File System), резервирование с физической реплики и резервирование файлов, находящихся вне PGDATA (например, файлов конфигурации), поддержка хранилищ S3.

https://postgrespro.ru/docs/enterprise/13/app-pgprobackup

# Локальная работа

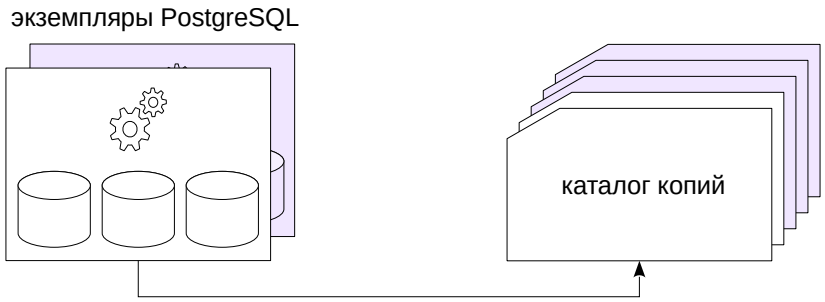
Локальная работа

Полная копия

Разностное копирование

Копирование изменений

# Локальная работа




Базы данных и каталог резервных копий на одном сервере

Простая подготовка

- инициализация каталога резервных копий
- добавление экземпляра СУБД для копирования
- назначение прав роли для копирования

10

При локальной работе каталог резервных копий размещается на том же сервере, что и копируемые экземпляры. В каталоге копий хранятся резервные копии и данные о них (метаинформация).

Утилита pg_probackup выполняет только горячее резервирование. В самой простой стратегии резервного копирования в локальном каталоге резервных копий формируется базовая копия без журнала предзаписи. Если включено архивирование WAL, сегменты журнала также копируются в каталог.

Пользователь, запускающий pg_probackup, должен иметь полный доступ к каталогу копий. Каталог может содержать резервные копии разных кластеров баз данных; перед первым копированием кластера в каталог нужно выполнить команду pg_probackup add-instance для инициализации подкаталогов и метаданных.

Пользователь должен также иметь доступ на чтение к каталогам данных этих кластеров. Подробнее:

https://postgrespro.ru/docs/enterprise/13/app-initdb#APP-INITDB-ALLOW-GROUP-ACCESS

https://paquier.xyz/postgresql-2/postgres-11-group-access/

### Подготовка pg_probackup

Чтобы не вводить в командной строке полный путь к pg_probackup, мы использовали символьную ссылку:

```
student$ type pg_probackup; ls -l $(which pg_probackup)
```

```
pg_probackup is /usr/bin/pg_probackup
lrwxrwxrwx 1 root root 34 янв 12 11:11 /usr/bin/pg_probackup -> /opt/pgpro/ent-13/bin/pg_probackup
```

При локальной работе пользователь, запускающий pg_probackup, должен также иметь доступ на чтение файлов каталога данных копируемого экземпляра. Для этого достаточно:

- при инициализации кластера запустить утилиту initdb с ключом -g;
- дать группе postgres право чтения каталога PGDATA;
- включить пользователя в группу postgres.

Все эти действия были выполнены при настройке виртуальной машины курса.

---

Работа с pg_probackup начинается с создания и инициализации каталога резервных копий.

```
student$ sudo mkdir /var/probackup
```

Утилите pg_probackup необходим доступ в этот каталог. Мы будем запускать утилиту от имени пользователя student, поэтому сделаем его владельцем каталога.

```
student$ sudo chown student:student /var/probackup
```

Для инициализации каталога резервных копий используется команда init утилиты pg_probackup, опция -B указывает местоположение каталога:

```
student$ pg_probackup init -B /var/probackup
```

```
INFO: Backup catalog '/var/probackup' successfully initialized
```

Заглянем в каталог, чтобы проверить результаты инициализации.

```
student$ tree --noreport /var/probackup
```

```
/var/probackup
├── backups
└── wal
```

При инициализации были созданы два подкаталога:

- backups — для резервных копий,
- wal — для архива журнала предзаписи.

---

Утилита pg_probackup имеет встроенную систему помощи. Получить информацию о команде можно командой help:

```
student$ pg_probackup help init
```

Или так:

```
student$ pg_probackup init --help
```

```
pg_probackup init -B backup-path

  -B, --backup-path=backup-path    location of the backup storage area
                [--s3=s3-interface-provider]
```

---

Следующий шаг — добавление копируемого экземпляра в каталог копий — выполняется командой add-instance:

- ключ -D указывает путь к каталогу данных копируемого экземпляра (PGDATA);
- ключ --instance задает имя экземпляра в каталоге копий.

```
student$ pg_probackup add-instance -B /var/probackup -D /var/lib/pgpro/ent-13 --instance ent-13
```

```
INFO: Instance 'ent-13' successfully initialized
```

Снова проверим содержимое каталога копий.

```
student$ tree --noreport /var/probackup
```

```
/var/probackup
├── backups
│   └── ent-13
│       └── pg_probackup.conf
└── wal
```

```
└── ent-13
```

В результате:

- в каталогах backups и wal появились подкаталоги с именем экземпляра;
- для экземпляра создан файл конфигурации pg_probackup.conf.

Содержимое файла конфигурации можно посмотреть средствами ОС, а также командой show-config утилиты pg_probackup:

```
student$ pg_probackup show-config -B /var/probackup --instance ent-13
```

```
# Backup instance information
pgdata = /var/lib/pgpro/ent-13
system-identifier = 7323121610635147817
xlog-seg-size = 16777216
# Connection parameters
pgdatabase = student
# Archive parameters
archive-timeout = 5min
# Logging parameters
log-level-console = INFO
log-level-file = OFF
log-format-console = PLAIN
log-format-file = PLAIN
log-filename = pg_probackup.log
log-rotation-size = 0TB
log-rotation-age = 0d
# Retention parameters
retention-redundancy = 0
retention-window = 0
wal-depth = 0
# Compression parameters
compress-algorithm = none
compress-level = 1
# Remote access parameters
remote-proto = ssh
```

Утилита сгенерировала и сохранила в параметре system-identifier уникальный идентификатор экземпляра. Перед выполнением резервного копирования сохраненный идентификатор сверяется с вычисленным по резервируемому экземпляру, при несовпадении резервирование не выполняется. Это защищает копии от ошибочных действий администратора.

Рекомендуется использовать для резервного копирования непривилегированную роль с атрибутом replication. Создадим такую роль и одноименную базу данных, к которой эта роль будет подключаться:

```
student$ psql
```

```
=> CREATE ROLE backup LOGIN REPLICATION;
```

```
CREATE ROLE
```

```
=> CREATE DATABASE backup OWNER backup;
```

```
CREATE DATABASE
```

Для резервного копирования роль backup должна иметь привилегии на выполнение некоторых функций схемы pg_catalog в базе backup, предоставим их.

```
=> \c backup
```

```
You are now connected to database "backup" as user "student".
```

```
=>
BEGIN;
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_start_backup(text, boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stop_backup(boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
COMMIT;
```

```
BEGIN
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
COMMIT
```

```
=> \c student

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

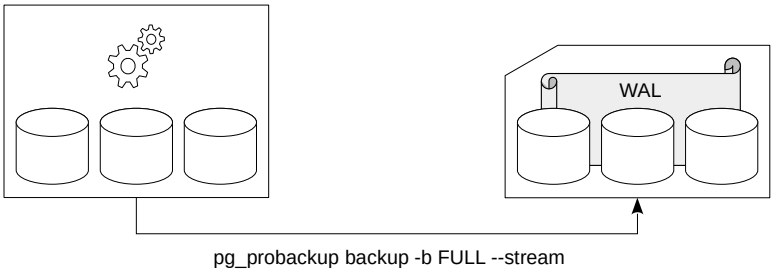
Утилита pg_probackup использует стандартные параметры подключения к базе данных. Для удобства их можно записать в файл конфигурации с помощью команды set-config:

```
student$ pg_probackup set-config -B /var/probackup --instance ent-13 -d backup -U backup
```

Вот что получилось:

```
student$ pg_probackup show-config -B /var/probackup --instance ent-13 | grep -A2 '# Connection parameters'

# Connection parameters
pgdatabase = backup
pguser = backup
```

# Полная копия




Полная резервная копия содержит все данные экземпляра

Потоковый способ доставки → автономная копия



Команда `pg_probackup backup -b FULL` формирует резервную копию, состоящую из базовой копии кластера и сегментов журнала предзаписи.

Сегменты журнала по умолчанию доставляются файловым способом — подразумевается ключ --archive, при этом требуется настройка архивации (о ней поговорим в следующей теме).

Потоковая доставка журнала задается ключом --stream. В этом случае резервная копия является самодостаточной, поэтому ее называют *автономной* — копия включает в себя все сегменты журнала предзаписи, необходимые для восстановления согласованности из базовой копии на момент начала копирования.

# Сжатие


Сжимаются файлы данных и файлы сегментов WAL

```
--compress-algorithm=zlib
--compress-level=1
```



Работа pg_probackup может быть существенно оптимизирована посредством сжатия файлов данных и сегментов WAL. Во-первых, экономится место хранения резервных копий, а во-вторых, потоки данных меньше загружают подсистему ввода-вывода. Взамен требуются ресурсы процессора для выполнения операций сжатия.

Сжатие включается ключом --compress, в котором можно указать алгоритм и уровень сжатия. Для команды backup поддерживаются алгоритмы zlib, lz4, zstd и pglz, если соответствующие библиотеки установлены в системе.

### Полная потоковая резервная копия

Для потоковой доставки WAL используется протокол репликации. Значения параметров по умолчанию и разрешения в pg_hba.conf позволяют его использовать:

```
=> SELECT name, setting
FROM pg_settings
WHERE name IN ('wal_level','max_wal_senders','max_replication_slots');

SELECT type, database, user_name, address, auth_method
FROM pg_hba_file_rules()
WHERE 'replication' = ANY(database);
```

```
         name          | setting
-----------------------+---------
 max_replication_slots | 10
 max_wal_senders       | 10
 wal_level             | replica
(3 rows)

 type  |   database    | user_name |  address  |  auth_method
-------+---------------+-----------+-----------+---------------
 local | {replication} | {all}     |           | trust
 host  | {replication} | {all}     | 127.0.0.1 | scram-sha-256
 host  | {replication} | {all}     | ::1       | scram-sha-256
(3 rows)
```

Получим полную резервную копию с потоковой доставкой записей WAL:

- ключ -b FULL запрашивает автономную копию;
- ключ --stream включает потоковую доставку WAL (по умолчанию используется файловая архивация);
- ключ --temp-slot создает временный слот репликации, который не даст серверу удалить еще не скопированные сегменты WAL.

```
student$ pg_probackup backup -b FULL -B /var/probackup --instance ent-13 --stream --temp-slot

INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAF7, backup mode: FULL, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: PGDATA size: 334MB
INFO: Current Start LSN: 0/10000028, TLI: 1
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/11000000 currentpos 0/11000000
INFO: backup->stop_lsn 0/1000B998
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 4s
INFO: Validating backup S7DAF7
INFO: Backup S7DAF7 data files are valid
INFO: Backup S7DAF7 resident size: 350MB
INFO: Backup S7DAF7 completed
```

Проверим каталог.

```
student$ pg_probackup show -B /var/probackup --instance ent-13

================================================================================================================================================
 Instance  Version  ID      Recovery Time                Mode  WAL Mode  TLI  Time   Data   WAL  Zratio  Start LSN   Stop LSN    Status
================================================================================================================================================
 ent-13    13       S7DAF7  2024-01-16 21:37:55.703572+03  FULL  STREAM    1/0    4s  334MB  16MB    1,00  0/10000028  0/1000B998  OK
```

При выполнении резервного копирования pg_probackup вычисляет и сохраняет контрольные суммы для всех файлов. По умолчанию после копирования и перед восстановлением выполняется проверка целостности копии путем сравнения контрольных сумм.

Можно проверить целостность копии вручную с помощью команды validate:

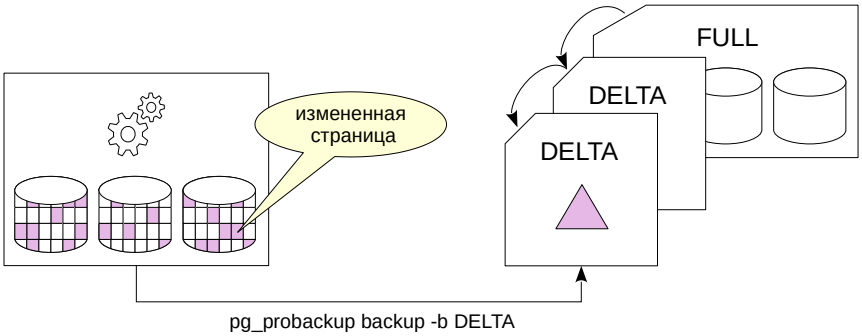
```
student$ pg_probackup validate -B /var/probackup --instance ent-13 -i S7DAF7

INFO: Validating backup S7DAF7
INFO: Backup S7DAF7 data files are valid
INFO: Backup S7DAF7 WAL segments are valid
INFO: Backup S7DAF7 is valid.
INFO: Validate of backup S7DAF7 completed.
```

Команда show показывает информацию о копии по ее идентификатору.

```
student$ pg_probackup show -B /var/probackup --instance ent-13 -i S7DAF7

#Configuration
backup-mode = FULL
stream = true
compress-alg = none
compress-level = 1
from-replica = false

#Compatibility
block-size = 8192
xlog-block-size = 8192
checksum-version = 1
program-version = 2.6.7
server-version = 13
cfs-backup-version = 1

#Result backup info
timelineid = 1
start-lsn = 0/10000028
stop-lsn = 0/1000B998
start-time = '2024-01-16 21:37:55+03'
end-time = '2024-01-16 21:37:59+03'
recovery-xid = 742
recovery-time = '2024-01-16 21:37:55.703572+03'
data-bytes = 350402779
wal-bytes = 16777216
uncompressed-bytes = 350080699
pgdata-bytes = 350080305
status = OK
primary_conninfo = 'user=backup reusepass=yes channel_binding=prefer port=5432 sslmode=prefer sslcompression=0 ssl_min_protocol_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres ta
content-crc = 3129148887
```

Поддерживается также вывод информации о копии в формате JSON.

# Разностное копирование




Разностное копирование — вариант инкрементального

Читаются все файлы данных, но в копию попадают только измененные с последнего копирования страницы

Полная и инкрементальные копии образуют цепочку



При восстановлении кластера из инкрементальной копии pg_probackup восстанавливает родительскую полную копию, а затем применяет к ней последовательно инкрементальные копии, останавливаясь после целевой.

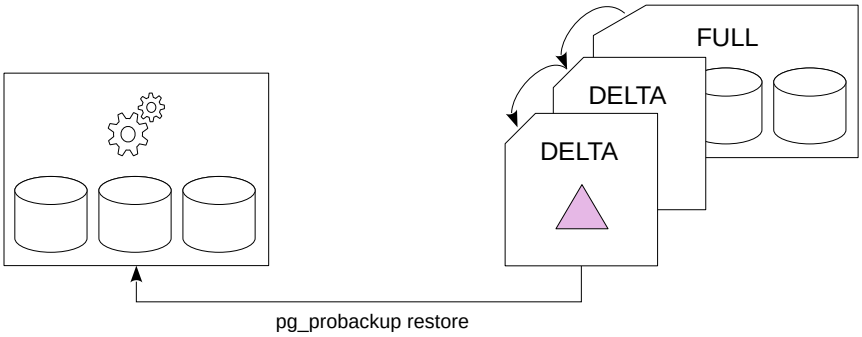
Утилита pg_probackup поддерживает три режима инкрементального резервного копирования: DELTA, PAGE и PTRACK.

Инкрементальное копирование DELTA — разностное резервное копирование. При этом страницы файлов в каталоге данных сравниваются со страницами из последней резервной копии. В новую резервную копию попадают только измененные с момента последнего копирования страницы.

Достоинством режима DELTA является простота: не требуется архив WAL и не надо устанавливать дополнительные расширения.

Однако есть и недостаток: в этом случае объем ввода-вывода может быть близок к объему при полном резервном копировании.

# Восстановление




Команда restore восстанавливает кластер из копии

По умолчанию выбирается последняя копия

Если указана инкрементальная копия, применяется цепочка копий, начиная с полной



Восстановление запускается командой restore утилиты pg_probackup.

Если для восстановления указана инкрементальная копия, pg_probackup автоматически восстанавливает родительскую полную копию и затем последовательно применяет нужные инкрементальные копии.

Если идентификатор копии не указан, выполняется восстановление из самой последней имеющейся копии (включая копии из цепочки, если последней является инкрементальная копия).

### Разностное копирование

Произведем какие-нибудь изменения, например, создадим таблицу:

```
=> CREATE TABLE IF NOT EXISTS t1 (msg text);
```

```
CREATE TABLE
```

И запишем в нее данные:

```
=> INSERT INTO t1 VALUES ('Полная копия создана. Сделаем разностную.');
```

```
INSERT 0 1
```

Получим разностную резервную копию:

```
student$ pg_probackup backup -B /var/probackup --instance ent-13 -b DELTA --stream
```

```
INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAFC, backup mode: DELTA, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: Parent backup: S7DAF7
INFO: PGDATA size: 334MB
INFO: Current Start LSN: 0/12000028, TLI: 1
INFO: Parent Start LSN: 0/10000028, TLI: 1
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/13000000 currentpos 0/13000000
INFO: backup->stop_lsn 0/120001C0
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup S7DAFC
INFO: Backup S7DAFC data files are valid
INFO: Backup S7DAFC resident size: 16MB
INFO: Backup S7DAFC completed
```

Что теперь в каталоге?

```
student$ pg_probackup show -B /var/probackup --instance ent-13
```

```
========================================================================================================================================
 Instance  Version  ID      Recovery Time                Mode   WAL Mode  TLI  Time   Data   WAL  Zratio  Start LSN   Stop LSN    Status
========================================================================================================================================
 ent-13    13       S7DAFC  2024-01-16 21:38:00.508718+03  DELTA  STREAM    1/1     0  484kB  16MB    1,00  0/12000028  0/120001C0  OK
 ent-13    13       S7DAF7  2024-01-16 21:37:55.703572+03  FULL   STREAM    1/0    4s  334MB  16MB    1,00  0/10000028  0/1000B998  OK
```

Обратите внимание, насколько разностная копия меньше полной.

---

Попробуем восстановить кластер из такой копии. Остановим экземпляр.

```
student$ sudo systemctl stop postgrespro-ent-13.service
```

Удаляем содержимое PGDATA.

```
postgres$ rm -rf /var/lib/pgpro/ent-13/*
```

Восстанавливаемый экземпляр обычно запускается от имени пользователя ОС postgres, а каталог PGDATA принадлежит пользователю postgres и группе postgres. Есть два варианта задания прав на файлы каталога PGDATA:

- 700 — полные права только для владельца;
- 750 — дополнительно права на чтение и доступ в каталог для членов группы postgres.

Обычно восстановление производится от имени суперпользователя ОС root с дальнейшей установкой владельца и прав на PGDATA.

---

Восстанавливаем кластер от имени root:

```
student$ sudo -E /opt/pgpro/ent-13/bin/pg_probackup restore -B /var/probackup --instance ent-13 -i S7DAFC
```

```
INFO: Validating parents for backup S7DAFC
INFO: Validating backup S7DAF7
INFO: Backup S7DAF7 data files are valid
INFO: Validating backup S7DAFC
INFO: Backup S7DAFC data files are valid
INFO: Backup S7DAFC WAL segments are valid
INFO: Backup S7DAFC is valid.
INFO: Restoring the database from backup at 2024-01-16 21:38:00+03
INFO: Start restoring backup files. PGDATA size: 350MB
INFO: Backup files are restored. Transfered bytes: 350MB, time elapsed: 0
INFO: Restore incremental ratio (less is better): 100% (350MB/350MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 3s
INFO: Restore of backup S7DAFC completed.
```

Устанавливаем владельца и группу:

```
student$ sudo chown -R postgres: /var/lib/pgpro/ent-13
```

Устанавливаем права на чтение для группы, это требуется для потоковой доставки WAL.

```
student$ sudo chmod -R g+rX /var/lib/pgpro/ent-13
```

```
student$ sudo ls -l /var/lib/pgpro/ent-13
```

```
total 140
-rw-r----- 1 postgres postgres   241 янв 16 21:38 backup_label
drwxr-x--- 9 postgres postgres  4096 янв 16 21:38 base
drwxr-xr-x 2 postgres postgres  4096 янв 16 21:38 conf.d
-rw-r----- 1 postgres postgres    44 янв 16 21:38 current_logfiles
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 global
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 log
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_commit_ts
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_dynshmem
-rw-r----- 1 postgres postgres  4792 янв 16 21:38 pg_hba.conf
-rw-r----- 1 postgres postgres  1636 янв 16 21:38 pg_ident.conf
drwxr-x--- 4 postgres postgres  4096 янв 16 21:38 pg_logical
drwxr-x--- 4 postgres postgres  4096 янв 16 21:38 pg_multixact
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_notify
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_replslot
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_serial
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_snapshots
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_stat
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_stat_tmp
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_subtrans
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_tblspc
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_twophase
-rw-r----- 1 postgres postgres     3 янв 16 21:38 PG_VERSION
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_wal
drwxr-x--- 2 postgres postgres  4096 янв 16 21:38 pg_xact
-rw-r----- 1 postgres postgres   184 янв 16 21:38 postgresql.auto.conf
-rw-r----- 1 postgres postgres 34164 янв 16 21:38 postgresql.conf
```

Запускаем экземпляр.

```
student$ sudo systemctl start postgrespro-ent-13.service

student$ psql

=> SELECT * FROM t1;

                  msg
-----------------------------------------
 Полная копия создана. Сделаем разностную.
(1 row)
```

Мы восстановили данные с помощью разностной копии.

---

Удалим полную резервную копию. При этом удалится и дочерняя разностная.

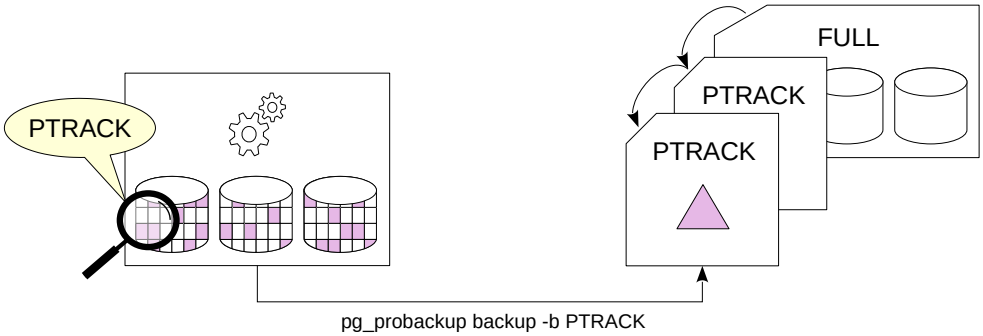
```
student$ pg_probackup delete -B /var/probackup --instance ent-13 -i S7DAF7

INFO: Resident data size to free by delete of backup S7DAF7 : 367MB
INFO: Delete: S7DAF7 2024-01-16 21:37:55+03
INFO: Delete: S7DAFC 2024-01-16 21:38:00+03
```

Что осталось в каталоге?

```
student$ pg_probackup show -B /var/probackup --instance ent-13

========================================================================================================================
 Instance  Version  ID  Recovery Time  Mode  WAL Mode  TLI  Time  Data  WAL  Zratio  Start LSN  Stop LSN  Status
========================================================================================================================
```

Все резервные копии удалены.

# Копирование изменений




Механизм PTRACK отслеживает изменения страниц

В резервную копию помещаются лишь измененные страницы, обнаруженные PTRACK

Требуется подключение расширения ptrack



PTRACK — еще один режим инкрементального резервного копирования базы Postgres Pro на уровне страниц, для которого используется механизм PTRACK, предоставляемый ядром Postgres Pro Enterprise и одноименным расширением. Механизм PTRACK отслеживает изменения страниц в специальной карте. Утилита pg_probackup использует API этого механизма для получения информации о страницах, измененных с определенного момента.

Отличие от режима DELTA состоит в том, что для получения списка страниц, подлежащих копированию, не требуется читать файлы данных и последнюю резервную копию.

Этот режим инкрементального резервного копирования может оказаться самым быстрым, особенно в сценариях, когда интенсивно изменяются одни и те же страницы.

### Копирование изменений с помощью PTRACK

Подключим библиотеку PTRACK к экземпляру.

```
=> ALTER SYSTEM SET shared_preload_libraries = 'ptrack';
```

```
ALTER SYSTEM
```

```
=> ALTER SYSTEM SET client_min_messages TO error;
```

```
ALTER SYSTEM
```

Экземпляр необходимо перезагрузить.

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

```
student$ psql
```

Установим расширение в базу backup.

```
=> \c backup
```

```
You are now connected to database "backup" as user "student".
```

```
=> CREATE EXTENSION IF NOT EXISTS ptrack;
```

```
CREATE EXTENSION
```

```
=> \c student
```

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

Необходимо задать значение параметра ptrack.map_size, рекомендуется 1/1024 объема кластера Postgres Pro:

```
=> ALTER SYSTEM SET ptrack.map_size = '1MB';
```

```
ALTER SYSTEM
```

Изменение этого параметра также требует рестарта.

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

Снова сделаем полную резервную копию с потоковой доставкой WAL.

```
student$ pg_probackup backup -b FULL -B /var/probackup --instance ent-13 --stream --temp-slot
```

```
INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAFI, backup mode: FULL, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: PGDATA size: 335MB
INFO: Current Start LSN: 0/14000028, TLI: 1
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/15000000 currentpos 0/15000000
INFO: backup->stop_lsn 0/140001C0
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 4s
INFO: Validating backup S7DAFI
INFO: Backup S7DAFI data files are valid
INFO: Backup S7DAFI resident size: 351MB
INFO: Backup S7DAFI completed
```

Добавим в таблицу еще одну запись.

```
student$ psql
```

```
=> INSERT INTO t1 VALUES ('Проверим PTRACK.');
```

```
INSERT 0 1
```

Теперь выполним инкрементальное резервное копирование в режиме PTRACK.

```
student$ pg_probackup backup -B /var/probackup --instance ent-13 -b PTRACK --stream
```

```
INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAFN, backup mode: PTRACK, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: Parent backup: S7DAFI
INFO: PGDATA size: 335MB
INFO: Current Start LSN: 0/16000028, TLI: 1
INFO: Parent Start LSN: 0/14000028, TLI: 1
INFO: Extracting pagemap of changed blocks
INFO: Pagemap successfully extracted, time elapsed: 0 sec
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/17000000 currentpos 0/17000000
INFO: backup->stop_lsn 0/160060D0
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup S7DAFN
INFO: Backup S7DAFN data files are valid
INFO: Backup S7DAFN resident size: 17MB
INFO: Backup S7DAFN completed
```

В каталоге резервных копий появится инкрементальная копия:

```
student$ /opt/pgpro/ent-13/bin/pg_probackup show -B /var/probackup --instance ent-13
```

```
==========================================================================================================================================
 Instance  Version  ID      Recovery Time                  Mode    WAL Mode  TLI  Time    Data   WAL  Zratio  Start LSN   Stop LSN    Status
==========================================================================================================================================
 ent-13    13       S7DAFN  2024-01-16 21:38:11.921716+03  PTRACK  STREAM    1/1    1s  1180kB  16MB    1,00  0/16000028  0/160060D0  OK
 ent-13    13       S7DAFI  2024-01-16 21:38:07.385870+03  FULL    STREAM    1/0    5s   335MB  16MB    1,00  0/14000028  0/140001C0  OK
```

---

Удалим из каталога копируемый экземпляр.

```
student$ pg_probackup del-instance -B /var/probackup --instance ent-13
```

```
INFO: Delete: S7DAFN 2024-01-16 21:38:11+03
INFO: Delete: S7DAFI 2024-01-16 21:38:07+03
INFO: Instance 'ent-13' successfully deleted
```

После удаления экземпляра из каталога также удаляются все его резервные копии.

# Итоги

Штатные средства резервного копирования и восстановления не всегда достаточны

Утилита pg_probackup предоставляет все необходимые возможности

Поддерживается полное и инкрементальное резервное копирование, сжатие, параллельное резервирование и восстановление

# Практика

1. Подготовьте каталог резервных копий и зарегистрируйте в нем копируемый экземпляр.
2. Зарегистрируйте роль backup и создайте базу данных backup.
3. Выполните полное резервное копирование.
4. Внесите какие-нибудь изменения в данные кластера и получите разностную копию.
5. Восстановите экземпляр.



1. Создайте каталог /var/probackup, инициализируйте его с помощью pg_probackup init, а затем добавьте в каталог экземпляр командой pg_probackup add-instance так же, как в демонстрации.
2. Зарегистрировав роль backup, создайте одноименную базу данных и по аналогии с демонстрацией назначьте командой GRANT требуемые права.
3. Команда pg_probackup backup, помимо указания каталога копий для полного резервного копирования, должна в этом случае также включать параметр --stream.
4. При получении разностной копии не забудьте параметры -b DELTA и --stream.
5. Перед восстановлением остановите работающий экземпляр СУБД, удалите все содержимое его каталога данных, а затем проведите восстановление командой pg_probackup restore.

### 1. Каталог резервных копий и регистрация экземпляра

Подготавливаем каталог в файловой системе.

```
student$ sudo mkdir /var/probackup
```

```
student$ sudo chown student: /var/probackup
```

Инициализируем каталог копий.

```
student$ pg_probackup init -B /var/probackup
```

```
INFO: Backup catalog '/var/probackup' successfully initialized
```

Регистрируем экземпляр.

```
student$ pg_probackup add-instance -B /var/probackup -D /var/lib/pgpro/ent-13 --instance ent-13
```

```
INFO: Instance 'ent-13' successfully initialized
```

---

### 2. Роль и база данных для подключения

```
student$ psql
```

Создадим роль backup и базу данных, к которой будет подключаться роль backup.

```
=> CREATE ROLE backup LOGIN REPLICATION;
```

```
CREATE ROLE
```

```
=> CREATE DATABASE backup OWNER backup;
```

```
CREATE DATABASE
```

Предоставляем права.

```
=> \c backup
```

```
You are now connected to database "backup" as user "student".
```

```
=>
BEGIN;
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_start_backup(text, boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stop_backup(boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
COMMIT;
```

```
BEGIN
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
COMMIT
```

```
=> \c student
```

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

---

### 3. Полная копия

Формируем полную копию:

```
student$ pg_probackup backup -B /var/probackup --instance ent-13 -b FULL --stream
```

```
INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAVP, backup mode: FULL, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
WARNING: Current PostgreSQL role is superuser. It is not recommended to run pg_probackup under superuser.
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: PGDATA size: 334MB
INFO: Current Start LSN: 0/10000028, TLI: 1
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/11000000 currentpos 0/11000000
INFO: backup->stop_lsn 0/100001C0
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 4s
INFO: Validating backup S7DAVP
INFO: Backup S7DAVP data files are valid
INFO: Backup S7DAVP resident size: 350MB
INFO: Backup S7DAVP completed
```

---

### 4. Разностная копия

Небольшая активность...

```
student$ /opt/pgpro/ent-13/bin/createuser user1
```

```
student$ /opt/pgpro/ent-13/bin/createdb -O user1 db1
```

Формируем разностную копию.

```
student$ pg_probackup backup -B /var/probackup --instance ent-13 -b DELTA --stream
```

```
INFO: Backup start, pg_probackup version: 2.6.7, instance: ent-13, backup ID: S7DAVU, backup mode: DELTA, wal mode: STREAM, remote: false, compress-algorithm: none, compress-level: 1
INFO: This PostgreSQL instance was initialized with data block checksums. Data block corruption will be detected
WARNING: Current PostgreSQL role is superuser. It is not recommended to run pg_probackup under superuser.
INFO: Database backup start
INFO: wait for pg_start_backup()
INFO: Parent backup: S7DAVP
INFO: PGDATA size: 342MB
INFO: Current Start LSN: 0/12000028, TLI: 1
```

```
INFO: Parent Start LSN: 0/10000028, TLI: 1
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: stop_stream_lsn 0/13000000 currentpos 0/13000000
INFO: backup->stop_lsn 0/12000A10
INFO: Getting the Recovery Time from WAL
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 1s
INFO: Validating backup S7DAVU
INFO: Backup S7DAVU data files are valid
INFO: Backup S7DAVU resident size: 24MB
INFO: Backup S7DAVU completed
```

Проверка.

```
student$ pg_probackup show -B /var/probackup --instance ent-13
```

```
========================================================================================================================================
 Instance  Version  ID      Recovery Time                  Mode   WAL Mode  TLI  Time    Data   WAL  Zratio  Start LSN   Stop LSN    Status
========================================================================================================================================
 ent-13    13       S7DAVU  2024-01-16 21:47:54.759731+03  DELTA  STREAM    1/1    1s  8399kB  16MB    1,00  0/12000028  0/12000A10  OK
 ent-13    13       S7DAVP  2024-01-16 21:47:49.838183+03  FULL   STREAM    1/0    4s   334MB  16MB    1,00  0/10000028  0/100001C0  OK
```

---

### 5. Восстановление из резервной копии

Останавливаем экземпляр.

```
student$ sudo systemctl stop postgrespro-ent-13.service
```

Удаляем содержимое PGDATA.

```
postgres$ rm -rf /var/lib/pgpro/ent-13/*
```

Восстанавливаем с правами root.

```
student$ sudo pg_probackup restore -B /var/probackup --instance ent-13
```

```
INFO: Validating parents for backup S7DAVU
INFO: Validating backup S7DAVP
INFO: Backup S7DAVP data files are valid
INFO: Validating backup S7DAVU
INFO: Backup S7DAVU data files are valid
INFO: Backup S7DAVU WAL segments are valid
INFO: Backup S7DAVU is valid.
INFO: Restoring the database from backup at 2024-01-16 21:47:54+03
INFO: Start restoring backup files. PGDATA size: 358MB
INFO: Backup files are restored. Transfered bytes: 358MB, time elapsed: 0
INFO: Restore incremental ratio (less is better): 100% (358MB/358MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 3s
INFO: Restore of backup S7DAVU completed.
```

Меняем владельца и группу.

```
student$ sudo chown -R postgres: /var/lib/pgpro/ent-13
```

Устанавливаем права на чтение для группы, это требуется для потоковой передачи WAL.

```
student$ sudo chmod -R g+rX /var/lib/pgpro/ent-13
```

```
student$ sudo ls -l /var/lib/pgpro/ent-13
```

```
total 140
-rw-r-----  1 postgres postgres   241 янв 16 21:47 backup_label
drwxr-x--- 10 postgres postgres  4096 янв 16 21:47 base
drwxr-xr-x  2 postgres postgres  4096 янв 16 21:47 conf.d
-rw-r-----  1 postgres postgres    44 янв 16 21:47 current_logfiles
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 global
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 log
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_commit_ts
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_dynshmem
-rw-r-----  1 postgres postgres  4792 янв 16 21:47 pg_hba.conf
-rw-r-----  1 postgres postgres  1636 янв 16 21:47 pg_ident.conf
drwxr-x---  4 postgres postgres  4096 янв 16 21:47 pg_logical
drwxr-x---  4 postgres postgres  4096 янв 16 21:47 pg_multixact
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_notify
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_replslot
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_serial
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_snapshots
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_stat
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_stat_tmp
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_subtrans
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_tblspc
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_twophase
-rw-r-----  1 postgres postgres     3 янв 16 21:47 PG_VERSION
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_wal
drwxr-x---  2 postgres postgres  4096 янв 16 21:47 pg_xact
-rw-r-----  1 postgres postgres   184 янв 16 21:47 postgresql.auto.conf
-rw-r-----  1 postgres postgres 34164 янв 16 21:47 postgresql.conf
```

Запускаем экземпляр.

```
student$ sudo systemctl start postgrespro-ent-13.service
```

```
student$ psql
```

```
=> \du
```

```
                                 List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 backup    | Replication                                                | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 student   | Superuser                                                  | {}
 user1     |                                                            | {}
```

```
=> \l
```

```
                                  List of databases
   Name    |  Owner   | Encoding |     Collate     |    Ctype    |   Access privileges
-----------+----------+----------+-----------------+-------------+-----------------------
 backup    | backup   | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 |
 db1       | user1    | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 |
 demo      | student  | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 |
 student   | postgres | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 | =c/postgres          +
           |          |          |                 |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8@icu | en_US.UTF-8 | =c/postgres          +
           |          |          |                 |             | postgres=CTc/postgres
(7 rows)
```

Данные восстановлены.