Авторы: Алексей Береснев, Илья Баштанов, Павел Толмачев

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

Темы

Установка и начальная настройка

Обновление сервера

Миграция с других СУБД

Установка и настройка

Архитектуры и ОС

Пакеты Postgres Pro

Настройка Postgres Pro

Информация о сервере

Нумерация версий

Архитектуры и ОС

64-разрядные архитектуры

- AMD/Intel 64 (x86_64)
- ARM64 (aarch64)
- Эльбрус 2000 (e2kv3, e2kv4)
- Power PC, Power ISA (ppc64le)

ОС на базе Linux

- Red Hat Enterprise Linux, CentOS, Rocky Linux, Oracle Linux, Rosa/POCA, РЕД ОС, AlterOS, AlmaLinux
- Debian, Ubuntu, Astra Linux
- Альт
- SUSE Linux Enterprise Server

Postgres Pro может работать на серверных операционные системах на базе Linux. Поддерживаются только 64-х разрядные архитектуры. Список поддерживаемых систем можно посмотреть на сайте Postgres Pro https://postgrespro.ru/products/postgrespro/supported_os и в разделе документации https://postgrespro.ru/docs/enterprise/13/binary-installation-on-linux#SUPPORTED-LINUX-OS

Платформа Windows поддерживается в версиях до 14 включительно.

Более детальную информацию по поддержке архитектур, отличных от x86-64, можно запросить на info@postgrespro.ru.

Пакеты Postgres Pro

Пакет верхнего уровня

- postgrespro-ent-13
- все доступные компоненты сервера
- автоматическая настройка

Детальный выбор пакетов

- только необходимые компоненты
- ручная настройка

Сторонние пакеты для Postgres Pro

- дополнительные расширения и утилиты

Функционал Postgres Pro разделен на пакеты.

Пакет верхнего уровня postgrespro-ent-13 позволяет быстро установить и настроить Postgres Pro для серверных и клиентских систем (будут установлены все доступные компоненты Postgres Pro без возможности выбора). Такой способ установки не рекомендуется использовать при обновлении или миграции системы.

Для более детальной установки и настройки рекомендуется отобрать отдельные пакеты. Можно установить только серверную часть, клиентские утилиты, документацию и т. п. В этом случае действия по настройке, такие как инициализация кластера, придется выполнять самостоятельно.

Пакеты называются одинаково для всех операционных систем (кроме незначительных отличий в суффиксе названия пакета для разработчиков postgrespro-ent-13-devel и пакета с отладочной информацией postgrespro-ent-13-dbg).

Существуют и сторонние пакеты, собранные специально для Postgres Pro Enterprise. Эти пакеты устанавливают дополнительные утилиты и расширения: libzstd для сжатия данных, oracle-fdw-ent-13 – обертку сторонних данных для Oracle, pgpro-pgbadger – анализатор журналов сообщений и др.

Полный перечень пакетов есть в документации:

https://postgrespro.ru/docs/enterprise/13/binary-installation-on-linux#CHOOSING-PGPRO-PACKAGES

Настройка Postgres Pro

pg-wrapper — управление ссылками

- путь поиска PATH и страницы man

pg-setup — управление кластером

- инициализация кластера
- проверка доступных портов
- изменение конфигурации сервера
- включение и отключение автоматического запуска службы
- запуск, остановка или перезапуск службы
- настройка кластера

Проверка контрольных сумм

- включена по умолчанию

В данном курсе рассматривается настройка и работа Postgres Pro Enterprise в среде Linux.

Для добавления ссылок в путь поиска PATH и для подключения man-страниц используется скрипт pg-wrapper.

https://postgrespro.ru/docs/enterprise/13/pg-wrapper

Для инициализации кластера Postgres Pro рекомендуется скрипт pg-setup

https://postgrespro.ru/docs/enterprise/13/pg-setup

Скрипт pg-setup может создавать базы данных и инициализировать кластер Postgres Pro. Запускать скрипт нужно от имени пользователя root, действия с базами данных скрипт выполняет под ролью postgres.

По умолчанию скрипт инициализирует кластер БД с включенными контрольными суммами, их можно отключить ключом --no-data-checksums. Кластер инициализируется с поддержкой библиотеки ICU, в которой реализована платформонезависимая сортировка

https://postgrespro.ru/docs/enterprise/13/collation#COLLATION-MANAGING

Скрипт pg-setup также позволяет запустить сервер, остановить его и настроить автоматический запуск.

https://postgrespro.ru/docs/enterprise/13/binary-installation-on-linux#STARTING-SERVER-AUTOMATICALLY

Информация о сервере

pgpro_controldata

- вывод информации о кластере и/или сервере
- чтение младших мажорных версий и редакций
- сравнение параметров сервера, влияющих на совместимость

SQL-функции

- проверка информации о системе после установки
- версия и редакция Postgres Pro
- идентификатор состояния исходного кода

pg_config

- версия и редакция Postgres Pro

С помощью утилиты pgpro_controldata можно вывести управляющую информацию о кластере PostgreSQL/Postgres Pro и параметры совместимости кластера и/или сервера. Без указания параметров утилита pgpro_controldata работает аналогично утилите pg_controldata. Устанавливается отдельным пакетом.

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

После установки Postgres Pro в системе можно использовать дополнительные функции для вывода информации о системе.
С помощью этих функций можно узнать версию Postgres Pro, название редакции и идентификатор состояния исходного кода (хеш коммита), из которого был собран пакет — эти сведения пригодятся для обращения в техподдержку.

https://postgrespro.ru/docs/enterprise/13/functions-info#FUNCTIONS-INFO-SESSION-TABLE

Аналогичная информация выводится и с помощью утилиты pg_config.

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

Проверка системы после установки

В виртуальной машине курса установлен пакет postgrespro-ent-13-server, исполняемые файлы находятся в каталоге /opt/pgpro/ent-13/bin:

```
student$ ls -C /opt/pgpro/ent-13/bin/
```

```
check-db-dir  pg_archivecleanup  pg_isready      pg_standby       postmaster
clusterdb     pg_basebackup      pg_probackup    pg_test_fsync    psql
createdb      pgbench            pg_receivewal   pg_test_timing   reindexdb
createuser    pg_checksums       pg_recvlogical  pg_upgrade       vacuumdb
dropdb        pg_controldata     pg_resetwal     pg_verifybackup  vacuumlo
dropuser      pg_ctl             pg_restore      pg_waldump
initdb        pg_dump            pg_rewind       pg-wrapper
oid2name      pg_dumpall         pg-setup        postgres
```

Для инициализации кластера была использована утилита pg-setup:

```
student$ sudo /opt/pgpro/ent-13/bin/pg-setup initdb -g -D /var/lib/pgpro/ent-13 --auth=trust
```

Этой же утилитой pg-setup была настроена служба с автозапуском, так что экземпляром можно управлять в стиле ОС:

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

```
● postgrespro-ent-13.service - Postgres Pro ent 13 database server
     Loaded: loaded (/lib/systemd/system/postgrespro-ent-13.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-01-16 21:31:51 MSK; 848ms ago
   Main PID: 251064 (postgres)
      Tasks: 10 (limit: 3411)
     Memory: 44.5M
        CPU: 33ms
     CGroup: /system.slice/postgrespro-ent-13.service
             ├─251064 /opt/pgpro/ent-13/bin/postgres -D /var/lib/pgpro/ent-13
             ├─251065 "postgres: logger " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251067 "postgres: checkpointer " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251068 "postgres: background writer " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251069 "postgres: walwriter " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251070 "postgres: autovacuum launcher " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251071 "postgres: stats collector " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251072 "postgres: logical replication launcher " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─251081 "postgres: cfs-worker-0 " "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             └─251097 "postgres: student postgres [local] idle" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""

янв 16 21:31:51 student systemd[1]: Starting Postgres Pro ent 13 database server...
янв 16 21:31:51 student postgres[251064]: 2024-01-16 21:31:51.042 MSK [251064] LOG:  Start CFS version 0.54 supported compression algorithms pglz,zlib,lz4,zstd encryption disabled GC e
янв 16 21:31:51 student postgres[251064]: 2024-01-16 21:31:51.045 MSK [251064] LOG:  redirecting log output to logging collector process
янв 16 21:31:51 student postgres[251064]: 2024-01-16 21:31:51.045 MSK [251064] HINT:  Future log output will appear in directory "log".
янв 16 21:31:51 student systemd[1]: Started Postgres Pro ent 13 database server.
```

Статус можно узнать и с помощью традиционной утилиты pg_ctl:

```
postgres$ /opt/pgpro/ent-13/bin/pg_ctl -w -D /var/lib/pgpro/ent-13 status
```

```
pg_ctl: server is running (PID: 251064)
/opt/pgpro/ent-13/bin/postgres "-D" "/var/lib/pgpro/ent-13"
```

Для запуска, остановки, перезапуска экземпляра сервера рекомендуется использовать те средства, которые уже используются в вашей компании.

Информацию об установленной системе (номер версии и название редакции) можно узнать разными способами, например, с помощью утилиты pg_config или SQL-функций:

```
=> SELECT pgpro_edition(), pgpro_version() \gx
```

```
-[ RECORD 1 ]-+------------------------------------------------------------------------------------------------------------------
pgpro_edition | enterprise
pgpro_version | PostgresPro (enterprise) 13.13.1 on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
```

Можно вывести значения конфигурационных параметров SQL-командой SHOW, как это делается в обычном PostgreSQL:

```
=> SHOW data_checksums;
   SHOW data_directory;
```

```
 data_checksums
----------------
 on
(1 row)

    data_directory
-----------------------
 /var/lib/pgpro/ent-13
(1 row)
```

Файл postgresql.conf находится в каталоге данных:

```
=> SHOW config_file;
```

```
              config_file
---------------------------------------
 /var/lib/pgpro/ent-13/postgresql.conf
(1 row)
```

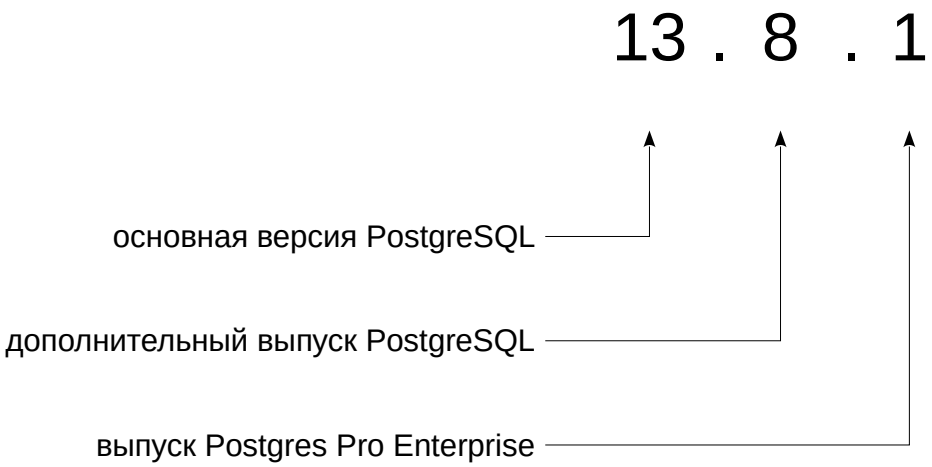
Директива include_dir добавляет к конфигурации содержимое всех файлов *.conf подкаталога conf.d. Это позволяет менять значения параметров и возвращаться к прежним значениям, не меняя основной файл конфигурации. В виртуальной машине директива добавлена в основной конфигурационный файл Postgres Pro Enterprise:

```
student$ grep '^include_dir' /var/lib/pgpro/ent-13/postgresql.conf
```

```
include_dir='conf.d'
```

```
=> \q
```

Нумерация версий



Номер версии Postgres Pro Enterprise состоит из трех частей: номера основной версии (major release), номера дополнительного выпуска (minor release) PostgreSQL и номера выпуска Postgres Pro. До версии 10 основной номер состоял из двух чисел (9.5, 9.6), поэтому выпуски нумеровались четырьмя числами.

Новая основная версия привносит изменение функционала: какие-то возможности добавляются и изменяются, реже — удаляются.

Дополнительные выпуски служат для переноса изменений, сделанных в соответствующем выпуске PostgreSQL, а также для исправления ошибок, проблем безопасности и производительности, найденных в предыдущих выпусках.

Выпуски Postgres Pro Enterprise служат исключительно для исправления ошибок, проблем безопасности и производительности, найденных в предыдущих выпусках Postgres Pro Enterprise.

При обновлении нужно учитывать, что в новом выпуске может быть прекращена поддержка определенных версий операционных систем.

https://postgrespro.ru/docs/enterprise/13/upgrading

Обновление сервера

Обновление Postgres Pro Enterprise

Переход с PostgreSQL и редакций Postgres Pro

Правила сортировки ICU

Обновление Enterprise

Обновление в пределах основной версии

- доп. выпуск PostgreSQL или выпуск Postgres Pro
- установить исполняемые файлы и перезапустить сервер

Обновление основной версии

- pg_dump + pg_restore
- pg_upgrade

В целом процедура обновления сервера Postgres Pro Enterprise аналогична процедуре обновления PostgreSQL. Внимательно прочитайте замечания ко всем выпускам между текущим и целевым.

В любом случае сначала устанавливаются пакеты новой версии или выпуска.

При обновлении на дополнительный выпуск или выпуск Postgres Pro Enterprise достаточно перезапустить сервер.

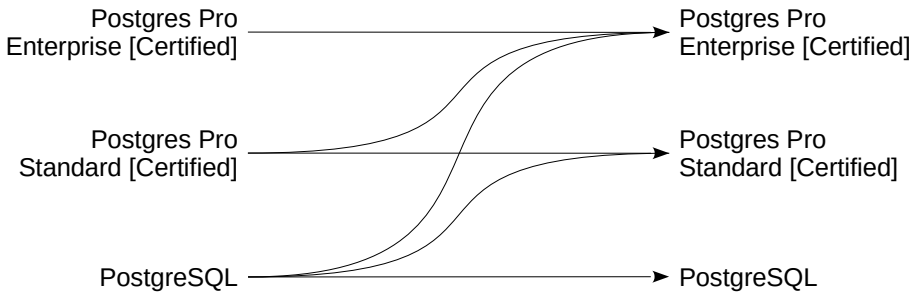
При обновлении основной версии для переноса данных есть два пути:

- сделать логическую резервную копию утилитами pg_dumpall или pg_dump и восстановить данные с помощью pg_restore / psql;
- воспользоваться утилитой pg_upgrade.

Если статистику оптимизатора не переносили из старой версии, после переноса данных нужно выполнить анализ.

После любого обновления следует проверить и при необходимости обновить версии расширений в базах данных.

Варианты перехода



Поддерживаются pg_dump+pg_restore и pg_upgrade

Понижение версии не поддерживается

В иерархии PostgreSQL < Postgres Pro Standard < Postgres Pro Enterprise возможен перенос данных с младшей на более старшую редакцию СУБД. При этом поддерживается как перенос с помощью логического резервирования и восстановления (pg_dump + pg_restore), так и перенос с помощью утилиты pg_upgrade. Возможные пути переноса и ограничения показаны в таблице.

При этом возможен переход только на ту же или бóльшую основную версию СУБД, понижение версии (downgrade) не поддерживается.

Также не поддерживается переход с помощью pg_upgrade на Postgres Pro Enterprise и Postgres Pro Enterprise Certified версий до 11 включительно.

Перенос данных из PostgreSQL

При миграции из PostgreSQL можно использовать средства логического резервного копирования или утилиту pg_upgrade.

В виртуальной машине установлен PostgreSQL 13 и инициализирован кластер prod. Запустим экземпляр.

```
student$ sudo pg_ctlcluster 13 prod start
```

В отдельной базе данных создадим таблицу:

```
student$ psql -p 5433
```

```
=> CREATE DATABASE install;
```

```
ERROR:  database "install" already exists
```

```
=> \c install
```

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

```
=> CREATE TABLE test(s) AS VALUES('Строка из PostgreSQL');
```

```
ERROR:  relation "test" already exists
```

Остановим экземпляр PostgreSQL.

```
student$ sudo pg_ctlcluster 13 prod stop
```

Будем переносить данные с помощью pg_upgrade. Нам понадобится пустой кластер Postgres Pro Enterprise.

```
student$ sudo mkdir /var/lib/pgpro/ent-13-prod
```

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

```
postgres$ /opt/pgpro/ent-13/bin/initdb -D /var/lib/pgpro/ent-13-prod --auth=trust
```

```
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locales
  COLLATE:  en_US.UTF-8
  CTYPE:    en_US.UTF-8
  MESSAGES: en_US.UTF-8
  MONETARY: ru_RU.UTF-8
  NUMERIC:  ru_RU.UTF-8
  TIME:     ru_RU.UTF-8
The default collation provider is "icu".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are enabled.

fixing permissions on existing directory /var/lib/pgpro/ent-13-prod ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Moscow
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    /opt/pgpro/ent-13/bin/pg_ctl -D /var/lib/pgpro/ent-13-prod -l logfile start
```

Для начала запустим утилиту в режиме проверки. Ключами -b и -B нужно указать пути к исполняемым файлам нового и старого кластера, а ключами -d и -D — пути к postgresql.conf.

```
postgres$ /opt/pgpro/ent-13/bin/pg_upgrade --check -b /usr/lib/postgresql/13/bin/ -B /opt/pgpro/ent-13/bin/ -d /etc/postgresql/13/prod -D /var/lib/pgpro/ent-13-prod
```

```
Finding the real data directory for the source cluster      ok
Performing Consistency Checks
-----------------------------
Checking cluster versions                                   ok
Checking cluster editions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for system-defined composite types in user tables  ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Checking for incompatible "xid" data type                   ok
Checking for hash indexes                                   ok
Checking for spgist indexes                                 ok
Checking for brin indexes                                   ok
Checking for external indexes                               ok

lc_collate values for database "postgres" do not match:  old "en_US.UTF-8@libc", new "en_US.UTF-8@icu"
Failure, exiting
```

Обнаружилась проблема: у исходного и целевого кластеров разные настройки для правила сортировки по умолчанию.

Правила сортировки ICU

Правила сортировки

- C, POSIX
- libc
- icu

ICU поддерживается на уровне кластера и БД

- в PostgreSQL только для столбцов и выражений

Указывается при инициализации

```
initdb            --locale=ru_RU.UTF-8@icu
createdb          --locale=ru_RU.UTF-8@icu
CREATE DATABASE ... LOCALE=ru_RU.UTF-8@icu
```


PostgreSQL и Postgres Pro Enterprise генерируют правила сортировки на основе двух библиотек («провайдеров»): традиционной библиотеки libc и современной платформонезависимой ICU, которая дает больше возможностей.

Postgres Pro Enterprise позволяет использовать правила обоих провайдеров на всех уровнях (кластер, база данных, столбец таблицы, выражение). В PostgreSQL правила провайдера libc можно использовать на любом уровне, а правила провайдера ICU не поддерживаются на уровне кластера и базы данных (поддержка добавлена в версии 15). Это ограничение нужно учесть при миграции данных из PostgreSQL.

Подробнее о правилах сортировки рассказывается в теме «Локализация» курса DBA2.

https://postgrespro.ru/docs/enterprise/13/collation

PostgreSQL на уровне кластера и базы данных допускает только правила сортировки провайдера libc (icu поддерживается с версии 15), поэтому придется заново инициализировать кластер Postgres Pro Enterprise с правилом сортировки en_US.UTF-8@libc.

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

postgres$ /opt/pgpro/ent-13/bin/initdb -D /var/lib/pgpro/ent-13-prod --auth=trust --locale=en_US.UTF-8@libc

The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default collation provider is "libc".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are enabled.

fixing permissions on existing directory /var/lib/pgpro/ent-13-prod ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Moscow
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    /opt/pgpro/ent-13/bin/pg_ctl -D /var/lib/pgpro/ent-13-prod -l logfile start
```

Еще раз запустим pg_upgrade в режиме проверки:

```
postgres$ /opt/pgpro/ent-13/bin/pg_upgrade --check -b /usr/lib/postgresql/13/bin/ -B /opt/pgpro/ent-13/bin/ -d /etc/postgresql/13/prod -D /var/lib/pgpro/ent-13-prod

Finding the real data directory for the source cluster      ok
Performing Consistency Checks
----------------------------
Checking cluster versions                                   ok
Checking cluster editions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for system-defined composite types in user tables  ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Checking for incompatible "xid" data type                   ok
Checking for hash indexes                                   ok
Checking for spgist indexes                                 ok
Checking for brin indexes                                   ok
Checking for external indexes                               ok
Checking for presence of required libraries                 ok
Checking database user is the install user                  ok
Checking for prepared transactions                          ok
Checking for new cluster tablespace directories             ok

*Clusters are compatible*
```

Все в порядке, можно переносить данные.

```
postgres$ /opt/pgpro/ent-13/bin/pg_upgrade -b /usr/lib/postgresql/13/bin/ -B /opt/pgpro/ent-13/bin/ -d /etc/postgresql/13/prod -D /var/lib/pgpro/ent-13-prod

Finding the real data directory for the source cluster      ok
Performing Consistency Checks
----------------------------
Checking cluster versions                                   ok
Checking cluster editions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for system-defined composite types in user tables  ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Checking for incompatible "xid" data type                   ok
Checking for hash indexes                                   ok
Checking for spgist indexes                                 ok
Checking for brin indexes                                   ok
Checking for external indexes                               ok
Creating dump of global objects                             ok
Creating dump of database schemas
  install
  postgres
  student
  template1
                                                            ok
Checking for presence of required libraries                 ok
Checking database user is the install user                  ok
Checking for prepared transactions                          ok
Checking for new cluster tablespace directories             ok

If pg_upgrade fails after this point, you must re-initdb the
new cluster before continuing.

Performing Upgrade
------------------
Analyzing all rows in the new cluster                       ok
Freezing all rows in the new cluster                        ok
Transforming commit log segments                            ok
Setting oldest XID for new cluster                          ok
Setting next transaction ID and epoch for new cluster       ok
Deleting files from new pg_multixact/offsets                ok
Converting old pg_multixact/offsets to new format           ok
Deleting files from new pg_multixact/members                ok
Converting old pg_multixact/members to new format           ok
Setting next multixact ID and offset for new cluster        ok
Resetting WAL archives                                      ok
Setting frozenxid and minmxid counters in new cluster       ok
Restoring global objects in the new cluster                 ok
Restoring database schemas in the new cluster
  template1
  install
```

```
  postgres
  student
                                                        ok
Setting minmxid counter in new cluster                  ok
Copying user relation files
  /var/lib/postgresql/13/prod/base/16386/2613
  /var/lib/postgresql/13/prod/base/16386/2683
  /var/lib/postgresql/13/prod/base/16386/16387
  /var/lib/postgresql/13/prod/base/16386/16390
  /var/lib/postgresql/13/prod/base/16386/16392
  /var/lib/postgresql/13/prod/base/13485/2613
  /var/lib/postgresql/13/prod/base/13485/2683
  /var/lib/postgresql/13/prod/base/16385/2613
  /var/lib/postgresql/13/prod/base/16385/2683
  /var/lib/postgresql/13/prod/base/1/2613
  /var/lib/postgresql/13/prod/base/1/2683
                                                        ok
Setting next OID for new cluster                        ok
Sync data directory to disk                             ok
Creating script to analyze new cluster                  ok
Creating script to delete old cluster                   ok
Checking for hash indexes                               ok
Checking for spgist indexes                             ok
Checking for brin indexes                               ok
Checking for external indexes                           ok
Checking for extension updates                          ok

Upgrade Complete
----------------
Optimizer statistics are not transferred by pg_upgrade so,
once you start the new server, consider running:
    ./analyze_new_cluster.sh

Running this script will delete the old cluster's data files:
    ./delete_old_cluster.sh
```

Задаем порт и запускаем сервер Postgres Pro Enterprise.

```
postgres$ echo "include_dir='conf.d'" >> /var/lib/pgpro/ent-13-prod/postgresql.conf

postgres$ mkdir /var/lib/pgpro/ent-13-prod/conf.d

postgres$ echo port=5434 > /var/lib/pgpro/ent-13-prod/conf.d/install.conf

postgres$ /opt/pgpro/ent-13/bin/pg_ctl -w -l /var/lib/pgpro/ent-13-prod/pgpro-ent.log -D /var/lib/pgpro/ent-13-prod start

waiting for server to start.... done
server started

student$ psql -p 5434 -d install
```

```
  => SELECT pgpro_version();
```

```
                                                pgpro_version
-------------------------------------------------------------------------------------------------------------------------
 PostgresPro (enterprise) 13.13.1 on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
(1 row)
```

```
  => SELECT * FROM test;
```

```
          s
----------------------
 Строка из PostgreSQL
(1 row)
```

Данные перенесены в Postgres Pro Enterprise.

Миграция с других СУБД

Миграция данных: oracle_fdw и другие обертки

Миграция других объектов

Миграция данных

SQL

- выгрузка + psql

Обертки сторонних данных

- универсальные для СУБД: ODBC, JDBC
- для реляционных СУБД: Oracle (oracle_fdw), MS SQL, MySQL, DB2, SQLite
- для NoSQL-СУБД
- файловые универсальные: file_fdw, CSV, JSON, TAR, XML, ZIP
- файловые специализированные: геоданные, LDAP, IMAP, RSS, WWW

При миграции данных из других СУБД можно использовать SQL как промежуточный формат. Конечно, это самый универсальный подход, но на практике при значительных объемах данных он неприменим. К тому же из-за разницы в диалектах SQL может потребоваться дополнительное преобразование данных.

Дополнительные удобства и более высокую производительность можно получить, используя механизм оберток сторонних данных (foreign data wrappers), встроенный в PostgreSQL и позволяющий обращаться к внешним данным как к локальным таблицам. Штатная обертка file_fdw позволяет читать данные из файлов различных форматов. В составе дистрибутива Postgres Pro Enterprise есть пакет oracle-fdw-ent-13 с расширением oracle_fdw, устанавливающий одноименную обертку. Помимо этого, существует множество сторонних оберток для различных СУБД, форматов файлов, сетевых протоколов, для специализированных форматов данных и др.

https://wiki.postgresql.org/wiki/Foreign_data_wrappers

Миграция объектов

Объекты SQL

- индексы, представления, последовательности

Подпрограммы

- процедуры, функции, триггеры
- orafce

Другие объекты

Миграция стандартных объектов, таких как индексы, представления, последовательности, относительно проста и поддается автоматизации.

Миграция других объектов — обычно более сложная задача, поскольку большинство из них реализованы в Postgres Pro Enterprise по-другому или вовсе не имеет аналогов. Например, пакеты Oracle не имеют аналогов в Postgres Pro версии 13 (они реализованы в версии 15, но синтаксис и возможности немного разные).

Миграция процедурного кода: функций, процедур, триггеров — гораздо сложнее из-за разнообразия логики, различий используемых языков и модулей. В составе дистрибутива Postgres Pro Enterprise есть пакет `orafce-ent-13`, устанавливающий расширение `orafce`. Расширение добавляет в базу данных Postgres Pro объекты, аналогичные объектам СУБД Oracle, это упрощает перенос процедурной логики.

https://github.com/orafce/orafce

Существует довольно большое количество сторонних решений, автоматизирующих процесс миграции; они экономят усилия и снижают количество ошибок. В статьях и выступлениях на конференциях описано множество успешных проектов по переносу информационных систем с других СУБД на PostgreSQL или Postgres Pro Enterprise, откуда также можно почерпнуть идеи и подходы. Однако в целом задача очень многообразна и сложна, поэтому автоматизировать ее полностью не представляется возможным.

Итоги

Postgres Pro Enterprise можно установить «по умолчанию» или выборочно

Имеются средства получения дополнительной информации

Поддерживается автоматический перенос данных из PostgreSQL и редакций Postgres Pro

Миграция с других СУБД — более сложная задача

Практика

1. Какой размер имеет сейчас буферный кеш сервера Postgres Pro Enterprise? Увеличьте его на 10 % с помощью команды ALTER SYSTEM.
2. Включите вывод в журнал сообщений информации обо всех выполняемых запросах, добавив конфигурационный файл в каталог conf.d.
   Запустите стандартный тест pgbench на 90 секунд.
3. Найдите в журнале сообщений информацию о выполненных запросах. Проанализируйте журнал с помощью утилиты pgbadger.

2. Создайте .conf-файл с произвольным именем в подкаталоге conf.d каталога PGDATA кластера и поместите в него необходимые настройки конфигурационных параметров. Содержимое этого файла будет учтено в конфигурации благодаря директиве *include_dir*, находящейся в конце конфигурационного файла postgresql.conf.

1. Размер кеша

Текущее значение параметра:

```
=> SHOW shared_buffers;
```

```
 shared_buffers
----------------
 242MB
(1 row)
```

Задаем на 10% больше:

```
=> ALTER SYSTEM SET shared_buffers='266 MB';
```

```
ALTER SYSTEM
```

Перезапуск и проверка:

```
=> \q
```

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

```
student$ psql
```

```
=> SHOW shared_buffers;
```

```
 shared_buffers
----------------
 266MB
(1 row)
```

2. Тест pgbench

Инициализируем pgbench в отдельной базе данных

```
=> CREATE DATABASE install;
```

```
CREATE DATABASE
```

```
student$ /opt/pgpro/ent-13/bin/pgbench -i install
```

```
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
100000 of 100000 tuples (100%) done (elapsed 0.05 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 0.17 s (drop tables 0.00 s, create tables 0.01 s, client-side generate 0.08 s, vacuum 0.04 s, primary keys 0.04 s).
```

Включаем вывод в журнал:

```
postgres$ echo log_min_duration_statement=0 > /var/lib/pgpro/ent-13/conf.d/install.conf
```

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

Проверяем значение параметра:

```
=> SHOW log_min_duration_statement;
```

```
 log_min_duration_statement
----------------------------
 0
(1 row)
```

Тест на 90 секунд

```
student$ /opt/pgpro/ent-13/bin/pgbench -T 90 -P 10 install
```

```
starting vacuum...end.
progress: 10.0 s, 383.0 tps, lat 2.606 ms stddev 0.694
progress: 20.0 s, 361.7 tps, lat 2.761 ms stddev 0.452
progress: 30.0 s, 366.2 tps, lat 2.727 ms stddev 0.606
progress: 40.0 s, 352.6 tps, lat 2.832 ms stddev 0.271
progress: 50.0 s, 353.9 tps, lat 2.822 ms stddev 0.306
progress: 60.0 s, 352.0 tps, lat 2.837 ms stddev 0.300
progress: 70.0 s, 362.9 tps, lat 2.751 ms stddev 0.371
progress: 80.0 s, 367.2 tps, lat 2.719 ms stddev 0.490
progress: 90.0 s, 352.1 tps, lat 2.836 ms stddev 0.311
transaction type: <builtin: TPC-B (sort of)>
default transaction isolation level: read committed
transaction maximum tries number: 1
scaling factor: 1
query mode: simple
number of clients: 1
number of threads: 1
duration: 90 s
number of transactions actually processed: 32516
latency average = 2.764 ms
latency stddev = 0.455 ms
tps = 361.287772 (including connections establishing)
tps = 361.294879 (excluding connections establishing)
```

3. Просмотр журнала и отчет pgBadger

Файлы журнала сообщений находятся в /var/lib/pgpro/ent-13/log

```
postgres$ ls -Ct /var/lib/pgpro/ent-13/log
```

```
postgresql-2024-01-16_214300.log  postgresql-2024-01-16_214157.log
postgresql-2024-01-16_214229.log  postgresql-2024-01-12_111205.log
postgresql-2024-01-16_214158.log
```

Несколько последних записей журнала

```
postgres$ tail /var/lib/pgpro/ent-13/log/postgresql-2024-01-16_214300.log
```

```
2024-01-16 21:43:28.873 MSK [293133] LOG:  duration: 0.139 ms  statement: UPDATE pgbench_branches SET bbalance = bbalance + 4135 WHERE bid = 1;
2024-01-16 21:43:28.874 MSK [293133] LOG:  duration: 0.101 ms  statement: INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (2, 1, 72974, 4135, CURRENT_TIMESTAMP);
2024-01-16 21:43:28.875 MSK [293133] LOG:  duration: 1.184 ms  statement: END;
2024-01-16 21:43:28.875 MSK [293133] LOG:  duration: 0.016 ms  statement: BEGIN;
2024-01-16 21:43:28.875 MSK [293133] LOG:  duration: 0.195 ms  statement: UPDATE pgbench_accounts SET abalance = abalance + -4570 WHERE aid = 22275;
2024-01-16 21:43:28.876 MSK [293133] LOG:  duration: 0.140 ms  rows: 1  size: 11 bytes  statement: SELECT abalance FROM pgbench_accounts WHERE aid = 22275;
2024-01-16 21:43:28.876 MSK [293133] LOG:  duration: 0.159 ms  statement: UPDATE pgbench_tellers SET tbalance = tbalance + -4570 WHERE tid = 1;
2024-01-16 21:43:28.876 MSK [293133] LOG:  duration: 0.140 ms  statement: UPDATE pgbench_branches SET bbalance = bbalance + -4570 WHERE bid = 1;
2024-01-16 21:43:28.882 MSK [293133] LOG:  duration: 5.958 ms  statement: INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (1, 1, 22275, -4570, CURRENT_TIMESTAMP);
2024-01-16 21:43:28.886 MSK [293133] LOG:  duration: 3.231 ms  statement: END;
```

pgBadger уже установлен в виртуальной машине:

```
postgres$ pgbadger --version
```

```
pgBadger version 11.6
```

Утилите нужно передать:

- имя html-файла для отчета
- формат журнала сообщений
- файл журнала или список файлов

Запускаем:

```
student$ pgbadger --outfile=/tmp/out.html --format=stderr /var/lib/pgpro/ent-13/log/postgresql-2024-01-16_214300.log
```

```
[>                        ] Parsed       0 bytes of 9298300 (0.00%), queries: 0, events: 0[>                        ] Parsed    1390 bytes of 9298300 (0.01%), queries: 8, events: 0[>
LOG: Ok, generating html report...
```

Утилита сгенерировала отчет в файле out.html.

```
student$ ls -l /tmp/out.html
```

```
-rw-rw-r-- 1 student student 889170 янв 16 21:43 /tmp/out.html
```

Заглянем в отчет:

Открываем файл /tmp/out.html...

```
student$ xdg-open /tmp/out.html
```