Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов

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

Темы

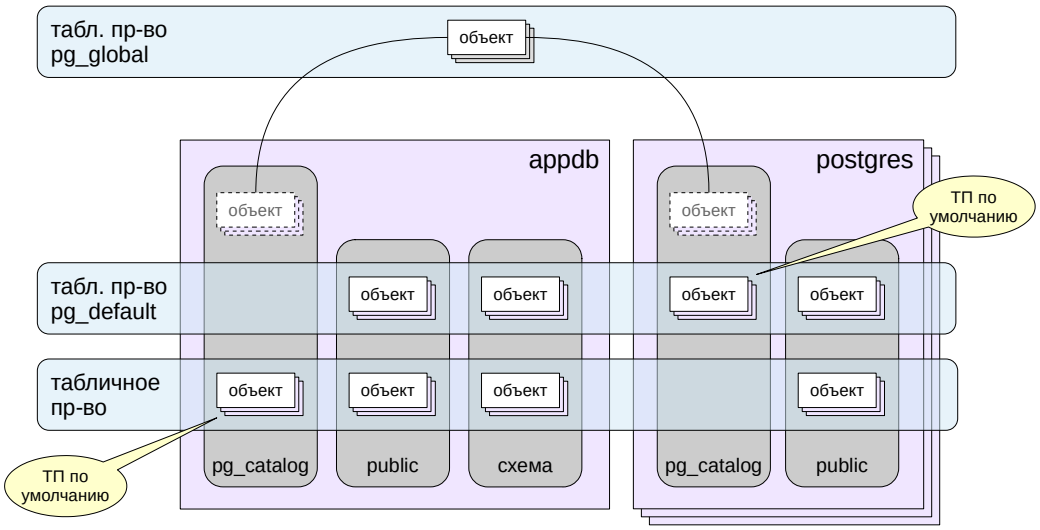
Табличные пространства и каталоги

Создание, изменение и удаление табличных пространств

Хранение данных в файловой системе

Перемещение данных

Табличные пространства



Табличные пространства (ТП) служат для организации физического хранения данных и определяют расположение данных в файловой системе.

Например, можно создать одно ТП на медленных дисках для архивных данных, а другое – на быстрых дисках для данных, с которыми идет активная работа.

При инициализации кластера создаются два ТП: pg_default и pg_global.

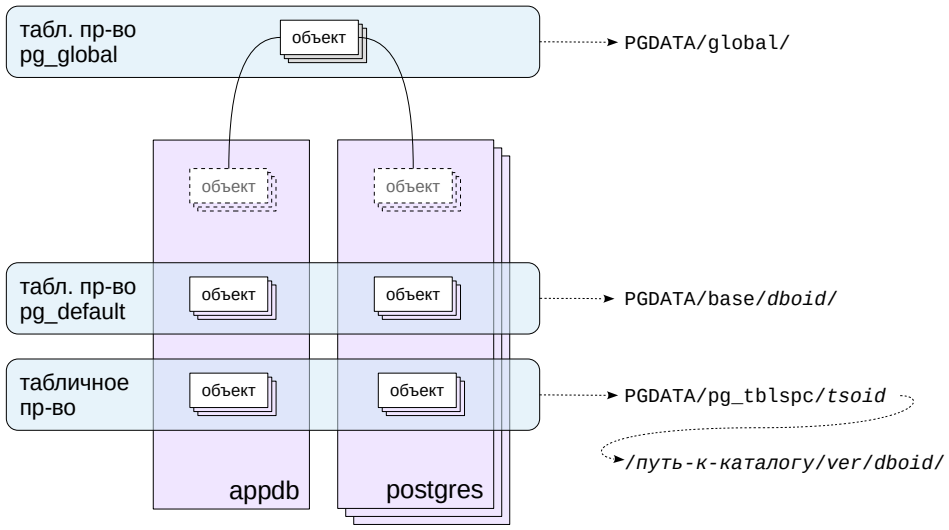
Одно и то же ТП может использоваться разными базами данных, а одна база данных может хранить данные в нескольких ТП.

При этом у каждой БД есть так называемое «ТП по умолчанию», в котором создаются все объекты, если не указано иное. В этом же ТП хранятся и объекты системного каталога. Изначально в качестве «ТП по умолчанию» используется ТП pg_default, но можно установить и другое.

ТП pg_global особенное: в нем хранятся те объекты системного каталога, которые являются общими для кластера.

https://postgrespro.ru/docs/postgresql/13/manage-ag-tablespaces

Каталоги



По сути, табличное пространство — это указание на каталог, в котором располагаются данные. Стандартные ТП pg_global и pg_default всегда находятся в PGDATA/global/ и PGDATA/base/ соответственно. При создании пользовательского ТП указывается произвольный каталог; для собственного удобства PostgreSQL создает на него символьную ссылку в каталоге PGDATA/pg_tblspc/.

Внутри каталога PGDATA/base/ данные дополнительно разложены по подкаталогам баз данных (для PGDATA/global/ это не требуется, так как данные в нем относятся к кластеру в целом).

Внутри каталога пользовательского ТП появляется еще один уровень вложенности: версия сервера PostgreSQL. Это сделано для удобства обновления сервера на другую версию.

Собственно объекты хранятся в файлах внутри этих каталогов — каждый объект в отдельных файлах.

Служебные табличные пространства

При создании кластера создаются два табличных пространства:

```
=> SELECT * FROM pg_tablespace;
```

```
 oid  |  spcname   | spcowner | spcacl | spcoptions
------+------------+----------+--------+------------
 1663 | pg_default |       10 |        |
 1664 | pg_global  |       10 |        |
(2 rows)
```

- pg_global — общие объекты кластера;
- pg_default — табличное пространство по умолчанию.

Пользовательские табличные пространства

Для нового табличного пространства нужен пустой каталог, владельцем которого является пользователь postgres.

```
student$ sudo mkdir /var/lib/postgresql/ts_dir
```

```
student$ sudo chown postgres /var/lib/postgresql/ts_dir
```

Теперь можно создать табличное пространство:

```
=> CREATE TABLESPACE ts LOCATION '/var/lib/postgresql/ts_dir';
```

```
CREATE TABLESPACE
```

Список табличных пространств можно получить и командой psql:

```
=> \db
```

```
                List of tablespaces
    Name    |  Owner   |          Location
------------+----------+----------------------------
 pg_default | postgres |
 pg_global  | postgres |
 ts         | student  | /var/lib/postgresql/ts_dir
(3 rows)
```

У каждой базы данных есть табличное пространство «по умолчанию». Создадим базу и назначим ей ts в качестве такого пространства:

```
=> CREATE DATABASE appdb TABLESPACE ts;
```

```
CREATE DATABASE
```

Теперь все создаваемые таблицы и индексы будут попадать в ts, если явно не указать другое.

Подключимся к базе:

```
=> \c appdb
```

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

Создадим таблицу:

```
=> CREATE TABLE t1(
  id integer GENERATED ALWAYS AS IDENTITY,
  name text
);
```

```
CREATE TABLE
```

При создании объекта можно указать табличное пространство явно:

```
=> CREATE TABLE t2(
  n numeric
) TABLESPACE pg_default;
```

```
CREATE TABLE
```

```
=> SELECT tablename, tablespace FROM pg_tables WHERE schemaname = 'public';
```

```
 tablename | tablespace
-----------+------------
 t1        |
 t2        | pg_default
(2 rows)
```

Пустое поле tablespace указывает на табличное пространство по умолчанию, а у второй таблицы поле заполнено.

Еще один способ задать табличное пространство без явного указания при создании объекта — предварительно установить табличное пространство в параметре default_tablespace.

Одно табличное пространство может использоваться для объектов нескольких баз данных.

```
=> CREATE DATABASE configdb;
```

```
CREATE DATABASE
```

У этой БД табличным пространством по умолчанию будет pg_default.

```
=> \c configdb
```

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

```
=> CREATE TABLE t(
  n integer
) TABLESPACE ts;
```

```
CREATE TABLE
```

Управление объектами в табличных пространствах

Таблицы (и другие объекты, например, индексы), можно перемещать между табличными пространствами.

```
=> \c appdb
```

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

```
=> ALTER TABLE t1 SET TABLESPACE pg_default;
```

```
ALTER TABLE
```

```
=> SELECT tablename, tablespace FROM pg_tables WHERE schemaname = 'public';
```

```
 tablename | tablespace
-----------+------------
 t2        | pg_default
 t1        | pg_default
(2 rows)
```

Можно переместить и все объекты из одного табличного пространства в другое:

```
=> ALTER TABLE ALL IN TABLESPACE pg_default SET TABLESPACE ts;
```

```
ALTER TABLE
```

```
=> SELECT tablename, tablespace FROM pg_tables WHERE schemaname = 'public';
```

```
 tablename | tablespace
-----------+------------
 t2        |
 t1        |
(2 rows)
```

Важно понимать, что перенос в другое табличное пространство (в отличие от переноса в другую схему) — физическая операция, связанная с копированием файлов данных из каталога в каталог. На время ее выполнения доступ к перемещаемому объекту полностью блокируется.

Размер табличного пространства

Мы уже рассматривали, как узнать объем, занимаемый базой данных. Можно узнать и объем объектов в табличном пространстве:

```
=> SELECT pg_size_pretty( pg_tablespace_size('ts') );
```

```
 pg_size_pretty
----------------
 7997 kB
(1 row)
```

Почему размер так велик, хотя в табличном пространстве всего несколько пустых таблиц?

Поскольку ts является табличным пространством по умолчанию для базы appdb, в нем хранятся объекты системного каталога. Они и занимают место.

В psql получить размер табличных пространств можно командой:

```
=> \db+
```

```
                                       List of tablespaces
    Name    |  Owner   |          Location          | Access privileges | Options |  Size   | Description
------------+----------+----------------------------+-------------------+---------+---------+-------------
 pg_default | postgres |                            |                   |         | 38 MB   |
 pg_global  | postgres |                            |                   |         | 575 kB  |
 ts         | student  | /var/lib/postgresql/ts_dir |                   |         | 7997 kB |
(3 rows)
```

Удаление табличного пространства

Табличное пространство можно удалить, но только в том случае, если оно пусто:

```
=> DROP TABLESPACE ts;
```

```
ERROR:  tablespace "ts" is not empty
```

В отличие от удаления схемы, в команде DROP TABLESPACE нельзя использовать фразу CASCADE: объекты табличного пространства могут принадлежать разным базам данных, а подключены мы только к одной.

Но можно выяснить, в каких базах есть зависимые объекты. В этом нам поможет системный каталог.

Сначала узнаем и запомним идентификатор табличного пространства:

```
=> SELECT oid FROM pg_tablespace WHERE spcname = 'ts';
```

```
  oid
-------
 16498
(1 row)
```

Затем получим список баз данных, в которых есть объекты из удаляемого пространства:

```
=> SELECT datname
FROM pg_database
WHERE oid IN (SELECT pg_tablespace_databases(16498));
```

```
 datname
----------
 configdb
 appdb
(2 rows)
```

Дальше подключаемся к каждой базе данных и получаем список объектов из pg_class:

```
=> \c configdb
```

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

```
=> SELECT relnamespace::regnamespace, relname, relkind
FROM pg_class
WHERE reltablespace = 16498;
```

```
 relnamespace | relname | relkind
--------------+---------+---------
 public       | t       | r
(1 row)
```

Таблица больше не нужна, удалим ее.

```
=> DROP TABLE t;
```

```
DROP TABLE
```

И вторая база данных. Поскольку ts является табличным пространством по умолчанию, у объектов в pg_class идентификатор табличного пространства равен нулю. Это, как нам уже известно, объекты системного каталога:

```
=> \c appdb
```

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

```
=> SELECT count(*) FROM pg_class WHERE reltablespace = 0;
```

```
 count
-------
   350
(1 row)
```

Табличное пространство по умолчанию можно сменить; при этом все таблицы из старого пространства физически переносятся в новое. Предварительно надо отключиться от базы.

```
=> \c postgres
```

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

```
=> ALTER DATABASE appdb SET TABLESPACE pg_default;
```

```
ALTER DATABASE
```

Вот теперь табличное пространство может быть удалено.

```
=> DROP TABLESPACE ts;
```

```
DROP TABLESPACE
```

Итоги

Табличные пространства — средство для организации физического хранения данных

Логическое (базы данных, схемы) и *физическое* (табличные пространства) разделения данных независимы

Практика

Почему при создании базы данных без предложения TABLESPACE табличным пространством по умолчанию становится pg_default?

1. Создайте новое табличное пространство.
2. Измените табличное пространство по умолчанию для базы данных template1 на созданное пространство.
3. Создайте новую базу данных.
   Проверьте, какое табличное пространство по умолчанию установлено для новой базы данных.
4. Посмотрите в файловой системе символическую ссылку в PGDATA на каталог табличного пространства.
5. Удалите созданное табличное пространство.

1. Новое табличное пространство

```
student$ sudo mkdir /var/lib/postgresql/ts_dir
```

```
student$ sudo chown postgres /var/lib/postgresql/ts_dir
```

```
=> CREATE TABLESPACE ts LOCATION '/var/lib/postgresql/ts_dir';
```

```
CREATE TABLESPACE
```

2. Табличное пространство по умолчанию для template1

```
=> ALTER DATABASE template1 SET TABLESPACE ts;
```

```
ALTER DATABASE
```

3. Новая база данных и проверка

```
=> CREATE DATABASE db;
```

```
CREATE DATABASE
```

```
=> SELECT spcname
FROM pg_tablespace
WHERE oid = (SELECT dattablespace FROM pg_database WHERE datname = 'db');
```

```
 spcname
---------
 ts
(1 row)
```

Табличное пространство по умолчанию — ts.

Вывод: табличное пространство по умолчанию определяется шаблоном, из которого клонируется новая база данных.

4. Символическая ссылка

```
=> SELECT oid AS tsoid FROM pg_tablespace WHERE spcname = 'ts';
```

```
 tsoid
-------
 16703
(1 row)
```

```
student$ sudo ls -l /var/lib/postgresql/13/main/pg_tblspc/16703
```

```
lrwxrwxrwx 1 postgres postgres 26 мая 11 13:21 /var/lib/postgresql/13/main/pg_tblspc/16703 -> /var/lib/postgresql/ts_dir
```

5. Удаление табличного пространства

```
=> ALTER DATABASE template1 SET TABLESPACE pg_default;
```

```
ALTER DATABASE
```

```
=> DROP DATABASE db;
```

```
DROP DATABASE
```

```
=> DROP TABLESPACE ts;
```

```
DROP TABLESPACE
```

Практика+

1. Установите параметр *random_page_cost* в значение 1.1 для табличного пространства pg_default.

1. Используйте команду ALTER TABLESPACE ... SET:

https://postgrespro.ru/docs/postgresql/13/sql-altertablespace

Параметры *seq_page_cost* и *random_page_cost* используются планировщиком запросов и задают примерную стоимость чтения с диска одной страницы данных при последовательном и произвольном доступе соответственно.

Чем меньше соотношение между этими параметрами, тем чаще планировщик будет предпочитать индексный доступ последовательному сканированию таблицы.

Более подробно параметры **_cost,* и в частности *random_page_cost*, рассматриваются в курсе QPT «Оптимизация запросов».

1. Установка random_page_cost для табличного пространства

Значения по умолчанию параметров seq_page_cost и random_page_cost больше подходят для медленных HDD-дисков. Предполагается, что доступ к произвольной странице данных в четыре раза дороже последовательного:

```
=> SELECT name, setting
FROM pg_settings
WHERE name IN ('seq_page_cost','random_page_cost');
```

```
       name       | setting
------------------+---------
 random_page_cost | 4
 seq_page_cost    | 1
(2 rows)
```

Если используются диски с разными характеристиками, для них можно создать разные табличные пространства и настроить подходящие соотношения этих параметров. Например для быстрых SSD-дисков значение random_page_cost можно уменьшить практически до значения seq_page_cost.

```
=> ALTER TABLESPACE pg_default SET (random_page_cost = 1.1);
```

```
ALTER TABLESPACE
```

Настройки, сделанные командой ALTER TABLESPACE, сохраняются в таблице pg_tablespace. Их можно посмотреть в psql следующей командой:

```
=> \db+
```

```
                                       List of tablespaces
    Name    |  Owner   | Location | Access privileges |        Options         |  Size  | Description
------------+----------+----------+-------------------+------------------------+--------+-------------
 pg_default | postgres |          |                   | {random_page_cost=1.1} | 31 MB  |
 pg_global  | postgres |          |                   |                        | 575 kB |
(2 rows)
```

Параметры *_cost можно установить и в postgresql.conf. Тогда они будут действовать для всех табличных пространств.