

Data organization

Tablespaces



Copyright

© Postgres Professional, 2015–2022

Authors: Egor Rogov, Pavel Luzanov, Ilya Bashtanov

Translated by Alexander Meleshko

Use of course materials

Non-commercial use of course materials (presentations, demonstrations) is allowed without restrictions. Commercial use is possible only with the written permission of Postgres Professional. It is prohibited to make changes to the course materials.

Feedback

Please send your feedback, comments and suggestions to:

edu@postgrespro.ru

Disclaimer

Postgres Professional assumes no responsibility for any damages and losses, including loss of income, caused by direct or indirect, intentional or accidental use of course materials. Postgres Professional company specifically disclaims any warranties on course materials. Course materials are provided “as is,” and Postgres Professional company has no obligations to provide maintenance, support, updates, enhancements, or modifications.

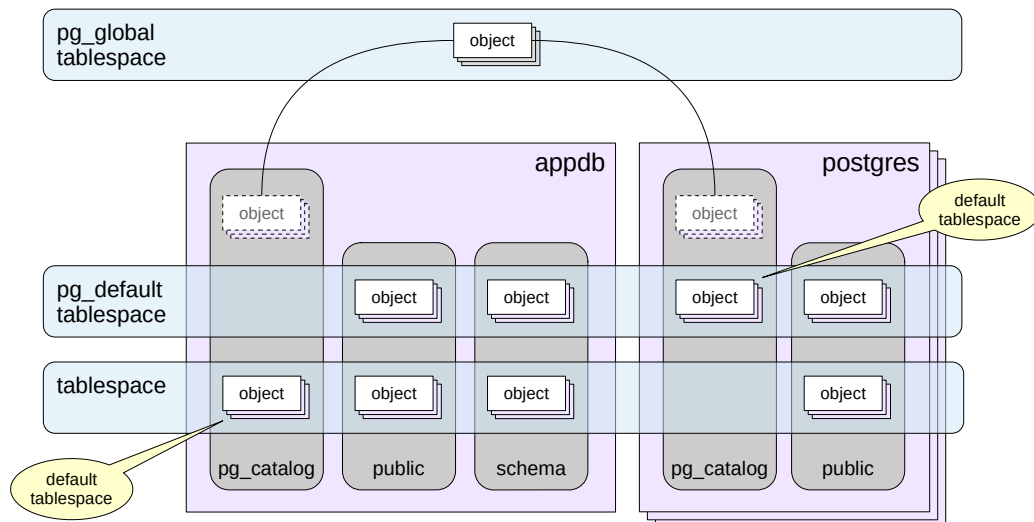
Tablespaces and catalogs

Creating, modifying, and deleting tablespaces

Storing data in the file system

Moving data

Tablespaces



Tablespaces are used to organize the physical storage of data and determine the location of data in the file system.

For example, one tablespace can be used on slow disks for archived data, and another on fast disks with frequent activity.

On cluster initialization, two tablespaces are created: `pg_default` and `pg_global`.

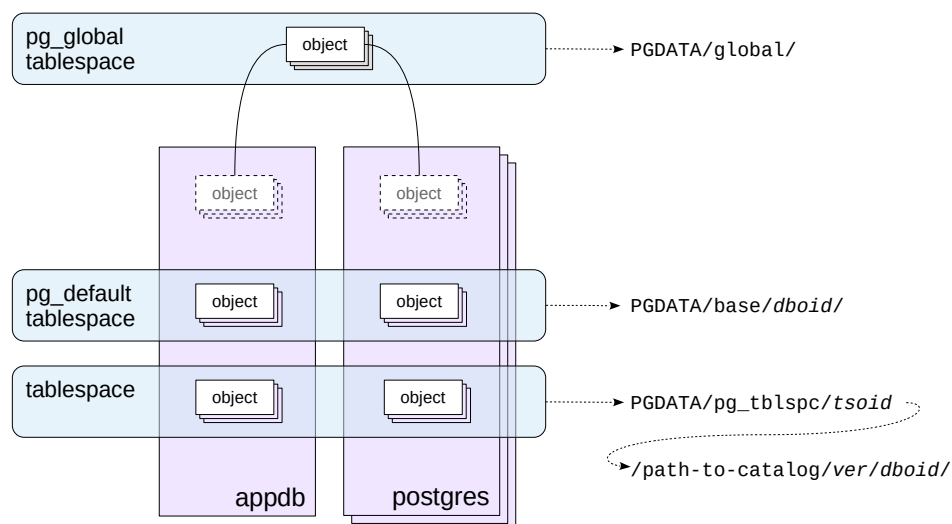
A tablespace can be used by multiple databases, and a database can use multiple tablespaces at once.

Each database has a default tablespace where all database objects are created (unless specified otherwise). System catalog objects are also stored in the default tablespace. Databases will use the `pg_default` tablespace as their default, unless another one is set by the user.

The `pg_global` tablespace is special as it stores only those objects that are shared by the whole cluster.

<https://postgrespro.com/docs/postgresql/13/manage-ag-tablespaces>

Directories



Essentially, a tablespace is a reference to the directory in which the data is located. The standard tablespaces `pg_global` and `pg_default` are always located in `PGDATA/global/` and `PGDATA/base/`, respectively. When a custom tablespace is created, an arbitrary directory can be specified. For convenience, PostgreSQL also creates a symbolic link to the directory in `PGDATA/pg_tblspc/`.

The `PGDATA/base/` directory comprises different directories for each database (unlike `PGDATA/global/`, which stores data referring to the whole cluster).

Inside a custom tablespace directory, there is another level of directories for different PostgreSQL server versions. This is helpful during server upgrade.

Finally, these directories are where the actual objects are stored, one or more files per object.

Service tablespaces

Upon a cluster creation, two tablespaces are generated:

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

oid	spcname	spcowner	spcacl	spcoptions
1663	pg_default	10		
1664	pg_global	10		

(2 rows)

- pg_global — shared cluster objects,
- pg_default — the tablespace to be used as default.

Custom tablespaces

A new tablespace needs an empty directory owned by the user postgres.

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

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

Now, a new tablespace can be created:

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

CREATE TABLESPACE

The following psql command returns a list of all tablespaces:

```
=> \db
```

List of tablespaces		
Name	Owner	Location
pg_default	postgres	
pg_global	postgres	
ts	student	/var/lib/postgresql/ts_dir

(3 rows)

Each database has a “default” tablespace. Let’s create a database and assign ts as its default:

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

CREATE DATABASE

This makes all tables and indexes created within the database fall into ts, unless specified otherwise.

Connect to the database:

```
=> \c appdb
```

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

Create a table:

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

CREATE TABLE

When creating objects, you may explicitly specify a tablespace for it:

```
=> 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)

```

An empty tablespace field means that the default tablespace is used. The second table has this field filled in.

Another way to assign a tablespace without defining it at object creation is to preemptively set it as the `default_tablespace` parameter value.

A single tablespace may contain objects from multiple databases.

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

```
CREATE DATABASE
```

This database's default tablespace will be `pg_default`.

```
=> \c configdb
```

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

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

```
CREATE TABLE
```

Managing objects within tablespaces

Tables (and other objects, such as indexes) can be moved between tablespaces.

```
=> \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)

```

You can move all objects from one tablespace to another:

```
=> 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)

```

Keep in mind that moving objects between tablespaces (unlike moving between schemas) is a physical operation that involves moving actual files from one catalog to another. Access to the moved objects is completely blocked for the duration of the operation.

Tablespace size

We already know how to find the size of a database. Now, we can learn how to get the size of objects in a tablespace:

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

```

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

```

Why is the size large, despite the tablespace containing just a few empty tables?

This is because ts is the default tablespace for the database appdb, so it is where the system catalog objects are stored, occupying the mysterious space.

The psql command to get the size of a tablespace is:

```
=> \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)

Dropping a tablespace

You can only delete a tablespace that is empty:

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

ERROR: tablespace "ts" is not empty

Unlike with schemas, there is no keyword CASCADE in the DROP TABLESPACE command. Objects within the tablespace may belong to multiple databases, while we are only connected to one.

You can still learn what databases contain dependant objects. This is where the system catalog comes in.

First, find and save the tablespace ID:

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

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

Next, get a list of databases which have objects in the tablespace we want to remove:

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

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

Then, connect to each of the databases and get a list of objects from 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)
```

The table is no longer needed, drop it.

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

DROP TABLE

Now, for the second database. Since ts is the default tablespace, the tablespace ID of the objects in pg_class equals zero. These are the system catalog objects, as we already know:

```
=> \c appdb
```

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

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

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

You can set another tablespace as default. This will move all the tables from the old one into the new one. You need to disconnect from the database first.

```
=> \c postgres
```

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

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

```
ALTER DATABASE
```

Finally, the tablespace can be deleted.

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

```
DROP TABLESPACE
```


Tablespaces organize physical data storage

Logical (databases, schemas) and *physical* (tablespaces) forms of data separation are independent

Why does pg_default become the default tablespace when creating a database without specifying the TABLESPACE keyword?

1. Create a new tablespace.
2. Set it as the default tablespace for the template1 database.
3. Create a new database.
Check which default tablespace is set for the new database.
4. Find the symbolic link to the tablespace catalog in PGDATA.
5. Delete the created tablespace.

1. A new tablespace

```
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. The default tablespace for template1

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

3. A new database, verification

```
=> CREATE DATABASE db;
CREATE DATABASE

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

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

The default tablespace is ts.

Conclusion: the default tablespace is determined by the template from which the new database is cloned.

4. Symbolic links

```
=> 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 Mar  7 13:54 /var/lib/postgresql/13/main/pg_tblspc/16703 -> /var/lib/postgresql/ts_dir
```

5. Drop the tablespace

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

=> DROP DATABASE db;
DROP DATABASE

=> DROP TABLESPACE ts;
DROP TABLESPACE
```

1. Set the *random_page_cost* parameter for the pg_default tablespace to 1.1.

1. Use the ALTER TABLESPACE ... SET command:

<https://postgrespro.com/docs/postgresql/13/sql-altertablespace>

The *seq_page_cost* and *random_page_cost* parameters are used by the planner. They refer to the approximate cost of reading one page of data from disk with sequential and random access, respectively.

The lower the ratio between these parameters, the more often the planner will prefer index access over sequential table scanning.

The parameters *_cost and, more specifically, *random_page_cost* are discussed in more detail in the “Query performance tuning” course (QPT).

1. Setting the random_page_cost for a tablespace

The default seq_page_cost and random_page_cost values are better suited for slower HDD drives. It is assumed that random page access is four times as costly as sequential page access:

```
=> 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)

If you use drives with different properties, you can create different tablespaces with appropriate seq_page_cost and random_page_cost values for each. For example, quick SSD drives can have the random_page_cost value almost as low as seq_page_cost.

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

ALTER TABLESPACE

Configuration adjustments made using the ALTER TABLESPACE command are stored in the pg_tablespace table. You can view them in psql with the command:

```
=> \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)

The *_cost parameters can also be set in postgresql.conf. This will apply them to all tablespaces.