

# Data organization

## Logical layout



### **Copyright**

© Postgres Professional, 2017, 2018, 2019.

Authors: Egor Rogov, Pavel Luzanov

### **Use of course materials**

Non-commercial use of course materials (presentations, demonstrations) is permitted without restrictions. Commercial use is possible only with the written permission of Postgres Professional. Changes to course materials are prohibited.

### **Feedback**

Send feedback, comments and suggestions to:

[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

### **Denial of responsibility**

In no event shall Postgres Professional be liable to any party for direct, indirect, special, incidental, or consequential damages, including lost profit, arising out of the use of course materials. Postgres Professional disclaims any warranties on course materials. Course materials are provided on an “as is” basis and Postgres Professional has no obligations to provide maintenance, support, updates, enhancements, or modifications.

Databases and templates

Schemas and search path

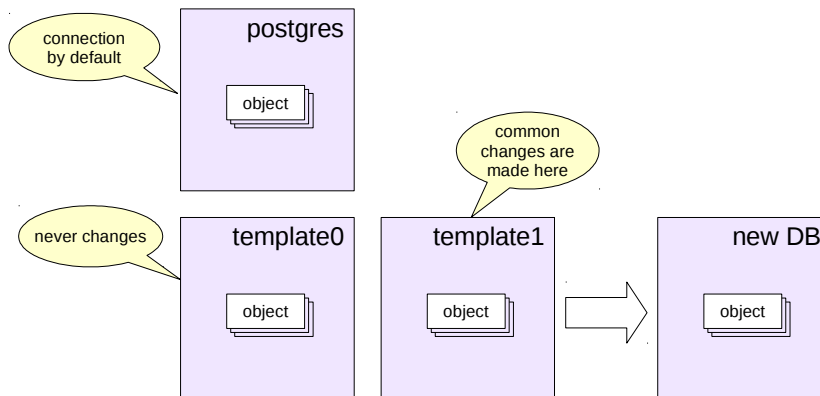
Special schemas

System catalog

# Database cluster

Cluster initialization creates three databases

New database is always cloned from an existing one



3

A PostgreSQL instance manages multiple databases, which are called a *database cluster*. During cluster initialization (by `initdb` command), three identical databases are created. All other databases created later by the user are cloned from an existing one.

**Template1** database is used by default to create new databases. You can add objects and extensions to it that should be copied to each new database.

**Template0** database must not be changed. It needs at least in two situations. First, to restore the database from the backup made by `pg_dump` utility (since not only the objects of this database will be included in this copy, but also the objects installed in `template1`). Second, when creating a new database with an encoding different from that specified during the cluster initialization.

The **postgres** database is used by default for `postgres` user to connect to. It is not mandatory, but some utilities relies on its presence, so it is not recommended to remove it even if it is not needed.

<https://postgrespro.com/docs/postgresql/11/manage-ag-templatedbs.html>

## Objects namespace

- logical grouping of database objects
- prevent naming conflicts between applications

## Schema and user are not the same

## Special schemas

- `public` — used for all objects by default
- `pg_catalog` — system catalog
- `information_schema` — standard representation of the system catalog
- `pg_temp` — for temporary tables
- ...

Schemas are namespaces for database objects. They allow to logically group objects in order to manage them, to prevent name conflicts between multiple users or applications.

In PostgreSQL, the schema and the user are different entities (although the default settings allow users to work comfortably with schemas of the same name).

There are several special schemas that are usually present in each database.

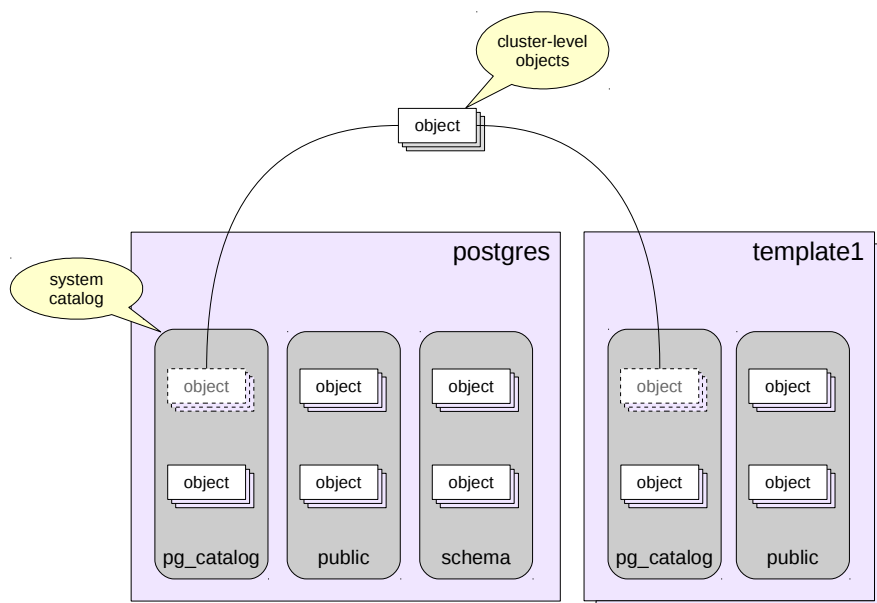
The **public** schema is used by default to store objects, unless other settings are made.

The **pg\_catalog** schema stores system catalog objects. The system catalog is a metainformation about objects belonging to a cluster, which is stored in the cluster itself in special tables. An alternative view of the system catalog (defined in the SQL standard) is provided by the **information\_schema**.

The **pg\_temp** schema is used to store temporary objects, usually tables. (In fact, tables are created in **pg\_temp\_1**, **pg\_temp\_2**, and so on. Each user gets his own temporary schema, but it is always referenced to as `pg_temp`.)

There are other schemas, but they are more technical in nature and not considered here.

<https://postgrespro.com/docs/postgresql/11/ddl-schemas.html>



Schemas belong to databases, database objects are distributed among schemas.

However, several system catalog tables store information that is common to the entire cluster. For example, list of databases, list of users and some other information. These tables are stored outside of any particular database, but they are equally visible from each database in `pg_catalog` schema.

## Determine the schema of an object

qualified name (*schema.name*) explicitly identifies the schema

unqualified name is looked up in the schemas specified in the search path

## Search path

specified by the *search\_path* parameter,  
*current\_schemas()* shows the actual value

non-existent and not available schemas are excluded

pg\_temp and pg\_catalog schemas are implicitly included in the first place,  
unless specified in *search\_path*

the first available schema explicitly specified in the search path  
is used to create objects

When specifying an object, it is necessary to determine which schema we are talking about, since objects with the same name can be stored in different schemas.

If a name of the object is qualified by a name of the schema, then the explicitly specified schema is used. If an unqualified name is used, PostgreSQL tries to look up the name in schemas listed in the search path, which is determined by the *search\_path* configuration parameter.

The actual search path may differ from the value of the *search\_path* parameter. The actual search path does not include non-existent schemas from *search\_path*, as well as schemas to which the user does not have access (one of the subsequent course topics is devoted to access control). In addition, at the beginning of the search path are added implicitly:

- pg\_catalog schema to always have access to the system catalog;
- pg\_temp schema if the user has created any temporary objects.

The actual search path, including implicit schemas, is returned by the *current\_schemas(true)* function call. Schemas are looked up in the specified order, from left to right. If there is no desired object in the first schema, the search continues in the following one and so on.

When you create a new object with an unqualified name, it falls into the first explicitly specified and available schema in the search path.

You can draw an analogy between the *search\_path* parameter and PATH variable in operating systems.

<https://postgrespro.com/docs/postgresql/11/runtime-config-client.html#guc-search-path>

## Description of all database cluster objects

- set of tables in each database (in `pg_catalog` schema) and several cluster-level objects
- set of views for convenience

## Access

- by SQL queries or special `psql` commands

## Conventions

- table names begin with `pg_`
- column names contain a three letter prefix
- hidden «oid» column of OID datatype as primary key
- object names are always stored in lower case

The system catalog stores meta-information about all cluster objects. It consists of a set of tables in each database (in `pg_catalog` schema) and several cluster-level tables. For convenience, several views are also defined on the tables.

<https://postgrespro.com/docs/postgresql/11/catalogs>

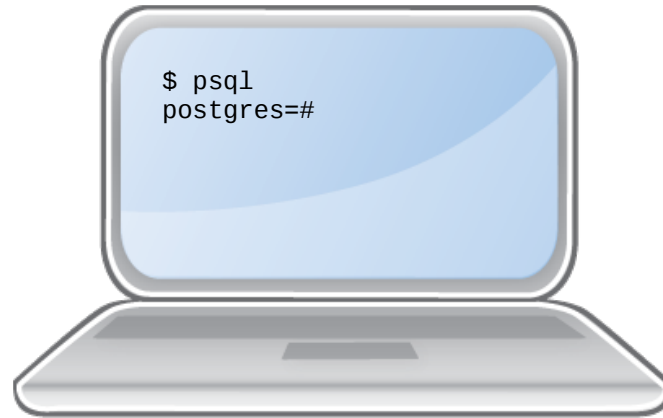
The system catalog can be accessed using ordinary SQL queries, and DDL commands lead to changes in the system catalog data. In addition, `psql` has a number of commands that allow you to conveniently browse the system catalog.

<https://postgrespro.com/docs/postgresql/11/app-psql.html>

All system catalog table names begin with `pg_`, for example, `pg_database`. Table columns begin with a prefix, usually corresponding to the table name, for example, `datname`. Object names are stored in lower case, for example, `'postgres'`.

The system catalog tables have primary keys and are linked by foreign keys. The primary key column is always called `oid` and has a special datatype of OID, an Object Identifier (32-bit integer). This is a hidden column that can be seen only by explicitly specifying its name.

<https://postgrespro.com/docs/postgresql/11/datatype-oid>





## Logically

- cluster contains databases

- database contains schemas

- schema contains specific objects (tables, indexes etc.)

Databases are created by cloning an existing database

Object schema is determined by the search path

Metadata of the contents of the database cluster is stored in the system catalog

1. Create a new database and connect to it.
2. Create a schema named student (the same as the user).
3. Create a schema named app.
4. Create several tables in both schemas.
5. Using psql get a description of the created schemas and a list of all tables in them.
6. Set the search path so that when connected to the database, tables from both schemas are accessible by unqualified names; the student schema must take precedence.
7. Test the settings.