Engineer

Schema

a named group of tables inside one database

What it is

A schema is a namespace inside a database: a container for tables, views, data types, functions and operators, so that the same name can be used in two schemas without conflict. PostgreSQL's manual compares schemas to directories in a file system, except that they cannot be nested: a database holds schemas, and schemas hold objects. The word has a second, looser sense, just as common: the schema of a database is its whole structure, every table, column, key and constraint. Both senses appear below, and the plural, schemas, almost always means the namespaces.

A billing area of an application can live in its own schema, and a session can choose which schemas it looks in:

```
CREATE SCHEMA billing;

CREATE TABLE billing.invoices (
  id     bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  total  numeric(12, 2) NOT NULL
);

SHOW search_path;                  -- "$user", public
SET search_path = billing, public;
SELECT count(*) FROM invoices;     -- now resolves to billing.invoices
```

A name written with its schema, such as `billing.invoices`, always means the same table. A name written without one is looked up through search_path, a list of schemas the server walks in order until it finds a match. The default list is `"$user", public`: a schema named after the current role, if one exists, then the public schema every new database starts with. The first existing schema in the path is also where CREATE TABLE puts a table whose name carries no schema,

which is why objects land in public unless someone says otherwise. A name found in none of the listed schemas raises an error, even when another schema of the same database holds it.

Use the concept in a decision

Put “What is a schema?” to work in a real decision. We can help relate the concept to the system you are building or buying. A structure that matches how the business actually works, so the questions you need answered later are still possible to ask.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Designed and ran 6 ETL/ELT pipelines — 6 ETL pipelines across three database engines

Designed and ran 6 ETL/ELT pipelines Get in touch →

Why it matters

The first reason is organisation. A database with a few hundred tables needs a level above the table, and schemas provide it: billing in one, accounts in another, a third-party tool’s tables in a third, clear of everyone else’s names. A client connection reaches a single database, but schemas are not rigidly separated: a role can use objects in any schema of the database it is connected to, privileges permitting, and one query can join tables from several of them.

The second reason is access control. USAGE on a schema lets a role look up the objects inside it, and CREATE lets it add new ones. Taking USAGE away blocks access to the objects, although their names can still be read from the system catalogs, so a schema guards the data and not the knowledge of what exists. `CREATE SCHEMA name AUTHORIZATION role` hands a new schema to a role, and names beginning with `pg_` are reserved for the system.

The third reason is security, because schemas take part in name resolution. Most names in queries are written without a schema, so, in the manual’s words, adding a schema to search_path effectively trusts every role that can create objects in it. Such a role can make an ordinary query run functions of its choosing, as though the person running the query had called them. The secure patterns keep schemas that others can write to off a user’s path, for example one private schema per user, created with `CREATE SCHEMA alice AUTHORIZATION alice`, which the `"$user"` entry then finds first. PostgreSQL 15 moved new databases in that direction: it removed the right of every role to create objects in public and made the database owner, through the pg_database_owner role, the owner of public.

Portability is the last point. The SQL standard has schemas but no public schema, and the manual notes that some SQL systems might not implement schemas at all. Qualified names travel better than code that leans on search_path.

What it looks like in practice

The data model in Modeled 348 tables across 34 Postgres schemas uses schemas in the first sense, as the level above the table. The maritime domain, with its professionals, companies, ships, jobs and reviews, is laid out as normalized PostgreSQL schemas organised by domain: 348 tables across 34 of them. Table names are plural inside singular-named schemas, without exception, which also sidesteps a lot of reserved-word collisions, and real foreign keys and constraints hold the relationships between them. Adding a feature usually means extending the schema along the existing grain rather than working against it, which the page calls the only reason 34 schemas is a structure rather than a sprawl.

A schema can also be a boundary between two jobs. In Built Stripe and in-app purchase entitlements Stripe card payments and Apple and Google in-app purchases all converge on a payments schema of 9 tables and 34 functions. A separate access schema of 11 tables and 34 functions answers “may this account do this?” without knowing which storefront paid for it, and that answer, re-checked on the server on every request, gates the company directory, search and data export. The cost, as the page puts it, is two schemas where a smaller product would want one, plus an entitlement lookup on requests that would otherwise have been free.

The second sense of the word is the subject of 1,022 migrations behind a schema drift gate. Most projects describe a schema twice, in the initial setup that builds it from scratch and in the migrations that grew it, and the two drift apart unless something checks them. Here the database is versioned as 1,022 migrations, numbered from 036 through 1102, and a CI job builds it twice on every change, once fresh and once with every migration replayed in order, then compares the two: the seeded data as well as the structure. A migration that backfills a lookup table wrongly does as much damage as one that forgets a column, and only one of those shows up in a schema diff. Any disagreement fails the build with the difference printed, so every migration has to work replayed and from cold.

Where it goes wrong

A function planted in a shared schema. The PostgreSQL wiki's guide to CVE-2018-1058 shows the attack: one user creates `lower(varchar)` in public. The built-in `lower` lives in pg_catalog and takes text, so for a varchar column the planted function is the closer match, and a colleague who runs `SELECT lower(full_name) FROM a` runs the planted code instead. Qualify calls in sensitive code as `pg_catalog.lower`, take CREATE on public away from the roles that do not need it, and list what is already there with `\df public.*`.

An upgrade that kept the old permissions. PostgreSQL 15's safer default applies to new clusters and newly created databases. Upgrading a cluster or restoring a dump preserves the permissions public already had, so a server on version 15 or later can still let every role create objects there. The manual calls the default path with CREATE granted on public never a secure pattern, and notes that it is the default in databases upgraded from 14 or earlier. In such a database, run `REVOKE CREATE ON SCHEMA public FROM PUBLIC`, then audit public for objects named like those in pg_catalog.

A table created in the wrong place. A migration creates `invoices` without a schema, and the table goes into the first existing schema on the path of whichever role runs it. The application, connecting with another path, then reads a different table or gets `relation "invoices" does not exist`. Qualify every name in migrations, and check `SELECT current_schema()` in the session that runs them.

A drop that reaches past the schema. `DROP SCHEMA billing` refuses while the schema holds anything, because RESTRICT is the default. With CASCADE it drops every object inside and every object that depends on those, and the manual warns that this can remove objects in schemas other than the one named. Its advice is to run the DROP without CASCADE first and read the DETAIL lines, which list what would go.

A name the system resolves first. pg_catalog is implicitly searched before the schemas listed in search_path when the path leaves it out. When it is named explicitly, its position in the path determines the search order. A table of your own whose name begins with `pg_` can be shadowed when a later version adds a system table of the same name, so the manual advises against such names. Reserved words such as `user` and `order` cause a milder version of the trouble: a table named that way needs double quotes in every statement.

Further reading

- 5.10. Schemas — the manual's chapter: qualified names, the public schema, search_path, secure usage patterns and portability.
- CREATE SCHEMA — the command's syntax, the AUTHORIZATION clause, the reserved `pg_` prefix and where PostgreSQL departs from the standard.
- 5.8. Privileges — what USAGE and CREATE mean on a schema, alongside every other privilege.
- E.20. Release 15 — the release that removed CREATE on public from new databases and kept it for upgraded ones.
- A Guide to CVE-2018-1058: Protect Your Search Path — the project's walk-through of the planted-function attack, with the fixes and the commands to audit a database.

Glossary

Choose for your actual constraints

Bring a representative dataset, the questions it must answer and the current bottleneck. We can agree how to measure quality, throughput and recovery.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is a schema?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/schema/