TP No. 1 — Part 1: Getting started with PostgreSQL

(On Windows)

M. Laïdi FOUGHALI

l.foughali@univ-skikda.dz

(Website ⇒ www.al-moualime.com)

University of Skikda — Department of Computer Science

1st Year Master RSD/AI

Advanced Databases (BDDA)

Updated on 2026-09-22 at 22:27:00

Outline

What is PostgreSQL?

PostgreSQL is an **Object-Relational Database Management System (ORDBMS)**, derived from the **POSTGRES (v4.2)** university project developed at the University of California, Berkeley.

Main characteristics:

- Open-source heir to the original code of the Berkeley academic project.
- Compliant with a large part of the **SQL** standard (ISO/IEC 9075).
- Offers advanced features:
  - Complex queries, foreign keys, triggers.
  - Updatable views and full transactional integrity.
  - Multiversion concurrency control (**MVCC**).

Extensibility: PostgreSQL makes it possible to add new **data types**, **functions**, **operators**, **index methods**, as well as custom **procedural languages** (PL/pgSQL, PL/Python, etc.).

PostgreSQL license

Distributed under a **free and permissive** license, PostgreSQL can be used, modified and redistributed without restriction, for *private*, *academic* or *commercial* purposes.

A brief history of PostgreSQL

PostgreSQL stems from the **POSTGRES** project, launched in 1986 at the University of California, Berkeley, under the direction of Professor **Michael Stonebraker**.

Historical evolution:

- **POSTGRES (1986–1993)** — a visionary research project introducing **active rules** (ancestors of *triggers*), **advanced transactions** and **object-oriented storage**. Used in *science*, *medicine* and *GIS*.
- **Postgres95 (1994)** — addition of the **SQL** language, creation of the **psql** client and simplification of the source code.
- **PostgreSQL (1996 $\rightarrow$ ...)** — adoption of the current name, reflecting full SQL compatibility. **Open, community-driven** development, continuously enriched.

Conclusion

Today, **PostgreSQL** stands out as the **reference open-source DBMS**, favored in both academia and industry.

PostgreSQL client/server architecture

PostgreSQL follows a **client/server** model: the client sends SQL queries, the server interprets them, performs the requested operations and returns the results.

Client side: Clients can be interactive tools (`psql`, **pgAdmin**), web applications or scripts. They communicate with the server over **TCP/IP** using a default port (5432).

Server side: The main `postgres` process listens for incoming connections. When a client connects, it creates a **dedicated child process** that exclusively handles that session. This process receives the query and passes it to the **parser**, which translates it into a syntax tree, then to the **planner/optimizer**, which determines the most efficient execution strategy. Finally, the **executor** applies this plan and returns the results to the client.

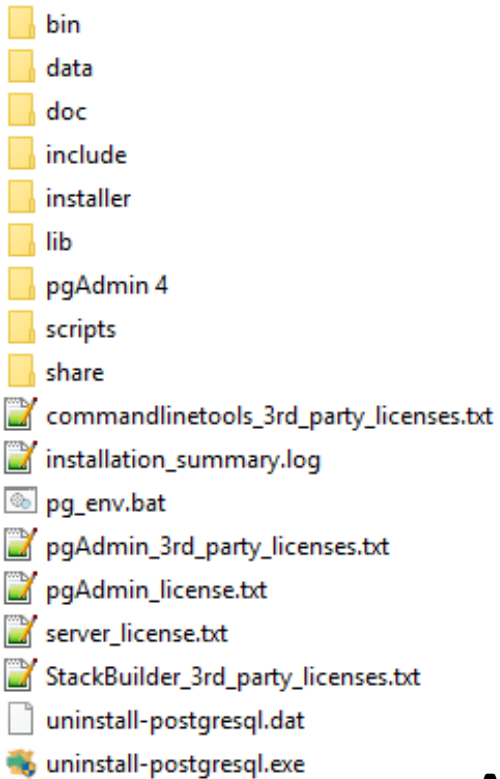
Advantage: This architecture allows **concurrent**, stable and secure operation, where each session is isolated, but all share the same data files managed by the main server.

Installation procedure (PostgreSQL version 16)

- **Download**: Official website → https://www.postgresql.org/download/windows (executable: `postgresql-16.x-windows-x64.exe`).
- **Running the installer**: run it as administrator (`right-click → Run as administrator`).
- **Installation paths** (to remember): `C:\Program Files\PostgreSQL\16\` (useful to find the binaries and configuration files).
- **Components to install**:
  - PostgreSQL Server
  - pgAdmin 4
  - Command Line Tools (`psql`)
  - Stack Builder (optional)
- **Initial configuration**:
  - Superuser password: `postgres`.
  - Default port: `5432`.
  - Windows service created: `postgresql-x64-16`.

Installation contents

The installation contents look like this:



- **bin/**: main executables.
- **data/**: data directory (PostgreSQL cluster).
- **doc/**: local documentation.
- **include/**: header files for development.
- **lib/**: dynamic libraries (DLL).
- **pgAdmin 4/**: graphical administration interface.
- **scripts/**: utility scripts.
- **share/**: shared files (init, dictionaries, templates).
- **StackBuilder/**: installer for additional extensions.

Associated files: `installation_summary.log`, `pg_env.bat`, `server_license.txt`, `uninstall-postgresql.exe`, etc.

The `bin` folder — Main executables

The **bin/** directory contains the essential programs used to **start, administer and interact** with the PostgreSQL server.

- **postgres.exe** — starts the main server.
- **psql.exe** — command-line client.
- **pg_ctl.exe** — starts, stops or restarts the server.
- **initdb.exe** — initializes a database cluster.
- **pg_dump(.exe)** / **pg_dumpall(.exe)** — export and backup.
- **pg_restore.exe** — restore from a backup.
- **createdb(.exe)** / **dropdb(.exe)** — create and drop databases.
- **createuser(.exe)** / **dropuser(.exe)** — manage user roles.
- **vacuumdb.exe** — clean up and optimize tables.
- **pgAdmin4.exe** — graphical administration interface.

These executables form the basis of all PostgreSQL administration on Windows.

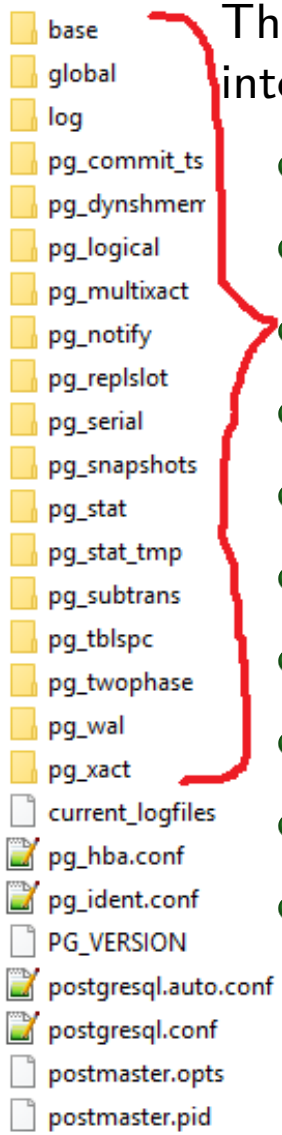
SQL Shell — `psql`

The **psql.exe** program is the official command-line client.

- **Functions**:
  - Connecting to the PostgreSQL server.
  - Running SQL commands (`SELECT`, `INSERT`, `UPDATE`, etc.).
  - Internal commands (`\d`, `\l`, `\c`, ...).
  - Running SQL scripts (`\i file.sql`).
- **Advantages**: complete, fast, powerful for administration and testing.
- **Access**: from the Start menu (SQL Shell) or via a terminal: `psql -U postgres -d db_name`.

⇒ The main tool for learning and practicing SQL with PostgreSQL.

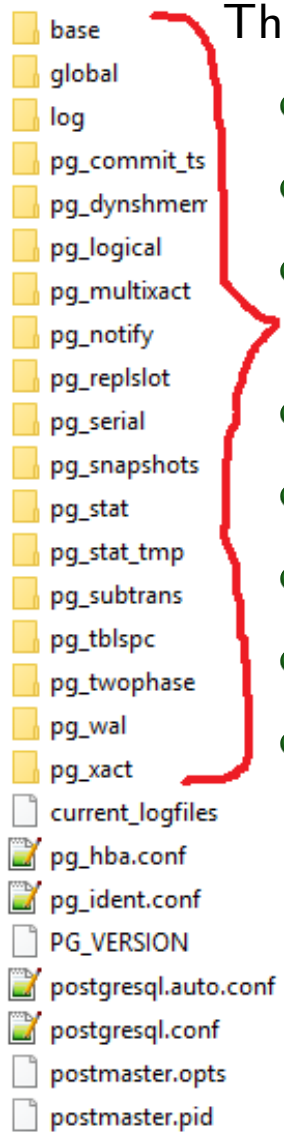
Cluster — data/



The **data/** directory contains the **whole PostgreSQL cluster**, organized into essential subfolders:

- **base/**: stores the binary data of user databases.
- **global/**: contains the cluster-wide roles and catalogs.
- **pg_wal/**: transaction logs (Write Ahead Log).
- **pg_xact/**: transaction states (commit/rollback).
- **pg_multixact/**: management of shared locks.
- **pg_stat/**, **pg_stat_tmp/**: server statistics.
- **pg_logical/**: logical replication support.
- **pg_subtrans/**: tracking of nested subtransactions.
- **pg_twophase/**: two-phase transactions (2PC).
- **pg_tblspc/**: links to external tablespaces.

Cluster - configuration files



The essential files of the **data/** directory are:

- **postgresql.conf**: main configuration (port, memory, logs, etc.).
- **pg_hba.conf**: access-control rules and authentication methods.
- **pg_ident.conf**: mapping between system users and PostgreSQL users.
- **postgresql.auto.conf**: parameters changed via `ALTER SYSTEM`.
- **PG_VERSION**: PostgreSQL version in use.
- **postmaster.pid**: PID of the running server process.
- **postmaster.opts**: launch options used at startup.
- **current_logfiles**: location of the active logs.

Installation objects

During installation, PostgreSQL automatically creates several essential elements:

- **The postgres database**: default database to connect to immediately.
- **The postgres role**: default role with all privileges.
- **The template databases**: serve as templates for creating new databases.

Schema in PostgreSQL

A **schema** is a logical storage space inside a database. It contains objects such as tables, views, functions, sequences, etc. By default, every database contains a **public** schema.

Getting started with PostgreSQL:

- Connect with the `postgres` role.
- Work in the **postgres** database.
- Create new databases from **template0** or **template1**.
- Organize objects into **schemas**.

The postgres database

During installation, **PostgreSQL** automatically creates a database named **postgres**. It is a special database serving as the **initial working environment**.

Main uses:

- Default connection database (if no other is specified).
- Test and experimentation database for running simple commands.
- Administration database for managing roles and databases from the very first connection.

Good practices:

- Never drop it: it is a **fallback database** that always guarantees a connection to the server.
- Use it for quick tests, but create dedicated databases for application projects.

Key takeaway

The postgres database is a **utility and administration database**, which must be kept even when other databases are created.

The postgres role

During installation, **PostgreSQL** automatically creates a special role named **postgres**. This role is the system's **superuser**.

Main characteristics:

- Has all privileges: creating databases, managing roles, configuring the server.
- Used for the initial configuration and the first connections.
- Can access the server via `psql`, **pgAdmin** or any other compatible SQL client.

Good practices:

- Avoid using `postgres` directly for application databases.
- Create dedicated roles for each project with limited, appropriate privileges.
- Reserve `postgres` exclusively for administration and maintenance tasks.

Key takeaway

The `postgres` role is the **default superuser**. It must be used only for administration and never for everyday development.

The template0 and template1 template databases

During installation, **PostgreSQL** creates two template databases that serve as a reference for any new database. Every database created is a **copy** of one of these templates.

template0:

- Minimal database, pristine and never modified.
- Used to create completely "clean" databases, without customization.

template1:

- Customizable template database.
- Extensions, functions or languages added here are copied into new databases.

Choosing the template at creation time:

```
CREATE DATABASE db_name TEMPLATE template0;
```

```
CREATE DATABASE db_name; -- uses template1 by default
```

Key takeaway

Every new PostgreSQL database is a copy of one of these two templates.

Roles in PostgreSQL

In **PostgreSQL**, there is no distinction between **user** and **group**: everything is represented by a **role** (`ROLE`).

Fundamental principle

A role can:

- connect to the server (like a user);
- own objects: databases, tables, views, functions, etc.;
- receive privileges or grant them to other roles;
- group other roles (group function).

Note

The `postgres` role is the **superuser** created automatically during installation. It has all privileges and can administer the whole server.

Login roles (LOGIN)

A **login role** is a role allowed to connect to the **PostgreSQL** server. It corresponds to a **regular user** of the system.

```
-- Creating a login role (with a password)
CREATE ROLE seller LOGIN PASSWORD 'seller123';

-- Equivalent command: CREATE USER is an alias for
--   CREATE ROLE ... LOGIN
CREATE USER seller PASSWORD 'seller123';
```

- Can open a session via `psql`, `pgAdmin`, etc.
- Has its own privileges and objects.
- Generally used for application connections.

Example

The `seller` role, used in the `pizzerias` database, is a login role.

Roles without login (NOLOGIN)

A **role without login** does not have the LOGIN attribute. It is used to **group several users** in order to manage their privileges collectively.

```
-- Creating a "group" role
CREATE ROLE employees;

-- Granting the "employees" role to several users
GRANT employees TO seller, waiter, manager;
```

- Cannot connect directly to the server.
- Simplifies privilege management (inheritance principle).
- All members inherit the privileges granted to the group.

Example

If the employees role has read access on a table, then seller and waiter automatically inherit it.

[1/4] Schemas — concept and purpose

Definition

A **schema** is a **logical namespace** inside a database. It groups objects — **tables**, **views**, **functions**, **sequences** — to structure and isolate the components of a single project.

A schema makes it possible to:

- **Organize** the database by functional domain (for example: *application* (`app`), *reference data* (`ref`), *audit* (`audit`), *log* (`log`), etc.).
- **Separate** privileges and responsibilities according to roles or users.
- **Simplify** management and avoid object-name conflicts.

Key idea

A schema acts as a **logical folder** inside a single database: it brings **clarity**, **structure** and **security**, without requiring several separate databases.

[2/4] The default schema — `public`

When a database is created, PostgreSQL automatically adds a schema named **public** to it. By default, all users are allowed to create objects in it.

Example

```
-- Implicitly creates the table in the "public" schema
CREATE TABLE clients (id SERIAL PRIMARY KEY, name TEXT);

-- Explicit equivalent
CREATE TABLE public.clients (id SERIAL PRIMARY KEY, name TEXT);
```

Note

If no schema is specified, PostgreSQL uses **public**. For a structured project, it is advisable to create your own application schemas.

[3/4] Creation and configuration

A **schema** always has an **owner** (role) defined when it is created through the `AUTHORIZATION` clause, which grants it all rights on its objects.
The **search_path** specifies the order in which PostgreSQL searches schemas when an object name is not qualified (`schema.table`). The first schema in the path is used by default.

Creating a schema

```
-- Creation and definition of the owner
CREATE SCHEMA IF NOT EXISTS app AUTHORIZATION app_owner;
```

Search path

```
-- Search order: first app, then public
ALTER ROLE seller SET search_path TO app, public;

-- Display the current path
SHOW search_path;
```

[4/4] Usage and privileges

Essential privileges

```
-- Allow use of the schema and creation of objects
GRANT USAGE, CREATE ON SCHEMA app TO employees;

-- Privileges on existing objects (tables)
GRANT SELECT, INSERT, UPDATE, DELETE
  ON ALL TABLES IN SCHEMA app TO employees;

-- (Optional) Privileges on existing sequences
GRANT USAGE ON ALL SEQUENCES IN SCHEMA app TO employees;
```

Good practices

- Always qualify objects: `schema.table`.
- Restrict privileges on the `public` schema.
- Create one schema per application domain.
- (Optional) Propagate privileges to future objects: `ALTER DEFAULT PRIVILEGES`.

Working on a database: owner or privileges

To work on a database, there are two situations:

- **Owner role** — it has all rights on the database.
- **Non-owner role** — it must be granted the necessary privileges.

Example: granting privileges on a database

```
-- Grant all privileges on the "pizzerias" database to the "seller" role
GRANT ALL PRIVILEGES ON DATABASE pizzerias TO seller;
```

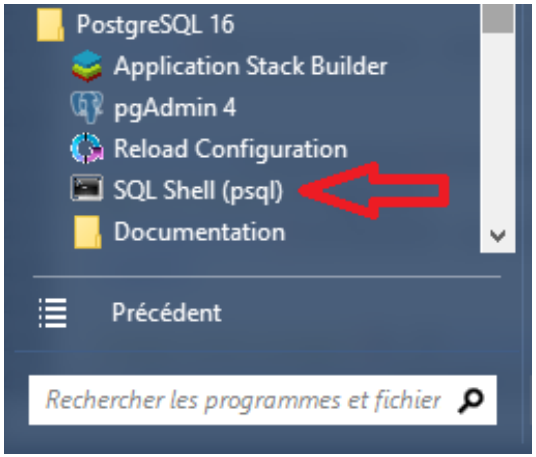
Privilege levels

- *Schema* — USAGE, CREATE (object creation).
- *Tables* — SELECT, INSERT, UPDATE, DELETE.
- *Sequences* — USAGE, SELECT, UPDATE (for SERIAL/IDENTITY).
- *Functions / procedures* — EXECUTE.
- *Default privileges* — `ALTER DEFAULT PRIVILEGES`.
- *Group roles* — `GRANT group TO user`.

Connecting to the PostgreSQL server

To connect to PostgreSQL:

- **Method 1:** launch the **SQL Shell (psql)** from its icon.



- **Method 2:** open a terminal and run the following command:

```
C:\PostgreSQL\16\bin\psql.exe -h localhost -U postgres -d postgres -p 5432
```

If the **psql** path is in the **PATH**:

```
C:\> psql -h localhost -U postgres -d postgres -p 5432
```

Internal psql commands

Internal psql commands make it possible to administer the server without going through the SQL language. They always start with a backslash \.

Main commands

```
\l              -- List databases
\c dbname       -- Connect to a database
\dt             -- List tables
\d table        -- Describe a table
\du             -- List roles and users
\dn             -- List schemas
\conninfo       -- Information about the active connection
\?              -- Help on psql commands
\h command      -- Help on an SQL command (e.g. \h SELECT)
\i file         -- Run an SQL script
\o file         -- Redirect output to a file
\timing         -- Toggle timing measurement
\password       -- Change a role's password

\q              -- Quit psql
```

Teaching sample database

Goal: have a simple database used as an example to illustrate creation, privilege management and SQL queries in PostgreSQL.

Database used:

- `pizzerias` — sample database for getting started: pizza sales (simple tables, foreign keys, cascades).

Resources

SQL scripts, lab PDFs and course materials available at:
www.al-moualime.com/bdda