TP N° 1 — Initiation à PostgreSQL

(Sous Windows)

M. Laïdi FOUGHALI

l.foughali@univ-skikda.dz

(Support ⇒ al-moualime.com)

Université de Skikda — Département Informatique

1re Année Master RSD/IA

Bases de Données Avancées (BDA)

Octobre 2025

Version 1 — 2025-10-13 à 23:26:47

Plan

Qu'est-ce que PostgreSQL ?

PostgreSQL est un **Système de Gestion de Bases de Données Relationnelles Objet (SGBDRO)**, issu du projet **POSTGRES (v4.2)** développé à l'Université de Californie à Berkeley.

Caractéristiques principales :

- Héritier libre du code original du projet universitaire de Berkeley.
- Conforme à une large partie du standard **SQL**.
- Offre des fonctionnalités avancées :
  - Requêtes complexes, clés étrangères, déclencheurs (triggers).
  - Vues modifiables et intégrité transactionnelle complète.
  - Contrôle de concurrence multiversion (**MVCC**).

Extensibilité : PostgreSQL permet l'ajout de nouveaux **types de données**, **fonctions**, **opérateurs**, **méthodes d'indexation** et même de **langages de procédure**.

Licence PostgreSQL

Sous licence **libre et permissive**, PostgreSQL peut être utilisé, modifié et distribué librement pour tout usage : *privé, académique ou commercial*.

Bref historique de PostgreSQL

PostgreSQL trouve son origine dans le projet **POSTGRES**, lancé en 1986 à l'Université de Californie (Berkeley) par le professeur **Michael Stonebraker**.

Étapes clés de son évolution :

- **POSTGRES (1986–1993)** : projet universitaire innovant introduisant les notions de règles, de transactions et de stockage orienté objet. Utilisé dans des domaines variés (sciences, médecine, systèmes d'information géographiques).
- **Postgres95 (1994)** : ajout du langage **SQL**, création du client **psql**, simplification du code et intégration de nouvelles fonctionnalités (agrégats, clause `GROUP BY`, etc.).
- **PostgreSQL (1996 → ...)** : adoption du nom actuel pour marquer l'intégration complète de SQL. Le développement devient communautaire et ouvert, enrichi en continu de fonctionnalités modernes.

Conclusion

Aujourd'hui, **PostgreSQL** est reconnu comme la **base de données libre de référence**, adoptée à la fois dans le milieu académique et dans le monde industriel.

Architecture client/serveur de PostgreSQL

PostgreSQL adopte une architecture **client/serveur** clairement séparée.

- **Côté client** :
  - Composants qui envoient des requêtes SQL et reçoivent les résultats.
  - Exemples : `psql`, **pgAdmin**, applications web, scripts et outils tiers.
- **Côté serveur** :
  - Processus principal `postgres` : écoute les connexions entrantes.
  - À chaque nouvelle connexion, il crée un **processus dédié**.
  - Assure la gestion des fichiers, transactions, verrous et contrôles d'accès.

Communication : Les échanges entre client et serveur se font via **TCP/IP**. Le client interagit avec un serveur unique, mais celui-ci gère en parallèle plusieurs connexions simultanées.

PostgreSQL repose sur une séparation claire des rôles :

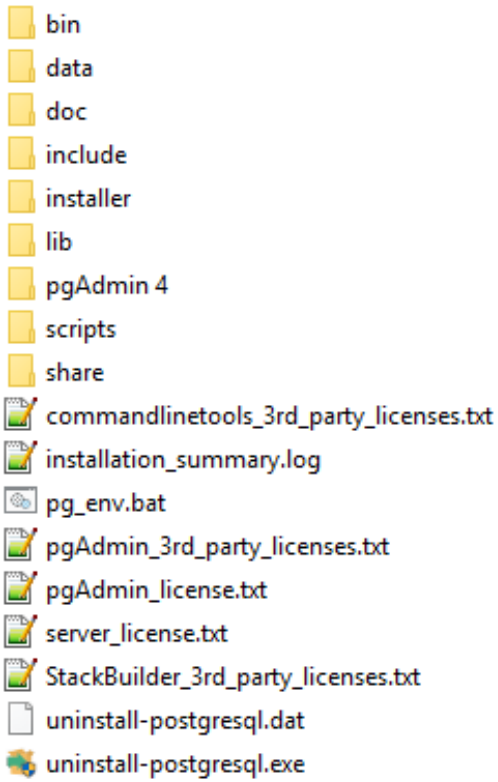
- **Clients** : formulent les requêtes SQL (via `psql`, **pgAdmin**, scripts, applications...).
- **Serveur** : exécute les requêtes, gère les transactions et renvoie les résultats.

Procédure d'installation (PostgreSQL version 16)

- **Téléchargement** : Site officiel → https://www.postgresql.org/download/windows (exécutable : `postgresql-16.x-windows-x64.exe`).
- **Exécution de l'installeur** : lancer en mode administrateur (`clic droit → Exécuter en tant qu'administrateur`).
- **Chemins d'installation** (à mémoriser) : `C:\Program Files\PostgreSQL\16\` (utile pour retrouver les fichiers binaires et de configuration).
- **Composants à installer** :
  - PostgreSQL Server
  - pgAdmin 4
  - Command Line Tools (`psql`)
  - Stack Builder (optionnel)
- **Configuration initiale** :
  - Mot de passe superutilisateur : `postgres`.
  - Port par défaut : 5432.
  - Service Windows créé : `postgresql-x64-16`.

Contenu de l'installation

Le contenu de l'installation se présente ainsi :



- **bin/** : exécutables principaux.
- **data/** : répertoire des données (cluster PostgreSQL).
- **doc/** : documentation locale.
- **include/** : fichiers d'en-tête pour le développement.
- **lib/** : bibliothèques dynamiques (DLL).
- **pgAdmin 4/** : interface graphique d'administration.
- **scripts/** : scripts utilitaires.
- **share/** : fichiers partagés (init, dictionnaires, modèles).
- **StackBuilder/** : installateur d'extensions complémentaires.

Fichiers associés : `installation_summary.log`, `pg_env.bat`, `server_license.txt`, `uninstall-postgresql.exe`, etc.

Dossier `bin` — Exécutables principaux

Le dossier **bin** contient les exécutables essentiels :

- **postgres.exe** : lance le serveur (processus principal).
- **psql.exe** : client en ligne de commande.
- **pg_ctl.exe** : démarre, arrête ou redémarre le serveur.
- **initdb.exe** : initialise un cluster de bases.
- **pg_dump.exe** / **pg_dumpall.exe** : sauvegarde.
- **pg_restore.exe** : restauration depuis une sauvegarde.
- **createdb.exe** / **dropdb.exe** : création ou suppression de bases.
- **createuser.exe** / **dropuser.exe** : gestion des rôles.
- **vacuumdb.exe** : nettoyage et optimisation.
- **pgAdmin4.exe** : interface graphique locale.

⇒ Ces outils sont indispensables pour **administrer, sauvegarder et interroger** PostgreSQL.

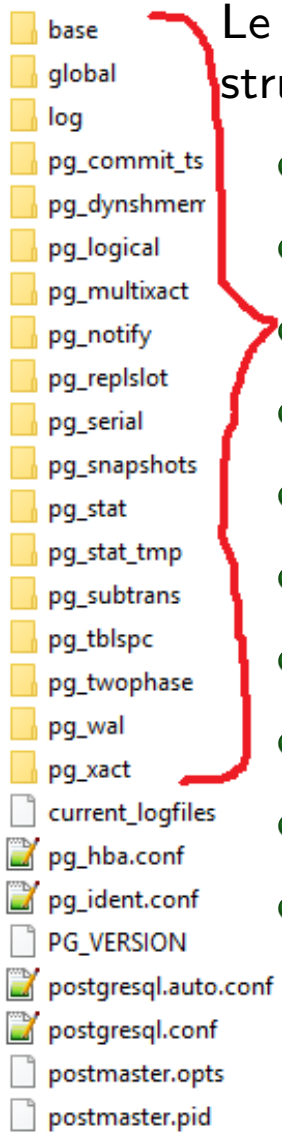
SQL Shell — `psql`

Le programme **psql.exe** est le client en ligne de commande officiel.

- **Fonctions** :
  - Connexion au serveur PostgreSQL.
  - Exécution de commandes SQL (`SELECT`, `INSERT`, `UPDATE`, etc.).
  - Commandes internes (`\d`, `\l`, `\c`, ...).
  - Exécution de scripts SQL (`\i fichier.sql`).
- **Avantages** : complet, rapide, puissant pour l'administration et les tests.
- **Accès** : depuis le menu Démarrer (SQL Shell) ou via terminal : `psql -U postgres -d nom_base`.

⇒ Outil principal pour apprendre et pratiquer le SQL sous PostgreSQL.

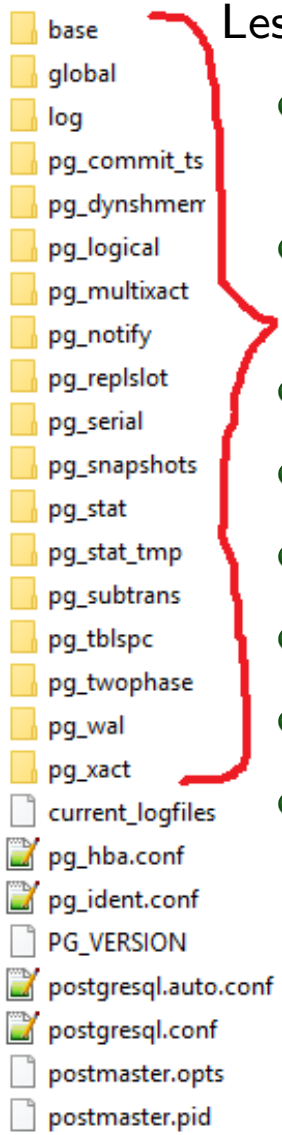
Cluster — data/



Le répertoire **data/** contient l'**ensemble du cluster PostgreSQL**, structuré en sous-dossiers essentiels :

- **base/** : stocke les données binaires des bases utilisateur.
- **global/** : contient les rôles et catalogues globaux du cluster.
- **pg_wal/** : journaux de transactions (Write Ahead Log).
- **pg_xact/** : états des transactions (commit/rollback).
- **pg_multixact/** : gestion des verrous partagés.
- **pg_stat/**, **pg_stat_tmp/** : statistiques du serveur.
- **pg_logical/** : support de la réplication logique.
- **pg_subtrans/** : suivi des sous-transactions imbriquées.
- **pg_twophase/** : transactions en deux phases (2PC).
- **pg_tblspc/** : liens vers des tablespaces externes.

Cluster - fichiers de configuration



Les fichiers essentiels du répertoire **data/** sont :

- **postgresql.conf** : configuration principale (port, mémoire, logs, etc.).
- **pg_hba.conf** : règles de contrôle des accès et méthodes d'authentification.
- **pg_ident.conf** : mappage utilisateurs système et PostgreSQL.
- **postgresql.auto.conf** : paramètres modifiés via `ALTER SYSTEM`.
- **PG_VERSION** : version de PostgreSQL utilisée.
- **postmaster.pid** : PID du processus serveur en cours d'exécution.
- **postmaster.opts** : options de lancement utilisées au démarrage.
- **current_logfiles** : emplacement des journaux actifs.

Objets de l'installation

Lors de l'installation, PostgreSQL crée automatiquement plusieurs éléments essentiels :

- **La base postgres** : base par défaut pour se connecter immédiatement.
- **Le rôle postgres** : rôle par défaut ayant tous les privilèges.
- **Les bases modèles (templates)** : servent de modèle pour créer de nouvelles bases.

Schéma sous PostgreSQL

Un **schéma** est un espace logique de rangement à l'intérieur d'une base de données. Il contient des objets comme des tables, vues, fonctions, séquences, etc. Par défaut, chaque base contient un schéma **public**.

Pour commencer avec PostgreSQL :

- Se connecter avec le rôle `postgres`.
- Travailler dans la base **postgres**.
- Créer de nouvelles bases à partir de **template0** ou **template1**.
- Organiser les objets dans des **schémas**.

La base postgres

Lors de l'installation, PostgreSQL crée automatiquement une base appelée **postgres**. C'est une base spéciale, fournie comme environnement de travail initial.

Utilités principales :

- Base de connexion par défaut (si aucune autre n'est précisée).
- Base de test et d'expérimentation pour exécuter des commandes simples.
- Base d'administration pour gérer rôles et bases dès la première connexion.

Bonnes pratiques :

- Ne jamais supprimer cette base : elle sert de **sécurité**.
- L'utiliser pour des tests rapides, mais créer ses propres bases pour les projets.

À retenir

La base `postgres` est une **base utilitaire et d'administration**, à conserver même si l'on travaille sur d'autres bases.

Le rôle postgres

PostgreSQL crée automatiquement un rôle spécial : le rôle **postgres**. Il s'agit du **superutilisateur** du système.

Caractéristiques principales :

- Possède tous les privilèges : création de bases, gestion des rôles, configuration.
- Utilisé pour la configuration initiale et les premières connexions.
- Peut se connecter via `psql`, **pgAdmin** ou d'autres clients SQL.

Bonnes pratiques :

- Ne pas utiliser directement `postgres` pour les bases applicatives.
- Créer des rôles spécifiques selon les projets et leur attribuer les droits nécessaires.
- Réserver `postgres` aux tâches d'administration et de maintenance.

À retenir

Le rôle `postgres` est le **superutilisateur par défaut**. Il doit être réservé à l'administration et non à l'usage quotidien.

Les bases modèles `template0` et `template1`

PostgreSQL crée les base à partir d'une **base modèle**. Ces modèles assurent que chaque base possède la structure minimale nécessaire au SGBD.

- **template0** :
  - Base minimale, intacte (jamais modifiée).
  - Sert uniquement à créer des bases complètement « propres ».
  - Utilisée lorsqu'on veut éviter toute personnalisation héritée.
- **template1** :
  - Base modèle personnalisable.
  - Les extensions, langues, fonctions ou objets ajoutés ici seront recopiés dans toutes les nouvelles bases.
  - Sert de modèle pratique pour appliquer automatiquement une configuration commune.

À retenir

- **template0** = modèle vierge, utilisé pour créer sans personnalisation.
- **template1** = modèle enrichi, utilisé pour propager vos personnalisations.
- **Toute nouvelle base est une copie de l'un de ces deux modèles.**

Les rôles dans PostgreSQL

Dans PostgreSQL, il n'existe pas de distinction stricte entre **utilisateur** et **groupe** : tout est un **rôle** (`ROLE`).

Principe fondamental

Un rôle peut :

- se connecter (comme un utilisateur) ;
- posséder des objets (bases, tables, vues...) ;
- recevoir ou attribuer des privilèges ;
- regrouper d'autres rôles (comme un groupe).

Remarque

Le rôle **postgres** est un **superutilisateur** créé automatiquement lors de l'installation du SGBD. Il possède tous les privilèges et peut administrer l'ensemble du serveur.

Les rôles de connexion (LOGIN)

Un **rôle de connexion** est un rôle capable de se connecter au serveur PostgreSQL. Il correspond à un **utilisateur classique**.

```
-- Création d'un rôle de connexion
CREATE ROLE vendeur LOGIN PASSWORD 'vendeur123';

-- Équivalent (plus explicite)
CREATE USER vendeur PASSWORD 'vendeur123';
```

- Peut ouvrir une session via `psql`, `pgAdmin`, etc.
- Possède ses propres privilèges et objets.
- Est utilisé pour les connexions applicatives.

Exemple

Le rôle `vendeur` que nous créons pour la base `pizzerias` est un rôle de connexion.

Les rôles sans connexion (groupes)

Un **rôle sans connexion** n'a pas l'attribut `LOGIN`. Il sert à **regrouper plusieurs utilisateurs** pour gérer les privilèges collectivement.

```
-- Creation d'un role "groupe"
CREATE ROLE employes;

-- Attribution du role "employes" a plusieurs utilisateurs
GRANT employes TO vendeur, serveur, gerant;
```

- Ne peut pas se connecter directement.
- Sert a simplifier la gestion des droits (heritage).
- Tous les membres heritent des privileges accordes au groupe.

Exemple

Si `employes` a le droit de lecture sur une table, alors `vendeur` et `serveur` l'auront aussi automatiquement.

[1/3] Schémas — Notion et utilité

Définition

Un **schéma** est un **espace de noms logique** à l'intérieur d'une base de données. Il regroupe les objets — **tables**, **vues**, **fonctions**, **séquences** — pour structurer et isoler les composants d'un même projet.

Un schéma permet de :

- **Organiser** la base par domaine fonctionnel (`app`, `ref`, `audit`, etc.).
- **Séparer** les droits et responsabilités selon les rôles ou utilisateurs.
- **Simplifier** la gestion et éviter les conflits de noms d'objets.

Idée clé

Un schéma agit comme un **dossier logique** à l'intérieur d'une même base : il apporte de la **clarté**, de la **structure** et de la **sécurité**, sans nécessiter la création de plusieurs bases distinctes.

[2/3] Création et configuration

Création d'un schéma

```
-- Créer un schéma et en définir le propriétaire
CREATE SCHEMA IF NOT EXISTS app AUTHORIZATION app_owner;
```

Configurer le chemin de recherche

```
-- Résolution des noms sans préfixe : d'abord app, puis public
ALTER ROLE vendeur SET search_path TO app, public;

-- Vérifier le chemin courant
SHOW search_path;
```

Rappel

Le `search_path` détermine l'ordre de recherche des schémas. En cas de doublon de nom, le premier trouvé est utilisé.

[3/3] Utilisation et droits

Droits essentiels

```
-- Autoriser l'utilisation du schéma et la création d'objets
GRANT USAGE, CREATE ON SCHEMA app TO employes;

-- Droits sur les objets existants
GRANT SELECT, INSERT, UPDATE, DELETE
  ON ALL TABLES IN SCHEMA app TO employes;
```

Bonnes pratiques

- Toujours qualifier les objets : `schema.table`.
- Restreindre les accès au schéma `public`.
- Créer un schéma par domaine applicatif.

Travailler sur une base : propriétaire ou privilèges

Pour travailler sur une base, deux approches sont possibles :

- **Être propriétaire** — le rôle détient tous les droits sur la base.
- **Ne pas être propriétaire** — le rôle doit disposer des privilèges nécessaires.

Exemple : accorder les privilèges sur une base

```
-- Affecter les privilèges sur la base "pizzerias" au rôle "vendeur"
GRANT ALL PRIVILEGES ON DATABASE pizzerias TO vendeur;
```

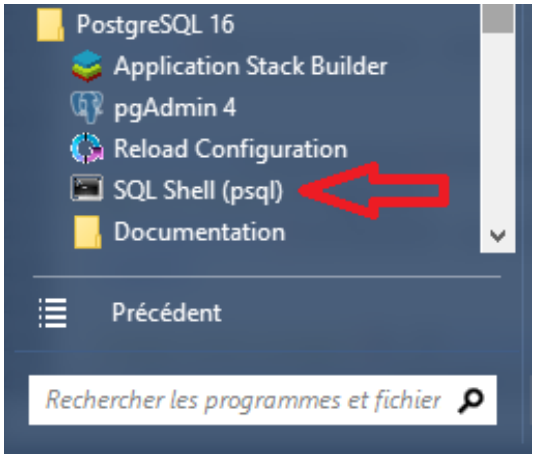
Quelques niveaux de privilèges

- *Schéma* — USAGE (+ CREATE pour créer des objets).
- *Tables* — SELECT, INSERT, UPDATE, DELETE.
- *Séquences* — USAGE, SELECT, UPDATE (pour SERIAL/IDENTITY).
- *Fonctions / Procédures* — EXECUTE.
- *Privilèges par défaut* — ALTER DEFAULT PRIVILEGES.
- *Rôles-groupe* — `GRANT groupe TO utilisateur`.

Connexion au serveur PostgreSQL

Pour se connecter à PostgreSQL :

- **Méthode 1 :** lancer le **SQL Shell (psql)** depuis son icône.



- **Méthode 2 :** ouvrir un terminal et exécuter la commande suivante :

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

Si le chemin de **psql** est dans le **PATH** :

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

Commandes internes de psql

Les commandes internes de psql permettent d'administrer le serveur sans passer par le langage SQL. Elles commencent toujours par une barre oblique inversée \.

Commandes principales

```
\l              -- Lister les bases de données
\c dbname       -- Se connecter à une base
\dt             -- Lister les tables
\d table        -- Décrire une table
\du             -- Lister les rôles et utilisateurs
\dn             -- Lister les schémas
\conninfo       -- Informations sur la connexion active
\?              -- Aide sur les commandes psql
\h commande     -- Aide sur une commande SQL (ex: \h SELECT)
\i fichier      -- Exécuter un script SQL
\o fichier      -- Rediriger la sortie vers un fichier
\timing         -- Activer/désactiver la mesure du temps
\password       -- Modifier le mot de passe d'un rôle

\q              -- Quitter psql
```

Base exemple pédagogique

But : disposer d'une base simple prise comme exemple pour illustrer la création, la gestion des droits et les requêtes SQL dans PostgreSQL.

Base utilisée :

- `pizzerias` — base exemple pour l'initiation : ventes de pizzas (tables simples, clés étrangères, cascades).

Ressources

Scripts SQL, PDF des TPs et supports disponibles sur : `bda.al-moualime.com`