



# PostgreSQL Administration

---

PGRE-SQL – Rev 9

[m2iformation.fr](http://m2iformation.fr)





# SOMMAIRE

---

I. PRÉSENTATION GÉNÉRALE	P.9
1. Préface	p.11
2. Fonctionnalités	p.15
3. Documentation	p.17
4. Déclaration d'un bug	p.21
5. Les versions de PostgreSQL	p.23
II. INSTALLATION	P.25
1. Pré-installation avec les sources	p.27
2. Installation avec les sources	p.33
3. Post-installation	p.41
4. Autres méthodes d'installation	p.47
III. MISE EN ŒUVRE D'UN SERVEUR DE DONNÉES POSTGRESQL	P.59
1. Initialisation d'un serveur PostgreSQL	p.61
2. Démarrage et arrêt de PostgreSQL	p.65
3. Configuration du serveur	p.73
4. Démarrage et arrêt automatique du serveur	p.85
IV. CRÉATION D'UNE BASE	P.91
1. Introduction	p.93
2. Création d'une base de données	p.95
3. Gestion des tablespaces	p.105
V. AUTHENTIFICATION DES CLIENTS	P.111
1. Authentification avec le fichier pg_hba.conf.	p.113
VI. GESTION DE LA SÉCURITÉ	P.123
1. Les rôles et les priviléges	p.125
2. Gestion des rôles de type utilisateur ou groupe	p.127
3. Gestion des priviléges	p.137

# SOMMAIRE

---

<b>VII. MAINTENANCE D'UN SERVEUR POSTGRESQL</b>	<b>p.145</b>
1. Opération de maintenance sur un serveur PostgreSQL	p.147
2. Gestion de l'espace disque	p.149
3. Nettoyage d'une base de données	p.159
4. Utilisation manuelle de VACCUM	p.163
5. Le fichier de log	p.169
6. pgBadger	p.173
<b>VIII. PSQL ET PGADMIN 4</b>	<b>p.175</b>
1. psql	p.177
2. pgAdmin 4	p.189
<b>IX. TABLES ET INDEX</b>	<b>p.195</b>
1. Les tables	p.197
2. Les index	p.207
<b>X. SAUVEGARDES ET RESTAURATIONS</b>	<b>p.213</b>
1. Généralités	p.215
2. Sauvegardes logiques	p.217
3. Restaurations logiques	p.233
4. Sauvegardes physiques	p.239

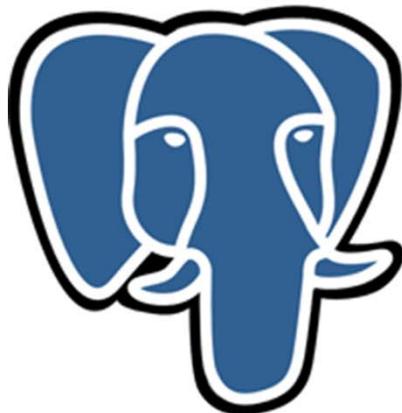
# SOMMAIRE

---

XI. OPTIMISATION	p.255
1. Introduction	p.257
2. Lecture d'un plan d'exécution	p.259
3. Paramètres d'optimisation	p.263
4. Contrôle de l'activité et statistique du serveur	p.267
5. Contrib pg_stat_statements	p.273
XII. ANNEXES	p.275
Sommaire	p.277



# BASES DE DONNEES



Postgre<sup>SQL</sup>

## PostgreSQL Administration

"Ce produit a été fabriqué selon une organisation qualité conforme à la norme  
**ISO 9001:2008 certifiée par l' AFAQ N° 1993/1747 d".**

«La loi du 11 mars 1957 complétée par la loi du 3 juillet 1985 interdit les copies ou reproduction destinées à une utilisation collective. Toute représentation ou reproduction intégrale ou partielle faite par quelque procédé que ce soit, sans le consentement de l'auteur ou de ses ayants causes, est illicite et constitue une contrefaçon sanctionnée par les articles 425 et suivants du code pénal».



# 1. PRÉSENTATION GÉNÉRALE

---

## Objectif

A la fin de ce module, vous aurez une vue d'ensemble du SGBDR PostgreSQL, de ses principales fonctionnalités et les sources d'informations disponibles pour utiliser correctement un serveur de bases de données PostgreSQL :

- Préface (communauté PostgreSQL, site Internet...)
- Fonctionnalités
- Documentation
- Les projets annexes
- Déclaration d'un bug
- Les versions mineures et majeures

# Présentation générale

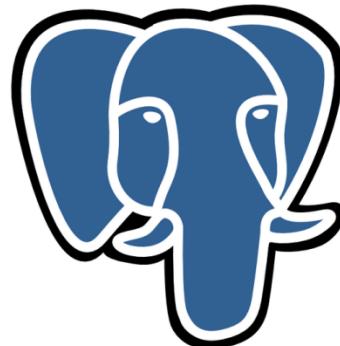
---



# Présentation générale

---

## 1. Préface



PostgreSQL

**PostgreSQL** est un Système de Gestion de Bases de Données Relationnelles (**SGBDR**) initialement créé à travers le projet nommé Ingres développé à l'université de Berkeley à partir de 1977. En 1986, une autre équipe poursuit le développement de Ingres et initie le projet de création d'un SGBDR PostgreSQL qui donnera naissance en **1996**, grâce à la communauté de développement de logiciels libres, au SGBDR **PostgreSQL**.

**PostgreSQL** est actuellement le SGBDR Open Source le plus aboutit. Il est robuste, fiable, portable sur de nombreux OS et possède un grand nombre de fonctionnalités que l'on ne trouve généralement que dans les SGBDR commerciaux.

**PostgreSQL** est un produit Open Source dont les sources sont librement récupérables, modifiables et utilisables. Ce logiciel Open Source libéré des droits de distribution n'est pas forcément gratuit s'il est pris en charge, supporté techniquement (installation, configuration, administration ...) par un distributeur commercial.

### Communauté PostgreSQL

La communauté PostgreSQL est réunie sur le site Internet:

<https://www.postgresql.org>

# Présentation générale

The screenshot shows the official PostgreSQL website at <https://www.postgresql.org>. The header includes a logo, navigation links (Home, About, Download, Documentation, Community, Developers, Support, Donate, Your account), and a search bar. A banner at the top announces the release of PostgreSQL 11.1, 10.6, 9.6.11, 9.5.15, 9.4.20, and 9.3.25 on November 8th, 2018. The main title "PostgreSQL: The World's Most Advanced Open Source Relational Database" is displayed over a background image of a person working on a computer. Below the title are two buttons: "Download →" and "New to PostgreSQL?". The "New to PostgreSQL?" section includes a sub-section titled "Latest Releases" with a list of recent versions and their release dates.

PostgreSQL: The world's most advanced open source relational database

8th November 2018: PostgreSQL 11.1, 10.6, 9.6.11, 9.5.15, 9.4.20, and 9.3.25 Released!

Home About Download Documentation Community Developers Support Donate Your account

PostgreSQL: The World's Most Advanced Open Source Relational Database

Download → New to PostgreSQL?

New to PostgreSQL?

Latest Releases

2018-11-08 - PostgreSQL 11.1, 10.6, 9.6.11, 9.5.15, 9.4.20, and 9.3.25 Released! The PostgreSQL Global Development Group has released an update to all supported versions of our database system, **including 11.1, 10.6, 9.6.11, 9.5.15, 9.4.20, and 9.3.25**. This release fixes one **security issue** as well as bugs reported over the last three months. This update is also the final release for PostgreSQL 9.3, which is now end-of-life and will no longer receive any bug or security fixes. If your environment still uses PostgreSQL 9.3, please make plans to update to a community supported versions as soon as possible.

11.1 · 2018-11-08 · [Notes](#)  
10.6 · 2018-11-08 · [Notes](#)  
9.6.11 · 2018-11-08 · [Notes](#)

# Présentation générale

---

- Le site officiel met à disposition des utilisateurs une multitude d'informations :
  - Un support technique à travers des listes de diffusion (mailing lists) classées par thèmes (pgsql-general, pgsql-admin, pgsql-bugs...) : lien « **Community** » → « **Mailing Lists** »,
  - La possibilité de déclarer un bug : lien « **Support** » → « **Report a Bug** »,
  - La documentation officielle consultable en ligne ou téléchargeable : lien « **Documentation** »,
  - Une liste des sites miroirs pour télécharger les logiciels sources, binaires .... de PostgreSQL : lien « **Download** »,
  - Des liens vers les différentes communautés internationales de PostgreSQL (ex: la communauté Française de PostgreSQL sur <https://www.postgresql.fr>) : lien « **Community** » → « **International Sites** » → « **Français** »,
  - Un moteur de recherche,
  - Informations pour les développeurs (les roadmaps, les todo lists, informations coding et testing sur les versions à venir, Developer FAQ ... ) : lien « **Developers** »,
  - Documents techniques (performances, migration, sécurité ...) à travers un wiki : lien « **Documentation** » → « **Wiki** »,
  - Une liste de liens vers d'autres sites Internet (contributors, sponsors ..),
  - produits utilisables avec PostgreSQL (outils administration, interfaces applicatifs ...),
  - ...
- Quelques contributeurs de PostgreSQL(lien « **Community** » → « **Contributors** ») :

Mozilla, 2ndQuadrant (Simon Riggs), NEC, NTT (réPLICATION), Google, VMware (Heikki Linnakangas), HP, Red Hat, Skype, EnterpriseDB (Bruce Momjian, Dave Page...), Dalibo, Microsoft Skype (projet skytools), Crunchy Data Solutions (Tom Lane, Stephen Frost, Joe Conway, Greg Smith) ...

- Quelques références :

Yahoo (2 petaoctets), Météo France, RATP, IGN, CNAF ([http://www.postgresql-sessions.org/\\_media/pg\\_sessions\\_projet\\_cnaf.pdf](http://www.postgresql-sessions.org/_media/pg_sessions_projet_cnaf.pdf)), Le Bon Coin, Zalando, TripAdvisor...

# Présentation générale

The screenshot shows a web browser displaying the PostgreSQL About page at <https://www.postgresql.org>. The page has a dark blue header with the PostgreSQL logo and navigation links for Home, About, Download, Documentation, Community, Developers, Support, Donate, Your account, and a search bar. The main content area is titled "ABOUT" and includes sections for "WHAT IS POSTGRESQL?", "WHY USE POSTGRESQL?", and a list of features. The "WHAT IS POSTGRESQL?" section describes PostgreSQL as a powerful, open-source object-relational database system. The "WHY USE POSTGRESQL?" section highlights its many features, including support for the SQL standard and various data types like JSON/JSONB and Geometry. The feature list includes Data Types, Data Integrity, and Concurrency/Performance.

**WHAT IS POSTGRESQL?**

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the **POSTGRES** project at the University of California at Berkley and has more than 30 years of active development on the core platform.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on **all major operating systems**, has been **ACID**-compliant since 2001, and has powerful add-ons such as the popular **PostGIS** geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.

**Getting started** with using PostgreSQL has never been easier - pick a project you want to build, and let PostgreSQL safely and robustly store your data.

**WHY USE POSTGRESQL?**

PostgreSQL comes with **many features** aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being **free and open source**, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from **different programming languages** without recompiling your database!

PostgreSQL tries to conform with the **SQL standard** where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 10 release in October 2017, PostgreSQL conforms to at least 160 of the 179 mandatory features for SQL:2011 Core conformance, where as of this writing, no relational database meets full conformance with this standard.

Below is an inexhaustive of various features found in PostgreSQL, with more being added in every **major release**:

- **Data Types**
  - Primitives: Integer, Numeric, String, Boolean
  - Structured: Date/Time, Array, Range, UUID
  - Document: JSON/JSONB, XML, Key-value (Hstore)
  - Geometry: Point, Line, Circle, Polygon
  - Customizations: Composite, Custom Types
- **Data Integrity**
  - UNIQUE, NOT NULL
  - Primary Keys
  - Foreign Keys
  - Exclusion Constraints
  - Explicit Locks, Advisory Locks
- **Concurrency, Performance**
  - Indexing: B-tree, Multicolumn, Expressions, Partial

# Présentation générale

---

## 2. Fonctionnalités

PostgreSQL est un des SGBDR les plus complets dans le monde open source et offre nombre de fonctionnalités que l'on ne trouve généralement que dans les SGBDR commerciaux.

### A. Liste des principales fonctionnalités de PostgreSQL

**PostgreSQL** est totalement **ACID** (**A**tomicité, **C**ohérence, **I**solation, **D**urabilité) et est le SGBD le plus respectueux du standard ANSI-SQL 2008.

**PostgreSQL** est un serveur transactionnel gérant les accès concurrents (**MVCC**) dans un environnement multi-utilisateurs.

Gestion de l'intégrité référentielle (contraintes d'intégrité référentielle, triggers), du modèle Objet-Relationnel des données.

Supporte un très grand nombre de types de données (INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP, stockage des binary large objects (photos, sons, ou vidéos). Possibilité de créer ses propres types de données.

Support de différents langages de procédure côté serveur. **PostgreSQL** gère en natif le langage de procédures PL/pgSQL mais peut aussi utiliser les langages Perl, Python, Java, PHP, TCL, Ruby...comme langages de procédures intégrés.

**PostgreSQL** fonctionne en Client/serveur avec un processus par utilisateur.

Interfaces de programmation souples et nombreux. Les API de développement pour le SGBDR **PostgreSQL** sont nombreux : Python, Perl, PHP, ODBC, Java / JDBC, TCL, C/C++, Ruby...

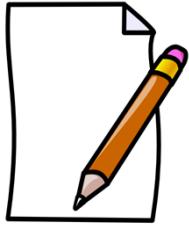
Optimiseur / Planificateur de requête sophistiqué (optimisation statistique CBO des requêtes, optimisation des accès aux tables avec l'utilisation des index, optimiseur génétique...).

**WAL** (Write Ahead Logging). Journal enregistrant les modifications effectuées par les transactions, Point In Time Recovery (PITR), sauvegardes à chaud (online/hot backups), serveur Warm Standby et High Availability, Streaming Replication avec réPLICATION asynchrone ou synchrone....

Gestion de tablespaces, tables partitionnées, requêtes parallélisées, triggers, vues, ...

# Présentation générale

---



# Présentation générale

---

## 3. Documentation

### A. Les différentes sources de documentation

La documentation (des différentes versions) de PostgreSQL est téléchargeable à partir du site Internet [www.postgresql.org](http://www.postgresql.org) ou visualisable sur ce même site (lien **Documentation**).

#### Preface

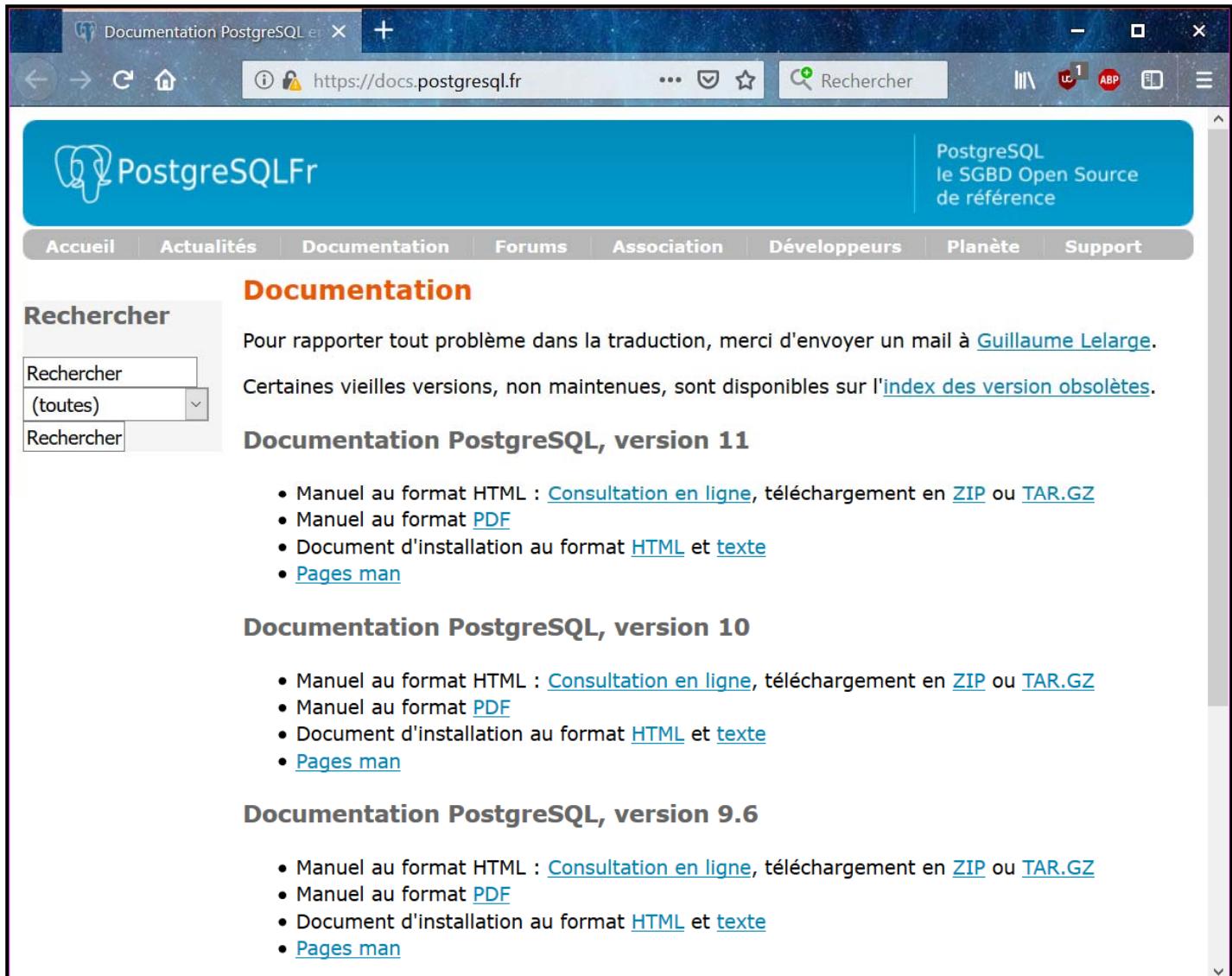
- I. Tutorial
  - II. The SQL Language
  - III. Server Administration
  - IV. Client Interfaces
  - V. Server Programming
  - VI. Reference
  - VII. Internals
  - VIII. Appendixes
- Bibliography
- Index

Il existe une documentation en français sur le site Internet de la communauté française PostgreSQL :

<https://docs.postgresql.fr>

Voir l'exemple page suivante.

# Présentation générale



The screenshot shows a web browser window displaying the PostgreSQLFr Documentation website. The URL in the address bar is https://docs.postgresql.fr. The page title is "Documentation PostgreSQL". The header includes the PostgreSQL logo, the text "PostgreSQLFr", and "PostgreSQL le SGBD Open Source de référence". The navigation menu at the top includes links for Accueil, Actualités, Documentation, Forums, Association, Développeurs, Planète, and Support. On the left, there is a search bar with a dropdown menu set to "(toutes)". The main content area is titled "Documentation". It contains instructions for reporting translation issues and links to old versions of the documentation. Below this, three sections are listed: "Documentation PostgreSQL, version 11", "Documentation PostgreSQL, version 10", and "Documentation PostgreSQL, version 9.6", each with a list of download links.

Documentation PostgreSQL

PostgreSQLFr

PostgreSQL  
le SGBD Open Source  
de référence

Accueil | Actualités | Documentation | Forums | Association | Développeurs | Planète | Support

**Documentation**

Rechercher

Rechercher (toutes) Rechercher

Pour rapporter tout problème dans la traduction, merci d'envoyer un mail à [Guillaume Lelarge](#). Certaines vieilles versions, non maintenues, sont disponibles sur l'[index des version obsolètes](#).

**Documentation PostgreSQL, version 11**

- Manuel au format HTML : [Consultation en ligne](#), téléchargement en [ZIP](#) ou [TAR.GZ](#)
- Manuel au format [PDF](#)
- Document d'installation au format [HTML](#) et [texte](#)
- [Pages man](#)

**Documentation PostgreSQL, version 10**

- Manuel au format HTML : [Consultation en ligne](#), téléchargement en [ZIP](#) ou [TAR.GZ](#)
- Manuel au format [PDF](#)
- Document d'installation au format [HTML](#) et [texte](#)
- [Pages man](#)

**Documentation PostgreSQL, version 9.6**

- Manuel au format HTML : [Consultation en ligne](#), téléchargement en [ZIP](#) ou [TAR.GZ](#)
- Manuel au format [PDF](#)
- Document d'installation au format [HTML](#) et [texte](#)
- [Pages man](#)

# Présentation générale

---

En plus de ces documentations, il existe d'autres sources d'informations vous aidant à installer et utiliser PostgreSQL :

- Les textes d'aides **--help**

Les pages de la documentation Reference Manual visualisables pour chaque commande : **commande --help**

- Les **FAQs**

Les Frequently Asked Questions pour PostgreSQL sur le site Internet [www.postgresql.org](http://www.postgresql.org) ou dans le source récupéré sur Internet.

- Les fichiers **READMEs** (récupérés avec les sources)
- Des documents techniques à travers le lien **wiki** de la zone shortcuts (Todo list, howtos, articles, guides d'installation, administration, maintenance, performances, migration...)
- Le site Internet [www.postgresql.org](http://www.postgresql.org)

Informations sur les dernières livraisons (versions), les caractéristiques / améliorations à venir et autres informations pour rendre votre travail avec PostgreSQL plus productif.

- Les forums, Yourself !!

PostgreSQL est un logiciel libre et dépend donc de la communauté des utilisateurs pour son évolution (documentation, support, code source...). Vous pourrez profiter de l'expérience des autres et faire profiter les autres de votre expérience sur ce produit à travers les listes de diffusion, les forums, les moteurs de recherche, la déclaration d'un bug, le test des versions candidates ou en vous impliquant dans la traduction de la documentation officielle ou la relecture des traductions...

# Présentation générale

---

## B. Les projets annexes

Il existe un très grand nombre de produits utilisables avec **PostgreSQL**. Ces produits sont disponibles sous formes de « **contrib** » (contributions) livrés avec les sources PostgreSQL (mais non installés) ou sous forme de **produits logiciels récupérables sur Internet** chez les éditeurs de logiciels.

Quelques exemples de produits :

- **pgAdmin** Logiciel libre, outil graphique d'administration de serveurs PostgreSQL (<http://www.pgadmin.org>).
- **pg\_rman** Outil de gestion des sauvegardes et restaurations des serveurs PostgreSQL ([http://ossc-db.github.io/pg\\_rman/index.html](http://ossc-db.github.io/pg_rman/index.html)).
- **pgPool-II** Gestionnaire de connexions (pooling de connexions) généralement utilisé par les applications Web gérant un grand nombre de connexions. Possibilité de faire aussi de la répartition de charge sur plusieurs serveurs PostgreSQL ou de la réPLICATION. (<http://pgpool.projects.postgresql.org>).
- **pgBouncer** Gestionnaire de connexions plus simple, efficace et moins consommateur que pgPool. Il car il ne fait que du pooling de connexion. (<http://pgbouncer.projects.postgresql.org> ).
- **pgBadger** Outil d'analyse des fichiers de log du serveur PostgreSQL permettant de repérer les requêtes consommatrices, longues ou fréquentes, de créer des rapports détaillés sous différentes formes (<http://dalibo.github.io/pgbadger/>).
- **pgCuu** (PostgreSQL Cluster utilization) est un outil de contrôle et d'audit des performances de PostgreSQL (<http://pgcluu.darold.net/>).
- **PoWA** Outil dédié PostgreSQL d'analyse de la charge des bases de données (<http://dalibo.github.io/powa>).
- **PostGIS** Module de gestion de données spatiales pour un serveur PostgreSQL.
- Exemples de contrib livrées avec le source PostgreSQL : **auto\_explain**, **oid2name**, **pg\_stat\_statements**, **file\_fdw** ... Ces contributions sont détaillées dans la documentation officielle « Appendix F. Additional Supplied Modules ».

# Présentation générale

---

## 4. Déclaration d'un bug

### A. Comment et où déclarer un bug

Le chapitre « Préface » paragraphe « Bug Reporting Guidelines » de la documentation vous aide à déclarer un bug qui pourra être géré efficacement.

Avant de rédiger votre bug :

- Lisez ce paragraphe (conseils de rédaction du bug, description et déroulement précis des faits, sources, fichiers, outils, interfaces de l'exemple utilisé, les sorties (les messages d'erreur client et serveur, une copie du terminal, rien ne se passe...) produites, les options et variables d'environnement, la version de PostgreSQL (select version();), la machine utilisée...),
- Lisez ou relisez la documentation pour vérifier que vous pouvez réellement faire ce que vous tentez de faire,
- Vérifiez dans la « TODO list » et dans les FAQs si votre bug est déjà connu ou pas.

Vous pouvez envoyer vos bugs à la mailing liste

***pgsql-bugs@postgresql.org***

ou remplir la grille de déclaration de bug disponible sur le site Internet

***<https://www.postgresql.org>***

# Présentation générale

The screenshot shows a web browser window for the PostgreSQL Support website (<https://www.postgresql.org>). The page title is "PostgreSQL: Support". The main content area is titled "Support ?" and discusses various support options available for users, including documentation, community sections, mailing lists, IRC, and Slack. It also mentions commercial support and professional services. A sidebar on the left lists "Quick Links" such as Support, Versioning, Policy, Security, Professional Services, Hosting Solutions, and Report a Bug. An arrow points from the "Report a Bug" link in the sidebar to the "Bug Reporting" section below. The footer of the page includes a "Rechercher" search bar and a "Your account" link.

PostgreSQL has a wide variety of community and commercial support options available for users, including:

- Documentation
- The **Community** section, which details the support options available to users from the PostgreSQL community
- **Mailing Lists**
- **IRC**
- **Slack**

**Commercial support** is also available from one of the many companies providing **professional services** to the PostgreSQL community. A list of companies that provide **PostgreSQL-specific hosting** is also available.

## Bug Reporting

Found a bug in PostgreSQL? Please read over our **bug reporting guidelines** and then report it using our **bug reporting form**.

You can see previous bug reports, and track your own on the [pgsql-bugs@lists.postgresql.org](mailto:pgsql-bugs@lists.postgresql.org) mailing list.

## Security Issue Reporting

If you believe you have found a security issue, please send an email to [security@postgresql.org](mailto:security@postgresql.org).

# Présentation générale

---

## 5. Les versions de PostgreSQL

### A. Versions majeures et mineures

Une version majeure apporte de nouvelles fonctionnalités, changements de comportement, nouvelles commandes, paramètres... Il y a une nouvelle version majeure tous les 1,5 an environ.

Une version mineure ne contient que des corrections de bugs et de failles de sécurité. Il y a une nouvelle version mineure tous les 3 mois environ.

**Ancienne numérotation (avant la version 10)** : sous la forme de trois nombres **N.N.N**. Les deux premiers nombres indiquant la version majeure et le troisième nombre la version mineure. Exemple : **9.6.11** version majeure 9.6 et douzième version mineure (de 0 à 11) de cette version majeure.

**Nouvelle numérotation depuis la version 10** : sous la forme de deux nombres **N.N**. Le premier nombre indiquant la version majeure et le deuxième nombre la version mineure. Exemple : **11.1** version majeure 11 et deuxième version mineure (0 à 1) de cette version majeure.

Les versions majeures **8.n** positionnent PostgreSQL dans les SGBD professionnels de haut niveau (tablespaces, procédures stockées en Java, PITR, réplication asynchrone, version native pour Windows...). Les versions majeures **9.n** enrichissent les solutions de haute disponibilité / réplication, performances et vues matérialisées.

### B. Versions actuelles

Version **9.5** (janvier 2016) upsert, row level security, index brin, opérateurs cube, rollup et grouping set (EOL février 2021). Version mineure actuelle **9.5.15**.

Version **9.6** (septembre 2016) **parallélisme** des requêtes, améliorations des checkpoints du monitoring ... (EOL novembre 2021). Version mineure actuelle **9.6.11**.

Version **10** (octobre 2017) numérotation des versions, nommage, partitionnement, **réPLICATION logique**, parallélisme des requêtes étendues, améliorations des performances ... (EOL novembre 2022). Version mineure actuelle **10.6**.

Version **11** (octobre 2018) amélioration du partitionnement et du parallélisme, Just-In-Time compilation, améliorations des performances ... (EOL novembre 2023). Version mineure actuelle **11.1**.

# Présentation générale

<https://www.postgresql.org> -----> Support -----> Versioning Policy -----> Releases

The screenshot shows a web browser window with the title "PostgreSQL: Versioning Policy". The address bar displays the URL "https://www.p...". The main content is a table titled "Releases" with the following data:

Version	Current minor	Supported	First Release	Final Release
11	11.1	Yes	October 18, 2018	November 9, 2023
10	10.6	Yes	October 5, 2017	November 10, 2022
9.6	9.6.11	Yes	September 29, 2016	November 11, 2021
9.5	9.5.15	Yes	January 7, 2016	February 11, 2021
9.4	9.4.20	Yes	December 18, 2014	February 13, 2020
9.3	9.3.25	No	September 9, 2013	November 8, 2018
9.2	9.2.24	No	September 10, 2012	November 9, 2017
9.1	9.1.24	No	September 12, 2011	October 27, 2016
9.0	9.0.23	No	September 20, 2010	October 8, 2015
8.4	8.4.22	No	July 1, 2009	July 24, 2014
8.3	8.3.23	No	February 4, 2008	February 7, 2013
8.2	8.2.23	No	December 5, 2006	December 5, 2011
8.1	8.1.23	No	November 8, 2005	November 8, 2010
8.0	8.0.26	No	January 19, 2005	October 1, 2010
7.4	7.4.30	No	November 17, 2003	October 1, 2010

## 2. INSTALLATION

---

### Objectif

A la fin de ce module, vous serez capable d'installer le logiciel PostgreSQL sur votre système :

- Pré-installation avec les sources sous Linux
- Installation avec les sources sous Linux
- Post-installation
- Autres méthodes d'installation :
  - Exemple de récupération de la méthode d'installation via des rpm
  - Installation via les « One click installer »
  - Exemple installation sous Windows
- Upgrading des bases de données existantes

# Installation

---



# Installation

---

## 1. Pré-installation avec les sources

### A. Introduction

Ce module présente les prérequis et les principales étapes de l'installation et de la configuration de PostgreSQL à partir des **sources**. Ce n'est pas un document de référence remplaçant la documentation officielle de PostgreSQL (PostgreSQL 10.n Documentation, fichier texte INSTALL livré avec les sources, Release Notes...).

### B. Packages logiciels utiles

L'exemple d'installation de PostgreSQL dans ce cours a été fait sur une plateforme Linux 7.2 64 bits à partir des **sources** récupérées sur le site Internet officiel de PostgreSQL. En plus du code source de PostgreSQL, des outils logiciels GNU sont nécessaires pour l'installation. La plupart des distributions Linux possèdent déjà ces outils.

### C. GNU make

GNU make est nécessaire (d'autres programmes make peuvent ne pas fonctionner). GNU make est souvent installé sous le nom gmake. Version minimum de gmake est 3.80.

Vérification de la présence et de la version de gmake :

```
[root@CentOS764b ~]# gmake --version
GNU Make 3.82
Built for x86_64-redhat-linux-gnu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[root@CentOS764b ~]#
```

# Installation

---

## D. Compilateur C ISO/ANSI

Pour la compilation des sources de PostgreSQL, vous avez besoin d'un compilateur C ISO/ANSI, il est préférable d'utiliser une version récente du compilateur C de GNU qui est GCC. PostgreSQL est connu pour être compilé avec une grande variété de compilateurs de différents vendeurs.

## E. GNU zip et GNU tar

L'utilitaire de compression / décompression de fichiers appelé **gzip** (ou bzip2) sera utile pour décompresser le fichier contenant la source de **PostgreSQL** récupéré sur Internet.

L'utilitaire **tar** sera utile pour décompresser l'archive contenant la source de PostgreSQL récupéré sur Internet.

## F. GNU Readline library

La GNU Readline Library est utilisée par défaut pour faciliter l'utilisation de **psql** grâce à l'historique des commandes. Si cette bibliothèque est déjà présente sur le système, elle sera utilisée par défaut et **psql** intégrera ces facilités. Si vous ne voulez pas utiliser cette bibliothèque, vous aurez à spécifier l'option **--without-readline** à la configuration des sources.

Vérification de la présence de Readline :

```
[root@CentOS764b ~]# rpm -qa | grep readline
readline-6.2-9.el7.x86_64
readline-devel-6.2-9.el7.x86_64
[root@CentOS764b ~]#
```

# Installation

## G. Packages optionnels

Les packages suivants sont optionnels. Ils ne sont pas nécessaires dans la configuration par défaut, mais sont utiles quand certaines options de configuration de PostgreSQL sont validées.

### ▪ Perl / Python

L'utilisation des langages de programmation serveur PL/Perl ou PL/Python nécessite une installation complète de Perl ou Python.

Vérification de la présence d'un package :

```
[root@CentOS764b ~]# rpm -qa|grep perl
perl-HTTP-Tiny-0.033-3.el7.noarch
perl-Scalar-List-Utils-1.27-248.el7.x86_64
perl-Socket-2.010-3.el7.x86_64
perl-Pod-Simple-3.28-4.el7.noarch
perl-Data-Dumper-2.145-3.el7.x86_64
perl-Pod-Usage-1.63-3.el7.noarch
perl-Filter-1.49-3.el7.x86_64
perl-Time-Local-1.2300-2.el7.noarch
.../...
.../...
.../...
.../...
```

Les installations de ces différents produits ne seront pas traitées dans ce cours.

### ▪ Kerberos, OpenSSL, OpenLDAP...

Pour supporter les services d'authentification ou de cryptage de session.

Si vous avez besoin d'un package GNU, vous pouvez l'obtenir sur le site :

<http://www.gnu.org>

# Installation

## H. Espace disque

Comparé à d'autres SGBD, PostgreSQL nécessite très peu d'espace disque.

Vous avez besoin approximativement de **100 Mo** dans l'arborescence source durant la compilation et de **20 Mo** pour le répertoire d'installation.

Un cluster de base de données vide consomme environ **35 Mo**. Une base prend en espace disque environ **5 fois** la taille du fichier plat texte contenant les mêmes données.

Si vous désirez exécuter les tests de régression, vous aurez besoin temporairement de **150 Mo** supplémentaires.

Vérification de l'espace disque avec la commande **df** :

```
[root@CentOS764b ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
/dev/mapper/centos-root    50G   5.3G   45G  11% /
devtmpfs              2.0G     0   2.0G  0% /dev
tmpfs                 2.1G   88K   2.1G  1% /dev/shm
tmpfs                 2.1G   8.7M   2.1G  1% /run
tmpfs                 2.1G     0   2.1G  0% /sys/fs/cgroup
/dev/mapper/centos-home    26G   33M   26G  1% /home
/dev/sda1               497M  157M  341M 32% /boot
tmpfs                 412M   20K   412M  1% /run/user/0
```

## I. Récupération des sources PostgreSQL

Les sources **PostgreSQL 11.n** peuvent être téléchargés à partir du site Internet officiel :

<https://www.postgresql.org>

Download

↳ Source code : The Source code can be found in the main file browser

    ↳ Répertoire v11.n

# Installation

Le fichier contenant l'intégralité de PostgreSQL est : **postgresql-11.n.tar.gz**

The screenshot shows a web browser window titled "PostgreSQL: File Browser". The URL in the address bar is <https://www.postgresql.org>. The page displays a "File Browser" interface with a sidebar containing "Quick Links" for "Downloads", "Software Catalogue", and "File Browser". The main content area shows a list of "Directories" under the heading "Top → source". A large black arrow points from the text "Top → source" towards the list of versions. The list includes:

- [Parent Directory]
- v11.1
- v11.0
- v10.6
- v10.5
- v10.4
- v10.3
- v10.2
- v10.1
- v10.0
- v9.6.11
- v9.6.10
- v9.6.9
- v9.6.8
- v9.6.7
- v9.6.6

# Installation

---



# Installation

---

## 2. Installation avec les sources

### A. Installation via les sources de PostgreSQL

Copier les sources de PostgreSQL dans un répertoire. Ce répertoire, est utilisé pour **décompresser** le fichier tar.gz récupéré sur Internet puis pour **installer** et **configurer** les sources PostgreSQL.

Ce répertoire ne sera pas celui qui contiendra, plus tard, les fichiers composant le logiciel et vos bases de données.

Dans l'environnement Linux, on utilise généralement le répertoire **/usr/local/src**.

Après avoir copié l'archive dans le répertoire désiré, décompresser l'archive en utilisant l'utilitaire **tar** qui extrait les fichiers sources de l'archive.

# Installation

Exemple d'extraction des fichiers sources de l'archive :

```
[root@CentOS764b ~]# ls postgresql-11*
postgresql-11.1.tar.gz
[root@CentOS764b ~]# cp postgresql-11.1.tar.gz /usr/local/src/
[root@CentOS764b ~]# cd /usr/local/src/
[root@CentOS764b src]# ls postgresql-11*
postgresql-11.1.tar.gz
[root@CentOS764b src]# tar -xzvf postgresql-11.1.tar.gz
postgresql-11.1/
postgresql-11.1/.dir-locals.el
postgresql-11.1/contrib/
postgresql-11.1/contrib/tcn/
postgresql-11.1/contrib/tcn/tcn.control
postgresql-11.1/contrib/tcn/Makefile
postgresql-11.1/contrib/tcn/tcn.c
postgresql-11.1/contrib/tcn/tcn--1.0.sql
postgresql-11.1/contrib/sslinfo/
postgresql-11.1/contrib/sslinfo/sslinfo--1.0--1.1.sql
postgresql-11.1/contrib/sslinfo/sslinfo.control
postgresql-11.1/contrib/sslinfo/sslinfo.c
postgresql-11.1/contrib/sslinfo/sslinfo--unpackaged--1.0.sql
.../...
.../...
.../...
postgresql-11.1/doc/src/Makefile
postgresql-11.1/doc/KNOWN_BUGS
postgresql-11.1/doc/Makefile
postgresql-11.1/doc/TODO
postgresql-11.1/doc/MISSING_FEATURES
postgresql-11.1/doc/bug.template
postgresql-11.1/HISTORY
postgresql-11.1/Makefile
postgresql-11.1/README
postgresql-11.1/COPYRIGHT
postgresql-11.1/GNUmakefile.in
postgresql-11.1/.gitattributes
postgresql-11.1/aclocal.m4
postgresql-11.1/configure.in
postgresql-11.1/INSTALL
[root@CentOS764b src]# ls postgres*
postgresql-11.1.tar.gz

postgresql-11.1:
aclocal.m4 config configure configure.in contrib COPYRIGHT
doc GNUmakefile.in HISTORY INSTALL Makefile README src
[root@CentOS764b src]#
```

# Installation

---

## B. Configuration des sources

La première étape de l'installation est de configurer les sources avant de les compiler et de choisir les options dont vous avez besoin. Cette étape se fait avec le script **configure**. Ce script, effectue tout un ensemble de tests de variables systèmes et détecte les spécificités de votre O.S.

La configuration par défaut (en lançant le script avec la commande `./configure`) valide un serveur PostgreSQL correspondant à la plupart des besoins.

Pour ajouter des options qui valident des fonctionnalités, vous devez les indiquer au lancement du script **configure**. La liste des options est récupérable en exécutant la commande :

« `./configure --help` ».

### ■ Exemples d'options utilisées pour **configure** :

#### **--prefix**

Installe tous les fichiers dans le répertoire PREFIX au lieu de `/usr/local/pgsql`. Les fichiers seront installés dans différents sous-répertoires (bin, doc, include, lib, man, share). Aucun fichier n'est installé directement dans le répertoire PREFIX. Pour des besoins spécifiques, vous pouvez choisir des sous-répertoires personnalisés avec les options `--bindir=directory` `--mandir=directory` `--includedir=directory`...).

Exemple : `./configure --prefix=/usr/local/pg10`

#### **--with-perl**

Valide le langage serveur PL/Perl de PostgreSQL à l'installation.

#### **--with-segsize=SEGSIZE**

Détermine le *segment size* en giga bytes. Les tables volumineuses sont composées de plusieurs fichiers O.S. Chaque fichier a une taille maximum définie par ce *segment size* pour éviter les problèmes de tailles maximum des fichiers qui peuvent exister sur certains O.S. La valeur par défaut de *segment size* est 1 giga bytes.

#### **--with-blocksize=BLOCKSIZE**

Définit la taille des blocs en kilo bytes. C'est l'unité de stockage et des I/O pour les tables. La valeur par défaut est 8k. La valeur doit être multiple de 2 entre 1k et 32k.

#### **--with-krb5**

#### **--with-openssl**

#### **--with-ldap**

#### **--without-readline**

...

# Installation

## Exemple :

```
[root@CentOS764b src]# cd postgresql-11.1/
[root@CentOS764b postgresql-11.1]# pwd
/usr/local/src/postgresql-11.1
[root@CentOS764b postgresql-11.1]# ./configure
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking which template to use... linux
checking whether NLS is wanted... no
checking for default port number... 5432
checking for block size... 8kB
checking for segment size... 1GB
checking for WAL block size... 8kB
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking whether gcc supports -Wdeclaration-after-statement, for CFLAGS... yes
.../...
.../...
.../...
configure: using CPPFLAGS= -D_GNU_SOURCE
configure: using LDFLAGS= -Wl,--as-needed
configure: creating ./config.status
config.status: creating GNUmakefile
config.status: creating src/Makefile.global
config.status: creating src/include/pg_config.h
config.status: creating src/include/pg_config_ext.h
config.status: creating src/interfaces/ecpg/include/ecpg_config.h
config.status: linking src/backend/port/tas/dummy.s to src/backend/port/tas.s
config.status: linking src/backend/port/dynloader/linux.c to
src/backend/port/dynloader.c
config.status: linking src/backend/port/posix_sema.c to
src/backend/port/pg_sema.c
config.status: linking src/backend/port/sysv_shmem.c to
src/backend/port/pg_shmem.c
config.status: linking src/backend/port/dynloader/linux.h to
src/include/dynloader.h
config.status: linking src/include/port/linux.h to src/include/pg_config_os.h
config.status: linking src/makefiles/Makefile.linux to src/Makefile.port
[root@CentOS764b postgresql-11.1]#
```

# Installation

Exemple de **configure** en 8.4.2, avec erreur sur un serveur sans package krb5 :

```
[root@vmcentos53R2 postgresql-8.4.2]# ./configure --with-krb5
checking build system type... i686-pc-Linux-gnu
checking host system type... i686-pc-Linux-gnu
checking which template to use... Linux
checking whether to build with 64-bit integer date/time support... yes
checking whether NLS is wanted... no
checking for default port number... 5432
checking for block size... 8kB
checking for segment size... 1GB
checking for WAL block size... 8kB
checking for WAL segment size... 16MB
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking if gcc supports -Wdeclaration-after-statement... yes
checking if gcc supports -Wendif-labels... yes
checking if gcc supports -fno-strict-aliasing... yes
checking if gcc supports -fwrapv... yes
checking whether the C compiler still works... yes
checking how to run the C preprocessor... gcc -E
checking allow thread-safe client libraries... no
checking whether to build with Tcl... no
checking whether to build Perl modules... no
checking whether to build Python modules... no
checking whether to build with GSSAPI support... no
checking whether to build with Kerberos 5 support... yes
checking whether to build with PAM support... no
checking whether to build with LDAP support... no
.../...
.../...
.../...
checking zlib.h usability... yes
checking zlib.h presence... yes
checking for zlib.h... yes
checking krb5.h usability... no
checking krb5.h presence... no
checking for krb5.h... no
configure: error: header file <krb5.h> is required for Kerberos 5
[root@vmcentos53R2 postgresql-8.4.2]#
```

# Installation

---

## Remarques :

Pour connaître les options de configuration qui ont été utilisées pour installation de PostgreSQL, utiliser la commande ***pg\_config --configure***.

Si après l'installation, vous désirez ajouter une fonctionnalité, il faudra revenir à cette étape, reconfigurer les sources et continuer les étapes suivantes pour construire et installer PostgreSQL.

# Installation

## C. Compilation des sources

Après avoir configuré les sources, vous pouvez les compiler avec la commande **gmake**. La dernière ligne affichée doit être «**All of PostgreSQL is successfully made. Ready to install.** ».

**Exemple :**

```
[root@CentOS764b postgresql-11.1]# pwd
/usr/local/src/postgresql-11.1
[root@CentOS764b postgresql-11.1]# gmake
gmake -C ./src/backend generated-headers
gmake[1] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend »
gmake -C catalog distprep generated-header-symlinks
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend/catalog »
gmake[2]: Rien à faire pour « distprep ».
prereqdir=`cd './' >/dev/null && pwd` && \
cd '../../../../../src/include/catalog/' && for file in pg_proc_d.h pg_type_d.h
pg_attribute_d.h pg_class_d.h pg_attrdef_d.h pg_constraint_d.h pg_inherits_d.h
pg_index_d.h pg_operator_d.h pg_opfamily_d.h pg_opclass_d.h pg_am_d.h pg_amop_d.h
pg_amproc_d.h pg_language_d.h pg_largeobject_metadata_d.h pg_largeobject_d.h
pg_aggregate_d.h pg_statistic_ext_d.h pg_statistic_d.h pg_rewrite_d.h pg_trigger_d.h
pg_event_trigger_d.h pg_description_d.h pg_cast_d.h pg_enum_d.h pg_namespace_d.h
pg_conversion_d.h pg_depend_d.h pg_database_d.h pg_db_role_setting_d.h
pg_tablespace_d.h pg_pltemplate_d.h pg_authid_d.h pg_auth_members_d.h pg_shdepend_d.h
pg_shdescription_d.h pg_ts_config_d.h pg_ts_config_map_d.h pg_ts_dict_d.h
pg_ts_parser_d.h pg_ts_template_d.h pg_extension_d.h pg_foreign_data_wrapper_d.h
pg_foreign_server_d.h pg_user_mapping_d.h pg_foreign_table_d.h pg_policy_d.h
pg_replication_origin_d.h pg_default_acl_d.h pg_init_privs_d.h pg_seclabel_d.h
pg_shseclabel_d.h pg_collation_d.h pg_partitioned_table_d.h pg_range_d.h
pg_transform_d.h pg_sequence_d.h pg_publication_d.h pg_publication_rel_d.h
pg_subscription_d.h pg_subscription_rel_d.h schemapg.h; do \
    rm -f $file && ln -s "$prereqdir/$file" . ; \
.....
.....
gmake -C test/perl all
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/test/perl »
gmake[2]: Rien à faire pour « all ».
gmake[2] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/test/perl »
gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src »
gmake -C config all
gmake[1] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/config »
gmake[1]: Rien à faire pour « all ».
gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/config »
All of PostgreSQL successfully made. Ready to install.
[root@CentOS764b postgresql-11.1]#
```

# Installation

## D. Installation des fichiers

L'installation de PostgreSQL se fait avec la commande « **gmake install** ». Les fichiers seront installés dans les répertoires précisés au moment de la configuration des sources (--prefix, --bindir, --libdir, --docdir...). Le répertoire par défaut est **/usr/local/pgsql**. Il est possible d'installer la partie cliente uniquement (voir le document d'installation).

**Exemple :**

```
[root@CentOS764b postgresql-11.1]# pwd
/usr/local/src/postgresql-11.1
[root@CentOS764b postgresql-11.1]# gmake install
gmake -C ./src/backend generated-headers
gmake[1] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend »
gmake -C catalog distprep generated-header-symlinks
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend/catalog »
gmake[2]: Rien à faire pour « distprep ».
gmake[2]: Rien à faire pour « generated-header-symlinks ».
gmake[2] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/backend/catalog »
gmake -C utils distprep generated-header-symlinks
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend/utils »
gmake[2]: Rien à faire pour « distprep ».
gmake[2]: Rien à faire pour « generated-header-symlinks ».
...
...
...
/usr/bin/install      -c          -m      644           ./Makefile.shlib
'/usr/local/pgsql/lib/pgxs/src/Makefile.shlib'
/usr/bin/install      -c          -m      644           ./nls-global.mk
'/usr/local/pgsql/lib/pgxs/src/nls-global.mk'
gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src »
gmake -C config install
gmake[1] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/config »
/usr/bin/mkdir -p '/usr/local/pgsql/lib/pgxs/config'
/usr/bin/install      -c          -m      755           ./install-sh
'/usr/local/pgsql/lib/pgxs/config/install-sh'
/usr/bin/install      -c          -m      755           ./missing
'/usr/local/pgsql/lib/pgxs/config/missing'
gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/config »
PostgreSQL installation complete.
```

# Installation

---

## 3. Post-installation

### A. Création du propriétaire de PostgreSQL

Créer un compte utilisateur Linux pour le serveur PostgreSQL qui sera propriétaire des fichiers. Le nom de cet utilisateur est généralement **postgres**. Cet utilisateur sera l'administrateur de PostgreSQL.

Création de l'utilisateur **postgres** sous root :

```
[root@CentOS764b postgresql-11.1]# adduser postgres
```

### B. Variables d'environnement de l'utilisateur postgres

Si vous avez installé PostgreSQL dans le répertoire **/usr/local/pgsql** ou dans un répertoire qui n'est pas balayé par défaut dans la recherche de programme, vous devez ajouter **/usr/local/pgsql/bin** (ou le répertoire indiqué au niveau du paramètre --bindir lors de la configuration des sources) dans votre **PATH**.

#### Extrait du fichier /home/postgres/.bash\_profile

```
[root@CentOS764b postgresql-11.1]# su - postgres
[postgres@CentOS764b ~]$ pwd
/home/postgres
[postgres@CentOS764b ~]$ nano .bash_profile
```

```
# User specific environment and startup programs

PATH=$PATH:$HOME/bin:/usr/local/pgsql/bin
export PATH
```

# Installation

### **C. Plateformes supportées**

PostgreSQL a été vérifié par la communauté de développement pour fonctionner sur un grand nombre de plateformes comme AIX, BSD/OS, FreeBSD, HP-UX, **Linux**, MacOS X, Solaris, Tru64 UNIX, Windows (32 et 64 bits).

La liste complète des plateformes supportées et non supportées se trouve dans la documentation « [PostgreSQL 11.n Comprehensive Manual](#) » Part III « Server Administration » chapitre 16 « Installation from source code ».

PostgreSQL: Documentation

Documentation → PostgreSQL 11

Supported Versions: [Current \(11\)](#) / [10](#) / [9.6](#) / [9.5](#) / [9.4](#)

Development Versions: [devel](#)

Unsupported versions: [9.3](#) / [9.2](#) / [9.1](#) / [9.0](#) / [8.4](#) / [8.3](#) / [8.2](#) / [8.1](#) / [8.0](#) / [7.4](#) / [7.3](#) / [7.2](#) / [7.1](#)

16.6. Supported Platforms

Prev      Up      Chapter 16. Installation from Source Code      Home      Next

## 16.6. Supported Platforms

A platform (that is, a CPU architecture and operating system combination) is considered supported by the PostgreSQL development community if the code contains provisions to work on that platform and it has recently been verified to build and pass its regression tests on that platform. Currently, most testing of platform compatibility is done automatically by test machines in the [PostgreSQL Build Farm](#). If you are interested in using PostgreSQL on a platform that is not represented in the build farm, but on which the code works or can be made to work, you are strongly encouraged to set up a build farm member machine so that continued compatibility can be assured.

In general, PostgreSQL can be expected to work on these CPU architectures: x86, x86\_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc 64, ARM, MIPS, MIPSEL, and PA-RISC. Code support exists for M68K, M32R, and VAX, but these architectures are not known to have been tested recently. It is often possible to build on an unsupported CPU type by configuring with `--disable-spinlocks`, but performance will be poor.

PostgreSQL can be expected to work on these operating systems: Linux (all recent distributions), Windows (Win2000 SP4 and later), FreeBSD, OpenBSD, NetBSD, macOS, AIX, HP/UX, and Solaris. Other Unix-like systems may also work but are not currently being tested. In most cases, all CPU architectures supported by a given operating system will work. Look in [Section 16.7](#) below to see if there is information specific to your operating system, particularly if using an older system.

# Installation

## D. Tests de régression

Vous avez la possibilité d'effectuer un ensemble de tests livrés par PostgreSQL (opérations SQL de base, test des fonctionnalités avancées...) sur le nouveau serveur installé (ces tests de régression peuvent aussi être effectués avant l'installation « gmake install »).

Ces tests vérifient que PostgreSQL fonctionne comme prévu sur votre machine.

Ces tests peuvent mettre en évidence des problèmes que vous auriez à l'exécution du serveur PostgreSQL.

Ces tests sont **optionnels** mais il est conseillé de les exécuter avec la commande « **gmake check** ».

Ces tests **ne doivent pas être exécutés sous root**.

- Exemple :

```
[root@CentOS764b postgresql-11.1]# su - postgres
Dernière connexion : lundi 26 novembre 2018 à 15:46:27 CET sur pts/1
[postgres@CentOS764b ~]$ cd /usr/local/src/postgresql-11.1/
[postgres@CentOS764b postgresql-11.1]$
[postgres@CentOS764b postgresql-11.1]# gmake check
gmake -C ./src/backend generated-headers
gmake[1] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend »
gmake -C catalog distprep generated-header-symlinks
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend/catalog »
gmake[2]: Rien à faire pour « distprep ».
gmake[2]: Rien à faire pour « generated-header-symlinks ».
gmake[2] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/backend/catalog »
gmake -C utils distprep generated-header-symlinks
gmake[2] : on entre dans le répertoire « /usr/local/src/postgresql-11.1/src/backend/utils »
gmake[2]: Rien à faire pour « distprep ».
gmake[2]: Rien à faire pour « generated-header-symlinks ».
gmake[2] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/backend/utils »
gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/backend »
....
```

# Installation

```
gmake[2] : on quitte le répertoire « /usr/local/src/postgresql-11.1/contrib/spi »
rm -rf ./testtablespace
mkdir ./testtablespace
PATH="/usr/local/src/postgresql-11.1/tmp_install/usr/local/pgsql/bin:$PATH"
LD_LIBRARY_PATH="/usr/local/src/postgresql-11.1/tmp_install/usr/local/pgsql/lib"
..7.../src/test/regress/pg_regress --temp-instance=./tmp_check --inputdir=. --
bindir=      --dlpath=. --max-concurrent-tests=20 --schedule=./parallel_schedule
===== creating temporary instance =====
===== initializing database system =====
===== starting postmaster =====
running on port 60849 with PID 14708
===== creating database "regression" =====
CREATE DATABASE
ALTER DATABASE
===== running regression test queries =====
test tablespace          ... ok
parallel group (20 tests): regproc pg_lsn varchar txid boolean text float4 int2 oid
char int4 money float8 uuid name int8 bit enum rangetypes numeric
    boolean          ... ok
    char            ... ok
    name           ... ok
    varchar         ... ok
    text            ... ok
    int2            ... ok
    int4            ... ok
    int8            ... ok
    oid             ... ok
    float4          ... ok
    float8          ... ok
    bit              ... ok
    numeric         ... ok
    txid            ... ok
    uuid            ... ok
    enum            ... ok
    money           ... ok
    rangetypes      ... ok
    pg_lsn          ... ok
    regproc          ... ok
test strings          ... ok
test numerology       ... ok
parallel group (20 tests): lseg point abstime timetz tinterval circle path time
macaddr line tstypes macaddr8 reftime interval date inet box polygon timestamp
timestamptz
....
```

# Installation

```
json ... ok
jsonb ... ok
json_encoding ... ok
indirect_toast ... ok
equivclass ... ok
parallel group (19 tests): conversion prepare plancache limit returning xml temp
without_oid copy2 sequence with polymorphism truncate rowtypes rangefuncs domain
largeobject alter_table plpgsql
    plancache ... ok
    limit ... ok
    plpgsql ... ok
    copy2 ... ok
    temp ... ok
    domain ... ok
    rangefuncs ... ok
    prepare ... ok
    without_oid ... ok
    conversion ... ok
    truncate ... ok
    alter_table ... ok
    sequence ... ok
    polymorphism ... ok
    rowtypes ... ok
    returning ... ok
    largeobject ... ok
    with ... ok
    xml ... ok
parallel group (7 tests): hash_part reloptions identity partition_join
partition_aggregate partition_prune indexing
    identity ... ok
    partition_join ... ok
    partition_prune ... ok
    reloptions ... ok
    hash_part ... ok
    indexing ... ok
    partition_aggregate ... ok
test event_trigger ... ok
test fast_default ... ok
test stats ... ok
===== shutting down postmaster =====
===== removing temporary instance =====

=====
All 189 tests passed.
=====

gmake[1] : on quitte le répertoire « /usr/local/src/postgresql-11.1/src/test/regress »
[postgres@CentOS764b postgresql-11.1]$
```

## Remarques :

Le fichier **repertoire\_installation\_des\_sources/src/test/regress/README** et la documentation vous indiquent plus précisément comment interpréter les résultats de ces tests.

# Installation

---



# Installation

---

## 4. Autres méthodes d'installation

### A. Packages binaires

PostgreSQL est aussi disponible sous forme de packages pour la plupart des plateformes Linux (RHEL, Fedora, CentOS, Debian, Ubuntu, SUSE...).

**Exemple** de récupération des packages binaires de **PostgreSQL 11** pour Linux RHEL/CentOS/Fedora/Scientific/Oracle variants :

<https://www.postgresql.org> ----> Download ----> Zone « Binary Packages » Linux lien « Red Hat Family Linux » ----> Zone « Direct RPM download »

Pour récupérer la méthode d'installation via les packages pour Linux RHEL, Fedora et CentOS :

<http://www.postgresql.org> ----> Download ----> Zone « Binary Packages » Linux lien « Red Hat Family » ----> Zone « Direct RPM download » -----> Onglet « Yum Howto » -----> Zone « Installation procedure and details » lien « here »

# Installation

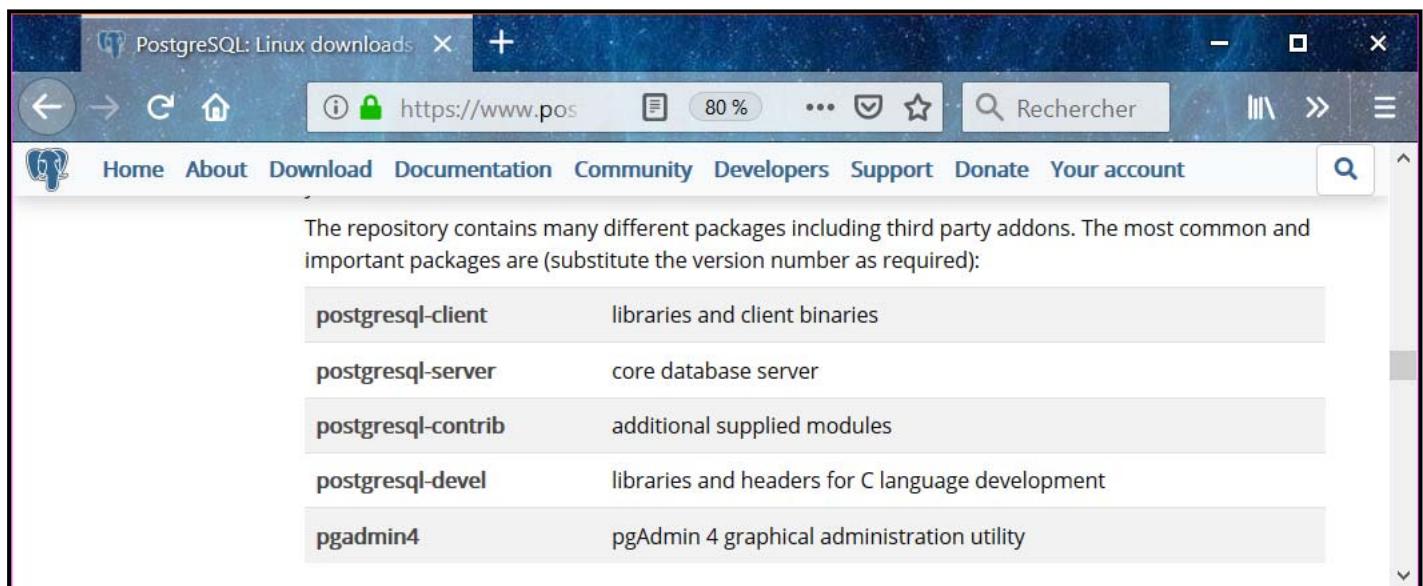
Les packages binaires peuvent aussi être récupérés via le **repository** de PostgreSQL :

<https://www.postgresql.org> ----> Download ----> Zone « Binary Packages » Linux lien « Red Hat Family Linux » ----> Zone « PostgreSQL Yum Repository ».

The screenshot shows a web browser displaying the PostgreSQL Yum Repository page. The URL in the address bar is <https://www.postgresql.org/download/linux/redhat/yum/>. The page title is "PostgreSQL Yum Repository".  
The content includes:

- A note: "The PostgreSQL Yum Repository will integrate with your normal systems and patch management, and provide automatic updates for all supported versions of PostgreSQL throughout the support **lifetime** of PostgreSQL."
- A note: "The PostgreSQL Yum Repository currently supports:"
  - Red Hat Enterprise Linux
  - CentOS
  - Scientific Linux
  - Oracle Linux
  - Fedora\*
- A note: "\*Note: due to the shorter support cycle on Fedora, all supported versions of PostgreSQL are not available on this platform. We do not recommend using Fedora for server deployments."
- Instructions: "To use the PostgreSQL Yum Repository, follow these steps:"
  1. Select version:  
→ 11
  2. Select platform:  
→ CentOS 7
  3. Select architecture:  
→ x86\_64
  4. Install the repository RPM:  
→ `yum install https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm`
  5. Install the client packages:  
→ `yum install postgresql11`
  6. Optionally install the server packages:  
→ `yum install postgresql11-server`
  7. Optionally initialize the database and enable automatic start:  
→ `/usr/pgsql-11/bin/postgresql-11-setup initdb  
systemctl enable postgresql-11  
systemctl start postgresql-11`

# Installation



The repository contains many different packages including third party addons. The most common and important packages are (substitute the version number as required):

<code>postgresql-client</code>	libraries and client binaries
<code>postgresql-server</code>	core database server
<code>postgresql-contrib</code>	additional supplied modules
<code>postgresql-devel</code>	libraries and headers for C language development
<code>pgadmin4</code>	pgAdmin 4 graphical administration utility

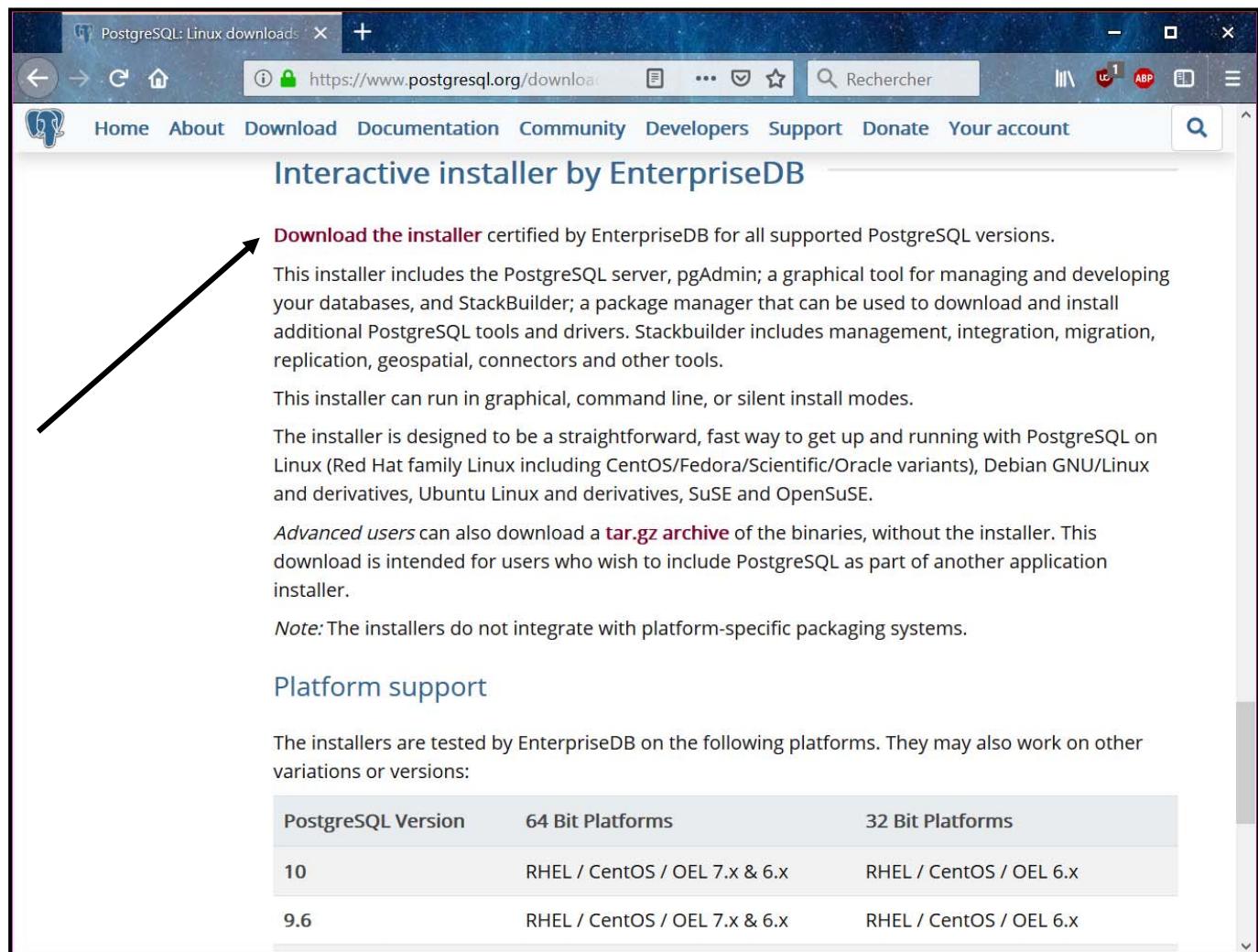
# Installation

## B. Installation via les « One click installer » (Graphical Installer)

Il est aussi possible d'installer PostgreSQL en utilisant des « installer » graphiques (type Windows) disponibles sous Linux, Mac OS x, FreeBSD, Solaris, Windows.

Pour Linux, ils sont disponibles en 32 et 64 bits incluant PostgreSQL, pgAdmin... L'installer a été testé avec un nombre de distributions Linux récentes et devrait fonctionner sur **Ubuntu** 6.06 ou plus, **Fedora** 6 ou plus, **CentOS/Red Hat Enterprise Linux** 4 ou plus et d'autres.

<http://www.postgresql.org> ----> Download ----> Zone « Binary Packages » Linux lien « Red Hat Family » ----> Zone « Interactive installer by EnterpriseDB » lien « Download the installer ».

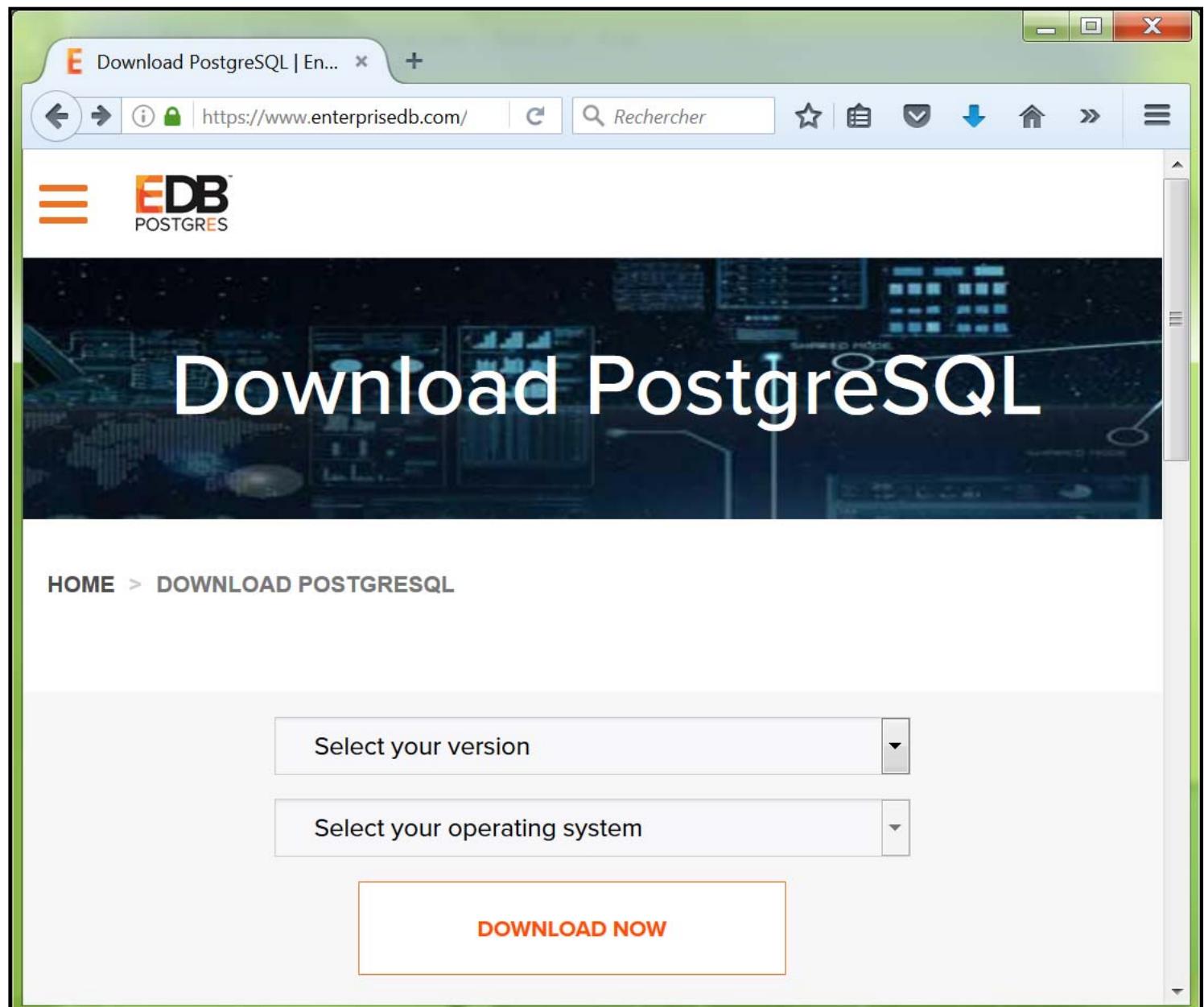


The screenshot shows a web browser window with the URL <https://www.postgresql.org/download/linux/redhat/>. The page title is "Interactive installer by EnterpriseDB". A large black arrow points from the left towards the "Download the installer" link. The text on the page describes the installer, which includes the PostgreSQL server, pgAdmin, StackBuilder, and other tools, and can run in graphical, command line, or silent modes. It supports various Linux distributions like Red Hat, CentOS, Fedora, Scientific, Oracle, Debian, Ubuntu, SuSE, and OpenSuSE. Advanced users can download a tar.gz archive of the binaries. A note states that installers do not integrate with platform-specific packaging systems. A table at the bottom lists PostgreSQL versions 10 and 9.6 along with their supported 64-bit and 32-bit platforms.

PostgreSQL Version	64 Bit Platforms	32 Bit Platforms
10	RHEL / CentOS / OEL 7.x & 6.x	RHEL / CentOS / OEL 6.x
9.6	RHEL / CentOS / OEL 7.x & 6.x	RHEL / CentOS / OEL 6.x

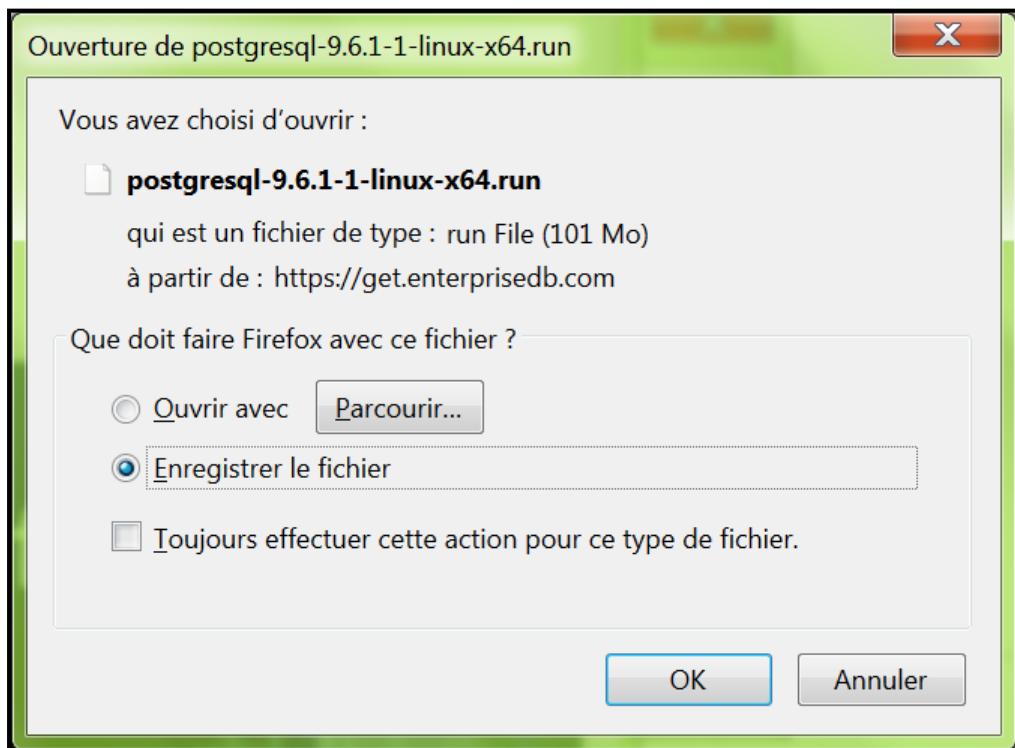
# Installation

---



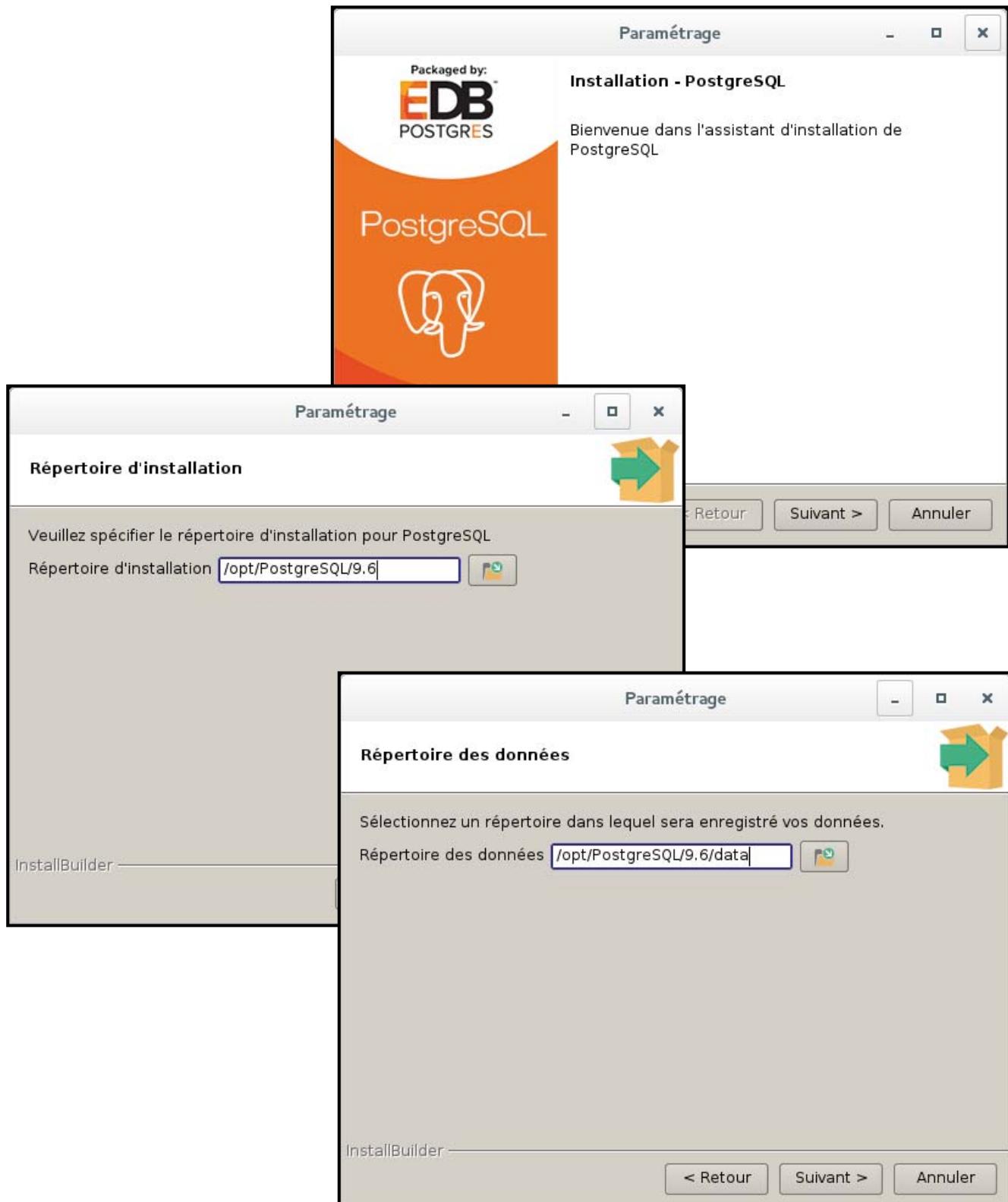
# Installation

---

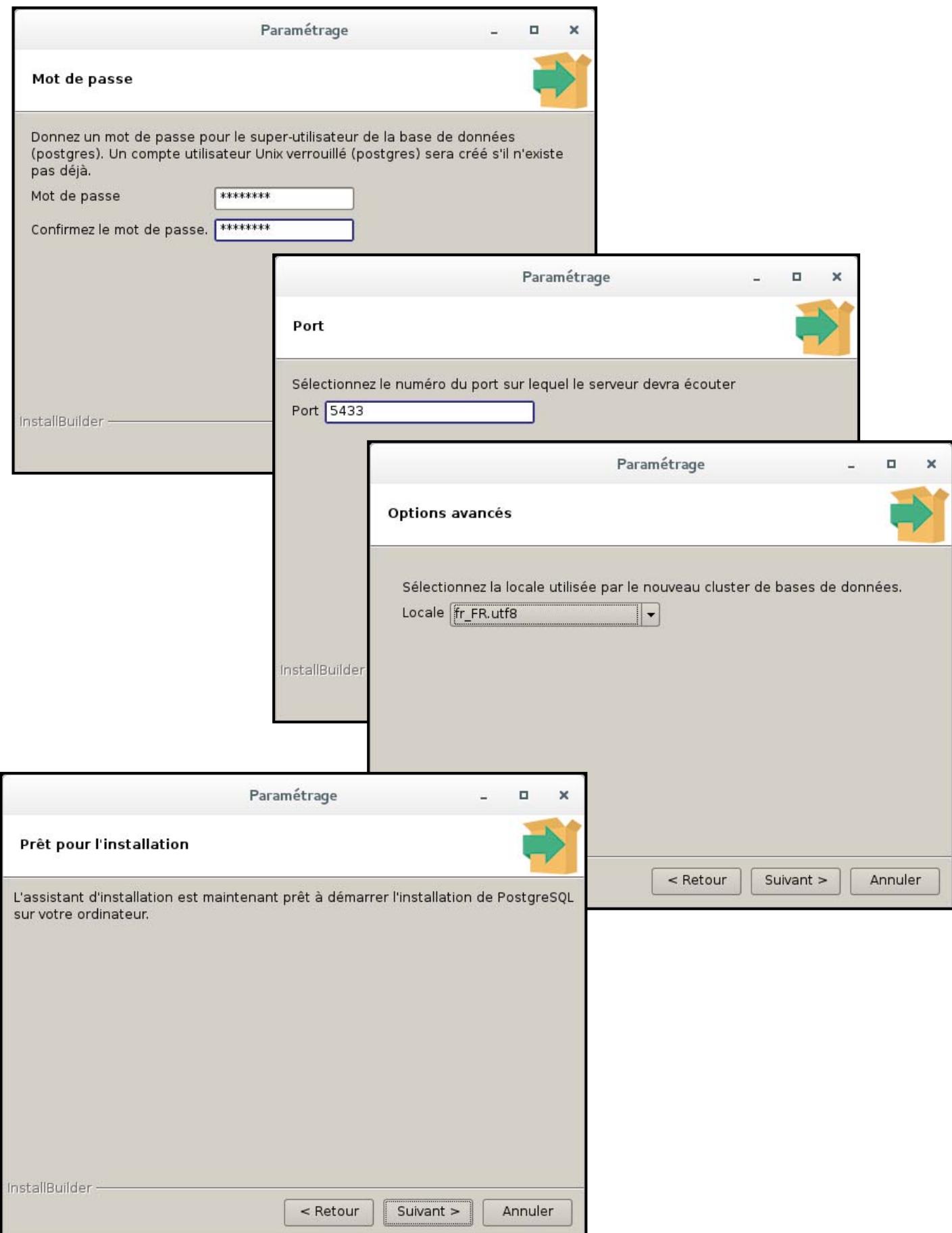


# Installation

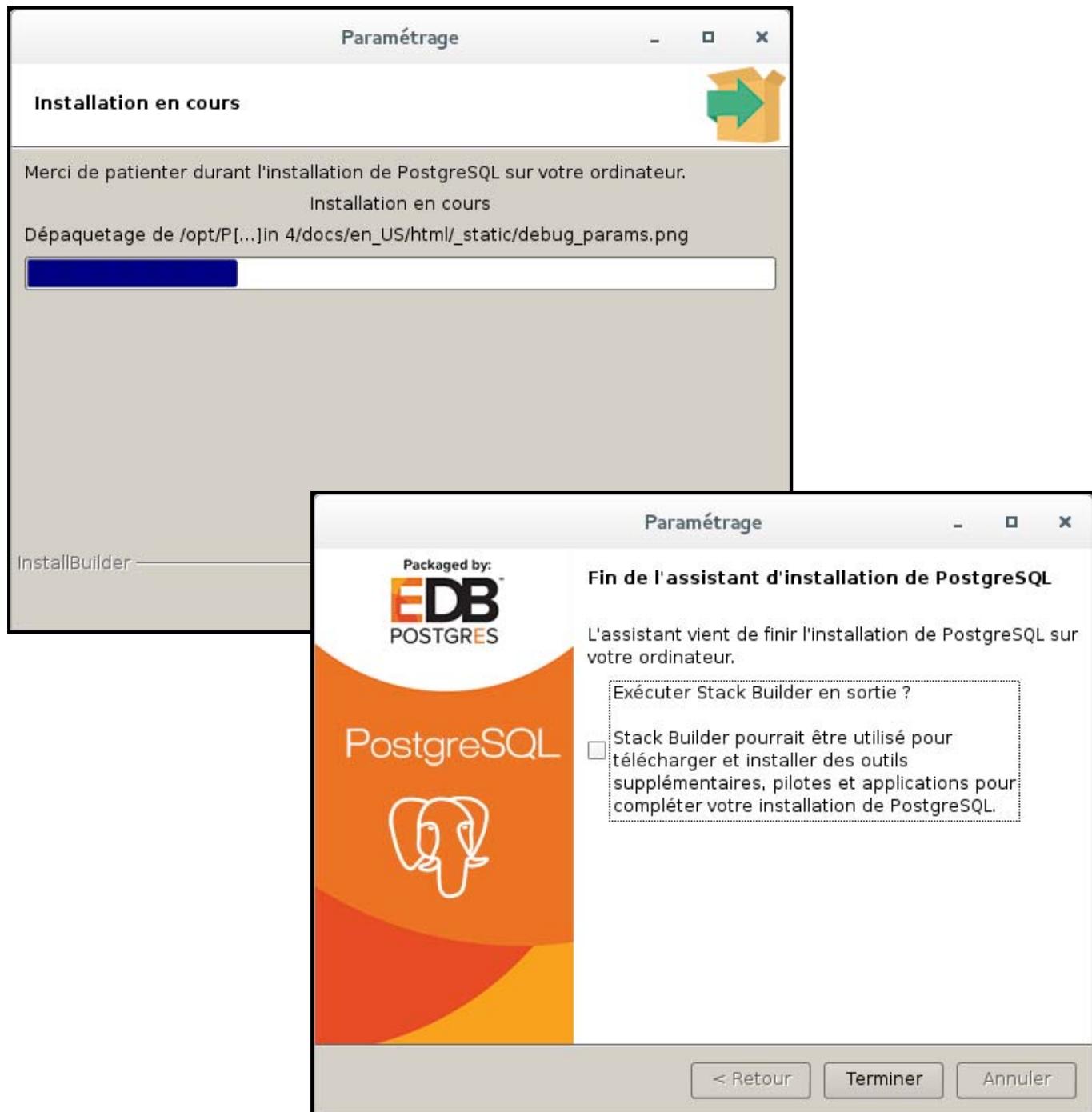
```
[root@CentOS764b TP]# chmod 777 postgresql-9.6.1-1-linux-x64.run  
[root@CentOS764b TP]# ./postgresql-9.6.1-1-linux-x64.run
```



# Installation



# Installation



# Installation

---



# Installation

---

## C. Upgrading

Le format interne de stockage des données peut changer d'une version **majeure** de PostgreSQL à une autre. Par conséquent, si vous upgradez une installation existante qui n'a pas un numéro de version « 11.n », vous devez sauvegarder et restaurer vos données (si vous upgradez une installation « 11.n », la nouvelle version peut utiliser les fichiers de données actuels donc vous pouvez sauter les étapes de sauvegarde et restauration ci-dessous car ils sont inutiles) :

**1- Sauvegarde des données**                            **pg\_dumpall > outputfile**

**2- Arrêt de l'ancien serveur**                            **pg\_ctl stop**

**3- Renommer ou supprimer le répertoire de l'ancienne installation**

**mv /usr/local/pgsql /usr/local/pgsql.old**

**4- Installer la nouvelle version**

**5- Créer un nouveau database cluster (initdb -D...)**

**6- Restaurer les modifications de vos précédents fichiers pg\_hba.conf et postgresql.conf**

**7- Démarrer le nouveau serveur**                            **pg\_ctl start**

**8- Restaure vos données depuis la sauvegarde (faite en 1) en utilisant le psql de la nouvelle version**

**psql -d postgres -f outputfile**

**Remarques** : voir la documentation pour plus d'informations.

# Installation

---



## **3. MISE EN ŒUVRE D'UNE INSTANCE POSTGRESQL**

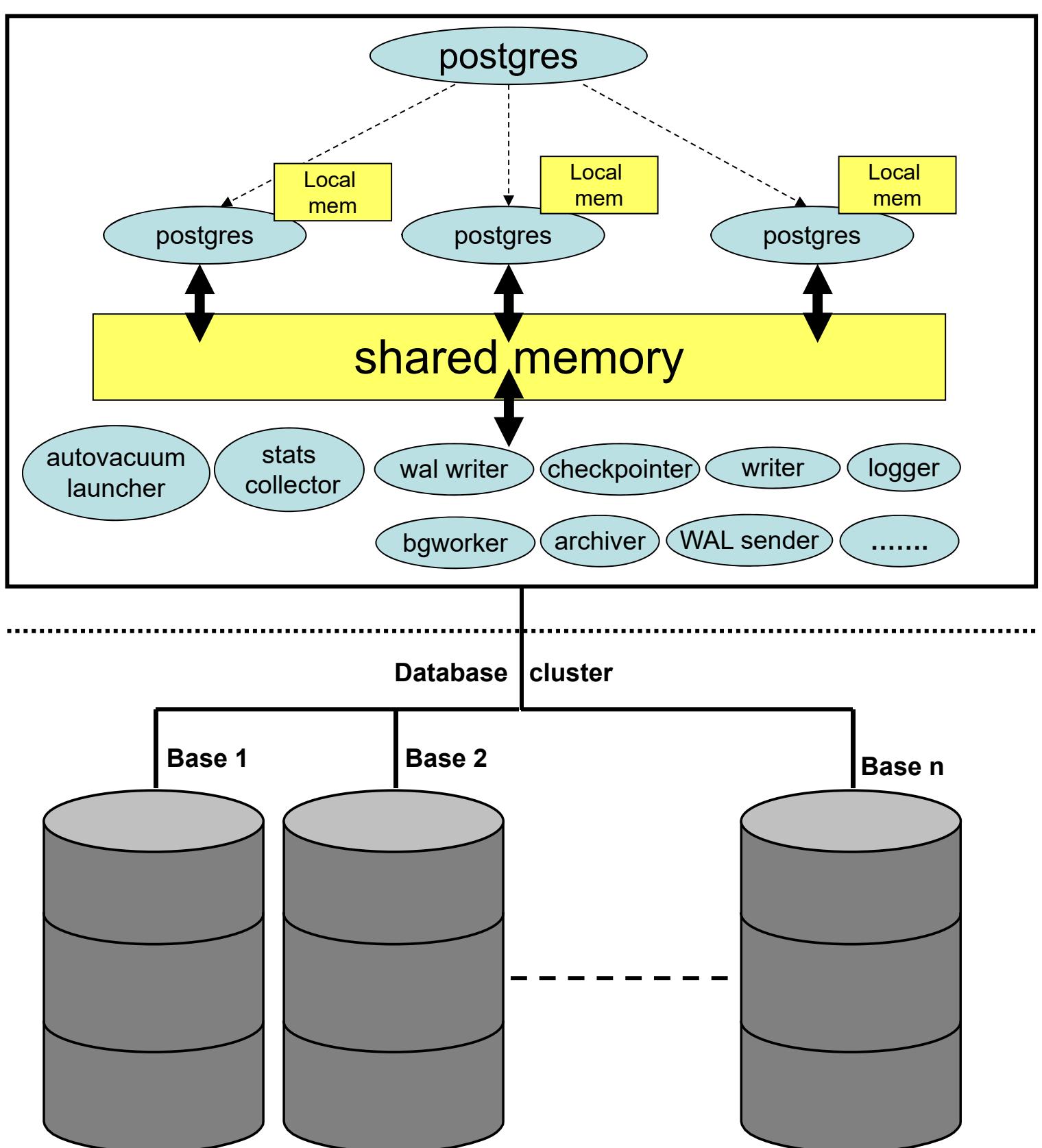
---

### **Objectif**

A la fin de ce module, vous serez capable de mettre en œuvre un serveur de données PostgreSQL :

- Initialisation d'une instance PostgreSQL
- Démarrage et arrêt d'une instance PostgreSQL avec postgres et pg\_ctl
- Configuration de l'instance (fichier postgresql.conf)
- Démarrage et arrêt automatique de l'instance PostgreSQL

# Mise en œuvre d'un serveur de données PostgreSQL



# Mise en œuvre d'un serveur de données PostgreSQL

---

## 1. Initialisation d'un serveur PostgreSQL

### A. Le compte utilisateur de PostgreSQL

Le serveur PostgreSQL est géré avec l'utilisateur Linux **postgres** qui lui est dédié. Cet utilisateur doit être le seul propriétaire des données gérées par le serveur PostgreSQL.

### B. Création d'un Database Cluster

Avant de faire quoi ce soit, vous devez initialiser une zone de stockage, pour les bases de données, sur disque appelée **Database Cluster** ou dit autrement créer une instance.

Un **Database Cluster** est un ensemble de bases de données gérées par une **seule instance** PostgreSQL.

Après création de l'instance, le **Database Cluster** contient 3 bases de données (template1, template0 et postgres).

D'un point de vue système de fichiers, le **Database Cluster** sera un simple répertoire dans lequel toutes les données seront stockées (répertoire appelé **data directory** ou **data area**). Il n'y a pas de répertoire par défaut.

La création de l'instance ou **Database Cluster** se fait avec la commande **initdb**, le choix du répertoire des données avec l'option -D ou la variable d'environnement système **PGDATA** (depuis la 9.0, possibilité de faire un initdb avec pg\_ctl).

Sous Windows, lors de l'installation, un **Database Cluster** est créé par défaut dans le répertoire **C:\Program Files\PostgreSQL\11.n\data**. La commande **initdb** est utilisable en mode ligne dans une fenêtre « Invite de commandes » sous Windows.

Si vous installez PostgreSQL en utilisant un « **Graphical installer** », une instance ou **Database Cluster** est créé dans la foulée de l'installation du produit PostgreSQL (voir écran « Data Directory » en bas de la page 53 du module installation).

L'exécution de cette commande se fera avec l'utilisateur linux **postgres**. Cet utilisateur doit avoir les droits d'écriture dans le répertoire qui sera donné au niveau de l'option -D.

# Mise en œuvre d'un serveur de données PostgreSQL

Exemple d'initialisation d'un *Database Cluster* :

```
[root@CentOS764b ~]# su - postgres
Dernière connexion : lundi 26 novembre 2018 à 15:56:50 CET sur pts/1
[postgres@CentOS764b ~]$ initdb -D /home/postgres/data
The files belonging to this database system will be owned by user
"postgres".
This user must also own the server process.

The database cluster will be initialized with locale "fr_FR.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "french".

Data page checksums are disabled.

creating directory /home/postgres/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

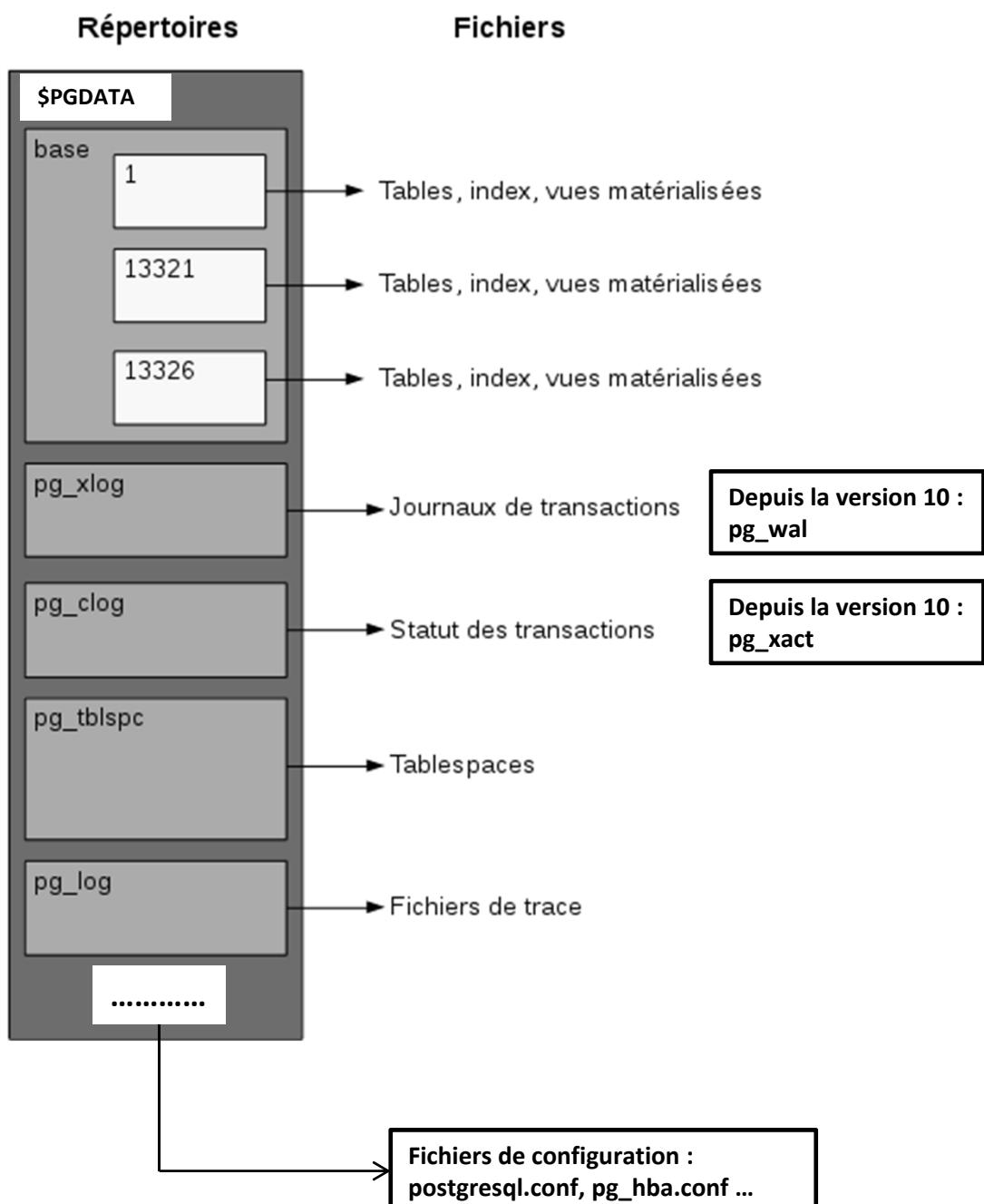
Success. You can now start the database server using:

pg_ctl -D /home/postgres/data -l logfile start

[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

Extrait de l'arborescence créée :



# Mise en œuvre d'un serveur de données PostgreSQL

---

## Remarques :

- Option **-X** de initdb permet de créer le répertoire des journaux de transaction WAL (pg\_xlog) en dehors de la zone des données PGDATA pour améliorer les performances et la sécurité.
- Option **-U** permet de choisir le nom du super utilisateur PostgreSQL qui sera créé dans le nouveau serveur de données (par défaut, on prend le même nom que l'utilisateur qui soumet la commande initdb).
- ...

# Mise en œuvre d'un serveur de données PostgreSQL

---

## 2. Démarrage et arrêt d'une instance PostgreSQL

### A. Introduction

Avant qu'un client ne puisse se connecter, vous devez démarrer le serveur base de données (l'instance). A la fin de la commande initdb, on vous indique comment démarrer le serveur PostgreSQL.

Dans ce module, nous verrons différentes manières de démarrer/arrêter une instance PostgreSQL en utilisant pg\_ctl (postgres) ou un script SysV.

La commande **pg\_ctl** fonctionne de la même manière quel que soit le système d'exploitation et doit être utilisée par le propriétaire de PostgreSQL qui est généralement **postgres**.

Le script de type SysV (un exemple est livré avec PostgreSQL) est lui prévu pour être lancé par un autre utilisateur, exemple : **root**.

### B. Démarrage et arrêt de PostgreSQL avec pg\_ctl / postgres

Le serveur base de données est appelé **postgres**. Il doit savoir où se trouvent les données à utiliser. Ceci est fait avec l'option **-D**.

Il est possible de démarrer le serveur de données avec la commande de bas niveau « **postgres** » en utilisant le compte dédié à PostgreSQL (généralement **postgres**) en background.

# Mise en œuvre d'un serveur de données PostgreSQL

Exemple de lancement de `postgres` en arrière plan (background) :  
**PostgreSQL 10 : un process de plus : logical replication launcher**

```
[postgres@CentOS764b ~]$ postgres -D /home/postgres/data > serverlog 2>&1 &
[1] 3318
[postgres@CentOS764b ~]$ ps fxo pid,cmd | grep [p]ostgres

3318  \_ postgres -D /home/postgres/data
3320  |   \_ postgres: checkpointer
3321  |   \_ postgres: background writer
3322  |   \_ postgres: walwriter
3323  |   \_ postgres: autovacuum launcher
3324  |   \_ postgres: stats collector
3325  |   \_ postgres: logical replication launcher
[postgres@CentOS764b ~]$
```

## Remarque :

Il est possible de préciser d'autres options au lancement de la commande `postgres`. Quelques options seront vues dans les pages suivantes.

**Depuis la 9.5**, pour identifier plus facilement plusieurs instances fonctionnant en même temps sur la même machine, il est possible d'affecter un nom à une instance en utilisant le paramètre de configuration « `cluster_name` ». Exemple en donnant la valeur **PROD** au paramètre « `cluster_name` » :

```
[postgres@CentOS764b ~]$ ps fxo pid,cmd | grep [p]ostgres

3992  \_ postgres -D /home/postgres/data
3994  |   \_ postgres: PROD: checkpointer
3995  |   \_ postgres: PROD: background writer
3996  |   \_ postgres: PROD: walwriter
3997  |   \_ postgres: PROD: autovacuum launcher
3998  |   \_ postgres: PROD: stats collector
3999  |   \_ postgres: PROD: logical replication launcher
[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

---

Les syntaxes Shell devenant rapidement fastidieuses, **postgres** n'est généralement pas appelé directement mais est lancé par l'utilitaire **pg\_ctl** (exemple indiqué à la fin de l'exécution de la commande initdb) ou un script SysV. **pg\_ctl** est un habillage de la commande **postgres** facilitant les tâches d'administration d'un serveur PostgreSQL. **pg\_ctl** permet de démarrer / arrêter / redémarrer / vérifier l'état d'un serveur PostgreSQL.

La syntaxe de **pg\_ctl** peut être obtenue avec la commande « pg\_ctl --help ».

La commande **start** de pg\_ctl démarre le serveur postmaster de PostgreSQL. Cette commande doit être exécutée par l'utilisateur **postgres**.

La commande **stop** de pg\_ctl arrête le serveur postmaster de PostgreSQL. La commande doit être exécutée par l'utilisateur auquel appartient le process.

La commande **restart** de pg\_ctl effectue dans la foulée les commandes stop puis start de pg\_ctl. Les options du dernier démarrage étant enregistrées dans un fichier (postmaster.opts) du répertoire de données, elle seront donc récupérées pour le restart. Ne pas utiliser ce fichier pour placer vos options personnalisées de démarrage car ce fichier est écrasé à chaque start.

La commande **reload** de pg\_ctl provoque la relecture des fichiers de configuration de postmaster (postgresql.conf, pg\_hba.conf...). Cela permet de changer des options dans les fichiers de configuration qui ne nécessitent pas un redémarrage complet pour prendre effet.

La commande **status** de pg\_ctl vous permet de vérifier l'état de votre serveur.

Pour faciliter l'utilisation de la commande pg\_ctl, vous pouvez charger la variable d'environnement système **PGDATA** avec la valeur du répertoire de données pour éviter d'avoir à préciser ce répertoire avec l'option -D à chaque commande pg\_ctl.

**Sous Windows**, il est possible d'utiliser ou pas le « Gestionnaire des services » pour démarrer / arrêter un serveur PostgreSQL. Les options **register** / **unregister** de pg\_ctl permettent de créer / supprimer un service associé un serveur PostgreSQL.

# Mise en œuvre d'un serveur de données PostgreSQL

## Exemples :

```
[postgres@CentOS764b ~]$ pg_ctl -help
pg_ctl is a utility to initialize, start, stop, or control a PostgreSQL server.

Usage:

pg_ctl init[db] [-D DATADIR] [-s] [-o OPTIONS]
pg_ctl start      [-D DATADIR] [-l FILENAME] [-W] [-t SECS] [-s]
                  [-o OPTIONS] [-p PATH] [-c]
pg_ctl stop       [-D DATADIR] [-m SHUTDOWN-MODE] [-W] [-t SECS] [-s]
pg_ctl restart    [-D DATADIR] [-m SHUTDOWN-MODE] [-W] [-t SECS] [-s]
                  [-o OPTIONS] [-c]
pg_ctl reload     [-D DATADIR] [-s]
pg_ctl status     [-D DATADIR]
pg_ctl promote   [-D DATADIR] [-W] [-t SECS] [-s]
pg_ctl kill       SIGNALNAME PID

Common options:

-D, --pgdata=DATADIR      location of the database storage area
-s, --silent                only print errors, no informational messages
-t, --timeout=SECS         seconds to wait when using -w option
-V, --version               output version information, then exit
-w, --wait                  wait until operation completes (default)
-W, --no-wait               do not wait until operation completes
-?, --help                  show this help, then exit
If the -D option is omitted, the environment variable PGDATA is used.

Options for start or restart:

-c, --core-files          allow postgres to produce core files
-l, --log=FILENAME         write (or append) server log to FILENAME
-o, --options=OPTIONS      command line options to pass to postgres
                           (PostgreSQL server executable) or initdb
-p PATH-TO-POSTGRES        normally not necessary

Options for stop or restart:

-m, --mode=MODE            MODE can be "smart", "fast", or "immediate"

Shutdown modes are:

smart           quit after all clients have disconnected
fast            quit directly, with proper shutdown (default)
immediate      quit without complete shutdown; will lead to recovery on restart

Allowed signal names for kill:
ABRT HUP INT QUIT TERM USR1 USR2

Report bugs to <pgsql-bugs@postgresql.org>.
[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

---

## C. Principales options

### ▪ **D datadir**

Spécifie le répertoire contenant les fichiers de la base de données. Si cela n'est pas spécifié, la variable d'environnement système **PGDATA** est utilisée.

### ▪ **I filename**

Spécifie le nom du fichier de log utilisé par le serveur pour y consigner les informations d'activité des bases de données de votre cluster (démarrages, arrêts, erreurs...).

### ▪ **m mode**

Utilisé avec la commande « **stop** » et « **restart** » de pg\_ctl pour spécifier le mode d'arrêt de la base de données. 3 modes sont possibles :

- **smart** attend que les clients se déconnectent.
- **fast** (par défaut) n'attend pas que les clients se déconnectent, toutes les transactions actives sont « rollbackées » et les clients sont déconnectés de force et la base est fermée.
- **immediate** avorte tous les process du serveur sans fermeture « propre », un recovery sera provoqué au prochain démarrage de la base.

L'utilisation de la première lettre de ces 3 modes est possible.

### ▪ **o «options»**

Spécifie les options passées directement à postgres. La liste des options utilisables est récupérable avec la commande « **postgres --help** ».

### ▪ **help**

Pour obtenir une aide de la commande pg\_ctl (liste et brève description des paramètres).

**Remarque :** Depuis la 9.0, possibilité de faire un initdb avec pg\_ctl

(pg\_ctl initdb -D /home/postgres/data).

# Mise en œuvre d'un serveur de données PostgreSQL

- Exemples :

```
[postgres@CentOS764b ~]$ pg_ctl start -l /home/postgres/serverlog
waiting for server to start.... done
server started
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: server is running (PID: 4042)
/usr/local/pgsql/bin/postgres
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ cat serverlog
2018-11-27 09:29:25.159 CET [3992] LOG: listening on IPv6 address "::1", port 5432
2018-11-27 09:29:25.160 CET [3992] LOG: listening on IPv4 address "127.0.0.1", port 5432
2018-11-27 09:29:25.162 CET [3992] LOG: listening on Unix socket "/tmp/.s.PGSQL.5432"
2018-11-27 09:29:25.172 CET [3993] LOG: database system was shut down at 2018-11-27
09:29:15 CET
2018-11-27 09:29:25.179 CET [3992] LOG: database system is ready to accept connections
2018-11-27 09:31:54.265 CET [3992] LOG: received fast shutdown request
2018-11-27 09:31:54.267 CET [3992] LOG: aborting any active transactions
2018-11-27 09:31:54.268 CET [3992] LOG: background worker "logical replication launcher"
(PID 3999) exited with exit code 1
2018-11-27 09:31:54.268 CET [3994] LOG: shutting down
2018-11-27 09:31:54.285 CET [3992] LOG: database system is shut down
2018-11-27 09:32:24.166 CET [4042] LOG: listening on IPv6 address "::1", port 5432
2018-11-27 09:32:24.167 CET [4042] LOG: listening on IPv4 address "127.0.0.1", port 5432
2018-11-27 09:32:24.169 CET [4042] LOG: listening on Unix socket "/tmp/.s.PGSQL.5432"
2018-11-27 09:32:24.179 CET [4043] LOG: database system was shut down at 2018-11-27
09:31:54 CET
2018-11-27 09:32:24.187 CET [4042] LOG: database system is ready to accept connections
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ pg_ctl stop
waiting for server to shut down.... done
server stopped
[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: no server running
[postgres@CentOS764b ~]$ pg_ctl start -l /home/postgres/serverlog -o "-B 2500"
waiting for server to start.... done
server started
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: server is running (PID: 4068)
/usr/local/pgsql/bin/postgres "-B" "2500<
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# select name,setting,short_desc from pg_settings
template1-#      where name like '%shared_buffers%';

   name    | setting |           short_desc
-----+-----+-----+
 shared_buffers | 2500     | Sets the number of shared memory buffers used by the server.
(1 row)

template1=# \q
[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

**Remarque** : dans les versions antérieures à la version 10, message « **server starting** » (dans tous les cas même si le serveur échoue au démarrage) et non « **server started** ».

## Exemple sur une version 9.6 :

```
[postgres@CentOS764b ~]$ pg_ctl start -l /home/postgres/serverlog -o "-N 3000"
server starting

[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: no server running

[postgres@CentOS764b ~]$ cat serverlog
LOG:  database system was shut down at 2017-01-12 16:23:07 CET
LOG:  MultiXact member wraparound protections are now enabled
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
LOG:  received fast shutdown request
LOG:  aborting any active transactions
LOG:  autovacuum launcher shutting down
LOG:  shutting down
LOG:  database system is shut down
LOG:  database system was shut down at 2017-01-12 16:25:57 CET
LOG:  MultiXact member wraparound protections are now enabled
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
LOG:  received fast shutdown request
LOG:  aborting any active transactions
LOG:  autovacuum launcher shutting down
LOG:  shutting down
LOG:  database system is shut down
FATAL:  could not create semaphores: Aucun espace disponible sur le
pÃ©riphÃ©rique
DETAIL: Failed system call was semget(5432121, 17, 03600).
HINT: This error does *not* mean that you have run out of disk space. It
occurs when either the system limit for the maximum number of semaphore sets
(SEMMNI), or the system wide maximum number of semaphores (SEMMNS), would be
exceeded. You need to raise the respective kernel parameter. Alternatively,
reduce PostgreSQL's consumption of semaphores by reducing its max_connections
parameter.

The PostgreSQL documentation contains more information about configuring
your system for PostgreSQL.
LOG:  database system is shut down
[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

---

## Remarques :

Comme indiqué précédemment, la liste des options de **postgres** (utilisables lorsque vousappelez directement **postgres** ou via **pg\_ctl** avec son paramètre **-o**) est récupérable avec la commande « **postgres --help** ». Quelques options intéressantes de **postgres** :

- **-D datadir** même signification que pour **pg\_ctl**.
- **-B nnn** spécifie le nombre de shared buffers utilisables par les process server.
- **-F** invalide les appels à **fsync** pour améliorer les performances.
- **-i** autorise les connexions distantes via TCP/IP.
- **-N nnn** spécifie le nombre maximum de connexions clients que **postgres** acceptera.
- **-p nnnn** spécifie le numéro de port (ou fichier socket) d'écoute des demandes de connexions pour l'instance. Si cela n'est pas précisé, on prendra la valeur de la variable d'environnement PGPORT ou la valeur choisie lors de la configuration des sources (généralement 5432).
- **-c name=value** positionne la valeur d'un paramètre run-time (voir documentation « **Part III. Server Administration chapitre 19 Server Configuration** » pour la liste et la description des paramètres). La plupart des autres options de **postgres** sont en fait des raccourcis de l'assignation de tels paramètres.

## Exemples : syntaxes différentes pour faire la même chose :

- **postgres -c work\_mem=1500**
- **pg\_ctl start -o "-c work\_mem=1500"**

## 3. Configuration du serveur

### A. La configuration de l'environnement via le fichier postgresql.conf

Il existe plus de **289** paramètres (décris dans la documentation **Part III. Server Administration chapitre 19 Server Configuration**) qui affecte directement le fonctionnement, les performances de vos bases de données.

Les principaux paramètres font l'objet de raccourcis à travers les paramètres des commandes `postgres` ou `pg_ctl` que nous avons vu dans les pages précédentes. Les paramètres qui ne possèdent pas de raccourcis, sont accessibles via les options `-c` ou `-o` de `postgres` et `pg_ctl`.

Il est aussi possible de centraliser et positionner l'ensemble des paramètres run-time d'un serveur PostgreSQL dans le fichier de configuration **postgresql.conf** se trouvant dans le répertoire des fichiers de données (datadir).

Les paramètres spécifiés via les options `-c` ou `-o` de `postgres` et `pg_ctl` sont prioritaires par rapport aux paramètres équivalents définis dans le fichier **postgresql.conf**

Ce fichier est lu au démarrage/redémarrage/rechargement de l'instance PostgreSQL (`pg_ctl start...` ou `pg_ctl restart...` `pg_ctl reload...`).

Un paramètre par ligne. Les lignes de commentaire débutent par un #.

# Mise en œuvre d'un serveur de données PostgreSQL

## Exemple de fichier postgresql.conf provenant PGDATA :

```
# -----
# PostgreSQL configuration file
# -----
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.) Whitespace may be used. Comments are introduced with
# "#" anywhere on a line. The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
#
# The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
#
# This file is read on server startup and when the server receives a SIGHUP
# signal. If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, run "pg_ctl reload", or execute
# "SELECT pg_reload_conf()". Some parameters, which are marked below,
# require a server shutdown and restart to take effect.
#
# Any parameter can also be given as a command-line option to the server, e.g.,
# "postgres -c log_connections=on". Some parameters can be changed at run time
# with the "SET" SQL command.
#
# Memory units: kB = kilobytes          Time units: ms  = milliseconds
#                 MB = megabytes           s    = seconds
#                 GB = gigabytes          min = minutes
#                 TB = terabytes          h    = hours
#                                         d    = days
#
# -----
# FILE LOCATIONS
# -----
#
# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.
#
#data_directory = 'ConfigDir'          # use data in another directory
#                                     # (change requires restart)
#hba_file = 'ConfigDir/pg_hba.conf'    # host-based authentication file
#                                     # (change requires restart)
#ident_file = 'ConfigDir/pg_ident.conf' # ident configuration file
#                                     # (change requires restart)
#
# If external_pid_file is not explicitly set, no extra PID file is written.
#external_pid_file = ''                # write an extra PID file
#                                     # (change requires restart)
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

#listen_addresses = 'localhost'          # what IP address(es) to listen on;
#                                         # comma-separated list of addresses;
#                                         # defaults to 'localhost'; use '*' for all
#                                         # (change requires restart)

#port = 5432
max_connections = 100
superuser_reserved_connections = 3        # (change requires restart)
.../...
.../...
#-----

# RESOURCE USAGE (except WAL)
#-----

# - Memory -

shared_buffers = 128MB                  # min 128kB
                                         # (change requires restart)
huge_pages = try                        # on, off, or try
                                         # (change requires restart)
temp_buffers = 8MB                      # min 800kB
max_prepared_transactions = 0           # zero disables the feature
                                         # (change requires restart)

# Caution: it is not advisable to set max_prepared_transactions nonzero unless
# you actively intend to use prepared transactions.
work_mem = 4MB                          # min 64kB
maintenance_work_mem = 64MB            # min 1MB
.../...
.../...
#-----

# WRITE-AHEAD LOG
#-----

# - Settings -
wal_level = replica                     # minimal, replica, or logical
                                         # (change requires restart)
fsync = on                             # flush data to disk for crash safety
                                         # (turning this off can cause
                                         # unrecoverable data corruption)
synchronous_commit = on                 # synchronization level;
                                         # off, local, remote_write, remote_apply, or on
                                         # the default is the first option
                                         # supported by the operating system:
                                         #   open_datasync
                                         #   fdatasync (default on Linux)
                                         #   fsync
                                         #   fsync_writethrough
                                         #   open_sync
                                         # recover from partial page writes
                                         # enable compression of full-page writes
                                         # also do full page writes of non-critical
                                         # updates
                                         # (change requires restart)
                                         # min 32kB, -1 sets based on shared_buffers
                                         # (change requires restart)
                                         # 1-10000 milliseconds

wal_buffers = -1
wal_writer_delay = 200ms

.../...
.../...
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
# - Checkpoints -
#checkpoint_timeout = 5min                                # range 30s-1d
max_wal_size = 1GB
min_wal_size = 80MB
#checkpoint_completion_target = 0.5                      # checkpoint target duration, 0.0 - 1.0
#checkpoint_flush_after = 256kB                         # measured in pages, 0 disables
#checkpoint_warning = 30s                               # 0 disables

# - Archiving -
#archive_mode = off                                     # enables archiving; off, on, or always
#archive_command = ''                                    # command to use to archive a logfile segment
# placeholders: %p = path of file to archive
#                 %f = file name only
# e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p
/mnt/server/archivedir/%f'
#archive_timeout = 0                                    # force a logfile segment switch after this
# number of seconds; 0 disables

#-----
# REPLICATION
#-----
# - Sending Servers -
# Set these on the master and on any standby that will send replication data.
#max_wal_senders = 10                                 # max number of wal sender processes
#                                                     # (change requires restart)
#wal_keep_segments = 0                               # in logfile segments; 0 disables
#wal_sender_timeout = 60s                           # in milliseconds; 0 disables

#max_replication_slots = 10                          # max number of replication slots
#                                                     # (change requires restart)
#track_commit_timestamp = off                       # collect timestamp of transaction commit
#                                                     # (change requires restart)

# - Master Server -
# These settings are ignored on a standby server.
.../...
.../...
#-----
# REPORTING AND LOGGING
#-----
# - Where to Log -
#log_destination = 'stderr'                         # Valid values are combinations of
#                                                     # stderr, csvlog, syslog, and eventlog,
#                                                     # depending on platform. csvlog
#                                                     # requires logging_collector to be on.

# This is used when logging to stderr:
#logging_collector = off                            # Enable capturing of stderr and csvlog
#                                                     # into log files. Required to be on for
#                                                     # csvlogs.
#                                                     # (change requires restart)

# These are only used if logging_collector is on:
#log_directory = 'log'                             # directory where log files are written,
#                                                     # can be absolute or relative to PGDATA
#log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'    # log file name pattern,
#                                                     # can include strftime() escapes
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
.../...
.../...
# - What to Log -
.../...
#log_checkpoints = off
#log_connections = off
#log_disconnections = off
#log_duration = off
#log_error_verbosity = default      # terse, default, or verbose messages
#log_hostname = off
#log_line_prefix = '%m [%p] '        # special values:
#                                         #   %a = application name
#                                         #   %u = user name
#                                         #   %d = database name
#                                         #   %r = remote host and port
#                                         #   %h = remote host
#                                         #   %p = process ID
#                                         #   %t = timestamp without milliseconds
#                                         #   %m = timestamp with milliseconds
#                                         #   %n = timestamp with milliseconds (as a Unix
#                                         #       epoch)
#                                         #   %i = command tag
#                                         #   %e = SQL state
#                                         #   %c = session ID
#                                         #   %l = session line number
#                                         #   %s = session start timestamp
#                                         #   %v = virtual transaction ID
#                                         #   %x = transaction ID (0 if none)
#                                         #   %q = stop here in non-session
#                                         #       processes
#                                         #   %% = '%'
#                                         # e.g. '<%u%%%d> '
# log_lock_waits = off
#log_statement = 'none'
#log_replication_commands = off
#log_temp_files = -1                  # log temporary files equal or larger
#                                         # than the specified size in kilobytes;
#                                         # -1 disables, 0 logs all temp files
log_timezone = 'Europe/Paris'
.../...
.../...
#-----
# STATISTICS
#-----
```

```
# - Query and Index Statistics Collector -

#track_activities = on
#track_counts = on
#track_io_timing = off
#track_functions = none              # none, pl, all
#track_activity_query_size = 1024    # (change requires restart)
#stats_temp_directory = 'pg_stat_tmp'
.../...
.../...
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
#-----
# AUTOVACUUM
#-----

#autovacuum = on                                # Enable autovacuum subprocess? 'on'
# requires track_counts to also be on.
# -1 disables, 0 logs all actions and
# their durations, > 0 logs only
# actions running at least this number
# of milliseconds.
#autovacuum_min_duration = -1                  # max number of autovacuum subprocesses
# (change requires restart)
#autovacuum_max_workers = 3                      # time between autovacuum runs
#autovacuum_naptime = 1min                       # min number of row updates before
#autovacuum_vacuum_threshold = 50                 # vacuum
#autovacuum_analyze_threshold = 50                # min number of row updates before
#autovacuum_vacuum_scale_factor = 0.2             # analyze
#autovacuum_analyze_scale_factor = 0.1            # fraction of table size before vacuum
.../...
#-----
# CLIENT CONNECTION DEFAULTS
#-----

# - Statement Behavior -
#search_path = '"$user", public'                 # schema names
#row_security = on
#default_tablespace = ''                         # a tablespace name, '' uses the default
#temp_tablespaces = ''                           # a list of tablespace names, '' uses
#                                             # only default tablespace
#check_function_bodies = on
#default_transaction_isolation = 'read committed'
.../...
.../...
# - Locale and Formatting -
datestyle = 'iso, dmy'
#intervalstyle = 'postgres'
timezone = 'Europe/Paris'
.../...
.../...
# These settings are initialized by initdb, but they can be changed.
lc_messages = 'fr_FR.UTF-8'                     # locale for system error message
#                                             # strings
lc_monetary = 'fr_FR.UTF-8'                     # locale for monetary formatting
lc_numeric = 'fr_FR.UTF-8'                       # locale for number formatting
lc_time = 'fr_FR.UTF-8'                          # locale for time formatting

# default configuration for text search
default_text_search_config = 'pg_catalog.french'
.../...
.../...
#-----
# LOCK MANAGEMENT
#-----

#deadlock_timeout = 1s                           # min 10
#max_locks_per_transaction = 64                 # (change requires restart)
.../...
.../...
```

# Mise en œuvre d'un serveur de données PostgreSQL

## B. Principaux paramètres

### ▪ **listen\_addresses**

Spécifie la (les) adresse(s) TCP/IP sur laquelle le serveur PostgreSQL écoute les demandes de connexion des clients. Si la liste est vide, seule les connexions via les sockets Unix seront acceptées (connexions locales). Si la valeur \* est positionnée (correspond à toutes les interfaces IP disponibles), le serveur accepte des connexions en provenance du réseau. Valeur par défaut : **localhost**.

### ▪ **max\_connections**

Indique le nombre maximum de connexions concurrentes au serveur de base de données. La valeur par défaut typique est de **100** connexions. L'augmentation de ce paramètre peut obliger PostgreSQL à réclamer plus de mémoire partagée System V ou de sémaphores que ne permet pas la configuration par défaut du système d'exploitation. Voir la Section 18.4.1, « Mémoire partagée et sémaphore » pour plus d'informations sur la façon d'ajuster ces paramètres, si nécessaire.

### ▪ **port**

Le port TCP sur lequel le serveur écoute ; **5432** par défaut. Le même numéro de port est utilisé pour toutes les adresses IP que le serveur écoute. Ce paramètre ne peut être configuré qu'au lancement du serveur.

### ▪ **shared\_buffers**

Spécifie le nombre de buffers mémoire partagés utilisables par le serveur. La valeur par défaut en général est **128MB** (Shared\_buffers minimum 128KB). Des valeurs significativement plus importantes sont généralement nécessaires pour de bonnes performances. Plusieurs dizaines de mégaoctets sont recommandées pour des installations en production (environ  $\frac{1}{4}$  de RAM pour un serveur dédié). Ce paramètre ne peut être configuré qu'au lancement du serveur.

### ▪ **work\_mem**

Spécifie la taille de l'espace mémoire individuel utilisable par chaque processus pour les tris et hachages avant de swapper sur des fichiers disques temporaires. La valeur par défaut est **4MB**. Il est possible d'augmenter cette valeur pour éviter de swapper sur disque. Mais attention, chaque utilisateur peut, suivant la complexité des requêtes, être amené à consommer plusieurs fois la taille indiquée par ce paramètre. Il est à mettre en adéquation avec le nombre de clients maximum (**max\_connections**). Des tris sont générés par des requêtes comme « select... order by... », des jointures (merge join, hash join).

# Mise en œuvre d'un serveur de données PostgreSQL

---

- **maintenance\_work\_mem**

Spécifie la taille de l'espace mémoire maximum réservé aux opérations de maintenance (VACUUM, création d'index et création de clés étrangères). La valeur par défaut est **64MB** (min 1 MB).

- **min\_wal\_size** (le paramètre `checkpoint_segments` n'existe plus la version 9.5)

Tant que l'occupation disque reste sous la valeur de ce paramètre, les anciens fichiers WAL sont toujours recyclés pour une utilisation future lors des checkpoints, plutôt que supprimés. Ceci peut être utile pour s'assurer qu'un espace fichiers WAL suffisant est réservé pour faire face à des pics dans l'usage des WAL, par exemple lorsque d'importants travaux batch sont lancés. La valeur par défaut est **80 Mo**. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

- **max\_wal\_size** (le paramètre `checkpoint_segments` n'existe plus la version 9.5)

Taille maximale de l'augmentation des fichiers WAL entre deux checkpoints automatiques des WAL. C'est une limite souple ; la taille des fichiers WAL peut excéder `max_wal_size` sous certaines circonstances, comme une surcharge du serveur, une commande `archive_command` qui échoue, ou une configuration haute pour `wal_keep_segments`. La valeur par défaut est **1 Go**. Augmenter ce paramètre peut augmenter le temps nécessaire pour le rejet suite à un crash. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

- **checkpoint\_timeout**

Temps maximum entre deux points de vérification automatique des WAL, en secondes. La valeur par défaut est de **5 minutes**. Augmenter ce paramètre peut accroître le temps nécessaire à une récupération après un arrêt brutal.

- **lc\_messages    lc\_monetary    lc\_numeric    lc\_time**

Permet de gérer les spécificités linguistiques comme la langue pour l'affichage des messages, le signe de la monnaie, les caractères formatant les nombres (entiers / décimaux) et le format de l'affichage de la date et l'heure.

- **synchronous\_commit**

- **checkpoint\_completion\_target**

- **archive\_mode**

- **log\_destination    log\_filename            log\_...**

- **autovacuum**

- **statement\_timeout**

- ...

# Mise en œuvre d'un serveur de données PostgreSQL

## Exemple de démarrage de PostgreSQL avec un fichier postgresql.conf personnalisé :

Connections via TCP/IP autorisées, 200 connexions maximum, port d'écoute 5432, 300 MB de shared buffers, une zone de tri de 10 MB, un espace mémoire réservé aux opérations de maintenance de 500 MB maximum, un checkpoint 7 minutes, spécificités linguistiques de la France.

```
[postgres@CentOS764b ~]$ export PGDATA=/home/postgres/data
[postgres@CentOS764b ~]$ echo $PGDATA
/home/postgres/data
[postgres@CentOS764b ~]$ pwd
/home/postgres
[postgres@CentOS764b ~]$ cat data/postgresql.conf
#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -
listen_addresses = '*'                                # what IP address(es) to listen on;
                                                       # comma-separated list of addresses;
                                                       # defaults to 'localhost', '*' = all
                                                       # (change requires restart)
port = 5432                                         # (change requires restart)
max_connections = 200                                 # (change requires restart)
# Note: Increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).
#-----
# RESOURCE USAGE (except WAL)
#-----
# - Memory -
shared_buffers = 300MB                               # min 128kB
                                                       # (change requires restart)
#temp_buffers = 8MB                                  # min 800kB
work_mem = 10MB                                    # min 64kB
maintenance_work_mem = 500MB                         # min 1MB
#-----
# WRITE AHEAD LOG
#-----
# - Checkpoints -
checkpoint_timeout = 7min                           # range 30s-1h
checkpoint_completion_target = 0.5                  # checkpoint target duration, 0.0 - 1.0
checkpoint_warning = 30s                            # 0 disables
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
lc_messages = 'fr_FR.UTF-8'                                # locale for system error message
lc_monetary = 'fr_FR.UTF-8'                                # strings
lc_numeric = 'fr_FR.UTF-8'                                 # locale for monetary formatting
lc_time = 'fr_FR.UTF-8'                                   # locale for number formatting
# default configuration for text search                  # locale for time formatting
#_default_text_search_config = 'pg_catalog.french'

[postgres@CentOS764b ~]$ pg_ctl start -l /home/postgres/serverlog
waiting for server to start.... done
server started
[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: server is running (PID: 4485)
/usr/local/pgsql/bin/postgres
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# select name,setting from pg_settings;
          name           |           setting
-----+-----
 .../...
archive_command          | (disabled)
archive_mode              | off
 .../...
checkpoint_timeout        | 420
 .../...
fsync                     | on
 .../...
lc_collate               | fr_FR.UTF-8
lc_ctype                  | fr_FR.UTF-8
lc_messages               | fr_FR.UTF-8
lc_monetary                | fr_FR.UTF-8
lc_numeric                 | fr_FR.UTF-8
lc_time                   | fr_FR.UTF-8
listen_addresses          | *
 .../...
maintenance_work_mem     | 512000
max_connections            | 200
 .../...
port                      | 5432
 .../...
shared_buffers             | 38400
 .../...
work_mem                  | 10240
 .../...

(289 rows)

template1=#

```

# Mise en œuvre d'un serveur de données PostgreSQL

## C. Depuis la version 9.4

La commande SQL « **ALTER SYSTEM** » est utilisée pour modifier des paramètres de configuration du serveur pour l'instance complète.

Cette méthode peut être plus pratique que la méthode traditionnelle qui consiste à modifier manuellement le fichier **postgresql.conf**.

**ALTER SYSTEM** écrit la nouvelle valeur du paramètre indiqué dans le nouveau fichier **postgresql.auto.conf** (qui se trouve dans PGDATA), qui est lu en plus du fichier **postgresql.conf**.

Configurer un paramètre à DEFAULT avec **ALTER SYSTEM**, ou utiliser la variante **RESET**, supprime le paramètre du fichier **postgresql.auto.conf**.

Utilisez **RESET ALL** pour supprimer tous les paramètres configurés dans ce fichier.

Suivant le paramètre (dynamique ou statique) modifié par **ALTER SYSTEM**, un reload ou restart du serveur sera nécessaire.

### Exemple :

```
[postgres@CentOS764b ~]$ psql template1
psql (10.0)
Type "help" for help.

template1=# show checkpoint_timeout;
checkpoint_timeout
-----
7min
(1 row)

template1=# \! cat /home/postgres/data/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
template1=
template1=# alter system set checkpoint_timeout = '5min';
ALTER SYSTEM
template1=# \! cat /home/postgres/data/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
checkpoint_timeout = '5min'
template1=
template1=# show checkpoint_timeout;
checkpoint_timeout
-----
7min
(1 row)
template1=# \q
[postgres@CentOS764b ~]$
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
[postgres@CentOS764b ~]$ pg_ctl reload
server signaled
[postgres@CentOS764b ~]$ psql template1
psql (10.0)
Type "help" for help.

template1=# show checkpoint_timeout;
checkpoint_timeout
-----
5min
(1 row)

template1=# alter system set checkpoint_timeout to default;
ALTER SYSTEM
template1=# \! cat /home/postgres/data/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
template1=# show checkpoint_timeout;
checkpoint_timeout
-----
5min
(1 row)

template1=# \q
[postgres@CentOS764b ~]$ pg_ctl reload
server signaled
[postgres@CentOS764b ~]$ psql template1
psql (10.0)
Type "help" for help.

template1=# show checkpoint_timeout;
checkpoint_timeout
-----
7min
(1 row)

template1=#

```

# Mise en œuvre d'un serveur de données PostgreSQL

## 4. Démarrage et arrêt automatique du serveur

Il est possible d'utiliser un script de style SysV pour PostgreSQL. Le script SysV de PostgreSQL permet de contrôler PostgreSQL à travers l'utilisation du système des niveaux d'exécution (runlevels) System V.

Ce script est un habillage de la commande pg\_ctl prévu pour être exécuté par root plutôt que par l'administrateur de PostgreSQL (postgres). Ce script simplifie les procédures de démarrage / arrêt de serveur PostgreSQL.

Ce script permet de démarrer / arrêter PostgreSQL via la commande « **service** » ou/et démarrer / arrêter PostgreSQL **automatiquement** au chargement / arrêt du système Linux.

Un script est livré avec PostgreSQL et fonctionne avec la plupart des distributions Linux (Red Hat, Mandrake, SuSe...). Ce script se trouve dans **/usr/local/src/postgresql-11.n/contrib/start-scripts** et se nomme **linux**.

Ce script est en format texte et peut être éventuellement personnalisé à vos besoins.

Copier ce script dans **/etc/rc.d/init.d/postgresql** et le rendre exécutable.

### Exemple :

```
[root@CentOS764b ~]# cd /usr/local/src/postgresql-11.1/contrib/start-scripts/
[root@CentOS764b start-scripts]# ll
total 8
-rw-r--r--. 1 1107 1107 1467 7 nov. 00:56 freebsd
-rw-r--r--. 1 1107 1107 3552 7 nov. 00:56 linux
drwxrwxrwx. 2 1107 1107 81 7 nov. 00:59 macos
[root@CentOS764b start-scripts]# cp linux /etc/rc.d/init.d/postgresql11_PROD
[root@CentOS764b start-scripts]# chmod a+x /etc/rc.d/init.d/postgresql11_PROD
[root@CentOS764b start-scripts]#
```

# Mise en œuvre d'un serveur de données PostgreSQL

Adapter éventuellement le script à votre environnement :

```
#!/bin/sh
# chkconfig: 2345 98 02
# description: PostgreSQL RDBMS
# This is an example of a start/stop script for SysV-style init, such
# as is used on Linux systems. You should edit some of the variables
# and maybe the 'echo' commands.
#
# Place this file at /etc/init.d/postgresql (or
# /etc/rc.d/init.d/postgresql) and make symlinks to
# /etc/rc.d/rc0.d/K02postgresql
# /etc/rc.d/rc1.d/K02postgresql
# /etc/rc.d/rc2.d/K02postgresql
# /etc/rc.d/rc3.d/S98postgresql
# /etc/rc.d/rc4.d/S98postgresql
# /etc/rc.d/rc5.d/S98postgresql
# Or, if you have chkconfig, simply:
# chkconfig --add postgresql
#
# Proper init scripts on Linux systems normally require setting lock
# and pid files under /var/run as well as reacting to network
# settings, so you should treat this with care.
# Original author: Ryan Kirkpatrick <pgsql@rkirkpat.net>
# contrib/start-scripts/linux
## EDIT FROM HERE

→ # Installation prefix
prefix=/usr/local/pgsql

→ # Data directory
PGDATA="/home/postgres/data"

→ # Who to run the postmaster as, usually "postgres". (NOT "root")
PGUSER=postgres

→ # Where to keep a log file
PGLOG="$PGDATA/serverlog"

# It's often a good idea to protect the postmaster from being killed by the
# OOM killer (which will tend to preferentially kill the postmaster because
# of the way it accounts for shared memory). To do that, uncomment these
# three lines:
#PG_OOM_ADJUST_FILE=/proc/self/oom_score_adj
#PG_MASTER_OOM_SCORE_ADJ=-1000
#PG_CHILD_OOM_SCORE_ADJ=0
# Older Linux kernels may not have /proc/self/oom_score_adj, but instead
# /proc/self/oom_adj, which works similarly except for having a different
# range of scores. For such a system, uncomment these three lines instead:
#PG_OOM_ADJUST_FILE=/proc/self/oom_adj
#PG_MASTER_OOM_SCORE_ADJ=-17
#PG_CHILD_OOM_SCORE_ADJ=0

## STOP EDITING HERE
```

# Mise en œuvre d'un serveur de données PostgreSQL

```
# The path that is to be used for the script
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# What to use to start up the postmaster. (If you want the script to wait
# until the server has started, you could use "pg_ctl start" here.)
DAEMON="$prefix/bin/postmaster"
# What to use to shut down the postmaster
PGCTL="$prefix/bin/pg_ctl"
set -e
# Only start if we can find the postmaster.
test -x $DAEMON ||
{
    echo "$DAEMON not found"
    if [ "$1" = "stop" ]
    then exit 0
    else exit 5
    fi
}
# If we want to tell child processes to adjust their OOM scores, set up the
# necessary environment variables. Can't just export them through the "su".
if [ -e "$PG_OOM_ADJUST_FILE" -a -n "$PG_CHILD_OOM_SCORE_ADJ" ]
then
    DAEMON_ENV="PG_OOM_ADJUST_FILE=$PG_OOM_ADJUST_FILE"
    PG_OOM_ADJUST_VALUE=$PG_CHILD_OOM_SCORE_ADJ
fi
# Parse command line parameters.
case $1 in
    start)
        echo -n "Starting PostgreSQL: "
        test -e "$PG_OOM_ADJUST_FILE" && echo "$PG_MASTER_OOM_SCORE_ADJ" >
"$PG_OOM_ADJUST_FILE"
        su - $PGUSER -c "$DAEMON_ENV $DAEMON -D '$PGDATA' >>$PGLOG 2>&1 &""
        echo "ok"
        ;;
    stop)
        echo -n "Stopping PostgreSQL: "
        su - $PGUSER -c "$PGCTL stop -D '$PGDATA' -s"
        echo "ok"
        ;;
    restart)
        echo -n "Restarting PostgreSQL: "
        su - $PGUSER -c "$PGCTL stop -D '$PGDATA' -s"
        test -e "$PG_OOM_ADJUST_FILE" && echo "$PG_MASTER_OOM_SCORE_ADJ" >
"$PG_OOM_ADJUST_FILE"
        su - $PGUSER -c "$DAEMON_ENV $DAEMON -D '$PGDATA' >>$PGLOG 2>&1 &""
        echo "ok"
        ;;
    reload)
        echo -n "Reload PostgreSQL: "
        su - $PGUSER -c "$PGCTL reload -D '$PGDATA' -s"
        echo "ok"
        ;;
    status)
        su - $PGUSER -c "$PGCTL status -D '$PGDATA'"
        ;;
    *)
        # Print help
        echo "Usage: $0 {start|stop|restart|reload|status}" 1>&2
        exit 1
        ;;
esac
exit 0
```

# Mise en œuvre d'un serveur de données PostgreSQL

Si vous ne désirez pas utiliser ce script pour démarrer/arrêter automatiquement PostgreSQL au chargement/arrêt du système Linux, il n'y a pas d'autres manipulations à faire.

Vous pouvez maintenant utiliser la commande de gestion des services pour démarrer/arrêter manuellement PostgreSQL.

**Exemple :**

```
[root@CentOS764b start-scripts]# service postgresql11_PROD start
Starting PostgreSQL: ok
[root@CentOS764b start-scripts]# service postgresql11_PROD status
pg_ctl: server is running (PID: 4886)
/usr/local/pgsql/bin/postgres "-D" "/home/pgsql/data"
[root@CentOS764b start-scripts]# service postgresql11_PROD stop
Stopping PostgreSQL: ok
[root@CentOS764b start-scripts]# service postgresql11_PROD status
pg_ctl: no server running
[root@CentOS764b start-scripts]#
```

# Mise en œuvre d'un serveur de données PostgreSQL

Si vous désirez utiliser ce script pour démarrer/arrêter **automatiquement** PostgreSQL au chargement/arrêt du système Linux, vous devez utiliser la commande **systemctl** pour ajouter le service postgresql.

## Exemple :

```
[root@CentOS764b start-scripts]# systemctl enable postgresql11_PROD
postgresql11_PROD.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig postgresql11_PROD on

[root@CentOS764b start-scripts]# systemctl start postgresql11_PROD
[root@CentOS764b ~]# systemctl status postgresql11_PROD
● postgresql11_PROD.service - SYSV: PostgreSQL RDBMS
   Loaded: loaded (/etc/rc.d/init.d/postgresql11_PROD)
   Active: active (exited) since mar. 2018-11-27 10:24:56 CET; 1min 24s ago
     Docs: man:systemd-sysv-generator(8)

      Process: 1296 ExecStart=/etc/rc.d/init.d/postgresql11_PROD start (code=exited,
status=0/SUCCESS)

nov. 27 10:24:55 CentOS764b systemd[1]: Starting SYSV: PostgreSQL RDBMS...
nov. 27 10:24:55 CentOS764b su[1302]: (to postgres) root on none
nov. 27 10:24:56 CentOS764b postgresql11_PROD[1296]: Starting PostgreSQL: ok
nov. 27 10:24:56 CentOS764b systemd[1]: Started SYSV: PostgreSQL RDBMS.

[root@CentOS764b ~]#
```

## Remarque :

Sous Windows, il est possible d'utiliser le gestionnaire des services pour automatiser le démarrage / arrêt d'un serveur PostgreSQL.

# Mise en œuvre d'un serveur de données PostgreSQL

---



## 4. CRÉATION D'UNE BASE

---

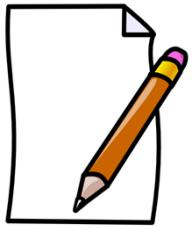
### Objectif

A la fin de ce module, vous serez capable de créer, modifier, supprimer une base et gérer les tablespaces dans un serveur PostgreSQL :

- Introduction
- Création d'une base de données avec « `create database` » et « `createdb` »
- Informations sur les bases de données
- Modification d'une base
- Suppression d'une base
- Création, modification, suppression de tablespaces

# Création d'une base

---



# Création d'une base

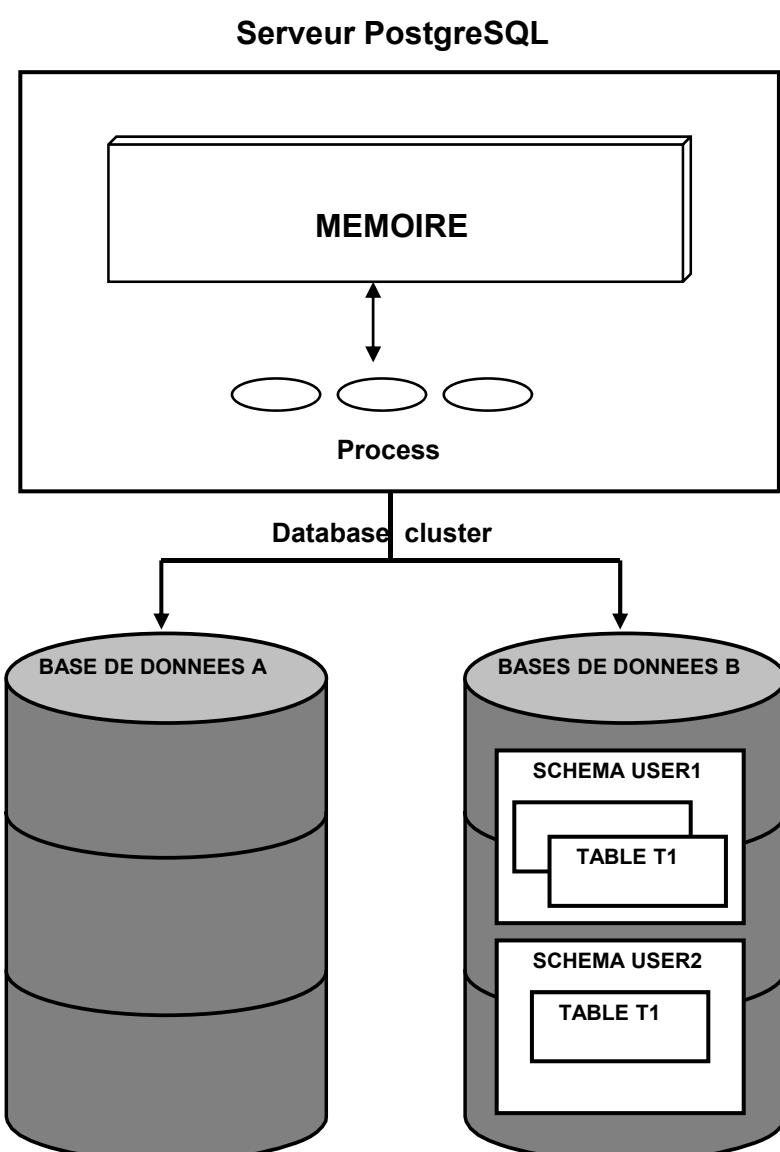
## 1. Introduction

Une base de données est une collection nommée d'objets.

Chaque objet (table, index, fonction...) appartient à une et une seule base. Des catalogues systèmes (pg\_database, pg\_group, pg\_stat...) appartiennent à un groupe de base de données.

Plus précisément, une base de données est un ensemble de schémas qui contient les objets (tables, index...). Un schéma est un espace de nom dans une base de données permettant de rassembler logiquement les objets d'un utilisateur ou les objets d'une application.

Du plus grand vers le plus petit, nous avons donc : une instance, une base de données, un schéma, un objet (table ou autre).



# Création d'une base

---

Les bases de données sont physiquement séparées dans des répertoires différents. Si un serveur PostgreSQL héberge des utilisateurs ou projets qui doivent être séparés ou n'ont rien à voir ensemble, il est préférable de les mettre dans des bases de données différentes, voir des instances différentes.

Si les projets ou les utilisateurs sont en relation et doivent utiliser mutuellement des ressources, ils peuvent être mis dans la même base de données mais éventuellement dans des schémas différents.

# Création d'une base

---

## 2. Crédit d'une base de données

### A. Introduction

Pour créer une base de données, l'instance PostgreSQL doit être démarrée. A la création d'une instance PostgreSQL (initdb), 3 bases de données sont créées. Les bases modèles **template0** et **template1** vont être utilisées pour créer vos bases de données. La base de données **postgres** est une base par défaut utilisée par les outils. Le serveur n'a pas besoin que la base **postgres** existe mais beaucoup d'outils ou programmes externes supposent son existence.

La base modèle **template0** est un modèle vierge de toutes modifications (non modifiable, non connectable). La base modèle **template1** (à l'origine identique au modèle template0) peut être modifiée en y ajoutant par exemple des fonctions, des objets d'administration (tables, vues, séquences ...) spécifiques aux bases de votre site.

Vous ne pouvez vous connecter qu'aux bases template1 et postgres.

### B. Création d'une base de données

Il existe 2 méthodes pour créer une base de données :

- la commande SQL **CREATE DATABASE**
- la commande Shell **createdb**

Pour utiliser ces commandes, il faut posséder le privilège **createdb**. Vous vérifiez ce privilège dans la table **pg\_shadow** si vous êtes super utilisateur (postgres) ou dans la vue **pg\_user** si vous êtes connecté avec un utilisateur standard. La colonne **usecreatedb** de ces vues vous renseignera (t = true = privilège validé).

# Création d'une base

- Exemple :

```
[postgres@CentOS764b ~]$ psql test -U paul
psql (11.1)
Type "help" for help.

test=> select * from pg_shadow;

ERROR: permission denied for view pg_shadow

test=> select usename, usecreatedb, passwd from pg_user;

usename | usecreatedb | passwd
-----+-----+-----
postgres | t          | *****
paul    | f          | *****
(2 rows)

test=> create database test1;
ERROR: permission denied to create database
test=> \c template1 postgres
You are now connected to database "template1" as user "postgres".
template1=# create database test1;
CREATE DATABASE
template1=#
```

- Le créateur de la base de données devient le propriétaire de la nouvelle base (ici test1). Seul le propriétaire (ici postgres) pourra la supprimer (supprimant tous les objets stockés dans cette base et pouvant appartenir à plusieurs propriétaires).

# Création d'une base

## C. Création d'une base avec CREATE DATABASE

Vous devez être connecté à une des bases de données du serveur pour pouvoir exécuter la commande CREATE DATABASE.

Les bases de données **template1** et **postgres** créées à l'initialisation du serveur (initdb) peuvent être utilisées pour créer votre réelle première base. Vous pouvez vous connecter à la base template1 et ensuite créer votre base. Une fois créée, vous pourrez vous connecter directement à cette nouvelle base.

Quand vous créez une nouvelle base, vous clonez par défaut la base **template1**. Toute modification effectuée dans cette base, sera donc propagée dans les bases créées par la suite. Ce qui veut dire qu'il ne faut pas utiliser la base template1 comme base de travail réelle.

Syntaxe de la commande SQL « CREATE DATABASE » :

La syntaxe est récupérable sous psql (\help create database) ou dans la documentation « Reference ».

```
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# \h create database
Command:      CREATE DATABASE
Description:  create a new database
Syntax:
CREATE DATABASE name
  [ [ WITH ] [ OWNER [=] user_name ]
    [ TEMPLATE [=] template ]
    [ ENCODING [=] encoding ]
    [ LC_COLLATE [=] lc_collate ]
    [ LC_CTYPE [=] lc_ctype ]
    [ TABLESPACE [=] tablespace_name ]
    [ ALLOW_CONNECTIONS [=] allowconn ]
    [ CONNECTION LIMIT [=] connlimit ]
    [ IS_TEMPLATE [=] istemplate ] ]

template1=#
```

# Création d'une base

---

## D. Paramètres

- ***name***

Nom de la base créée.

- **OWNER = *dbowner***

Nom du propriétaire de la nouvelle base de données. Si omis, le propriétaire sera l'utilisateur qui a exécuté la commande.

- **TABLESPACE = '*tablespacename*'**

Nom du tablespace qui sera associé à cette nouvelle base ou DEFAULT pour utiliser le tablespace de la base prise en modèle. Ce tablespace sera le tablespace par défaut utilisé pour les objets créés dans cette base.

- **TEMPLATE = *template***

Nom du modèle utilisé pour créer la nouvelle base. Si omis, le modèle par défaut sera utilisé (template1).

- **ENCODING = '*encoding*'**

Jeux de caractères utilisés pour la nouvelle base. La liste des jeux de caractères supportée se trouve dans le chapitre 23 « Localization » de la documentation.

# Création d'une base

Exemple :

```
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# \l
                                         List of databases
  Name   |  Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
template0 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
template1 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
(3 rows)

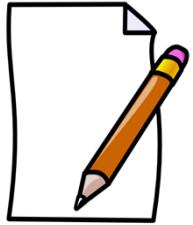
template1=# create role paul login password 'paul';
CREATE ROLE
template1=# create database test1 with owner=paul;
CREATE DATABASE
template1=# create database test2 with template=test1;
CREATE DATABASE
template1=# \q
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql -l
                                         List of databases
  Name   |  Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
template0 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
template1 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
test1    | paul     | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
test2    | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
(5 rows)

[postgres@CentOS764b ~]$ createdb -U postgres test3
[postgres@CentOS764b ~]$ psql -l
                                         List of databases
  Name   |  Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
template0 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
template1 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
test1    | paul     | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
test2    | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
test3    | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | 
(6 rows)

[postgres@CentOS764b ~]$
```

# Création d'une base

---



# Création d'une base

## F. Informations sur les bases

```
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# select
template1-# datname,datdba,username,datistemplate,datallowconn,dattablespace,spcname
template1-#           from pg_database,pg_user,pg_tablespace
template1-# where datdba=usesysid and dattablespace=pg_tablespace.oid;

   datname  |  datdba  |  usename  |  datistemplate  |  datallowconn  |  dattablespace  |  spcname
-----+-----+-----+-----+-----+-----+-----+
  postgres  |    10   |  postgres  | f            | t            |                | 1663  | pg_default
template1 |    10   |  postgres  | t            | t            |                | 1663  | pg_default
template0 |    10   |  postgres  | t            | f            |                | 1663  | pg_default
test2     |    10   |  postgres  | f            | t            |                | 1663  | pg_default
test3     |    10   |  postgres  | f            | t            |                | 1663  | pg_default
test1     |  16387 |  paul      | f            | t            |                | 1663  | pg_default
(6 rows)
```

# Création d'une base

## G. Modification d'une base

Comme nous l'avons vu dans le module « Mise au point de l'environnement d'une instance PostgreSQL », une instance PostgreSQL dispose d'un grand nombre de paramètres de configuration. Vous pouvez définir des valeurs par défaut spécifiques à une base avec la commande **ALTER DATABASE**.

Ces modifications de paramètres d'environnement sont sauvegardées et prises en compte pour les connexions qui suivent. Les modifications de paramètres par alter database sont prioritaires par rapport à ce qui a pu être défini au démarrage du serveur ou dans le fichier de configuration postgresql.conf. Les utilisateurs peuvent encore modifier la valeur de certains paramètres au niveau de leur session (set session param=valeur).

Les modifications effectuées avec la commande ALTER DATABASE restent valables après un arrêt et redémarrage du serveur.

Pour reprendre les valeurs initiales, utilisez la commande « alter database *dbname* reset *varname* ».

Syntaxe de la commande :

```
template1=# \h alter database
Command:      ALTER DATABASE
Description: change a database
Syntax:
ALTER DATABASE name [ [ WITH ] option [ ... ] ]

where option can be:

ALLOW_CONNECTIONS allowconn
CONNECTION LIMIT connlimit
IS_TEMPLATE istemplate

ALTER DATABASE name RENAME TO new_name

ALTER DATABASE name OWNER TO { new_owner | CURRENT_USER | SESSION_USER }

ALTER DATABASE name SET TABLESPACE new_tablespace

ALTER DATABASE name SET configuration_parameter { TO | = } { value | DEFAULT
}
ALTER DATABASE name SET configuration_parameter FROM CURRENT
ALTER DATABASE name RESET configuration_parameter
ALTER DATABASE name RESET ALL

template1=#

```

# Création d'une base

---

## H. Suppression d'une base

Une base de données peut être détruite de 2 manières différentes :

- avec la commande SQL « **DROP DATABASE *name*** »
- la commande Shell « **dropdb *dbname*** »

Seul le propriétaire de la base (celui qui l'a créée) ou un super utilisateur peut supprimer une base.  
Tous les objets de cette base seront supprimés.

Vous ne pouvez pas supprimer la base sur laquelle vous êtes connecté.

La commande de suppression d'une base doit être faite en étant connecté à une autre base.

# Création d'une base

## Exemple :

```
[postgres@CentOS764b ~]$ psql -l
                                         List of databases
   Name    |  Owner   | Encoding | Collate | Ctype | Access privileges
---+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
template0 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres=CTc/postgres
          |          |          |          |          |          |
template1 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          |          |
test1    | paul     | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
test2    | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres=CTc/postgres
test3    | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres=CTc/postgres
(6 rows)

[postgres@CentOS764b ~]$ psql test1 -U paul
psql (11.1)
Type "help" for help.

test1=> drop database test2;
ERROR: must be owner of database test2
test1=> drop database test1;
ERROR: cannot drop the currently open database
test1=> \c template1 paul
You are now connected to database "template1" as user "paul".
template1=> drop database test1;
DROP DATABASE
template1=> \c template1 postgres
You are now connected to database "template1" as user "postgres".
template1=# drop database test2;
DROP DATABASE
template1=# \q
[postgres@CentOS764b ~]$ dropdb test3
[postgres@CentOS764b ~]$ psql -l
                                         List of databases
   Name    |  Owner   | Encoding | Collate | Ctype | Access privileges
---+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
template0 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres=CTc/postgres
          |          |          |          |          |          |
template1 | postgres | UTF8    | fr_FR.UTF-8 | fr_FR.UTF-8 | =c/postgres      +
          |          |          |          |          |          |
(3 rows)
```

[postgres@CentOS764b ~]\$

# Création d'une base

---

## 3. Gestion des tablespaces

### A. Introduction

Les utilisateurs d'une base de données manipulent des objets logiques (tables, index...) sans connaître leur localisation physique dans les répertoires et les fichiers représentant respectivement les bases de données et les objets.

Un tablespace définit physiquement l'endroit où sont stockées les données. Un tablespace contient des objets (bases de données, tables, index...).

Un tablespace correspond à un répertoire (au niveau O.S) dans lequel seront stockés les fichiers représentant des objets (tables, index) ou les répertoires représentant les bases de données. Les tablespaces permettent aux super-utilisateurs de choisir l'emplacement dans le système de fichiers où seront créés les fichiers représentant les objets ou les répertoires représentant les bases de données d'un serveur PostgreSQL.

Par défaut, 2 tablespaces sont créés automatiquement à l'initialisation du serveur (`initdb`) dans le répertoire des données (**PGDATA**) :

- **pg\_default** : tablespace par défaut des bases de données template1 et template0 (et donc le tablespace par défaut des autres bases de données sauf si la clause TABLESPACE est utilisée au CREATE DATABASE). Physiquement, cela correspond au répertoire **\$PGDATA/base**.
- **global** : utilisé pour les catalogues systèmes. C'est-à-dire les objets globaux à toutes les bases de données de l'instance. Physiquement, cela correspond au répertoire **\$PGDATA/global**.

# Création d'une base

## B. Création / modification / suppression d'un tablespace

La création d'un tablespace se fait avec la commande SQL « CREATE TABLESPACE ».

```
template1=# \h create tablespace

Command:      CREATE TABLESPACE
Description: define a new tablespace
Syntax:
CREATE TABLESPACE tablespace_name
    [ OWNER { new_owner | CURRENT_USER | SESSION_USER } ]
    LOCATION 'directory'
    [ WITH ( tablespace_option = value [, ...] ) ]
```

## C. Paramètres

### ▪ **tablespace\_name**

Nom du tablespace à créer (ne doit pas commencer par pg\_).

### ▪ **OWNER**

Nom du propriétaire du tablespace. Si omis, le propriétaire est le créateur. Seul un super-utilisateur peut créer un tablespace mais peut, avec ce paramètre rendre propriétaire un utilisateur standard.

### ▪ **LOCATION**

Chemin absolu du répertoire qui sera utilisé par le tablespace. Ce répertoire doit exister, être vide et appartenir à l'utilisateur O.S qui a initialisé le serveur (postgres).

# Création d'une base

---

Le super-utilisateur doit créer le tablespace et ensuite donner aux utilisateurs standards le droit de créer des objets (GRANT CREATE ON TABLESPACE *tablespacename* TO...) dans ce tablespace.

Les ayant droits pourront créer dans ce tablespace des tables, des index et des bases de données entières en citant le nom du tablespace dans la commande de création de l'objet concerné (CREATE TABLE T1 (...) TABLESPACE *tablespacename*);.

La vue **pg\_tablespace** et la commande spécifique de psql « \db+ » permettent de visualiser les tablespaces existants.

Lors de la création d'un nouveau tablespace, PostgreSQL crée un **fichier lien symbolique** dans \$PGDATA/pg\_tblspc qui pointe vers le répertoire utilisé pour ce nouveau tablespace.

## Exemples :

```
[postgres@CentOS764b ~] $ mkdir /home/postgres/tbsp1
[postgres@CentOS764b ~] $ psql template1
psql (11.1)
Type "help" for help.

template1=# create tablespace tbsp1 location
' /home/postgres/tbsp1';
CREATE TABLESPACE
template1=#
```

# Création d'une base

```
template1=# \db
              List of tablespaces
   Name    | Owner    |      Location
-----+-----+-----+
pg_default | postgres |
pg_global  | postgres |
tbsp1       | postgres | /home/postgres/tbsp1
(3 rows)

template1=# select * from pg_tablespace;
      spcname | spcowner | spcacl | spcoptions
-----+-----+-----+-----+
pg_default |        10 |         |
pg_global  |        10 |         |
tbsp1       |        10 |         |
(3 rows)

template1=# select oid,* from pg_tablespace;
     oid | spcname | spcowner | spcacl | spcoptions
-----+-----+-----+-----+
  1663 | pg_default |        10 |         |
  1664 | pg_global  |        10 |         |
 16391 | tbsp1       |        10 |         |
(3 rows)

template1=# \! ls -l /home/postgres/data/pg_tblspc
total 0
lrwxrwxrwx. 1 postgres postgres 20 27 nov. 11:24 16391 -> /home/postgres/tbsp1
template1# create database test3 tablespace = tbsp1;
CREATE DATABASE
template1#
```

# Création d'une base

La commande SQL “ALTER TABLESPACE” permet :

- De renommer un tablespace
- De changer son propriétaire

La commande SQL “DROP TABLESPACE” permet de supprimer un tablespace.

```
template1=# drop tablespace tbspl;
ERROR:  tablespace "tbspl" is not empty

template1=# select pg_database.oid,datname,dattablespace,spcname
template1-#      from pg_database join pg_tablespace
template1-#          on pg_database.dattablespace=pg_tablespace.oid;

   oid  |  datname  |  dattablespace  |  spcname
-----+-----+-----+-----+
 13284 | postgres  |          1663 | pg_default
     1 | template1 |          1663 | pg_default
 13283 | template0 |          1663 | pg_default
 16392 | test3    |          16391 | tbspl
(4 rows)

template1=# \c template1
You are now connected to database "template1" as user "postgres".

template1=# drop database test3;
DROP DATABASE

template1=# drop tablespace tbspl;
DROP TABLESPACE
template1=#

```

# Création d'une base

---



## 5. AUTHENTIFICATION DES CLIENTS

---

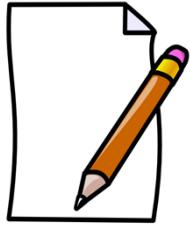
### Objectif

A la fin de ce module, vous serez capable de mettre en place l'authentification des connexions votre serveur PostgreSQL :

- L'authentification des clients
- Le fichier pg\_hba.conf
- Les méthodes d'authentification
- Mise en place d'un fichier pg\_hba.conf personnalisé avec connexions distantes

# Authentification des clients

---



# Authentification des clients

---

## 1. Authentification avec le fichier pg\_hba.conf

Une application cliente se connecte à un serveur de bases de données en fournissant un nom d'utilisateur PostgreSQL et éventuellement un mot de passe.

La connexion à une base de données avec un nom d'utilisateur détermine tout un ensemble de droits d'accès aux objets de la base et de droits d'utilisation de commandes SQL. Il est donc essentiel de contrôler quels sont les utilisateurs qui peuvent se connecter.

### A. Le fichier pg\_hba.conf

L'authentification de « provenance » des clients est faite avec le fichier **pg\_hba.conf** (Host-Based Authentication) créé par défaut dans le répertoire de données PGDATA lors de l'initialisation du serveur PostgreSQL (initdb).

Le fichier **pg\_hba.conf** configure l'accès des clients en fonction des hôtes, définit le type d'authentification utilisé et agit comme un filtre IP pour les demandes de connexion. L'authentification est faite par PostgreSQL avant la connexion à une base de données où ensuite les droits de l'utilisateur s'appliqueront.

L'authentification basée sur les hôtes permet par exemple :

- de restreindre l'accès à une base pour des clients (hôtes) spécifiques ou un groupe d'adresses IP,
- d'indiquer si l'accès s'applique à une seule base, plusieurs bases ou toutes les bases du serveur PostgreSQL,
- ...

Le fichier **pg\_hba.conf** permet donc de préciser qui peut se connecter, à quelle base, à partir de quelle machine (hôte) et jusqu'à quel degrés les utilisateurs doivent prouver leur identité pour accéder aux données.

Le fichier **pg\_hba.conf** est un fichier texte modifiable contenant une suite d'entrées définissant les valeurs utilisées par PostgreSQL pour authentifier un hôte donné. Une seule entrée d'hôte par ligne du fichier. Chaque entrée est composée de plusieurs champs séparés par des espaces. Une ligne de commentaire commence par #.

#### Fichier pg\_hba.conf par défaut :

# Authentification des clients

```
# PostgreSQL Client Authentication Configuration File
# =====
#
# Refer to the "Client Authentication" section in the PostgreSQL
# documentation for a complete description of this file. A short
# synopsis follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access. Records take one of these forms:
#
# local      DATABASE  USER   METHOD  [OPTIONS]
# host       DATABASE  USER   ADDRESS  METHOD  [OPTIONS]
# hostssl    DATABASE  USER   ADDRESS  METHOD  [OPTIONS]
# hostnossal DATABASE  USER   ADDRESS  METHOD  [OPTIONS]
#
# (The uppercase items must be replaced by actual values.)
#
# The first field is the connection type: "local" is a Unix-domain
# socket, "host" is either a plain or SSL-encrypted TCP/IP socket,
# "hostssl" is an SSL-encrypted TCP/IP socket, and "hostnossal" is a
# plain TCP/IP socket.
#
# DATABASE can be "all", "sameuser", "samerole", "replication", a
# database name, or a comma-separated list thereof. The "all"
# keyword does not match "replication". Access to replication
# must be enabled in a separate record (see example below).
#
# USER can be "all", a user name, a group name prefixed with "+", or a
# comma-separated list thereof. In both the DATABASE and USER fields
# you can also write a file name prefixed with "@" to include names
# from a separate file.
#
# ADDRESS specifies the set of hosts the record matches. It can be a
# host name, or it is made up of an IP address and a CIDR mask that is
# an integer (between 0 and 32 (IPv4) or 128 (IPv6) inclusive) that
# specifies the number of significant bits in the mask. A host name
# that starts with a dot (.) matches a suffix of the actual host name.
# Alternatively, you can write an IP address and netmask in separate
# columns to specify the set of hosts. Instead of a CIDR-address, you
# can write "samehost" to match any of the server's own IP addresses,
# or "samenet" to match any address in any subnet that the server is
# directly connected to.
#
# METHOD can be "trust", "reject", "md5", "password", "scram-sha-256",
# "gss", "sspi", "ident", "peer", "pam", "ldap", "radius" or "cert".
# Note that "password" sends passwords in clear text; "md5" or
# "scram-sha-256" are preferred since they send encrypted passwords.
#
# OPTIONS are a set of options for the authentication in the format
# NAME=VALUE. The available options depend on the different
# authentication methods -- refer to the "Client Authentication"
# section in the documentation for a list of which options are
# available for which authentication methods.
#
# Database and user names containing spaces, commas, quotes and other
# special characters must be quoted. Quoting one of the keywords
# "all", "sameuser", "samerole" or "replication" makes the name lose
# its special character, and just match a database or username with
# that name.
```

# Authentification des clients

```
# This file is read on server startup and when the server receives a
# SIGHUP signal. If you edit the file on a running system, you have to
# SIGHUP the server for the changes to take effect, run "pg_ctl reload",
# or execute "SELECT pg_reload_conf()".
#
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.
#
# CAUTION: Configuring the system for local "trust" authentication
# allows any local user to connect as any PostgreSQL user, including
# the database superuser. If you do not trust all your local users,
# use another authentication method.

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
# IPv6 local connections:
host all all ::1/128 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all trust
host replication all 127.0.0.1/32 trust
host replication all ::1/128 trust
```

A partir de la version 10, les 3 dernières lignes ne sont plus diésées

# Authentification des clients

---

## B. Les différents champs d'une entrée du fichier pg\_hba.conf

### ▪ TYPE

Il existe plusieurs types de connexion. Le type est toujours précisé en premier dans une entrée du fichier pg\_hba.conf.

- *host*

Une entrée *host* précise les hôtes distants autorisés à se connecter. Le serveur PostgreSQL doit avoir été démarré avec le paramètre `listen_addresses='*' pour accepter les connexions distantes.`

- *local*

Une entrée *local* équivaut à une entrée host sauf qu'il n'y a pas besoin de préciser l'hôte autorisé à se connecter. Cette entrée est utile pour les connexions clientes locales via les sockets UNIX.

- *hostssl*

Une entrée *hostssl* précise les hôtes (locaux ou distants) autorisés à se connecter au serveur PostgreSQL en utilisant une session chiffrée SSL.

### ▪ DATABASE

Nom de(s) base(s) de données auxquelles l'hôte peut se connecter.

- *all*

Le client peut se connecter à toutes les bases de données hébergées par le serveur PostgreSQL.

# Authentification des clients

---

- *nom*

Le client ne peut se connecter qu'à la base *nom*.

- *sameuser*

Le client ne peut se connecter qu'à une base du même nom que l'utilisateur qui est authentifié.

- *nom1,nom2,....*

Le client peut se connecter à cette liste de bases de données.

...

## ■ USER

Spécifie les noms d'utilisateurs PostgreSQL correspondant à cette entrée. Plusieurs noms peuvent être cités (séparés par une virgule). Le mot-clé « all » signifie que le client peut se connecter avec n'importe quel nom utilisateur du serveur PostgreSQL.

## ■ ADDRESS

Ce champ spécifie l'ensemble des hosts correspondant à cette entrée. Il est composé d'une adresse IP et d'un masque réseau. Les deux valeurs définissent les adresses IP des machines clientes qui pourront se connecter. Ce champs est valable uniquement pour les entrées *host* et *hostssl*.

# Authentification des clients

---

## ▪ METHOD

Il existe plusieurs types de connexion. Le type est toujours précisé en premier dans une entrée du fichier pg\_hba.conf.

Spécifie la méthode utilisée par le serveur PostgreSQL pour authentifier un utilisateur tentant de se connecter.

- *trust*

La connexion (de confiance) est acceptée sans condition. Cette méthode permet à quiconque pouvant se connecter à une base de le faire avec le nom d 'utilisateur PostgreSQL de son choix sans avoir à donner un mot de passe.

- *reject*

La connexion est systématiquement refusée. Utile pour filtrer certains hôtes d'un groupe de hôtes.

- *password*

Cette méthode spécifie que l 'utilisateur qui se connecte devra fournir un mot de passe. Ce mot de passe circule en clair sur le réseau et correspond au mot de passe stocké dans le catalogue de PostgreSQL (pg\_shadow).

- *md5*

Idem *password* sauf que le mot de passe est crypté sur le réseau en format md5.

- *scram-sha-256*

Idem *md5* mais plus sécurisée car plus difficile à déchiffrer.

- *krb5 / pam / ldap, radius...*

# Authentification des clients

Exemple de fichier pg\_hba.conf personnalisé :

```
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.

# CAUTION: Configuring the system for local "trust" authentication
# allows any local user to connect as any PostgreSQL user, including
# the database superuser. If you do not trust all your local users,
# use another authentication method.

# TYPE   DATABASE        USER        ADDRESS            METHOD
#
# "local" is for Unix domain socket connections only
local  all            all                     trust
# IPv4 local connections:
host   all            all          127.0.0.1/32      password
host   test3          paul         192.168.56.1/32    md5
# IPv6 local connections:
host   all            all          ::1/128           trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication   all                     trust
host   replication   all          127.0.0.1/32      trust
host   replication   all          ::1/128           trust
```

# Authentification des clients

Test du fichier pg\_hba.conf personnalisé côté serveur :

```
[postgres@CentOS764b ~]# cat /etc/hosts
127.0.0.1    CentOS764b localhost localhost.localdomain localhost4
                  localhost4.localdomain4
::1          localhost localhost.localdomain localhost6
                  localhost6.localdomain6

[postgres@CentOS764b ~]# uname -a
Linux CentOS764b 3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015
x86_64 x86_64 x86_64 GNU/Linux

[postgres@CentOS764b ~]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 08:00:27:DC:14:A6
          inet adr:192.168.56.101 Bcast:192.168.56.255
                    Mask:255.255.255.0
          adr inet6: fe80::a00:27ff:fedc:14a6/64 Scope:Lien
                    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                    RX packets:3560 errors:0 dropped:0 overruns:0 frame:0
                    TX packets:2311 errors:0 dropped:0 overruns:0 carrier:0
                    collisions:0 lg file transmission:1000
                    RX bytes:329890 (322.1 KiB)   TX bytes:302851 (295.7 KiB)

[postgres@CentOS764b ~]# echo $PGDATA
/home/postgres/data/
[postgres@CentOS764b ~]# pg_ctl reload
server signaled
[postgres@CentOS764b ~]# psql template1
psql (11.1)
Type "help" for help.

template1=# \q

[postgres@CentOS764b ~]# psql test3 -U paul -h localhost
Password for user paul:
psql (11.1)
Type "help" for help.

test3=>
```

# Authentification des clients

Test du fichier pg\_hba.conf personnalisé côté client distant :

```
c:\Program Files\PostgreSQL\9.3\bin> type c:\Windows\System32\drivers\etc\hosts
# Copyright (c) 1993-2009 Microsoft Corp.
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
# For example:
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10      x.acme.com          # x client host
192.168.56.101      CentOS764b
```

```
c:\Program Files\PostgreSQL\9.3\bin> ipconfig
Configuration IP de Windows
Carte Ethernet VirtualBox Host-Only Network :
  Suffrage DNS propre à la connexion. . . .
  Adresse IPv4. . . . . : 192.168.56.1
  Masque de sous-réseau. . . . . : 255.255.255.0
  Passerelle par défaut. . . . . :
```

```
c:\Program Files\PostgreSQL\9.3\bin> psql -d template1 -U paul -h CentOS764b
psql: FATAL:  no pg_hba.conf entry for host "192.168.56.1", user "paul", database
"template1"
```

```
c:\Program Files\PostgreSQL\9.3\bin> psql -d test3 -U paul -h CentOS764b
Mot de passe pour l'utilisateur paul :
```

```
psql (9.3.1, serveur 11.1)
ATTENTION : psql version majeure 9.3, version majeure du serveur 10.
            Certaines fonctionnalités de psql pourraient ne pas fonctionner.
Attention : l'encodage console (850) diffère de l'encodage Windows (1252).
            Les caractères 8 bits peuvent ne pas fonctionner correctement.
            Voir la section « Notes aux utilisateurs de Windows » de la page
            référence de psql pour les détails.
Saisissez « help » pour l'aide.
```

```
test3=>
```

# Authentification des clients

---

Depuis la version **10**, il est possible de visualiser le contenu du fichier pg\_hba.conf en interrogeant la vue **pg\_hba\_files\_rules**.

## Exemples :

```
template1=# select line_number,type,database,user_name,address,auth_method,
template1-#           error from pg_hba_file_rules;

line_number | type    | database      | user_name | address       | auth_method | error
-----+-----+-----+-----+-----+-----+-----+
84 | local  | {all}        | {all}      |             | trust        |
86 | host   | {all}        | {all}      | 127.0.0.1   | password    |
87 | host   | {test3}      | {paul}     | 192.168.56.1 | md5         |
89 | host   | {all}        | {all}      | ::1          | trust        |
92 | local  | {replication} | {all}      |             | trust        |
93 | host   | {replication} | {all}      | 127.0.0.1   | trust        |
94 | host   | {replication} | {all}      | ::1          | trust        |
(7 rows)

template1=#

```

## **6. GESTION DE LA SÉCURITÉ**

---

### **Objectif**

A la fin de ce module, vous serez capable de gérer la sécurité de votre serveur PostgreSQL en terme de rôles de type utilisateur ou groupe et de privilèges :

- Introduction
- Gestion des rôles de type « utilisateur »
- Gestion des rôles de type « groupe »
- Gestion des privilèges

# Gestion de la sécurité

---



# Gestion de la sécurité

---

## 1. Les rôles et les privilèges

### A. Introduction

Le système d'exploitation sur lequel est installé un serveur PostgreSQL (Linux, Unix...) possède des mécanismes de sécurité plus ou moins fiables. Le serveur PostgreSQL possède aussi un ensemble de contrôles sophistiqués :

- Filtrage IP des demandes de connexion des hôtes clients (module précédent),
- Authentification des utilisateurs accédant à un serveur PostgreSQL (nom d'utilisateur et mot de passe à fournir à la connexion),
- Utilisation de rôles rassemblant les utilisateurs ayant les mêmes besoins,
- Notion d'utilisateur / rôle permet de gérer l'attribution des droits (privilèges) et de contrôler les actions possibles,
- Chiffrage des sessions,
- ...

### B. Les rôles

PostgreSQL gère l'ensemble des privilèges en utilisant le concept de rôle. Un rôle peut être vu comme un utilisateur (rôle de type utilisateur) ou un groupe d'utilisateurs (rôle de type groupe). La notion de rôle de type groupe facilite l'administration des privilèges.

Les rôles sont des objets globaux liés à une instance PostgreSQL et non à une base particulière d'un serveur. Les informations relatives aux rôles, privilèges... sont enregistrés dans le catalogue de PostgreSQL.

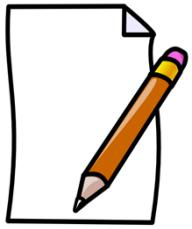
### C. Les privilèges

Quand un objet est créé, il appartient au créateur de cet objet (owner). Initialement, seul le propriétaire de cet objet (et les super-utilisateurs) pourra l'utiliser.

Pour permettre à d'autres d'utiliser ce nouvel objet, des privilèges doivent être gérés. Ces privilèges peuvent être validés / enlevés au niveau rôle de type utilisateur ou groupe.

# Gestion de la sécurité

---



# Gestion de la sécurité

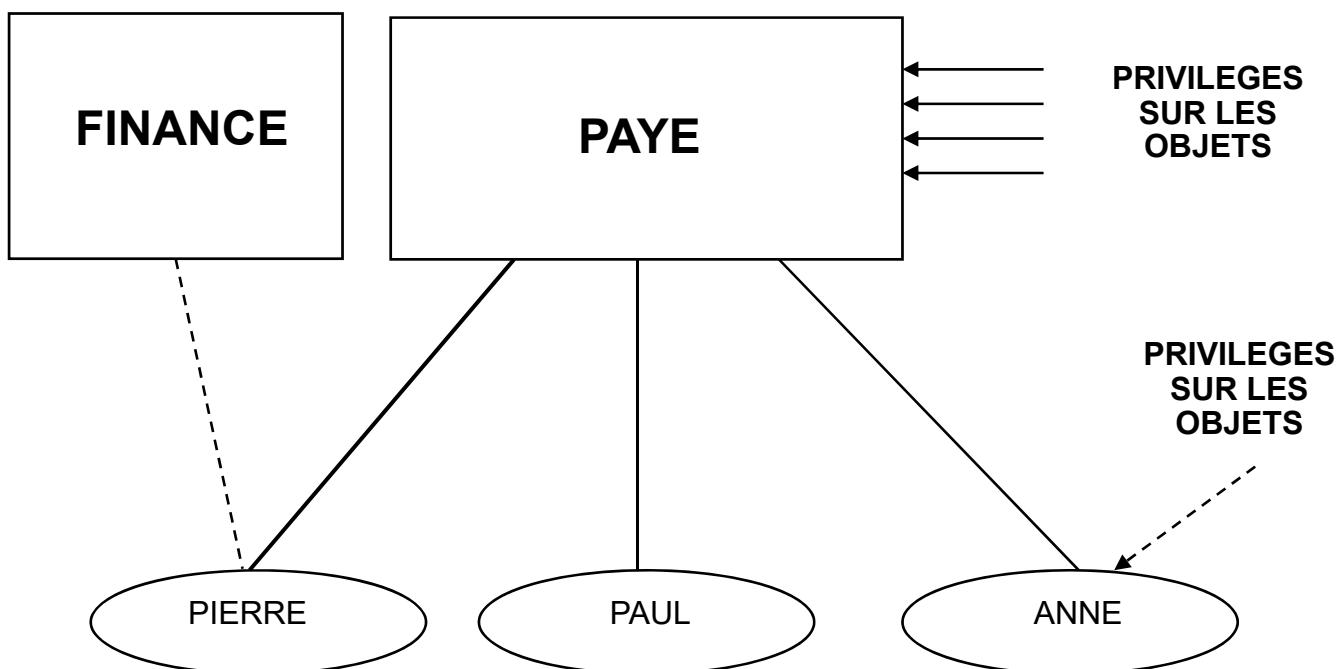
## 2. Gestion des rôles de type utilisateur ou groupe

L'utilisation des rôles de type groupe facilite l'administration des privilèges sur un serveur PostgreSQL.

Les super-utilisateurs créent les rôles de type groupe, rassemblent sous ces rôles de type groupe les différents rôles de type utilisateur ayant les mêmes besoins (administrateurs, développeurs, utilisateurs applicatifs...) et valident un ensemble de privilèges au niveau de ces rôles de type groupe. Les privilèges validés au niveau d'un rôle de type groupe sont accessibles aux membres (rôles de type utilisateur) de ce rôle de type groupe.

Un utilisateur peut être membre de plusieurs rôles de type groupe. Il cumule alors les privilèges de ces différents rôles de type groupe.

Les privilèges sur les objets peuvent aussi être positionnés au niveau d'un rôle de type utilisateur.



# Gestion de la sécurité

---

## A. Mode opératoire pour la mise en place d'une nouvelle application

Lors de la mise en place d'une nouvelle application utilisant une base de données d'un serveur PostgreSQL, il convient de créer un environnement (rôles de type utilisateur et groupes, priviléges...) sous PostgreSQL pour cette nouvelle application :

1. Créer un rôle de type groupe (CREATE ROLE) ou utiliser un groupe existant
2. Créer les rôles de type utilisateur (CREATE ROLE)
3. Valider le rôle de type groupe aux rôles de type utilisateur (GRANT / REVOKE)
4. Valider les priviléges sur les objets au niveau rôle de type groupe ou utilisateur (GRANT / REVOKE).

## B. Création / modification d'un rôle de type groupe ou utilisateur

Un super-utilisateur peut créer un rôle de type groupe ou utilisateur avec la commande SQL « **CREATE ROLE...** ».

La vue **pg\_roles** permet de faire l'état des lieux des rôles de type groupe ou utilisateur de votre serveur.

La commande SQL « **ALTER ROLE...** » permet à un super-utilisateur de modifier les caractéristiques d'un rôle de type groupe ou utilisateur.

Pour se connecter à PostgreSQL, il faut s'identifier à l'aide d'un nom d'utilisateur et éventuellement un mot de passe (dépend de la méthode d'authentification utilisée, voir module précédent).

# Gestion de la sécurité

```
template1=# \h create role

Command:      CREATE ROLE
Description:  define a new database role
Syntax:
CREATE ROLE name [ [ WITH ] option [ ... ] ]

where option can be:

    SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| REPLICATION | NOREPPLICATION
| BYPASSRLS | NOBYPASSRLS
| CONNECTION LIMIT connlimit
| [ ENCRYPTED ] PASSWORD 'password'
| VALID UNTIL 'timestamp'
| IN ROLE role_name [, ...]
| IN GROUP role_name [, ...]
| ROLE role_name [, ...]
| ADMIN role_name [, ...]
| USER role_name [, ...]
| SYSID uid
```

## Principaux paramètres :

### ▪ *name*

Nom du rôle de type groupe ou utilisateur à créer.

# Gestion de la sécurité

---

- **CREATEDB**  
**NOCREATEDB**

Indique si ce nouveau rôle peut créer ou pas des bases de données (par défaut nocreatedb).

- **SUPERUSER**  
**NOSUPERUSER**

Indique si ce nouvel utilisateur est un super-utilisateur ou pas (par défaut nosuperuser). Si superuser, le nouvel utilisateur créé a tous les priviléges.

- **CREATEROLE**  
**NOCREATEROLE**

Indique si ce nouveau rôle pourra créer des rôles (utiliser la commande CREATE ROLE). Par défaut nocreaterole.

- **LOGIN**  
**NOLOGIN**

Indique si ce nouveau rôle peut se connecter ou pas. Un rôle avec l'attribut LOGIN est vu comme rôle de type utilisateur qui pourra être utilisé pour se connecter à une base de données. Un rôle avec l'attribut NOLOGIN est vu comme un rôle de type groupe utile pour gérer les priviléges. Par défaut nologin.

- **PASSWORD '*password*'**

Mot de passe du nouveau rôle de type utilisateur.

# Gestion de la sécurité

---

- **INHERIT**  
**NOINHERIT**

Indique si le nouveau rôle hérite des priviléges des rôles dont il est membre. Par défaut inherit.

- **VALID UNTIL ‘*date*’**

Le mot de passe du rôle de type utilisateur créé expirera à la date indiquée. Après cette date, le mot de passe devra être modifié et la date d'expiration repoussée. Si ce paramètre n'est pas utilisé, l'utilisateur créé sera toujours valide.

- ....

# Gestion de la sécurité

```
template1=# \h alter role
Command:      ALTER ROLE
Description:  change a database role
Syntax:
ALTER ROLE role_specification [ WITH ] option [ ... ]

where option can be:

    SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
    | BYPASSRLS | NOBYPASSRLS
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED ] PASSWORD 'password'
    | VALID UNTIL 'timestamp'

ALTER ROLE name RENAME TO new_name

ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] SET
configuration_parameter { TO | = } { value | DEFAULT }
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] SET
configuration_parameter FROM CURRENT
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] RESET
configuration_parameter
ALTER ROLE { role_specification | ALL } [ IN DATABASE database_name ] RESET ALL

where role_specification can be:

    role_name
    | CURRENT_USER
    | SESSION_USER
```

## Principaux paramètres :

### ▪ ***name***

Nom du rôle de type groupe ou utilisateur à modifier.

### ▪ ***RENAME TO newname***

Nouveau nom du rôle.

### ▪ ***SET configuration\_parameter { TO | = } { value | DEFAULT }***

Positionne la valeur par défaut du paramètre à la valeur donnée.

# Gestion de la sécurité

---

## C. Suppression des rôles de type groupe ou utilisateur

La suppression d'un rôle de type groupe ou utilisateur se fait avec la commande SQL « DROP ROLE... ».

Cette commande est généralement utilisée par les super-utilisateurs.

La suppression d'un rôle de type groupe ou utilisateur n'est possible que s'il n'est pas propriétaire d'une base. S'il est propriétaire d'une base, il faudra d'abord supprimer cette base avant de supprimer ce rôle de type groupe ou utilisateur.

```
template1=# \h drop role
Command:      DROP ROLE
Description: remove a database role
Syntax:
DROP ROLE [ IF EXISTS ] name [, ...]
```

# Gestion de la sécurité

## Exemples :

```
[postgres@CentOS764b ~]$ psql template1 -U postgres
psql (11.1)
Type "help" for help.

template1=# create role pierre login password 'pierre';
CREATE ROLE
template1=# create role jean login password 'jean';
CREATE ROLE
template1=# create role anne login password 'anne';
CREATE ROLE
template1=# create role paye;
CREATE ROLE
template1=# select rolname,rolsuper,rolcreatedb,
template1-#           rolcancanlogin,rolpassword,oid
template1-#         from pg_roles;

          rolname        | rolsuper | rolcreatedb | rolcancanlogin | rolpassword | oid
-----+-----+-----+-----+-----+-----+
postgres          | t       | t       | t       | *****      | 10
pg_monitor        | f       | f       | f       | *****      | 3373
pg_read_all_settings | f       | f       | f       | *****      | 3374
pg_read_all_stats  | f       | f       | f       | *****      | 3375
pg_stat_scan_tables | f       | f       | f       | *****      | 3377
pg_read_server_files | f       | f       | f       | *****      | 4569
pg_write_server_files | f       | f       | f       | *****      | 4570
pg_execute_server_program | f       | f       | f       | *****      | 4571
pg_signal_backend   | f       | f       | f       | *****      | 4200
paul              | f       | f       | t       | *****      | 16387
pierre             | f       | f       | t       | *****      | 16393
jean              | f       | f       | t       | *****      | 16394
anne               | f       | f       | t       | *****      | 16395
paye               | f       | f       | f       | *****      | 16396
(14 rows)

template1=# grant paye to pierre,jean,anne;
GRANT ROLE
template1=# \du

                                         List of roles
   Role name   |          Attributes          | Member of
-----+-----+-----+
anne    |                                     | {paye}
jean   |                                     | {paye}
paul   |                                     | {}
paye   | Cannot login                      | {}
pierre  |                                     | {paye}
postgres| Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

# Gestion de la sécurité

```
template1=# \dg
          List of roles
Role name | Attributes | Member of
-----+-----+-----+
anne    |           | {paye}
jean    |           | {paye}
paul    |           | {}
paye    | Cannot login | {}
pierre   |           | {paye}
postgres | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

template1=# \c test3 anne
You are now connected to database "test3" as user "anne".

test3=> \password
Enter new password:
Enter it again:

test3=> \c test3 postgres
You are now connected to database "test3" as user "postgres".

test3=# select username,password from pg_shadow where username ='jean';

username | password
-----+-----
jean    | md5522e6bf0be8955a0de9531dad8f01ccb
(1 row)

test3=# set password_encryption to "scram-sha-256";
SET
test3=# create role patrick login password 'racine';
CREATE ROLE
test3=# select username,password from pg_shadow where username in ('jean','patrick');

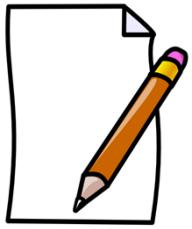
username | password
-----+-----
jean    | md5522e6bf0be8955a0de9531dad8f01ccb
patrick | SCRAM-SHA-256$4096:PDkejqO2bP42iPbxNomnbA==$1TaQ85LrTZNlffaE+C10zhf7dTxKXTDH54wB7JRUSks=
| :neKU/c50TuY4A971SpURsnZo90in8QQzoeC/H0yrRX4=
(2 rows)

test3=#

```

# Gestion de la sécurité

---



# Gestion de la sécurité

## 3. Gestion des privilèges

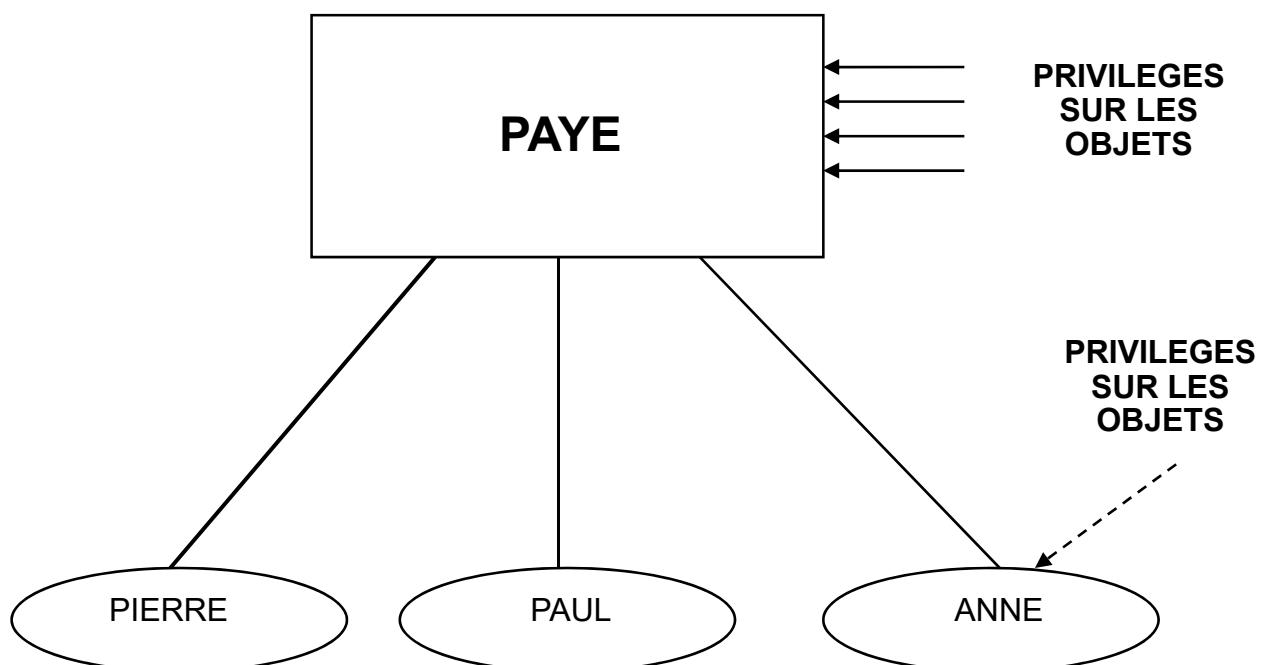
### A. Principe

Quand vous créez un objet dans une base, vous en êtes le propriétaire. Par défaut, seul le propriétaire peut faire ce qu'il veut avec cet objet. Afin de permettre à d'autres utilisateurs d'utiliser cet objet, des privilèges doivent être validés.

Les privilèges d'accès sont gérés par PostgreSQL à travers des **ACL** (Access Control Lists). Ces ACL définissent de manière très précise les types d'accès (consultations, mises à jours des objets d'une base) autorisés pour chaque utilisateur ou groupe d'utilisateurs. Les super-utilisateurs et les propriétaires d'objets distribuent les privilèges avec les commandes SQL **GRANT** et **REVOKE**.

Les privilèges peuvent être validés / retirés au niveau rôle de type utilisateur ou groupe. Initialement, les utilisateurs et les groupes n'ont aucun privilège.

La commande **\dp** de **psql** permet à un propriétaire d'objets de faire l'état des lieux des privilèges accordés aux utilisateurs / groupes sur un ou l'ensemble de ses objets.



# Gestion de la sécurité

---

Les privilèges qui existent dans PostgreSQL sont listés ci-dessous. Les symboles associés aux privilèges sont ceux utilisés dans les résultats de la commande `\dp` de `psql`.

**SELECT** (symbole **r**)

Autorise la lecture d'une table, vue ou séquence.

**INSERT** (**a**)

Autorise l'insertion de nouvelles lignes dans une table.

**UPDATE** (**w**)

Autorise la modification de lignes existantes dans une table.

**DELETE** (**d**)

Autorise la suppression de lignes existantes dans une table.

**EXECUTE** (**X**)

Autorise l'utilisation d'une fonction.

**CREATE** (**C**)

**REFERENCES** (**x**)

**TRUNCATE** (**D**)

**TRIGGER** (**t**)

**TEMP** (**T**)

**USAGE** (**U**)

**ALL PRIVILEGES** (**arwdDxt**)

# Gestion de la sécurité

## B. Validation de privilèges

La validation des privilèges se fait avec la commande SQL **GRANT**.

```
template1=# \h GRANT
Command:      GRANT
Description: define access privileges
Syntax:
  GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES |
    TRIGGER }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
        | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
  ..../.....
  ..../.....
  GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespace_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

  GRANT role_name [, ...] TO role_name [, ...] [ WITH ADMIN OPTION ]
```

Plusieurs privilèges peuvent être validés en une seule fois (les privilèges seront séparés par des virgules).

**tablename** Nom de la table pour laquelle vous validez un / des privilèges.

**role\_name** Nom du rôle de type utilisateur ou groupe à qui vous accordez le / les privilèges.

**PUBLIC** Le mot-clé PUBLIC signifie que le / les privilèges sont validés pour tous les utilisateurs.

# Gestion de la sécurité

## Exemples :

```
[postgres@CentOS764b ~]$ psql template1 -U postgres
psql (11.1)
Type "help" for help.

template1=# create role admpaye login password 'admpaye';
CREATE ROLE
template1=# \c test3 admpaye;
You are now connected to database "test3" as user "admpaye".
test3=> create table t1 (coll numeric);
CREATE TABLE
test3=> create table t2 (coll char(4));
CREATE TABLE
test3=> insert into t1 values (11111111111111);
INSERT 0 1
test3=> insert into t1 values (22222222222222);
INSERT 0 1
test3=> insert into t2 values ('aaaa');
INSERT 0 1
test3=> insert into t2 values ('bbbb');
INSERT 0 1
test3=> \dp

          Access privileges
 Schema | Name | Type | Access privileges| Column privileges | Policies
-----+-----+-----+-----+-----+-----+
 public | t1   | table|               |               |
 public | t2   | table|               |               |
(2 rows)

test3=>
```

# Gestion de la sécurité

```
test3=> grant select,insert,update on t1 to paye;
GRANT
test3=> grant select on t2 to public;
GRANT
test3=> grant delete on t1 to anne;
GRANT
test3=> \dp
          Access privileges
 Schema |Name|Type |      Access privileges      | Column privileges | Policies
-----+---+---+-----+-----+
public |t1 |table| admpaye=arwdDxt/admpaye+|           |
       |   |     | paye=arw/admpaye      +|           |
       |   |     | anne=d/admpaye          |           |
public |t2 |table| admpaye=arwdDxt/admpaye+|           |
       |   |     | =r/admpaye            |           |
(2 rows)

test3=> \c test3 postgres
You are now connected to database "test3" as user "postgres".

test3=# create role philippe login password 'philippe';
CREATE rôle
test3=# \c test3 pierre
You are now connected to database "test3" as user "pierre".

test3=> select * from t1;

      col1
-----
11111111111111
22222222222222
(2 rows)

test3=> select * from t2;

      col1
-----
aaaa
bbbb
(2 rows)

test3=> delete from t1;
ERROR: permission denied for relation t1
test3=> insert into t1 values (555555555555);
INSERT 0 1
test3=>
```

# Gestion de la sécurité

```
test3=> \c test3 anne
You are now connected to database "test3" as user "anne".

test3=> delete from t1 where col1=11111111111111;
DELETE 1
test3=> insert into t1 values (666666666666);
INSERT 0 1
test3=> select * from t1;

      col1
-----
22222222222222
 555555555555
 66666666666666
(3 rows)

test3=> \c test3 philippe
You are now connected to database "test3" as user "philippe".

test3=> select * from t2;

      col1
-----
aaaa
bbbb
(2 rows)

test3=> select * from t1;

ERROR: permission denied for relation t1
test3=>
```

# Gestion de la sécurité

### **C. Invalidation de privilèges**

L'invalidation de priviléges se fait avec la commande SQL **REVOKE**.

```
test=# \h REVOKE
Command:      REVOKE
Description:  remove access privileges
Syntax:
-----
REVOKE [ GRANT OPTION FOR ]
  { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES |
TRIGGER }
  [, ...] | ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...]
    | ALL TABLES IN SCHEMA schema_name [, ...] }
FROM { [ GROUP ] role_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]
-----
.....
.....
REVOKE [ ADMIN OPTION FOR ]
role_name [, ...] FROM role_name [, ...]
[ CASCADE | RESTRICT ]
```

# Gestion de la sécurité

## Exemple :

```
test3=> \c test3 admpaye
You are now connected to database "test3" as user "admpaye".

test3=> revoke update,insert on t1 from paye;
REVOKE
test3=> revoke delete on t1 from anne;
REVOKE
test3=> \c test3 pierre
You are now connected to database "test3" as user "pierre".

test3=> update t1 set col1=8888 where col1=22222222222222;
ERROR: permission denied for relation t1
test3=> select * from t1;

      col1
-----
22222222222222
    555555555555
    66666666666666
(3 rows)

test3=> \c test3 anne
You are now connected to database "test3" as user "anne".

test3=> delete from t1 where col1=22222222222222;
ERROR: permission denied for relation t1
test3=> insert into t1 values (7777);
ERROR: permission denied for relation t1
test3=> select * from t1;

      col1
-----
22222222222222
    555555555555
    66666666666666
(3 rows)

test3=> \c test3 postgres
You are now connected to database "test3" as user "postgres".
test3=# drop role admpaye;
ERROR: role "admpaye" cannot be dropped because some objects depend on it
DETAIL: owner of table t2
owner of table t1
test3=# drop role pierre;
DROP ROLE
test3=#

```

## 7. MAINTENANCE D'UNE INSTANCE POSTGRESQL

---

### Objectif

A la fin de ce module, vous connaîtrez les principales opérations de maintenance à effectuer sur les bases de données d'un serveur PostgreSQL :

- Opérations de maintenance sur un serveur PostgreSQL
- Gestion de l'espace disque, espace disque utilisé (vues système, fonction SQL d'administration, contrib oid2name)
- Nettoyage d'une base de données
- Autovacuum
- Utilitaire VACUUM
- Le fichier de log
- pgBadger

# Maintenance d'un serveur PostgreSQL

---



# Maintenance d'un serveur PostgreSQL

---

## 1. Opérations de maintenance sur un serveur PostgreSQL

### A. Introduction

Certaines opérations de maintenance doivent être effectuées régulièrement sur un serveur PostgreSQL pour qu'il fonctionne de manière optimale. Ces opérations sont répétitives et peuvent être facilement automatisées avec des outils systèmes standard (cron, Task Scheduler de Windows...).

Une des principales opérations de maintenance pour un dba est la mise au point d'une **stratégie de sauvegarde** des données. Sans sauvegarde récente, vous ne pourrez pas restaurer une base après une panne (panne disque, panne secteur, crash système...). Les mécanismes de sauvegardes et restaurations seront vus dans le module « **Sauvegardes et restaurations** ».

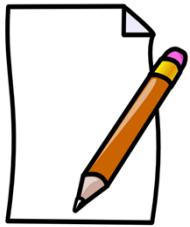
Les autres principales opérations de maintenance à effectuer sont :

- Gérer l'espace disque du serveur PostgreSQL,
- Nettoyer périodiquement les bases de données de votre serveur (**vacuum**),
- Surveillez périodiquement le **fichier de log** du serveur,
- Reconstruire périodiquement les index (commande SQL REINDEX) dans certains cas.

Mise à part ces quelques opérations périodiques de maintenance, un serveur PostgreSQL ne nécessite pas énormément de maintenance comparé à d'autres SGBD.

# Maintenance d'un serveur PostgreSQL

---



# Maintenance d'un serveur PostgreSQL

---

## 2. Gestion de l'espace disque

### A. Introduction

Mise à part l'utilisation des tablespaces, l'administrateur n'a pratiquement aucun moyen d'action sur l'agencement de l'espace disque consommé par les bases de données sous PostgreSQL.

### B. Espace disque utilisé

Chaque table est composée d'un fichier disque principal dans lequel la plupart des données sont stockées (un fichier TOAST peut être associé à la table si des colonnes de cette table stockent des valeurs étendues. Si ce fichier toast existe, un index lui est associé). Des index peuvent également être associés à la table de base. Chaque table ou index est stocké dans un fichier distinct ou plusieurs si la taille du fichier dépasse **1 Go** par défaut (paramètre `--with-segsize=SEGSIZE` de l'utilitaire configure).

Vous pouvez contrôler l'espace disque de plusieurs manières :

- avec psql en interrogeant les tables systèmes,
- avec psql en utilisant les fonctions SQL d'administration,
- avec les outils de **contrib/oid2name**

# Maintenance d'un serveur PostgreSQL

## C. Utilisation de psql et des vues systèmes

Requêtes pour connaître l'espace disque consommé par une table ou un index.

```
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# select relname,relkind,relfilenode,relpages
prod-#          from pg_class
prod-# where relname like 'emp%'
prod-# order by relpages desc;

   relname  | relkind | relfilenode | relpages
-----+-----+-----+-----
  emp      | r       |      16434 |      1944
  emp_ename | i       |      16452 |       798
(2 rows)
```

Pour connaître l'espace disque utilisé par les fichiers TOAST associés à une table (ici table **t1**), on utilise une requête similaire à la suivante :

```
prod=# SELECT relname, relpages
prod-# FROM pg_class,
prod-#       (SELECT reltoastrelid
prod-#            FROM pg_class
prod-#           WHERE relname = 't1') AS ss
prod-# WHERE oid = ss.reltoastrelid OR
prod-#       oid = (SELECT reltoastidxid
prod-#            FROM pg_class
prod-#           WHERE oid = ss.reltoastrelid)
prod-# ORDER BY relname;

   relname  | relpages
-----+-----
pg_toast_16454 |      0
pg_toast_16454_index |      1
(2 rows)
```

La taille des blocs est de **8K** par défaut...

**Attention** : l'information **relpages** est mise à jour uniquement par VACUUM ou ANALYZE et par quelques commandes DDL comme CREATE INDEX. La taille affichée des objets peut être erronée si la base de données n'a pas été récemment nettoyée ou analysée.

# Maintenance d'un serveur PostgreSQL

Espace consommé par les index de la table **emp** :

```
prod=# select c2.relname,c2.relpages
prod-#      from pg_class c,pg_class c2,pg_index i
prod-# where c.relname='emp' and c.oid=i.indrelid and
c2.oid=i.indexrelid
prod-# order by c2.relname;

   relname   |  relpages
-----+-----
 emp_ename |      798
(1 row)
```

Avec pgAdmin3

The screenshot shows the pgAdmin III interface. The left pane is the 'Navigateur d'objets' (Object Navigator) displaying the database structure. The right pane has tabs for 'Propriétés' (Properties), 'Statistiques' (Statistics), 'Dépendances' (Dependencies), and 'Objets dépendants' (Dependent objects). The 'Propriétés' tab is selected, showing statistics for the 'emp' table. The 'Taille des index' (Index Size) is listed as 6384 kB. The 'SQL' pane at the bottom contains the creation script for the 'emp' table.

Statistique	Valeur
Dernier VACUUM	2010-07-10 11:07:41.074024+02
Dernier auto-VACUUM	
Dernier ANALYZE	
Dernier auto-ANALYZE	
Taille de la table	15 MB
Taille de la table TOAST	aucun
Taille des index	6384 kB

```
-- Table: emp

-- DROP TABLE emp;

CREATE TABLE emp
(
    empno integer NOT NULL,
    ename character varying(10),
    job character varying(9),
    mgr integer,
    hiredate date,
    sal numeric(7,2),
    comm numeric(7,2),
    deptno integer
)
WITH (
    OIDS=FALSE
);
```

Chargement des détails sur les objets Table...Exécuté. 0,00 secondes

# Maintenance d'un serveur PostgreSQL

## C. Utilisation des fonctions SQL d'administration

PostgreSQL met à disposition des fonctions systèmes d'administration (chapitre **9.24 System Administration functions**) qui permettent de récupérer avec psql les tailles d'une table, d'un index, d'un tablespace ou d'une base sans avoir besoin d'effectuer un **vacuum** ou un **analyze**. Les fonctions **pg\_relation\_filenode** et **pg\_relation\_filepath** permettent de localiser un fichier représentant un objet

Nom	Code de retour	Description
<code>pg_column_size(any)</code>	int	Nombre d'octets utilisés pour stocker une valeur particulière (éventuellement compressée)
<code>pg_database_size(oid)</code>	bigint	Espace disque utilisé par la base de données d'OID indiqué
<code>pg_database_size(name)</code>	bigint	Espace disque utilisé par la base de données de nom indiqué
<code>pg_indexes_size(regclass)</code>	bigint	Espace disque total utilisé par les index attachés à la table dont l'OID ou le nom est indiqué
<code>pg_relation_size(relation regclass, fork text)</code>	bigint	Espace disque utilisé par le fork indiqué, 'main', 'fsm' ou 'vm', d'une table ou index d'OID ou de nom indiqué.
<code>pg_relation_size(relation regclass)</code>	bigint	Raccourci pour <code>pg_relation_size(..., 'main')</code>
<code>pg_size_pretty(bigint)</code>	text	Convertit une taille en octets en format interprétable par l'utilisateur avec unités
<code>pg_table_size(regclass)</code>	bigint	Espace disque utilisé par la table spécifiée, en excluant les index (mais en incluant les données TOAST, la carte des espaces libres et la carte de visibilité)
<code>pg_tablespace_size(oid)</code>	bigint	Espace disque utilisé par le tablespace ayant cet OID
<code>pg_tablespace_size(name)</code>	bigint	Espace disque utilisé par le tablespace ayant ce nom
<code>pg_total_relation_size(regclass)</code>	bigint	Espace disque total utilisé par la table spécifiée, en incluant toutes les données TOAST et les index

Nom	Type en retour	Description
<code>pg_relation_filenode(relation regclass)</code>	oid	Numéro filenode de la relation indiquée
<code>pg_relation_filepath(relation regclass)</code>	text	Chemin et nom du fichier pour la relation indiquée

# Maintenance d'un serveur PostgreSQL

## Exemple :

```
prod=# select pg_total_relation_size('emp');

 pg_total_relation_size
-----
 22487040
(1 row)
prod=# select pg_table_size('emp');

 pg_table_size
-----
 15949824
(1 row)
prod=# select pg_indexes_size('emp');

 pg_indexes_size
-----
 6537216
(1 row)
prod=# select oid,datname from pg_database;

 oid | datname
-----+-----
 1   | template1
11866 | template0
11874 | postgres
16396 | test3
16397 | test
16414 | prod
(6 rows)
prod=# select oid,relname from pg_class where relname='emp';

 oid | relname
-----+-----
 16434 | emp
(1 row)
prod=# select pg_relation_filepath('emp');

 pg_relation_filepath
-----
 base/16414/16434
(1 row)
prod=# select pg_relation_filenode('emp');
 pg_relation_filenode
-----
 16434
(1 row)
prod=# ! ls -l $PGDATA/base/16414/16434
-rw----- 1 postgres postgres 15925248 jui 10 11:13
/home/postgres/data/base/16414/16434
```

# Maintenance d'un serveur PostgreSQL

```
prod=# select pg_database_size('prod');

pg_database_size
-----
27847548
(1 row)

prod=# select pg_size_pretty(pg_database_size('prod'));

pg_size.pretty
-----
27 MB
(1 row)

prod=#
```

# Maintenance d'un serveur PostgreSQL

## D. Utilisation des outils de contrib/oid2name

Dans l'arborescence source de PostgreSQL, le répertoire « contrib/oid2name » met à disposition l'environnement permettant de créer les lignes de commande **oid2name**. Les lignes de commande **oid2name** affichent les noms des répertoires O.S correspondants aux bases de données et les noms des fichiers O.S correspondant aux tables ce qui permettra ensuite, avec des commandes O.S de récupérer l'espace disque consommé.

Sous Windows, la commande ligne **oid2name** existe par défaut (elle se trouve dans la binaire du logiciel PostgreSQL).

Sous Linux, création des lignes de commande **oid2name** avec l'utilisateur **root** et exemple d'utilisation :

```
[root@CentOS764b ~]$ cd /usr/local/src/postgresql-11.1/contrib/oid2name/
[root@CentOS764b oid2name]# make
.../...
.../...
.../...
.../...
.../...
[root@CentOS764b oid2name]# make install
.../...
.../...
.../...
.../...
.../...
[root@CentOS764b oid2name]#
```

# Maintenance d'un serveur PostgreSQL

```
[postgres@CentOS764b ~]$ oid2name -help

oid2name helps examining the file structure used by PostgreSQL.
Usage:
  oid2name [OPTION]...
Options:
  -d DBNAME      database to connect to
  -f FILENODE    show info for table with given file node
  -H HOSTNAME    database server host or socket directory
  -i              show indexes and sequences too
  -o OID         show info for table with given OID
  -p PORT        database server port number
  -q              quiet (don't show headers)
  -s              show all tablespaces
  -S              show system objects too
  -t TABLE       show info for named table
  -U NAME        connect as specified database user
  -V, --version   output version information, then exit
  -x              extended (show additional columns)
  -?, --help     show this help, then exit
The default action is to show all database OIDs.
Report bugs to <pgsql-bugs@postgresql.org>.
```

```
[postgres@CentOS764b ~]$ oid2name

All databases:
  Oid  Database Name  Tablespace
-----
  11874      postgres  pg_default
  16414      prod      pg_default
  11866      template0 pg_default
  1          template1 pg_default
  16397      test      pg_default
  16396      test3     pg_default
[postgres@CentOS764b ~]$ oid2name -d prod -x

From database "prod":
  Filenode  Table Name  Oid  Schema  Tablespace
-----
  16440      bonus     16440  public   pg_default
  16437      dept      16437  public   pg_default
  16434      emp       16434  public   pg_default
  16446      salgrade  16446  public   pg_default
  16454      t1        16454  public   tbsp1
[postgres@CentOS764b ~]$ oid2name -d prod -t emp -x

From database "prod":
  Filenode  Table Name  Oid  Schema  Tablespace
-----
  16434      emp       16434  public   pg_default
```

# Maintenance d'un serveur PostgreSQL

```
[postgres@CentOS764b ~]$ oid2name -d prod -f 16454
From database "prod":
  Filenode  Table Name
-----
    16454        t1
[postgres@CentOS764b ~]$ oid2name -s
All tablespaces:
  Oid  Tablespace Name
-----
  1663      pg_default
  1664      pg_global
  16453     tbspl
```

## Conseils

Le plus important pour un dba, concernant la surveillance de l'espace disque, est de s'assurer que les disques, contenant les bases de données, ne soient pas dangereusement pleins.

Libérer de l'espace sur le disque avant qu'il ne soit complètement plein pour éviter les problèmes mais aussi pour des raisons d'optimisation. Certains fichiers systèmes ne fonctionnent pas de manière optimisée lorsqu'ils sont presque pleins.

# Maintenance d'un serveur PostgreSQL

---



# Maintenance d'un serveur PostgreSQL

---

## 3. Nettoyage d'une base de données

### A. Le « VACUUMING » d'une base de données

La commande **VACUUM** de PostgreSQL doit être exécutée régulièrement sur chaque table pour récupérer ou réutiliser de l'espace disque occupé par des lignes supprimées ou mises à jour, mettre à jour les statistiques utilisées par l'optimiseur de PostgreSQL. La fréquence des opérations de **VACUUM** dépend de l'utilisation de votre base de données (transactionnelle, décisionnelle, infocentre...).

**VACUUM** peut s'exécuter en parallèle avec l'exploitation normale d'une base de données (utilisation possible de dml select, insert, update, delete mais pas de ddl).

### B. Autovacuum

Depuis la version **8** de PostgreSQL, le démon **autovacuum** permet d'automatiser l'exécution des commandes **VACUUM** et **ANALYZE**.

Quand autovacuum est validé, il contrôle les tables qui ont eu un grand nombre de lignes inserted / updated / deleted. Ces contrôles utilisent les mécanismes de collecte de statistiques. Donc autovacuum ne peut être utilisé que si le paramètre **track\_counts** (paramètre de validation du statistics collector) est à true.

Depuis la **8.3.0**, autovacuum est validé par défaut (**autovacuum = on**) et les paramètres de configuration correspondants correctement positionnés.

Un nouveau process « **autovacuum launcher** » a en charge de démarrer des process « **autovacuum worker** » sur chaque base de données toutes les **autovacuum\_naptime** secondes (par défaut 60 secondes). Si le serveur a  $N$  bases de données, un nouveau autovacuum worker sera lancé tous les  $\text{autovacuum\_naptime}/N$ . A chaque exécution, le worker process vérifie chaque table dans la base et des vacuum ou analyze sont déclenchés si nécessaire.

Le travail effectué par autovaccum convient généralement à une majorité de serveur PostgreSQL. Les paramètres de réglage de autovaccum peuvent être utilisés pour affiner ce travail. Le dba peut remplacer l'activité automatique du démon autovaccum et gérer manuellement les commandes vacuum et utiliser des outils de planification (cron, Task Scheduler ...).

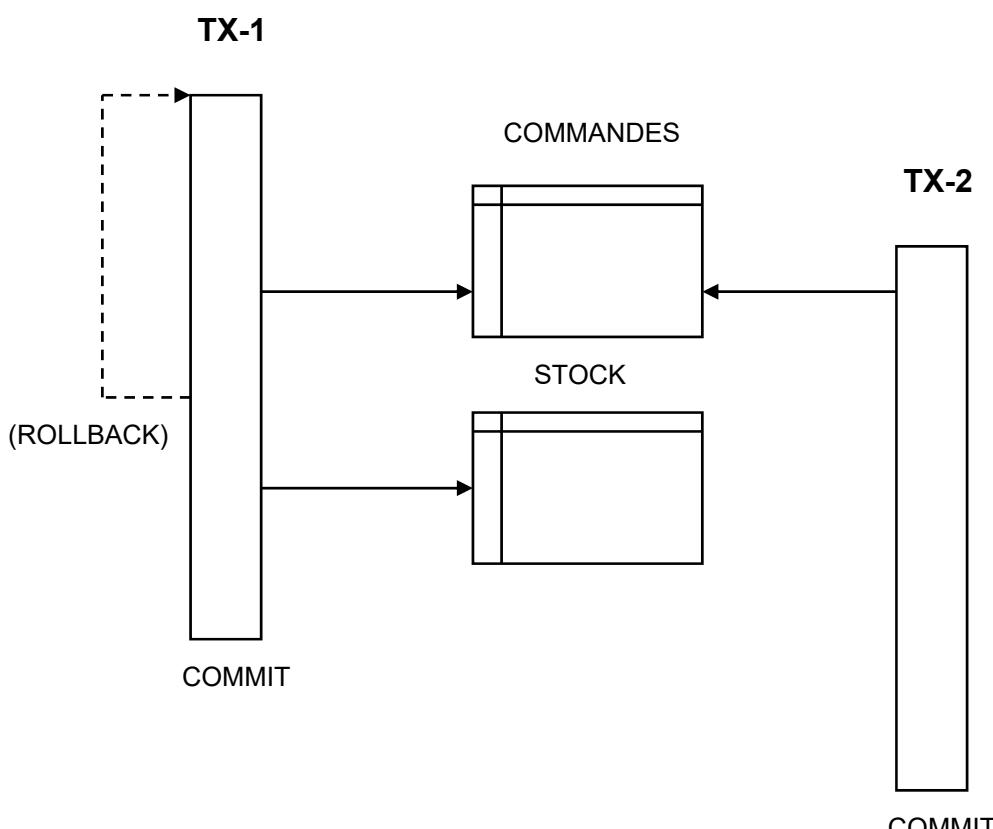
# Maintenance d'un serveur PostgreSQL

## C. Récupération d'espace disque

Un UPDATE ou DELETE d'une ligne dans une transaction ne supprime pas immédiatement l'ancienne version de la ligne. Cette ancienne version de la ligne constitue l'image avant des données en cours de modification. Cette approche est nécessaire dans un environnement transactionnel de concurrence d'accès. La ligne ne doit pas être supprimée tant qu'elle est potentiellement visible, utile pour d'autres transactions en cours (lecture consistante). Ces images avant peuvent aussi servir à effectuer des rollback de transactions et des récupérations suite à des crash d'instance ou incidents matériels.

Mais d'un autre côté, une ligne supprimée par une transaction validée (commit) ne sera plus utile à aucune autre future transaction. L'espace disque qu'elle occupe doit être récupéré pour être à nouveau utilisé par de nouvelles lignes afin d'éviter une consommation infinie d'espace disque.

Cette récupération d'espace est faite par **VACUUM** qui repère les lignes obsolètes dans les tables et les index et récupère l'espace correspondant. Cet espace pourra être de nouveau utilisé. Dans la plupart des cas, il est recommandé de programmer un **VACUUM** de toute la base **une fois par jour** à un moment de faible charge. Une table fréquemment mise à jour aura besoin d'être plus souvent purgée. Il peut être utile de programmer périodiquement des **VACUUM** uniquement sur certaines tables.



# Maintenance d'un serveur PostgreSQL

---

## D. Mise à jour des statistiques de l'optimiseur

L'optimiseur de PostgreSQL dépend des informations statistiques récoltées sur les tables pour optimiser les requêtes. Ces statistiques sont générées par la commande **ANALYZE** qui peut être invoquée seule ou optionnellement à travers **VACUUM**.

Il est important d'avoir des statistiques pertinentes sinon les « mauvais » plans d'optimisation des requêtes peuvent dégrader les performances de votre base de données.

Sur la plupart des serveurs, il est généralement recommandé de planifier un ANALYZE par jour au niveau base de données à une période de faible charge. Il peut être combiné utilement avec un VACUUM. Comme pour la récupération d'espace, les mises à jour fréquentes des statistiques sont plus utiles sur les tables souvent mises à jour.

Le daemon autovacuum contrôle l'activité des tables (lignes insérées, mises à jours ou supprimées) et exécute des ANALYZES si nécessaire.

Le dba peut toujours planifier manuellement des ANALYZE.

# Maintenance d'un serveur PostgreSQL

---



# Maintenance d'un serveur PostgreSQL

## 4. Utilisation manuelle de VACUUM

### A. Syntaxe de la commande VACUUM

```
prod=# \h vacuum
Command:      VACUUM
Description:  garbage-collect and optionally analyze a database
Syntax:
VACUUM [ ( { FULL | FREEZE | VERBOSE | ANALYZE | DISABLE_PAGE_SKIPPING }
[, ...] ) ] [ table_name [ (column_name [, ...] ) ] ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table_name ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table_name [ (column_name
[, ...] ) ] ]
```

**VACUUM** utilisé sans mots-clés ou identificateur nettoie toutes les tables de la base de données sur laquelle vous êtes connecté.

**VACUUM table** nettoie la table indiquée dans la base de données à laquelle vous êtes connecté.

**VACUUM ANALYZE** ou **VACUUM ANALYZE table** nettoie toutes les tables ou la table indiquée puis met à jour les statistiques utilisées par l'optimiseur pour déterminer le meilleur plan d'exécution d'une requête.

L'option **VERBOSE** génère plus d'informations concernant l'exécution de l'utilitaire VACUUM.

#### Remarques :

L'option **FULL** ne doit pas être utilisée systématiquement. Elle est utile quand vous avez supprimé la plupart des lignes d'une table et que vous voulez diminuer physiquement et de manière significative la taille disque occupée. Depuis la **9.0**, l'option **FULL** crée la totalité de la table dans un nouveau fichier. Cette option nécessite l'exclusivité de la table et de l'espace disque pendant la phase de création.

Comme d'autres commandes SQL, VACUUM possède un équivalent en ligne de commande : **vacuumdb**. La principale différence avec la commande SQL VACUUM est que vous pouvez effectuer un VACUUM sur toutes les bases de données d'un serveur PostgreSQL.

# Maintenance d'un serveur PostgreSQL

## Exemples de VACUUM manuel (autovacuum à off) sur une base de données :

```
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# select relname,relpages,relfilenode from pg_class
prod-#           where relname like 'emp%';

   relname   |  relpages  |  relfilenode
-----+-----+-----+
  emp      |       1 |      16406
  emp_ename |       2 |      16424
(2 rows)

prod=# vacuum verbose analyze;
INFO:  vacuuming "public.dept"
INFO:  "dept": found 0 removable, 4 nonremovable row versions in 1 out of 1
pages
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 631
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
INFO:  analyzing "public.dept"
INFO:  "dept": scanned 1 of 1 pages, containing 4 live rows and 0 dead rows; 4
rows in sample, 4 estimated total rows
.../...
.../...
INFO:  vacuuming "public.emp"
INFO:  index "emp_ename" now contains 458752 row versions in 1612 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
INFO:  "emp": found 0 removable, 458752 nonremovable row versions in 3926 out
of 3926 pages
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 633
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.02 s, system: 0.00 s, elapsed: 0.03 s.
INFO:  analyzing "public.emp"
INFO:  "emp": scanned 3926 of 3926 pages, containing 458752 live rows and 0
dead rows; 30000 rows in sample, 458752 estimated total rows
INFO:  vacuuming "pg_catalog.pg_statistic"
INFO:  index "pg_statistic_relid_att_inh_index" now contains 407 row versions
in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
INFO:  "pg_statistic": found 0 removable, 407 nonremovable row versions in 16
out of 16 pages
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 634
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
```

# Maintenance d'un serveur PostgreSQL

```
.../  
.../  
INFO:  vacuuming "pg_toast.pg_toast_13076"  
INFO:  index "pg_toast_13076_index" now contains 0 row versions in 1 pages  
DETAIL: 0 index row versions were removed.  
0 index pages have been deleted, 0 are currently reusable.  
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.  
INFO:  "pg_toast_13076": found 0 removable, 0 nonremovable row versions in  
0 out of 0 pages  
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 676  
There were 0 unused item pointers.  
Skipped 0 pages due to buffer pins, 0 frozen pages.  
0 pages are entirely empty.  
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.  
INFO:  analyzing "information_schema.sql_sizing_profiles"  
INFO:  "sql_sizing_profiles": scanned 0 of 0 pages, containing 0 live rows  
and 0 dead rows; 0 rows in sample, 0 estimated total rows  
VACUUM  
prod=# select relname,relpages,relfilenode from pg_class  
prod-#           where relname like 'emp%';  
  
relname | relpages | relfilenode  
-----+-----+-----  
emp     |      3926 |      16406  
emp_ename |     1612 |      16424  
(2 rows)  
  
prod=#
```

# Maintenance d'un serveur PostgreSQL

## Exemples de VACUUM manuel sur une table :

```
prod=# select oid,datname from pg_database
prod-#           where datname ='prod';

     oid  | datname
-----+-----
 16405  | prod
(1 row)

prod=# \q
[postgres@CentOS764b ~]$ cd $PGDATA/base/16405
[postgres@CentOS764b 16405]$ ls -l|grep 16406

-rw-----. 1 postgres postgres 32161792 23 janv. 12:56 16406
-rw-----. 1 postgres postgres      24576 23 janv. 12:56 16406_fsm
-rw-----. 1 postgres postgres      8192 23 janv. 12:56 16406_vm

[postgres@CentOS764b 16405]$ psql prod
psql (11.1)
Type "help" for help.

prod=# delete from emp where deptno=10;
DELETE 98304
prod=# \! ls -l|grep 16406

-rw-----. 1 postgres postgres 32161792 23 janv. 12:56 16406
-rw-----. 1 postgres postgres      24576 23 janv. 12:56 16406_fsm
-rw-----. 1 postgres postgres      8192 23 janv. 12:56 16406_vm

prod=# vacuum verbose analyze emp;

INFO:  vacuuming "public.emp"
INFO:  scanned index "emp_ename" to remove 98304 row versions
DETAIL:  CPU: user: 0.05 s, system: 0.00 s, elapsed: 0.05 s
INFO:  "emp": removed 98304 row versions in 3926 pages
DETAIL:  CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s
INFO:  index "emp_ename" now contains 360448 row versions in 1612 pages
DETAIL:  98304 index row versions were removed.
343 index pages have been deleted, 0 are currently reusable.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
INFO:  "emp": found 98304 removable, 360448 nonremovable row versions in
      3926 out of 3926 pages
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 677
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.09 s, system: 0.00 s, elapsed: 0.09 s.
INFO:  analyzing "public.emp"
INFO:  "emp": scanned 3926 of 3926 pages, containing 360448 live rows and 0
      dead rows; 30000 rows in sample, 360448 estimated total rows
VACUUM
```

# Maintenance d'un serveur PostgreSQL

```
prod=# \! ls -l|grep 16406
-rw-----. 1 postgres postgres 32161792 23 janv. 13:11 16406
-rw-----. 1 postgres postgres      24576 23 janv. 13:11 16406_fsm
-rw-----. 1 postgres postgres      8192 23 janv. 13:11 16406_vm

prod=# delete from emp where deptno=20;
DELETE 163840
prod=# \! ls -l|grep 16406
-rw-----. 1 postgres postgres 32161792 23 janv. 13:11 16406
-rw-----. 1 postgres postgres      24576 23 janv. 13:11 16406_fsm
-rw-----. 1 postgres postgres      8192 23 janv. 13:11 16406_vm

prod=# select count(*) from emp;
 count
-----
 196608
(1 row)

prod=# vacuum full verbose analyze emp;
INFO:  vacuuming "public.emp"
INFO:  "emp": found 163840 removable, 196608 nonremovable row versions in
3926 pages
DETAIL:  0 dead row versions cannot be removed yet.
CPU: user: 0.07 s, system: 0.03 s, elapsed: 0.19 s.
INFO:  analyzing "public.emp"
INFO:  "emp": scanned 1740 of 1740 pages, containing 196608 live rows and
0 dead rows; 30000 rows in sample, 196608 estimated total rows
VACUUM

prod=# \! ls -l|grep 16406
-rw-----. 1 postgres postgres          0 23 janv. 13:12 16406

prod=# select oid,relname,relpages,relfilenode from pg_class
prod-#      where relname like 'emp%';

   oid  |  relname  |  relpages  |  relfilenode
-----+-----+-----+-----+
  16406 |  emp      |      1740 |      16433
  16424 |  emp_ename |        542 |      16436
(2 rows)

prod=# \! ls -l|grep 16433
-rw-----. 1 postgres postgres 14254080 23 janv. 13:12 16433
prod=#

```

# Maintenance d'un serveur PostgreSQL

---



# Maintenance d'un serveur PostgreSQL

---

## 5. Le fichier de log

### A. Gestion du fichier de log du serveur PostgreSQL

Il peut être intéressant d'utiliser le fichier de log d'un serveur PostgreSQL (plutôt que de le diriger vers /dev/null) pour diagnostiquer un problème.

Il enregistre les évènements importants d'un serveur PostgreSQL (démarrages, arrêts, les erreurs, les checkpoints, consommation des WAL...). Ce fichier doit être analysé et nettoyé (copié, archivé, supprimé...) régulièrement car il peut s'étendre rapidement en fonction de la précision des évènements journalisés.

Plusieurs méthodes sont possibles pour utiliser un fichier de log d'un serveur PostgreSQL :

1. Utilisation d'un fichier de log (stderr) défini au démarrage du serveur avec l'option **-I filename** de pg\_ctl.
2. Utilisation d'un fichier de log (stderr) défini à travers des paramètres de configuration **log\_destination='stderr'**, **logging\_collector=on**. D'autres paramètres peuvent être utilisés pour définir le nom du fichier de log (**log\_filename**), le répertoire qui contiendra ce fichier (**log\_directory**).
3. Utilisation du fichier de syslog du système d'exploitation en positionnant les paramètres de configuration **log\_destination**, **syslog\_facility**, **syslog\_ident**
4. Utilisation d'un mécanisme de rotation des fichiers de log du serveur PostgreSQL avec les paramètres **log\_destination='stderr'**, **logging\_collector=on**, **log\_truncate\_on\_rotation=on**.  
La rotation des fichiers de logs du serveur permet de démarrer sur des nouveaux fichiers de logs et de supprimer les anciens après une période de temps raisonnable.

Remarque : pgBadger est un outil externe intéressant pour analyser facilement et rapidement le contenu des fichiers de log d'un serveur PostgreSQL.

# Maintenance d'un serveur PostgreSQL

## B. Contenu du fichier log

De nombreux paramètres de configuration définissent les différents évènements qui seront loggés ainsi que le niveau de précision des évènements loggés (**log\_connections**, **log\_disconnections**, **log\_statement**, **log\_hostname**, **log\_duration**, **log\_line\_prefix** ...).

**Exemple d'utilisation du logging collector (stderr) pour le fichier de log définis à travers des paramètres de configuration (postgresql.conf) d'une instance PostgreSQL :**

```
log_destination = 'stderr'
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_min_duration_statement = 0
log_checkpoints = on
log_connections = on
log_disconnections = on
log_duration = on
log_line_prefix = '%r-%d-%u-%t'
log_statement = 'all'
log_hostname = on
```

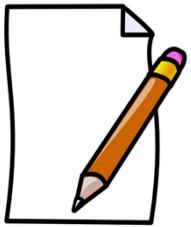
```
[postgres@CentOS764b ~]$ pg_ctl stop
waiting for server to shut down.... done
server stopped
[postgres@CentOS764b ~]$ pg_ctl start
waiting for server to start.....2018-01-23 13:29:26 CET LOG:  listening on IPv4
address "0.0.0.0", port 5432
---2018-01-23 13:29:26 CET LOG:  listening on IPv6 address ":::", port 5432
---2018-01-23 13:29:26 CET LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
---2018-01-23 13:29:26 CET LOG:  redirecting log output to logging collector
process
---2018-01-23 13:29:26 CET HINT:  Future log output will appear in directory
"log".
done
server started
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ ls -l /home/postgres/data/log/
total 4
-rw-----. 1 postgres postgres 170 23 janv. 13:29 postgresql-2018-01-
23_132926.log
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ cat /home/postgres/data/log/postgresql-2018-01-
23_132926.log
```

# Maintenance d'un serveur PostgreSQL

```
---2018-01-23 13:29:26 CET LOG: database system was shut down at 2018-01-23 13:29:09 CET
---2018-01-23 13:29:26 CET LOG: database system is ready to accept connections
[local]-[unknown]-[unknown]-2018-01-23 13:33:52 CET LOG: connection received: host=[local]
[local]-prod-postgres-2018-01-23 13:33:52 CET LOG: connection authorized: user=postgres database=prod
[local]-prod-postgres-2018-01-23 13:34:26 CET LOG: statement: create table emp2 as select * from emp;
[local]-prod-postgres-2018-01-23 13:34:26 CET LOG: duration: 137.047 ms
[local]-prod-postgres-2018-01-23 13:34:57 CET LOG: statement: insert into emp2 select * from emp;
[local]-prod-postgres-2018-01-23 13:34:57 CET LOG: duration: 150.660 ms
[local]-prod-postgres-2018-01-23 13:34:59 CET LOG: statement: insert into emp2 select * from emp;
[local]-prod-postgres-2018-01-23 13:35:00 CET LOG: duration: 296.713 ms
[local]-prod-postgres-2018-01-23 13:35:07 CET LOG: statement: insert into emp2 select * from emp;
[local]-prod-postgres-2018-01-23 13:35:08 CET LOG: duration: 979.653 ms
[local]-prod-postgres-2018-01-23 13:35:09 CET LOG: statement: insert into emp2 select * from emp;
[local]-prod-postgres-2018-01-23 13:35:12 CET LOG: duration: 2402.311 ms
[local]-prod-postgres-2018-01-23 13:35:12 CET LOG: statement: insert into emp2 select * from emp;
---2018-01-23 13:35:14 CET LOG: checkpoint starting: xlog
[local]-prod-postgres-2018-01-23 13:35:18 CET LOG: duration: 5848.066 ms
---2018-01-23 13:35:18 CET LOG: checkpoint complete: wrote 35658 buffers (92.9%); 0 WAL file(s) added, 0 removed, 0 recycled; write=4.203 s, sync=0.145 s, total=4.456 s; sync files=16, longest=0.055 s, average=0.008 s; distance=405024 kB, estimate=405024 kB
[local]-[unknown]-[unknown]-2018-01-23 13:36:34 CET LOG: connection received: host=[local]
[local]-prod-paul-2018-01-23 13:36:34 CET LOG: connection authorized: user=paul database=prod
[local]-prod-paul-2018-01-23 13:36:55 CET LOG: disconnection: session time: 0:00:21.698 user=paul
database=prod host=[local]
localhost(45430)-[unknown]-[unknown]-2018-01-23 13:37:06 CET LOG: connection received: host=localhost
port=45430
localhost(45430)-prod-paul-2018-01-23 13:37:06 CET LOG: connection authorized: user=paul database=prod
localhost(45430)-prod-paul-2018-01-23 13:37:50 CET ERROR: syntax error at or near ";" at character 24
localhost(45430)-prod-paul-2018-01-23 13:37:50 CET STATEMENT: create database test4 j;
localhost(45430)-prod-paul-2018-01-23 13:38:06 CET LOG: disconnection: session time: 0:00:59.720 user=paul
database=prod host=localhost port=45430
[local]-prod-postgres-2018-01-23 13:38:14 CET LOG: disconnection: session time: 0:04:21.540 user=postgres
database=prod host=[local]
---2018-01-23 13:38:20 CET LOG: received fast shutdown request
---2018-01-23 13:38:20 CET LOG: aborting any active transactions
---2018-01-23 13:38:20 CET LOG: worker process: logical replication launcher (PID 5672) exited with exit code 1
---2018-01-23 13:38:20 CET LOG: shutting down
---2018-01-23 13:38:20 CET LOG: checkpoint starting: shutdown immediate
---2018-01-23 13:38:20 CET LOG: checkpoint complete: wrote 19104 buffers (49.8%); 0 WAL file(s) added, 0 removed, 25 recycled; write=0.336 s, sync=0.372 s, total=0.731 s; sync files=2, longest=0.372 s, average=0.186 s; distance=211264 kB, estimate=385648 kB
---2018-01-23 13:38:20 CET LOG: database system is shut down
```

# Maintenance d'un serveur PostgreSQL

---



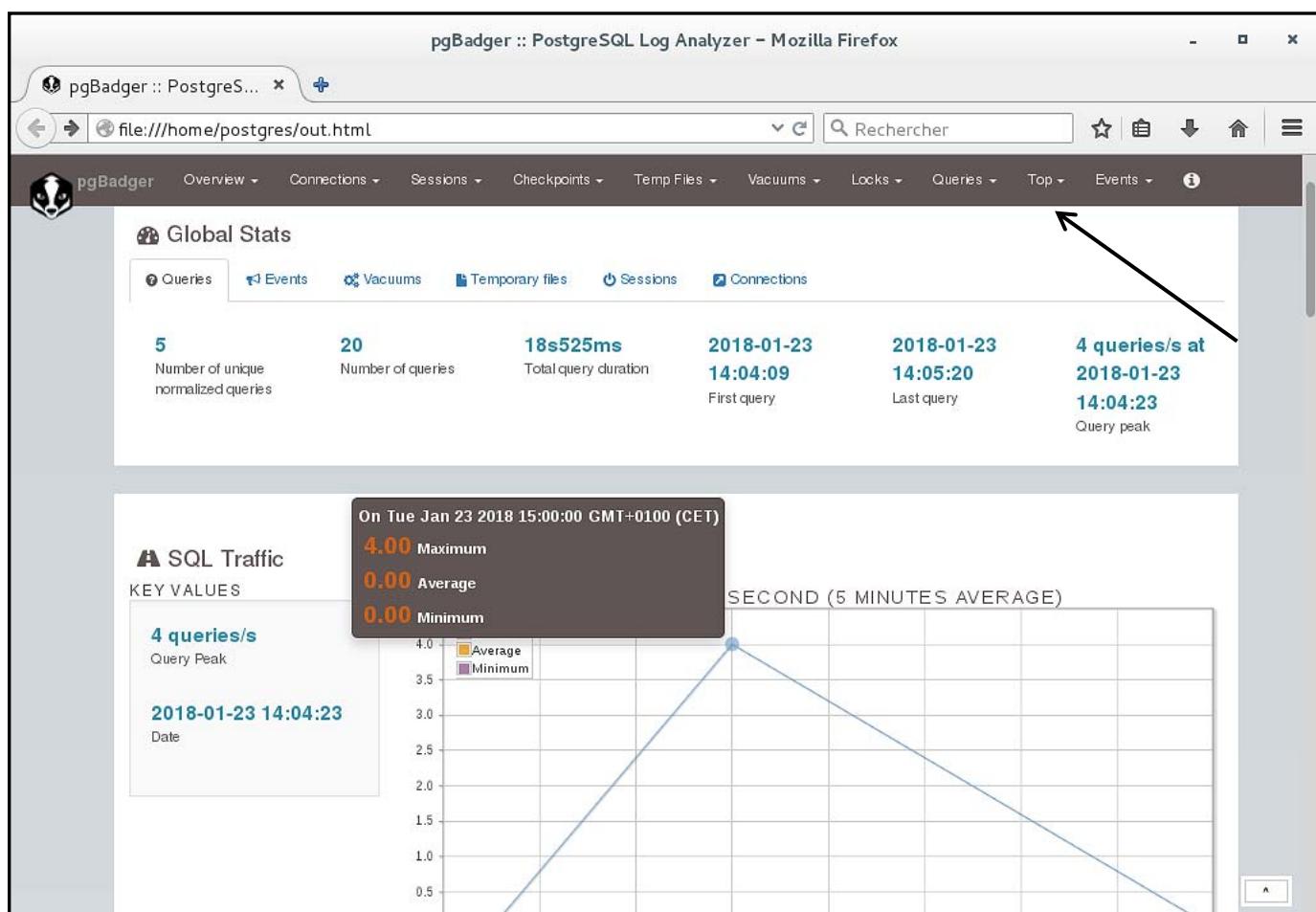
# Maintenance d'un serveur PostgreSQL

## 6. pgBadger

### A. Analyseur du fichier de log de PostgreSQL

**pgBadger** est un outil d'analyse rapide de fichiers de log d'un serveur PostgreSQL. C'est un script Perl développé par Dalibo (Gilles Darold) qui remplace avantageusement l'outil php **pgFouine**. **pgBadger** est un logiciel libre distribué sous licence BSD. **pgBadger** nécessite uniquement une version récente de Perl (5.6 minimum).

**pgBadger** génère des rapports détaillés à partir du contenu du fichier de log. Il peut être utilisé pour déterminer rapidement les requêtes les plus lourdes à optimiser, les requêtes les plus fréquentes ...



# Maintenance d'un serveur PostgreSQL

The screenshot shows the pgBadger :: PostgreSQL Log Analyzer interface. The main window displays a table titled "Most frequent queries (N)" with four rows of data. The columns are: Rank, Times executed, Total duration, Min duration, Max duration, Avg duration, and Query. The queries listed are:

Rank	Times executed	Total duration	Min duration	Max duration	Avg duration	Query
1	6	0ms	0ms	0ms	0ms	<code>INSERT INTO emp2 SELECT * FROM emp2;</code>
2	1	0ms	0ms	0ms	0ms	<code>SELECT n.nspname AS "schema", c.relname AS "name", CASE c.relkind WHEN 'r' THEN 'TABLE' WHEN 'v' THEN 'VIEW' WHEN 'i' THEN 'INDEX' WHEN 'S' THEN 'SEQUENCE' WHEN 'f' THEN 'FUNCTION' WHEN 'p' THEN 'PROCEDURE' END AS "type", pg_catalog.pg_get_userbyid (c.relpowner) AS "owner" FROM pg_catalog.pg_class c LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace WHERE c.relkind IN ('r','v','i','S') AND n.nspname &lt;&gt; '' AND n.nspname &lt;&gt; 'pg_temp' AND n.nspname !~ '^pg_.*' AND pg_catalog.pg_table_is_visible (c.oid) ORDER BY 0, 0;</code>
3	1	0ms	0ms	0ms	0ms	<code>DROP TABLE emp2;</code>
4	1	0ms	0ms	0ms	0ms	<code>SELECT n.nspname AS "schema", c.relname AS "name", CASE c.relkind WHEN 'r' THEN 'TABLE' WHEN 'v' THEN 'VIEW' WHEN 'i' THEN 'INDEX' WHEN 'S' THEN 'SEQUENCE' WHEN 'f' THEN 'FUNCTION' WHEN 'p' THEN 'PROCEDURE' END AS "type", pg_catalog.pg_get_userbyid (c.relpowner) AS "owner", pg_catalog.pg_size_pretty (pg_catalog.pg_table_size (c.oid)) AS "size", pg_catalog.obj_description (c.oid, '') AS "description" FROM pg_catalog.pg_class c LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace WHERE c.relkind IN ('r','v','i','S') AND n.nspname &lt;&gt; '' AND n.nspname &lt;&gt; 'pg_temp' AND n.nspname !~ '^pg_.*' AND pg_catalog.pg_table_is_visible (c.oid) ORDER BY 0, 0;</code>

## **8. PSQL ET PGADMIN 4**

---

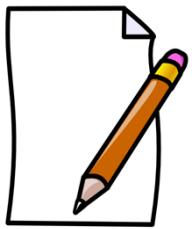
### **Objectif**

A la fin de ce module, vous aurez une vue d'ensemble des outils psql et pgAdmin 4 de PostgreSQL :

- psql
- pgAdmin 4

# psql et pgAdmin

---



# psql et pgAdmin

## 1. psql

### A. Outil client psql

C'est l'outil client de base le plus accessible (installé par défaut lors de l'installation de PostgreSQL). C'est un outil interactif en format ligne de commande vous permettant de vous connecter à un serveur PostgreSQL et d'exécuter des commandes SQL standards ou des commandes spécifiques de psql.

### B. Lancement de psql

Pour utiliser psql, vous devez ajouter le répertoire binaire de PostgreSQL (**exemple** : /usr/local/pgsql/bin) à votre variable d'environnement système PATH. Sous Windows: démarrer >> Programmes >> PostgreSQL 11.1 >> « psql » ou « invite de commandes ».

#### Exemple:

```
[postgres@CentOS764b ~]$ echo $PATH  
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/postgres/.local/bin:/home/postgres/bin:/usr/local/pgsql/bin  
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ psql template1  
psql (11.1)  
Type "help" for help.  
  
template1=#
```

### C. Utilisation de psql

Au démarrage de psql, vous avez un message contenant la version de l'outil client et éventuellement la version du serveur si celle-ci est différente. A la suite de ce message, vous avez l'invite de psql contenant le nom de base à laquelle vous êtes connectés.

# psql et pgAdmin

---

Toutes les commandes spécifiques à psql commencent par un anti-slash.

Il y a deux possibilités d'utiliser psql. La plus classique est la saisie d'une requête à l'invite de psql puis sa soumission (par enter) et la seconde est l'utilisation sous de batch en exécutant un ensemble de commandes SQL stockées dans un fichier O.S.

Avec la méthode classique, la requête peut être saisie sur plusieurs lignes et se termine par un point virgule.

## D. Commandes spécifiques de psql

La commande spécifique \? permet d'obtenir la liste des commandes spécifiques de psql.

# psql et pgAdmin

## Exemple :

```
template1=# \?

General
\copyright           show PostgreSQL usage and distribution terms
\crosstabview [COLUMNS] execute query and display results in crosstab
\errverbose          show most recent error message at maximum verbosity
\g [FILE] or ;      execute query (and send results to file or |pipe)
\gdesc               describe result of query, without executing it
\gexec               execute query, then execute each value in its result
\gset [PREFIX]        execute query and store results in psql variables
\gx [FILE]            execute query and store results in psql variables
                      as \g, but forces expanded output mode
\q                  quit psql
\watch [SEC]          execute query every SEC seconds

Help
\? [commands]        show help on backslash commands
\? options            show help on psql command-line options
\? variables          show help on special variables
\h [NAME]             help on syntax of SQL commands, * for all commands

Query Buffer
\e [FILE] [LINE]     edit the query buffer (or file) with external editor
\ef [FUNCNAME [LINE]] edit function definition with external editor
\ev [VIEWNAME [LINE]] edit view definition with external editor
\p                  show the contents of the query buffer
\q                  reset (clear) the query buffer
\sa [FILE]            display history or save it to file
\w FILE              write query buffer to file

Input/Output
\copy ...             perform SQL COPY with data stream to the client host
\echo [STRING]         write string to standard output
\i FILE               execute commands from file
\ir FILE              as \i, but relative to location of current script
\o [FILE]              send all query results to file or |pipe
\qecho [STRING]        write string to query output stream (see \o)

Conditional
\if EXPR              begin conditional block
\elseif EXPR           alternative within current conditional block
\else                 final alternative within current conditional block
\endif                end conditional block

Informational
(options: S = show system objects, + = additional detail)
\dt[S+]   NAME        list tables, views, and sequences
\da[S+]   [PATTERN]    describe table, view, sequence, or index
\daa[+]   [PATTERN]    list aggregates
\daA[+]   [PATTERN]    list access methods
\dtb[+]   [PATTERN]    list tablespaces
\dc[S+]   [PATTERN]    list conversions
\dcC[+]   [PATTERN]    list casts
\dd[S+]   [PATTERN]    show object descriptions not displayed elsewhere
\ddD[S+]   [PATTERN]    list domains
\ddp [PATTERN]        list default privileges
\de[S+]   [PATTERN]    list foreign tables
\det[+]   [PATTERN]    list foreign tables
\des[+]   [PATTERN]    list foreign servers
\deu[+]   [PATTERN]    list user mappings
\dew[+]   [PATTERN]    list foreign-data wrappers
```

# psql et pgAdmin

```
\df[antw] [S+] [PATRN]    list [only agg/normal/window] functions
\df[+] [PATTERN]          list text search configurations
\dfd[+] [PATTERN]         list text search dictionaries
\dfp[+] [PATTERN]         list text search parsers
\dft[+] [PATTERN]         list text search templates
\dg[S+] [PATTERN]         list roles
\di[S+] [PATTERN]         list indexes
.../...
.../...
\dp      [PATTERN]         list table, view, and sequence access privileges
\drds [PATRN1 [PATRN2]]   list per-database role settings
\drp[+] [PATTERN]          list replication publications
\drs[+] [PATTERN]          list replication subscriptions
\ds[S+] [PATTERN]          list sequences
\dt[S+] [PATTERN]          list tables
\dt[S+] [PATTERN]          list data types
\du[S+] [PATTERN]          list roles
\dv[S+] [PATTERN]          list views
\dx[+] [PATTERN]           list extensions
\dy      [PATTERN]          list event triggers
\l[+]  [PATTERN]           list databases
\sf[+] FUNCNAME            show a function's definition
\sv[+] VIEWNAME             show a view's definition
\z       [PATTERN]           same as \dp

Formatting
\a                  toggle between unaligned and aligned output mode
\c [STRING]          set table title, or unset if none
\f [STRING]          show or set field separator for unaligned query output
\H                  toggle HTML output mode (currently off)
\pset [NAME [VALUE]] set table output option
(NAME := {border|columns|expanded|fieldsep|fieldsep_zero|
footer|format|linestyle|null|numericlocale|pager|
pager_min_lines|recordsep|recordsep_zero|tableattr|title|
tuples_only|unicode_border_linestyle|
unicode_column_linestyle|unicode_header_linestyle})
\t [on|off]           show only rows (currently off)
\T [STRING]          set HTML <table> tag attributes, or unset if none
\x [on|off|auto]     toggle expanded output (currently off)

Connection
\c[onnect] {[DBNAME|-
USER|- HOST|- PORT|-]} | conninfo}
connect to new database (currently "template1")
\conninfo             display information about current connection
\encoding [ENCODING] show or set client encoding
\password [USERNAME] securely change the password for a user

Operating System
\cd [DIR]             change the current working directory
\setenv NAME [VALUE]  set or unset environment variable
\timing [on|off]       toggle timing of commands (currently off)
\! [COMMAND]          execute command in shell or start interactive shell

Variables
\prompt [TEXT] NAME   prompt user to set internal variable
\set [NAME [VALUE]]   set internal variable, or list all if no parameters
\unset NAME           unset (delete) internal variable

Large Objects
\lo_export LOBOID FILE
\lo_import FILE [COMMENT]
\lo_list
\lo_unlink LOBOID      large object operations
```

# psql et pgAdmin

## Principales commandes spécifiques de psql :

<b>\c[onnect] [DBNAME - [USER] - HOST - PORT  -]</b>	connexion à une autre base de données
<b>\password</b>	changer son mot de passe
<b>\conninfo</b>	informations sur la connexion courante
<b>\d</b>	liste les tables, vues et séquences
<b>\di</b>	liste les index
<b>\d NAME</b>	description d'un objet (table, index,vue ou séquence)
<b>\db[+]</b>	liste les tablespaces
<b>\du[+] [PATTERN]</b>	liste les utilisateurs
<b>\dp [PATTERN]</b>	liste les objets et leurs privilèges
<b>\dT [PATTERN]</b>	affiche les types de données
<b>\l[+]</b>	liste les bases de données de l'instance
<b>\h [NAME]</b>	aide sur la syntaxe d'une commande SQL
<b>\i FILE</b>	exécution de commandes SQL à partir d'un fichier
<b>\o FILE</b>	envoie le résultat des requêtes dans un fichier spool
<b>\q</b>	quitte psql
<b>\set [NAME [VALUE]]</b>	affichage du contenu des variables internes (\set) ou définition d'une variable interne (\set name value)
<b>\t</b>	affiche uniquement les lignes
<b>\timing</b>	affiche le temps d'exécution des requêtes
<b>\unset NAME</b>	suppression d'une variable interne
<b>\! [COMMAND]</b>	exécution de commandes O.S (\! ps -ef) ou passage dans le mode shell (\!) puis retour sous psql via exit
<b>\cd [DIR]</b>	changer la working directory courante au niveau de l'O.S

# psql et pgAdmin

## Exemples :

```
[postgres@CentOS764b ~]$ psql template1
psql (11.1)
Type "help" for help.

template1=# \conninfo
You are connected to database "template1" as user "postgres" via socket in "/tmp"
at port "5432".

template1=# \c prod admpaye
You are now connected to database "prod" as user "admpaye".
prod=> \d+
          List of relations
 Schema |   Name    | Type  | Owner |      Size      | Description
-----+-----+-----+-----+-----+-----+
 public | bonus    | table | admpaye | 0 bytes | 
 public | dept     | table | admpaye | 8192 bytes | 
 public | emp      | table | admpaye | 8192 bytes | 
 public | salgrade | table | admpaye | 8192 bytes | 
(4 rows)

prod=> \di+
          List of relations
 Schema |   Name    | Type  | Owner | Table | Size | Description
-----+-----+-----+-----+-----+-----+
 public | emp_ename | index | admpaye | emp   | 16 kB | 
(1 row)

prod=> \d dept
          Table "public.dept"
 Column |           Type           | Modifiers
-----+-----+-----+
 deptno | integer | 
 dname  | character varying(14) | 
 loc    | character varying(13) | 

prod=> \dp
          Access privileges
 Schema |   Name    | Type  | Access privileges | Column access privileges
-----+-----+-----+-----+-----+
 public | bonus    | table | 
 public | dept     | table | admpaye=arwdDxt/admpaye+ |
 |           |           | =r/admpaye | 
 public | emp      | table | admpaye=arwdDxt/admpaye+ |
 |           |           | paye=arw/admpaye + |
 |           |           | anne=d/admpaye | 
 public | salgrade | table | 
(4 rows)
```

# psql et pgAdmin

```
prod=> \du
                                         List of roles
Role name | Attributes                         | Member of
-----+-----+-----+
admpaye   |
anne       |
jean      |
paul       |
paye      | Cannot login
philippe   |
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | { }

prod=> \h CREATE VIEW
Command:      CREATE VIEW
Description: define a new view
Syntax:
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW name [ ( column_name [, ...] ) ]
          AS query

prod=> \! ls
data  data842  demotab.txt  fic1  serverlog  tbsp1  tbsp1_842

prod=> \! less fic1
select * from dept;

prod=> \i fic1
deptno | dname    | loc
-----+-----+-----+
  10 | ACCOUNTING | NEW YORK
  20 | RESEARCH   | DALLAS
  30 | SALES      | CHICAGO
  40 | OPERATIONS | BOSTON
(4 rows)

prod=> \o fic2
prod=> select * from dept;

prod=> \o
prod=> \! less fic2
deptno | dname    | loc
-----+-----+-----+
  10 | ACCOUNTING | NEW YORK
  20 | RESEARCH   | DALLAS
  30 | SALES      | CHICAGO
  40 | OPERATIONS | BOSTON
(4 rows)
```

# psql et pgAdmin

```
prod=> \t
Showing only tuples.
prod=> select * from dept;
  10 | ACCOUNTING | NEW YORK
  20 | RESEARCH   | DALLAS
  30 | SALES      | CHICAGO
  40 | OPERATIONS | BOSTON
prod=> \t
Tuples only is off.

prod=> \timing
Timing is on.
prod=> select * from dept;

  deptno | dname      | loc
-----+-----+-----
    10 | ACCOUNTING | NEW YORK
    20 | RESEARCH   | DALLAS
    30 | SALES      | CHICAGO
    40 | OPERATIONS | BOSTON
(4 rows)
Time: 1,964 ms

prod=> \timing
Timing is off.
prod=> \password
Enter new password:
Enter it again:

template1=# \set
AUTOCOMMIT = 'on'
COMP_KEYWORD_CASE = 'preserve-upper'
DBNAME = 'template1'
ECHO = 'none'
ECHO_HIDDEN = 'off'
ENCODING = 'UTF8'
FETCH_COUNT = '0'
HISTCONTROL = 'none'
HISTSIZE = '500'
HOST = '/tmp'
IGNOREEOF = '0'
LAST_ERROR_MESSAGE =
LAST_ERROR_SQLSTATE = '00000'
ON_ERROR_ROLLBACK = 'off'
ON_ERROR_STOP = 'off'
PORT = '5432'
PROMPT1 = '%/%R%# '
PROMPT2 = '%/%R%# '
PROMPT3 = '>> '
QUIET = 'off'
SERVER_VERSION_NAME = '11.1'
SERVER_VERSION_NUM = '110001'
SHOW_CONTEXT = 'errors'
SINGLELINE = 'off'
SINGLESTEP = 'off'
USER = 'postgres'
VERBOSITY = 'default'
VERSION = 'PostgreSQL 11.1 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623
(Red Hat 4.8.5-4), 64-bit'
VERSION_NAME = '11.1'
VERSION_NUM = '110001'

prod=> \!
[postgres@CentOS764b ~]$ ls
data data842 demotab.txt fic1 fic2 serverlog tbspl tbspl_842
[postgres@CentOS764b ~]$ exit
exit
prod=> \q
[postgres@CentOS764b ~]$
```

# psql et pgAdmin

## E. Options de lancement de psql

La syntaxe complète de psql est :

```
psql [ options ] [ dbname [ username ] ]
```

<b>dbname</b>	Paramètre optionnel indiquant explicitement la base de données à laquelle vous désirez vous connecter (équivaut aux options <b>-d dbname</b> ou <b>--dbname dbname</b> ). Si dbname n'est pas précisée, psql tente de se connecter à une base portant le même nom que l'utilisateur système qui a lancé psql.
<b>username</b>	Précise le compte utilisateur sous lequel vous vous connectez. Si user n'est pas précisé, psql tente de se connecter à la base sous un compte utilisateur portant le même nom que l'utilisateur système qui a lancé psql.
<b>options</b>	Il est possible de préciser une ou plusieurs options au lancement de psql.

Principales options de lancement de psql :

<b>-d dbname</b>	base de données à laquelle vous désirez vous connecter avec psql (idem <b>--dbname=dbname</b> )
------------------	---

# psql et pgAdmin

<b>-U <i>username</i></b>	Connexion sous le compte utilisateur indiqué (idem --username= <i>username</i> )
<b>-f <i>filename</i></b>	Pas de lancement de psql en interactif mais exécution par psql du contenu du fichier indiqué et psql se termine après ce (idem --file= <i>filename</i> )
<b>-c <i>command</i></b>	Pas de lancement de psql en interactif mais exécution par psql de la requête SQL indiquée (pas de commande spécifique psql) (idem command= <i>command</i> )
<b>-h <i>hostname</i></b>	Précise le nom de l'hôte sur lequel s'exécute le serveur PostgreSQL (inutile si le client et le serveur sont sur la même machine) (idem --host= <i>hostname</i> )
<b>-l</b>	Affiche la liste des bases de données disponibles (idem --list)
<b>-o <i>filename</i></b>	Redirige la sortie de psql vers le fichier indiqué (idem --output= <i>filename</i> )
<b>-p <i>port</i></b>	Précise le port TCP/IP sur lequel postgres est en attente de connexion. Sans cette option, on prend la valeur précisée par la variable système PGPORT (dont la valeur par défaut est 5432) (idem --port= <i>port</i> )
<b>-W</b>	Demande un mot de passe avant de se connecter à la base de données (idem --password)

# psql et pgAdmin

Exemples :

```
[postgres@CentOS764b ~]$ psql -d prod -U admpaye -c 'select * from dept' -W
Password for user admpaye:
deptno | dname      | loc
-----+-----+
 10  | ACCOUNTING | NEW YORK
 20  | RESEARCH   | DALLAS
 30  | SALES      | CHICAGO
 40  | OPERATIONS | BOSTON
(4 rows)

[postgres@CentOS764b ~]$ less fic1
select * from dept;

[postgres@CentOS764b ~]$ psql -d prod -U admpaye -f /home/postgres/fic1
deptno | dname      | loc
-----+-----+
 10  | ACCOUNTING | NEW YORK
 20  | RESEARCH   | DALLAS
 30  | SALES      | CHICAGO
 40  | OPERATIONS | BOSTON
(4 rows)

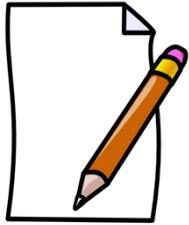
[postgres@CentOS764b ~]$ psql -d prod -U admpaye -f /home/postgres/fic1 -o
/home/postgres/result

[postgres@CentOS764b ~]$ less result
deptno | dname      | loc
-----+-----+
 10  | ACCOUNTING | NEW YORK
 20  | RESEARCH   | DALLAS
 30  | SALES      | CHICAGO
 40  | OPERATIONS | BOSTON
(4 rows)

[postgres@CentOS764b ~]$
```

# psql et pgAdmin

---



# psql et pgAdmin

---

## 2. pgAdmin 4

### A. Introduction

**pgAdmin 4** est un outil graphique d'administration des bases de données d'un serveur PostgreSQL. Il est le plus couramment utilisé parmi les outils d'administration de PostgreSQL (phpPgAdmin, SQL Manager, Druid...).

C'est un logiciel libre téléchargeable à partir du site [www.postgresql.org](http://www.postgresql.org).

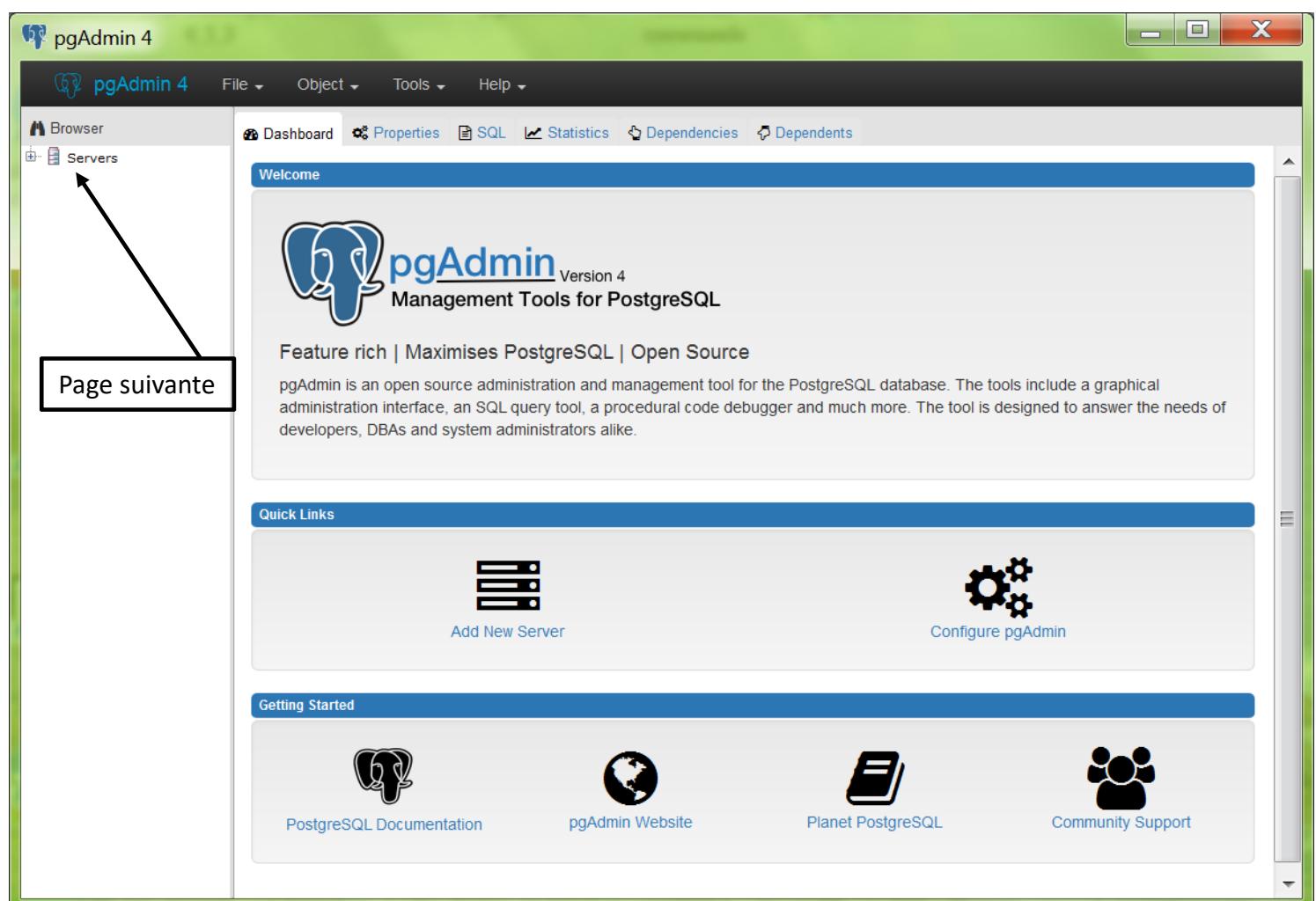
Il est disponible en plusieurs langues dont le français et utilisable sous Windows, Linux, FreeBSD, MAC OSX... pour administrer des serveurs PostgreSQL à partir de la version 7.3. Le(s) serveur(s) peut être local ou distant (sous Linux, Windows....).

**pgAdmin 4** peut convenir à différents types d'utilisateurs. Son interface graphique facilite l'administration et supporte toutes les caractéristiques de PostgreSQL.

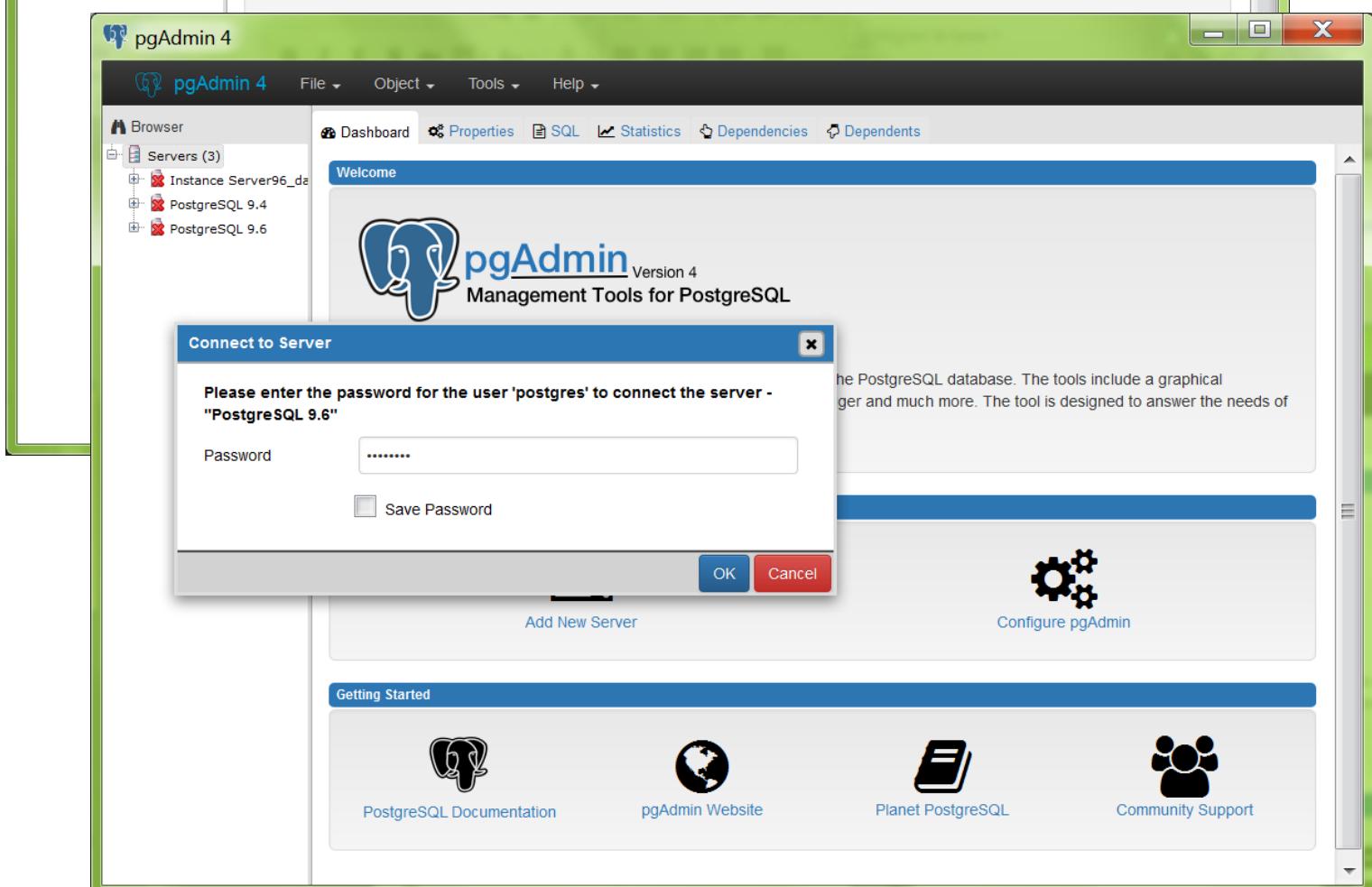
**pgAdmin 4** a été complètement réécrit en 2016 et largement amélioré par rapport **pgAdmin 3**. Il est disponible en standalone pour des utilisateurs individuels ou son code Web peut être déployé directement sur un serveur Web et utilisable par un ou plusieurs utilisateurs à travers un navigateur.

# psql et pgAdmin

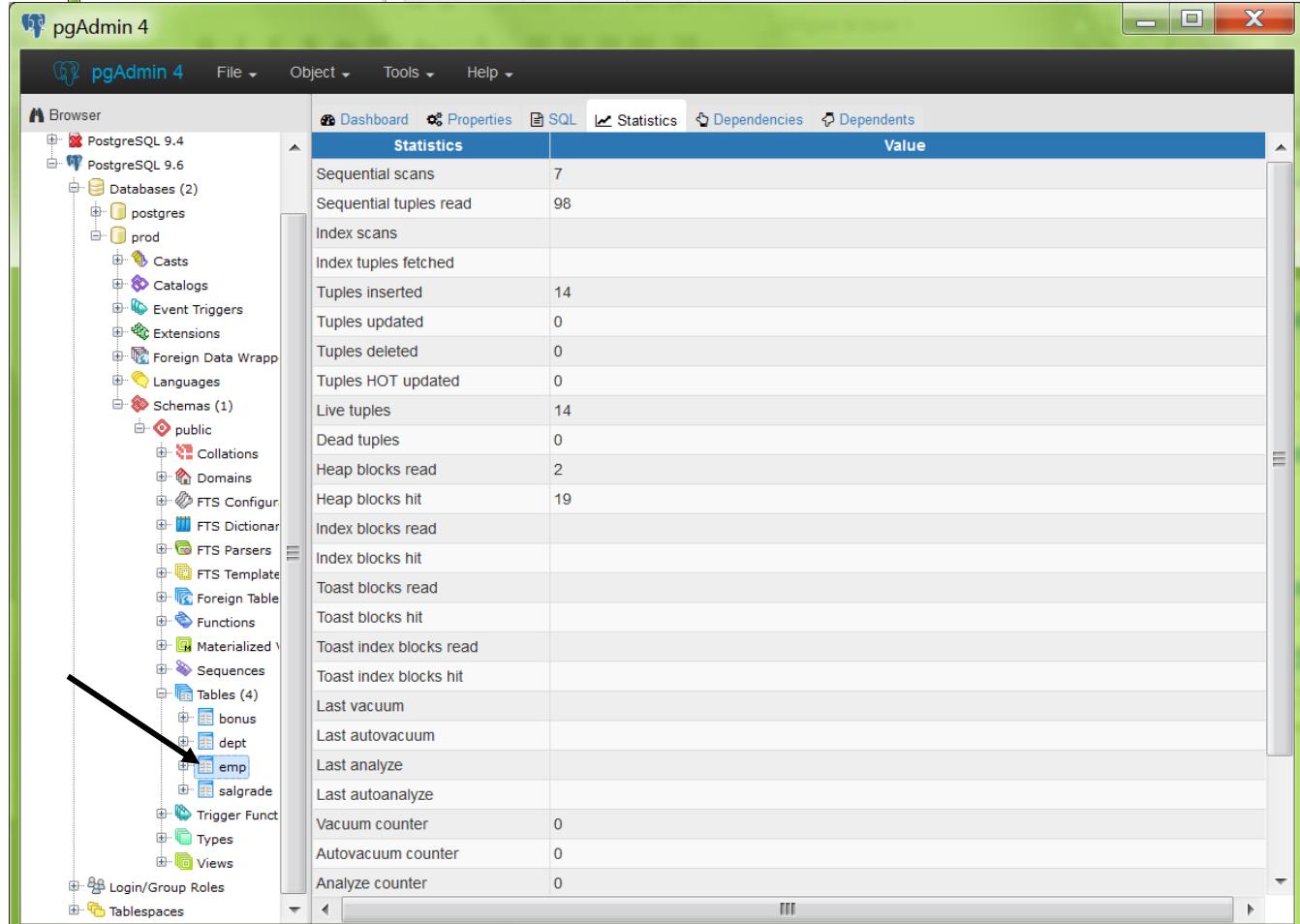
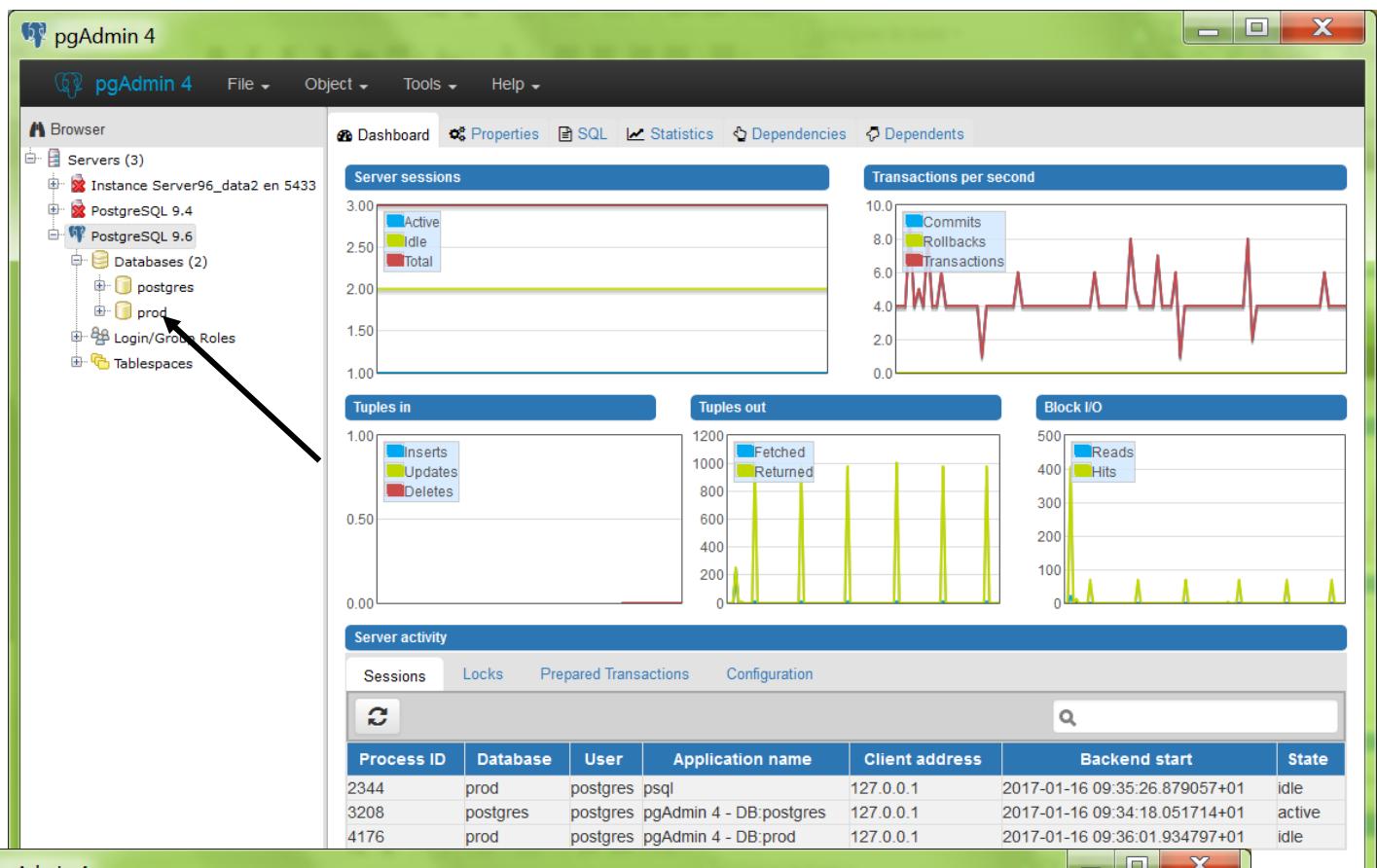
## B. Exemples de captures d'écrans



# psql et pgAdmin



# psql et pgAdmin



# psql et pgAdmin

The image consists of two screenshots of the pgAdmin 4 interface. The top screenshot shows the 'Tools' dropdown menu open, with the 'Query Tool' option selected. The bottom screenshot shows a right-click context menu over a SQL query in the 'Query Tool' window, with the 'Explain Analyze (Shift+F7)' option highlighted. A large black arrow points from the 'Tools' menu in the top screenshot down to the 'Explain Analyze' option in the bottom screenshot. A smaller black box labeled 'Page suivante' is located at the bottom right of the bottom screenshot.

pgAdmin 4

pgAdmin 4

Tools

Query Tool

Reload Configuration

Pause Replay of WAL

Resume Replay of WAL

Add Named Restore Point...

Import/Export...

Maintenance...

Backup...

Backup Globals...

Backup Server...

Restore...

Grant Wizard...

Heap blocks read 2

Heap blocks hit 19

Index blocks read

prod on postgres@PostgreSQL 9.6

1 select ename,loc from emp join dep

Execute/Refresh (F5)

Explain (F7)

Explain Analyze (Shift+F7) **Explain Options**

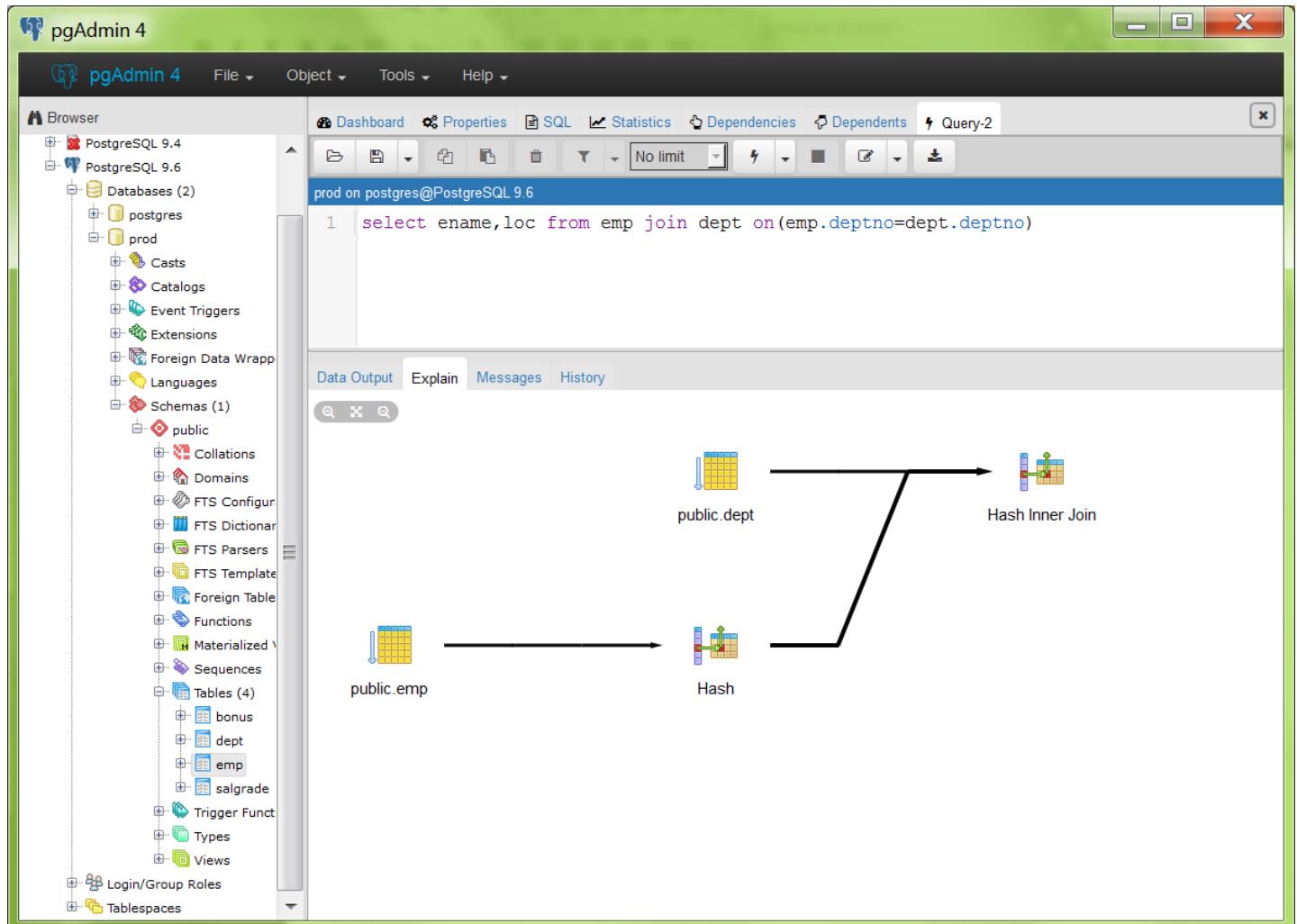
✓ Auto commit?

Auto rollback?

ename character... loc character...

ename	loc
MILLER	NEW YORK
KING	NEW YORK
CLARK	NEW YORK
FORD	DALLAS
ADAMS	DALLAS
SCOTT	DALLAS
JONES	DALLAS
SMITH	DALLAS
JAMES	CHICAGO
TURNER	CHICAGO
BLAKE	CHICAGO
MARTIN	CHICAGO
WARD	CHICAGO
ALLEN	CHICAGO

# psql et pgAdmin



## 9. TABLES ET INDEX

---

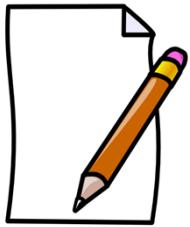
### Objectif

A la fin de ce module, vous aurez une vue d'ensemble des objets de type table et index d'un serveur PostgreSQL :

- Tables relationnelles
- Types de données
- Création, modification de la structure et suppression d'une table
- Index (types d'index, conseils...)
- Création et suppression, réorganisation d'un index
- Contraintes d'intégrité
- Types de contraintes
- Création et suppression d'une contrainte

# Tables et index

---



# Tables et index

---

## 1. Les tables

Les commandes SQL CREATE TABLE, ALTER TABLE, DROP TABLE permettent de créer, modifier la structure, détruire des tables.

Chaque colonne a un nom et un type de données délimitant l'ensemble des valeurs possibles assignables à cette colonne. Le nombre de lignes est variable.

Le langage SQL ne garantit pas l'ordre d'affichage des lignes d'une table sauf s'il est explicitement demandé (ORDER BY).

### A. Types de données

PostgreSQL supporte un grand nombre de types de données (built-in data types) et donne aussi la possibilité aux utilisateurs de créer leurs propres types de données.

La liste de tous les types de données et leurs caractéristiques (nom, alias, description, tailles consommées, plages de valeurs, limites...) sont dans le chapitre **8 « Data Types »** de la documentation PostgreSQL.

Les principales catégories de types de données sont les suivantes :

- Types caractères de longueur fixe ou variable (char, varchar...)
- Types numériques (numeric, integer, bigint...)
- Types dates (date, time, timestamp...)
- Types géométriques (point, polygon, circl ...)
- ...

# Tables et index

Liste des built-in data types extrait de la documentation :

Nom	Alias	Description
bigint	int8	Entier signé sur 8 octets
bigserial	serial8	Entier sur 8 octets à incrémentation automatique
bit [ (n) ]		Suite de bits de longueur fixe
bit varying [ (n) ]	varbit	Suite de bits de longueur variable
boolean	bool	Booléen (Vrai/Faux)
box		Boîte rectangulaire dans le plan
bytea		Donnée binaire (« tableau d'octets »)
character [ (n) ]	char [ (n) ]	Chaîne de caractères de longueur fixe
character varying [ (n) ]	varchar [ (n) ]	Chaîne de caractères de longueur variable
cidr		Adresse réseau IPv4 ou IPv6
circle		Cercle dans le plan
date		Date du calendrier (année, mois, jour)
double precision	float8	Nombre à virgule flottante de double précision (sur huit octets)
inet		Adresse d'ordinateur IPv4 ou IPv6
integer	int, int4	Entier signé sur 4 octets
interval [ champs ] [ (p) ]		Intervalle de temps
json		Données texte JSON
jsonb		Données binaires JSON, décomposées
line		Droite (infinie) dans le plan
lseg		Segment de droite dans le plan
macaddr		Adresse MAC (pour <i>Media Access Control</i> )
macaddr8		Adresse MAC (pour <i>Media Access Control</i> ) (format EUI-64)
money		Montant monétaire
numeric [ (p, s) ]	decimal [ (p, s) ]	Nombre exact dont la précision peut être précisée
path		Chemin géométrique dans le plan
pg_lsn		Séquence numérique de journal (Log Sequence Number)
point		Point géométrique dans le plan
polygon		Chemin géométrique fermé dans le plan
real	float4	Nombre à virgule flottante de simple précision (sur quatre octets)
smallint	int2	Entier signé sur 2 octets
smallserial	serial2	Entier sur 2 octets à incrémentation automatique
serial	serial4	Entier sur 4 octets à incrémentation automatique
text		Chaîne de caractères de longueur variable
time [ (p) ] [ without time zone ]		Heure du jour (pas du fuseau horaire)
time [ (p) ] with time zone	timetz	Heure du jour, avec fuseau horaire

Nom	Alias	Description
timestamp [ (p) ] [ without time zone ]		Date et heure (pas du fuseau horaire)
timestamp [ (p) ] with time zone	timestamptz	Date et heure, avec fuseau horaire
tsquery		requête pour la recherche plein texte
tsvector		document pour la recherche plein texte
txid_snapshot		image de l'identifiant de transaction au niveau utilisateur
uuid		identifiant unique universel
xml		données XML

# Tables et index

## Exemples de description de types de données extrait de la documentation :

### ▪ Types numériques

Nom	Taille de stockage	Description	Étendue
smallint	2 octets	entier de faible étendue	de -32768 à +32767
integer	4 octets	entier habituel	de -2147483648 à +2147483647
bigint	8 octets	grand entier	de -9223372036854775808 à +9223372036854775807
decimal	variable	précision indiquée par l'utilisateur, valeur exacte	jusqu'à 131072 chiffres avant le point décimal ; jusqu'à 16383 après le point décimal
numeric	variable	précision indiquée par l'utilisateur, valeur exacte	jusqu'à 131072 chiffres avant le point décimal ; jusqu'à 16383 après le point décimal
real	4 octets	précision variable, valeur inexacte	précision de 6 décimales
double precision	8 octets	précision variable, valeur inexacte	précision de 15 décimales
smallserial	2 bytes	Entier sur 2 octets à incrémentation automatique	1 to 32767
serial	4 octets	entier à incrémentation automatique	de 1 à 2147483647
bigserial	8 octets	entier de grande taille à incrémentation automatique	de 1 à 9223372036854775807

### ▪ Types caractères

Nom	Description
character varying( <i>n</i> ), varchar( <i>n</i> )	Longueur variable avec limite
character( <i>n</i> ), char( <i>n</i> )	longueur fixe, complété par des espaces
text	longueur variable illimitée

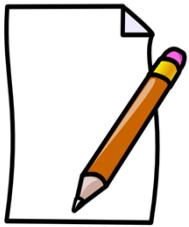
...

...

...

# Tables et index

---



# Tables et index

## B. Création d'une table

La commande SQL CREATE TABLE permet de créer une table. La commande doit au moins fournir le nom de la table et la description des colonnes (noms des colonnes et types des données).

```
template1=# \h create table
Command: CREATE TABLE
Description: define a new table
Syntax:
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ]
table_name ( [
{ column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
| table_constraint
| LIKE source_table [ like_option ... ] }
[, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ PARTITION BY { RANGE | LIST } ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] )
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]

CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ]
table_name
OF type_name [ (
{ column_name [ WITH OPTIONS ] [ column_constraint [ ... ] ]
| table_constraint }
[, ... ]
) ]
[ PARTITION BY { RANGE | LIST } ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] )
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]

CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ]
table_name
PARTITION OF parent_table [ (
{ column_name [ WITH OPTIONS ] [ column_constraint [ ... ] ]
| table_constraint }
[, ... ]
) ] FOR VALUES partition_bound_spec
[ PARTITION BY { RANGE | LIST } ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] )
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]

where column_constraint is:

.../...
.../...

prod=# CREATE TABLE EMPBIS
prod-#          (EMPNO INTEGER NOT NULL,
prod-#             ENAME VARCHAR(10),
prod-#             JOB VARCHAR(9),
prod-#             MGR INTEGER,
prod-#             HIREDATE DATE,
prod-#             SAL NUMERIC(7,2),
prod-#             COMM NUMERIC(7,2),
prod-#             DEPTNO INTEGER)
prod-# TABLESPACE TBSP1;
CREATE TABLE
```

# Tables et index

## Informations sur les tables :

```
prod=# \d+ empbis
                                         Table "public.empbis"
 Column |      Type       | Collation | Nullable | Default | Storage | Stats target | Description
-----+----------------+-----+-----+-----+-----+-----+-----+
empno  | integer       |           | not null |          | plain   |              |
ename   | character varying(10) |           |           |          | extended |              |
job    | character varying(9)  |           |           |          | extended |              |
mgr    | integer        |           |           |          | plain   |              |
hiredate | date         |           |           |          | plain   |              |
sal    | numeric(7,2)   |           |           |          | main    |              |
comm   | numeric(7,2)   |           |           |          | main    |              |
deptno | integer        |           |           |          | plain   |              |
Tablespace: "tbsp1"

prod=# select * from pg_tables where tablename='empbis';
          schemaname | tablename | tableowner | tablespace | hasindexes | hasrules | hastriggers | rowsecurity
-----+-----+-----+-----+-----+-----+-----+-----+
public | empbis   | postgres   | tbsp1     | f          | f        | f          | f

(1 row)

prod=# \d+
          List of relations
 Schema | Name | Type | Owner | Size | Description
-----+-----+-----+-----+-----+-----+
public | bonus | table | admpaye | 8192 bytes |
public | dept  | table | admpaye | 40 kB   |
public | emp   | table | admpaye | 14 MB   |
public | emp2  | table | postgres | 870 MB  |
public | empbis | table | postgres | 0 bytes  |
public | salgrade | table | admpaye | 48 kB   |
(6 rows)
```

# Tables et index

## C. Modification de la structure d'une table

La commande SQL ALTER TABLE permet de modifier la structure d'une table existante. Il est possible de :

- ajouter / supprimer une colonne
- ajouter / supprimer une contrainte
- changer le nom de la table ou le nom d'une colonne
- changer le propriétaire de la table
- ajouter / supprimer la valeur par défaut d'une colonne

```
prod=# \h alter table
Command:      ALTER TABLE
Description: change the definition of a table
Syntax:
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ]
    action [, ... ]
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ]
    RENAME [ COLUMN ] column_name TO new_column_name
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ]
    RENAME CONSTRAINT constraint_name TO new_constraint_name
ALTER TABLE [ IF EXISTS ] name
    RENAME TO new_name
ALTER TABLE [ IF EXISTS ] name
    SET SCHEMA new_schema
ALTER TABLE ALL IN TABLESPACE name [ OWNED BY role_name [, ... ] ]
    SET TABLESPACE new_tablespace [ NOWAIT ]
ALTER TABLE [ IF EXISTS ] name
    ATTACH PARTITION partition_name FOR VALUES partition_bound_spec
ALTER TABLE [ IF EXISTS ] name
    DETACH PARTITION partition_name

where action is one of:

    ADD [ COLUMN ] [ IF NOT EXISTS ] column_name data_type [ COLLATE collation ] [ column_constraint
[ ... ] ]
    DROP [ COLUMN ] [ IF EXISTS ] column_name [ RESTRICT | CASCADE ]
    ALTER [ COLUMN ] column_name [ SET DATA ] TYPE data_type [ COLLATE collation ] [ USING expression
]
    ALTER [ COLUMN ] column_name SET DEFAULT expression
    ALTER [ COLUMN ] column_name DROP DEFAULT
    ALTER [ COLUMN ] column_name { SET | DROP } NOT NULL
    ALTER [ COLUMN ] column_name ADD GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY [ (
sequence_options ) ]
    ALTER [ COLUMN ] column_name { SET GENERATED { ALWAYS | BY DEFAULT } | SET sequence_option |
RESTART [ [ WITH ] restart ] } [...]
    ALTER [ COLUMN ] column_name DROP IDENTITY [ IF EXISTS ]
    ALTER [ COLUMN ] column_name SET STATISTICS integer
    ALTER [ COLUMN ] column_name SET ( attribute_option = value [, ... ] )
    ALTER [ COLUMN ] column_name RESET ( attribute_option [, ... ] )
    ALTER [ COLUMN ] column_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
    ADD table_constraint [ NOT VALID ]
    ADD table_constraint_using_index
    ALTER CONSTRAINT constraint_name [ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY
IMMEDIATE ]
    VALIDATE CONSTRAINT constraint_name
    DROP CONSTRAINT [ IF EXISTS ] constraint_name [ RESTRICT | CASCADE ]
.../...
.../...
.../...
```

# Tables et index

```
SET TABLESPACE new_tablespace
SET { LOGGED | UNLOGGED }
SET ( storage_parameter = value [, ...] )
RESET ( storage_parameter [, ...] )
INHERIT parent_table
NO INHERIT parent_table
.../...
/
```

## Exemples :

```
prod=# alter table empbis add service varchar(10);
ALTER TABLE

prod=# \d empbis

Table "public.empbis"
 Column |      Type       | Collation | Nullable | Default
-----+---------------+-----+-----+-----+
 empno | integer       |           | not null |
 ename | character varying(10) |           |           |
 job   | character varying(9)  |           |           |
 mgr   | integer       |           |           |
 hiredate | date         |           |           |
 sal    | numeric(7,2)  |           |           |
 comm   | numeric(7,2)  |           |           |
 deptno | integer       |           |           |
 service | character varying(10) |           |           |
Tablespace: "tbsp1"

prod=# alter table empbis drop service;
ALTER TABLE
```

## D. Suppression d'une table

La commande SQL DROP TABLE permet de supprimer une table.

```
prod=# \h drop table
Command:     DROP TABLE
Description: remove a table
Syntax:
DROP TABLE name [, ...] [ CASCADE | RESTRICT ]
prod=# drop table empbis;
DROP TABLE
prod=#

```

# Tables et index

---

## E. Verrous et concurrence d'accès

Plusieurs transactions peuvent accéder simultanément à une même donnée. Afin d'obtenir un haut débit transactionnel, PostgreSQL garantit le partage des données (cohérence et consistance des données utilisées par les utilisateurs) en assurant une **gestion implicitement automatique des verrous** pour la plupart des commandes (INSERT, UPDATE, DELETE, ...) ou **explicitement** avec la requête SQL **LOCK** pour le verrouillage par une application dans des situations où la gestion implicite faite par MVCC ne convient pas.

Le mode d'isolation par défaut d'une transaction sous PostgreSQL est « **Read committed** ».

Par défaut, je vois les modifications effectuées dans ma transaction. Les autres transactions ne voient que les données « « commitées » » et ne voient pas les données en cours de modification (non « « commitées » ») par ma transaction. Seules les modifications des données effectuées par les transactions « « commitées » » sont visibles par les autres transactions.

Plusieurs versions concurrentes des lignes existent. À son démarrage, une transaction récupère la version cohérente au niveau base de données des lignes.

Une lecture ne bloque jamais une écriture et vice versa. L'écriture bloque tout autre écriture sur la ligne.

# Tables et index

---

Un verrou peut être exclusif (EXCLUSIVE) empêchant le partage des données verrouillées avec d'autres transactions et réservant exclusivement ces données à la transaction qui les verrouille. Un verrou peut aussi être partagé (SHARE) permettant à plusieurs transactions de se partager des lignes ce qui empêche une transaction de prendre l'exclusivité d'un objet. Un verrou peut avoir une portée ligne ou table. Les verrous sont posés pour toute la durée de la transaction.

PostgreSQL détecte automatiquement les situations d'attentes croisées de transactions (deadlock) et les résout en abordant une des transactions. Pour plus d'informations : documentation PostgreSQL chapitre **13 « Concurrency Control »**.

**SELECT** : Ne pose aucun verrou exclusif, ni au niveau ligne, ni au niveau table. Des requêtes en lecture (select) sont toujours possibles sans verrouillage intempestif de ligne ou de table.

**INSERT / UPDATE / DELETE** : La transaction a l'exclusivité des lignes concernées par mes INSERT, UPDATE, DELETE mais les autres transactions peuvent modifier les autres lignes.

**SELECT FOR UPDATE** : Pose d'un verrou exclusif au niveau ligne, accès possible en lecture sur la ou les lignes verrouillées mais pas en écriture pour les autres transactions, accès en lecture partageable et écriture sur toutes les autres lignes de la table. Ordre SQL généralement utilisé de manière implicite par les outils de développement. Libération des verrous au COMMIT de la transaction.

# Tables et index

---

## 2. Les index

### A. Généralités

Les index sont des objets optionnels associés aux tables, créés dans le but d'accélérer l'exécution des requêtes en fournissant un chemin plus rapide d'accès aux données d'une table.

Il est possible d'avoir plusieurs index pour une table.

La présence d'un index ne nécessite aucun changement dans l'écriture SQL d'une requête.

PostgreSQL utilisera l'index quand il jugera qu'il est plus efficace de passer par l'index que de faire un balayage complet de la table. Il est donc nécessaire de générer périodiquement des statistiques (ANALYZE) pour que l'optimiseur de requêtes puisse établir des plans d'exécution efficaces.

Un index peut être créé ou supprimé à n'importe quel moment sans affecter les tables concernées ou les autres index de ces tables concernées.

Les index augmentent la charge du système car PostgreSQL met à jour automatiquement les index pour refléter les insertions / modifications / suppressions de lignes dans les tables correspondantes.

Un index peut être « unique » (pas de doublons sur les valeurs indexées) ou « non-unique », mono ou multi (composites) colonnes.

Il existe plusieurs types d'index. Par défaut, PostgreSQL crée un index de type **B-tree**. C'est le type d'index le plus efficace dans la plupart des situations. PostgreSQL recommande d'utiliser ce type d'index.

# Tables et index

---

## B. Conseils pour l'indexation des tables

Créer un Index si vous recherchez fréquemment moins de 15% des lignes d'une grande table.

Indexer les colonnes fréquemment utilisées comme critères d'accès à une table (clause WHERE).

PostgreSQL crée automatiquement des index pour les contraintes d'intégrité PRIMARY et UNIQUE. Créer des index sur les FOREIGN KEY permettant des jointures plus rapides pour résoudre les relations maître-détail.

Indexer de préférence les colonnes possédant une bonne sélectivité ou des colonnes composites, les colonnes fréquemment citées en consultation.

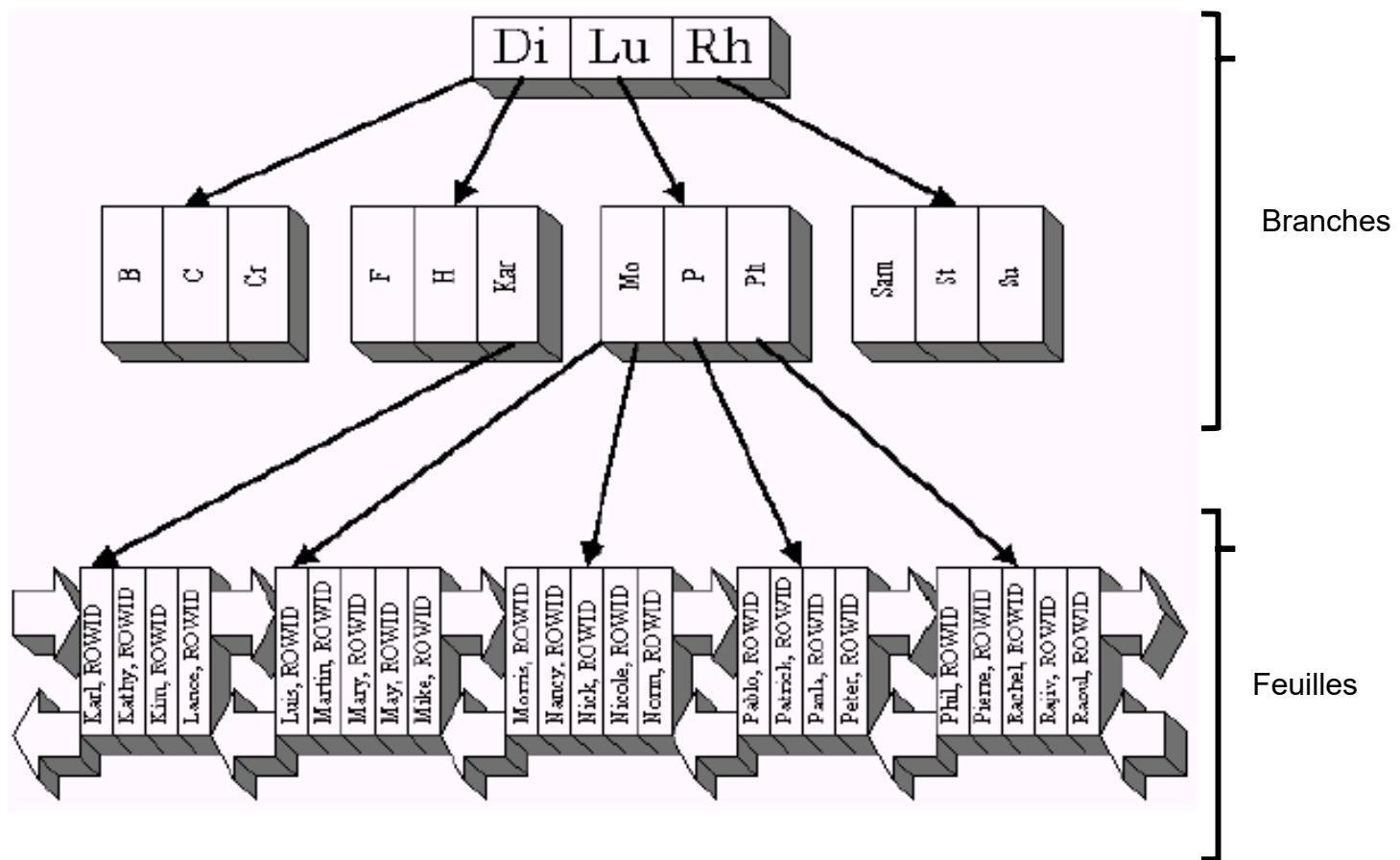
Éviter d'indexer les petites tables, les colonnes fréquemment modifiées, les colonnes peu sélectives, les tables avant leur chargement initial.

Il est conseillé de réorganiser périodiquement les index (REINDEX) qui « vivent » beaucoup.

...

# Tables et index

## C. Principe d'un index B-tree



### Avantages :

- même profondeur pour tous les blocs feuilles : accès approximativement identique pour toutes les valeurs
- accès rapide à une ligne bien identifiée ou à un groupe de lignes
- les valeurs de clé sont triées dans les blocs feuilles (pas les données indexées)

# Tables et index

## D. Création d'un index B-tree

La commande SQL CREATE INDEX permet de créer un index.

Exemple :

```
prod=# \h create index
Command:      CREATE INDEX
Description: define a new index
Syntax:
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] name ] ON table_name [
USING method ]
  ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC
] [ NULLS { FIRST | LAST } ] [, ...] )
  [ WITH ( storage_parameter = value [, ...] ) ]
  [ TABLESPACE tablespace_name ]
  [ WHERE predicate ]

test1=# create index emp_ename on emp(ename);
CREATE INDEX

test1=# \d emp
Table "public.emp"
 Column |          Type          | Collation | Nullable | Default
-----+---------------------+-----+-----+-----+
empno  | integer            |           | not null |
ename   | character varying(10) |           |           |
job     | character varying(9) |           |           |
mgr     | integer             |           |           |
hiredate | date               |           |           |
sal     | numeric(7,2)        |           |           |
comm    | numeric(7,2)        |           |           |
deptno  | integer             |           |           |
Indexes:
"emp_ename" btree (ename)

test1=# select * from pg_indexes where tablename='emp';
schemaname | tablename | indexname | tablespace | indexdef
-----+-----+-----+-----+-----+
public     | emp       | emp_ename |           | CREATE INDEX emp_ename ON emp USING
btree (ename)
(1 row)
```

# Tables et index

## E. Suppression d'un index

La commande SQL DROP INDEX permet de supprimer un ou plusieurs index.

### Suppression d'un index non utilisé :

```
test1=# select relname,indexrelname,idx_scan
test1-#         from pg_stat_all_indexes
test1-# where idx_scan=0 and relname not like 'pg_%';

   relname    |    indexrelname    |  idx_scan
-----+-----+-----+
 clients | clients_pk_numcli |      0
 commandes | commandes_pk_numcom |      0
 emp     | emp_ename        |      0
(3 rows)

test1=# drop index emp_ename;
DROP INDEX
test1=#

```

## F. Réorganisation d'un index

La commande SQL REINDEX reconstruit un index en utilisant les données de la table. Vous pouvez utiliser cette commande quand un index est désorganisé, corrompu ou pour récupérer de l'espace.

```
test1=# \h reindex
Command:      REINDEX
Description: rebuild indexes
Syntax:
REINDEX [ ( VERBOSE ) ] { INDEX | TABLE | SCHEMA | DATABASE | SYSTEM } name
test1=#

```

# Tables et index

---



## **10. SAUVEGARDES ET RESTAURATIONS**

---

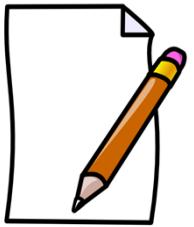
### **Objectif**

A la fin de ce module, vous aurez acquis les éléments nécessaires pour sauvegarder et restaurer de manière cohérente les données des bases de données d'un serveur PostgreSQL :

- Généralités
- Sauvegardes logiques
- Utilisation de pg\_dump, pg\_dumpall
- Restaurations logiques à partir d'un fichier texte et psql
- Restaurations logiques à partir d'un fichier binaire et pg\_restore
- Sauvegardes physiques serveur fermé
- Le mode archive (archivage des WAL)
- Sauvegardes physiques serveur ouvert
- Restauration complète ou partielle (PITR) du serveur.

# Sauvegardes et restauration

---



# Sauvegardes et restauration

---

## 1. Généralités

Une stratégie de sauvegarde est déterminée par de nombreux critères liés aux contraintes particulières de chaque client :

- la sécurité
- la disponibilité
- le type d'utilisation (OLTP, DSS, data warehouse...)
- la volumétrie de la base
- la charge (nombre d'utilisateurs)
- le budget
- ...

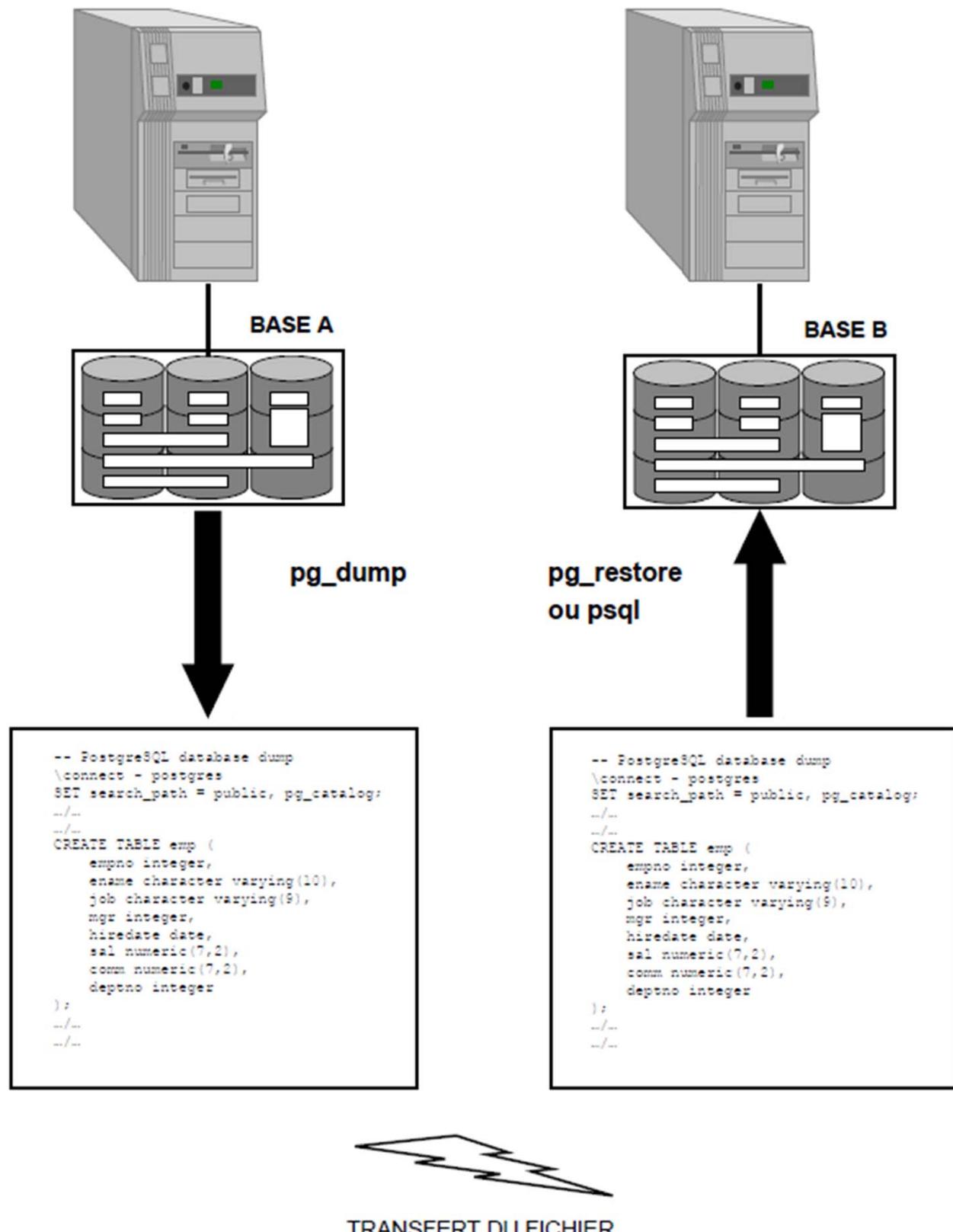
Les solutions matériels et systèmes pour éviter les pannes sont très efficaces (systèmes redondants, disques RAID, SAN, clusterisation).

En fonction de votre configuration matériel, il faut définir une stratégie de sauvegarde fiable pour les bases de données PostgreSQL, compatible avec les exigences de votre entreprise.

Il existe plusieurs approches différentes pour mettre au point cette stratégie de sauvegarde des données PostgreSQL :

- les sauvegardes « logiques » (génération d'un fichier texte SQL ou binaire),
- les sauvegardes physiques,
- le mode archive,
- site de secours, configuration H.A (Haute Disponibilité) appelée *warm standby* ou *log shipping*,
- réPLICATION synchrone / asynchrone.

# Sauvegardes et restauration



# Sauvegardes et restauration

---

## 2. Sauvegardes logiques

### A. Généralités

La sauvegarde logique (SQL dump) génère par défaut un fichier texte de commandes SQL qui peut être utilisé pour recréer une base de données dans l'état où elle était au moment de la sauvegarde logique.

**pg\_dump** est un outil client standard PostgreSQL connecté au serveur en tant que dba pour sauvegarder, en général, toute une base de données.

**pg\_dump** peut être utilisé en local ou à distance (options **-h host** et **-p port**). Comme tous les outils clients, **pg\_dump** se connecte par défaut à la base de données avec le même nom que celui utilisé au niveau de l'O.S. L'option **-U** ou la variable d'environnement **PGUSER** permettent de spécifier un autre nom d'utilisateur base de données pour se connecter. Les connexions de **pg\_dump** subissent les mêmes règles d'authentification que les autres connexions.

Les sauvegardes générées par **pg\_dump** (et pg\_dumpall) sont consistantes (mises à jours effectuées pendant l'exécution de **pg\_dump** ne seront pas dans la sauvegarde). **pg\_dump** peut s'exécuter en même temps que l'exploitation normale d'une base de données.

**pg\_dump** peut être un complément intéressant à vos sauvegardes physiques O.S.

### B. Cas d'utilisation

**pg\_dump / pg\_restore** sont des outils qui peuvent être utilisés pour migrer des bases de données, transférer des données d'une base de données PostgreSQL vers une autre.

Ils peuvent être utiles dans certains cas de **destructions logiques** (drop table..., delete from...) malencontreuses. Vous pouvez restaurer logiquement une table (précédemment sauvegardée) supprimée par accident. La table restaurée datera du moment de la sauvegarde.

Pour des considérations de performances, il est conseillé d'effectuer les sauvegardes logiques avec pg\_dump ou pg\_dumpall à des périodes où les bases de données sauvegardées sont faiblement utilisées.

# Sauvegardes et restauration

## C. Utilisation de pg\_dump

Syntaxe de la commande **pg\_dump** :

```
[postgres@CentOS764b ~]$ pg_dump --help
pg_dump dumps a database as a text file or to other formats.

Usage:
  pg_dump [OPTION]... [DBNAME]

General options:
  -f, --file=FILENAME          output file or directory name
  -F, --format=c|d|t|p          output file format (custom, directory, tar,
                               plain text (default))
  -j, --jobs=NUM                use this many parallel jobs to dump
  -v, --verbose                 verbose mode
  -V, --version                 output version information, then exit
  -Z, --compress=0-9             compression level for compressed formats
  --lock-wait-timeout=TIMEOUT   fail after waiting TIMEOUT for a table lock
  .../...
  .../...
```

### Principales options :

**-a ou --data-only**  
sauvegarde des données uniquement (sans la définition des structures). Le fichier de sauvegarde contient les requêtes COPY lues sur stdin et les données correspondantes ou les requêtes INSERT et les données correspondantes si l'option **--insert** est utilisée.

**-c --clean**  
génère les requêtes SQL pour supprimer les objets existants (drop) avant de générer les requêtes pour créer les objets et charger les données correspondantes.

**-C --create**  
génère en sortie le « create database » et la connexion à cette base au début de la sauvegarde.

**--inserts**  
génère en sortie les requêtes INSERT (plutôt que les requêtes COPY) et les données correspondantes.

**-f filename --file=filename**  
indique le nom du fichier de sortie (si omis, utilisation du stdout). L'utilisation du signe > est possible pour indiquer le nom du fichier de sortie.

# Sauvegardes et restauration

**-F** {c | d | t | p} --format {c | d | t | p}

Indique le format du fichier de sortie :

**c** (custom) crée un fichier archive en binaire compressé utilisable par pg\_restore

**d** (directory) produit une archive au format répertoire utilisable en entrée de pg\_restore. Cela créera un répertoire avec un fichier pour chaque table et blob exporté, ainsi qu'un fichier « Table des matières » décrivant les objets exportés dans un format machine que pg\_restore peut lire. Une archive au format répertoire peut être manipulée avec des outils Unix standard; par exemple, les fichiers d'une archive non-compressée peuvent être compressés avec l'outil gzip. Ce format est compressé par défaut et supporte les sauvegardes parallélisées.

**t** (tar) crée un fichier archive en binaire (tar) utilisable par pg\_restore.

**p** (plain-text) crée un fichier texte en sortie (valeur par défaut).

**-h *hostname***                           **--host=*hostname***

Nom du système sur lequel se trouve le serveur postgresql (par défaut local).

Numéro de port TCP/IP sur lequel écoute le serveur postgresql (par défaut 5432).

**-o** --no-owner

Ignore le propriétaire lors de la sauvegarde. Lors de la restauration, les objets appartiendront à l'utilisateur qui lancera la restauration.

**-s** --schema-only

Produit les requêtes SQL de définition des objets (DDL) sans les données correspondantes.

### Nom de la table à sauvegarder:

**-v** --verbose

Pour obtenir plus d'informations sur l'exécution de pg\_dump.

**-n schema**                   **--schema=schema**

### Nom du schéma à sauvegarder

Pour paralléliser le dump.

# Sauvegardes et restauration

## Exemples :

```
[postgres@CentOS764b ~]$ pg_dump prod -f proddump -C -v
pg_dump: last built-in OID is 16383
pg_dump: reading extensions
pg_dump: identifying extension members
pg_dump: reading schemas
pg_dump: reading user-defined tables
pg_dump: reading user-defined functions
pg_dump: reading user-defined types
pg_dump: reading procedural languages
pg_dump: reading user-defined aggregate functions
pg_dump: reading user-defined operators
pg_dump: reading user-defined access methods
pg_dump: reading user-defined operator classes
pg_dump: reading user-defined operator families
pg_dump: reading user-defined text search parsers
pg_dump: reading user-defined text search templates
pg_dump: reading user-defined text search dictionaries
pg_dump: reading user-defined text search configurations
pg_dump: reading user-defined foreign-data wrappers
pg_dump: reading user-defined foreign servers
pg_dump: reading default privileges
pg_dump: reading user-defined collations
pg_dump: reading user-defined conversions
pg_dump: reading type casts
pg_dump: reading transforms
pg_dump: reading table inheritance information
pg_dump: reading event triggers
pg_dump: finding extension tables
pg_dump: finding inheritance relationships
pg_dump: reading column info for interesting tables
pg_dump: finding the columns and types of table "public.emp"
pg_dump: finding the columns and types of table "public.dept"
pg_dump: finding the columns and types of table "public.bonus"
pg_dump: finding the columns and types of table "public.salgrade"
pg_dump: flagging inherited columns in subtables
pg_dump: reading indexes
pg_dump: reading indexes for table "public.emp"
pg_dump: flagging indexes in partitioned tables
pg_dump: reading extended statistics
pg_dump: reading constraints
pg_dump: reading triggers
pg_dump: reading rewrite rules
pg_dump: reading policies
pg_dump: reading row security enabled for table "public.emp"
pg_dump: reading policies for table "public.emp"
pg_dump: reading row security enabled for table "public.dept"
pg_dump: reading policies for table "public.dept"
pg_dump: reading row security enabled for table "public.bonus"
pg_dump: reading policies for table "public.bonus"
pg_dump: reading row security enabled for table "public.salgrade"
pg_dump: reading policies for table "public.salgrade"
pg_dump: reading publications
pg_dump: reading publication membership
```

# Sauvegardes et restauration

```
pg_dump: reading publication membership for table "public.emp"
pg_dump: reading publication membership for table "public.dept"
pg_dump: reading publication membership for table "public.bonus"
pg_dump: reading publication membership for table "public.salgrade"
pg_dump: reading subscriptions
pg_dump: reading large objects
pg_dump: reading dependency data
pg_dump: saving encoding = UTF8
pg_dump: saving standard_conforming_strings = on
pg_dump: saving search_path =
pg_dump: saving database definition
pg_dump: creating DATABASE "prod"
pg_dump: connecting to new database "prod"
pg_dump: creating TABLE "public.bonus"
pg_dump: creating TABLE "public.dept"
pg_dump: creating TABLE "public.emp"
pg_dump: creating TABLE "public.salgrade"
pg_dump: processing data for table "public.bonus"
pg_dump: dumping contents of table "public.bonus"
pg_dump: processing data for table "public.dept"
pg_dump: dumping contents of table "public.dept"
pg_dump: processing data for table "public.emp"
pg_dump: dumping contents of table "public.emp"
pg_dump: processing data for table "public.salgrade"
pg_dump: dumping contents of table "public.salgrade"
pg_dump: creating INDEX "public.emp_ename"
pg_dump: creating ACL "public.TABLE emp<<
[postgres@CentOS764b ~]$ 
[postgres@CentOS764b ~]$ ls -l|grep prod
-rw-rw-r--. 1 postgres postgres 4412 29 nov. 14:48 proddump
[postgres@CentOS764b ~]$ cat proddump
--
-- PostgreSQL database dump
--

-- Dumped from database version 11.1
-- Dumped by pg_dump version 11.1

-- Started on 2018-11-29 14:48:31 CET

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

--
-- TOC entry 3100 (class 1262 OID 16410)
-- Name: prod; Type: DATABASE; Schema: -; Owner: postgres
--

CREATE DATABASE prod WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE =
'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

ALTER DATABASE prod OWNER TO postgres;
```

# Sauvegardes et restauration

```
\connect prod

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_with_oids = false;

-- 
-- TOC entry 198 (class 1259 OID 16417)
-- Name: bonus; Type: TABLE; Schema: public; Owner: admpaye
--

CREATE TABLE public.bonus (
    ename character varying(10),
    job character varying(9),
    sal numeric,
    comm numeric
);

ALTER TABLE public.bonus OWNER TO admpaye;

-- 
-- TOC entry 197 (class 1259 OID 16414)
-- Name: dept; Type: TABLE; Schema: public; Owner: admpaye
--

CREATE TABLE public.dept (
    deptno integer,
    dname character varying(14),
    loc character varying(13)
);

ALTER TABLE public.dept OWNER TO admpaye;
```

# Sauvegardes et restauration

```
--  
-- TOC entry 196 (class 1259 OID 16411)  
-- Name: emp; Type: TABLE; Schema: public; Owner: admpaye  
  
CREATE TABLE public.emp (  
    empno integer NOT NULL,  
    ename character varying(10),  
    job character varying(9),  
    mgr integer,  
    hiredate date,  
    sal numeric(7,2),  
    comm numeric(7,2),  
    deptno integer  
);  
  
ALTER TABLE public.emp OWNER TO admpaye;  
  
--  
-- TOC entry 199 (class 1259 OID 16423)  
-- Name: salgrade; Type: TABLE; Schema: public; Owner: admpaye  
  
CREATE TABLE public.salgrade (  
    grade numeric,  
    losal numeric,  
    hisal numeric  
);  
  
ALTER TABLE public.salgrade OWNER TO admpaye;  
  
--  
-- TOC entry 3093 (class 0 OID 16417)  
-- Dependencies: 198  
-- Data for Name: bonus; Type: TABLE DATA; Schema: public; Owner: admpaye  
  
COPY public.bonus (ename, job, sal, comm) FROM stdin;  
\.
```

# Sauvegardes et restauration

```
--  
-- TOC entry 3092 (class 0 OID 16414)  
-- Dependencies: 197  
-- Data for Name: dept; Type: TABLE DATA; Schema: public; Owner: admpaye  
--  
  
COPY public.dept (deptno, dname, loc) FROM stdin;  
10      ACCOUNTING      NEW YORK  
20      RESEARCH        DALLAS  
30      SALES           CHICAGO  
40      OPERATIONS      BOSTON  
\.  
  
--  
-- TOC entry 3091 (class 0 OID 16411)  
-- Dependencies: 196  
-- Data for Name: emp; Type: TABLE DATA; Schema: public; Owner: admpaye  
--  
  
COPY public.emp (empno, ename, job, mgr, hiredate, sal, comm, deptno) FROM stdin;  
7369    SMITH    CLERK    7902    1980-12-17    800.00  \N      20  
7499    ALLEN    SALESMAN  7698     1981-02-20    1600.00 300.00  30  
7521    WARD     SALESMAN  7698     1981-02-22    1250.00 500.00  30  
7566    JONES    MANAGER   7839    1981-04-02    2975.00 \N      20  
7654    MARTIN   SALESMAN  7698     1981-09-28    1250.00 1400.00 30  
7698    BLAKE    MANAGER   7839    1981-05-01    2850.00 \N      30  
7782    CLARK    MANAGER   7839    1981-06-09    2450.00 \N      10  
7788    SCOTT    ANALYST   7566    1982-12-09    3000.00 \N      20  
7839    KING     PRESIDENT \N      1981-11-17    5000.00 \N      10  
7844    TURNER   SALESMAN  7698     1981-09-08    1500.00 0.00   30  
7876    ADAMS    CLERK    7788    1983-01-12    1100.00 \N      20  
7900    JAMES    CLERK    7698     1981-12-03    950.00  \N      30  
7902    FORD     ANALYST   7566    1985-12-07    3000.00 \N      20  
7934    MILLER   CLERK    7782    1999-04-15    1300.00 \N      10  
\.  
  
--  
-- TOC entry 3094 (class 0 OID 16423)  
-- Dependencies: 199  
-- Data for Name: salgrade; Type: TABLE DATA; Schema: public; Owner: admpaye  
--  
  
COPY public.salgrade (grade, losal, hisal) FROM stdin;  
1      700      1200  
2      1201     1400  
3      1401     2000  
4      2001     3000  
5      3001     9999  
\.
```

# Sauvegardes et restauration

---

```
--  
-- TOC entry 2969 (class 1259 OID 16431)  
-- Name: emp_ename; Type: INDEX; Schema: public; Owner: admpaye  
  
CREATE INDEX emp_ename ON public.emp USING btree (ename);  
  
--  
-- TOC entry 3101 (class 0 OID 0)  
-- Dependencies: 196  
-- Name: TABLE emp; Type: ACL; Schema: public; Owner: admpaye  
  
GRANT SELECT,INSERT,UPDATE ON TABLE public.emp TO paye;  
GRANT DELETE ON TABLE public.emp TO u1;  
  
-- Completed on 2018-11-29 14:48:31 CET  
  
--  
-- PostgreSQL database dump complete  
  
[postgres@CentOS764b ~]$
```

# Sauvegardes et restauration

---



# Sauvegardes et restauration

#### D. Utilisation de pg\_dumpall

Cet utilitaire permet de sauvegarder tout un cluster (serveur) de bases de données PostgreSQL. **pg\_dumpall** sauvegarde chaque base de données d'un cluster ainsi que les informations globales, communes à un cluster de base de données (les users, groups ...).

Certains paramètres de pg\_dump sont utilisables avec pg\_dumpall (-c,-d, -h, -p, -v ..). Ce paramètre est spécifique à **pg dumpall** :

**-g** --globals-only

Sauvegarde uniquement les objets globaux (users et groups).

**-t --tablespaces-only**

Sauvegarde uniquement les tablespaces, pas les bases de données ou les rôles.

**-r** --roles-only

Sauvegarde uniquement les rôles, pas les bases de données ou les tablespaces.

## Exemples :

```
[postgres@CentOS764b ~]$ pg_dumpall > dumpall_glo -g  
[postgres@CentOS764b ~]$ cat dumpall_glo  
--  
-- PostgreSQL database cluster dump  
--  
  
SET default_transaction_read_only = off;  
  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;
```

# Sauvegardes et restauration

```
--  
-- Roles  
--  
  
CREATE ROLE admpaye;  
ALTER ROLE admpaye WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN  
NOREPLICATION NOBYPASSRLS PASSWORD 'md5070720fa6eb244143187e21250e41c9d';  

```

# Sauvegardes et restauration

```
[postgres@CentOS764b ~]$ pg_dumpall > testdumpall
[postgres@CentOS764b ~]$ cat testdumpall
-- PostgreSQL database cluster dump
--
SET default_transaction_read_only = off;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;

--
-- Roles
--

CREATE ROLE admpaye;
ALTER ROLE admpaye WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN
NOREPLICATION NOBYPASSRLS PASSWORD 'md5070720fa6eb244143187e21250e41c9d';
CREATE ROLE anne;
ALTER ROLE anne WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION
NOBYPASSRLS PASSWORD 'md5e0b6ae475cd443e3c53866d6ff47b687';
CREATE ROLE jean;
ALTER ROLE jean WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION
NOBYPASSRLS PASSWORD 'md5522e6bf0be8955a0de9531dad8f01ccb';
CREATE ROLE patrick;
ALTER ROLE patrick WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN
NOREPLICATION NOBYPASSRLS PASSWORD 'SCRAM-SHA-
256$4096:PDkejqO2bP42iPbxNomnbA==$1TaQ85LrTZNlfafE+C10zhf7dTxXTDH54wB7JRUSks=:neKU/c5
0TuY4A971SpURsnZo90in8QQzoec/H0yrRX4=';
CREATE ROLE paul;
ALTER ROLE paul WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION
NOBYPASSRLS PASSWORD 'md52a6df21e82a3b929df6a7076c4be1d06';
CREATE ROLE paye;
ALTER ROLE paye WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB NOLOGIN NOREPLICATION
NOBYPASSRLS;
CREATE ROLE philippe;
ALTER ROLE philippe WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN
NOREPLICATION NOBYPASSRLS PASSWORD 'md53c7049523ea0357358eb5b35e25d371c';
CREATE ROLE postgres;
ALTER ROLE postgres WITH SUPERUSER INHERIT CREATEROLE CREATEDB LOGIN REPLICATION
BYPASSRLS;
CREATE ROLE u1;
ALTER ROLE u1 WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION
NOBYPASSRLS PASSWORD 'md58026a39c502750413402a90d9d8bae3c';
CREATE ROLE u2;
ALTER ROLE u2 WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION
NOBYPASSRLS PASSWORD 'md5a76d8c8015643c6a837661a10142016e';

--
-- Role memberships
--

GRANT paye TO anne GRANTED BY postgres;
GRANT paye TO jean GRANTED BY postgres;
GRANT paye TO u1 GRANTED BY postgres;
GRANT paye TO u2 GRANTED BY postgres;

--
-- Tablespaces
--

CREATE TABLESPACE tbsp1 OWNER postgres LOCATION '/home/postgres/tbsp1';
```

# Sauvegardes et restauration

```
\connect template1
--
-- PostgreSQL database dump
--
-- Dumped from database version 11.1
-- Dumped by pg_dump version 11.1
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;
--
-- PostgreSQL database dump complete
--

\connect postgres
--
-- PostgreSQL database dump
--
-- Dumped from database version 11.1
-- Dumped by pg_dump version 11.1

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;
--
-- PostgreSQL database dump complete
--

--
-- PostgreSQL database dump
--
-- Dumped from database version 11.1
-- Dumped by pg_dump version 11.1
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;
--
-- Name: prod; Type: DATABASE; Schema: -; Owner: postgres
--
CREATE DATABASE prod WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE =
'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

ALTER DATABASE prod OWNER TO postgres;

\connect prod
```

# Sauvegardes et restauration

```
.../...
.../...
.../...

-- PostgreSQL database dump complete
--


-- PostgreSQL database dump
--


-- Dumped from database version 11.1
-- Dumped by pg_dump version 11.1

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

-- Name: test3; Type: DATABASE; Schema: -; Owner: postgres
--


CREATE DATABASE test3 WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE =
'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

ALTER DATABASE test3 OWNER TO postgres;

\connect test3

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';
SET default_with_oids = false;

-- Name: t1; Type: TABLE; Schema: public; Owner: admpaye
--


CREATE TABLE public.t1 (
    col1 numeric
);

ALTER TABLE public.t1 OWNER TO admpaye;
```

# Sauvegardes et restauration

---

```
.../...
.../...
.../...

COPY public.t1 (col1) FROM stdin;
1111111111111
2222222222222
\.

-- 
-- Data for Name: t2; Type: TABLE DATA; Schema: public; Owner: admpaye
--

COPY public.t2 (col1) FROM stdin;
aaaa
bbbb
\.

-- 
-- Name: TABLE t1; Type: ACL; Schema: public; Owner: admpaye
--

GRANT SELECT ON TABLE public.t1 TO paye;

-- 
-- Name: TABLE t2; Type: ACL; Schema: public; Owner: admpaye
--

GRANT SELECT ON TABLE public.t2 TO PUBLIC;

-- 
-- PostgreSQL database dump complete
-- 

-- 
-- PostgreSQL database cluster dump complete
-- 

[postgres@CentOS764b ~]$
```

# Sauvegardes et restauration

---

## 3. Restaurations logiques

### A. Restaurations d'une base de données

Si la base de données a été sauvegardée dans un fichier **texte** par **pg\_dump**, vous pourrez utiliser ce fichier en infile de l'outil **psql** pour restaurer la base de données.

Si le fichier de sauvegarde est dans un autre format (tar ou tar compressé), vous utiliserez l'utilitaire **pg\_restore** pour restaurer la base de données.

Une base de données peut être restaurée vers une base vide ou non existante. Cela dépend de la façon dont elle a été sauvegardée (seules les données ont été sauvegardées où vous avez inclus les requêtes SQL pour créer la base et les objets).

### B. Restauration d'une base à partir d'un fichier texte avec l'outil psql

Le fichier texte créé par **pg\_dump** est prévu pour être utilisé en infile de **psql**. Suivant les options utilisées lors de la sauvegarde, **psql** pourra être appelé de différentes façons.

Si par exemple, l'option **-C** (ajoutant le `create database` dans la sauvegarde) a été utilisée lors de la sauvegarde, la requête SQL `create database` est incluse dans le fichier texte, ce qui signifie que la base de données a été supprimée ou qu'elle n'existe pas encore sur le système où vous allez faire votre restauration.

Si l'option **-C** n'a pas été utilisée lors de la sauvegarde, il faudra créer la base de données avant d'effectuer la restauration dans cette base vide.

# Sauvegardes et restauration

## Exemples :

```
[postgres@CentOS764b ~]$ pg_dump prod -f proddump -C
[postgres@CentOS764b ~]$ dropdb prod
[postgres@CentOS764b ~]$ psql template1 -f proddump
SET
SET
SET
SET
SET
  set_config
-----
(1 row)
SET
SET
SET
SET
CREATE DATABASE
ALTER DATABASE
You are now connected to database "prod" as user "postgres".
SET
SET
SET
SET
SET
  set_config
-----
(1 row)
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
COPY 0
COPY 4
COPY 14
COPY 5
CREATE INDEX
GRANT
GRANT
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# select * from dept;
   deptno |    dname     |      loc
-----+-----+-----+
      10 | ACCOUNTING | NEW YORK
      20 | RESEARCH   | DALLAS
      30 | SALES      | CHICAGO
      40 | OPERATIONS | BOSTON
(4 row)
```

# Sauvegardes et restauration

---

```
[postgres@CentOS764b ~]$ pg_dump prod -f proddump
[postgres@CentOS764b ~]$ dropdb prod
[postgres@CentOS764b ~]$ createdb prod
[postgres@CentOS764b ~]$ psql prod -f proddump
SET
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
COPY 0
COPY 4
COPY 14
COPY 5
CREATE INDEX
GRANT
GRANT
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# select * from dept;
   deptno |    dname     |      loc
-----+-----+-----+
      10 | ACCOUNTING | NEW YORK
      20 | RESEARCH   | DALLAS
      30 | SALES      | CHICAGO
      40 | OPERATIONS | BOSTON
(4 rows)
```

# Sauvegardes et restauration

---

## C. Restauration d'une base à partir d'un fichier tar avec pg\_restore

L'utilitaire **pg\_restore** permet de restaurer une base de données sauvegardée avec pg\_dump dans un fichier archive en format binaire (tar) ou binaire compressé. En fonction des paramètres utilisés lors de la sauvegarde avec pg\_dump, certains paramètres de pg\_restore ne seront pas utilisables.

Syntaxe de la commande **pg\_restore** :

```
pg_restore [ options... ]
```

### Principales options :

La plupart des options de **pg\_restore** reflètent celles de pg\_dump. Dans certains cas, si l'option est utilisée à pg\_dump, il faudra l'utiliser à pg\_restore pour que la fonctionnalité attendue soit possible. Autres options de pg\_restore:

**-I**                          ou                          **--list**

Liste un résumé du contenu de l'archive.

**-x**                          ou                          **no-acl**

N'effectue pas les requêtes GRANT et REVOKE éventuellement présentes dans le fichier de sauvegarde.

**-j**

Restauration parallélisée (depuis la 8.4).

...

**Remarque :** Si l'option -d n'est pas utilisée, pg\_restore affiche à l'écran les requêtes SQL de restauration au lieu de restaurer la base.

Si vous utilisez l'option -C pour créer (restaurer) une de vos bases de données à partir de rien, il faut utiliser l'option -d pour indiquer le nom d'une base initiale à laquelle se connecter (template1 par exemple). Le nom de cette base n'est pas important car il n'est utilisé que pour une connexion temporaire permettant de créer votre base à restaurer.

# Sauvegardes et restauration

## Exemples :

```
[postgres@CentOS764b ~]$ pg_dump prod -f proddump.tar -F t -C  
[postgres@CentOS764b ~]$ dropdb prod  
[postgres@CentOS764b ~]$ pg_restore proddump.tar  
--  
-- PostgreSQL database dump  
--  
  
-- Dumped from database version 11.1  
-- Dumped by pg_dump version 11.1  
  

```

# Sauvegardes et restauration

```
[postgres@CentOS764b ~]$ pg_restore -d template1 -C proddump.tar -v
pg_restore: connecting to database for restore
pg_restore: creating DATABASE "prod"
pg_restore: connecting to new database "prod"
pg_restore: connecting to database "prod" as user "postgres"
pg_restore: creating TABLE "public.bonus"
pg_restore: creating TABLE "public.dept"
pg_restore: creating TABLE "public.emp"
pg_restore: creating TABLE "public.salgrade"
pg_restore: processing data for table "public.bonus"
pg_restore: processing data for table "public.dept"
pg_restore: processing data for table "public.emp"
pg_restore: processing data for table "public.salgrade"
pg_restore: creating INDEX "public.emp_ename"
pg_restore: creating ACL "public.TABLE emp"
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# drop table dept ;
DROP TABLE
prod=# \q
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ pg_restore -d prod -t dept proddump.tar -v
pg_restore: connecting to database for restore
pg_restore: creating TABLE "public.dept"
pg_restore: processing data for table "public.dept"
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql prod
psql (11.1)
Type "help" for help.

prod=# select * from dept;
   deptno |    dname    |      loc
-----+-----+-----
      10 | ACCOUNTING | NEW YORK
      20 | RESEARCH   | DALLAS
      30 | SALES      | CHICAGO
      40 | OPERATIONS | BOSTON
(4 rows)

prod=#

```

# Sauvegardes et restauration

---

## 4. Sauvegardes physiques

### A. Principe

Toutes les informations manipulées par un serveur PostgreSQL sont stockées dans des fichiers du système d'exploitation (variable d'environnement PGDATA et éventuellement le ou les répertoires représentant vos tablespaces).

Donc l'autre stratégie de sauvegarde d'une base de données est de copier physiquement (avec des utilitaires physiques du système d'exploitation) les fichiers de données que PostgreSQL utilise pour stocker ses données.

Vous utiliserez, dans vos scripts systèmes, votre utilitaire physique préféré (tar, cp, cpio...) pour copier physiquement les fichiers composant votre serveur.

Pour effectuer une sauvegarde physique cohérente (ou une restauration physique) d'une instance PostgreSQL en mode **noarchive** (mode de fonctionnement par défaut), le serveur de bases de données PostgreSQL doit être **fermé**.

Extrait de la documentation officielle chapitre 25.2 « Sauvegarde de niveau système de fichiers »

Quiconque s'est aventuré dans les détails de l'organisation de la base de données peut être tenté de ne sauvegarder et restaurer que certaines tables ou bases de données particulières. Ce n'est *pas* utilisable sans les fichiers journaux de validation pg\_clog/\* qui contiennent l'état de la validation de chaque transaction. Un fichier de table n'est utilisable qu'avec cette information. Bien entendu, il est impossible de ne restaurer qu'une table et les données pg\_clog associées car cela rendrait toutes les autres tables du serveur inutilisables. Les sauvegardes du système de fichiers fonctionnent, de ce fait, uniquement pour les sauvegardes et restaurations complètes d'un cluster de bases de données.

Donc en cas de problème physique, il faudra restaurer l'ensemble du database cluster (plus éventuellement les tablespaces) et non pas des bouts de serveur.

**ATTENTION** : Toute l'activité postérieure à la sauvegarde restaurée est **perdue**.

# Sauvegardes et restauration

---

## Exemple :

```
[postgres@CentOS764b ~]$ echo $PGDATA  
/home/postgres/data  
[postgres@CentOS764b ~]$ pg_ctl status  
pg_ctl: server is running (PID: 3660)  
/usr/local/pgsql/bin/postgres  
[postgres@CentOS764b ~]$ pg_ctl stop  
waiting for server to shut down.... done  
server stopped  
[postgres@CentOS764b ~]$ tar -cf backup_data.tar /home/postgres/data  
tar: Suppression de « / » au début des noms des membres  
  
[postgres@CentOS764b ~]$ tar -cf backup_tbsp1.tar /home/postgres/tbsp1  
tar: Suppression de « / » au début des noms des membres  
  
[postgres@CentOS764b ~]$ ls -l *.tar  
  
-rw-rw-r--. 1 postgres postgres 1131499520 15 mars 15:37 backup_data.tar  
-rw-rw-r--. 1 postgres postgres 10240 15 mars 15:37 backup_tbsp1.tar  
-rw-rw-r--. 1 postgres postgres 23552 15 mars 15:13 testldump.tar  
[postgres@CentOS764b ~]$
```

# Sauvegardes et restauration

---

## B. Journalisation WAL et checkpoint

La journalisation **WAL** (Write Ahead Logging), appelée aussi journal des transactions, maintient à tout moment la consistance, l'intégrité et la fiabilité des données en cas d'incident logiciel (crash) ou matériel (destruction disque). Les fichiers journaux WAL de PostgreSQL (répertoire **\$PGDATA/pg\_xlog** jusqu'en 9.6 et **\$PGDATA/pg\_wal** en version 10) enregistrent séquentiellement les modifications effectuées sur les données par les transactions (insert, update...). Le contenu de ces journaux est utilisé lors des reprises après incident en rejouant le contenu de ces fichiers.

Le travail effectué par les transactions (insert, update...) et les données modifiées sont, dans un premier temps, enregistré séquentiellement dans les buffers WAL puis au plus tard au commit de la transaction, ces buffers sont écrits sur disque dans les fichiers WAL. Les blocs de données modifiés en shared buffers par cette transaction « committee » seront ensuite écrits sur disque au plus tard au prochain checkpoint.

Périodiquement, les buffers WAL sont écrits sur disque dans les fichiers WAL (au commit d'une transaction, buffers WAL pleins, lors d'un checkpoint...). Il est plus rapide d'écrire périodiquement et séquentiellement (nombre d'I/O réduit) dans les fichiers WAL le travail effectué par les transactions que d'écrire sur disque tous les blocs de données mis à jour par ces transactions. Par défaut, PostgreSQL force l'O.S à écrire immédiatement sur disque (pg\_xlog ou pg\_wal) les buffers WAL modifiés.

# Sauvegardes et restauration

---

Un checkpoint est un évènement pendant lequel le process background **writer** écrit sur disque tous les blocs de données modifiés (dirty) en mémoire partagée (**shared\_buffers**) qui n'ont pas encore été réécrits sur disque. Un checkpoint est un point de validation générale de l'instance. Des checkpoints sont déclenchés en fonction des paramètres d'instance suivants :

**min\_wal\_size** (le paramètre `checkpoint_segments` n'existe plus dans la version 9.5) : Tant que l'occupation disque reste sous la valeur de ce paramètre, les anciens fichiers WAL sont toujours recyclés pour une utilisation future lors des checkpoints, plutôt que supprimés. Ceci peut être utile pour s'assurer qu'un espace fichiers WAL suffisant est réservé pour faire face à des pics dans l'usage des WAL (valeur par défaut est **80 Mo**).

**max\_wal\_size** (le paramètre `checkpoint_segments` n'existe plus dans la version 9.5) : Taille maximale de l'augmentation des fichiers WAL entre deux checkpoints automatique des WAL. C'est une limite souple ; la taille des fichiers WAL peut excéder `max_wal_size` sous certaines circonstances, comme une surcharge du serveur, une commande `archive_command` qui échoue, ou une configuration haute pour `wal_keep_segments`. La valeur par défaut est **1 Go**.

**CHECKPOINT\_TIMEOUT** = nombre maximum de secondes entre 2 checkpoints automatiques. Par défaut **5 minutes**.

D'autres évènements peuvent provoquer des checkpoints (arrêt normal de la base, la commande SQL `CHECKPOINT...`).

# Sauvegardes et restauration

---

## C. Archivage des fichiers WAL et récupération à un instant donné (PITR)

Depuis la version **8.0** de PostgreSQL, il est possible d'archiver les fichiers WAL (paramètres du fichier `postgresql.conf` à configurer) en utilisant le mode **archive**.

Ce qui permet de garder un historique de tout ce qui a été fait par les transactions depuis la dernière sauvegarde physique. Si l'archivage n'est pas validé (mode `noarchive`), les fichiers WAL sont utilisés puis recyclés au fur et à mesure. Nous n'avons donc pas tout l'historique de ce qui a été fait par les transactions depuis la dernière sauvegarde physique.

Le mode archivage des journaux WAL (**mode archive**) offre une troisième stratégie de sauvegarde des bases de données qui permet :

- de restaurer l'instance PostgreSQL **sans perte de données** en rejouant tous les fichiers WAL archivés ou en s'arrêtant à un certain point dans le temps (**PITR**),
- d'effectuer des sauvegardes complètes **serveur ouvert** (à chaud) ou fermé (à froid),
- les fichiers WAL archivés produits par un serveur de production en mode archive peuvent être transférés en continu vers une autre machine chargée avec la même sauvegarde de niveau système de fichiers et obtenir ainsi un système de secours ou de reprise intermédiaire (**warm standby**)

Comme pour la sauvegarde physique serveur fermé, cette méthode ne supporte que la restauration d'un cluster de bases de données complet et pas d'une partie du cluster. Il faudra aussi prévoir une zone disque plus ou moins importante, qui dépend de l'activité transactionnelle de votre serveur, pour stocker les fichiers WAL archivés produits entre deux sauvegardes.

Cette solution est plus complexe à administrer que les autres solutions (`pg_dump`, sauvegarde O.S serveur fermé). Cette solution est à envisager quand le niveau de fiabilité ou de sécurité du serveur PostgreSQL est élevé.

A ce jour, la majorité des serveurs PostgreSQL ne sont pas configurés en mode archive.

# Sauvegardes et restauration

## D. Mise en œuvre du mode archive

Quand le mode **archive** est validé, les fichiers WAL doivent être copiés ailleurs avant de pouvoir être recyclés. Pour activer le mode archive, il faut positionner les paramètres suivants dans **postgresql.conf** :

```
wal_level = replica  
archive_mode = on  
archive_command = commande_shell_à_utiliser_pour_archiver
```

Pour fournir autant de flexibilité que possible à l'administrateur, PostgreSQL permet de préciser la commande Shell à exécuter pour copier le fichier WAL complet à l'endroit désiré (commande simple cp, rsync ... ou un script plus élaboré). Seule obligation : la commande doit renvoyer un code 0 uniquement si la copie a fonctionné.

Exemple :

```
[postgres@CentOS764b ~]$ nano data/postgresql.conf  
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ cat data/postgresql.conf |grep wal_level  
  
wal_level = replica  
  
[postgres@CentOS764b ~]$ cat data/postgresql.conf |grep archive_mode  
  
archive_mode = on  
  
[postgres@CentOS764b ~]$ cat data/postgresql.conf |grep archive_command  
  
archive_command = 'test ! -f /home/postgres/arch/%f && cp %p  
/home/postgres/arch/%f'  
  
[postgres@CentOS764b ~]$ mkdir /home/postgres/arch  
[postgres@CentOS764b ~]$
```

# Sauvegardes et restauration

```
[postgres@CentOS764b ~]$ pg_ctl start
waiting for server to start....2018-03-16 13:56:04 CET [7244]: [1-1] LOG:  listening on
IPv4 address "0.0.0.0", port 5432
2018-03-16 13:56:04 CET [7244]: [2-1] LOG:  listening on IPv6 address ":::", port 5432
2018-03-16 13:56:04 CET [7244]: [3-1] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5432"
2018-03-16 13:56:04 CET [7244]: [4-1] LOG:  redirecting log output to logging collector
process
2018-03-16 13:56:04 CET [7244]: [5-1] HINT:  Future log output will appear in directory
"log".
done
server started

[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: server is running (PID: 7244)
/usr/local/pgsql/bin/postgres

[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql test1
psql (11.1)
Type "help" for help.

test1=# create table t1 as select * from pg_class,pg_attribute;
SELECT 985726

test1=# ! ls -l /home/postgres/arch
total 311296
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FD
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FE
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FF
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000000
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000001
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000002
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000003
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000004
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000005
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000006
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000007
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000008
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000001000000009
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000A
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000B
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000C
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000D
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000E
-rw----- 1 postgres postgres 16777216 16 mars 13:56 0000000100000000100000000F
test1=#
```

# Sauvegardes et restauration

---

**Exemples comportant différentes manipulations à titre indicatif (exemples développés plus précisément dans la formation administration avancée) :**

- Sauvegarde serveur ouvert,
- Création de tables,
- Simulation de la destruction du serveur,
- Restauration complète du serveur PostgreSQL.

# Sauvegardes et restauration

- Sauvegarde serveur ouvert (à chaud) :

```
[postgres@CentOS764b ~]$ echo $PGDATA
/home/postgres/data
[postgres@CentOS764b ~]$ pg_ctl status
pg_ctl: server is running (PID: 7244)
/usr/local/pgsql/bin/postgres
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql test1
psql (11.1)
Type "help" for help.

test1=# select pg_start_backup('Sauvegarde online de bas niveau',true);

pg_start_backup
-----
1/11000028
(1 row)

test1=# ! ls -l /home/postgres/data|grep backup
-rw-----. 1 postgres postgres 221 16 mars 14:01 backup_label

test1=# ! cat /home/postgres/data/backup_label
START WAL LOCATION: 1/11000028 (file 000000010000000100000011)
CHECKPOINT LOCATION: 1/11000060
BACKUP METHOD: pg_start_backup
BACKUP FROM: master
START TIME: 2018-03-16 14:01:03 CET
LABEL: Sauvegarde online de bas niveau

test1=# \q
[postgres@CentOS764b ~]$ mkdir /home/postgres/sauvegardes
[postgres@CentOS764b ~]$ tar -cf /home/postgres/sauvegardes/sauve_PGDATA.tar $PGDATA
tar: Suppression de « / » au début des noms des membres
[postgres@CentOS764b ~]$ tar -cf /home/postgres/sauve_tbspl.tar
/home/postgres/tbspl
tar: Suppression de « / » au début des noms des membres
[postgres@CentOS764b ~]$
[postgres@CentOS764b ~]$ psql test1
psql (11.1)
Type "help" for help.

test1=# select pg_stop_backup();

NOTICE: pg_stop_backup complete, all required WAL segments have been archived
pg_stop_backup
-----
1/11000130
(1 row)

test1=# ! ls -l /home/postgres/arch | grep backup
-rw-----. 1 postgres postgres 318 16 mars 14:02
000000010000000100000011.00000028.backup

test1=# ! cat /home/postgres/arch/000000010000000100000011.00000028.backup

START WAL LOCATION: 1/11000028 (file 000000010000000100000011)
STOP WAL LOCATION: 1/11000130 (file 000000010000000100000011)
CHECKPOINT LOCATION: 1/11000060
BACKUP METHOD: pg_start_backup
BACKUP FROM: master
START TIME: 2018-03-16 14:01:03 CET
LABEL: Sauvegarde online de bas niveau
STOP TIME: 2018-03-16 14:02:39 CET
test1=#

```

# Sauvegardes et restauration

```
test1=# \! ls -l /home/postgres/data/pg_wal | grep backup
-rw----- 1 postgres postgres 318 16 mars 14:02
000000010000000100000011.00000028.backup

test1=# create table t2 as select * from pg_class,pg_attribute;
SELECT 1020175

test1=# \! ls -l /home/postgres/arch
total 671748
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FD
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FE
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000FF
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000001
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000002
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000003
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000004
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000005
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000006
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000007
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000008
-rw----- 1 postgres postgres 16777216 16 mars 13:56 000000010000000000000000100000009
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000A
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000B
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000C
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000D
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000E
-rw----- 1 postgres postgres 16777216 16 mars 13:56 00000001000000000000000010000000F
-rw----- 1 postgres postgres 16777216 16 mars 14:01 0000000100000000100000001000000010
-rw----- 1 postgres postgres 16777216 16 mars 14:02 0000000100000000100000001000000011
-rw----- 1 postgres postgres 318 16 mars 14:02
000000010000000100000011.00000028.backup
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000012
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000013
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000014
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000015
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000016
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000017
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000018
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000019
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001A
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001B
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001C
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001D
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001E
-rw----- 1 postgres postgres 16777216 16 mars 14:09 00000001000000010000001F
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000020
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000021
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000022
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000023
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000024
-rw----- 1 postgres postgres 16777216 16 mars 14:09 000000010000000100000025
test1=#

```

# Sauvegardes et restauration

- Simulation destruction du serveur PostgreSQL :

```
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ ps -ef|grep postgres  
  
root      3001  2875  0 10:12 pts/0    00:00:00 su - postgres  
postgres  3002  3001  0 10:12 pts/0    00:00:00 -bash  
postgres  7244     1  0 13:56 pts/0    00:00:00 /usr/local/pgsql/bin/postgres  
postgres  7245  7244  0 13:56 ?        00:00:00 postgres: logger process  
postgres  7247  7244  0 13:56 ?        00:00:00 postgres: checkpointer process  
postgres  7248  7244  0 13:56 ?        00:00:00 postgres: writer process  
postgres  7249  7244  0 13:56 ?        00:00:00 postgres: wal writer process  
postgres  7250  7244  0 13:56 ?        00:00:00 postgres: archiver process  last was  
000000010000000100000025  
postgres  7251  7244  0 13:56 ?        00:00:00 postgres: stats collector process  
postgres  7252  7244  0 13:56 ?        00:00:00 postgres: bgworker: logical replication  
launcher  
postgres  7547  3002  0 14:13 pts/0    00:00:00 ps -ef  
postgres  7548  3002  0 14:13 pts/0    00:00:00 grep --color=auto postgres  
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ kill -9 7244  
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ ps -ef|grep postgres  
  
root      3001  2875  0 10:12 pts/0    00:00:00 su - postgres  
postgres  3002  3001  0 10:12 pts/0    00:00:00 -bash  
postgres  7549  3002  0 14:14 pts/0    00:00:00 ps -ef  
postgres  7550  3002  0 14:14 pts/0    00:00:00 grep --color=auto postgres  
[postgres@CentOS764b ~]$  
[postgres@CentOS764b ~]$ mv /home/postgres/data /home/postgres/data_ko  
[postgres@CentOS764b ~]$ mv /home/postgres/tbspl /home/postgres/tbspl_ko  
[postgres@CentOS764b ~]$
```

# Sauvegardes et restauration

## ▪ Restauration complète du serveur PostgreSQL

```
[postgres@CentOS764b ~]$ cd /
[postgres@CentOS764b /]$ tar -xf /home/postgres/sauvegardes/sauve_PGDATA.tar
[postgres@CentOS764b /]$ tar -xf /home/postgres/sauvegardes/sauve_tbspl.tar
[postgres@CentOS764b /]$ echo $PGDATA
/home/postgres/data

[postgres@CentOS764b /]$ cd $PGDATA
[postgres@CentOS764b data]$ ls -l
total 80
-rw-----. 1 postgres postgres 221 16 mars 14:01 backup_label
drwx----- 9 postgres postgres 86 15 mars 15:20 base
-rw-----. 1 postgres postgres 44 16 mars 13:56 current_logfiles
drwx----- 2 postgres postgres 4096 16 mars 13:56 global
drwx----- 2 postgres postgres 4096 16 mars 13:56 log
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_commit_ts
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_dynshmem
-rw-----. 1 postgres postgres 4619 9 nov. 11:48 pg_hba.conf
-rw-----. 1 postgres postgres 1636 8 nov. 13:16 pg_ident.conf
drwx----- 4 postgres postgres 65 16 mars 14:01 pg_logical
drwx----- 4 postgres postgres 34 8 nov. 13:16 pg_multixact
drwx----- 2 postgres postgres 17 16 mars 13:56 pg_notify
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_replslot
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_serial
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_snapshots
drwx----- 2 postgres postgres 6 16 mars 13:56 pg_stat
drwx----- 2 postgres postgres 6 6 févr. 17:16 pg_stat_tmp
drwx----- 2 postgres postgres 17 8 nov. 13:16 pg_subtrans
drwx----- 2 postgres postgres 18 9 nov. 10:48 pg_tblspc
drwx----- 2 postgres postgres 6 8 nov. 13:16 pg_twophase
-rw-----. 1 postgres postgres 3 8 nov. 13:16 PG_VERSION
drwx----- 3 postgres postgres 4096 16 mars 14:01 pg_wal
drwx----- 2 postgres postgres 17 8 nov. 13:16 pg_xact
-rw-----. 1 postgres postgres 88 8 nov. 13:16 postgresql.auto.conf
-rw-----. 1 postgres postgres 22803 15 mars 16:16 postgresql.conf
-rw-----. 1 postgres postgres 30 16 mars 13:56 postmaster.opts
-rw-----. 1 postgres postgres 77 16 mars 13:56 postmaster.pid
-rw-r--r--. 1 postgres postgres 836 9 nov. 10:18 serverlog
-rw-----. 1 postgres postgres 27 16 mars 14:01 tablespace_map

[postgres@CentOS764b data]$ rm -f $PGDATA/pg_wal/* 
[postgres@CentOS764b data]$ rm -f $PGDATA/postmaster.pid
[postgres@CentOS764b data]$ ls /home/postgres/data_ko/pg_wal/
000000010000000100000011          000000010000000100000015  00000001000000010000001A
00000001000000010000001F          000000010000000100000024
000000010000000100000011.00000028.backup 000000010000000100000016  00000001000000010000001B
000000010000000100000020          000000010000000100000025
000000010000000100000012          000000010000000100000017  00000001000000010000001C
000000010000000100000021          000000010000000100000026
000000010000000100000013          000000010000000100000018  00000001000000010000001D
000000010000000100000022 archive_status
000000010000000100000014          000000010000000100000019  00000001000000010000001E
000000010000000100000023
[postgres@CentOS764b data]$
```

# Sauvegardes et restauration

```
[postgres@CentOS764b data]$ cp /home/postgres/data_ko/pg_wal/000000010000000100000026  
/home/postgres/data/pg_wal/  
[postgres@CentOS764b data]$  
[postgres@CentOS764b data]$ nano $PGDATA/recovery.conf  
[postgres@CentOS764b data]$ cat $PGDATA/recovery.conf  
  
restore_command = 'cp /home/postgres/arch/%f %p'  
  
[postgres@CentOS764b data]$  
[postgres@CentOS764b data]$ pg_ctl start  
waiting for server to start....2018-03-16 14:26:03 CET [7682]: [1-1] LOG:  listening on IPv4  
address "0.0.0.0", port 5432  
2018-03-16 14:26:03 CET [7682]: [2-1] LOG:  listening on IPv6 address ":::", port 5432  
2018-03-16 14:26:03 CET [7682]: [3-1] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"  
2018-03-16 14:26:03 CET [7682]: [4-1] LOG:  redirecting log output to logging collector  
process  
2018-03-16 14:26:03 CET [7682]: [5-1] HINT:  Future log output will appear in directory  
"log".  
done  
server started  
  
[postgres@CentOS764b data]$ pg_ctl status  
pg_ctl: server is running (PID: 7682)  
/usr/local/pgsql/bin/postgres  
  
[postgres@CentOS764b data]$  
[postgres@CentOS764b data]$ ls -l /home/postgres/data/log/  
total 780  
-rw----- 1 postgres postgres 3414 23 janv. 13:38 postgresql-2018-01-23_132926.log  
-rw----- 1 postgres postgres 7862 23 janv. 14:26 postgresql-2018-01-23_135623.log  
-rw----- 1 postgres postgres 5878 24 janv. 16:30 postgresql-2018-01-24_144300.log  
-rw----- 1 postgres postgres 28270 5 févr. 14:20 postgresql-2018-02-05_095620.log  
-rw----- 1 postgres postgres 46762 6 févr. 17:16 postgresql-2018-02-06_150312.log  
-rw----- 1 postgres postgres 657312 15 mars 15:36 postgresql-2018-03-15_141239.log  
-rw----- 1 postgres postgres 1767 15 mars 16:35 postgresql-2018-03-15_162120.log  
-rw----- 1 postgres postgres 3825 16 mars 10:49 postgresql-2018-03-16_102046.log  
-rw----- 1 postgres postgres 2878 16 mars 11:04 postgresql-2018-03-16_110158.log  
-rw----- 1 postgres postgres 2467 16 mars 11:28 postgresql-2018-03-16_111014.log  
-rw----- 1 postgres postgres 1796 16 mars 11:56 postgresql-2018-03-16_115624.log  
-rw----- 1 postgres postgres 3079 16 mars 11:58 postgresql-2018-03-16_115645.log  
-rw----- 1 postgres postgres 4012 16 mars 12:17 postgresql-2018-03-16_115843.log  
-rw----- 1 postgres postgres 2879 16 mars 13:55 postgresql-2018-03-16_135408.log  
-rw----- 1 postgres postgres 1509 16 mars 14:01 postgresql-2018-03-16_135604.log  
-rw----- 1 postgres postgres 3856 16 mars 14:26 postgresql-2018-03-16_142603.log  
[postgres@CentOS764b data]$
```

# Sauvegardes et restauration

```
[postgres@CentOS764b data]$ cat /home/postgres/data/log/postgresql-2018-03-16_142603.log

2018-03-16 14:26:03 CET [7684]: [1-1] LOG: database system was interrupted; last known up at 2018-03-16
14:01:03 CET
2018-03-16 14:26:03 CET [7684]: [2-1] LOG: starting archive recovery
2018-03-16 14:26:03 CET [7684]: [3-1] LOG: restored log file "000000010000000100000011" from archive
2018-03-16 14:26:03 CET [7684]: [4-1] LOG: redo starts at 1/11000028
2018-03-16 14:26:03 CET [7684]: [5-1] LOG: consistent recovery state reached at 1/11000130
2018-03-16 14:26:03 CET [7682]: [6-1] LOG: database system is ready to accept read only connections
2018-03-16 14:26:03 CET [7684]: [6-1] LOG: restored log file "000000010000000100000012" from archive
2018-03-16 14:26:03 CET [7684]: [7-1] LOG: restored log file "000000010000000100000013" from archive
2018-03-16 14:26:03 CET [7684]: [8-1] LOG: restored log file "000000010000000100000014" from archive
2018-03-16 14:26:03 CET [7684]: [9-1] LOG: restored log file "000000010000000100000015" from archive
2018-03-16 14:26:03 CET [7684]: [10-1] LOG: restored log file "000000010000000100000016" from archive
2018-03-16 14:26:04 CET [7684]: [11-1] LOG: restored log file "000000010000000100000017" from archive
2018-03-16 14:26:04 CET [7684]: [12-1] LOG: restored log file "000000010000000100000018" from archive
2018-03-16 14:26:04 CET [7684]: [13-1] LOG: restored log file "000000010000000100000019" from archive
2018-03-16 14:26:04 CET [7684]: [14-1] LOG: restored log file "00000001000000010000001A" from archive
2018-03-16 14:26:04 CET [7684]: [15-1] LOG: restored log file "00000001000000010000001B" from archive
2018-03-16 14:26:04 CET [7684]: [16-1] LOG: restored log file "00000001000000010000001C" from archive
2018-03-16 14:26:04 CET [7684]: [17-1] LOG: restored log file "00000001000000010000001D" from archive
2018-03-16 14:26:04 CET [7684]: [18-1] LOG: restored log file "00000001000000010000001E" from archive
2018-03-16 14:26:05 CET [7684]: [19-1] LOG: restored log file "00000001000000010000001F" from archive
2018-03-16 14:26:05 CET [7684]: [20-1] LOG: restored log file "000000010000000100000020" from archive
2018-03-16 14:26:05 CET [7684]: [21-1] LOG: restored log file "000000010000000100000021" from archive
2018-03-16 14:26:05 CET [7684]: [22-1] LOG: restored log file "000000010000000100000022" from archive
2018-03-16 14:26:05 CET [7684]: [23-1] LOG: restored log file "000000010000000100000023" from archive
2018-03-16 14:26:05 CET [7684]: [24-1] LOG: restored log file "000000010000000100000024" from archive
2018-03-16 14:26:05 CET [7684]: [25-1] LOG: restored log file "000000010000000100000025" from archive
cp: cannot stat '/home/postgres/arch/000000010000000100000026': No such file or directory
2018-03-16 14:26:05 CET [7684]: [26-1] LOG: invalid record length at 1/26A4C638: wanted 24, got 0
2018-03-16 14:26:05 CET [7684]: [27-1] LOG: redo done at 1/26A4C600
2018-03-16 14:26:05 CET [7684]: [28-1] LOG: last completed transaction was at log time 2018-03-16
14:09:51.771611+01
cp: cannot stat '/home/postgres/arch/000000010000000100000026': No such file or directory
cp: cannot stat '/home/postgres/arch/00000002.history': No such file or directory
2018-03-16 14:26:05 CET [7684]: [29-1] LOG: selected new timeline ID: 2
cp: cannot stat '/home/postgres/arch/00000001.history': No such file or directory
2018-03-16 14:26:06 CET [7684]: [30-1] LOG: archive recovery complete
2018-03-16 14:26:06 CET [7686]: [1-1] LOG: checkpoint starting: end-of-recovery immediate wait
2018-03-16 14:26:08 CET [7686]: [2-1] LOG: checkpoint complete: wrote 38196 buffers (99.5%); 0 WAL
file(s) added, 0 removed, 0 recycled; write=2.638 s, sync=0.155 s, total=2.807 s; sync files=21,
longest=0.081 s, average=0.007 s; distance=354609 kB, estimate=354609 kB
2018-03-16 14:26:08 CET [7682]: [7-1] LOG: database system is ready to accept connections
[postgres@CentOS764b data]$
```

# Sauvegardes et restauration

```
[postgres@CentOS764b data]$ ll
total 84
-rw-----. 1 postgres postgres 221 16 mars 14:01 backup_label.old
drwx-----. 9 postgres postgres 86 15 mars 15:20 base
-rw-----. 1 postgres postgres 44 16 mars 14:26 current_logfiles
drwx-----. 2 postgres postgres 4096 16 mars 14:26 global
drwx-----. 2 postgres postgres 4096 16 mars 14:26 log
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_commit_ts
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_dynshmem
-rw-----. 1 postgres postgres 4619 9 nov. 11:48 pg_hba.conf
-rw-----. 1 postgres postgres 1636 8 nov. 13:16 pg_ident.conf
drwx-----. 4 postgres postgres 65 16 mars 14:26 pg_logical
drwx-----. 4 postgres postgres 34 8 nov. 13:16 pg_multixact
drwx-----. 2 postgres postgres 17 16 mars 14:26 pg_notify
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_replslot
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_serial
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_snapshots
drwx-----. 2 postgres postgres 6 16 mars 13:56 pg_stat
drwx-----. 2 postgres postgres 6 6 févr. 17:16 pg_stat_tmp
drwx-----. 2 postgres postgres 17 8 nov. 13:16 pg_subtrans
drwx-----. 2 postgres postgres 18 16 mars 14:26 pg_tblspc
drwx-----. 2 postgres postgres 6 8 nov. 13:16 pg_twophase
-rw-----. 1 postgres postgres 3 8 nov. 13:16 PG_VERSION
drwx-----. 3 postgres postgres 4096 16 mars 14:26 pg_wal
drwx-----. 2 postgres postgres 17 8 nov. 13:16 pg_xact
-rw-----. 1 postgres postgres 88 8 nov. 13:16 postgresql.auto.conf
-rw-----. 1 postgres postgres 22803 15 mars 16:16 postgresql.conf
-rw-----. 1 postgres postgres 30 16 mars 14:26 postmaster.opts
-rw-----. 1 postgres postgres 77 16 mars 14:26 postmaster.pid
-rw-rw-r--. 1 postgres postgres 49 16 mars 14:25 recovery.done
-rw-rw-r--. 1 postgres postgres 836 9 nov. 10:18 serverlog
-rw-----. 1 postgres postgres 27 16 mars 14:01 tablespace_map.old
[postgres@CentOS764b data]$
[postgres@CentOS764b data]$ psql test1
psql (10.0)
Type "help" for help.

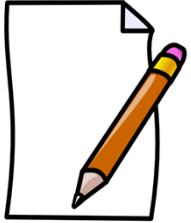
test1=# \d
      List of relations
 Schema |   Name    | Type  | Owner
-----+-----+-----+
 public | bonus    | table | postgres
 public | clients  | table | postgres
 public | commandes | table | postgres
 public | dept    | table | postgres
 public | emp     | table | postgres
 public | salgrade | table | postgres
 public | t1      | table | postgres
 public | t2      | table | postgres
 public | tabpartrange | table | postgres
 public | tabpartrange_r1 | table | postgres
 public | tabpartrange_r2 | table | postgres
(11 rows)

test1=#

```

# Sauvegardes et restauration

---



## 11. OPTIMISATION

---

### Objectif

Vous donner une introduction à l'optimisation de PostgreSQL :

- Introduction
- Exécution d'une requête
- Lecture d'un plan d'exécution (commande EXPLAIN, opérations élémentaires)
- Principaux paramètres d'optimisation liés à la mémoire, espace mémoire consommé, conseils
- Contrôle de l'activité et statistiques du serveur
- Contrib pg\_stat\_statements
- Contrib pgbench

# Optimisation

---



# Optimisation

---

## 1. Introduction

### A. Mise à jour des statistiques de l'optimiseur

Comme cela a été évoqué dans le module « Maintenance d'un serveur PostgreSQL », l'optimiseur de PostgreSQL dépend des informations statistiques récoltées sur les tables pour optimiser les requêtes. Ces statistiques sont générées par la commande **ANALYZE** qui peut être invoquée seule ou couplée à l'opération **VACUUM**.

Il est important d'avoir des statistiques pertinentes sinon les « mauvais » plans d'optimisation des requêtes peuvent dégrader les performances de votre base de données.

**VACUUM** et **ANALYZE** mettent à jour les tables systèmes :

- **pg\_class** (reltuples et relpages)
- **pg\_statistic** utilisé en interne par PostgreSQL. La vue **pg\_stats** a été créée pour fournir un accès sécurisé et plus lisible aux données statistiques de la table système **pg\_statistic**. La vue **pg\_stats** affiche des données statistiques sur les tables et les valeurs d'index avec une entrée pour chaque colonne (avg\_width, n\_distinct, most\_common\_vals, most\_common\_freqs...)

### B. Exécution d'une requête

Lorsqu'un client soumet une requête SQL sur un serveur PostgreSQL, plusieurs opérations sont effectuées.

- 1/ Traduction (parsing) de la requête (vérification syntaxique, contrôle des objets (tables, colonnes...) et des priviléges d'accès à ces objets, création d'un « parse tree »).
- 2/ L'optimiseur analyse le parse tree, génère des plans d'exécution et choisit le meilleur.
- 3/ Exécution de la requête.
- 4/ Envoie du résultat au client.

# Optimisation

---



# Optimisation

---

## 2. Lecture d'un plan d'exécution

### A. La commande EXPLAIN

Il est possible de visualiser le plan d'exécution d'une requête retenu par l'optimiseur avec la commande SQL « **EXPLAIN** ».

Le plan d'exécution récupéré par la commande **EXPLAIN** indique comment les objets référencés dans la requête seront accédés (lecture séquentielle, parcours d'index...) et quels algorithmes de jointure seront utilisés.

Les coûts mesurés sont en unités récupération de bloc disque. Un coût de 1 correspond à une lecture d'un bloc disque.

Les résultats renvoyés par EXPLAIN sont :

- Les opérations élémentaires pour exécuter la requête
- Une estimation du coût d'exécution avant que la première ligne ne soit renvoyée
- Le coût total estimé pour renvoyer toutes les lignes
- Le nombre de lignes estimées en sortie pour chaque opération élémentaire
- La longueur moyenne estimée (en octet) des lignes en sortie pour chaque opérations élémentaire

Exemple :

```
prod=# explain select empno,ename from emp;
          QUERY PLAN
-----
 Seq Scan on emp  (cost=0.00..15.60 rows=560 width=42)
(1 row)
```

# Optimisation

Il est possible de demander le plan d'exécution et d'exécuter la requête correspondante avec la commande **EXPLAIN ANALYZE**. Le résultat affiche :

- Les mêmes informations (estimées) de la commande EXPLAIN
- Pour chaque opération élémentaire, les temps réels (en millisecondes) pour renvoyer la première ligne, toutes les lignes et le nombre de lignes renvoyées
- Le temps d'exécution total de la requête

## Exemple :

```
prod=# explain analyze select empno,ename from emp;
```

### QUERY PLAN

```
-----  
-----  
Seq Scan on emp (cost=0.00..15.60 rows=560 width=42) (actual time=0.640..0.683  
rows=14 loops=1)  
Planning time: 0.088 ms  
Execution time: 0.737 ms  
(3 rows)
```

Si les statistiques sont trop anciennes ou que la commande ANALYZE n'a jamais été effectuée sur la table, les coûts estimés sont médiocres et l'optimiseur peut choisir un mauvais plan d'exécution. Il est essentiel d'avoir des statistiques fiables et donc d'effectuer périodiquement des ANALYZE sur les objets.

Mêmes opérations après avoir produits des statistiques :

# Optimisation

```
prod=# select count(*) from emp;
      count
-----
      14
(1 row)

prod=# analyze emp;
ANALYZE
prod=# explain select empno,ename from emp;

          QUERY PLAN
-----
 Seq Scan on emp  (cost=0.00..1.14 rows=14 width=10)
(1 row)

prod=# explain analyze select empno,ename from emp;

          QUERY PLAN
-----
 Seq Scan on emp  (cost=0.00..1.14 rows=14 width=10)
(actual time=0.130..0.172 rows=14 loops=1)
 Planning time: 0.152 ms
 Execution time: 0.261 ms
(3 rows)

prod=# create index emp_empno on emp (empno);
CREATE INDEX
prod=# explain analyze select empno,ename from emp where empno = 7369;

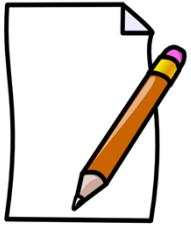
          QUERY PLAN
-----
 Seq Scan on emp  (cost=0.00..1.18 rows=1 width=10)
(actual time=0.000..0.009 rows=1 loops=1)
   Filter: (empno = 7369)
   Rows Removed by Filter: 13
 Planning time: 0.339 ms
 Execution time: 0.258 ms
(5 rows)

prod=#

```

# Optimisation

---



# Optimisation

---

## 3. Paramètres d'optimisation

Il existe un grand nombre de paramètres (déscrits dans la documentation **Part III. Server Administration chapitre 19 Server Configuration**) qui affectent directement le fonctionnement, la sécurité, la configuration, les performances d'un serveur PostgreSQL. Certains paramètres liés à la mémoire sont sensibles pour l'optimisation.

### A. Principaux paramètres d'optimisation liés à la mémoire

#### **shared\_buffers** (-B de postmaster)

Nombre de buffers mémoire partagés utilisables par le serveur (par défaut 128 MB en 10). Shared\_buffers doit être supérieur à 128 Ko + 16Ko fois la valeur de max\_connections. Des valeurs beaucoup plus importantes que la valeur par défaut sont nécessaires pour avoir de bonnes performances (exemple : de ¼ à 40% de la RAM sur un serveur UNIX/Linux dédié à PostgreSQL). L'augmentation de ce paramètre doit être supportée par l'O.S (voir le paramètre shmmmax).

#### **max\_connections** (-N de postmaster)

Nombre maximum de connexions clients concurrentes au serveur de base de données (valeur par défaut 100). L'augmentation de ce paramètre peut obliger PostgreSQL à réclamer plus de mémoire partagée System V ou de sémaphores que ne le permet la configuration par défaut du système d'exploitation (voir le § 18.4.1, « Mémoire partagée et sémaphore »). La valeur de ce paramètre est importante pour déterminer d'autres paramètres (comme work\_mem) car certaines ressources mémoires sont ou peuvent être allouées par client. Le nombre maximum de clients suggère la taille mémoire maximum potentiellement utilisée. Avec un bon matériel, PostgreSQL peut supporter plusieurs centaines de connexions.

#### **work\_mem**

Spécifie la taille de l'espace mémoire réservé aux tris internes avant de swapper sur des fichiers disque temporaires. La valeur par défaut est 4 Mbytes. Des tris sont générés par des requêtes comme « select...order by... », des jointures (merge join, hash join). Une requête complexe peut exécuter plusieurs tris en parallèle et consommer donc plusieurs fois cette valeur. De plus, chaque session peut consommer une ou plusieurs fois cette valeur de work\_mem (exemple de valeur utilisée : 10 Mo).

# Optimisation

---

- **maintenance\_work\_mem**

Spécifie la taille mémoire maximum que peuvent utiliser les opérations de maintenance (VACUUM, CREATE INDEX ...). La valeur par défaut est 64 Mo. Cette valeur peut être significativement augmentée car il y a rarement plusieurs opérations de ce type simultanément (exemple : 1 Go au maximum sur un serveur dédié ayant au moins 4 Go de RAM).

- **default\_statistics\_target**

PostgreSQL ne collecte pas énormément d'informations statistiques sur vos objets. Si vous n'obtenez pas de bons plans d'exécution, particulièrement sur les grosses tables, il faut augmenter `default_statistics_target` et régénérer des ANALYZE (ou attendre que autovacuum le fasse pour vous). `Default_statistics_target` doit être au moins égal à 100 (par défaut à 10 en 8.3 et à 100 depuis la 8.4). Pour chaque colonne, les 100 valeurs les plus fréquentes et 100 histogrammes sont stockés dans `pg_stats` comme échantillon représentatif des données. Des valeurs supérieures pour ce paramètre sont possibles mais augmentent le temps d'exécution d'ANALYZE, la taille de `pg_stats` et le temps de planification des requêtes.

- **effective\_cache\_size**

Influence l'optimiseur par rapport à la taille du cache disque O.S (estimation de la taille du cache disque O.S récupérable en additionnant les nombres free et cached renvoyés par les commandes free ou top). Si une donnée recherchée n'est pas dans le cache de PostgreSQL (`shared_buffers`), quelle est la chance d'avoir cette donnée dans cache disque O.S. Le fait de donner une estimation de la taille du cache disque O.S permet à l'optimiseur d'avoir une idée plus précise de trouver cette donnée dans le cache disque O.S et ainsi d'influencer le plan d'exécution de la requête (je passe ou pas par l'index). Généralement entre 50 et 75% de la taille de la RAM. `effective_cache_size` n'alloue pas d'espace mémoire et est uniquement utilisée comme information par l'optimiseur pour savoir comment exécuter une requête. La valeur par défaut est 4 Go.

...

# Optimisation

---

## B. Conseils pour les performances

Certaines considérations sont à prendre en compte pour les machines dédiées aux serveurs PostgreSQL :

- Disques rapides car l'accès aux données est plus déterminant que la puissance CPU.
- Il est préférable d'avoir plusieurs disques de tailles moyennes plutôt qu'un seul de taille énorme.
- Les disques en RAID 10 (RAID 1 (mirroring) + 0 (stripping)) sont plus performants que les disques en RAID 5.
- Utiliser les systèmes de fichiers ext2, ext3, ext4 sur Linux et NTFS sur Windows.
- Mettre les journaux WAL (`pg_xlog`) sur un autre disque pour améliorer la sécurité et les performances. Depuis la 8.3, il est possible d'utiliser l'option -X dans `initdb` pour choisir un répertoire pour les journaux WAL en dehors de la PGDATA.
- Mettre les applications sur des serveurs différents de votre serveur de données PostgreSQL.
- Augmenter très rapidement les valeurs des paramètres du fichier `postgresql.conf` liées la mémoire (`shared_buffers`, `work_mem`, `maintenance_work_mem...`) par rapport à votre RAM (`shared_buffers` à ¼ de la RAM...).

# Optimisation

---



# Optimisation

---

## 4. Contrôle de l'activité et statistiques du serveur

La plupart des outils systèmes de contrôle d'activité (ps, top, iostat, vmstat...) peuvent être utilisés pour contrôler un serveur PostgreSQL.

PostgreSQL peut aussi collecter de nombreuses statistiques sur l'activité d'un serveur PostgreSQL :

- Nombre d'accès aux tables et index en terme de blocs ou lignes parcourus
- Requête en cours d'exécution
- Connexions en cours
- ...

### A. Le collecteur de statistiques

En 9.1, le collecteur de statistiques est activé par défaut (process background « stats collector process »).

Les anciens paramètres **stats\_block\_level** et **stats\_row\_level** sont remplacés en 9.0 par le paramètre **track\_counts** positionné par défaut à on car autovacuum utilise les informations statistiques récupérées.

L'ancien paramètre **stats\_command\_string** (statistiques sur les requêtes en cours) est renommé en **track\_activities** et est positionné à on par défaut.

Le paramètre **stats\_reset\_on\_server\_start** disparaît car il est possible d'utiliser la fonction système **pg\_stat\_reset()**.

# Optimisation

---

Les statistiques sont consultables dans les vues :

**pg\_stat\_activity** informations sur les sessions en cours (nom de la base, process serveur, nom de l'utilisateur, requête en cours, heure de début de la requête, heure de début de la connexion...).

**pg\_stat\_database** informations sur l'activité des bases de données du serveur (nom de la base, nombre de process serveur actifs connectés à cette base, nombre de transactions commitées et rollbackées, nombre de blocs lus sur disque et nombre de blocs lus en mémoire).

**pg\_stat\_all\_tables** informations sur l'utilisation des tables (nom de la table, nombre de seq scan, nombre de lignes lus par les seq scan, nombre d'index scan, nombre de lignes insérées, mises à jour ou supprimées...). **pg\_stat\_user\_tables** pour les tables des utilisateurs uniquement et **pg\_stat\_sys\_tables** pour les tables systèmes uniquement.

**pg\_statio\_all\_tables** informations sur les tables en terme d'I/O (par table nombre de blocs lus sur disque, nombre de blocs lus en mémoire, nombre de blocs d'index lus sur disque, nombre de blocs d'index lus en mémoire...). **pg\_statio\_user\_tables** pour les tables des utilisateurs uniquement et **pg\_statio\_sys\_tables** pour les tables systèmes uniquement.

**pg\_stat\_all\_indexes** informations sur l'utilisation des index (nom de la table et de l'index, nombre de parcours d'index, nombre d'entrées de l'index renvoyés par ces parcours d'index, nombre de lignes actives de tables récupérées par ces parcours d'index). **pg\_stat\_user\_indexes** pour les index des utilisateurs uniquement et **pg\_stat\_sys\_indexes** pour les index systèmes uniquement.

**pg\_statio\_all\_indexes** informations sur les index en terme d'I/O (nombre de blocs d'index lus sur disque, nombre de blocs d'index lus en mémoire...). **pg\_statio\_user\_indexes** pour les index des utilisateurs uniquement et **pg\_statio\_sys\_indexes** pour les index systèmes uniquement.

# Optimisation

Exemple de statistiques validées par défaut :

The screenshot shows the pgAdmin III interface with a query window titled "Query - template1 sur postgres@192.168.91.129 : 5432 \*". The query is "select \* from pg\_stat\_database;". The results are displayed in a table:

	datid oid	datname name	numbackend integer	xact_commit bigint	xact_rollback bigint	blkss_read bigint	blkss_hit bigint	tup_returned bigint	tup_fetched bigint	tup_inserted bigint	tup_updated bigint	tup_deleted bigint
1	1	template1	2	1917	13	2039	129004	664820	36561	0	0	0
2	11510	template0	0	0	0	0	0	0	0	0	0	0
3	11511	postgres	1	2183	5	2045	127706	634348	37541	0	0	0
4	16417	prod	3	3725	56	729279	30168913	36958332	94215	7373916	4457014	4473835
5	16544	test	1	306	1	402	20352	95014	6038	71	25	13
6	16398	test3	2	1447	1	2070	100913	525097	29381	0	0	0

At the bottom, it says "OK.", "Unix", "Ligne 1 Col 31 Caract. 31", "6 lignes.", and "16 ms".

The screenshot shows the pgAdmin III interface with a query window titled "Query - template1 sur postgres@192.168.91.129 : 5432 \*". The query is "select \* from pg\_stat\_activity;". The results are displayed in a table:

	datid oid	datname name	procpid integer	usesysid oid	username name	current_query text	waiting boolean	xact_start timestamp with tz	query_start timestamp with tz	backend_start timestamp with tz	client_addr	client_port integer
1	1	template1	4142	10	postgres	<IDLE>	f		2008-03-14 02:12:19	2008-03-14 02:00:57	-1	
2	11511	postgres	4152	10	postgres	<IDLE>	f		2008-03-14 02:16:27	2008-03-14 02:02:23	192.168.91.1	1061
3	16417	prod	4154	10	postgres	<IDLE>	f		2008-03-14 02:02:45	2008-03-14 02:02:34	192.168.91.1	1062
4	16544	test	4161	10	postgres	<IDLE>	f		2008-03-14 02:03:40	2008-03-14 02:03:29	192.168.91.1	1065
5	16398	test3	4244	10	postgres	<IDLE>	f		2008-03-14 02:13:46	2008-03-14 02:13:36	192.168.91.1	1072
6	16398	test3	4164	10	postgres	<IDLE>	f		2008-03-14 02:04:02	2008-03-14 02:03:51	192.168.91.1	1067
7	16417	prod	4247	10	postgres	<IDLE>	f		2008-03-14 02:14:05	2008-03-14 02:13:55	192.168.91.1	1073
8	1	template1	4261	10	postgres	select * from pg_stat_activity;	f	2008-03-14 02:22:	2008-03-14 02:22:12	2008-03-14 02:16:30	192.168.91.1	1074
9	16417	prod	4287	10	postgres	insert into empbis select * from empbis	f	2008-03-14 02:22:	2008-03-14 02:22:12	2008-03-14 02:21:26	192.168.91.1	1075

At the bottom, it says "OK.", "Unix", "Ligne 1 Col 31 Caract. 31", "9 lignes.", and "16 ms".

# Optimisation

Query - prod sur postgres@192.168.91.129 : 5432 \*

Fichier Édition Requêtes Favoris Macros Affichage Aide

prod sur postgres@192.168.91.129 : 5432

```
select * from pg_stat_user_tables;
```

Bloc notes

Panneau sortie

Sortie de données Expliquer (Explain) Messages Historique

	relid oid	schemaname name	relname name	seq_scan bigint	seq_tup_read bigint	idx_scan bigint	idx_tup_fetch bigint	n_tup_ins bigint	n_tup_upd bigint	n_tup_del bigint
1	16639	public	bonus	0	0			0	0	0
2	16652	public	empbis	9	457856			229376	0	49152
3	16468	public	t1	5	0			0	0	0
4	16633	public	emp	22	11228	1	64	896	0	0
5	16645	public	salgrade	0	0			5	0	0
6	16636	public	dept	28	112			4	0	0

OK. Unix Ligne 1 Col 9 Caract. 9 6 lignes. 15 ms

Query - prod sur postgres@192.168.91.129 : 5432 \*

Fichier Édition Requêtes Favoris Macros Affichage Aide

prod sur postgres@192.168.91.129 : 5432

```
select * from pg_stat_user_tables;
```

Bloc notes

Panneau sortie

Sortie de données Expliquer (Explain) Messages Historique

	n_tup_upd bigint	n_tup_del bigint	n_tup_hot_up bigint	n_live_tup bigint	n_dead_tup bigint	last_vacuum timestamp wit	last_autovacuum timestamp with tin	last_analyze timestamp with ti	last_autoanalyze timestamp with ti
1	0	0	0	0	0				
2	0	49152	0	180224	49152				2008-03-14 02:22:30
3	0	0	0	0	0				
4	0	0	0	896	0				2008-03-12 08:23:01
5	0	0	0	5	0				
6	0	0	0	4	0				

OK. Unix Ligne 1 Col 9 Caract. 9 6 lignes. 15 ms

Query - prod sur postgres@192.168.91.129 : 5432 \*

Fichier Édition Requêtes Favoris Macros Affichage Aide

prod sur postgres@192.168.91.129 : 5432

```
select relid,relname,heap_blk_re,heap_blk_hi, idx_blk_re, idx_blk_hi  
from pg_statio_user_tables;
```

Bloc notes

Panneau sortie

Sortie de données Expliquer (Explain) Messages Historique

	relid oid	relname name	heap_blk_re bigint	heap_blk_hi bigint	idx_blk_re bigint	idx_blk_hi bigint
1	16652	empbis	1944	286298		
2	16639	bonus	0	0		
3	16645	salgrade	1	4		
4	16636	dept	2	30		
5	16633	emp	24	1001	5	1275
6	16468	t1	0	0		

OK. Unix Ligne 1 Col 14 Caract. 14 6 lignes. 16 ms

# Optimisation

```
[postgres@CentOS764b ~]$ ps -ef|grep postgres
root      2974  2870  0 15:04 pts/0    00:00:00 su - postgres
postgres  2975  2974  0 15:04 pts/0    00:00:00 -bash
postgres  3017      1  0 15:04 pts/0    00:00:00 /usr/local/pgsql/bin/postgres
postgres  3018  3017  0 15:04 ?        00:00:00 postgres: logger process
postgres  3020  3017  0 15:04 ?        00:00:00 postgres: checkpointer process
postgres  3021  3017  0 15:04 ?        00:00:00 postgres: writer process
postgres  3022  3017  0 15:04 ?        00:00:00 postgres: wal writer process
postgres  3023  3017  0 15:04 ?        00:00:00 postgres: archiver process
postgres  3024  3017  0 15:04 ?        00:00:00 postgres: stats collector process
postgres  3025  3017  0 15:04 ?        00:00:00 postgres: bgworker: logical replication
launcher
postgres  3492  3457  0 15:28 pts/1    00:00:00 psql prod
postgres  3493  3017  0 15:28 ?        00:00:00 postgres: postgres prod [local] idle
postgres  3710  3587  0 15:30 pts/2    00:00:00 psql test3 -h localhost
postgres  3711  3017  0 15:30 ?        00:00:00 postgres: postgres test3 localhost(42288)
idle
postgres  3849  3804  0 15:31 pts/3    00:00:00 psql template1
postgres  3850  3017  0 15:31 ?        00:00:00 postgres: postgres template1 [local] idle
postgres  3851  2975  0 15:32 pts/0    00:00:00 ps -ef
postgres  3852  2975  0 15:32 pts/0    00:00:00 grep --color=auto postgres
[postgres@CentOS764b ~]$
```

# Optimisation

---



# Optimisation

---

## 5. Contrib pg\_stat\_statements

La contribt **pg\_stat\_statements** permet de récupérer des statistiques d'exécution des requêtes exécutées sur un serveur PostgreSQL :

- nombre d'exécutions,
- durée total d'exécution de la requête SQL en secondes,
- nombre total de blocs partagés lus dans le cache par l'ordre SQL,
- nombre total de blocs partagés lus sur disque par l'ordre SQL,
- nombre total de blocs partagés écrits sur disque par l'ordre SQL,
- ...

Installer la contrib sous root :

```
cd /usr/local/src/postgresql-10.n/contrib/pg_stat_statements  
make  
make install
```

Positionner ces paramètres dans `postgresql.conf` et redémarrer le serveur :

```
shared_preload_libraries = 'pg_stat_statements'  
pg_stat_statements.max = 10000  
pg_stat_statements.track = all
```

Exécuter dans chaque base dans laquelle vous voulez utiliser cette contrib (ou dans `template1` pour propager la contrib chaque fois que vous créez une base) la commande :

```
mabase=# CREATE EXTENSION pg_stat_statements;  
CREATE EXTENSION
```

# Optimisation

Exemple :

```
mibase=# \x
Expanded display is on.
mibase=# SELECT query, calls, total_time,
  100.0 * shared_blk_hit / nullif(shared_blk_hit + shared_blk_read, 0) AS
    hit_percent
        FROM pg_stat_statements ORDER BY total_time DESC LIMIT 7;

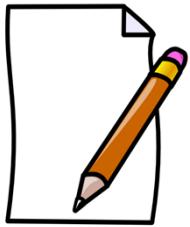
-[ RECORD 1 ]-----
query      | alter table pgbench_accounts add primary key (aid)
calls       | 1
total_time  | 12.606094
hit_percent | 10.8176739461655663
-[ RECORD 2 ]-----
query      | UPDATE pgbench_branches SET bbalance = bbalance + 2627 WHERE bid = 8;
calls       | 1
total_time  | 1.355714
hit_percent | 100.0000000000000000
-[ RECORD 3 ]-----
query      | UPDATE pgbench_branches SET bbalance = bbalance + -3701 WHERE bid = 7;
calls       | 1
total_time  | 1.152789
hit_percent | 100.0000000000000000
-[ RECORD 4 ]-----
query      | UPDATE pgbench_branches SET bbalance = bbalance + -3986 WHERE bid = 10;
calls       | 1
total_time  | 0.291548
hit_percent | 100.0000000000000000
-[ RECORD 5 ]-----
query      | select count(*) from pgbench_accounts ;
calls       | 1
total_time  | 0.270228
hit_percent | 10.7711138310893513
-[ RECORD 6 ]-----
query      | UPDATE pgbench_tellers SET tbalance = tbalance + 618 WHERE tid = 81;
calls       | 1
total_time  | 0.170703
hit_percent | 100.0000000000000000
-[ RECORD 7 ]-----
query      | UPDATE pgbench_branches SET bbalance = bbalance + 4433 WHERE bid = 8;
calls       | 1
total_time  | 0.118451
hit_percent | 100.0000000000000000
```

## **12. ANNEXES**

---

# Annexes

---



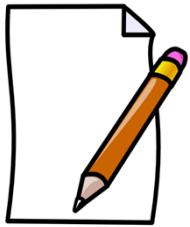
# Annexes

---

1.	Installation via des RPMs sur RHEL / Fedora .....	p.279
2.	Postgres / postmaster .....	p.287
3.	PG_CTL .....	p.293
4.	Extrait du fichier postgresql.conf .....	p.297
5.	Liste des commandes SQL .....	p.317
6.	Tables et vues systèmes .....	p.319
7.	Les vues et les fonctions pour les statistiques .....	p.321
8.	Les fonctions .....	p.325
9.	Les triggers .....	p.327

# Annexes

---



# Annexes

---

## 1. Installation via des RPMs sur RHEL / Fedora

### Installing PostgreSQL on Red Hat Enterprise Linux / Fedora Core

Last Update: Mon Sep 03, 2007

Written by

Devrim GÜNDÜZ (PGDG RPM Maintainer) [devrim@PostgreSQL.org](mailto:devrim@PostgreSQL.org) and

Lamar Owen (PGDG RPM Maintainer) [lowen@pari.edu](mailto:lowen@pari.edu)

(All contributions are welcome. Please feel free to drop us e-mail about your comments.)

This article will talk about the details of installing PostgreSQL Database Server on Red Hat Enterprise Linux / Fedora Core. RPMs are ready-to-install binary files for many Linux distributions. Please note that this article is written for 8.2 and above, so some information may not fit your version.

#### General information

PostgreSQL Global Development Group (PGDG) builds RPMs for various Linux distributions. At the time of this writing, we have RPMs and SRPMs for the following platforms<sup>1</sup>:

- Red Hat Enterprise Linux 3, 4 and 5
- Fedora Core 6,7,8

On PgFoundry, there is a project for building RPMs:

<http://pgfoundry.org/projects/pgsqlrpms>

Furthermore, there are -hackers, -general and -announce mailing lists. All discussions are open to everyone.

#### Obtaining the RPMs

The RPMs are available on PostgreSQL FTP site and all of its mirrors. For your convenience, you may try using web interface of PostgreSQL FTP site. That site will suggest you the suitable mirror(s) for you. The URL is:

<http://www.PostgreSQL.org/ftp>

# Annexes

---

## Which RPM for what purpose?

PGDG ships the following RPMs:

postgresql-libs	The postgresql-libs package provides the essential shared libraries for any PostgreSQL client program or interface. You will need to install this package to use any other PostgreSQL package or any clients that need to connect to a PostgreSQL server.
postgresql	If you want to manipulate a PostgreSQL database on a local or remote PostgreSQL server, you need this package. You also need to install this package if you're installing the postgresql-server package.
postgresql-contrib	The postgresql-contrib package contains contributed packages that are included in the PostgreSQL distribution.
postgresql-devel	The postgresql-devel package contains the header files and libraries needed to compile C or C++ applications which will directly interact with a PostgreSQL database management server and the ecpg Embedded C Postgres preprocessor. You need to install this package if you want to develop applications which will interact with a PostgreSQL server.
postgresql-docs	The postgresql-docs package includes the SGML source for the documentation as well as the documentation in PDF format and some extra documentation. Install this package if you want to help with the PostgreSQL documentation project, or if you want to generate printed documentation.
postgresql-server	The postgresql-server package includes the programs needed to create and run a PostgreSQL server, which will in turn allow you to create and maintain PostgreSQL databases. You should install postgresql-server if you want to create and maintain your own PostgreSQL databases and/or your own PostgreSQL server. You also need to install the postgresql package and its requirements.
postgresql-tcl	The postgresql-tcl package contains the PgTcl client library and its documentation.
postgresql-jdbc	The postgresql-jdbc package includes the .jar files needed for Java programs to access a PostgreSQL database.
postgresql-plperl	The postgresql-plperl package contains the PL/Perl procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-pltcl	The postgresql-pltcl package contains PL/Tcl procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-plpython	The postgresql-plpython package contains the PL/Python procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-python	The postgresql-python package includes a module for developers to use when writing Python code for accessing a PostgreSQL database.
postgresql-test	The postgresql-test package includes the sources and pre-built binaries of various tests for the PostgreSQL database management system, including regression tests and benchmarks.

# Annexes

---

## Which packages should I use?

If you feel lazy about reading the descriptions above, here is a shortcut for minimal scenarios:

- If you want to run a PostgreSQL server, install `postgresql-libs`, `postgresql` and `postgresql-server`.
- If you want to run a client, install `postgresql-libs` and `postgresql` rpms.

For some cases, you might just want to install `postgresql-libs` for some packages like PHP.

Package names also include version and architecture information. Official PostgreSQL Global Development Group RPM's have a 'PGDG' after the release number. Other RPMset's as distributed with Linux distributions may have a different release number and initials. The version numbering is the same as PostgreSQL.

It is preferable for the distribution-specific set to be the one used, as the PGDG set is intentionally generic. So, if your distro has a set of RPMs, use them in preference. If you want to stay up-to-date on the PostgreSQL core itself, use the PGDG generic set -- but understand that it is a GENERIC set.

These RPMs no longer support any sort of upgrading process other than that documented in the regular documentation. That is, you must dump, upgrade, initdb, and restore your data. You must remove the old server subpackage, install the new package and restore the data from dump.

## RPM File Locations

To be in compliance with the Linux FHS, the PostgreSQL PGDG RPMs install files in a manner not consistent with most of the PostgreSQL documentation. According to the standard PostgreSQL documentation, PostgreSQL is installed under the directory `/usr/local/pgsql`, with executables, source, and data existing in various subdirectories.

Different distributions have different ideas of some of these file locations. In particular, the documentation directory can be `/usr/doc`, `/usr/doc/packages`, `/usr/share/doc`, `/usr/share/doc/packages`, or some other similar path. The Red Hat / Fedora Core locations are listed below:

Executables	<code>/usr/bin</code>
Libraries	<code>/usr/lib</code>
Documentation	<code>/usr/share/doc/postgresql-x.y.z</code> <code>/usr/share/doc/postgresql-x.y.z/contrib</code>
Contrib	<code>/usr/share/pgsql/contrib</code>
Data	<code>/var/lib/pgsql/data</code>
Backup area	<code>/var/lib/pgsql/backup</code>
Templates	<code>/usr/share/pgsql</code>
Procedural Languages	<code>/usr/lib/pgsql</code>
Development Headers	<code>/usr/include/pgsql</code>
Other shared data	<code>/usr/share/pgsql</code>
Regression tests	<code>/usr/lib/pgsql/test/regress</code> (in the <code>-test</code> package)

# Annexes

---

Executables	/usr/bin
Documentation SGML	/usr/share/doc/postgresql-docs-x.y.z

The above list references the Red Hat / Fedora Core structure. These locations may change for other distributions. Use of 'rpm -ql' for each package is recommended as the 'Official' location source.

These RPMs are designed to be LSB-compliant -- if you find this not to be the case, please let us know by way of the pgsqlrpms-hackers@PgFoundry.org mailing list.

## Installing and Upgrading PostgreSQL RPMs

Installing PGDG RPMs are as easy as installing any RPMs

```
rpm -ivh package_name.version.arch.rpm
```

Unless specified, on minor release upgrades (i.e., upgrading from 8.2.0 to 8.2.1 or 8.2.4, etc<sup>2</sup>), you may use usual RPM upgrade process:

```
rpm -Uvh package_name.version.arch.rpm
```

Please note that on every new major version upgrade, you have to follow dump/reload sequence. However please **don't forget to read the Release Notes before upgrading because on some cases minor versions may need a dump/reload process. At those times if you use -U switch, then you will probably lose data! Please refer to "How do I perform a major upgrade?" section.**

Many of the RPMs are signed with the PGP key of the package builder. Directories contain a CURRENT\_MAINTAINER file which includes the name/email of the package builder and link to their PGP key. You might want to import the signature before you'll install the RPM:

```
rpm --import http://link/to/the/pgp/key
```

Unless this is not done, you can install/upgrade the RPMs but you'll be thrown a warning.

You may subscribe to pgsqlrpms-general@pgfoundry.org list for more details.

## Removing RPMs

You might want to take a full dump (and possibly a filesystem level backup) before removing RPMs. Removing an PostgreSQL RPM is as easy as removing any RPM:

```
rpm -e package_name
```

Before the removal of server package, if there are any running processes, all are stopped. You don't need to stop it.

## Starting PostgreSQL for the first time and init script

Red Hat Linux uses the System V Init package. A startup script for PostgreSQL is provided in the server package, as /etc/rc.d/init.d/postgresql. To start the postmaster, with sanity checking, as root, run

```
service postgresql start
```

To shut the postmaster down,

```
service postgresql stop
```

There are other parameters to this script -- execute 'service postgresql' for a listing.

On some cases you might want to edit this init file. For example, you might want to pass a --locale=... parameter to initdb, etc.

---

<sup>2</sup> 8.1 is the major version number, and 0,1,3, etc are minor version numbers

# Annexes

---

This script does the following sets defaults for configuration variables and then performs the given action (stop, start, etc). During the startup, first the script checks whether the database cluster has been initialized or not. If not, then you must run:

```
service postgresql initdb
```

This option is new as of 8.2 . Before 8.2, `service postgresql start` was calling `initdb` when the cluster was not initialized.

Then the service is started as usual.

You may also reload the server for some changes to take effect; but please look at the PostgreSQL documentation for the conditions that the database server needs a restart or a reload.

## Starting PostgreSQL automatically at system startup

To get this script to run at system startup run:

```
chkconfig postgresql on
```

and the proper symlinks will be created. See the `chkconfig` man page for more information. Note that this is manual -- while the startup script can include tags to allow `chkconfig` to automatically perform the symlinking, this is not done at this time.

## How do I perform a major upgrade?

Currently, PostgreSQL RPMs does not provide a data upgrade feature among major releases (or clearly, upgrades that require an `initdb`.) This work is under progress. In order to upgrade to a major version, you should follow the following steps:

- Take a full dump using `pg_dumpall`
- You might want to take a filesystem-level backup also. This is intentional.
- Check the backups! (Do it again!)
- Now, stop the database server:  
`/sbin/service postgresql stop`

- Remove all `postgresql` rpms. Please note that you'll probably need a `--nodeps` switch at the end:

```
/bin/rpm -e ` /bin/rpm -qa | grep postgresql`  
/bin/rpm -e ` /bin/rpm -qa | grep postgresql` --nodeps
```

During the removal of packages, some scripts will be run to remove `postgres` user and to uninstall `postgresql` service, etc.

Please note that removing of `postgresql-server` RPM will not also remove `/var/lib/pgsql`

- Remove<sup>3</sup> database cluster:

```
/bin/rm -rf /var/lib/pgsql
```

- Install new RPM sets

- Start database server

```
/sbin/service postgresql start
```

- Edit conf files, if you need.

- Reload the data to the new server (You may need to edit your data).

... and you're done!

---

<sup>3</sup> Better: `/bin/rm -rf ~postgres` or best take a filesystem-level backup of this directory by using `/bin/mv` for example

# Annexes

---

## Rebuilding from Source RPM

If your distribution is not supported by the binary RPM's from PostgreSQL.org, you will need to rebuild from the source RPM. Download the .src.rpm for this release. You will need to be root to rebuild, unless you have already set up a non-root build environment.

Install the source RPM with rpm -i, then CD to the rpm building area (on Red Hat or Fedora Core this is /usr/src/redhat by default). You will have to have a full development environment to rebuild the full RPM set.

This release of the RPMset includes the ability to conditionally build sets of packages. The parameters, their defaults, and the meanings are:

beta	0	#build with cassert and do not strip the binaries
pltcl	1	#build the postgresql-pltcl package.
jdbc	1	#build the postgresql-jdbc package.
plperl	1	#build the postgresql-plperl package.
test	1	#build the postgresql-test package.
plpython	1	#build the postgresql-plpython package.
pltcl	1	#build the pltcl portion of the postgresql-pl package.
plperl	1	#build the plperl portion of the postgresql-pl package.
ssl	1	#use OpenSSL support.
kerberos	1	#use Kerberos 5 support.
nls	1	#build with national language support.
pam	1	#build with PAM support.
runselftest	1	#do "make check" during the build.
xml	1	#build contrib/xml2
pgfts	0	#Build with --enable-thread-safety

To use these defines, invoke a rebuild like this:

```
rpm --rebuild --define 'plperl 0' --define 'pltcl 0' \
--define 'test 0' --define 'runselftest 1' \
--define 'kerberos 0' postgresql-8.2.4-1PGDG.src.rpm
```

This line would disable the plperl, pltcl, and test subpackages, enable the regression test run during build, and disable kerberos support. You might need to disable runselftest if there is an installed version of PostgreSQL that is a different major version from what you are trying to build. The self test tends to pick up the installed libpq.so shared library in place of the one being built :-), so if that isn't compatible the test will fail. Also, you can't use runselftest when doing the build as root.

More of these conditionals may be added/removed in the future.

# Annexes

---

## Contrib Files

The contents of the contrib tree are packaged into the -contrib subpackage and are processed with make and make install. There is documentation in /usr/share/doc/postgresql-contrib-VERSION for these modules. Most of the modules are in /usr/lib/pgsql for loadable modules, and binaries are in /usr/bin. In the future these files may be split out, depending upon function and dependencies.

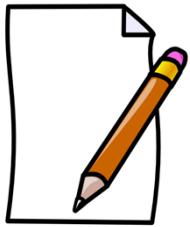
## More Information

You can get more information at <http://www.postgresql.org>

Please help make this packaging better -- let us know if you find problems, or better ways of doing things. You can reach us by e-mail at [pgsqlrpms-hackers@PgFoundry.org](mailto:pgsqlrpms-hackers@PgFoundry.org).

# Annexes

---



# Annexes

---

## 2. Postgres / postmaster

### Nom

postgres — Serveur de bases de données PostgreSQL™

### Synopsis

`postgres [option..]`

### Description

`postgres` est le serveur de bases de données PostgreSQL™. Pour qu'une application cliente puisse accéder à une base de données, elle se connecte (soit via le réseau soit localement) à un processus `postgres` en cours d'exécution. L'instance `postgres` démarre ensuite un processus serveur séparé pour gérer la connexion.

Une instance `postgres` gère toujours les données d'un seul cluster. Un cluster est un ensemble de bases de données stocké à un même emplacement dans le système de fichiers (le « répertoire des données »). Plus d'un processus `postgres` peut être en cours d'exécution sur un système à un moment donné, s'ils utilisent des répertoires différents et des ports de communication différents (voir ci-dessous). Quand `postgres` se lance, il a besoin de connaître l'emplacement du répertoire des données. Cet emplacement doit être indiquée par l'option `-D` ou par la variable d'environnement PGDATA ; il n'y a pas de valeur par défaut. Typiquement, `-D` ou PGDATA pointe directement vers le répertoire des données créé par `initdb(1)`. D'autres dispositions de fichiers possibles sont discutés dans Section 18.2, « Emplacement des fichiers ». Un répertoire de données est créé avec `initdb(1)`. Un répertoire de données est créé avec `initdb(1)`.

Par défaut, `postgres` s'exécute en avant-plan et affiche ses messages dans le flux standard des erreurs. En pratique, `postgres` devrait être exécuté en tant que processus en arrière-plan, par exemple au lancement.

La commande `postgres` peut aussi être appelé en mode mono-utilisateur. L'utilisation principal de ce mode est lors du « bootstrap » utilisé par `initdb(1)`. Quelque fois, il est utilisé pour du débogage et de la récupération suite à un problème (mais noter qu'exécuter un serveur en mode mono-utilisateur n'est pas vraiment convenable pour déboguer le serveur car aucune communication inter-processus réaliste et aucun verrouillage n'intervient.) Quand il est appelé en mode interactif à partir du shell, l'utilisateur peut saisir des requêtes et le résultat sera affiché à l'écran mais dans une forme qui est plus utile aux développeurs qu'aux utilisateurs. Dans le mode mono-utilisateur, la session ouverte par l'utilisateur sera configurée avec l'utilisateur d'identifiant 1 et les droits implicites du superutilisateur lui sont donnés. Cet utilisateur n'a pas besoin d'exister, donc le mode mono-utilisateur peut être utilisé pour récupérer manuellement après certains types de dommages accidentels dans les catalogues systèmes.

### Options

`postgres` accepte les arguments suivants en ligne de commande. Pour une discussion détaillée des options, consultez Chapitre 18, Configuration du serveur. Vous pouvez éviter de saisir la plupart de ces options en les initialisant dans le fichier de configuration. Certaines options (sûres) peuvent aussi être configurées à partir du client en cours de connexion d'une façon dépendante de l'application, configuration qui ne sera appliquée qu'à cette session. Par exemple si la variable d'environnement PGOPTIONS est configurée, alors les clients basés sur libpq passeront cette chaîne au serveur qui les interprétera comme les options en ligne de commande de `postgres`.

#### Général

`-A 0 | 1`

Active les vérifications d'assertion à l'exécution, donc une aide au débogage pour détecter les erreurs de programmation. Cette option est seulement disponible si les assertions ont été activées lors de la compilation de PostgreSQL™. Dans ce cas, la valeur par défaut est « on ».

`-B ntampons`

Configure le nombre de tampons partagés utilisés par les processus serveur. La valeur par défaut de ce paramètre est choisi automatiquement par `initdb`. Indiquer cette option est équivalent à configurer le paramètre `shared_buffers`.

`-c nom=valeur`

Configure un paramètre d'exécution nommé. Les paramètres de configuration supportés par PostgreSQL™ sont décrits dans Chapitre 18, Configuration du serveur. La plupart des autres options en ligne de commande sont en fait des formes courtes d'une affectation de paramètres. `-c` peut apparaître plusieurs fois pour configurer différents paramètres.

`-d niveau-débogage`

Configure le niveau de débogage. Plus haute est sa valeur, plus importante seront les traces écrites dans les journaux. Les valeurs vont de 1 à 5. Il est aussi possible de passer `-d 0` pour une session spécifique qui empêchera le niveau des traces serveur du processus `postgres` parent d'être propagé jusqu'à cette session.

# Annexes

---

- D *repdonnées*  
Indique le répertoire des données ou des fichier(s) de configuration. Voir Section 18.2, « Emplacement des fichiers » pour les détails.
- e  
Configure le style de date par défaut à « European », c'est-à-dire l'ordre DMY pour les champs en entrée. Ceci cause aussi l'affichage de la date avant le mois dans certains formats de sortie de date. Voir Section 8.5, « Types date/heure » pour plus d'informations.
- F  
Désactive les appels `fsync` pour améliorer les performances au risque de corrompre des données dans l'idée d'un arrêt brutal du système. Spécifier cette option est équivalent à désactiver le paramètre de configuration `fsync`. Lisez la documentation détaillée avant d'utiliser ceci !
- h *hôte*  
Indique le nom d'hôte ou l'adresse IP sur lequel `postgres` attend les connexions TCP/IP d'applications clientes. La valeur peut aussi être une liste d'adresses séparées par des virgules ou \* pour indiquer l'attente sur toutes les interfaces disponibles. Une valeur vide indique qu'il n'attend sur aucune adresse IP, auquel cas seuls les sockets de domaine Unix peuvent être utilisés pour se connecter au serveur. Par défaut, attend les connexions seulement sur `localhost`. Spécifier cette option est équivalent à la configurer dans le paramètre `listen_addresses`.
- i  
Autorise les clients distants à se connecter via TCP/IP (domaine Internet). Sans cette option, seules les connexions locales sont autorisées. Cette option est équivalent à la configuration du paramètre `listen_addresses` à \* dans `postgresql.conf` ou via -h.  
  
Cette option est obsolète car il ne permet plus l'accès à toutes les fonctionnalités de `listen_addresses`. Il est généralement mieux de configurer directement `listen_addresses`.
- k *directory*  
Indique le répertoire de la socket de domaine Unix sur laquelle `postgres` est en attente des connexions des applications clientes. La valeur par défaut est habituellement `/tmp` mais cela peut se changer au moment de la construction.
- l  
Active les connexions sécurisées utilisant SSL. PostgreSQL™ doit avoir été compilé avec SSL pour que cette option soit disponible. Pour plus d'informations sur SSL, référez-vous à Section 17.8, « Connexions tcp/ip sécurisées avec ssl ».
- N *max-connections*  
Initialise le nombre maximum de connexions clientes que le serveur acceptera. La valeur par défaut de ce paramètre est choisie automatiquement par `initdb`. Indiquer cette option est équivalent à configurer le paramètre `max_connections`.
- o *extra-options*  
Les options en ligne de commande indiquées dans `extra-options` sont passées à tous les processus serveur exécutés par ce processus `postgres`. Si la chaîne d'option contient des espaces, la chaîne entière doit être entre guillemets.  
  
Cette option est obsolète ; toutes les options en ligne de commande des processus serveur peuvent être spécifiées directement sur la ligne de commande de `postgres`.
- p *port*  
Indique le port TCP/IP ou l'extension du fichier socket de domaine Unix sur lequel `postgres` attend les connexions des applications clientes. Par défaut, la valeur de la variable d'environnement `PGPORT` environment ou, si cette variable n'est pas configurer, la valeur connue à la compilation (habituellement 5432). Si vous indiquez un port autre que celui par défaut, alors toutes les applications clientes doivent indiquer le même numéro de port soit dans les options en ligne de commande soit avec `PGPORT`.
- s  
Affiche une information de temps et d'autres statistiques à la fin de chaque commande. Ceci est utile pour créer des rapports de performance ou pour configurer finement le nombre de tampons.
- S *work-mem*  
Indique la quantité de mémoire à utiliser par les tris internes et par les hachages avant d'utiliser des fichiers disque temporaires. Voir la description du paramètre `work_mem` dans Section 18.4.1, « Mémoire ».
- nom=valeur  
Configure un paramètre à l'exécution ; c'est une version courte de -c.
- describe-config  
Cette option affiche les variables de configuration internes du serveur, leurs descriptions et leurs valeurs par défaut dans un format COPY délimité par des tabulations. Elle est conçue principalement pour les outils d'administration.

# Annexes

---

## Options semi-internes

Les options décrites ici sont utilisées principalement dans un but de débogage et pouvant quelque fois aider à la récupération de bases de données très endommagées. Il n'y a aucune raison pour les utiliser dans la configuration d'un système en production. Elles sont listées ici à l'intention des développeurs PostgreSQL™. De plus, une de ces options pourrait disparaître ou changer dans le futur sans avertissement.

**-f { s | i | m | n | h }**

Interdit l'utilisation de parcours et de méthode de jointure particulières. *s* and *i* désactivent respectivement les parcours séquentiels et d'index alors que *n*, *m* et *h* désactivent respectivement les jointures de boucles imbriquées, jointures de fusion et hash.

Ni les parcours séquentiels ni les jointures de boucles imbriquées ne peuvent être désactivés complètement ; les options *-fs* et *-fn* ne font que décourager l'optimiseur d'utiliser ce type de plans.

**-n**

Cette option est présente pour les problèmes de débogage du genre mort brutal d'un processus serveur. La stratégie habituelle dans cette situation est de notifier tous les autres processus serveur qu'ils doivent se terminer, puis réinitialiser la mémoire partagée et les sémaphores. Tout ceci parce qu'un processus serveur errant peut avoir corrompu certains états partagés avant de terminer. Cette option spécifie seulement que *postgres* ne réinitialisera pas les structures de données partagées. Un développeur système avec quelques connaissances peut utiliser un débogueur pour examiner l'état de la mémoire partagée et des sémaphores.

**-O**

Autorise la modification de la structure des tables système. C'est utilisé par *initdb*.

**-P**

Ignore les index système lors de la lecture des tables système (mais les met à jour lors de la modification des tables). Ceci est utile lors de la récupération d'index système endommagés.

**-t pa[rses] | pl[anner] | e[xecutor]**

Affiche les statistiques en temps pour chaque requête en relation avec un des modules majeurs du système. Cette option ne peut pas être utilisée avec l'option *-s*.

**-T**

Cette option est présente pour les problèmes de débogage du genre mort brutal d'un processus serveur. La stratégie habituelle dans cette situation est de notifier tous les autres processus serveur qu'ils doivent se terminer, puis réinitialiser la mémoire partagée et les sémaphores. Tout ceci parce qu'un processus serveur errant peut avoir corrompu certains états partagés avant de terminer. Cette option spécifie seulement que *postgres* arrêtera tous les autres processus serveur en leur envoyant le signal *SIGSTOP* mais ne les arrêtera pas. Ceci permet aux développeurs système de récupérer manuellement des « core dumps » de tous les processus serveur.

**-v protocole**

Indique le numéro de version utilisé par le protocole interface/moteur pour une session particulière. Cette option est uniquement utilisée en interne.

**-W secondes**

Un délai de ce nombre de secondes survient quand un nouveau processus serveur est lancé, une fois la procédure d'authentification terminée. Ceci a pour but de permettre au développeur d'attacher un débogueur au processus serveur.

## Options en mode mono-utilisateur

Les options suivantes s'appliquent uniquement en mode mono-utilisateur.

**--single**

Sélectionne le mode mono-utilisateur. Cette option doit être la première sur la ligne de commande.

**base**

Indique le nom de la base à accéder. Il doit être le dernier argument. Si elle est omise, le nom de l'utilisateur est utilisé par défaut.

**-E**

Affiche toutes les commandes.

**-j**

Désactive l'utilisation du retour chariot comme délimiteur d'instruction.

**-r fichier**

# Annexes

---

Envoie toute la sortie des traces du serveur dans *fichier*. Dans le mode normal, cette option est ignorée et `stderr` est utilisé par tous les processus.

## Environnement

### PGCLIENTENCODING

Jeu de caractères utilisé par défaut par tous les clients. (Les clients peuvent surcharger ce paramètre individuellement.) Cette valeur est aussi configurable dans le fichier de configuration.

### PGDATA

Emplacement du répertoire des données par défaut

### PGDATESTYLE

Valeur par défaut du paramètre en exécution `datestyle`. (Cette variable d'environnement est obsolète.)

### PGPORT

Numéro de port par défaut (à configurer de préférence dans le fichier de configuration)

### TZ

Fuseau horaire du serveur

## Diagnostiques

Un message d'erreur mentionnant `semget` ou `shmget` indique probablement que vous devez configurer votre noyau pour fournir la mémoire partagée et les sémaphores adéquates. Pour plus de discussion, voir Section 17.4, « Gérer les ressources du noyau ». Vous pouvez aussi repousser la configuration du noyau en diminuant `shared_buffers` pour réduire la consommation de la mémoire partagée utilisée par PostgreSQL™, et/ou en diminuant `max_connections` pour réduire la consommation de sémaphores.

Un message d'erreur suggérant qu'un autre serveur est déjà en cours d'exécution devra vous demander une vérification attentive, par exemple en utilisant la commande `ps` should be checked carefully, for example by using the command

```
$ ps ax | grep postgres
```

ou

```
$ ps -ef | grep postgres
```

suivant votre système. Si vous êtes certain qu'il n'y a aucun serveur en conflit, vous pouvez supprimer le fichier verrou mentionné dans le message et tenter de nouveau.

Un message d'erreur indiquant une incapacité à se lier à un port indique que ce port est déjà utilisé par des processus autres que PostgreSQL™. Vous pouvez aussi obtenir cette erreur si vous quittez `postgres` et le relancez immédiatement en utilisant le même port ; dans ce cas, vous devez tout simplement attendre quelques secondes pour que le système d'exploitation ferme bien le port avant de tenter de nouveau. Enfin, vous pouvez obtenir cette erreur si vous indiquez un numéro de port que le système considère comme réservé. Par exemple, beaucoup de versions d'Unix considèrent les numéros de port sous 1024 comme de « confiance » et permettent seulement leur accès par le superutilisateur Unix.

## Notes

L'outil `pg_ctl(1)` est utilisable pour lancer et arrêter le serveur `postgres` de façon sûre et confortable.

Si possible, ne pas utiliser `SIGKILL` pour tuer le serveur `postgres` principal. Le fait empêchera `postgres` de libérer les ressources système (c'est-à-dire mémoire partagée et sémaphores) qu'il détient avant de s'arrêter. Ceci peut poser problèmes lors du lancement d'un `postgres` frais.

Pour terminer le serveur `postgres` normalement, les signaux `SIGTERM`, `SIGINT` ou `SIGQUIT` peuvent être utilisés. Le premier attendra que tous les clients terminent avant de quitter, le second forcera la déconnexion de tous les clients et le troisième quittera immédiatement sans arrêt propre. Ce dernier amènera une récupération lors du redémarrage.

Le signal `SIGHUP` rechargea les fichiers de configuration du serveur. Il est aussi possible d'envoyer `SIGHUP` à un processus serveur individuel mais ce n'est pas perceptible.

Pour annuler une requête en cours d'exécution, envoyez le signal `SIGINT` au processus exécutant cette commande.

Le serveur `postgres` utilise `SIGTERM` pour indiquer aux processus serveur de quitter normalement et `SIGQUIT` pour terminer sans le nettoyage habituel. Ces signaux ne devraient pas être utilisés par les utilisateurs. Il est aussi déconseillé d'envoyer `SIGKILL` à un processus serveur -- le serveur `postgres` principal interprétera ceci comme un arrêt brutal et forcera tous les autres processus serveur à quitter dans le cas d'une procédure standard de récupération après arrêt brutal.

# Annexes

## Bogues

Les options `--` ne fonctionneront pas sous FreeBSD et OpenBSD. Utilisez `-c` à la place. C'est un bogue dans les systèmes d'exploitation affectés ; une prochaine version de PostgreSQL™ fournira un contournement si ce n'est pas corrigé.

## Utilisation

Pour démarrer un serveur en mode mono-utilisateur, utilisez une commande comme

```
postgres --single -D /usr/local/pgsql/data autres-options ma_base
```

Fournissez le bon chemin vers le répertoire des bases avec l'option `-D` ou assurez-vous que la variable d'environnement PGDATA est configurée. De plus, spécifiez le nom de la base particulière avec laquelle vous souhaitez travailler.

Habituellement, le serveur en mode mono-utilisateur traite le retour chariot comme le terminateur d'une commande ; il n'y a pas le concept du point-virgule contrairement à psql. Pour saisir une commande sur plusieurs lignes, vous devez saisir un antislash juste avant un retour chariot, sauf pour le dernier.

Mais si vous utilisez l'option `-j`, alors le retour chariot ne termine pas une commande. Dans ce cas, le serveur lira l'entrée standard jusqu'à une marque de fin de fichier (EOF), puis il traitera la saisie comme une seule chaîne de commande. Les retours chariot avec antislash ne sont pas traités spécialement dans ce cas.

Pour quitter la session, saisissez EOF (habituellement, Control+D). Si vous avez utilisé l'option `-j`, deux EOF consécutifs sont nécessaires pour quitter.

Notez que le serveur en mode mono-utilisateur ne fournit pas de fonctionnalités avancées sur l'édition de lignes (par exemple, pas d'historique des commandes).

## Exemples

Pour lancer postgres en tâche de fond avec les valeurs par défaut, saisissez :

```
$ nohup postgres >logfile 2>&1 </dev/null &
```

Pour lancer postgres avec un port spécifique :

```
$ postgres -p 1234
```

Cette commande exécutera un postgres qui communiquera sur le port 1234. Pour se connecter à ce serveur en utilisant psql, vous aurez besoin de saisir :

```
$ psql -p 1234
```

ou de configurer la variable d'environnement PGPORT :

```
$ export PGPORT=1234
$ psql
```

Les paramètres nommés peuvent être configurés suivant deux façons :

```
$ postgres -c work_mem=1234
$ postgres --work-mem=1234
```

Ces deux formes surchargent le paramétrage qui pourrait exister pour `work_mem` dans `postgresql.conf`. Notez que les tirets bas dans les noms de paramètres sont écrits avec soit des tirets bas soit des tirets sur la ligne de commande. Sauf pour les expériences à court terme, il est probablement mieux de modifier le paramétrage dans `postgresql.conf` que de se baser sur une option en ligne de commande.

## Voir aussi

`initdb(1)`, `pg_ctl(1)`

## Nom

`postmaster` — Serveur de bases de données PostgreSQL™

## Synopsis

```
postmaster [option...]
```

## Description

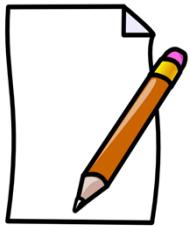
`postmaster` est un alias obsolète de `postgres`.

## Voir aussi

`postgres(1)`

# Annexes

---



# Annexes

---

## 3. pg\_ctl

### Nom

`pg_ctl` — initialiser, démarrer, arrêter ou redémarrer le serveur PostgreSQL™

### Synopsis

```
pg_ctl init[db] [-s] [-D répertoire_données] [-o options]
pg_ctl start [-w] [-t secondes] [-s] [-D répertoire_données] [-l nomfichier] [-o options] [-p chemin] [-c]
pg_ctl stop [-W] [-t secondes] [-s] [-D répertoire_données] [-m [s[mart]] | [f[ast]] | [i[mmediate]] ]
pg_ctl restart [-w] [-t secondes] [-s] [-D répertoire_données] [-c] [-m [s[mart]] | [f[ast]] | [i[mmediate]] ] [-o op-
tions]
pg_ctl reload [-s] [-D répertoire_données]
pg_ctl status [-D répertoire_données]
pg_ctl kill [nom_signal] [id_processus]
pg_ctl register [-N nom_service] [-U nom_utilisateur] [-P mot_de_passe] [-D répertoire_données] [-w] [-t
secondes] [-o options]
pg_ctl unregister [-N nom_service]
```

### Description

`pg_ctl` est un outil qui permet d'initialiser une instance, de démarrer, d'arrêter, ou de redémarrer un serveur PostgreSQL™ (`postgres(1)`). Il permet également d'afficher le statut d'un serveur en cours d'exécution.

Bien que le serveur puisse être démarré manuellement, `pg_ctl` encapsule les tâches comme la redirection des traces ou le détachement du terminal et du groupe de processus. Il fournit également des options intéressantes pour contrôler l'arrêt.

Le mode `init` ou `initdb` crée une nouvelle grappe PostgreSQL™. Une grappe est un ensemble de bases contrôlées par une même instance du serveur. Ce mode invoque la commande `initdb(1)` pour les détails.

En mode `start`, un nouveau serveur est démarré. Le serveur est démarré en tâche de fond et l'entrée standard est attachée à `/dev/null` (or `nul` on Windows). On Unix-like systems, by default, the server's standard output and standard error are sent to `pg_ctl`'s standard output (not standard error). The standard output of `pg_ctl` should then be redirected to a file or piped to another process such as a log rotating program like `rotatelogs`; otherwise `postgres` will write its output to the controlling terminal (from the background) and will not leave the shell's process group. On Windows, by default the server's standard output and standard error are sent to the terminal. These default behaviors can be changed by using `-l` to append server output to a log file.

En mode `stop`, le serveur en cours d'exécution dans le répertoire indiqué est arrêté. Trois méthodes différentes d'arrêt peuvent être choisies avec l'option `-m` : le mode « `smart` » attend la fin de la sauvegarde en ligne (PITR) et la déconnexion de tous les clients. C'est la valeur par défaut. Si le serveur est en mode récupération (`recovery`), la récupération et la réPLICATION en continu sont arrêtées dès que tous les clients se sont déconnectés. Le mode « `fast` » n'attend pas la déconnexion des clients et stoppe la sauvegarde en ligne (PITR). Toutes les transactions actives sont annulées et les clients sont déconnectés. Le serveur est ensuite arrêté. Le mode « `immediate` » tue tous les processus serveur sans leur laisser la possibilité de s'arrêter proprement. Cela conduit à une récupération au redémarrage.

Le mode `restart` exécute un arrêt suivi d'un démarrage. Ceci permet de modifier les options en ligne de commande de `postgres`.

Le mode `reload` envoie simplement au processus `postgres` un signal `SIGHUP`. Le processus relit alors ses fichiers de configuration (`postgresql.conf`, `pg_hba.conf`, etc.). Cela permet de modifier les options des fichiers de configuration qui ne requièrent pas un redémarrage complet pour être prises en compte.

Le mode `status` vérifie si un serveur est toujours en cours d'exécution sur le répertoire de données indiqué. Si c'est le cas, le PID et les options en ligne de commande utilisées lors de son démarrage sont affichés.

Le mode `kill` permet d'envoyer un signal à un processus spécifique. Ceci est particulièrement utile pour Microsoft Windows™, qui ne possède pas de commande `kill`. `--help` permet d'afficher la liste des noms de signaux supportés.

Le mode `register` permet d'enregistrer un service système sur Microsoft Windows™.

Le mode `unregister` permet, sur Microsoft Windows™, d'annuler un service système précédemment enregistré avec la commande `register`.

### Options

`-c`

Tente d'autoriser la création de fichiers core suite à un arrêt brutal du serveur, sur les plateformes où cette fonctionnalité est

# Annexes

---

disponible, en augmentant la limite logicielle qui en dépend. C'est utile pour le déboguage et pour diagnostiquer des problèmes en permettant la récupération d'une trace de la pile d'un processus serveur en échec.

-D *répertoire\_données*

Indique l'emplacement des fichiers de la base de données sur le système de fichiers. Si cette option est omise, la variable d'environnement PGDATA est utilisée.

-l *nomfichier*

Ajoute la sortie des traces du serveur dans *nomfichier*. Si le fichier n'existe pas, il est créé. L'umask est configuré à 077, donc l'accès au journal des traces est, par défaut, interdit aux autres utilisateurs.

-m *mode*

Précise le mode d'arrêt. *mode* peut être *smart*, *fast* ou *immediate*, ou la première lettre d'un de ces trois mots.

-o *options*

Indique les options à passer directement à la commande *postgres*.

Ces options sont habituellement entourées par des guillemets simples ou doubles pour s'assurer qu'elles soient passées groupées.

-p *chemin*

Indique l'emplacement de l'exécutable *postgres*. Par défaut, l'exécutable *postgres* est pris à partir du même répertoire que *pg\_ctl* ou, si cela échoue, à partir du répertoire d'installation codé en dur. Il n'est pas nécessaire d'utiliser cette option sauf cas inhabituel, comme lorsque des erreurs concernant l'impossibilité de trouver l'exécutable *postgres* apparaissent.

Dans le mode *init*, cette option indique de manière analogue la localisation de l'exécutable *initdb*.

-s

Affichage des seules erreurs, pas de messages d'information.

-t

Le nombre de secondes à attendre pour la fin du lancement ou de l'arrêt.

-w

Attendre la fin du démarrage ou de l'arrêt. Le temps d'attente par défaut est de 60 secondes, option par défaut pour les arrêts. Un arrêt réussi est indiqué par la suppression du fichier PID. Pour le démarrage, un *psql -l* se terminant avec succès indique que tout va bien. *pg\_ctl* tente d'utiliser le bon port pour *psql*. Si la variable d'environnement PGPORT existe, elle est utilisée. Sinon, un port configuré dans le fichier *postgresql.conf* est cherché. Si aucun des deux n'est utilisé, le port par défaut avec lequel PostgreSQL™ a été compilé (5432 par défaut) est utilisé. Lorsqu'il attend, *pg\_ctl* renvoie un code de sortie précis fondé sur le succès du démarrage ou de l'arrêt.

-W

Ne pas attendre la fin du démarrage ou de l'arrêt. C'est la valeur par défaut pour les démarriages et redémarrages.

## Options Windows

-N *nom\_service*

Nom du service système à enregistrer. Le nom est utilisé à la fois comme nom de service et comme nom affiché.

-P *mot\_de\_passe*

Mot de passe de l'utilisateur qui démarre le service.

-U *nom\_utilisateur*

Nom de l'utilisateur qui démarre le service. Pour les utilisateurs identifiés sur un domaine, on utilise le format DOMAIN\nom\_utilisateur.

## Environnement

PGDATA

Emplacement par défaut du répertoire des données.

PGHOST

Nom d'hôte par défaut ou localisation de la socket de domaine Unix pour *psql(1)* (utilisée par l'option -w).

PGPORT

Port par défaut pour *psql(1)* (utilisé par l'option -w).

Pour les autres paramètres serveur, voir *postgres(1)*. Cet outil, comme la plupart des autres outils PostgreSQL™, utilise aussi les

# Annexes

variables d'environnement supportées par la bibliothèque libpq (voir Section 31.13, « Variables d'environnement »).

## Fichiers

### postmaster.pid

Ce fichier, situé dans le répertoire des données, est utilisé pour aider pg\_ctl à déterminer si le serveur est actuellement en cours d'exécution.

### postmaster.opts

Si ce fichier existe dans le répertoire des données, pg\_ctl (en mode restart) passe le contenu du fichier comme options de postgres, sauf en cas de surcharge par l'option -o. Le contenu de ce fichier est aussi affiché en mode status.

### postgresql.conf

Ce fichier, situé dans le répertoire des données, est analysé pour trouver le port à utiliser avec psql lorsque -w est donné en mode start.

## Notes

Attendre le démarrage complet n'est pas une opération bien définie et peut échouer si le contrôle d'accès est configuré pour qu'un client local ne puisse pas se connecter sans interaction manuelle (par exemple, une authentification par mot de passe). Pour des variables de connexion supplémentaires, voir Section 31.13, « Variables d'environnement ». Pour les mots de passe, voir aussi Section 31.14, « Fichier de mots de passe ».

## Exemples

### Lancer le serveur

Démarrer un serveur :

```
$ pg_ctl start
```

Démarrer un serveur, avec blocage tant que le serveur n'est pas complètement démarré :

```
$ pg_ctl -w start
```

Pour un serveur utilisant le port 5433 et s'exécutant sans fsync, on utilise :

```
$ pg_ctl -o "-F -p 5433" start
```

### Arrêt du serveur

```
$ pg_ctl stop
```

arrête le serveur. Utiliser l'option -m permet de contrôler *comment* le serveur s'arrête.

### Redémarrage du serveur

Redémarrer le serveur est pratiquement équivalent à l'arrêter puis à le démarrer à nouveau si ce n'est que pg\_ctl sauvegarde et réutilise les options en ligne de commande qui étaient passées à l'instance précédente. Pour redémarrer le serveur de la façon la plus simple, on utilise :

```
$ pg_ctl restart
```

Redémarrer le serveur, en attendant l'arrêt et le redémarrage :

```
$ pg_ctl -w restart
```

Redémarrer en utilisant le port 5433 et en désactivant fsync après redémarrage :

```
$ pg_ctl -o "-F -p 5433" restart
```

### Affichage de l'état du serveur

Exemple de statut affiché à partir de pg\_ctl :

```
$ pg_ctl status
```

# Annexes

---

```
pg_ctl : le serveur est en cours d'exécution (pid : 13718)
/usr/local/pgsql/bin/postgres "-D" "/usr/local/pgsql/data" "-p" "5433" "-B" "128"
C'est la ligne de commande qui sera appelée en mode redémarrage.
```

## Voir aussi

[initdb\(1\)](#), [postgres\(1\)](#)

## 4. Extrait du fichier postgresql.conf

Extrait (de la documentation officielle) des principaux paramètres de configuration d'un serveur PostgreSQL. La totalité des paramètres du fichier de configuration du serveur se trouve dans la documentation officielle chapitre 18 « Server Configuration ».

### 18.1. Paramètres de configuration

Tous les noms de paramètres sont insensibles à la casse. Chaque paramètre prend une valeur d'un de ces cinq types : booléen, entier, nombre à virgule flottante, chaîne de caractères ou énumération. Les unités par défaut peuvent être récupérées en référençant pg\_settings.unit. Les valeurs booléennes peuvent être on, off, true, false, yes, no, 1, 0 ou tout préfixe non ambigu de celles-ci (toutes ces écritures sont insensibles à la casse).

Certains paramètres indiquent une valeur de taille mémoire ou de durée. Ils ont chacun une unité implicite, soit Ko, soit blocs (typiquement 8 Ko), soit millisecondes, soit secondes, soit minutes. Les unités par défaut peuvent être obtenues en interrogant pg\_settings.unit. Pour simplifier la saisie, une unité différente peut être indiquée de façon explicite. Les unités mémoire valides sont kB (kilo-octets), MB (Méga-octets) et GB (Giga-octets) ; les unités de temps valides sont ms (millisecondes), s (secondes), min (minutes), h (heures), et d (jours). Les unités de mémoire sont des multiples de 1024, pas de 1000.

Les paramètres de type « enum » sont spécifiés de la même façon que les paramètres de type chaîne, mais sont restreints à un jeu limité de valeurs. Les valeurs autorisées peuvent être obtenues de pg\_settings.enumvals. Les paramètres enum sont insensibles à la casse.

Une façon d'initialiser ces paramètres est d'édition le fichier postgresql.conf qui est normalement placé dans le répertoire des données (une copie par défaut est installé ici quand le répertoire des données est initialisé). Un exemple de contenu peut être :

```
# Ceci est un commentaire
log_connections = yes
log_destination = 'syslog'
search_path = '"$user", public'
shared_buffers = 128MB
```

Un paramètre est indiqué par ligne. Le signe égal entre le nom et la valeur est optionnel. Les espaces n'ont pas de signification et les lignes vides sont ignorées. Les symboles dièse (#) désignent le reste de la ligne comme un commentaire. Les valeurs des paramètres qui ne sont pas des identificateurs simples ou des nombres doivent être placées entre guillemets simples. Pour intégrer un guillemet simple dans la valeur d'un paramètre, on écrit soit deux guillemets (c'est la méthode préférée) soit un antislash suivi du guillemet.

En plus de la configuration des paramètres, le fichier postgresql.conf peut contenir des *directives d'inclusion* indiquant un autre fichier à lire et dont le contenu doit être traité à ce niveau comme partie intégrante du fichier de configuration. Les directives d'inclusion ressemblent simplement à :

```
include 'nom_fichier'
```

Si le nom du fichier n'est pas un chemin absolu, il est pris comme relatif au répertoire contenant le fichier de configuration le référençant. Les inclusions peuvent être imbriquées.

Le fichier de configuration est relu à chaque fois que le processus serveur principal reçoit un signal SIGHUP (pg\_ctl reload est le moyen le plus simple de l'envoyer). Le processus serveur principal propage aussi ce signal aux processus serveur en cours d'exécution de façon à ce que les sessions existantes obtiennent aussi la nouvelle valeur. Il est également possible d'envoyer le signal directement à un seul processus serveur. Quelques paramètres ne peuvent être initialisés qu'au lancement du serveur ; tout changement de leur valeur dans le fichier de configuration est ignoré jusqu'au prochain démarrage du serveur.

Une autre façon de configurer ces paramètres est de les passer comme option de la commande postgres :

```
postgres -c log_connections=yes -c log_destination='syslog'
```

Les options de la ligne de commande surchargent le paramétrage effectué dans le fichier postgresql.conf. Ce qui signifie que la valeur d'un paramètre passé en ligne de commande ne peut plus être modifiée et rechargeée à la volée à l'aide du fichier postgresql.conf. C'est pourquoi, bien que la méthode de la ligne de commande paraisse pratique, elle peut coûter en flexibilité par la suite.

Il est parfois utile de donner une option en ligne de commande pour une session particulière unique. La variable d'environnement PGOPTIONS est utilisée côté client à ce propos :

```
env PGOPTIONS='-c geqo=off' psql
```

(Cela fonctionne pour toute application client fondée sur libpq, et non pas seulement pour psql.) Cela ne fonctionne pas pour les

# Annexes

paramètres fixés au démarrage du serveur ou qui doivent être précisés dans `postgresql.conf`.

De plus, il est possible d'affecter un ensemble de paramètres à un utilisateur ou à une base de données. Quand une session est lancée, les paramètres par défaut de l'utilisateur et de la base de données impliqués sont chargés. Les commandes `ALTER USER(7)` et `ALTER DATABASE(7)` sont respectivement utilisées pour configurer ces paramètres. Les paramètres par base de données surchargent ceux passés sur la ligne de commande de `postgres` ou du fichier de configuration et sont à leur tour surchargés par ceux de l'utilisateur ; les deux sont surchargés par les paramètres de session.

Quelques paramètres peuvent être modifiés dans les sessions SQL individuelles avec la commande `SET(7)`, par exemple :

```
SET ENABLE_SEQSCAN TO OFF;
```

Si `SET` est autorisé, il surcharge toutes les autres sources de valeurs pour le paramètre. Quelques paramètres ne peuvent pas être changés via `SET` : s'ils contrôlent un comportement qui ne peut pas être modifié sans relancer le serveur PostgreSQL™, par exemple. De plus, quelques paramètres peuvent être modifiés via `SET` ou `ALTER` par les superutilisateurs.

La commande `SHOW(7)` permet d'inspecter les valeurs courantes de tous les paramètres.

La table virtuelle `pg_settings` (décrite dans la Section 45.55, « `pg_settings` ») autorise aussi l'affichage et la mise à jour de paramètres de session à l'exécution. Elle est équivalente à `SHOW` et `SET` mais peut être plus facile à utiliser parce qu'elle peut être jointe avec d'autres tables et que ses lignes peuvent être sélectionnées en utilisant des conditions personnalisées. Elle contient aussi davantage d'informations sur les valeurs autorisées pour les paramètres.

## 18.3.1. Paramètres de connexion

`listen_addresses` (string)

Indique les adresses TCP/IP sur lesquelles le serveur écoute les connexions en provenance d'applications clientes. La valeur prend la forme d'une liste de noms d'hôte ou d'adresses IP numériques séparés par des virgules. L'entrée spéciale \* correspond à toutes les interfaces IP disponibles. Si la liste est vide, le serveur n'écoute aucune interface IP, auquel cas seuls les sockets de domaine Unix peuvent être utilisées pour s'y connecter. La valeur par défaut est `localhost`, ce qui n'autorise que les connexions TCP/IP locales de type « `loopback` ». Bien que l'authentification client (Chapitre 19, Authentification du client) permet un contrôle très fin sur les accès au serveur, `listen_addresses` contrôle les interfaces pouvant accepter des tentatives de connexion, ce qui permet d'empêcher des demandes de connexion ambiguës sur des interfaces réseau non sécurisées. Ce paramètre ne peut être configuré qu'au lancement du serveur.

`port` (integer)

Le port TCP sur lequel le serveur écoute ; 5432 par défaut. Le même numéro de port est utilisé pour toutes les adresses IP que le serveur écoute. Ce paramètre ne peut être configuré qu'au lancement du serveur.

`max_connections` (integer)

Indique le nombre maximum de connexions concurrentes au serveur de base de données. La valeur par défaut typique est de 100 connexions, mais elle peut être moindre si les paramètres du noyau ne le supportent pas (ce qui est déterminé lors de l'`initdb`). Ce paramètre ne peut être configuré qu'au lancement du serveur.

L'augmentation de ce paramètre peut obliger PostgreSQL™ à réclamer plus de mémoire partagée System V ou de sémaphores que ne le permet la configuration par défaut du système d'exploitation. Voir la Section 17.4.1, « Mémoire partagée et sémaphore » pour plus d'informations sur la façon d'ajuster ces paramètres, si nécessaire.

Lors de l'exécution d'un serveur en attente, vous devez configurer ce paramètre à la même valeur ou à une valeur plus importante que sur le serveur maître. Sinon, des requêtes pourraient ne pas être autorisées sur le serveur en attente.

`superuser_reserved_connections` (integer)

Indique le nombre de connecteurs (« slots ») réservés aux connexions des superutilisateurs PostgreSQL™. Au plus `max_connections` connexions peuvent être actives simultanément. Dès que le nombre de connexions simultanément actives atteint `max_connections` moins `superuser_reserved_connections`, les nouvelles connexions ne sont plus acceptées que pour les superutilisateurs, et aucune nouvelle connexion de réPLICATION ne sera acceptée.

La valeur par défaut est de trois connexions. La valeur doit être plus petite que la valeur de `max_connections`. Ce paramètre ne peut être configuré qu'au lancement du serveur.

.../...

# Annexes

## 18.3.2. Sécurité et authentification

`authentication_timeout (integer)`

Temps maximum pour terminer l'authentification du client, en secondes. Si un client n'a pas terminé le protocole d'authentification dans ce délai, le serveur ferme la connexion. Cela protège le serveur des clients bloqués occupant une connexion indéfiniment. La valeur par défaut est d'une minute. Ce paramètre peut être configuré au lancement du serveur et dans le fichier `postgresql.conf`.

`ssl (boolean)`

Active les connexions SSL. Lire la Section 17.8, « Connexions tcp/ip sécurisées avec `ssl` » avant de l'utiliser. Désactivé par défaut. Ce paramètre ne peut être configuré qu'au lancement du serveur. La communication SSL n'est possible qu'avec des connexions TCP/IP.

.../...

`db_user_namespace (boolean)`

Active les noms d'utilisateur par base de données. Désactivé par défaut, ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

Si ce paramètre est activé, les utilisateurs doivent être créés sous la forme `nomutilisateur@nom_base`. Quand `nomutilisateur` est passé par un client se connectant, @ et le nom de la base de données sont ajoutés au nom de l'utilisateur et ce nom d'utilisateur spécifique à la base est recherché par le serveur. Lorsque des utilisateurs dont le nom contient un @ sont créés dans l'environnement SQL, ce nom doit être placé entre guillemets.

`db_user_namespace` permet aux représentations des noms d'utilisateurs du client et du serveur de différer. Les vérifications sont toujours faites avec les noms d'utilisateurs du serveur, ce qui fait que les méthodes d'authentification doivent être configurées pour le nom d'utilisateur du serveur, pas pour celui du client. Comme `md5` utilise le nom d'utilisateur comme sel à la fois sur le client et le serveur, `md5` ne peut pas être utilisé conjointement avec `db_user_namespace`.

Ce paramètre activé, il reste possible de créer des utilisateurs globaux ordinaires. Il suffit pour cela d'ajouter @ au nom du client, e.g. `joe@`. Le @ est supprimé avant que le serveur ne recherche ce nom.



### Note

Cette fonctionnalité, temporaire, sera supprimée lorsqu'une solution complète sera trouvée.

.../...

## 18.4.1. Mémoire

`shared_buffers (integer)`

Initialise la quantité de mémoire que le serveur de bases de données utilise comme mémoire partagée. La valeur par défaut, en général 32 Mo, peut être automatiquement abaissée si la configuration du noyau ne la supporte pas (déterminé lors de l'exécution de l'`initdb`). Ce paramètre doit être au minimum de 128 Ko + 16 Ko par `max_connections`. (Des valeurs personnalisées de `BLCKSZ` agissent sur ce minimum.) Des valeurs significativement plus importantes que ce minimum sont généralement nécessaires pour de bonnes performances. Ce paramètre ne peut être configuré qu'au lancement du serveur.

# Annexes

---

Si vous disposez d'un serveur dédié à la base de données, avec 1 Go de mémoire ou plus, une valeur de départ raisonnable pour ce paramètre est de 25% la mémoire de votre système. Certains cas peuvent nécessiter une valeur encore plus importante pour le `shared_buffers` mais comme PostgreSQL™ profite aussi du cache du système d'exploitation, il est peu probable qu'une allocation de plus de 40% de la mémoire fonctionnera mieux qu'une valeur plus restreinte. Des valeurs importantes pour le paramètre `shared_buffers` requièrent généralement une augmentation proportionnelle du `checkpoint_segments`, pour étendre dans le temps les écritures de grandes quantités de données, nouvelles ou modifiées.

Sur des systèmes comprenant moins d'1 Go de mémoire, un pourcentage plus restreint est approprié pour laisser une place suffisante au système d'exploitation. De plus, sur Windows, les grandes valeurs pour `shared_buffers` ne sont pas aussi efficaces. Vous pouvez avoir de meilleurs résultats en conservant un paramétrage assez bas et en utilisant le cache du système d'exploitation à la place. L'échelle habituelle pour `shared_buffers` sur des systèmes Windows va de 64 Mo à 512 Mo.

L'augmentation de ce paramètre peut obliger PostgreSQL™ à réclamer plus de mémoire partagée System V que ce que la configuration par défaut du système d'exploitation ne peut gérer. Voir la Section 17.4.1, « Mémoire partagée et sémaphore » pour de plus amples informations sur l'ajustement de ces paramètres, si nécessaire.

## `temp_buffers` (integer)

Configure le nombre maximum de tampons temporaires utilisés par chaque session de la base de données. Ce sont des tampons locaux à la session utilisés uniquement pour accéder aux tables temporaires. La valeur par défaut est de 8 Mo. Ce paramètre peut être modifié à l'intérieur de sessions individuelles mais seulement jusqu'à la première utilisation des tables temporaires dans une session ; les tentatives suivantes de changement de cette valeur n'ont aucun effet sur cette session.

Une session alloue des tampons temporaires en fonction des besoins jusqu'à atteindre la limite donnée par `temp_buffers`. Positionner une valeur importante pour les sessions qui ne le nécessitent pas ne coûte qu'un descripteur de tampon, soit environ 64 octets, par incrément de `temp_buffers`. Néanmoins, si un tampon est réellement utilisé, 8192 autres octets sont consommés pour celui-ci (ou, plus généralement, `BLCKSZ` octets).

## `max_prepared_transactions` (integer)

Configure le nombre maximum de transactions simultanément dans l'état « préparées » (voir PREPARE TRANSACTION(7)). Zéro, la configuration par défaut, désactive la fonctionnalité des transactions préparées. Ce paramètre ne peut être configuré qu'au lancement du serveur.

Si vous ne prévoyez pas d'utiliser les transactions préparées, ce paramètre devrait être positionné à zéro pour éviter toute création accidentelle de transactions préparées. Au contraire, si vous les utilisez, il peut être intéressant de positionner `max_prepared_transactions` au minimum à au moins `max_connections` pour que chaque session puisse avoir sa transaction préparée.

Augmenter ce paramètre peut conduire PostgreSQL™ à réclamer plus de mémoire partagée System V que ne le permet la configuration par défaut du système d'exploitation. Voir la Section 17.4.1, « Mémoire partagée et sémaphore » pour les informations concernant la façon d'ajuster ces paramètres, si nécessaire.

Lors de l'exécution d'un serveur en attente, vous devez configurer ce paramètre à la même valeur ou à une valeur plus importante que sur le serveur maître. Sinon, des requêtes pourraient ne pas être autorisées sur le serveur en attente.

## `work_mem` (integer)

Indique la quantité de mémoire que les opérations de tri interne et les tables de hachage peuvent utiliser avant de basculer sur des fichiers disque temporaires. La valeur par défaut est de 1 Mo. Pour une requête complexe, il peut y avoir plusieurs opérations de tri ou de hachage exécutées en parallèle ; chacune peut utiliser de la mémoire à hauteur de cette valeur avant de commencer à placer les données dans des fichiers temporaires. De plus, de nombreuses sessions peuvent exécuter de telles opérations simultanément. La mémoire totale utilisée peut, de ce fait, atteindre plusieurs fois la valeur de `work_mem` ; il est nécessaire de garder cela à l'esprit lors du choix de cette valeur. Les opérations de tri sont utilisées pour ORDER BY, DISTINCT et les jointures de fusion. Les tables de hachage sont utilisées dans les jointures de hachage, les agrégations et le traitement des sous-requêtes IN fondés sur le hachage.

## `maintenance_work_mem` (integer)

Indique la quantité maximale de mémoire que peuvent utiliser les opérations de maintenance telles que VACUUM, CREATE INDEX et ALTER TABLE ADD FOREIGN KEY. La valeur par défaut est de 16 Mo. Puisque seule une de ces opérations peut être exécutée à la fois dans une session et que, dans le cadre d'un fonctionnement normal, peu d'opérations de ce genre sont exécutées concurrentiellement sur une même installation, il est possible d'initialiser cette variable à une valeur bien plus importante que `work_mem`. Une grande valeur peut améliorer les performances des opérations VACUUM et de la restauration des sauvegardes.

Notez que quand autovacuum fonctionne, un maximum de `autovacuum_max_workers` fois cette quantité de mémoire peut être utilisé, donc faites attention à ne pas configurer la valeur par défaut de façon trop importante.

# Annexes

---

## `max_stack_depth (integer)`

Indique la profondeur maximale de la pile d'exécution du serveur. La configuration idéale pour ce paramètre est la limite réelle de la pile assurée par le noyau (configurée par `ulimit -s` ou équivalent local) à laquelle est soustraite une marge de sécurité d'un Mo environ. La marge de sécurité est nécessaire parce que la profondeur de la pile n'est pas vérifiée dans chaque routine du serveur mais uniquement dans les routines clés potentiellement récursives telles que l'évaluation d'une expression. Le paramétrage par défaut est de 2 Mo, valeur faible qui implique peu de risques. Néanmoins, elle peut s'avérer trop petite pour autoriser l'exécution de fonctions complexes. Seuls les superutilisateurs peuvent modifier ce paramètre.

Configurer ce paramètre à une valeur plus importante que la limite réelle du noyau signifie qu'une fonction récursive peut occasionner un arrêt brutal d'un processus serveur particulier. Sur les plateformes où PostgreSQL™ peut déterminer la limite du noyau, il interdit de positionner cette variable à une valeur inadéquate. Néanmoins, toutes les plateformes ne fournissent pas cette information, et une grande attention doit être portée au choix de cette valeur.

## 18.4.2. Usage des ressources du noyau

### `max_files_per_process (integer)`

Positionne le nombre maximum de fichiers simultanément ouverts par sous-processus serveur. La valeur par défaut est de 1000 fichiers. Si le noyau assure une limite par processus, il n'est pas nécessaire de s'intéresser à ce paramètre. Toutefois, sur certaines plateformes (notamment les systèmes BSD) le noyau autorise les processus individuels à ouvrir plus de fichiers que le système ne peut effectivement en supporter lorsqu'un grand nombre de processus essayent tous d'ouvrir ce nombre de fichiers. Si le message « Too many open files » (« Trop de fichiers ouverts ») apparaît, il faut essayer de réduire ce paramètre. Ce paramètre ne peut être configuré qu'au lancement du serveur.

.../...

.../...

# Annexes

---

## 18.4.3. Report du VACUUM en fonction de son coût

Lors de l'exécution des commandes VACUUM(7) et ANALYZE(7), le système maintient un compteur interne qui conserve la trace du coût estimé des différentes opérations d'entrée/sortie réalisées. Quand le coût accumulé atteint une limite (indiquée par `vacuum_cost_limit`), le processus traitant l'opération s'arrête un court moment (précisé par `vacuum_cost_delay`). Puis, il réinitialise le compteur et continue l'exécution.

Le but de cette fonctionnalité est d'autoriser les administrateurs à réduire l'impact des entrées/sorties de ces commandes en fonction de l'activité des bases de données. Nombreuses sont les situations pour lesquelles il n'est pas très important que les commandes de maintenance telles que VACUUM et ANALYZE se finissent rapidement, mais il est généralement très important que ces commandes n'interfèrent pas de façon significative avec la capacité du système à réaliser d'autres opérations sur les bases de données. Le report du VACUUM en fonction de son coût fournit aux administrateurs un moyen d'y parvenir.

Cette fonctionnalité est désactivée par défaut pour les commandes VACUUM lancées manuellement. Pour l'activer, la variable `vacuum_cost_delay` doit être initialisée à une valeur différente de zéro.

`vacuum_cost_delay (integer)`

Indique le temps, en millisecondes, de repos du processus quand la limite de coût a été atteinte. La valeur par défaut est zéro, ce qui désactive la fonctionnalité de report du VACUUM en fonction de son coût. Une valeur positive active cette fonctionnalité. Sur de nombreux systèmes, la résolution réelle du sleep est de 10 millisecondes ; configurer `vacuum_cost_delay` à une valeur qui n'est pas un multiple de 10 conduit alors au même résultat que de le configurer au multiple de 10 supérieur.

Lors d'utilisation de vacuum basée sur le coût, les valeurs appropriées pour `vacuum_cost_delay` sont habituellement assez petites, de l'ordre de 10 à 20 millisecondes. Il est préférable d'ajuster la consommation de ressource de vacuum en changeant les autres paramètres de coût de vacuum.

.../...

# Annexes

---

## 18.4.4. Processus d'écriture en arrière-plan

Il existe un processus serveur séparé appelé *background writer* dont le but est d'écrire les tampons « sales » (parce que nouveaux ou modifiés). Ce processus écrit les tampons partagés pour que les processus serveur gérant les requêtes des utilisateurs n'aient jamais ou peu fréquemment à attendre qu'une écriture se termine. Néanmoins, ce processus d'écriture en tâche de fond implique une augmentation globale de la charge des entrées/sorties disque car, quand une page fréquemment modifiée pourrait n'être écrite qu'une seule fois par CHECKPOINT, le processus d'écriture en tâche de fond pourrait l'avoir écrit plusieurs fois si cette page a été modifiée plusieurs fois dans le même intervalle. Les paramètres discutés dans cette sous-section peuvent être utilisés pour configurer finement son comportement pour les besoins locaux.

### `bgwriter_delay (integer)`

Indique le délai entre les tours d'activité du processus d'écriture en arrière-plan. À chaque tour, le processus écrit un certain nombre de tampons modifiés (contrôlable par les paramètres qui suivent). Puis, il s'endort pour `bgwriter_delay` millisecondes et recommence. La valeur par défaut est de 200 millisecondes. Sur de nombreux systèmes, la résolution réelle du sleep est de 10 millisecondes ; positionner `bgwriter_delay` à une valeur qui n'est pas un multiple de 10 peut avoir le même résultat que de le positionner au multiple de 10 supérieur. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### `bgwriter_lru_maxpages (integer)`

Nombre maximum de tampons qui peuvent être écrits à chaque tour par le processus d'écriture en tâche de fond. Le configurer à zéro désactive l'écriture en tâche de fond (sauf en ce qui concerne l'activité des points de vérification). La valeur par défaut est de 100 tampons. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### `bgwriter_lru_multiplier (floating point)`

Le nombre de tampons sales écrits à chaque tour est basé sur le nombre de nouveaux tampons qui ont été requis par les processus serveur lors des derniers tours. Le besoin récent moyen est multiplié par `bgwriter_lru_multiplier` pour arriver à une estimation du nombre de tampons nécessaire au prochain tour. Les tampons sales sont écrits pour qu'il y ait ce nombre de tampons propres, réutilisables. (Néanmoins, au maximum `bgwriter_lru_maxpages` tampons sont écrits par tour.) De ce fait, une configuration de 1.0 représente une politique d'écriture « juste à temps » d'exactement le nombre de tampons prédicts. Des valeurs plus importantes fournissent une protection contre les pics de demande, alors qu'une valeur plus petite laisse intentionnellement des écritures aux processus serveur. La valeur par défaut est de 2. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

Des valeurs plus faibles de `bgwriter_lru_maxpages` et `bgwriter_lru_multiplier` réduisent la charge supplémentaire des entrées/sorties induite par le processus d'écriture en arrière-plan. En contrepartie, la probabilité que les processus serveurs effectuent plus d'écritures par eux-mêmes augmente, ce qui retarde les requêtes interactives.

.../...

# Annexes

## 18.5. Write Ahead Log

Voir aussi la Section 29.4, « Configuration des journaux de transaction » pour les détails concernant l'optimisation des WAL.

### 18.5.1. Paramètres

#### wal\_level (enum)

wal\_level détermine la quantité d'informations écrite dans les journaux de transactions. La valeur par défaut est minimal, ce qui permet d'écrire seulement les informations nécessaires pour survivre à un arrêt brutal ou à un arrêt immédiat. archive ajoute quelques enregistrements supplémentaires pour permettre l'archivage des journaux de transactions. hot\_standby en ajoute encore plus pour permettre l'exécution de requêtes en lecture seule sur le serveur en attente. Ce paramètre peut seulement être configuré au lancement du serveur.

Au niveau minimal, certains enregistrements dans les journaux de transactions peuvent être évités, par exemple pour des commandes CREATE INDEX, CLUSTER et COPY sur une table qui a été créée ou tronquée dans la même transaction (voir Section 14.4.7, « Désactiver l'archivage des journaux de transactions et la réPLICATION en flux »). Mais, du coup, les journaux au niveau minimal ne contiennent pas suffisamment d'informations pour reconstruire les données à partir d'une sauvegarde de base et des journaux de transactions. Donc, les niveaux archive ou hot\_standby doivent être utilisés pour activer l'archivage des journaux de transactions (archive\_mode) et la réPLICATION en flux.

Au niveau hot\_standby, en plus des informations que trace déjà le niveau archive, plus d'informations sont nécessaires pour reconstruire le statut des transactions en cours à partir du journal de transactions. Pour activer les requêtes en lecture seule sur un serveur en attente, wal\_level doit être configuré à hot\_standby sur le serveur principal et hot\_standby doit être activé sur le serveur en attente. Il existe une différence mesurable de performances entre l'utilisation des niveaux hot\_standby et archive, donc un retour d'expérience serait apprécié si l'impact est ressenti en production.

#### fsync (boolean)

Si ce paramètre est activé, le serveur PostgreSQL™ tente de s'assurer que les mises à jour sont écrites physiquement sur le disque à l'aide d'appels système `fsync()` ou de méthodes équivalentes (voir wal\_sync\_method). Cela permet de s'assurer que le cluster de bases de données peut revenir à un état cohérent après une panne matérielle ou du système d'exploitation.

Bien que désactiver `fsync` améliore fréquemment les performances, cela peut avoir pour conséquence une corruption des données non récupérables dans le cas d'un arrêt inattendu ou brutal du système. Donc, il est seulement conseillé de désactiver `fsync` si vous pouvez facilement recréer la base de données complète à partir de données externes.

Quelques exemples de circonstances permettant de désactiver `fsync` : le chargement initial d'une nouvelle instance à partir d'une sauvegarde, l'utilisation de l'instance pour traiter des statistiques sur une base horaire qui est ensuite recréée, la création d'un clone d'une base en lecture seule, clone qui serait recréé fréquemment et n'est pas utilisé pour du failover. La haute qualité du matériel n'est pas une justification suffisante pour désactiver `fsync`.

Dans de nombreuses situations, désactiver synchronous\_commit pour les transactions non critiques peut fournir une grande partie des performances de la désactivation de `fsync`, sans les risques associés de corruption de données.

`fsync` ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Si ce paramètre est désactivé (`off`), il est intéressant de désactiver aussi `full_page_writes`.

#### synchronous\_commit (boolean)

Indique si la validation des transactions doit attendre l'écriture des enregistrements WAL avant que la commande ne renvoie une indication de « réussite » au client. La configuration par défaut, et la plus sûre, est `on`. Quand ce paramètre est désactivé (`off`), il peut exister un délai entre le moment où le succès est rapporté et le moment où la transaction est vraiment protégée d'un arrêt brutal du serveur. (Le délai maximum est de trois fois `wal_writer_delay`.) Contrairement à `fsync`, la configuration de ce paramètre à `off` n'implique aucun risque d'incohérence dans la base de données : un arrêt brutal du système d'exploitation ou d'une base de données peut résulter en quelques transactions récentes prétendument validées perdues malgré tout. Cependant, l'état de la base de données est identique à celui obtenu si les transactions avaient été correctement annulées. C'est pourquoi la désactivation de `synchronous_commit` est une alternative utile quand la performance est plus importante que la sûreté de la transaction. Pour plus de discussion, voir Section 29.3, « Validation asynchrone (Asynchronous Commit) ».

Ce paramètre peut être changé à tout moment ; le comportement pour toute transaction est déterminé par la configuration en cours lors de la validation. Il est donc possible et utile d'avoir certaines validations validées en synchrone et d'autres en asynchrone. Par exemple, pour réaliser une validation asynchrone de transaction à plusieurs instructions avec une valeur par défaut inverse, on exécute l'instruction `SET LOCAL synchronous_commit TO OFF` dans la transaction.

#### wal\_sync\_method (enum)

Méthode utilisée pour forcer les mises à jour des WAL sur le disque. Si `fsync` est désactivé, alors ce paramètre est inapplicable, car les mises à jour des journaux de transactions ne sont pas du tout forcées. Les valeurs possibles sont :

- `open_datasync` (écrit les fichiers WAL avec l'option `O_DSYNC` de `open()`)
- `fdatasync` (appelle `fdatasync()` à chaque validation)
- `fsync_writethrough` (appelle `fsync()` à chaque validation, forçant le mode `write-through` de tous les caches disque en écriture)
- `fsync` (appelle `fsync()` à chaque validation)
- `open_sync` (écrit les fichiers WAL avec l'option `O_SYNC` de `open()`)

# Annexes

---

Ces options ne sont pas toutes disponibles sur toutes les plateformes. La valeur par défaut est la première méthode de la liste ci-dessus supportée par la plateforme. Les options open \* utilisent aussi O\_DIRECT s'il est disponible. L'outil `src/tools/fsync` disponible dans le code source de PostgreSQL permet de tester les performances des différentes méthodes de synchronisation. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## `full_page_writes (boolean)`

Quand ce paramètre est activé, le serveur écrit l'intégralité du contenu de chaque page disque dans les WAL lors de la première modification de cette page qui intervient après un point de vérification. C'est nécessaire car l'écriture d'une page lors d'un plantage du système d'exploitation peut n'être que partielle, ce qui conduit à une page sur disque qui contient un mélange d'anciennes et de nouvelles données. Les données de modification de niveau ligne stockées habituellement dans les WAL ne sont pas suffisantes pour restaurer complètement une telle page lors de la récupération qui suit la panne. Le stockage de l'image de la page complète garantit une restauration correcte de la page, mais au prix d'un accroissement de la quantité de données à écrire dans les WAL. (Parce que la relecture des WAL démarre toujours à un point de vérification, il suffit de faire cela lors de la première modification de chaque page survenant après un point de vérification. De ce fait, une façon de réduire le coût d'écriture de pages complètes consiste à augmenter le paramètre régulant les intervalles entre points de vérification.)

La désactivation de ce paramètre accélère les opérations normales, mais peut aboutir either `unrecoverable data corruption`, or `silent data corruption`, after a system failure. Les risques sont similaires à la désactivation de `fsync`, bien que moindres, and it should be turned off only based on the same circumstances recommended for that parameter.

La désactivation de ce paramètre n'affecte pas l'utilisation de l'archivage des WAL pour la récupération d'un instantané, aussi appelé PITR (voir Section 24.3, « Archivage continu et récupération d'un instantané (PITR) »).

Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Activé par défaut (on).

## `wal_buffers (integer)`

Quantité de mémoire utilisée en mémoire partagée pour les données WAL. La valeur par défaut est de 64 Ko. Ce paramètre nécessite uniquement d'être assez important pour contenir toutes les données WAL engendrées par une transaction typique, car les données sont écrites sur le disque à chaque validation de transaction. Ce paramètre ne peut être configuré qu'au lancement du serveur.

L'augmentation de ce paramètre peut conduire PostgreSQL™ à réclamer plus de tampons partagés System V que ne le permet la configuration par défaut du système d'exploitation. Voir la Section 17.4.1, « Mémoire partagée et sémaphore » pour les informations sur la façon d'ajuster ces paramètres, si nécessaire.

## `wal_writer_delay (integer)`

Indique le délai entre les tours d'activité pour l'enregistreur des WAL. À chaque tour, l'enregistreur place les WAL sur disque. Il s'endort ensuite pour `wal_writer_delay` millisecondes et recommence. La valeur par défaut est de 200 millisecondes (200ms). Pour de nombreux systèmes, la résolution réelle du sleep est de 10 millisecondes ; configurer `wal_writer_delay` à une valeur qui n'est pas un multiple de 10 a le même résultat que de le configurer au multiple de 10 immédiatement supérieur. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## `commit_delay (integer)`

Délai entre l'enregistrement d'une validation dans le tampon WAL et le vidage du tampon sur le disque, en microsecondes. Un délai différent de zéro peut autoriser la validation de plusieurs transactions en un seul appel `fsync()`, si la charge système est assez importante pour que des transactions supplémentaires soient prêtes dans l'intervalle donné. Mais le délai est perdu si aucune autre transaction n'est prête à être validée. De ce fait, le délai n'est traité que si, au minimum, `commit_siblings` autres transactions sont actives au moment où le processus serveur a écrit son enregistrement de validation. La valeur par défaut est zéro (pas de délai).

## `commit_siblings (integer)`

Nombre minimum de transactions concurrentes ouvertes en même temps nécessaires avant d'attendre le délai `commit_delay`. Une valeur plus importante rend plus probable le fait qu'au moins une autre transaction soit prête à valider pendant le délai. La valeur par défaut est de cinq transactions.

# Annexes

---

## 18.5.2. Points de vérification

### `checkpoint_segments (integer)`

Nombre maximum de journaux de transaction entre deux points de vérification automatique des WAL (chaque segment fait normalement 16 Mo). La valeur par défaut est de trois segments. Augmenter ce paramètre peut accroître le temps nécessaire à une récupération après un arrêt brutal. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### `checkpoint_timeout (integer)`

Temps maximum entre deux points de vérification automatique des WAL, en secondes. La valeur par défaut est de cinq minutes. Augmenter ce paramètre peut accroître le temps nécessaire à une récupération après un arrêt brutal. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### `checkpoint_completion_target (floating point)`

Pécise la cible pour la fin du CHECKPOINT, sous la format d'une fraction de temps entre deux CHECKPOINT. La valeur par défaut est 0.5. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### `checkpoint_warning (integer)`

Si deux points de vérification imposés par le remplissage des fichiers segment interviennent dans un délai plus court que celui indiqué par ce paramètre (ce qui laisse supposer qu'il faut augmenter la valeur du paramètre `checkpoint_segments`), un message est écrit dans le fichier de traces du serveur. Par défaut, 30 secondes. Une valeur nulle (0) désactive cet avertissement. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## 18.5.3. Archivage

### `archive_mode (boolean)`

Quand `archive_mode` est activé, les segments WAL remplis peuvent être archivés en configurant `archive_command`. `archive_mode` et `archive_command` sont des variables séparées de façon à ce que `archive_command` puisse être modifiée sans quitter le mode d'archivage. Ce paramètre ne peut être configuré qu'au lancement du serveur. `wal_level` doit être configuré à `archive` ou `hot_standby` pour activer `archive_mode`.

### `archive_command (string)`

Commande shell à exécuter pour archiver un segment terminé de la série des fichiers WAL. Tout %p dans la chaîne est remplacé par le chemin du fichier à archiver et tout %f par le seul nom du fichier. (Le chemin est relatif au répertoire de travail du serveur, c'est-à-dire le répertoire de données du cluster.) %% est utilisé pour intégrer un caractère % dans la commande. Il est important que la commande renvoie un code zéro seulement si elle a réussi l'archivage. Pour plus d'informations, voir Section 24.3.1, « Configurer l'archivage WAL ».

Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est ignoré sauf si `archive_mode` a été activé au lancement du serveur. Si `archive_command` est une chaîne vide (la valeur par défaut) alors que `archive_mode` est activé, alors l'archivage des journaux de transactions est désactivé temporairement mais le serveur continue d'accumuler les fichiers des journaux de transactions dans l'espoir qu'une commande lui soit rapidement proposée. Configurer `archive_command` à une commande qui ne fait rien tout en renvoyant true, par exemple /bin/true (REM sur Windows), désactive l'archivage mais casse aussi la chaîne des fichiers des journaux de transactions nécessaires pour la restauration d'une archive. Cela ne doit donc être utilisé quand lors de circonstances inhabituelles.

.../...  
.../...  
.../...

# Annexes

## 18.7. Remonter et tracer les erreurs

### 18.7.1. Où tracer

#### log\_destination (string)

PostgreSQL™ supporte plusieurs méthodes pour la journalisation des messages du serveur, dont `stderr`, `csvlog` et `syslog`. Sur Windows, `eventlog` est aussi supporté. Ce paramètre se configure avec la liste des destinations souhaitées séparées par des virgules. Par défaut, les traces ne sont dirigées que vers `stderr`. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

Si `csvlog` est la valeur de `log_destination`, les entrées du journal applicatif sont enregistrées dans le format CSV (« comma separated value »), ce qui est bien pratique pour les charger dans des programmes. Voir Section 18.7.4, « Utiliser les journaux au format CSV » pour les détails. `logging_collector` doit être activé pour produire des journaux applicatifs au format CSV.



#### Note

Sur la plupart des systèmes Unix, il est nécessaire de modifier la configuration du démon `syslog` pour utiliser l'option `syslog` de `log_destination`. PostgreSQL™ peut tracer dans les niveaux `syslog LOCAL0` à `LOCAL7` (voir `syslog_facility`) mais la configuration par défaut de `syslog` sur la plupart des plateformes ignore de tels messages. Il faut ajouter une ligne similaire à :

```
local0.*      /var/log/postgresql
```

dans le fichier de configuration de `syslog` pour obtenir ce type de journalisation.

#### logging\_collector (boolean)

Ce paramètre capture les messages au format texte et CSV envoyé à `stderr` et les redirige dans des journaux applicatifs. Cette approche est souvent plus utile que la journalisation avec `syslog`, car certains messages peuvent ne pas apparaître dans `syslog` (les messages d'échec de l'éditeur de liens en sont un bon exemple). Ce paramètre ne peut être configuré qu'au lancement du serveur.



#### Note

Le récupérateur des traces est conçu pour ne jamais perdre de messages. Cela signifie que, dans le cas d'une charge extrêmement haute, les processus serveur pourraient être bloqués à cause de l'envoi de messages de trace supplémentaires. Le collecteur pourrait accumuler dans ce cas du retard. `syslog` préfère supprimer des messages s'il ne peut pas les écrire. Il est donc moins fiable dans ces cas mais il ne bloquera pas le reste du système.

#### log\_directory (string)

Lorsque `logging_collector` est activé, ce paramètre détermine le répertoire dans lequel les fichiers de trace sont créés. Il peut s'agir d'un chemin absolu ou d'un chemin relatif au répertoire des données du cluster. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

#### log\_filename (string)

Lorsque `logging_collector` est activé, ce paramètre indique les noms des journaux applicatifs créés. La valeur est traitée comme un motif `strftime`. Ainsi les échappements `%` peuvent être utilisés pour indiquer des noms de fichiers horodatés. (S'il y a des échappements `%` dépendant des fuseaux horaires, le calcul se fait dans le fuseau précisé par `log_timezone`.) Notez que la fonction `strftime` du système n'est pas utilisée directement, ce qui entraîne que les extensions spécifiques à la plateforme (non-standard) ne fonctionneront pas.

Si vous spécifiez un nom de fichier sans échappements, vous devriez prévoir d'utiliser un utilitaire de rotation des journaux pour éviter le risque de remplir le disque entier. Dans les versions précédentes à 8.4, si aucun échappement `%` n'était présent, PostgreSQL™ aurait ajouté l'epoch de la date de création du nouveau journal applicatif mais ce n'est plus le cas.

Si la sortie au format CSV est activée dans `log_destination`, `.csv` est automatiquement ajouté au nom du journal horodaté. (Si `log_filename` se termine en `.log`, le suffixe est simplement remplacé.) Dans le cas de l'exemple ci-dessus, le nom du fichier CSV est `server_log.1093827753.csv`.

Ce paramètre ne peut être positionné que dans le fichier `postgresql.conf` ou en ligne de commande.

#### log\_rotation\_age (integer)

Lorsque `logging_collector` est activé, ce paramètre détermine la durée de vie maximale (en minutes) d'un journal individuel. Passé ce délai, un nouveau journal est créé. Initialiser ce paramètre à zéro désactive la création en temps compté de nouveaux journaux. Ce paramètre ne peut qu'être configuré dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

#### log\_rotation\_size (integer)

Lorsque `logging_collector` est activé, ce paramètre détermine la taille maximale (en kilooctets) d'un journal individuel. Passé cette taille, un nouveau journal est créé. Initialiser cette taille à zéro désactive la création en taille comptée de nouveaux journaux. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

# Annexes

## `log_truncate_on_rotation (boolean)`

Lorsque `logging_collector` est activé, ce paramètre impose à PostgreSQL™ de vider (écraser), plutôt qu'ajouter à, tout fichier journal dont le nom existe déjà. Toutefois, cet écrasement ne survient qu'à partir du moment où un nouveau fichier doit être ouvert du fait d'une rotation par temps compté, et non pas à la suite du démarrage du serveur ou d'une rotation par taille comptée. Si ce paramètre est désactivé (off), les traces sont, dans tous les cas, ajoutées aux fichiers qui existent déjà.

Par exemple, si ce paramètre est utilisé en combinaison avec un `log_filename` tel que `postgresql-%H.log`, il en résulte la génération de 24 journaux (un par heure) écrasés de façon cyclique.

Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

Exemple : pour conserver sept jours de traces, un fichier par jour nommé `server_log.Mon`, `server_log.Tue`, etc. et écraser automatiquement les traces de la semaine précédente avec celles de la semaine courante, on positionne `log_filename` à `server_log.%a`, `log_truncate_on_rotation` à `on` et `log_rotation_age` à 1440.

Exemple : pour conserver 24 heures de traces, un journal par heure, toute en effectuant la rotation plus tôt si le journal dépasse 1 Go, on positionne `log_filename` à `server_log.%H%M`, `log_truncate_on_rotation` à `on`, `log_rotation_age` à 60 et `log_rotation_size` à 1000000. Inclure %M dans `log_filename` permet à toute rotation par taille comptée qui survient d'utiliser un nom de fichier distinct du nom initial horodaté.

## `syslog_facility (enum)`

Lorsque les traces syslog sont activées, ce paramètre fixe le niveau (« facility ») utilisé par syslog. Les différentes possibilités sont LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7 ; LOCAL0 étant la valeur par défaut. Voir aussi la documentation du démon syslog du serveur. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## `syslog_ident (string)`

Si syslog est activé, ce paramètre fixe le nom du programme utilisé pour identifier les messages PostgreSQL™ dans les traces de syslog. La valeur par défaut est `postgres`. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## `silent_mode (boolean)`

Exécute silencieusement le serveur. Si ce paramètre est configuré, le serveur démarre automatiquement en tâche de fond et tout terminal de contrôle est dissocié. Ce paramètre ne peut être configuré qu'au démarrage du serveur.



### Attention

Quand ce paramètre est activé, la sortie standard et l'erreur standard sont redirigées vers le fichier `postmaster.log` dans le répertoire des données. Ce fichier ne bénéficie pas du système de rotation, donc il grossira indéfiniment sauf si la sortie du serveur est renvoyée ailleurs par d'autres paramétrages. Il est recommandé de configurer à `log_destination` à `syslog` ou que `logging_collector` soit activé lors de l'utilisation de cette option. Même avec ces mesures, les erreurs rapportées très tôt lors du démarrage pourraient apparaître dans le fichier `postmaster.log` plutôt que dans la destination normal des traces.

## 18.7.2. Quand tracer

### `client_min_messages (enum)`

Contrôle les niveaux de message envoyés au client. Les valeurs valides sont DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, LOG, NOTICE, WARNING, ERROR, FATAL, et PANIC. Chaque niveau inclut tous les niveaux qui le suivent. Plus on progresse dans la liste, plus le nombre de messages envoyés est faible. NOTICE est la valeur par défaut. LOG a ici une portée différente de celle de `log_min_messages`.

### `log_min_messages (enum)`

Contrôle les niveaux de message écrits dans les traces du serveur. Les valeurs valides sont DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, INFO, NOTICE, WARNING, ERROR, LOG, FATAL et PANIC. Chaque niveau inclut tous les niveaux qui le suivent. Plus on progresse dans la liste, plus le nombre de messages envoyés est faible. WARNING est la valeur par défaut. LOG a ici une portée différente de celle de `client_min_messages`. Seuls les superutilisateurs peuvent modifier la valeur de ce paramètre.

### `log_min_error_statement (enum)`

Contrôle si l'instruction SQL à l'origine d'une erreur doit être enregistrée dans les traces du serveur. L'instruction SQL en cours est incluse dans les traces pour tout message de严重ité indiquée ou supérieure. Les valeurs valides sont DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, INFO, NOTICE, WARNING, ERROR, LOG, FATAL et PANIC. ERROR est la valeur par défaut, ce qui signifie que les instructions à l'origine d'erreurs, de messages applicatifs, d'erreurs fatales ou de paniques sont tracées. Pour réellement désactiver le traçage des instructions échouées, ce paramètre doit être positionné à PANIC. Seuls les superutilisateurs peuvent modifier la valeur de ce paramètre.

# Annexes

## `log_min_duration_statement (integer)`

Trace la durée de toute instruction terminée dont le temps d'exécution égale ou dépasse ce nombre de millisecondes. Positionné à zéro, les durées de toutes les instructions sont tracées. -1 (valeur par défaut) désactive ces traces.

Par exemple, si le paramètre est positionné à 250ms, alors toutes les instructions SQL dont la durée est supérieure ou égale à 250 ms sont tracées.

Il est utile d'activer ce paramètre pour tracer les requêtes non optimisées des applications. Seuls les superutilisateurs peuvent modifier cette configuration.

Pour les clients utilisant le protocole de requêtage étendu, les durées des étapes Parse (analyse), Bind (lien) et Execute (exécution) sont tracées indépendamment.

### Note



Lorsque cette option est utilisée avec `log_statement`, le texte des instructions tracées du fait de `log_statement` n'est pas répété dans le message de trace de la durée. Si `syslog` n'est pas utilisé, il est recommandé de tracer le PID ou l'ID de session à l'aide de `log_line_prefix` de façon à pouvoir lier le message de l'instruction au message de durée par cet identifiant.

Tableau 18.1. « Niveaux de sévérité des messages » explique les niveaux de sévérité des messages utilisés par PostgreSQL™. Si la journalisation est envoyée à `syslog` ou à l'`eventlog` de Windows, les niveaux de sévérité sont traduits comme indiqué ci-dessous.

Tableau 18.1. Niveaux de sévérité des messages

Sévérité	Usage	<code>syslog</code>	<code>eventlog</code>
DEBUG1..DEBUG5	Fournit des informations successivement plus détaillées à destination des développeurs.	DEBUG	INFORMATION
INFO	Fournit des informations implicitement demandées par l'utilisateur, par exemple la sortie de VACUUM VERBOSE.	INFO	INFORMATION
NOTICE	Fournit des informations éventuellement utiles aux utilisateurs, par exemple la troncation des identifiants longs.	NOTICE	INFORMATION
WARNING	Fournit des messages d'avertissement sur d'éventuels problèmes. Par exemple, un COMMIT en dehors d'un bloc de transaction.	NOTICE	WARNING
ERROR	Rapporte l'erreur qui a causé l'annulation de la commande en cours.	WARNING	ERROR
LOG	Rapporte des informations à destination des administrateurs. Par exemple, l'activité des points de vérification.	INFO	INFORMATION
FATAL	Rapporte l'erreur qui a causé la fin de la session en cours.	ERR	ERROR
PANIC	Rapporte l'erreur qui a causé la fin de toutes les sessions.	CRIT	ERROR

# Annexes

## 18.7.3. Que tracer

`application_name (string)`

Le paramètre `application_name` peut être tout chaîne de moins de NAMEDATALEN caractères (64 caractères après une compilation standard). Il est typiquement configuré lors de la connexion d'une application au serveur. Le nom sera affiché dans la vue `pg_stat_activity` et inclus dans les traces du journal au format CSV. Il peut aussi être inclus dans les autres formats de traces en configurant le paramètre `log_line_prefix`. Tout caractère ASCII affichable peut être utilisé. Les autres caractères seront remplacés par des points d'interrogation (?).

`debug_print_parse (boolean), debug_print_rewritten (boolean), debug_print_plan (boolean)`

Ces paramètres activent plusieurs sorties de débogage. Quand positionnés, il affichent l'arbre d'interprétation résultant, la sortie de la réécriture de requête, ou le plan d'exécution pour chaque requête exécutée. Ces messages sont émis au niveau de trace LOG, par conséquent ils apparaîtront dans le journal applicatif du serveur, mais ne seront pas envoyés au client. Vous pouvez changer cela en ajustant `client_min_messages` et/ou `log_min_messages`. Ces paramètres sont désactivés par défaut.

`debug_pretty_print (boolean)`

Quand positionné, `debug_pretty_print` indente les messages produits par `debug_print_parse`, `debug_print_rewritten`, ou `debug_print_plan`. Le résultat est une sortie plus lisible mais plus verbueuse que le format « compact » utilisé quand ce paramètre est à off. La valeur par défaut est 'on'.

`log_checkpoints (boolean)`

Trace les points de vérification dans les journaux applicatifs. Diverses statistiques concernant chaque point de vérification sont incluses dans les journaux applicatifs, dont le nombre de tampons écrits et le temps passé à les écrire. Désactivé par défaut, ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

`log_connections (boolean)`

Trace chaque tentative de connexion sur le serveur, ainsi que la réussite de l'authentification du client. Désactivé par défaut, ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.



### Note

Quelques programmes clients, comme psql, tentent de se connecter deux fois pour déterminer si un mot de passe est nécessaire, des messages « connection received » dupliqués n'indiquent donc pas forcément un problème.

`log_disconnections (boolean)`

Affiche dans les traces du serveur une ligne similaire à `log_connections` mais à la fin d'une session, en incluant la durée de la session. Désactivé par défaut, ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

`log_duration (boolean)`

Trace la durée de toute instruction exécutée. Désactivé par défaut (off), seuls les superutilisateurs peuvent modifier ce paramètre.

Pour les clients utilisant le protocole de requêtage étendu, les durées des étapes Parse (analyse), Bind (lien) et Execute (exécution) sont tracées indépendamment.



### Note

À la différence de `log_min_duration_statement`, ce paramètre ne force pas le traçage du texte des requêtes. De ce fait, si `log_duration` est activé (on) et que `log_min_duration_statement` a une valeur positive, toutes les durées sont tracées mais le texte de la requête n'est inclus que pour les instructions qui dépassent la limite. Ce comportement peut être utile pour récupérer des statistiques sur les installations à forte charge.

`log_error_verbosity (enum)`

Contrôle la quantité de détails écrit dans les traces pour chaque message tracé. Les valeurs valides sont TERSE, DEFAULT et VERBOSE, chacun ajoutant plus de champs aux messages affichés. TERSE exclut des traces les informations de niveau DETAIL, HINT, QUERY et CONTEXT. La sortie VERBOSE inclut le code d'erreur SQLSTATE, le nom du code source, le nom de la fonction et le numéro de la ligne qui a généré l'erreur. Seuls les superutilisateurs peuvent modifier ce paramètre.

`log_hostname (boolean)`

Par défaut, les traces de connexion n'affichent que l'adresse IP de l'hôte se connectant. Activer ce paramètre permet de tracer aussi le nom de l'hôte. En fonction de la configuration de la résolution de nom d'hôte, les performances peuvent être pénalisées. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

`log_line_prefix (string)`

Il s'agit d'une chaîne de style printf affichée au début de chaque ligne de trace. Les caractères % débutent des « séquences d'échappement » qui sont remplacées avec l'information de statut décrite ci-dessous. Les échappements non reconnus sont ignorés. Les autres caractères sont copiés directement dans la trace. Certains échappements ne sont reconnus que par les processus de session et ne s'appliquent pas aux processus en tâche de fond comme le processus serveur principal. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. La valeur par défaut est une chaîne vide.

# Annexes

Echappement	Produit	Session seule
%a	Nom de l'application	yes
%u	Nom de l'utilisateur	oui
%d	Nom de la base de données	oui
%r	Nom ou adresse IP de l'hôte distant et port distant	oui
%h	Nom d'hôte distant ou adresse IP	yes
%p	ID du processus	non
%t	Estampille temporelle sans millisecondes	non
%m	Estampille temporelle avec millisecondes	non
%i	Balise de commande : type de commande	oui
%e	état SQL	no
%c	ID de session : voir ci-dessous	non
%l	Numéro de la ligne de trace de chaque session ou processus, commençant à 1	non
%s	Estampille temporelle du lancement du processus	oui
%v	Identifiant virtuel de transaction (backendID/localXID)	no
%x	ID de la transaction (0 si aucune affectée)	non
%q	Ne produit aucune sortie, mais indique aux autres processus de stopper à cet endroit de la chaîne. Ignoré par les processus de session.	non
%%	%	non

L'échappement %c affiche un identifiant de session quasi-unique constitué de deux nombres hexadécimaux sur quatre octets (sans les zéros initiaux) et séparés par un point. Les nombres représentent l'heure de lancement du processus et l'identifiant du processus. %c peut donc aussi être utilisé comme une manière de raccourcir l'affichage de ces éléments. Par exemple, pour générer l'identifiant de session à partir de pg\_stat\_activity, utilisez cette requête :

```
SELECT to_hex(EXTRACT(EPOCH FROM backend_start)::integer) || '.' ||  
       to_hex(procpid)  
  FROM pg_stat_activity;
```



## Astuce

Si log\_line\_prefix est différent d'une chaîne vide, il est intéressant d'ajouter une espace en fin de chaîne pour créer une séparation visuelle avec le reste de la ligne. Un caractère de ponctuation peut aussi être utilisé.



## Astuce

syslog produit ses propres informations d'horodatage et d'identifiant du processus. Ces échappements n'ont donc que peu d'intérêt avec syslog.

`log_lock_waits(boolean)`

Contrôle si une trace applicative est écrite quand une session attend plus longtemps que deadlock\_timeout pour acquérir un verrou. Ceci est utile pour déterminer si les attentes de verrous sont la cause des pertes de performance. Désactivé (off) par défaut.

`log_statement(enum)`

Contrôle les instructions SQL à tracer. Les valeurs valides sont none (off), ddl, mod et all (toutes les instructions). ddl trace toutes les commandes de définition comme CREATE, ALTER et DROP. mod trace toutes les instructions ddl ainsi que les instructions de modification de données INSERT, UPDATE, DELETE, TRUNCATE et COPY FROM. Les instructions PREPARE, EXECUTE et EXPLAIN ANALYZE sont aussi tracées si la commande qui les contient est d'un type approprié. Pour les clients utilisant le protocole de requête étendu, la trace survient quand un message Execute est reçu et les valeurs des paramètres de Bind sont incluses (avec doublement de tout guillemet simple embarqué).

La valeur par défaut est none. Seuls les superutilisateurs peuvent changer ce paramétrage.



## Note

Les instructions qui contiennent de simples erreurs de syntaxe ne sont pas tracées même si log\_statement est positionné à all car la trace n'est émise qu'après qu'une analyse basique soit réalisée pour déterminer le type d'instruction. Dans le cas du protocole de requête étendu, ce paramètre ne trace pas les instructions qui échouent avant la phase Execute (c'est-à-dire pendant l'analyse et la planification). log\_min\_error\_statement doit être positionné à ERROR pour tracer ce type d'instructions.

# Annexes

## log\_temp\_files (integer)

Contrôle l'écriture de traces sur l'utilisation des fichiers temporaires (noms et tailles). Les fichiers temporaires peuvent être créés pour des tris, des hachages et des résultats temporaires de requête. Une entrée de journal est générée pour chaque fichier temporaire au moment où il est effacé. Zéro implique une trace des informations sur tous les fichiers temporaires alors qu'une valeur positive ne trace que les fichiers dont la taille est supérieure ou égale au nombre indiqué (en kilo-octets). La valeur par défaut est -1, ce qui a pour effet de désactiver les traces. Seuls les superutilisateurs peuvent modifier ce paramètre.

## log\_timezone (string)

Configure le fuseau horaire utilisé par l'horodatage des traces. Contrairement à timezone, cette valeur est valable pour le cluster complet, de façon à ce que toutes les sessions utilisent le même. La valeur par défaut est unknown, ce qui signifie que l'environnement système sera utilisé pour trouver cette information. Voir Section 8.5.3, « Fuseaux horaires » pour plus d'informations. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

.../  
.../...

## 18.8.1. Collecteur de statistiques sur les requêtes et les index

Ces paramètres contrôlent la collecte de statistiques de niveau serveur. Lorsque celle-ci est activée, les données produites peuvent être visualisées à travers la famille de vues systèmes `pg_stat` et `pg_statio`. On peut se reporter à Chapitre 27, Surveiller l'activité de la base de données pour plus d'informations.

### track\_activities (boolean)

Active la collecte d'informations sur la commande en cours d'exécution dans chaque session, avec l'heure de démarrage de la commande. Ce paramètre est activé par défaut. Même si le paramètre est activé, cette information n'est pas visible par tous les utilisateurs, mais uniquement par les superutilisateurs et l'utilisateur possédant la session traitée ; de ce fait, cela ne représente pas une faille de sécurité. Seuls les superutilisateurs peuvent modifier ce paramètre.

### track\_activity\_query\_size (integer)

Spécifie le nombre d'octets réservés pour suivre la commande en cours d'exécution pour chaque session active, pour le champ `pg_stat_activity.current_query`. La valeur par défaut est 1024. Ce paramètre ne peut être positionné qu'au démarrage du serveur.

### track\_counts (boolean)

Active la récupération de statistiques sur l'activité de la base de données. Ce paramètre est activé par défaut car le processus autovacuum utilise les informations ainsi récupérées. Seuls les superutilisateurs peuvent modifier ce paramètre.

.../  
.../...

## 18.9. Nettoyage (vacuum) automatique

Ces paramètres contrôlent le comportement de la fonctionnalité appelée `autovacuum`. On peut se référer à la Section 23.1.5, « Le démon auto-vacuum » pour plus de détails.

### autovacuum (boolean)

Contrôle si le serveur doit démarrer le démon d'autovacuum. Celui-ci est activé par défaut. `track_counts` doit aussi être activé pour que ce démon soit démarré. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

Même si ce paramètre est désactivé, le système lance les processus autovacuum nécessaires pour empêcher le bouclage des identifiants de transaction. Voir Section 23.1.4, « Éviter les cycles des identifiants de transactions » pour plus d'informations.

### log\_autovacuum\_min\_duration (integer)

Trace chaque action réalisée par l'autovacuum si elle dure chacune plus de ce nombre de millisecondes. Le configurer à zéro trace toutes les actions de l'autovacuum. La valeur par défaut, -1, désactive les traces des actions de l'autovacuum.

Par exemple, s'il est configuré à 250ms, toutes les opérations VACUUM et ANALYZE qui durent plus de 250 ms sont tracées. Activer ce paramètre peut être utile pour tracer l'activité de l'autovacuum. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

### autovacuum\_max\_workers (integer)

Indique le nombre maximum de processus autovacuum (autre que le lanceur d'autovacuum) qui peuvent être exécutés simultanément. La valeur par défaut est 3. Ce paramètre ne peut être configuré qu'au lancement du serveur.

# Annexes

---

## autovacuum\_naptime (integer)

Indique le délai minimum entre les tours d'activité du démon autovacuum sur une base. À chaque tour, le démon examine une base de données et lance les commandes VACUUM et ANALYZE nécessaires aux tables de cette base. Le délai, mesuré en secondes, vaut, par défaut, une minute (1min). Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande.

## autovacuum\_vacuum\_threshold (integer)

Indique le nombre minimum de lignes mises à jour ou supprimées nécessaires pour déclencher un VACUUM sur une table. La valeur par défaut est de 50 lignes. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de surcharger ce paramètre pour toute table en modifiant les paramètres de stockage.

## autovacuum\_analyze\_threshold (integer)

Indique le nombre minimum de lignes insérées, mises à jour ou supprimées nécessaires pour déclencher un ANALYZE sur une table. La valeur par défaut est de 50 lignes. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de surcharger ce paramètre pour toute table en modifiant les paramètres de stockage.

## autovacuum\_vacuum\_scale\_factor (floating point)

Indique la fraction de taille de la table à ajouter à `autovacuum_vacuum_threshold` pour décider du moment auquel déclencher un VACUUM. La valeur par défaut est 0.2 (20 % de la taille de la table). Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de surcharger ce paramètre pour toute table en modifiant les paramètres de stockage.

## autovacuum\_analyze\_scale\_factor (floating point)

Indique la fraction de taille de la table à ajouter à `autovacuum_analyze_threshold` pour décider du moment auquel déclencher une commande ANALYZE. La valeur par défaut est 0.1 (10 % de la taille de la table). Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de surcharger ce paramètre pour toute table en modifiant les paramètres de stockage.

## autovacuum\_freeze\_max\_age (integer)

Indique l'âge maximum (en transactions) que le champ `pg_class.relfrozenxid` d'une table peut atteindre avant qu'une opération VACUUM ne soit forcée pour empêcher la réinitialisation de l'ID de transaction sur cette table. Le système lance les processus autovacuum pour éviter ce bouclage même si l'autovacuum est désactivé. La valeur par défaut est de 200 millions de transactions. Ce paramètre n'est lu qu'au démarrage du serveur mais il peut être diminué pour toute table en modifiant les paramètres de stockage. Pour plus d'informations, voir Section 23.1.4, « Éviter les cycles des identifiants de transactions ».

## autovacuum\_vacuum\_cost\_delay (integer)

Indique la valeur du coût de délai utilisée dans les opérations de VACUUM. Si -1 est indiqué, la valeur habituelle de `vacuum_cost_delay` est utilisée. La valeur par défaut est 20 millisecondes. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de le surcharger pour toute table en modifiant les paramètres de stockage.

## autovacuum\_vacuum\_cost\_limit (integer)

Indique la valeur de coût limite utilisée dans les opérations de VACUUM automatiques. Si -1 est indiqué (valeur par défaut), la valeur courante de `vacuum_cost_limit` est utilisée. La valeur est distribuée proportionnellement entre les processus autovacuum en cours d'exécution, s'il y en a plus d'un, de sorte que la somme des limites de chaque processus ne dépasse jamais la limite de cette variable. Ce paramètre ne peut être configuré que dans le fichier `postgresql.conf` ou indiqué sur la ligne de commande. Il est possible de le surcharger pour toute table par `changing storage parameters`.

# Annexes

---

## 18.10. Valeurs par défaut des connexions client

### 18.10.1. Comportement des instructions

#### `search_path (string)`

Cette variable précise l'ordre dans lequel les schémas sont parcourus lorsqu'un objet (table, type de données, fonction, etc.) est référencé par un simple nom sans schéma indiqué. Lorsque des objets de noms identiques existent dans plusieurs schémas, c'est le premier trouvé dans le chemin de recherche qui est utilisé. Il ne peut être fait référence à un objet qui ne fait partie d aucun des schémas indiqués dans le chemin de recherche qu'en précisant son schéma conteneur avec un nom qualifié (avec un point).

`search_path` doit contenir une liste de noms de schémas séparés par des virgules. Si un des éléments de la liste est la valeur spéciale `$user`, alors le schéma dont le nom correspond à la valeur renournée par `SESSION_USER` est substitué, s'il existe (sinon `$user` est ignoré).

Le schéma du catalogue système, `pg_catalog`, est toujours parcouru, qu'il soit ou non mentionné dans le chemin. Mentionné, il est alors parcouru dans l'ordre indiqué. Dans le cas contraire, il est parcouru avant tout autre élément du chemin.

De même, le schéma des tables temporaires, `pg_temp_nn`, s'il existe, est toujours parcouru. Il peut être explicitement ajouté au chemin à l'aide de l'alias `pg_temp`. S'il n'en fait pas partie, la recherche commence par lui (avant même `pg_catalog`). Néanmoins, seuls les noms de relation (table, vue, séquence, etc.) et de type de données sont recherchés dans le schéma temporaire. Aucune fonction et aucun opérateur n'y est jamais recherché.

Lorsque des objets sont créés sans précision de schéma cible particulier, ils sont placés dans le premier schéma listé dans le chemin de recherche. Une erreur est rapportée si le chemin de recherche est vide.

La valeur par défaut de ce paramètre est '`$user`', 'public' (la deuxième partie est ignorée s'il n'existe pas de schéma nommé `public`). Elle permet l'utilisation partagée d'une base de données (dans laquelle aucun utilisateur n'a de schéma privé et tous partagent l'utilisation de `public`), les schémas privés d'utilisateur ainsi qu'une combinaison de ces deux modes. D'autres effets peuvent être obtenus en modifiant le chemin de recherche par défaut, globalement ou par utilisateur.

La valeur courante réelle du chemin de recherche peut être examinée via la fonction SQL `current_schemas()`. Elle n'est pas identique à la valeur de `search_path`, car `current_schemas()` affiche la façon dont les requêtes apparaissant dans `search_path` sont résolues.

Pour plus d'informations sur la gestion des schémas, voir la Section 5.7, « Schémas ».

#### `default_tablespace (string)`

Cette variable indique le `tablespace` par défaut dans lequel sont créés les objets (tables et index) quand une commande `CREATE` ne l'explique pas.

La valeur est soit le nom d'un `tablespace` soit une chaîne vide pour indiquer l'utilisation du `tablespace` par défaut de la base de données courante. Si la valeur ne correspond pas au nom d'un `tablespace` existant, PostgreSQL™ utilise automatiquement le `tablespace` par défaut de la base de données courante. Si un `tablespace` différent de celui par défaut est indiqué, l'utilisateur doit avoir le droit `CREATE`. Dans le cas contraire, la tentative de création échouera.

Cette variable n'est pas utilisée pour les tables temporaires ; pour elles, `temp_tablespaces` est consulté à la place.

#### `temp_tablespaces (string)`

Cette variable indique le (ou les) `tablespace`(s) dans le(s)quel(s) créer les objets temporaires (tables temporaires et index sur des tables temporaires) quand une commande `CREATE` n'en explicite pas. Les fichiers temporaires créés par les tris de gros ensembles de données sont aussi créés dans ce `tablespace`.

Cette valeur est une liste de noms de `tablespaces`. Quand cette liste contient plus d'un nom, PostgreSQL™ choisit un membre de la liste au hasard à chaque fois qu'un objet temporaire doit être créé. En revanche, dans une transaction, les objets temporaires créés successivement sont placés dans les `tablespaces` successifs de la liste. Si l'élément sélectionné de la liste est une chaîne vide, PostgreSQL™ utilise automatiquement le `tablespace` par défaut de la base en cours.

Si `temp_tablespaces` est configuré interactivement, l'indication d'un `tablespace` inexistant est une erreur. Il en est de même si l'utilisateur n'a pas le droit `CREATE` sur le `tablespace` indiqué. Néanmoins, lors de l'utilisation d'une valeur précédemment configurée, les `tablespaces` qui n'existent pas sont ignorés comme le sont les `tablespaces` pour lesquels l'utilisateur n'a pas le droit `CREATE`. Cette règle s'applique, en particulier, lors de l'utilisation d'une valeur configurée dans le fichier `postgresql.conf`.

La valeur par défaut est une chaîne vide. De ce fait, tous les objets temporaires sont créés dans le `tablespace` par défaut de la base de données courante.

Voir aussi `default_tablespace`.

.../...  
.../...

# Annexes

---

## 18.10.2. Locale et formatage

### `datestyle (string)`

Configure le format d'affichage des valeurs de type date et heure, ainsi que les règles d'interprétation des valeurs ambiguës de dates saisies. Pour des raisons historiques, cette variable contient deux composantes indépendantes : la spécification du format en sortie (ISO, Postgres, SQL ou German) et la spécification en entrée/sortie de l'ordre année/mois/jour (DMY, MDY ou YMD). Elles peuvent être configurées séparément ou ensemble. Les mots clés Euro et European sont des synonymes de DMY ; les mots clés US, NonEuro et NonEuropean sont des synonymes de MDY. Voir la Section 8.5, « Types date/heure » pour plus d'informations. La valeur par défaut est ISO, MDY, mais initdb initialise le fichier de configuration avec une valeur qui correspond au comportement de la locale `lc_time` choisie.

.../  
.../...

### `lc_messages (string)`

Initialise la langue d'affichage des messages. Les valeurs acceptables dépendent du système ; voir Section 22.1, « Support des locales » pour plus d'informations. Si cette variable est initialisée à une chaîne vide (valeur par défaut), alors la valeur est héritée de l'environnement d'exécution du serveur.

Avec certains systèmes, cette catégorie de locale n'existe pas. Initialiser cette variable fonctionne toujours mais n'a aucun effet. De même, il est possible qu'il n'existe pas de traduction des messages dans la langue sélectionnée. Dans ce cas, les messages sont affichés en anglais.

Seuls les superutilisateurs peuvent modifier ce paramètre car il affecte aussi bien les messages envoyés dans les traces du serveur que ceux envoyés au client.

### `lc_monetary (string)`

Initialise la locale à utiliser pour le formatage des montants monétaires (pour la famille de fonctions `to_char`, par exemple). Les valeurs acceptables dépendent du système ; voir la Section 22.1, « Support des locales » pour plus d'informations. Si cette variable est initialisée à une chaîne vide (valeur par défaut), alors la valeur est héritée de l'environnement d'exécution du serveur, et une valeur incorrecte pourrait dégrader la lisibilité des traces du serveur.

### `lc_numeric (string)`

Initialise la locale à utiliser pour le formatage des nombres (pour la famille de fonctions `to_char`, par exemple). Les valeurs acceptables dépendent du système ; voir la Section 22.1, « Support des locales » pour plus d'informations. Si cette variable est initialisée à une chaîne vide (valeur par défaut), alors la valeur est héritée de l'environnement d'exécution du serveur.

### `lc_time (string)`

Initialise la locale à utiliser pour le formatage des valeurs de date et d'heure, par exemple avec la famille de fonctions `to_char`. Les valeurs acceptables dépendent du système ; voir la Section 22.1, « Support des locales » pour plus d'informations. Si cette variable est initialisée à une chaîne vide (valeur par défaut), alors la valeur est héritée de l'environnement d'exécution du serveur.

### `default_text_search_config (string)`

Sélectionne la configuration de recherche plein texte utilisée par les variantes des fonctions de recherche plein texte qui n'ont pas d'argument explicite pour préciser la configuration. Voir Chapitre 12, Recherche plein texte pour plus d'informations. La valeur par défaut est `pg_catalog.simple` mais initdb initialise le fichier de configuration avec une valeur qui correspond à la locale choisie pour `lc_ctype` s'il est possible d'identifier une configuration correspondant à la locale.

.../  
.../...

# Annexes

---

## 18.11. Gestion des verrous

`deadlock_timeout (integer)`

Temps total, en millisecondes, d'attente d'un verrou avant de tester une condition de verrou mort (*deadlock*). Le test de verrou mort est très coûteux, le serveur ne l'effectue donc pas à chaque fois qu'il attend un verrou. Les développeurs supposent (de façon optimiste ?) que les verrous morts sont rares dans les applications en production et attendent simplement un verrou pendant un certain temps avant de lancer une recherche de blocage. Augmenter cette valeur réduit le temps perdu en recherches inutiles de verrous morts mais retarder la détection de vraies erreurs de verrous morts. La valeur par défaut est une seconde (1s), ce qui est probablement la plus petite valeur pratique. Sur un serveur en pleine charge, elle peut être augmentée. Idéalement, ce paramétrage peut dépasser le temps typique d'une transaction de façon à augmenter la probabilité qu'un verrou soit relâché avant que le processus en attente ne décide de lancer une recherche de verrous morts.

Quand `log_lock_waits` est configuré, ce paramètre détermine aussi le temps d'attente avant qu'un message ne soit enregistré dans les journaux concernant cette attente. Pour comprendre ces délais de verrouillage, il peut être utile de configurer `deadlock_timeout` à une valeur extraordinairement basse.

`max_locks_per_transaction (integer)`

La table des verrous partagés trace les verrous sur `max_locks_per_transaction * (max_connections + max_prepared_transactions)` objets (c'est-à-dire des tables) ; de ce fait, au maximum ce nombre d'objets distincts peuvent être verrouillés simultanément. Ce paramètre contrôle le nombre moyen de verrous d'objets alloués pour chaque transaction ; des transactions individuelles peuvent verrouiller plus d'objets tant que l'ensemble des verrous de toutes les transactions tient dans la table des verrous. Il ne s'agit pas du nombre de lignes qui peuvent être verrouillées ; cette valeur n'a pas de limite. La valeur par défaut, 64, s'est toujours avérée suffisante par le passé, mais il est possible de l'augmenter si des clients accèdent à de nombreuses tables différentes au sein d'une unique transaction. Ce paramètre ne peut être initialisé qu'au lancement du serveur.

Augmenter ce paramètre peut obliger PostgreSQL™ à réclamer plus de mémoire partagée System V ou de sémaphores que ne le permet la configuration par défaut du système d'exploitation. Voir la Section 17.4.1, « Mémoire partagée et sémaphore » pour plus d'informations sur la façon d'ajuster ces paramètres, si nécessaire.

Lors de l'exécution d'un serveur en attente, vous devez configurer ce paramètre à la même valeur ou à une valeur plus importante que sur le serveur maître. Sinon, des requêtes pourraient ne pas être autorisées sur le serveur en attente.

.../...  
.../...

# Annexes

---

## 5. Liste des commandes SQL

<b>SQL Commands .....</b>	<b>1</b>
ABORT .....	1
ALTER DATABASE .....	1
ALTER GROUP .....	1
ALTER TABLE .....	1
ALTER TRIGGER .....	1
ALTER USER .....	1
ANALYZE .....	1
BEGIN .....	1
CHECKPOINT .....	1
CLOSE .....	1
CLUSTER .....	1
COMMENT .....	1
COMMIT .....	1
COPY .....	1
CREATE AGGREGATE .....	1
CREATE CAST .....	1
CREATE CONSTRAINT TRIGGER .....	1
CREATE CONVERSION .....	1
CREATE DATABASE .....	1
CREATE DOMAIN .....	1
CREATE FUNCTION .....	1
CREATE GROUP .....	1
CREATE INDEX .....	1
CREATE LANGUAGE .....	1
CREATE OPERATOR .....	1
CREATE OPERATOR CLASS .....	1
CREATE RULE .....	1
CREATE SCHEMA .....	1
CREATE SEQUENCE .....	1
CREATE TABLE .....	1
CREATE TABLE AS .....	1
CREATE TRIGGER .....	1
CREATE TYPE .....	1
CREATE USER .....	1
CREATE VIEW .....	1
DEALLOCATE .....	1
DECLARE .....	1
DELETE .....	1
DROP AGGREGATE .....	1
DROP CAST .....	1
DROP CONVERSION .....	1
DROP DATABASE .....	1
DROP DOMAIN .....	1

# Annexes

---

DROP FUNCTION .....	1
DROP GROUP .....	1
DROP INDEX .....	1
DROP LANGUAGE .....	1
DROP OPERATOR .....	1
DROP OPERATOR CLASS .....	1
DROP RULE .....	1
DROP SCHEMA .....	1
DROP SEQUENCE .....	1
DROP TABLE .....	1
DROP TRIGGER .....	1
DROP TYPE .....	1
DROP USER .....	1
DROP VIEW .....	1
END .....	1
EXECUTE .....	1
EXPLAIN .....	1
FETCH .....	1
GRANT .....	1
INSERT .....	1
LISTEN .....	1
LOAD .....	1
LOCK .....	1
MOVE .....	1
NOTIFY .....	1
PREPARE .....	1
REINDEX .....	1
RESET .....	1
REVOKE .....	1
ROLLBACK .....	1
SELECT .....	1
SELECT INTO .....	1
SET .....	1
SET CONSTRAINTS .....	1
SET SESSION AUTHORIZATION .....	1
SET TRANSACTION .....	1
SHOW .....	1
START TRANSACTION .....	1
TRUNCATE .....	1
UNLISTEN .....	1
UPDATE .....	1
VACUUM .....	1

# Annexes

---

## 6. Tables et vues systèmes

```
pg_aggregate
pg_am
pg_amop
pg_amproc
pg_attrdef
pg_attribute
pg_cast
pg_class
pg_constraint
pg_conversion
pg_database
pg_depend
pg_description
pg_group
pg_index
pg_inherits
pg_language
pg_largeobject
pg_listener
pg_namespace
pg_opclass
pg_operator
pg_proc
pg_rewrite
pg_shadow
pg_statistic
pg_trigger
pg_type
System Views
pg_indexes
pg_locks
pg_rules
pg_settings
pg_stats
pg_tables
pg_user
pg_views
```

# Annexes

---

Catalog Name	Purpose
pg_aggregate	aggregate functions
pg_am	index access methods
pg_amop	access method operators
pg_amproc	access method support procedures
pg_attrdef	column default values
pg_attribute	table columns ("attributes", "fields")
pg_cast	casts (data type conversions)
pg_class	tables, indexes, sequences ("relations")
pg_constraint	check constraints, unique / primary key constraints, foreign key constraints
pg_conversion	encoding conversion information
pg_database	databases within this database cluster
pg_depend	dependencies between database objects
pg_description	descriptions or comments on database objects
pg_group	groups of database users
pg_index	additional index information
pg_inherits	table inheritance hierarchy
pg_language	languages for writing functions
pg_largeobject	large objects
pg_listener	asynchronous notification
pg_namespace	namespaces (schemas)
pg_opclass	index access method operator classes
pg_operator	operators

Catalog Name	Purpose
pg_proc	functions and procedures
pg_rewrite	query rewriter rules
pg_shadow	database users
pg_statistic	optimizer statistics
pg_trigger	triggers
pg_type	data types

# Annexes

---

## 7. Les vues et les fonctions pour les statistiques

View Name	Description
pg_stat_activity	One row per server process, showing process ID, database, user, and current query. The current query column is only available to superusers; for others it reads as null. (Note that because of the collector's reporting delay, current query will only be up-to-date for long-running queries.)
pg_stat_database	One row per database, showing number of active backends, total transactions committed and total rolled back in that database, total disk blocks read, and total number of buffer hits (i.e., block read requests avoided by finding the block already in buffer cache).
pg_stat_all_tables	For each table in the current database, total numbers of sequential and index scans, total numbers of tuples returned by each type of scan, and totals of tuple insertions, updates, and deletes.
pg_stat_sys_tables	Same as pg_stat_all_tables, except that only system tables are shown.
pg_stat_user_tables	Same as pg_stat_all_tables, except that only user tables are shown.
pg_stat_all_indexes	For each index in the current database, the total number of index scans that have used that index, the number of index tuples read, and the number of successfully fetched heap tuples. (This may be less when there are index entries pointing to expired heap tuples.)
pg_stat_sys_indexes	Same as pg_stat_all_indexes, except that only indexes on system tables are shown.
pg_stat_user_indexes	Same as pg_stat_all_indexes, except that only indexes on user tables are shown.

# Annexes

---

View Name	Description
pg_statio_all_tables	For each table in the current database, the total number of disk blocks read from that table, the number of buffer hits, the numbers of disk blocks read and buffer hits in all the indexes of that table, the numbers of disk blocks read and buffer hits from the table's auxiliary TOAST table (if any), and the numbers of disk blocks read and buffer hits for the TOAST table's index.
pg_statio_sys_tables	Same as pg_statio_all_tables, except that only system tables are shown.
pg_statio_user_tables	Same as pg_statio_all_tables, except that only user tables are shown.
pg_statio_all_indexes	For each index in the current database, the numbers of disk blocks read and buffer hits in that index.
pg_statio_sys_indexes	Same as pg_statio_all_indexes, except that only indexes on system tables are shown.
pg_statio_user_indexes	Same as pg_statio_all_indexes, except that only indexes on user tables are shown.
pg_statio_all_sequences	For each sequence object in the current database, the numbers of disk blocks read and buffer hits in that sequence.
pg_statio_sys_sequences	Same as pg_statio_all_sequences, except that only system sequences are shown. (Presently, no system sequences are defined, so this view is always empty.)
pg_statio_user_sequences	Same as pg_statio_all_sequences, except that only user sequences are shown.

# Annexes

---

Function	Return Type	Description
<code>pg_stat_get_db_numbackends()</code>	integer	Number of active backends in database
<code>pg_stat_get_db_xact_commit()</code>	integer	Transactions committed in database
<code>pg_stat_get_db_xact_rollback()</code>	integer	Transactions rolled back in database
<code>pg_stat_get_db_blocks_fetched()</code>	bigint	Number of disk block fetch requests for database
<code>pg_stat_get_db_blocks_hit()</code>	bigint	Number of disk block requests found in cache for database
<code>pg_stat_get_numscans(oid)</code>	bigint	Number of sequential scans done when argument is a table, or number of index scans done when argument is an index
<code>pg_stat_get_tuples_returned()</code>	bigint	Number of tuples read by sequential scans when argument is a table, or number of index tuples read when argument is an index
<code>pg_stat_get_tuples_fetched()</code>	bigint	Number of valid (unexpired) table tuples fetched by sequential scans when argument is a table, or fetched by index scans using this index when argument is an index
<code>pg_stat_get_tuples_inserted()</code>	bigint	Number of tuples inserted into table
<code>pg_stat_get_tuples_updated()</code>	bigint	Number of tuples updated in table
<code>pg_stat_get_tuples_deleted()</code>	bigint	Number of tuples deleted from table
<code>pg_stat_get_blocks_fetched()</code>	bigint	Number of disk block fetch requests for table or index
<code>pg_stat_get_blocks_hit(oid)</code>	bigint	Number of disk block requests found in cache for table or index
<code>pg_stat_get_backend_idset()</code>	set of integer	Set of currently active backend IDs (from 1 to N where N is the number of active backends). See usage example below.
<code>pg_backend_pid()</code>	integer	Process ID of the attached backend
<code>pg_stat_get_backend_pid(in)</code>	integer	Process ID of all backend processes

# Annexes

---

Function	Return Type	Description
<code>pg_stat_get_backend_dbid()</code>	<code>integer</code>	Database ID of backend process
<code>pg_stat_get_backend_userid()</code>	<code>integer</code>	User ID of backend process
<code>pg_stat_get_backend_activity()</code>	<code>text</code>	Current query of backend process (NULL if caller is not superuser)
<code>pg_stat_reset()</code>	<code>boolean</code>	Reset all currently collected statistics.

## 8. Les fonctions

### A. Fonctions PL/pgSQL stockées

Les fonctions sont des sous-programmes stockées dans la base de données permettant la centralisation des traitements et fonctions spécifiques à l'entreprise. Toutes les transactions s'exécutent selon les mêmes règles de gestion.

Les fonctions offrent de nombreux avantages :

- amélioration de la productivité des développements,
- modularité, maintenabilité, réutilisation par d'autres applications,
- exécutables et partageables par les utilisateurs ayant le privilège EXECUTE,
- faire accéder des tables uniquement par des procédures sans passer par les ordres SQL habituels
- ...

Une fonction accepte des paramètres en entrée et en sortie et peut être appelée. Il existe plusieurs types de fonctions utilisateur :

- fonctions en langage de requêtes (fonctions écrites en SQL)
- fonctions en langage procédural (fonctions écrites en PL/pgSQL, PL/Perl, PL/Python, PL/Tcl)
- fonctions internes
- fonctions en langage C

Les langages de procédures doivent être installées dans toutes les bases de données amenées à l'utiliser. Pour les langages fournis avec la distribution standard (fonctions écrites en PL/pgSQL, PL/Perl, PL/Python, PL/Tcl), l'installation dans une base se fait avec la commande SQL **CREATE LANGUAGE** ou en ligne de commande avec **createlang** (exemple : `createlang plpgsql prod`). Par défaut, seul le langage PL/pgSQL est installé dans la base template1 et donc toutes les base créées à partir de template1.

La gestion des fonctions se fait avec les commandes SQL **CREATE FUNCTION** ou **CREATE OR REPLACE FUNCTION / ALTER FUNCTION / DROP FUNCTION**. En tant que **dba**, vous pouvez être amené à vérifier l'existence, l'état, le propriétaire, la validation / invalidation d'une fonction.

# Annexes

---

Exemple de fonction en langage PL/pgSQL :

```
test=# CREATE OR REPLACE FUNCTION reverse(varchar) RETURNS varchar AS $PROC$  
test$# DECLARE  
test$#   str_in ALIAS FOR $1;  
test$#   str_out varchar;  
test$#   str_temp varchar;  
test$#   position integer;  
test$# BEGIN  
test$#   -- Initialisation de str_out, sinon sa valeur reste à NULL  
test$#   str_out := '';  
test$#   -- Suppression des espaces en début et fin de chaîne  
test$#   str_temp := trim(both ' ' from str_in);  
test$#   -- position initialisée à la longueur de la chaîne  
test$#   -- la chaîne est traitée à l'envers  
test$#   position := char_length(str_temp);  
test$#   -- Boucle: Inverse l'ordre des caractères d'une chaîne de caractères  
test$# WHILE position > 0 LOOP  
test$#   -- la chaîne donnée en argument est parcourue  
test$#   -- à l'envers,  
test$#   -- et les caractères sont extraits individuellement au  
test$#   -- moyen de la fonction interne substring  
test$#   str_out := str_out || substring(str_temp, position, 1);  
test$#   position := position - 1;  
test$# END LOOP;  
test$# RETURN str_out;  
test$# END;  
test$# $PROC$ LANGUAGE plpgsql  
test$# IMMUTABLE;  
CREATE FUNCTION  
test=# select reverse('PostgreSQL à l''envers');  
  
reverse  
-----  
srevne'l à LQSergtsoP  
(1 row)  
  
test=#
```

# Annexes

---

## 9. Les triggers

### A. Principe

Les triggers (déclencheurs) sont des fonctions (programmes) déclenchées à la suite d'un évènement DML.

Les triggers DML permettent d'exécuter automatiquement des traitements assurant les règles de gestion particulières à l'entreprise généralement plus complexes que les contraintes d'intégrité déclaratives. Ces triggers DML sont associés à une table et seront déclenchés automatiquement lors des modifications DML effectuées sur cette table (INSERT, UPDATE, DELETE).

Un trigger permet de :

- déclencher des fonctions PL/pgSQL, PL/Perl, PL/Python, PL/Tcl stockées dans la base de données,
- réaliser un audit des valeurs des colonnes mémorisées dans la base,
- réaliser des modifications de données implicites et prévues à l'insu de l'utilisateur,
- imposer des procédures de sécurité sophistiquées,
- ...

La gestion des triggers se fait avec les commandes SQL **CREATE TRIGGER / ALTER TRIGGER / DROP TRIGGER**.

En tant que dba, vous pouvez être amené à vérifier l'existence, l'état, le propriétaire d'un trigger.

A la création du trigger, on précisera :

- les requêtes qui déclenchent le trigger (INSERT, UPDATE, DELETE),
- si le trigger est déclenché avant ou après l'exécution de la requête (**BEFORE, AFTER**),
- si le trigger doit être déclenché une seule fois pour toutes les lignes concernées par la requête ou autant de fois que le nombre de lignes concernées par la requête (clause **FOR EACH ROW** ou pas)

# Annexes

## Exemples de trigger DML :

```
test=# CREATE TABLE emp2 (
test(#     nom_employe      text NOT NULL,
test(#     salaire          integer
test(# );
CREATE TABLE

test=# CREATE TABLE emp_audit(
test(#     operation        char(1)  NOT NULL,
test(#     tampon           timestamp NOT NULL,
test(#     id_utilisateur   text      NOT NULL,
test(#     nom_employe      text      NOT NULL,
test(#     salaire          integer
test(# );
CREATE TABLE

test=# CREATE OR REPLACE FUNCTION audit_employe() RETURNS TRIGGER AS $emp_audit$
test$# BEGIN
test$# -- Ajoute une ligne dans emp_audit pour refléter l'opération réalisée
test$# -- sur emp2,
test$# -- utilise la variable spéciale TG_OP pour cette opération.
test$# IF (TG_OP = 'DELETE') THEN
test$#     INSERT INTO emp_audit SELECT 'D', now(), user, OLD.*;
test$#     RETURN OLD;
test$# ELSIF (TG_OP = 'UPDATE') THEN
test$#     INSERT INTO emp_audit SELECT 'U', now(), user, NEW.*;
test$#     RETURN NEW;
test$# ELSIF (TG_OP = 'INSERT') THEN
test$#     INSERT INTO emp_audit SELECT 'I', now(), user, NEW.*;
test$#     RETURN NEW;
test$# END IF;
test$# RETURN NULL; -- le résultat est ignoré car il s'agit d'un trigger AFTER
test$# END;
test$# $emp_audit$ language plpgsql;
CREATE FUNCTION

test=# CREATE TRIGGER emp_audit
test-#     AFTER INSERT OR UPDATE OR DELETE ON emp2
test-#     FOR EACH ROW EXECUTE PROCEDURE audit_employe();
CREATE TRIGGER
```

# Annexes

```
test=# select * from emp2;
nom_employe | salaire
-----+-----
(0 rows)

test=# select * from emp_audit;
operation | tampon | id_utilisateur | nom_employe | salaire
-----+-----+-----+-----+-----+
(0 rows)

test=# insert into emp2 values ('Durand',3000);
INSERT 0 1
test=# insert into emp2 values ('Dupont',4000);
INSERT 0 1
test=# insert into emp2 values ('Dumont',5000);
INSERT 0 1
test=# update emp2 set salaire=5000 where salaire < 5000;
UPDATE 2
test=# delete from emp2 where nom_employe='Durand';
DELETE 1
test=# select * from emp2;

nom_employe | salaire
-----+-----
Dumont      |    5000
Dupont      |    5000
(2 rows)

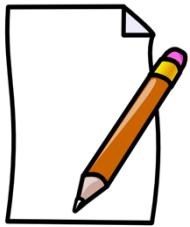
test=# select * from emp_audit;
operation | tampon | id_utilisateur | nom_employe | salaire
-----+-----+-----+-----+-----+
I         | 2008-07-22 15:18:37.373 | postgres   | Durand     | 3000
I         | 2008-07-22 15:18:55.017 | postgres   | Dupont     | 4000
I         | 2008-07-22 15:19:17.515 | postgres   | Dumont     | 5000
U         | 2008-07-22 15:20:08.942 | postgres   | Durand     | 5000
U         | 2008-07-22 15:20:08.942 | postgres   | Dupont     | 5000
D         | 2008-07-22 15:20:23.251 | postgres   | Durand     | 5000
(6 rows)

test=#

```

# Annexes

---





④ N°Azur **0 810 007 689**

PRIX D'UN APPEL LOCAL DEPUIS UN POSTE FIXE

Découvrez également l'ensemble des stages à votre disposition sur notre site

<http://www.m2iformation.fr>