Préambule

PostgreSQL est un serveur de bases de données relationnelles et objet. Il s'agit d'un logiciel libre, distribué sous les termes de la licence BSD, et ce, depuis 1996.

Le projet PostgreSQL, tel que nous le connaissons aujourd'hui, a dix-huit ans. Mais le projet POSTGRES a démarré en 1986, au sein de l'université de Berkeley en Californie, sous la direction de Michael Stonebraker. Ce projet est la suite du SGBD Ingres, d'où le nom (Post-Ingres). En 1994, un interpréteur SQL est ajouté et le code source est distribué sous le nom de Postgres95. Le langage SQL remplace alors le langage PostQUEL et c'est en 1996 que le projet est renommé PostgreSQL.

PostgreSQL est l'un des serveurs de bases de données les plus avancés parmi ceux distribués en tant que logiciels libres. Le terme « avancé » peut qualifier aussi bien l'ensemble des fonctionnalités présentes que la qualité d'écriture du code source ou encore la stabilité et la qualité d'exécution du logiciel.

Outre le code source, c'est l'ensemble du projet, la façon dont il est développé, la façon dont les décisions sont prises ainsi que l'ensemble des développeurs qui peuvent être qualifiés d'ouverts. Il s'agit d'un des points importants permettant de juger de la qualité du logiciel : tout le code source, toutes les décisions concernant l'ajout d'une fonctionnalité ou le rejet d'une modification sont disponibles à tous ceux qui souhaitent s'y intéresser.

L'ensemble des développeurs, quel que soit leur investissement, s'y expose et tout peut être sujet à discussion, chaque point peut être soumis à une question et mérite une réponse.

Administration SQL – Postgresql DB

C’est ce qui fait que PostgreSQL est aujourd’hui un logiciel libre et ouvert, dans tous les sens du terme, et l’un des serveurs de bases de données les plus avancés. Plusieurs dizaines de développeurs, aux quatre coins du globe, contribuent à PostgreSQL, et plusieurs entreprises participent directement au développement de PostgreSQL.

L’ensemble des acteurs est regroupé sous le terme PGDG pour PostgreSQL Developer Group, groupe informel publiant le logiciel. Il existe une équipe centrale dite « core team » et une équipe élargie de « committers » ayant la possibilité de publier le code de PostgreSQL dans le dépôt GIT du projet.

Le développement de PostgreSQL s’articule autour de « commitfest » réguliers et vise à produire une version majeure de PostgreSQL tous les ans. Lors de ces « commitfest », n’importe quel développeur peut proposer une nouvelle fonctionnalité sous forme de « patch », qui sera revu et, après discussions et consensus, sera validé dans le dépôt de source GIT de PostgreSQL.

L’ensemble du code source, la documentation et les listes de discussion sont disponibles sur le site webhttp://www.postgresql.org. Quelques versions binaires, prêtes à être installées, sont disponibles, y compris pour les systèmes Windows. La plupart des distributions GNU/Linux intègrent leurs propres paquets logiciels de PostgreSQL ainsi que des systèmes FreeBSD ou encore Sun Solaris.

Un « wiki » complète la documentation par de nombreuses informations pratiques, que ce soit sur la vie du projet ou des cas pratiques d’utilisation de PostgreSQL : http://wiki.postgresql.org/

Le projet fournit des binaires de toutes les versions supportées pour les systèmes GNU/Linux Red Hat, CentOS, Debian et Ubuntu.

Chaque version majeure est supportée par le projet pendant cinq ans, et des versions mineures sortent donc régulièrement, contenant des correctifs de bugs ou de sécurité. La plus ancienne version supportée au moment de l'écriture de ces lignes est la version 9.0.18, soit avec dix-huit versions mineures, et elle est supportée jusqu'en juillet 2015.

De nombreuses entreprises qui fournissent des services autour de PostgreSQL sont répertoriées dans le site officiel : http://www.postgresql.org/support/professional_support/europe/

L'association PostgreSQLFr participe à la promotion du serveur de bases de données dans la francophonie, notamment en travaillant sur les traductions des manuels et des outils et en animant un site web, comprenant un forum de discussion et des articles techniques et d'informations à l'adresse :http://www.postgresqlfr.org

Présentation des projets

De nombreux logiciels peuvent être associés au serveur PostgreSQL. Ils sont rassemblés dans la ferme de projets PgFoundry. Ils contribuent à la qualité du serveur, en mettant à disposition des outils graphiques, des pilotes pour l'accès aux serveurs depuis différents langages de programmation, de nombreuses extensions et des outils tiers, facilitant la maintenance et l'administration.

L'ensemble des projets hébergés par la ferme de projets PgFoundry est disponible sous licence de logiciels libres. Cette ferme de projets est accessible à l'adresse http://www.pgfoundry.org. Certains projets sont aussi développés dans un autre cadre que la ferme de projets PgFoundry.

Ceux utilisés dans cet ouvrage sont disponibles aux adresses suivantes :

- PgAdmin : http://www.pgadmin.org/
- phpPgAdmin : http://phppgadmin.sf.net/
- PgPool : http://pgfoundry.org/projects/pgpool/
- PgBouncer : https://wiki.postgresql.org/wiki/PgBouncer
- Londiste : https://wiki.postgresql.org/wiki/SkyTools

Objectifs de cet ouvrage

Cet ouvrage couvre la version 9.4 du serveur et vise à proposer un guide à l'administrateur de bases de données, lui donnant une vue d'ensemble du fonctionnement du serveur ainsi que les détails pratiques à connaître pour la maintenance et l'administration quotidienne de PostgreSQL.

Quelques points évoqués permettront à l'administrateur de comprendre l'ensemble des fonctionnalités disponibles pour les développeurs.

Le chapitre Installation couvre l'installation de PostgreSQL sur les systèmes d'exploitation Windows et GNU/Linux, en utilisant les fichiers sources et les fichiers binaires fournis.

Le chapitre Initialisation du système de fichiers couvre la préparation de l'environnement d'exécution du serveur, de l'initialisation du système de fichiers jusqu'au démarrage du serveur.

Le chapitre Connexions couvre les applications clientes pouvant être utilisées par l'exploitant ou l'administrateur, ainsi que les paramètres de sécurité pour l'ouverture de connexions depuis ces applications clientes.

Le chapitre Définition des données résume les différents aspects de PostgreSQL concernant le support du langage SQL.

Le chapitre Programmation évoque le thème de la programmation autour de PostgreSQL, aussi bien du côté serveur, avec les procédures stockées, que du côté client.

Le chapitre Exploitation concerne les différentes tâches d'administration et d'exploitation, de la configuration du serveur aux différentes tâches d'exploitation, en passant par les sauvegardes.

Administration SQL – Postgresql DB

Limites de l'outil

PostgreSQL est un outil dédié au stockage et au traitement de données. Une des questions importantes lors du choix d'un système de gestion de bases de données est celle des tailles de volumes acceptées par le logiciel. De façon générale, PostgreSQL s'appuie largement sur les capacités du système d'exploitation et n'impose donc pas de limites de volumes, à quelques exceptions près :

- La taille d'un champ est limitée à 1 gigaoctet. En pratique, cette taille est limitée par la quantité de mémoire vive disponible pour lire ce champ.
- La taille d'une ligne est limitée à 1,6 téraoctet.
- La taille d'une table est limitée à 32 téraoctets.
- Le nombre maximal de colonnes dans une table peut aller de 250 à 1600, selon la taille d'un bloc et la taille des types de données utilisés pour les colonnes.

Une fois que ces considérations sont prises en compte, PostgreSQL n'impose pas d'autres limites sur la taille d'une base de données, la quantité de lignes dans une table ou la taille d'une ligne, en revanche, le système d'exploitation utilisé peut apporter ses propres limitations.

Sources

PostgreSQL est livré sous la forme d'archives de fichiers sources, lors de la sortie d'une nouvelle version. Il s'agit du mode de distribution par défaut, comme pour tout logiciel libre. C'est donc à partir de ces fichiers sources que seront créés les paquets binaires pour Windows ou GNU/Linux. Le mode opératoire dans ce cas est relativement simple à partir du moment où les outils nécessaires sont préalablement installés. Ces outils disponibles dans toutes les distributions GNU/Linux sont :

- L'outil GNU Make.
- Un compilateur C ISO/ANSI (une version récente de GCC conviendra).
- L'outil tar, avec gzip ou bzip2.
- La bibliothèque GNU Readline.
- La bibliothèque de compression zlib.

De plus, certains outils ou installations complémentaires peuvent être à prévoir :

- Les outils MingW ou Cygwin pour une compilation pour un système Windows.
- Une installation des logiciels Perl, Python ou Tcl pour installer les langages de procédures stockées PL/Perl, PL/python et PL/Tcl.
- La bibliothèque Gettext pour activer le support des langues natives.
- Kerberos, OpenSSL, Pam, s'il est prévu de les utiliser.

Comme pour beaucoup d'autres logiciels libres, la construction des binaires à partir des sources s'appuie sur un script `configure` qui génère les fichiers Makefile, eux-mêmes contenant les instructions destinées au compilateur.

1. Téléchargement des sources

La première étape consiste à télécharger une archive de fichiers sources depuis le site web de PostgreSQL : http://www.postgresql.org/ftp/source/ ou ftp://ftp-archives.postgresql.org/pub/source/.

La version courante est la version 9.4 et la dernière livraison à ce jour est la 9.4.0. Les différents formats de livraison sont disponibles dans le répertoire v9.4.0. Les archives sont au format tar, compressées avec les outils gzip ou bzip2. Les fichiers .tar.bz2 et .tar.gz ont donc le même contenu. À chaque archive correspondent des fichiers .md5 et .sha256 contenant les sommes de contrôle permettant de vérifier la qualité du contenu téléchargé.

Les commandes suivantes permettent de vérifier la somme md5 d'un fichier, en la comparant avec la valeur indiquée dans le fichier .md5 ; exemple :

```
[user]$ cat postgresql-9.4.0.tar.bz2.md5
8cd6e33e1f8d4d2362c8c08bd0e8802b  postgresql-9.4.0.tar.bz2
[user]$ md5sum postgresql-9.4.0.tar.bz2
8cd6e33e1f8d4d2362c8c08bd0e8802b
```

Il est important que les sommes soient identiques. Dans le cas contraire, le téléchargement de l'archive est probablement défectueux.

Une fois le téléchargement de l'archive effectué, il faut l'ouvrir, avec la commande `tar`. Les versions récentes de tar détectent automatiquement la compression utilisée ; dans le cas contraire, il faut le préciser :

```
[user]$ tar xf postgresql-9.4.0.tar.bz2
```

2. Choix des options de compilation

Un répertoire `./postgresql-9.4.0/` est créé contenant, notamment, un script `configure` qui va permettre de préparer la compilation. Différentes options sont disponibles :

```
[user]$ cd postgresql-9.4.0/
[user]$./configure --help
```

Les options essentielles sont :

- `--prefix` : permet de préciser le répertoire d'installation de PostgreSQL. La valeur par défaut est /usr/local/pgsql.
- `--with-perl`, `--with-python` : active le support des procédures stockées, respectivement PL/Tcl, PL/Perl, PL/Python.
- `--with-pgport=PORTNUM` : définit le port TCP, 5432 par défaut.

Administration SQL – Postgresql DB

- `--with-blocksize=BLOCKSIZE` : définit la taille d'un bloc de données, en KB, 8KB par défaut.
- `--with-segsize=SEGSIZE` : définit la taille d'un segment de table, en GB, 1GB par défaut.
- `--with-wal-blocksize=BLOCKSIZE` : définit la taille d'un bloc de WAL, en KB, 8KB par défaut.
- `--with-wal-segsize=SEGSIZE` : définit la taille d'un fichier WAL, en MB, 16MB par défaut.
- `--with-selinux` : active le support de SELinux.
- `--without-readline` : désactive le support de Readline.
- `--without-zlib` : désactive le support de Zlib.
- `--with-gssapi` : active le support de GSSAPI.
- `--with-ldap` : active le support de LDAP.
- `--with-openssl` : active le support des connexions sécurisées.

Une fois le choix des options effectué, il faut exécuter le script `configure` avec ces options pour générer les fichiers Makefile. Par exemple :

```
[user]$./configure --prefix=/usr/local/pgsql940  --with-python --with-openssl
```

3. Compilation

Une fois les tests validés et les fichiers Makefile créés, la commande `make` interprète ces fichiers afin de lancer la compilation proprement dite :

```
[user]$ make
```

Puis, la cible `install` des fichiers Makefile crée le répertoire d'installation et place les fichiers générés dans l'arborescence :

```
[user]$ sudo make install
```

Les extensions sont disponibles dans le sous-répertoire `contrib` et ne sont pas compilées et installées par défaut. Les commandes suivantes permettent la compilation et l'installation des extensions :

```
[user]$ cd contrib
[user]$ make
[user]$ sudo make install
```

À partir de ce moment, les commandes clients, les fichiers de bibliothèques, les extensions et le serveur de PostgreSQL sont installés.

4. Étapes post-installation

Pour terminer l'installation, il faut enregistrer dans les variables d'environnement les répertoires où sont installées les bibliothèques et les commandes de PostgreSQL. Ceci va permettre l'utilisation des commandes dans des scripts ou directement depuis l'interpréteur de commandes :

```
[user]$ export PATH=$PATH:/usr/local/pgsql940/bin
[user]$ export LD_LIBRARY_PATH=/usr/local/pgsql940/lib
```

Les valeurs indiquées tiennent compte de la valeur de l'option `--prefix` du script `configure`. Il suffit de placer ces commandes dans le fichier de configuration de l'interpréteur pour rendre ces réglages permanents.

L'étape suivante de l'installation est la création d'un compte utilisateur non privilégié dans le système d'exploitation qui sera chargé de démarrer le serveur PostgreSQL. Il n'est pas nécessaire que ce compte puisse se connecter au système. Ce compte sera le propriétaire de tous les fichiers des bases de données, il est donc important d'utiliser un compte dédié à PostgreSQL afin d'isoler ces fichiers des autres comptes non privilégiés du système d'exploitation. Le compte utilisateur peut être créé comme suit :

```
[root]# adduser --home=/data/postgresql postgres
```

Une étape importante est l'initialisation du groupe de bases de données qui sera détaillé dans le chapitre suivant. Cette étape crée tous les répertoires et fichiers nécessaires à l'exécution d'un serveur.

Cette commande doit être lancée par l'utilisateur système postgres :

```
[root]# su - postgres
[postgres]$ mkdir data
[postgres]$ initdb data
```

La commande `initdb` est présentée en détail dans le chapitre Initialisation du système de fichiers.

5. Intégration dans le système d'exploitation

Enfin, la dernière étape est l'intégration du système d'exploitation au système de démarrage, grâce à un script de démarrage disponible dans le répertoire `contrib/start-scripts/` des sources de PostgreSQL. Un exemple est disponible pour les systèmes GNU/Linux, FreeBSD et Darwin (Mac OS X). Il suffit de copier le script souhaité dans le répertoire `/etc/init.d/` sous le nom `postgresql` ; par exemple :

```
[root]# cp contrib/start-scripts/linux
/etc/init.d/postgresql
```

puis, de rendre le script exécutable :

```
[root]# chmod a+x /etc/init.d/postgresql
```

Administration SQL – Postgresql DB

Ensuite, il convient de modifier le contenu pour l'adapter aux répertoires d'installation choisis. Les paramètres à ajuster sont la variable `prefix` et la variable `PGDATA` qu'il faut positionner en fonction des choix effectués à l'installation.

Par exemple :

```
prefix=/usr/local/pgsql9.4.0
PGDATA=/data/postgresql/data/
```

Une fois ces paramètres choisis, il faut enregistrer le script dans le système de démarrage. Dans les systèmes GNU/Linux dérivés de Red Hat (Fedora, Mandriva), la commande `chkconfig` permet cela :

```
[root]# chkconfig --add postgresql
```

Puis, il faut démarrer PostgreSQL :

```
[root]# service postgresql start
```

Dans les systèmes GNU/Linux dérivés de Debian, la commande `update-rc.d` permet d'enregistrer le service :

```
[root]# update-rc.d postgresql defaults
```

Administration SQL – Postgresql DB

Puis, il faut démarrer PostgreSQL :

```
[root]# invoke-rc.d postgresql start
```

Après ces commandes, le serveur PostgreSQL doit être démarré.

Linux : distributions Debian et Ubuntu

Le projet Debian fournit des paquetages de PostgreSQL pour sa distribution, mais la version stable actuelle ne fournit que la version 9.1 et la version stable de Ubuntu LTS fournit quant à elle la version 9.3.

Les versions stables de ces distributions ont une durée de vie plus longue que la période entre deux versions de PostgreSQL, ce qui ne leur permet pas d'intégrer toutes les versions majeures de PostgreSQL.

1. Dépôt apt.postgresql.org

Le projet PostgreSQL fournit des paquets binaires des cinq versions majeures pour les distributions Debian et Ubuntu.

Le dépôt fournit l'infrastructure d'hébergement des paquets, permettant de n'ajouter que la source du dépôt au système où doit être installé PostgreSQL.

Quelques paquets doivent être installés avant l'installation afin de déterminer la version du système et d'installer les clés numériques, avec la commande suivante :

```
[root]# apt-get install wget ca-certificate lsb-release
```

Le dépôt est signé par une clé numérique qu'il convient de télécharger préalablement, avec la commande suivante :

Administration SQL – Postgresql DB

```
[root]# wget --quiet -O
- https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -
```

La commande suivante permet de déterminer le nom de la version du système d'exploitation courant :

```
[root]# lsb_release -cs
```

Les lignes suivantes doivent être écrites dans un fichier de dépôts, soit dans le fichier principal `/etc/apt/sources.list`, soit dans un fichier dédié `/etc/apt/sources.list.d/pgdg.list` :

```
deb http://apt.postgresql.org/pub/repos/apt/ nomsysteme-pgdg main 9.4
deb-src http://apt.postgresql.org/pub/repos/apt/ nomsysteme-pgdg main 9.4
```

La version majeure en fin de ligne peut être omise, ce qui permet l'installation de toutes les versions de PostgreSQL simultanément. Quelle que soit la version majeure choisie, le fait de l'indiquer dans le fichier de dépôt permet de sécuriser l'installation, évitant des confusions de versions, par exemple sur des versions de la bibliothèque `libpq`.

Une fois la base de paquets rafraîchie avec la commande `apt-get update`, Les paquets sont disponibles pour l'installation. La commande suivante permet d'interroger la base de données des applications disponibles, en filtrant sur l'expression `^postgresql`.

Administration SQL – Postgresql DB

L'extrait suivant met en évidence les différentes versions disponibles :

```
[root] # apt-cache search ^postgresql
postgresql-9.4 - object-relational SQL database, version 9.4 server
postgresql-client-9.4 - front-end programs for PostgreSQL 9.4
postgresql-contrib-9.4 - additional facilities for PostgreSQL
postgresql-doc-9.4 - documentation for the PostgreSQL database
management system
postgresql-server-dev-9.4 - development files for PostgreSQL 9.4
server-side programming
```

Les paquets principaux sont :

- `postgresql-9.4` : contient les programmes serveurs, les bibliothèques de fonctions et les fichiers partagés servant à initialiser les instances.
- `postgresql-contrib-9.4` : contient les extensions fournies par PostgreSQL. D'autres extensions sont disponibles en tant que paquets individuels, par exemple `postgresql-9.4-ip4r`.
- `postgresql-client-9.4` : contient les programmes clients comme `psql` ou `pg_dump`.

Des scripts spécifiques à ces distributions Debian et Ubuntu ont été écrits et sont disponibles dans les paquets `postgresql-common` et `postgresql-client-common`. Ces scripts permettent d'administrer plusieurs versions différentes de PostgreSQL sur une même machine, ainsi que plusieurs instances d'une même version de PostgreSQL.

Pour installer la version 9.4, il suffit de lancer la commande suivante :

```
[root]# apt-get install postgresql-9.4 postgresql-contrib-9.4
```

Le système de résolution de dépendances installe automatiquement les paquets nécessaires. Par exemple, le paquet libpq5, contenant la bibliothèque implémentant le protocole client/serveur, est installé à cette étape puisque cette bibliothèque est requise parmi les paquets demandés.

Un groupe de base de données est automatiquement créé dans le répertoire`/var/lib/postgresql/9.4/main/`, avec les fichiers de configurations correspondants dans`/etc/postgresql/9.4/main/`. L'instance peut être démarrée avec la commande suivante :

```
[root]# pg_ctlcluster 9.4 main start
```

2. Distributions RPM

La plupart des distributions GNU/Linux qui utilisent le format RPM pour leurs paquetages intègrent une version de PostgreSQL. Il est donc souvent possible d'installer PostgreSQL directement depuis les médias d'installation de la distribution. Néanmoins, un projet hébergé dans la ferme de projet PgFoundry, se charge de réaliser des paquets RPM pour les versions récentes de PostgreSQL.

L'adresse du projet est : http://yum.postgresql.org/

Téléchargement

Les distributions Fedora et Red Hat sont supportées, ainsi que CentOS, Scientific Linux et Oracle Enterprise Linux, les trois dernières étant directement dérivées de Red Hat.

Les binaires fournis par le groupe PostgreSQL sont disponibles directement sur le site web, à l'adressehttp://yum.postgresql.org/repopackages.php. Selon la version de PostgreSQL souhaitée, par exemple la version 9.4 pour Red Hat Enterprise Server 6 pour un processeur de la famille Intel 64 bits, un fichier RPM installe la configuration nécessaire à YUM, en rendant disponibles dans le système les différents paquets de PostgreSQL. Il suffit de télécharger le fichier RPM mis à disposition par le lien de chaque système, ou plus simplement, de copier ce lien et de le passer en argument de la commande `rpm`, comme dans l'exemple suivant :

```
[root]# rpm -ivh http://yum.postgresql.org/9.4/redhat/
rhel-6-x86_64/pgdg-centos94-9.4-1.noarch.rpm
```

Ici, le serveur est situé géographiquement en France ; il est bien sûr possible de choisir un autre serveur.

Ce paquet n'installe pas PostgreSQL mais installe le fichier de dépôt pour YUM, ce qui permet à YUM d'installer les paquets PostgreSQL. Dans l'exemple présenté, le fichier installé est :

```
/etc/yum.repos.d/pgdg-94-centos.repo
```

Les paquets de la version PostgreSQL sont alors disponibles à l'installation.

Description des paquets

L'installation de PostgreSQL est séparée en plusieurs paquets pour permettre de sélectionner les fonctionnalités utiles.

Par exemple, il est possible de n'installer que les logiciels clients et non le serveur, ou de ne pas installer les extensions.

Les différents fichiers correspondent chacun à un besoin bien précis, de telle sorte qu'il n'est pas nécessaire de tous les installer :

- `postgresql94` : ce paquet contient les commandes clients nécessaires pour administrer un serveur local ou distant.
- `postgresql94-server` : ce paquet contient les fichiers nécessaires pour faire fonctionner un serveur PostgreSQL.
- `postgresql94-contrib` : ce paquet contient les contributions pour ajouter des fonctionnalités au serveur.

- `postgresql94-libs` : ce paquet contient les bibliothèques partagées utilisées par les logiciels clients et le serveur.
- `postgresql94-devel` : ce paquet contient tous les fichiers d'en-têtes et les bibliothèques nécessaires au développement d'applications agissant directement avec PostgreSQL.
- `postgresql94-docs` : ce paquet contient les documentations au format HTML.

Installation et mise à jour

L'installation s'effectue grâce à la commande `yum` :

```
[root]# yum install postgresql94 postgresql94-server
postgresql94-contrib postgresql94-docs
```

Dans le cas d'une nouvelle version mineure de PostgreSQL, il est possible de faire simplement une mise à jour en utilisant la commande `yum`, par exemple :

```
[root]# yum update
```

Attention ! cette méthode ne permet pas d'effectuer des mises à jour entre versions majeures de PostgreSQL, comme les versions 9.3 et 9.4. Ceci ne fonctionnera qu'avec des versions mineures, comme les versions 9.4.0 et 9.4.1.

Démarrage du service

Administration SQL – Postgresql DB

La distribution Red Hat Linux utilise, comme beaucoup d'autres, le système de démarrage SysV. Les fichiers RPM installent un script de démarrage postgresql dans le répertoire `/etc/init.d/`.

Ce script contient le chemin vers les données, initialisé dans la variable `PGDATA`. À l'installation, les données ne sont pas créées. Le chemin par défaut est `/var/lib/pgsql/9.4/data`. Il peut être modifié en éditant le script `/etc/init.d/postgresql-9.4`. Les commandes suivantes permettent d'initialiser et de démarrer le serveur :

```
[root]# service postgresql-9.4 initdb
[root]# service postgresql-9.4 start
```

Le paramètre `initdb` permet d'initialiser les fichiers du serveur PostgreSQL. Les détails de cette commande sont expliqués au chapitre suivant.

Cette même commande permet de redémarrer, recharger et arrêter le serveur PostgreSQL, avec les options respective `restart`, `reload` et `stop`.

Windows

L'installation sous Windows s'effectue grâce à un installeur graphique, projet fourni par EntrepriseDB et qui permet d'obtenir en quelques écrans un serveur PostgreSQL pleinement fonctionnel. Les binaires sont disponibles en téléchargement directement sur le site d'EnterpriseDB.

Cet installeur graphique permet aussi d'installer des contributions telles que PgBouncer, le pilote .Net Npsql et l'outil d'administration PgAdmin. EntrepriseDB est une des compagnies participant au développement de PostgreSQL.

1. Téléchargement

À partir de la page http://www.enterprisedb.com/products-services-training/pgdownload et selon la version de PostgreSQL souhaitée, il suffit de cliquer sur le système d'exploitation souhaité, par exemple`Win x86-64`, pour accéder au téléchargement de l'installeur. Le fichier à télécharger est un exécutable avec une extension .exe.

2. Installation

Une fois l'exécutable téléchargé, il suffit de double cliquer sur le fichier pour lancer l'installation, qui après avoir validé l'exécution (selon les autorisations du système) doit faire apparaître le premier écran. L'installeur est très classique pour un utilisateur régulier de Windows. Il suffit de cliquer sur **Next** pour avancer dans le processus d'installation.

La première option permet de sélectionner l'emplacement d'installation des programmes de PostgreSQL. Il n'y a pas de raison de modifier le choix par défaut :

L'étape suivante concerne le choix du répertoire des données. Par défaut, l'installeur propose de placer les données dans Program Files, ce qui n'est pas une bonne pratique. Il convient de choisir le répertoire et le disque dur ou sous-système disque qui accueillera les données, ayant suffisamment d'espace libre et correspondant aux besoins de performances requis.

Le mot de passe de l'utilisateur 'postgres' est important car c'est par ce moyen que l'administrateur peut se connecter à l'instance PostgreSQL, une fois démarrée, pour créer des objets comme des rôles ou des bases de données.

Le port par défaut de PostgreSQL est 5432 ; il peut être changé ultérieurement dans le fichier de configuration postgresql.conf.

Un clic sur **Finish** termine l'installation de PostgreSQL. Les fichiers sont alors créés dans le répertoire choisi et les données de l'instance sont créées.

L'outil Stack Builder peut être lancé pour installer des outils supplémentaires. L'option correspondante est cochée par défaut.

En fonction de l'instance de PostgreSQL choisie, une arborescence de logiciels complémentaires est proposée, parmi lesquels pgBouncer ou Npgsql.

Il suffit de cocher les cases des logiciels souhaités et Stack Builder lancera les installations demandées.

Une fois l'installation terminée, PostgreSQL est contrôlable comme n'importe quel service dans l'outil "Services" de Windows et PgAdmin est installé avec par défaut une connexion configurée pour se connecter à l'instance.

Installation silencieuse

L'outil d'installation pour Windows est aussi utilisable en mode silencieux (aucune interaction demandée à l'utilisateur). Tous les paramètres sont modifiables en passant des options à ce script d'installation. Pour cela, il suffit de placer celles-ci en ligne de commande ou dans un script et l'installation se fait alors de façon totalement transparente pour l'utilisateur.

L'installeur peut être appelé avec les options suivantes :

- `--help` : liste des options d'installation.

Administration SQL – Postgresql DB

- `--version` : version de l'installeur.
- `--optionfile <optionfile>` : fichier contenant les options d'installation.
- `--installer-language <installer-language>` : langue de l'installeur, parmi `en`, `es`et `fr`, par défaut `en`.
- `--extract-only <extract-only>` : extraction seule des fichiers. Désactivé (0) par défaut.
- `--disable-stackbuilder <disable-stackbuilder>` : désactive l'outil stackbuilder. Par défaut, l'outil est activé.
- `--mode <mode>` : mode d'installation, parmi `qt`, `win32` et `unattended`, par défaut `qt`.
- `--unattendedmodeui <unattendedmodeui>` : mode d'installation, parmi `none`,`minimal`, `minimalWithDialogs`, par défaut `minimal`.
- `--prefix <prefix>` : chemin d'installation des programmes. La valeur par défaut est :`C:\Program Files (x86)\PostgreSQL\9.4`.
- `--datadir <datadir>` : répertoire des données de l'instance PostgreSQL. La valeur par défaut est : `C:\Program Files (x86)\PostgreSQL\9.4\data`.
- `--superaccount <superaccount>` : nom du super-utilisateur de l'instance PostgreSQL. La valeur par défaut est `postgres`.
- `--superpassword <password>` : mot de passe du super-utilisateur de l'instance PostgreSQL.

Administration SQL – Postgresql DB

- `--serverport <serverport>` : port TCP de l'instance PostgreSQL. La valeur par défaut est 5432.
- `--locale <locale>` : paramètre de localisation de l'instance PostgreSQL. Par défaut, l'installeur utilise le paramètre du système d'exploitation.
- `--servicename <servicename>` : nom du service Windows de PostgreSQL.
- `--serviceaccount <serviceaccount>` : nom de l'utilisateur du système d'exploitation exécutant l'instance PostgreSQL. La valeur par défaut est `postgres`.
- `--servicepassword <servicepassword>` : mot de passe de l'utilisateur Windows exécutant l'instance PostgreSQL. La valeur par défaut est le mot de passe du super-utilisateur de l'instance PostgreSQL.
- `--install_runtimes <install_runtimes>` : installe ou non les dépendances à Microsoft Visual C++ avant l'installation de PostgreSQL. Par défaut, les bibliothèques sont installées.
- `--enable_acledit <enable_acledit>` : vérifie et donne les permissions en lecture sur le répertoire des données à l'utilisateur exécutant l'instance PostgreSQL. Désactivé par défaut.
- `--create_shortcuts <create_shortcuts>` : crée les entrées dans le menu Windows. Activé par défaut.
- `--debuglevel <debuglevel>` : verbosité de l'installeur entre 0 et 4, 2 par défaut.
- `--debugtrace <debugtrace>` : fichier de débogage.

Administration SQL – Postgresql DB

Exemple

L'exemple ci-dessous montre une installation silencieuse sans interface graphique, en modifiant le répertoire des données par défaut :

```
postgresql-9.4.0-windows-x64.exe --servicename postgresql94
--datadir C:\PostgreSQL\9.4\data --superpassword postgres
--mode unattended --unattendedmodeui none
```

Introduction

Une des étapes les plus importantes de l'installation de PostgreSQL consiste à créer un répertoire dans lequel le serveur écrira et lira les données des bases. Cette étape est un préalable nécessaire au démarrage de PostgreSQL.

Le choix du répertoire à utiliser n'est pas anodin ; toutes les données y seront stockées, sauf exception. Il est donc important de choisir un répertoire situé sur un sous-système disque rapide, si possible dédié à cette tâche afin d'optimiser les lectures et les écritures des données.

Décrire le choix du sous-système disque dépasse le cadre de cet ouvrage ; néanmoins, plusieurs types de ces sous-systèmes peuvent être listés en exemple :

- Contrôleur RAID avec disques SAS, configurés en RAID 10.
- Baie de stockage SAN, avec connexions Fibre Channel.
- SSD, carte FusionIO, ou tout simplement RAID logiciel.

Ce choix ne doit pas être négligé, l'acquisition de matériel spécifique est fréquente et doit être faite en fonction d'objectifs précis : performances attendues, durabilité, coût, capacité d'hébergement et d'administration... Cette liste n'est pas exhaustive.

Un deuxième aspect important à ce stade est le choix du système de fichiers utilisé. En fonction du système d'exploitation choisi, différents systèmes de fichiers peuvent être utilisés et le choix peut être déterminant en ce qui concerne les performances, la durabilité des données ou la maintenance. En ce qui concerne les systèmes d'exploitation GNU/Linux, parmi les nombreux

choix disponibles, Ext4 et XFS sont souvent préférés par les administrateurs de bases de données.

Le répertoire choisi doit être la propriété de l'utilisateur qui exécute le serveur PostgreSQL et qui a été créé lors de l'installation. Cet utilisateur est par défaut `postgres` même si n'importe quel nom peut être choisi.

Par exemple, si le répertoire où seront stockées les bases de données est `/var/lib/postgres/data`, les commandes qui suivent montrent sa création et la modification de ses attributs sous Linux :

```
[root]# mkdir -p /var/lib/postgres/data
[root]# chown -R postgres:postgres /var/lib/postgres
[root]# chmod -R 700 /var/lib/postgres
```

En plus de ce premier répertoire, d'autres peuvent être utilisés pour stocker les journaux binaires et d'autres espaces de tables. Ces deux notions seront expliquées dans les chapitres suivants.

Initialisation d'une instance

À chaque répertoire créé correspond une instance de PostgreSQL c'est-à-dire un serveur qui a donc son propre espace de stockage, qui accepte des connexions entrantes et dispose de ses propres fichiers de configuration.

Un répertoire peut aussi être appelé groupe de bases de données ou encore **cluster**. Le terme anglophone **cluster** peut parfois porter à confusion selon le contexte d'utilisation. Fréquemment utilisé dans le contexte des bases de données, il signifie simplement **groupe**. Il est ici utilisé dans le sens de groupe de base de données.

L'instance et le groupe de bases de données correspondent aux mêmes bases de données mais de deux points de vue différents : l'instance correspond à une exécution du serveur PostgreSQL et un groupe de base de données correspond à un conteneur de bases de données, sous la forme de répertoire et de fichiers.

L'initialisation d'une instance s'effectue par la commande `initdb`. En positionnant correctement les options et leurs valeurs, il est possible de personnaliser l'instance. Ces options et valeurs ont une influence non négligeable sur la façon dont se comporte le serveur, par exemple lors de la création de bases de données dans l'instance, lors de la création d'index ou lors de la recherche de données. Il convient donc de paramétrer correctement cette création.

1. Options de la commande

L'ensemble des options de la commande `initdb` correspond à la liste suivante :

Administration SQL – Postgresql DB

- `-A, --auth=METHOD` : méthode d'authentification par défaut.
- `-D, --pgdata=DATADIR` : chemin du répertoire des données.
- `-X, --xlogdir=XLOGDIR` : définit l'emplacement des journaux de transaction.
- `-k, --data-checksums` : active l'utilisation des sommes de contrôle des données.
- `-E, --encoding=ENCODING` : encodage par défaut des bases de données.
- `--locale=LOCALE` : paramètre régional par défaut des bases de données.
- `--lc-collate, --lc-ctype, --lc-messages, --lc-monetary, --lc-numeric, --lc-time` : détails des paramètres régionaux par défaut. Par défaut, les réglages du système d'exploitation sont utilisés.
- `--no-locale` : utilisation du paramètre régional C, équivalent à `--locale=C`.
- `-T, --text-search-config=CFG` : configuration par défaut de la recherche plein-texte.
- `-U, --username=NAME` : super-utilisateur de l'instance.
- `-W, --pwprompt` : demande interactivement le mot de passe du super-utilisateur.
- `--pwfile=FILE` : chemin du fichier contenant le mot de passe du super-utilisateur.
- `-L DIRECTORY` : indique l'emplacement des fichiers d'entrée.
- `-n, --noclean` : en cas d'erreur, ne pas supprimer le répertoire des données. Servira pour les débogages.
- `-N, --nosync` : ne pas demander d'écriture synchrone des données pendant la création.

- `-S`, `--sync-only` : ne demander une synchronisation que pour le répertoire des données.
- `-s`, `--show` : montre des réglages internes.

a. Options essentielles

Parmi la liste présentée, il est essentiel de comprendre et de choisir correctement quelques options. En particulier, certaines options ne peuvent pas être modifiées ultérieurement dans le répertoire de données ainsi créé.

L'option `-D` permet de choisir l'emplacement des données. Cet emplacement est important et tant que l'instance n'est pas démarrée, il est toujours possible de déplacer les données. Quoi qu'il en soit, c'est l'emplacement final des données de l'instance qui importe.

L'option `-A` permet de choisir la politique de sécurité par défaut, soit le contenu du fichier `pg_hba.conf` (voir chapitre Connexions - section Côté serveur (pg_hba.conf)). Cette politique par défaut peut permettre une initialisation des applications (bases de données et rôles) et être modifiée ultérieurement, une fois l'instance démarrée. Les valeurs possibles sont celles acceptées dans le fichier `pg_hba.conf`, par exemple : `trust`, `md5`, `ident`.

L'option `-X` permet de choisir l'emplacement des journaux de transactions (fichiers WAL). Cet emplacement est important, les transactions sont d'abord écrites dans ces fichiers puis

synchronisées sur le système de fichiers avant d'être réellement présentes dans les tables. Le choix d'un emplacement différent permet d'optimiser les performances pour ces fichiers particuliers. Cet emplacement ne peut pas être modifié une fois l'instance démarrée.

L'option `-k` permet d'activer le contrôle de la qualité des données écrites par le moyen de sommes de contrôles, calculées à l'écriture et vérifiées à la lecture. Ces sommes de contrôle ne sont pas activées par défaut et peuvent représenter un ralentissement des écritures et des lectures de données.

. Choix du jeu de caractères

Un paramètre supplémentaire concernant le choix du jeu de caractères peut être intégré à la commande `initdb`.

Un jeu de caractères détermine une correspondance entre un caractère et la façon dont il est stocké, sous forme de champ de bits. Avec PostgreSQL, chaque base de données a son propre jeu de caractères. Ce paramètre détermine la façon dont les caractères seront écrits sur le disque.

La création de l'instance crée une base de données modèle qui sert par la suite à la création des bases de données des applications. Le choix du jeu de caractères au moment de la création de l'instance correspond en fait au jeu de caractères de la base de données modèle `template1`.

La valeur de cette option peut être déterminée en fonction des variables d'environnement du système d'exploitation. Si cela n'est pas possible, alors la valeur par défaut est `SQL_ASCII`. Cette valeur ne correspond généralement pas aux besoins des applications qui utiliseront les bases de données. Les applications utilisent souvent des jeux de caractères comme `UTF-8` ou `LATIN1` qui permettent de stocker des caractères accentués.

Il est donc pertinent de choisir correctement le jeu de caractères initial, ce qui simplifie la création ultérieure des bases de données.

Le tableau suivant reprend les jeux de caractères que PostgreSQL connaît, limité à ceux généralement utilisés en Europe de l'Ouest :

Nom	Description	Langue	Octets/Caractère	Alias
SQL_ASCII	Non spécifié	Toutes les langues	1	
UTF8	Unicode, 8-bit	Toutes les langues	1-4	Unicode
LATIN1	ISO 8859-1, ECMA 94	Europe de l'Ouest	1	ISO88591
LATIN9	ISO 8859-15	LATIN1 avec l'Euro	1	ISO885915
WIN1252	Windows CP1252	Europe de l'Ouest	1	

Le jeu de caractères `SQL_ASCII` correspond à la table ASCII. Tous les caractères supérieurs à 127 insérés dans une table ne sont pas interprétés de telle sorte que n'importe quel caractère est accepté. Ce comportement empêche la conversion et la validation des caractères, ce qui rend les données stockées difficiles à exploiter.

Le jeu de caractères `UTF8` correspond à l'implémentation du standard Unicode qui est une tentative de rassembler en un seul jeu de caractères tous les alphabets connus. De ce fait, n'importe quel caractère est identifié de façon unique.

Les jeux de caractères `LATIN1`, `LATIN9` et `WIN1252` permettent de stocker des caractères correspondant aux alphabets des langues de l'Europe de l'Ouest. Leur portée n'est donc pas aussi étendue que l'Unicode.

Administration SQL – Postgresql DB

L'option de la commande `initdb` permettant de choisir le jeu de caractères par défaut est la suivante :

```
-E codage, --encoding=codage
```

où `codage` correspond au jeu de caractères sélectionné.

Dans les faits, il est préférable de régler correctement le système d'exploitation avec les paramètres de localisation souhaités.

c. Réglages des paramètres locaux

D'autres paramètres, appelés paramètres locaux, sont également disponibles.

Les paramètres locaux permettent à PostgreSQL de modifier son comportement sur les tris, les formatages de chiffres ou de dates et les messages. En effet, en fonction de la langue utilisée et de la région d'origine, les règles de tri ou le formatage d'une date peuvent être différents et PostgreSQL doit en tenir compte au moment de présenter les données ou pour générer un index.

PostgreSQL se base pour cela sur les paramètres locaux du système d'exploitation. Dans les systèmes d'exploitation Linux, la commande `locale -a` permet de connaître les paramètres locaux disponibles. Les variables d'environnement `LC_ALL` puis `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`,`LC_MONETARY`, `LC_NUMERIC`, `LC_TIME` et enfin `LANG` sont utiles à la commande `initdb` pour initialiser le groupe de bases de données.

Si aucune de ces variables d'environnement n'est disponible alors les paramètres locaux seront positionnés à `C`, qui correspond au standard ISO `C`.

L'utilisation de paramètres locaux spécifiques modifie le comportement de PostgreSQL, limitant par exemple l'utilisation d'index avec le mot-clé `LIKE` ou en ayant un impact négatif sur les performances.

Certains de ces paramètres ont une influence directe sur la façon dont PostgreSQL va écrire un index.

C'est donc `initdb` qui va initialiser les paramètres locaux avec les options suivantes :

```
--locale=locale
```

ou encore, séparément :

```
--lc-collate=locale, --lc-ctype=locale, --lc-messages=locale,
--lc-monetary=locale, --lc-numeric=locale --lc-time=locale
```

où `locale` correspond à un code de langue et de région. Par exemple, pour la langue française en France, le code est `fr_FR` et pour la langue anglaise aux États-Unis, le code est `en_US`.

2. Exécution de la commande

La commande `initdb` est habituellement exécutée avec le compte utilisateur propriétaire des fichiers. Dans l'exemple, il s'agit du compte `postgres`.

Administration SQL – Postgresql DB

Les commandes suivantes permettent de se connecter en tant que `postgres` au système, et de lancer la commande `initdb` :

```
[root]# su - postgres
[postgres]$ initdb -A trust -E UTF8 -locale=fr_FR -D /var/lib/postgresql/9.4/data
-X /pgxlog/9.4/wals
```

Dans cet exemple, la politique d'authentification est `trust` (pas de mot de passe), mais par défaut, l'instance n'autorise que des connexions locales. Le jeu de caractères est `UTF8`, le paramètre régional est`fr_FR`, le chemin des données est `/var/lib/postgresql/9.4/data` et les journaux de transactions sont dans le répertoire `/pgxlog/9.4/wals`.

La commande peut ne pas être dans les chemins par défaut (variable d'environnement `PATH`), c'est le cas lorsque PostgreSQL est installé par un système de paquetages comme dans les systèmes Debian ou Red Hat.

Par un système Debian, le chemin complet de la commande est`/usr/lib/postgresql/9.4/bin/initdb`. Dans un système Red Hat et apparenté, le chemin est`/usr/lib/pgsql-9.4/bin/initdb`. Voir les systèmes de paquetages au chapitre Installation.

Sous Windows, il est nécessaire de démarrer un interpréteur de commandes agissant en tant que `postgres`. Ceci peut s'effectuer en lançant la commande :

```
> runas /user:postgres cmd.exe
```

a. Contenu du répertoire créé

Cette commande crée une arborescence de répertoires et de fichiers. L'exemple suivant montre un extrait du contenu du répertoire `data`. Ce répertoire contient lui-même quelques fichiers et des répertoires.

Répertoires créés

- `base` : contient l'ensemble des bases de données et donc les tables et les indexes. Il s'agit de l'espace de tables par défaut.
- `global` : contient les tables globales, accessibles dans l'instance depuis toutes les bases de données. Il s'agit d'un espace de tables particulier.
- `pg_clog` : contient des fichiers utilisés pour connaître l'état des transactions en cours lorsque l'instance PostgreSQL est active.
- `pg_dynshmem` : contient des fichiers utilisés par le système d'allocation dynamique de mémoire.
- `pg_logical` : contient des données sur le décodage logique des données répliquées.
- `pg_multixact` : contient des données à propos des multitransactions.
- `pg_notify` : contient des données sur les notifications asynchrones (commandes`LISTEN/NOTIFY`).
- `pg_replslot` : contient les données des slots de réplication.

- `pg_serial` : contient des données sur les transactions sérialisables.
- `pg_snapshots` : contient les snapshots exportés.
- `pg_stat` : contient des fichiers permanents de statistiques.
- `pg_stat_tmp` : contient des fichiers temporaires de statistiques.
- `pg_subtrans` : contient l'état des sous-transactions.
- `pg_tblspc` : contient les liens symboliques vers les espaces de tables crées.
- `pg_twophase` : contient les fichiers de transactions préparées.
- `pg_xlog` : contient les journaux de transactions. Lorsque l'option `-X` de `initdb` a désigné un autre répertoire, il ne s'agit pas d'un répertoire mais d'un lien symbolique.

Le répertoire contenant le plus de données sera toujours le répertoire `base`. Les autres répertoires ne contiennent que des données temporaires ou d'une faible volumétrie.

Fichiers créés

- `PG_VERSION` : fichier texte contenant la version majeure de PostgreSQL, par exemple `9.4`.
- `postgresql.auto.conf` : fichier de configuration dans lequel la commande SQL `ALTER SYSTEM` écrit. Ce fichier est toujours lu en dernier.
- `postgresql.conf` : fichier de configuration.
- `pg_hba.conf` : fichier d'authentification.

- `pg_ident.conf` : fichier de correspondance des utilisateurs.

Tous les éléments nécessaires au démarrage du serveur sont maintenant présents.

3. Création d'instances supplémentaires

Il est possible, dans un même système, de créer plusieurs instances de PostgreSQL, en utilisant les mêmes programmes et en créant simplement un autre répertoire de groupe de base de données avec la commande `initdb`. Ces instances distinctes ont leurs propres fichiers de configuration, des bases de données différentes et acceptent des connexions différentes.

Mis à part le répertoire des données, une autre ressource est importante à distinguer : le port TCP. Les connexions au serveur passeront par une connexion TCP sur un port précis. Par défaut, le port TCP utilisé par PostgreSQL est `5432`. Afin de faire fonctionner plusieurs instances de PostgreSQL simultanément, il faut signifier à PostgreSQL de réceptionner les connexions entrantes sur un autre port que le port par défaut.

Il est possible de modifier ce port par défaut en modifiant une option passée au programme `postmaster` ; mais la méthode la plus simple et la plus sûre consiste à modifier le fichier de configuration `postgresql.conf`, présent dans le répertoire du groupe de bases de données et propre à chaque instance.

Ce fichier `postgresql.conf` est un fichier texte qui est modifiable avec n'importe quel éditeur de texte. Le paramètre à modifier est `port`. Il suffit ici de corriger la valeur par défaut et éventuellement de retirer le caractère # présent en début de ligne.

La nouvelle valeur à indiquer doit être un numéro de port TCP qui n'est pas déjà utilisé dans le système. Il est possible de vérifier les ports utilisés avec la commande `netstat` avec les systèmes Linux ou les systèmes Windows.

Pour les systèmes Linux :

```
[root]# netstat -ltn
```

Pour les systèmes Windows :

```
> netstat -a
```

Il suffit de choisir un port inutilisé, en incrémentant simplement la valeur par défaut :

```
port = 5433
```

Une fois le fichier enregistré, l'instance est prête à démarrer et peut accueillir des bases de données distinctes d'une autre instance.

Arrêt et démarrage du serveur

Comme l'indiquent les deux derniers messages du résultat de la commande `initdb`, il y a plusieurs méthodes pour démarrer le serveur PostgreSQL :

```
[postgres]$ postgres -D data
```

ou :

```
[postgres]$ pg_ctl -D data -l journaltrace start
```

La commande `postgres` correspond au programme du serveur. C'est ce programme qui, une fois chargé, acceptera les connexions entrantes. Il est possible de démarrer le serveur ainsi mais ce n'est pas la meilleure méthode.

La commande `pg_ctl` est dédiée au contrôle du serveur PostgreSQL et donc au démarrage du serveur `postgres`. Il est donc possible de démarrer PostgreSQL comme il est indiqué, puis d'arrêter, de redémarrer ou de recharger le serveur. L'option `-D` permet de préciser le répertoire contenant les données et les fichiers de configuration. L'option `-l` permet de rediriger les messages vers un fichier de traces et l'option `start` permet de préciser l'action à effectuer.

Les options `start`, `stop`, `restart` et `reload` permettent respectivement de démarrer, d'arrêter, de redémarrer et de recharger le serveur.

La commande `pg_ctl` est utilisée par le script de démarrage du système de démarrage des systèmes GNU/Linux pour arrêter et démarrer le serveur PostgreSQL. Elle permet aussi d'enregistrer PostgreSQL en tant que service dans les systèmes Windows.

Deux options permettent d'enregistrer et de retirer PostgreSQL du panneau de service des systèmes Windows. Cet enregistrement est effectué implicitement lors de l'installation de PostgreSQL dans un système Windows mais il peut être utile lors de la création d'une autre instance.

L'option `register` dispose d'options spécifiques qui permettent de préciser le nom du service tel qu'il apparaît dans le panneau de service (`-N`), du nom (`-U`) et du mot de passe (`-P`) de l'utilisateur du système et enfin de l'emplacement du groupe de base de données (`-D`). Par exemple, la commande suivante permet d'enregistrer une nouvelle instance de PostgreSQL. Cette commande doit être exécutée en tant que compte privilégié sur le système :

```
> pg_ctl register -N pgsql9.4-2 -U postgres -P postgres -D
c:\PostgreSQL\data2\
```

Afin de retirer une instance, l'option `unregister` peut être utilisée :

```
> pg_ctl unregister -N pgsql8.1-2
```

Cette commande désactive l'instance dans le panneau de service. Cette instance apparaît toujours mais est considérée comme désactivée. Elle ne disparaîtra qu'au prochain redémarrage du système.

Sessions

Une session, ouverte à partir de la connexion d'un logiciel client, ne concerne qu'une seule base de données dans le serveur et qu'un seul compte utilisateur. Ainsi, lorsqu'une application souhaite utiliser plusieurs bases de données, elle doit ouvrir d'autres connexions. De même, si une application souhaite ouvrir une session avec un autre compte utilisateur, une autre connexion doit être ouverte.

Une connexion entre un logiciel client et le serveur PostgreSQL nécessite l'existence de deux objets du côté du serveur : un compte utilisateur et une base de données. La première connexion peut s'effectuer avec un compte utilisateur et une base de données créée au moment de l'initialisation du groupe de bases de données.

1. Côté serveur (pg_hba.conf)

L'ouverture d'une connexion est contrôlée par PostgreSQL selon les règles d'un fichier de configuration :`pg_hba.conf`. Ce fichier, présent dans le répertoire du groupe de bases de

données, contient des règles définissant les autorisations et restrictions d'accès, en fonction du nom d'utilisateur, de la base de données, de l'origine et de la méthode utilisée.

Chaque ligne du fichier correspond à une règle. Lors d'une tentative de connexion, PostgreSQL parcourt les règles et les compare avec les paramètres de la connexion afin de déterminer l'ouverture ou non de la connexion.

Une règle est composée de quatre à cinq éléments :

- Le type de la connexion.
- Le nom de la base de données.
- Le nom d'utilisateur.
- Les paramètres réseau.
- La méthode d'authentification.

Le type de la connexion peut être :

- `local` : ce type de connexion identifie les connexions depuis un socket UNIX.
- `host` : ce type de connexion identifie les connexions depuis un réseau TCP/IP. Les connexions peuvent utiliser ou non le chiffrement SSL.
- `hostssl` ou `hostnossl` : de même, ces deux types de connexion identifient les connexions TCP/IP mais rendent explicite l'utilisation ou non du chiffrement SSL de la connexion.

Si un type de connexion est absent du fichier, la connexion est alors impossible. Par exemple, si le type de connexion absent est `local`, aucune connexion ne peut s'effectuer en utilisant un socket Unix.

Le nom de la base de données indique la base concernée par la règle. Plusieurs noms peuvent être indiqués, séparés par des virgules et les mots-clés `all`, `sameuser` et `samerole` peuvent être utilisés :

- Le mot-clé `all` identifie n'importe quelle base de données.
- Le mot-clé `sameuser` signifie que la base de données a le même nom que l'utilisateur de la connexion.
- Le mot-clé `samerole` signifie que l'utilisateur ouvrant la connexion doit être membre du rôle portant le même nom que celui de la base de données précisé dans la connexion.

Le nom d'utilisateur identifie le compte utilisé pour la connexion. Le mot-clé `all` identifie tous les comptes utilisateurs et lorsque le nom est directement suivi du caractère +, alors ce champ identifie tous les comptes membres du rôle indiqué.

Les paramètres réseau permettent d'identifier la provenance de la connexion. La notation peut prendre plusieurs formes :

- La notation CIDR, avec l'adresse de l'hôte ou du réseau, le caractère / puis le masque CIDR, par exemple : 192.168.1.0/24.
- La notation classique, avec le nom de l'hôte ou du réseau, un caractère [Espace] puis le masque, par exemple : 192.168.1.0.255.255.255.0.

Ces notations concernent seulement les types de connexions `host`, `hostssl` et `hostnossl`.

Enfin, la méthode d'authentification détermine la façon dont la connexion peut s'ouvrir :

- `trust` : connexion sans condition. Cette méthode permet d'ouvrir une connexion sans fournir de mot de passe. Elle doit être utilisée avec précaution.
- `md5` : cette méthode demande un mot de passe chiffré pour ouvrir une connexion. Cette méthode doit être préférée. Elle remplace les méthodes `password` et `crypt`, qui correspondent à d'anciennes versions de PostgreSQL.
- `reject` : cette méthode rejette toute tentative de connexion correspondant à la règle.

Les exemples qui suivent illustrent la façon d'écrire ce fichier de configuration :

```
local all postgres trust
host all postgres 192.168.0.0/24 md5
host clients clients 192.168.1.2/32 trust
host clients clients 0.0.0.0/0 md5
```

La première ligne autorise l'utilisateur `postgres` à se connecter sans donner de mot de passe à toutes les bases de données, mais seulement depuis un socket Unix.

La seconde ligne autorise l'utilisateur `postgres` à se connecter depuis n'importe quel hôte du réseau local `192.168.0.0/24`, en fournissant le mot de passe.

La troisième ligne autorise l'utilisateur `clients` à se connecter à la base de données clients sans fournir de mot de passe, mais seulement depuis l'hôte `192.168.1.2`.

Enfin, la quatrième ligne autorise l'utilisateur clients à se connecter depuis n'importe quel hôte à la base de données `clients`, en fournissant le mot de passe.

L'ordre des lignes est important. En effet, si la troisième et la quatrième ligne sont inversées, alors à aucun moment l'utilisateur `clients` ne pourra se connecter depuis l'hôte 192.168.1.2 sans fournir de mot de passe, puisque la correspondance sera faite avec la ligne précédente, indiquant une connexion depuis n'importe quel hôte.

L’écriture de ce fichier doit se faire avec méthode, en écrivant simplement les règles nécessaires et en respectant les règles de lecture de PostgreSQL : les règles les plus précises doivent être écrites en premier, suivies des règles génériques.

Outils

Plusieurs outils destinés aux administrateurs de bases de données sont fournis, certains avec le serveur, d'autres comme des projets externes à PostgreSQL. Dans tous les cas, ces outils utilisent le même protocole clients-serveurs utilisé pour les applications.

1. L'outil en ligne de commandes : psql

L'outil `psql` est fourni avec les outils du serveur PostgreSQL. Il est donc toujours disponible, dès que le serveur est correctement installé. Il s'agit d'un outil en ligne de commandes, fonctionnant en mode interactif, c'est-à-dire en permettant une interaction avec un utilisateur, ou en mode non interactif, par exemple en interprétant le contenu d'un fichier.

`psql` a deux fonctions :

- Il permet d'envoyer des instructions au serveur, par exemple des ordres SQL ou des commandes d'administration (création de tables, d'utilisateurs...).
- Il permet d'interpréter des instructions spéciales, complétant le jeu d'instructions SQL du serveur.

`psql` est un outil client, c'est-à-dire qu'il se connecte au serveur comme n'importe quel client, en fournissant donc un nom d'utilisateur, un mot de passe si nécessaire et le nom d'une base de données. À partir de ces données et de l'hôte depuis lequel `psql` est exécuté, le serveur accepte ou non la connexion.

Administration SQL – Postgresql DB

a. Options de connexion

L'exemple suivant ouvre une connexion avec le compte utilisateur `postgres`, sur la base de données`postgres`, sans fournir de mot de passe et sans préciser l'adresse du serveur. Implicitement, `psql`ouvrira une connexion en passant par le socket Unix, donc sur le même hôte :

```
[postgres]$ psql -U postgres postgres
postgres=#
```

L'option `–U` permet de préciser le nom d'utilisateur existant dans la base de données. Sans cette option, c'est le nom d'utilisateur du système d'exploitation lançant `psql` qui est utilisé, comme dans l'exemple suivant :

```
[postgres]$ psql postgres
postgres=#
```

Le dernier paramètre est le nom de la base de données. Lorsque ce paramètre n'est pas fourni, `psql`utilise le nom d'utilisateur comme nom de la base de données. L'exemple qui suit ouvre donc une connexion en tant qu'utilisateur `postgres`, vers la base de données `postgres`, en passant par le socket Unix :

```
[postgres]$ psql
```

```
postgres=#
```

D'autres options peuvent être utilisées pour ouvrir une connexion à un serveur ; par exemple le nom d'hôte ou le nom d'utilisateur :

- `-h nomhôte`, `--host nomhôte` : indique le nom d'hôte ou l'adresse IP du serveur. Cette option permet donc de se connecter à un serveur distant, c'est-à-dire différent de l'hôte sur lequel est exécuté `psql`.
- `-p port`, `--port port` : indique le port TCP sur lequel ouvrir la connexion. Ce port est donc le port utilisé par le serveur. Le port par défaut est 5432, mais une autre valeur peut être utilisée, selon la configuration du serveur.
- `-U nomutilisateur`, `--username nomutilisateur` : indique le nom d'utilisateur utilisé pour ouvrir la connexion.
- `-W`, `--password` : indique à `psql` de demander, de façon interactive, un mot de passe à l'utilisateur. Il n'est pas possible, avec `psql`, de donner un mot de passe directement dans la liste des options : un mot de passe est soit saisi par l'utilisateur de façon interactive, soit lu depuis un fichier spécifique.
- `-d <database>` : indique la base de données à laquelle se connecter.
- `-l`, `--list` : liste les bases de données existantes dans le serveur.
- `-e` : affiche toutes les requêtes SQL envoyées au serveur.
- `-f nomfichier`, `--file nomfichier` : interprète les instructions du fichier`nomfichier` puis quitte le programme.

- `-c <requete>` : exécute une requête SQL.
- `-L <fichier>` : écrit le résultat des requêtes dans un fichier, en plus de la sortie standard.
- `-o` : redirige le résultat des requêtes dans un fichier, comme la commande interne `\o`.
- `-s` : exécute les requêtes une par une, en demandant à l'utilisateur de confirmer l'exécution de chaque requête, lors de l'utilisation en mode non interactif.
- `-A` : désactive l'alignement des données en sortie. Équivaut à la commande interne `\a`.
- `-t` : désactive la production des en-têtes dans le résultat des données. Équivaut à la commande`\t`.
- `-X`, `--no-psqlrc` : ne lit pas le fichier `.psqlrc`.

b. Variables d'environnement

Certains de ces paramètres peuvent être précisés par l'intermédiaire de variables d'environnement. Il suffit d'enregistrer ces variables dans l'interpréteur de commandes du système où est lancé `psql` pour qu'elles soient interprétées :

- `PGDATABASE` : nom de la base de données sur laquelle se connecter.
- `PGHOST` et `PGHOSTADDR` : nom et adresse IP du serveur. PGHOST peut commencer par une barre oblique et dans ce cas il est interprété comme étant le répertoire du socket Unix.
- `PGPORT` : numéro de port TCP du serveur.
- `PGUSER` : nom d'utilisateur.

- `PGPASSWORD` : mot de passe de l'utilisateur.

c. Fichier de mots de passe

De plus, il est possible d'enregistrer les mots de passe dans un fichier dédié, afin d'éviter la saisie systématique. Ce fichier se situe dans le répertoire personnel de l'utilisateur du système d'exploitation qui lance `psql`.

Avec un système Unix, le fichier se nomme `.pgpass` et se situe dans le répertoire de l'utilisateur, accessible via le raccourci `~`, soit `~/.pgpass`.

Avec un système Windows, le fichier se nomme `pgpass.conf` et se trouve dans le répertoire `%APPDATA%\postgresql`.

Ce fichier contient autant de lignes qu'il y a de combinaisons de connexions. Le modèle d'une ligne est le suivant :

```
nomhote:port:database:nomutilisateur:motdepasse
```

Il est possible d'écrire les quatre premiers paramètres avec le caractère générique *. Il suffit d'écrire les lignes les plus spécifiques en premier, suivies par les lignes les plus génériques afin d'éviter toute ambiguïté d'interprétation du contenu. Par exemple, la ligne suivante indique le mot de passe (`passpg`) de l'utilisateur `postgres` pour n'importe quel serveur PostgreSQL :

```
*:*:*:postgres:passpg
```

Si, pour un serveur particulier, le mot de passe est différent, alors la ligne suivante doit être positionnée avant :

```
192.168.0.2:5432:*:postgres:pgpass
```

Pour des raisons de sécurité, ce fichier doit avoir des accès en lecture et en écriture restreints au propriétaire. Cela peut être effectué par la commande suivante, sous Unix :

```
chmod 600 ~/.pgpass
```

Sans ces restrictions, le fichier ne sera pas utilisé par `psql`.

d. Utilisation en mode interactif

Une fois connecté en mode interactif, l'utilisateur obtient un interpréteur permettant de saisir des requêtes et de lire les résultats. Les requêtes se divisent en deux familles de commandes :

- Les commandes SQL, qui sont interprétées par le serveur et utilisables depuis n'importe quel client.
- Les commandes spéciales, qui sont interprétées par `psql`.

Les requêtes SQL sont par exemple des requêtes `SELECT` ou `UPDATE`, ou des ordres `CREATE` ou`DROP`. Elles n'ont rien de spécifique à `psql`.

Administration SQL – Postgresql DB

Les commandes spéciales sont des raccourcis commençant par la barre oblique inversée (\) et permettent d'améliorer l'interactivité de `psql`. Par exemple, la commande spéciale suivante affiche l'ensemble des espaces de noms disponibles dans la base de données courante :

```
postgres=# \dn
         Liste des schémas
        Nom          | Propriétaire
———————————————————————+——————————————
 information_schema | postgres
 pg_catalog         | postgres
 pg_toast           | postgres
 public             | postgres
(4 lignes)
```

Administration SQL – Postgresql DB

Les lignes suivantes indiquent quelques commandes utiles :

Le caractère +, lorsqu'il est spécifié, permet d'afficher les droits associés à l'objet ou des descriptions supplémentaires. Le terme modèle correspond à un nom d'objet ; il est possible d'y utiliser des caractères génériques (*, ?).

- `\c [nomdb]` : ouvre une connexion à une autre base de données.
- `\l[+]` : liste les bases de données du serveur.
- `\du [modèle]`, `\dg [modèle]` : liste tous les rôles et groupes.
- `\h [commande]` : affiche l'aide sur la commande SQL.
- `\?` : affiche l'aide sur les commandes spéciales.
- `\d[+] [modèle]` : affiche le détail des relations décrites par le modèle.
- `\db[+] [modèle]` : liste tous les espaces de tables.
- `\dd [modèle]` : affiche les descriptions des objets. Le modèle permet de filtrer sur le nom des objets.
- `\dp [modèle]` : liste les privilèges.
- `\df[antw][+] [modèle]` : liste les fonctions, avec leurs arguments et les types de retour. Les caractères complémentaires peuvent limités respectivement aux agrégats, fonctions normales, fonctions déclencheurs et « window » fonctions.

- `\d[istvmS] [modèle]` : liste des objets, selon le caractère utilisé :
  - `i` : index
  - `s` : séquence
  - `t` : table
  - `v` : vue
  - `m` : vue matérialisée
  - `S` : objet système

Il est possible d'utiliser plusieurs caractères et le caractère `S` permet de n'afficher que les objets système ; sans ce caractère, seuls les objets non-système sont affichés.

- `\de[tsuw][+] [PATTERN]` : liste respectivement les tables étrangères, les serveurs, les correspondances utilisateurs et les « wrappers ».
- `\dn[+] [modèle]` : liste tous les schémas ou espaces de noms disponibles.
- `\dp [modèle]` : liste les privilèges des tables, vues et séquence indiquées par le modèle.
- `\timing` : affiche le temps d'exécution des requêtes. Cette commande active ou désactive l'affichage, selon l'état courant.
- `\watch [N]` : exécute la requête toutes les N secondes.
- `\g [FILE]` ou `\gset [PREFIX]` ou `;` : exécute la requête SQL et, dans le cas de `\g` et`\gset`, écrit le résultat dans un fichier ou une variable.

Administration SQL – Postgresql DB

- `\echo [STRING]` : écrit la chaîne sur la sortie standard.
- `\i fichier` : exécute les commandes depuis un fichier.
- `\ir fichier` : comme `\i`, mais relativement au script courant.
- `\o [FILE]` : écrit l'ensemble des résultats des requêtes dans le fichier.
- `\qecho [STRING]` : écrit la chaîne dans le flux sortie, utilisable avec `\o`.
- `\e [fichier] [ligne]` : édite la dernière requête ou le fichier indiqué, avec un éditeur externe.
- `\ef [fonction [ligne]]` : édite la fonction indiquée dans un éditeur externe.
- `\p` : montre la dernière requête.
- `\r` : vide la zone mémoire contenant la dernière requête.
- `\s [fichier]` : affiche l'historique des requêtes, ou, lorsqu'un fichier est indiqué, enregistre l'historique dans ce fichier.
- `\w fichier` : enregistre la zone mémoire contenant la dernière requête dans un fichier.
- `\copy ...` : exécute la commande `copy` depuis et vers le client, et non un fichier coté serveur.
- `\a` : bascule l'alignement d'affichage, actif par défaut. Équivalent au paramètre `-A` de la commande.
- `\t` : bascule l'affichage des en-têtes (noms des attributs). Équivalent au paramètre `-t` de la commande.
- `\f [chaine]` : montre ou définit le séparateur de champs pour le mode non aligné.

Administration SQL – Postgresql DB

- `\H` : bascule le mode d'affichage HTML, désactivé par défaut.
- `\T [chaine]` : définit les attributs du marqueur `<table>` dans l'affichage HTML.
- `\pset [{format | border | expanded | fieldsep | fieldsep_zero | footer | null | numericlocale | recordsep | recordsep_zero | tuples_only | title | tableattr | pager} [on | off]]` : définit les options de formatages indiquées.
- `\x [on|off|auto]` : bascule l'affichage étendu.
- `\q` : quitte `psql`.

e. Utilisation en mode non interactif

L'outil `psql` peut aussi être utilisé en mode non interactif ; c'est-à-dire que les instructions à interpréter sont lues depuis un fichier ou un flux, sans intervention de l'utilisateur. Ce mode est utile lors de l'utilisation de scripts d'automatisation de tâches, fonctionnant sans l'intervention de l'utilisateur.

Une première option utile est `-f`. Lorsqu'elle est passée en argument de `psql`, les instructions seront lues depuis un fichier :

```
$ psql -f fichier.sql
```

L'équivalent de cette commande pourrait être :

```
$ psql < fichier.sql
```

Les instructions sont interprétées de la même façon.

L'option `-c` permet de n'exécuter qu'une commande SQL, sans devoir la lire depuis un fichier ou l'entrée standard, tout en utilisant les arguments de contrôle de sortie, par exemple :

```
$ psql -d clients -At  -c 'select count(*) from clients'
42
```

Administration SQL – Postgresql DB

f. Fichier de configuration

L'outil `psql` peut être configuré au moyen du fichier `.psqlrc`, situé à la racine du répertoire personnel de l'utilisateur système et contenant différentes instructions. Le principe est le même que le fichier `.bashrc` du shell `bash`. Les réglages permettent de personnaliser le comportement interactif de `psql`. Par exemple, il est possible de modifier l'invite de commande en réglant les paramètres `PROMPT1` et `PROMPT2`, modifier l'historique des commandes SQL ou définir des commandes SQL dans une variable.

L'invite de commande peut se régler avec l'invite par défaut `PROMPT1` ou `PROMPT2` qui est l'invite obtenue lors de la saisie d'une commande SQL sur plusieurs lignes, avant la fin de commande (généralement un point-virgule).

Par exemple, la définition suivante de `PROMPT1` permet d'afficher dans l'ordre : le nom d'utilisateur (`%n`), le nom d'hôte (`%m`), le port TCP (`%>`), et la base de données courante (`%~`) :

```
\set PROMPT1  '%n@%m:%>/%~%x%# '
```

Le symbole `%x` ajoute un astérisque au prompt lorsqu'une transaction est ouverte avec la commande `BEGIN`. Le symbole `%#` affiche le caractère # lorsque l'utilisateur de la session est un super-utilisateur et le caractère > dans les autres cas.

Il est possible de modifier le comportement de l'historique en changeant sa taille, en ignorant les commandes commençant par un espace ou les commandes dupliquées (ou les deux) :

```
HISTSIZE=2000
HISTCONTROL= {ignoredups | ignorespace | ignoreboth}
```

Il est aussi possible de définir des commandes SQL, afin de bénéficier de la variable pour appeler rapidement la commande en mode interactif, par exemple :

```
\set locks 'select n.nspname, c.relname, l.pid, l.mode, l.granted
from pg_locks l join pg_class c on l.relation=c.oid join
pg_namespace n on c.relnamespace=n.oid ;'
```

Il suffit ensuite d'appeler, à l'invite de commande, la variable `:locks`.

2. L'outil graphique : pgAdmin III

L'outil pgAdmin III est un outil graphique basé sur le jeu de composants graphiques WxWidgets, permettant de se connecter à PostgreSQL. Il s'agit d'un projet séparé du serveur PostgreSQL, mais il est proposé lors de l'installation du serveur sur les systèmes Windows. Il s'agit d'un logiciel libre, dont les fichiers sources et les programmes binaires pour plusieurs systèmes d'exploitation, sont disponibles à partir du site web http://www.pgadmin.org/. Les systèmes de paquetages pour Red Hat et Debian le proposent dans une version récente.

La dernière version en date est la version 1.18.1.

L'interface de pgAdmin III se présente comme ci-dessous :

L'écran se divise en trois parties :

- Une arborescence à gauche, représentant les différents serveurs et permettant de naviguer dans tous les objets d'un serveur.
- Une fenêtre de propriétés en haut à droite, permettant de visualiser les différentes propriétés des objets sélectionnés depuis l'arbre de gauche.
- Une fenêtre en bas à droite, où pgAdminIII affiche les requêtes SQL qui permettent de créer l'objet sélectionné depuis l'arbre de gauche.

Pour chaque objet, un menu contextuel est disponible, proposant les actions que pgAdmin III permet, comme la suppression ou la création d'un nouvel objet, la modification des propriétés...

L'écran suivant montre l'interface, après avoir sélectionné un objet dans l'arbre de gauche.

Les propriétés sont affichées à droite, comme par exemple, les droits d'accès. La requête SQL est affichée dans le cadre en dessous, ce qui permet de comprendre pour chaque objet la façon dont il a été créé. Par défaut, seuls les objets non système apparaissent dans les différentes parties de l'interface.

L’écran suivant montre l’outil de création de requêtes SQL. Il est possible de l’activer depuis la barre d’outils, en cliquant sur l’icône ou depuis le menu **Tools** de la barre de menus, en sélectionnant la commande **Query Tools**.

Le résultat est affiché dans la partie inférieure de la fenêtre, comme dans l'exemple qui suit :

Il est alors possible de saisir n'importe quelle requête SQL afin qu'elle soit envoyée au serveur.

Il est aussi possible de créer des requêtes avec un assistant de création graphique, comme dans l'exemple suivant :

Il est possible d'ajouter autant de serveurs que souhaité. Les serveurs apparaissent dans l'arborescence de gauche. Pour ajouter un nouveau serveur, il suffit de cliquer sur le bouton situé à gauche de la barre d'outils. Une fois le nouveau serveur enregistré, il apparaît dans la liste des serveurs.

3. L'outil en ligne : phpPgAdmin

L'outil phpPgAdmin est un ensemble de scripts, écrits en langage PHP, permettant d'administrer un ou plusieurs serveurs PostgreSQL. Le fonctionnement de cet outil est semblable à un site web, c'est-à-dire qu'il nécessite un serveur web, comme Apache, et le moteur d'interprétation PHP.

Une fois installé et configuré dans le serveur, il ne nécessite pas d'autres manipulations pour être utilisé par autant de personnes que souhaité. Le seul outil dont un utilisateur se servira est un navigateur web. N'importe quel navigateur web graphique peut convenir à l'utilisation de phpPgAdmin.

a. Installation

Les scripts PHP sont fournis sous forme de paquetage pour les systèmes Debian, il suffit donc d'installer les paquets :

```
apt-get install phppgadmin
```

Cette commande installe automatiquement les dépendances utiles, comme Apache ou l'interpréteur PHP.

Pour les systèmes où il n'existe pas de paquetage prêt à l'emploi, il est nécessaire de procéder à l'installation d'un serveur web et de l'interpréteur PHP.

b. Configuration

La configuration de phpPgAdmin est centralisée dans un fichier texte, situé dans le répertoire conf de l'installation, ou, sur un système Debian, au chemin suivant :`/usr/share/phppgadmin/conf/config.inc.php`.

Le fichier se terminant par `dist` est le fichier modèle contenant une configuration minimale.

Le fichier `config.inc.php` contient toutes les instructions de configuration des scripts.

Un des points les plus importants de ce fichier de configuration est la déclaration des serveurs utilisables. En effet, depuis le navigateur web, l'utilisateur n'a pas la possibilité de se connecter au serveur PostgreSQL de son choix. C'est à l'administrateur de détailler les serveurs utilisables dans le fichier config.inc.php.

Administration SQL – Postgresql DB

La déclaration des serveurs peut s'effectuer simplement en éditant le script config.inc.php avec un éditeur de texte. Initialement, un seul serveur est déclaré, correspondant à une connexion locale, comme dans l'exemple qui suit :

```
$conf['servers'][0]['desc'] = 'PostgreSQL 9.4';
$conf['servers'][0]['host'] = 'serveurpg94';
$conf['servers'][0]['port'] = 5432;
$conf['servers'][0]['defaultdb'] = 'template1';
$conf['servers'][0]['pg_dump_path'] = '/usr/bin/pg_dump';
$conf['servers'][0]['pg_dumpall_path'] = '/usr/bin/pg_dumpall';
$conf['servers'][0]['slony_support'] = false;
$conf['servers'][0]['slony_sql'] = '/usr/share/pgsql';
```

Il suffit d'insérer les mêmes lignes, en incrémentant simplement l'indice de l'exemple [0] pour chaque nouvelle déclaration. L'exemple qui suit déclare un nouveau serveur, avec les propriétés de connexions :

```
$conf['servers'][1]['desc'] = 'Pg Slave 1';
$conf['servers'][1]['host'] = '192.168.1.12';
$conf['servers'][1]['port'] = 5432;
$conf['servers'][1]['defaultdb'] = 'template1';
$conf['servers'][1]['pg_dump_path'] = '/usr/bin/pg_dump';
```

```
$conf['servers'][1]['pg_dumpall_path'] = '/usr/bin/pg_dumpall';
$conf['servers'][1]['slony_support'] = false;
$conf['servers'][1]['slony_sql'] = '/usr/share/pgsql';
```

Quelques paramètres supplémentaires permettent de personnaliser le comportement des scripts de phpPgAdmin. Ces paramètres sont valables pour tous les serveurs déclarés dans la configuration.

L'affichage des objets système est désactivé par défaut. Il suffit de positionner à « vrai » la variable pour l'activer :

```
$conf['show_system'] = true ;
```

L'autorisation de la connexion des comptes privilégiés, par exemple le compte `postgres`, est désactivée par défaut. En positionnant ce paramètre à « faux », les connexions seront autorisées :

```
$conf['extra_login_security'] = false ;
```

L'affichage des seuls objets dont l'utilisateur est propriétaire est désactivé par défaut. Il suffit de positionner à « vrai » la variable pour activer ce comportement :

```
$conf['owned_only'] = false;
```

L'affichage des commentaires associés aux objets est activé par défaut :

Administration SQL – Postgresql DB

```
$conf['show_comments'] = true;
```

Les seuls paramètres qui ne sont pas indiqués dans le fichier configuration sont les noms d'utilisateurs et les mots de passe.

L'utilisateur les indique dans l'interface web, après avoir sélectionné le serveur souhaité dans l'écran d'introduction de l'outil.

Installation postgresql server windows !

Administration SQL – Postgresql DB