DALIBO

L'Expertise PostgreSQL

Présentation de PostgreSQL

Formateur : Jean-Paul Argudo
Contact : jean-paul.argudo@dalibo.com
Date : mars 2010

Table des matières

Introduction

- Partie 1 : Origines
- Partie 2 : Versions
- Partie 3 : Fonctionnalités
- Partie 4 : La communauté
- Partie 5 : Satellites

Licence Creative Commmons CC-BY-NC-SA

Partie 1 : Un peu d’histoire...

- Les origines
- La philosophie
- Les « pères fondateurs »
- Les sponsors

Principes fondateurs

- Sécurité des données (`ACID`)
- Respect des normes (`ANSI SQL`)
- Fonctionnalités
- Performances
- Simplicité du code

Depuis son origine, PostgreSQL a toujours privilégié la stabilité et le respect des standards plutôt que les performances.

Ceci explique en partie la réputation de relative lenteur et de complexité face aux autres SGBD du marché. Cette image est désormais totalement obsolète, notamment grâce aux avancées réalisées depuis les versions 8.x.

Origines

Années 1970 : `Ingres` est développé à Berkeley

1985 : `Ingres` est re-développé à partir de rien. Le nouveau projet est nommé `Postgres`.

1995 : Ajout du langage `SQL`. `Postgres` est renommé en `Postgres95`.

1996 : `Postgres95` devient PostgreSQL

1996 : Création du *PostgreSQL Global Development Group*

L'histoire de PostgreSQL remonte à la base de données `Ingres`, développée à Berkeley par Michael Stonebraker. Lorsque ce dernier décida en 1985 de recommencer le développement de zéro, il nomma le logiciel `Postgres`, comme raccourci de post-Ingres. Lors de l'ajout des fonctionnalités `SQL` en 1995 par deux étudiants chinois de Berkeley, `Postgres` fut renommé `Postgres95`. Ce nom fut changé à la fin de 1996 en PostgreSQL.

De longs débats enflammés animent toujours la communauté pour savoir s'il faut revenir au nom initial `Postgres`.

À l'heure actuelle, le nom `Postgres` est accepté comme un alias du nom officiel `PostgreSQL`.

Plus d'informations sur cette page :

http ://wiki.postgresql.org/wiki/Postgres

Origines (années 2000)

Apparitions de la communauté internationale
- ~ 2000 : Communauté japonaise

Site de la communauté japonaise : http ://www.postgresql.jp
- 2004 : Communauté francophone

Site de la communauté francophone : http ://www.postgresql.fr

Un peu de nostalgie :

http ://web.archive.org/web/20040224170256/http ://www.postgresqlfr.org/
- 2006 : SPI

En 2006, le PGDG intègre le « Software in the Public Interest », Inc. (SPI), une organisation à but non lucratif chargée de collecter et redistribuer des financements.

Site du SPI : http ://www.spi-inc.org
- 2007 : Communauté italienne

Site de la communauté italienne : http ://www.itpug.org
- 2008 : PostgreSQL Europe et US

En 2008, douze ans après la création du projet, des associations d'utilisateurs apparaissent pour soutenir, promouvoir et développer PostgreSQL à l'échelle internationale.

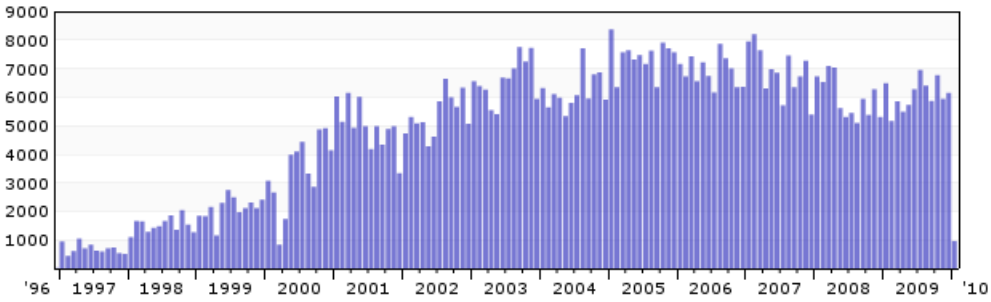
PostgreSQL UK organise une journée de conférences à Londres, PostgreSQL FR en organise une à Toulouse.

Site de la communauté européenne : http ://www.postgresql.eu
- 2009 : Boom des PGDay

Le PGDay européen s'est déroulé à Paris, les 6 et 7 novembre 2009. Il a été un grand succès avec plus de 180 personnes assistant à deux journées de conférences. Plus d'informations sur :

http ://2009.pgday.eu/start .

Progression du projet - 1



- Évolution du trafic des listes de diffusion de PostgreSQL
- Augmentation très importante jusqu'en 2005
- Reste constante par la suite
- 6000 messages en moyenne par mois

Ce graphe représente l'évolution du trafic des listes de diffusion du projet qui est corrolaire du nombre d'utilisateurs du logiciel.

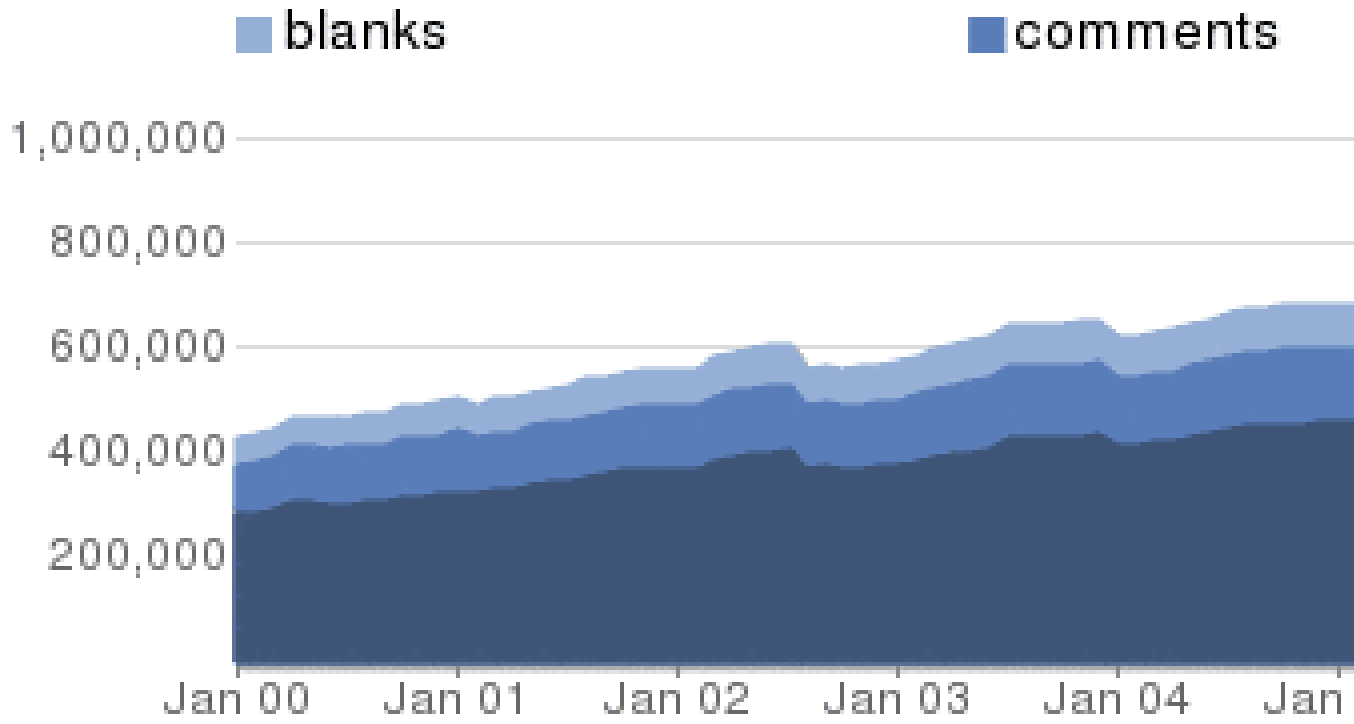
Source : http ://postgresql.markmail.org/

Comparaison des listes PostgreSQL et MySQL (datant de février 2008) : http ://markmail.blogspot.com/2008/0

On peut également visualiser l'évolution des contributions de la communauté PostgreSQL grâce au projet `Code Swarm` :

http ://www.vimeo.com/1081680

Progression du projet - 2



- Évolution du nombre de lignes dans le code de PostgreSQL
- Augmentation constante depuis 2000
- Environ 500000 lignes de code C actuellement
- Soit une augmentation de 25000 lignes de code C par an

Ce graphe représente l'évolution du nombre de lignes de code dans les sources de PostgreSQL. Cela permet de bien visualiser l'évolution du projet.

Il y a donc environ 900.000 lignes de code, pour environ 200 développeurs actifs.

Source : https ://www.ohloh.net/p/postgres/analyses/latest

Communauté

La communauté PostgreSQL est structurée en trois cercles concentriques :

- 1er cercle : Les développeurs ou « hackers »

Ils sont quelques dizaines. Ce sont ces personnes qui décident et réalisent les évolutions du projet.

http ://wiki.postgresql.org/wiki/Committers

- 2ème cercle : les contributeurs

Ils sont quelques centaines, principalement pour l'administration des serveurs, pour l'écriture de la documentation et pour sa traduction.

- 3ème cercle : les utilisateurs

Combien ?

Il est très difficile de savoir combien de personnes téléchargent PostgreSQL et complètement impossible de savoir combien de personnes et d'entreprises l'utilisent.

PostgreSQL Core Team

Le terme « Core Hackers » désigne les personnes qui sont dans la communauté depuis longtemps. Ces personnes désignent directement les nouveaux membres.

La « Core Team » est un ensemble de personnes doté d'un pouvoir assez limité. Ils peuvent décider de la sortie d'une version. Ce sont les personnes qui sont immédiatement au courant des failles de sécurité du serveur PostgreSQL. Tout le reste des décisions est pris par la communauté dans son ensemble après discussion, généralement sur la liste pgsql-hackers.

Détails sur les membres actuels de la *core team* :

- **Josh Berkus**, PostgreSQL Experts Inc, San Francisco (USA)
- **Peter Eisentraut**, F-Prot Antivirus, Aachen (Allemagne)
- **Marc G. Fournier**, Hub.Org Networking Services, Wolfville (Canada)
- **Tom Lane**, Red Hat, Pittsburgh (USA)
- **Bruce Momjian**, EnterpriseDB, Philadelphia (USA)
- **Dave Page**, EnterpriseDB, Oxfordshire (Angleterre)
- **Jan Wieck**, Afilias USA INC, Philadelphia (USA)

Le terme « hacker » peut porter à confusion, il s'agit ici de la définition « universitaire » :

http ://fr.wikipedia.org/wiki/Hacker_(université)

Contributeurs

Actuellement, PostgreSQL compte une centaine de « contributeurs » qui se répartissent les taches suivantes :

- Développement des projets satellites (Slony, pgAdmin, ...)
- Promotion du logiciel
- Administration des serveurs
- Rédaction de documentation
- Conférences
- Organisation de groupes locaux

Le *PGDG* a fêté son 10ième anniversaire à Toronto en juillet 2006. Ce « PostgreSQL Anniversary Summit » a réuni pas moins de 80 membres actifs du projet.

PGCon2009 a réuni 180 membres actifs à Ottawa.

Utilisateurs

- Vous !
- **Le succès d'un logiciel libre dépend de ses utilisateurs.**

Il est impossible de connaître précisément le nombre d'utlisateurs de PostgreSQL. On sait toutefois que ce nombre est en constante augmentation.

Il existe différentes manières de s'impliquer dans une communauté Open-Source. Dans le cas de PostgreSQL, vous pouvez :

- Déclarer un bug
- Tester les versions bêta
- Témoigner

Pourquoi participer ?

Au-delà de motivations idéologique ou technologique, il y a de nombreuses raisons objectives de participer au projet PostgreSQL.

- Déclarer un bug

Envoyer une description d'un problème applicatif aux développeurs est évidemment le meilleur moyen d'obtenir sa correction.

Attention toutefois à être précis et complet lorsque vous déclarez un bug ! Assurez-vous que vous pouvez le reproduire...

- Tester

Tester les versions « candidates » dans votre environnement (matériel et applicatif) est la meilleure garantie que votre système d'information sera compatible avec les futures versions du logiciel.

- Témoigner

Les retours d'expérience et les cas d'utilisations professionnelles sont autant de preuves de la qualité de PostgreSQL. Ces témoignages aident de nouveaux utilisateurs à opter pour PostgreSQL, ce qui renforce la communauté.

- Traduire et répondre aux questions

S'impliquer dans les efforts de traductions, de relecture ou dans les forums d'entraide ainsi que toute forme de transmission en général est un très bon moyen de vérifier et d'approndir ses compétences.

PostgreSQL, un projet mondial

Des contributeurs de toutes nations.

Quelques faits :

- Le projet est principalement anglophone
- Il existe une très grande communauté au Japon
- La communauté francophone est très dynamique mais on trouve très peu de développeurs francophones
- La communauté hispanophone est naissante

- Les développeurs du noyau (« core hackers ») vivent en Europe, au Japon et en Amérique du Nord

Sponsors

- Sun Microsystems
- NTT (Streaming Replication)
- Fujitsu
- Red Hat (Tom Lane)
- Skype (projet skytools)
- EnterpriseDB (Bruce Momjian, Dave Page, Heikki Linnakangas, ...)
- Dalibo (soutient la communauté française)

Depuis juin 2006, Sun Solaris embarque PostgreSQL dans sa distribution de base, comme base de données de référence pour ce système d'exploitation.

Le rachat de MySQL par SUN ne constitue pas un danger pour PostgreSQL. Au contraire, SUN a rappelé son attachement et son implication dans le projet.

http ://www.news.com/Sun-backs-open-source-database-PostgreSQL/2100-1014_3-5958850.html

http ://www.postgresqlfr.org/ ?q=node/1529

http ://www.lemondeinformatique.fr/actualites/lire-sun-encourage-a-essayer-la-version-83-de-postgresql

Le rachat de SUN par Oracle ne constitue pas non plus un danger.

NTT finance un groupe de développeurs sur PostgreSQL, ce qui lui a permis de fournir de nombreux patchs pour PostgreSQL, le dernier en date concernant un système de réplication interne au moteur. Ce système est en cours d'inclusion dans la version de la communauté. Plus d'informations sur :

http ://wiki.postgresql.org/wiki/Streaming_Replication

Fujitsu a participé à de nombreux développements aux débuts de PostgreSQL.

Red Hat emploie Tom Lane à plein temps pour travailler sur PostgreSQL. Il peut dédier une très grande partie de son temps de travail à ce projet, bien qu'il ait d'autres affectations au sein de Red Hat. Il maintient quelques paquets RPM, dont ceux du SGBD PostgreSQL. Il leur assure une maintenance sur leur anciennes versions pour les distributions Red Hat à grande durée de vie.

Skype est apparu il y a deux/trois ans. Ils proposent un certain nombre d'outils très intéressants : pgBouncer (pooler de connexion), Londiste (réplication par trigger), etc. Ce sont des outils qu'ils utilisent en interne et qu'ils publient sous licence BSD comme retour à la communauté.

EnterpriseDB est une société anglaise qui a décidé de fournir une version de PostgreSQL propriétaire fournissant une couche de compatibilité avec Oracle. Ils emploient plusieurs codeurs importants du projet PostgreSQL (dont deux font partie de la « Core Team »), et reversent un certain nombre de leurs travaux au sein du moteur communautaire.

Dalibo participe aussi à sa mesure à la communauté. Elle est membre Gold de PostgreSQL Europe. Elle sponsorise des événements comme le PGDay italien, les PGDay français ainsi que la communauté francophone. Enfin, elle participe à de nombreux développements. Plus d'informations sur :

http ://www.dalibo.org/contributions

Références

- Yahoo
- Météo France
- RATP

Yahoo

- Yahoo !

Le 21 mai 2008, à l'occasion du rendez-vous annuel de PostgreSQL au Canada (PGCon 2008), Yahoo a annoncé détenir ce qui semble être la plus grande base de données en production connue à ce jour.

- 2 Petaoctets !

Les informations stockées sont des données structurées à propos des visiteurs de leur site web. Ces données sont utilisées pour améliorer la publicité en ciblant plus précisément les annonces commerciales. Pour cela, Yahoo doit effectuer des analyses très profondes des données récoltées.

- La flexibilité de l'open-source

Yahoo utilise PostgreSQL auquel est ajoutée une sur-couche logicielle spécifique, conçue pour exploiter le cluster de machines hébergeant les données. Ce cluster est composé d'environ 1 000 serveurs de type PC répartis dans différents datacenters.

- Économie de serveurs, économie d'énergies

Selon Yahoo, le nombre de serveurs est 10 à 20 % inférieur à ce qui est nécessaire pour d'autres solutions conventionnelles comme Oracle, IBM's DB2 ou Teradata.

Source :

http ://blog.postgresql.fr/index.php ?post/drupal/293

Limites

Limites	Valeurs
Taille max. d'une base	Illimitée
Taille max. d'une table	32 To
Taille max. d'une ligne	1,6 To
Taille max. d'un champ	1 Go
Nombre max. de lignes par table	Illimité
Nombre max. de colonnes par table	de 250 à 1600
Nombre max. d'index par table	Illimité

Ces chiffres sont cités à titre indicatif pour illustrer la capacité de montée en charge (« scalabilité ») de PostgreSQL. Dans la pratique, ces limites ne sont jamais atteintes.

Pour plus de détails, consultez les pages de promotion du site postgresql.org

```
http ://www.postgresql.org/about/
http ://www.postgresql.org/community/propaganda
```

Roadmap

Pas de feuille de route officielle !

Se reporter aux documents suivants :

- *TODO list*
- Liste des commit fests

PostgreSQL est un projet Open-Source, non commercial et basé sur le volontariat. Il n'y a pas de liste formelle de fonctionnalités à implémenter pour les développeurs. Le « mantra » de PostgreSQL est de laisser les développeurs travailler sur les aspects qui les intéressent.

Les développeurs voulant participer mais ne sachant pas quoi coder peuvent trouver l'inspiration sur la liste ToDo. Cette liste comprends des fonctionnalités manquantes dans PostgreSQL. Elle est un agrégat des demandes des utilisateurs sur les liste de discussion.

TODO list : http ://wiki.postgresql.org/wiki/Todo

Cependant les choses évoluent et un nouveau processus de développement a été testé pendant le développement de la version 8.4. L'idée est d'instaurer des périodes de vérification des patchs en attente. Ces périodes sont appelées « commit fest » (le « commit » étant l'opération qui permet d'enregistrer un patch sur le dépôt des sources). Cela a plutôt bien fonctionné pour le développement de la 8.4. Après analyse, les développeurs se sont aperçus qu'il faudrait un gestionnaire pour diriger l'opération (gestionnaire qui ne serait pas un des développeurs). Le développement de la version 9.0 a bénéficié de cette amélioration, et il semble pour l'instant que c'est bénéfique. Un outil web a aussi été développé pour mieux gérer ce type de développement.

Site des commit fests : http ://commitfest.postgresql.org/

Les développeurs principaux tentent d'améliorer la gestion du développement et de faciliter l'ajout de patchs. Vous pouvez désormais suivre l'évolution des développement en cours sur le wiki officiel :

http ://wiki.postgresql.org/wiki/Development_information

Partie 2 : Les versions

- De 7.4 à 8.4
- 9.0 ?
- Quelle version choisir ?

Historique

- *1996* : v1.0
- *1997* : v6.0
- *1998* : v7.0
- *2003* : v7.4
- *janvier 2005* : v8.0
- *novembre 2005* : v8.1
- *décembre 2006* : v8.2
- *février 2008* : v8.3
- *juillet 2009* : v8.4
- *juillet 2010* : v9.0 ?

La version `8` marque l'entrée tant attendue de PostgreSQL dans le marché des *SGDB* de haut niveau, en apportant des fonctionnalités telles que les tablespaces, les procédures stockées en `Java`, le *Point In Time Recovery*, la réplication asynchrone ainsi qu'une version native pour `Windows`.

Versions courantes

Dernières releases (mars 2010) :

- version 7.4.28
- version 8.0.24
- version 8.1.20
- version 8.2.16
- version 8.3.10
- version 8.4.3

La philosophie générale des développeurs de PostgreSQL peut se résumer ainsi :

> « Notre politique se base sur la qualité, pas sur les dates de sortie »

Toutefois, même si cette philosophie reste très présente parmi les développeurs, depuis quelques années, les choses évoluent et la tendance actuelle est de livrer une version stable majeure tous les 12 à 15 mois.

Le support de la version 7.3 a été arrêté au début de l'année 2008. La même chose devrait arriver aux versions 7.4 et 8.0 milieu 2010. Pour plus de détails, voir :

http ://wiki.postgresql.org/wiki/PostgreSQL_Release_Support_Policy

Version 7.4

- fin 2003 (EOL, juillet 2010)
- « En finir avec les corruptions de données ! »
- Améliorations de `GROUP BY` et `IN / NOT IN`
- `VACUUM` plus efficace
- Apparition de la contribution `tsearch2`
- Apparition de la contribution `autovacuum`

`autovacuum` est directement intégré dans PostgreSQL à partir de la version 8.1.

`tsearch2` est directement intégré dans PostgreSQL à partir de la version 8.3.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-7-4.html

Cette version ne sera plus maintenue à partir de juillet 2010.

Version 8.0

- début 2005 (EOL, juillet 2010)
- Disponible en natif pour `Windows` (`NT4/2000/XP/2003/Vista/2008/7`)
- `Tablespaces`
- `Savepoints`
- `PITR`
- Améliorations de `CHECKPOINT` et `VACUUM`

Le port natif Windows de la version 8.0 est considéré comme une version beta dudit port.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-8-0.html

Cette version ne sera plus maintenue à partir de juillet 2010. Cette version n'est plus maintenue pour les plateformes Windows.

Version 8.1

- fin 2005 (EOL, novembre 2010)
- Rôles
- Paramètres `IN/OUT` dans les fonctions
- `Two-Phase Commit`
- `Autovacuum` intégré
- Meilleures performances sur les machines multi-processeurs
- Meilleur partionnement de tables
- `COPY` largement amélioré

On doit une grande partie des grosses améliorations de la 8.1 à l'effervescence sur le projet Bizgres, dont les *patchs* ont été reversés au projet PostgreSQL, qui les inclut désormais en standard.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-8-1.html

Cette version ne sera plus maintenue à partir de novembre 2010. Cette version n'est plus maintenue pour les plateformes Windows.

Version 8.2

- fin 2006 (EOL, décembre 2011)
- Compatibilité `SQL2003`
- Meilleur support de `Windows` et des migrations depuis `Oracle`
- Support de `LDAP` pour l'authentification
- `LogShipping`
- Amélioration des requêtes préparées
- Amélioration des performances
- bien d'autres encore...

Cette version dispose de peu de modifications majeures. Elle correspond plutôt à une stabilisation du projet. C'est d'ailleurs la première version déclarée stable pour la plateforme Windows.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-8-2.html

Cette version ne sera plus maintenue à partir de décembre 2011.

Version 8.3

- début 2008 (EOL, février 2013)
- Nombreuses améliorations sur les performances : `HOT`, commit asynchrone, etc.
- Recherche plein texte
- `XML`
- Journalisation `CSV`
- Nouveaux types : `enum`, `UUID`
- Nouvelle authentification `GSSAPI`/`SSPI`
- Bien d'autres encore...

Cette version comporte de nombreuses nouvelles évolutions, notamment des modules contrib qui atteignent enfin le coeur du projet.

Attention ! Cette version a la réputation d'être extrêmement rigoureuse !

Lors d'une migration, vous pourrez rencontrer des problèmes avec :

- l'encodage de votre base
- les conversions implicites

En effet, la version 8.3.x ne contient plus une série de conversion de types qui étaient réalisées implicitement dans les versions précédentes. Ceci peut vous amener à faire une revue et une correction de votre code avant la migration.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-8-3.html

ainsi que :

http ://www.dalibo.org/glmf103_postgresql_8.3_quoi_de_neuf

Cette version ne sera plus maintenue à partir de février 2013.

Version 8.4

- 1er juillet 2009 (EOL, juillet 2014)
- Fonctionnalités :
  - ⇒ Fonctions Window (clauses WITH, OVER) et SQL/MED ;
  - ⇒ CTE et requêtes récursives ;
  - ⇒ Paramètres par défaut et paramètres variadic pour les fonctions ;
  - ⇒ Restauration parallélisée d'une sauvegarde ;
  - ⇒ Droits sur les colonnes ;
- Mais aussi...
  - ⇒ Locale configurable par base de données ;
  - ⇒ Refonte du FSM ;
  - ⇒ VACUUM sélectif grâce au « visibility map » ;
  - ⇒ Support des certificats SSL ;
  - ⇒ Statistiques sur les fonctions ;
  - ⇒ Fonction pg_terminate_backend() ;
  - ⇒ Nouveaux modules contrib : pg_stat_statements, auto_explain, ...

Exemple de la volonté de ne pas intégrer de fonctionnalité pas suffisamment bien codé, le Hot Standby, fonctionnalité très attendue par les utilisateurs, a finalement été repoussé pour une prochaine version car les développeurs estimaient qu'il n'était pas assez fiable.

Pour plus de détails sur les nouveautés de cette version, consultez :

http ://www.postgresql.org/docs/current/interactive/release-8-4.html

Cette version ne sera plus maintenue à partir de juillet 2014.

Version 9.0

- Actuellement en version alpha 3
- Sortie prévue première moitié de 2010
- Fonctionnalités déjà intégrées :
  - ⇒ Hot Standby ;
  - ⇒ Streaming Replication ;
  - ⇒ Contraintes d'exclusion ;
  - ⇒ Beaucoup d'améliorations pour l'`EXPLAIN` ;
  - ⇒ Contrainte UNIQUE déferrable ;
  - ⇒ Droits par défaut, `GRANT ALL` ;
  - ⇒ Triggers : sur colonne, et clause `WHEN`.
- Mais aussi...
  - ⇒ Droit d'accès aux « Large Objects » ;
  - ⇒ Configurations par utilisateurs, par bases de données mais aussi par couple utilisateur/base ;
  - ⇒ Bloc de code anonyme.

Pour plus de détails sur les nouveautés de cette version, consultez la section 9.0 de la page suivante :

http ://wiki.postgresql.org/wiki/Development_information

La décision de passer en 9.0 a été prise 21 janvier 2010, la fontionnalité de réplication native étant une des évolutions majeures attendues.

Quelle version utiliser ?

- 7.3 et inférieures : **Migrer immédiatement !**
- 7.4.28 et 8.0.24 : Migrer le plus rapidement possible
- 8.1.20 : Mise à jour uniquement, planifier une migration
- 8.2.16 et 8.3.10 : Mise à jour uniquement
- 8.4.3 : Nouvelles installations
- 8.5 alpha 4 : Pour les nouveaux développements

Si vous avez une version 8.1 ou inférieure, planifiez le plus rapidement possible une migration vers la 8.4.

Actuellement, la version 8.4 doit être utilisée pour les installations. Cependant, les devéloppements ont plutôt intérêt à viser la version 9.0.

Vous pouvez consulter le tableau comparatif des versions sur le site officiel :

http ://www.postgresql.org/about/featurematrix

Versions dérivées

Il existe de nombreuses versions dérivées de PostgreSQL. Elles sont en général destinées à des cas d'utilisation très spécifiques. Leur code est souvent fermé.

- Red Hat Database
- Postgres Plus (compatibilité Oracle)
- greenplum / Netezza (data warehouse)

La liste des « forks » est disponible sur le wiki officiel :

http ://wiki.postgresql.org/wiki/PostgreSQL_derived_databases

Sauf cas très précis, il est recommandé d'utiliser la version officielle, libre et gratuite.

Partie 3 : Tour d’horizon technique

- Caractéristiques générales
- Le cœur
- Développement
- Sécurité
- SQL avancé
- Extensibilité

Caractéristiques

- Libre de tout droit (licence BSD)
- Robustesse prouvée sur plusieurs années
- Conçu pour une administration minimale
- Simplicité grâce à de bons outils d'administration
- Portable, fonctionne sur de nombreuses plates-formes
- Extensible, avec des *API* très bien documentées
- Plusieurs alternatives pour la haute-disponibilité et la réplication
- Support excellent, tant de la communauté que de la part d'entreprises spécialisées

Fonctionnalités : cœur

- Standard SQL
- Respect complet d'ACID
- MVCC (supérieur au verrou de lignes)
- Intégrité référentielle (clés étrangères)
- Index fonctionnels et partiels

MVCC (Multi Version Concurrency Control) est le mécanisme interne de PostgreSQL utilisé pour garantir la consistence des données lorsque plusieurs processus accèdent à la même table. C'est la qualité de l'implémentation de ce système qui fait de PostgreSQL un des meilleurs SGBD au monde : chaque transaction travaille dans son instantané de la base, cohérent du début à la fin de ses opérations. Par ailleurs les écrivains ne bloquent pas les lecteurs et les lecteurs ne bloquent pas les écrivains, contrairement aux SGBD s'appuyant sur des verrous de lignes.

Les index partiels sont des index limités à un sous-ensemble d'une table. Ces index sont définis par une condition et seules les lignes qui remplissent ce prédicat sont indexées.

Exemple

```
CREATE INDEX personne_nom_ix ON personne(nom)
WHERE NOT ( age >= 18  AND age <= 60 ) ;
```

Fonctionnalités : développement

Au niveau SGBD :
- Procédures stockées : PL/PgSQL
- 15 langages de procédures : PL/Perl, PL/Python, etc.

En externe :
- Interfaces natives pour ODBC, JDBC, C, PHP, Perl, etc.
- API ouverte et documentée

Voici la liste complète des langages procéduraux supportés

```
pl/pgsql
pl/sql
pl/python
pl/perl
pl/tcl
pl/sh
pl/R
pl/java
pl/js
pl/lolcode
pl/scheme
pl/php
pl/ruby
pl/j
pl/lua
pl/pgpsm
```

Retrouvez le tableau de fonctionnalités sur le wiki officiel :

http ://wiki.postgresql.org/wiki/PL_Matrix

Fonctionnalités : sécurité

- Fichier `pg_hba.conf`
- Filtrage `IP`
- Authentification
  - ⇒ Interne : mots de passe en clair ou chiffrés en `MD5`
  - ⇒ Externe : support natif de `identd`, `LDAP`, `GSSAPI`/`SSPI`
- Chiffrement de la connexion
  - ⇒ Support natif de `SSL`

Le chiffrage `crypt` existait pour les versions antérieures à la 8.4 mais, vu sa faiblesse, a été supprimé.

Le support des annuaires `LDAP` est disponible à partir de la version 8.2.

Le support de `GSSAPI`/`SSPI` est disponible à partir de la version 8.3. L'interface de programmation GSS API est un standard de l'IETF qui permet de sécuriser les services informatiques. La principale implémentation de GSSAPI est `Kerberos`. SSPI permet le Single Sign On sous MS Windows, de façon transparente, qu'on soit dans un domaine AD ou NTLM.

La gestion des certificats `SSL` est disponible à partir de la version 8.4.

Fonctionnalités : SQL

- Excellent support du `SQL ANSI`
- Objets `SQL` : tables, vues, règles, séquences, triggers
- Opérations : jointures, sous-requêtes, requêtes CTE, requêtes Window, etc.
- Curseurs
- Héritage
- Unicode et plus de 50 encodages

Fonctionnalités : extensibilité

Création de types de données et
- de leurs fonctions
- de leurs opérateurs
- de leurs règles
- de leurs aggrégats

Exemple de création d'un type

```
CREATE TYPE serveur AS (
       nom             text,
       adresse_ip      text,
       administrateur  text
       ) ;
```

Exemple de création d'un opérateur

```
CREATE OPERATOR + ( leftarg = stock, rightarg = stock,
    procedure = stock_fusion, commutator = + ) ;
```

Conformité SQL

Dernière version du standard SQL : SQL :2008

Aucun SGBD ne supporte complètement SQL :2008 à ce jour

- PostgreSQL tente de s'en approcher au maximum, au fil des versions
- Une bonne partie de SQL :2008 est supportée, bien que parfois avec des syntaxes différentes

ACID

- **Atomicité** (*Atomic*)
  Une transaction est entière : « tout ou rien ».
- **Cohérence** (*Consistent*)
  Une transaction amène la base d'un état stable à un autre.
- **Isolation** (*Isolated*)
  Les transactions n'agissent pas les unes sur les autres.
- **Durabilité** (*Durable*)
  Une transaction validée provoque des changements permanents.

Les propriétés ACID sont quatre propriétés essentielles d'un sous-système de traitement de transactions d'un système de gestion de base de données. On considère parfois que seuls les SGBD qui respectent ces propriétés sont dignes d'être considérées comme des bases de données.

MultiVersion Concurrency Control (MVCC)

MVCC fluidifie les mises à jour en évitant les blocages trop contraignants (verrous sur UPDATE) entre sessions et par conséquent de meilleures performances en contexte transactionnel.

MVCC maintient toutes les version de chaque tuple nécessaires, ainsi :

1. Chaque transaction voit son image de la base (appelée *snapshot*) telle qu'elle était lors du démarrage de la transaction, quoi que modifient les autres transactions.
2. Une lecture ne bloque pas une écriture.
3. Une écriture ne bloque pas une lecture.
4. Une écriture ne bloque les autres écritures que lors de la mise à jour de la **même version d'une ligne**.
5. MVCC permet la sauvegarde à chaud cohérente : la sauvegarde n'est qu'une

Enumerated list ends without a blank line ; unexpected unindent.

transaction cemme les autres, qui a donc une vision cohérente de la base.

Transactions

Intimement liées à ACID et MVCC :

- Une transaction est un ensemble d'opérations atomiques
- Le résultat d'une transaction est « tout ou rien »

Exemple

```
BEGIN ;

UPDATE salaires
SET montant = montant * 1.10
WHERE trig<>'jpa' ;

UPDATE salaires
SET montant = montant * 2
WHERE trig='jpa' ;

COMMIT ;
```

- SAVEPOINT disponible pour sauvegarde des modifications d'une transaction à un instant t

Un point de sauvegarde est une marque spéciale à l'intérieur d'une transaction qui autorise l'annulation de toutes les commandes exécutées après son établissement, restaurant la transaction dans l'état où elle était au moment de l'établissement du point de sauvegarde.

Vues

- Fondées sur une requête : masque la complexité
- Interface cohérente vers les données, même si les tables évoluent
- Abstraction qui permet de réduire la complexité des requêtes qui l'utilisent
- Moyen d'améliorer la sécurité en contrôlant l'accès aux données de manière sélective

À savoir que les vues sont exécutées en tant que l'utilisateur qui les a créé.

Exemple :

```
test=# CREATE TABLE personne (
test-# nom TEXT, prenom TEXT, num_cartecredit TEXT) ;
test=# INSERT INTO personne VALUES ('Duff','John','1234567890123456') ;

test=# CREATE VIEW personne_anon AS
test-# SELECT nom, prenom,
test-# substring(num_cartecredit,0,10)||'******' as num_cc_anon
test-# FROM personne ;

test=# SELECT * FROM personne_anon ;
 nom  | prenom |  num_cc_anon
------+--------+----------------
 Duff | John   | 1234567890******
```

À partir de la 8.4, il est possible de modifier une vue en lui ajoutant des colonnes à la fin, au lieu de devoir les détruire et recréer (ainsi que toutes les vues qui en dépendent, ce qui pouvait être fastidieux).

Exemple :

```
test=# CREATE  OR REPLACE VIEW personne_anon AS
test-# SELECT nom, prenom,
test-# substring(num_cartecredit,0,10)||'******' as num_cc_anon,
test-# md5(substring(num_cartecredit,0,10)) as num_md5_cc
test-# FROM personne ;

tests=# SELECT * FROM personne_anon ;
 nom  | prenom |   num_cc_anon   |            num_md5_cc
------+--------+-----------------+----------------------------------
 Duff | John   | 123456789****** | 25f9e794323b453885f5181f1b624d0b
```

- **Pas** de vues matérialisées à ce jour dans PostgreSQL

Des travaux sont en cours pour intégrer les vues matérialisées.

Pour l'instant, c'est possible en écrivant des fonctions PL/pgsql

http ://jonathangardner.net/PostgreSQL/materialized_views/matviews.html

Schémas

Ce sont des espaces de noms dans une base de données permettant :
- de grouper les objets d'une base de données
- de séparer les utilisateurs entre eux
- de contrôler plus efficacement les accès aux données
- d'éviter les conflits de noms dans les grosses bases de données

Les schémas sont très utiles pour les systèmes de réplication (Slony, bucardo).

Exemple de schéma :

```
CREATE SCHEMA paris ;
SET search_path TO paris ;
CREATE TABLE monuments (...) ;

CREATE SCHEMA limoges ;
SET search_path TO limoges ;
CREATE TABLE monuments (...) ;

-- Quels monuments portant
-- un nom identique peut-on trouver
-- à Paris et Limoges ?
SELECT paris.monuments.nom
FROM paris.monuments
WHERE paris.monuments.nom = limoges.monuments.nom
```

Contraintes

Elles permettent une vérification qualitative des données, au delà du type de données :

- `CHECK` : `prix > 0`
- `NOT NULL` : `id_client NOT NULL`
- Unicité : `id_client UNIQUE`
- Clés primaires : `UNIQUE NOT NULL` ==> `PRIMARY KEY (id_client)`
- Clés étrangères : `produit_id REFERENCES produits(id_produit)`

Les contraintes sont la garantie de conserver des données de qualité !

Triggers

- Exécutés avant (`BEFORE`) ou après (`AFTER`) une opération
- Opérations : `INSERT`, `UPDATE`, `DELETE`
- COPY déclenche le trigger INSERT
- Trigger TRUNCATE à partir de la version 8.4
- Soit pour l'ensemble de la requête (`FOR STATEMENT`)
- Soit pour chaque ligne impactée (`FOR EACH ROW`)
- Peuvent être écrits dans n'importe lequel des langages de procédure supportés par PostgreSQL (`C`, `PL/PgSQL`, `PL/Perl`, etc.)

Exemple :

```
test=# CREATE LANGUAGE plpgsql ;
test=# CREATE TABLE personne ( nom text, salaire integer) ;


test=# CREATE FUNCTION verif_salaire()
   RETURNS trigger AS $verif_salaire$
   BEGIN
       -- On verifie que les variables ne sont pas vides
       IF NEW.nom IS NULL THEN
           RAISE EXCEPTION 'le nom ne doit pas être null' ;
       END IF ;
       IF NEW.salaire IS NULL THEN
           RAISE EXCEPTION 'le salaire ne doit pas être null' ;
       END IF ;

       -- pas de baisse de salaires !
       IF NEW.salaire < OLD.salaire THEN
           RAISE EXCEPTION 'pas de baisse de salaire !' ;
       END IF ;

       RETURN NEW ;
   END ;
   $verif_salaire$ LANGUAGE plpgsql ;

test=# CREATE TRIGGER verif_salaire BEFORE INSERT OR UPDATE ON personne
test-# FOR EACH ROW EXECUTE PROCEDURE verif_salaire() ;
```

Héritage

- Même concept que l'héritage dans les langages orientés objet
- Une table héritée récupère toutes les colonnes de la table parente
- On peut limiter les requêtes à la table parente uniquement (`SELECT FROM ONLY table_mere ;`)
- L'héritage fonctionne avec tous les ordres *DML*

`DML` est l'acronyme de `Data Modification Language`, autrement dit les instructions de modifications des données (`INSERT` par exemple).

Exemple d'héritage

```
-- Table personne
create table personne (
        id int4 unique not null,
        nom text,
        prenom text
) ;

-- Table prof hérite de personne
create table prof (
        nbheures int4,
        specialite text
) inherits (personne) ;

-- Table etudiant hérite de personne
create table etudiant (
        section text
) inherits (personne) ;

--Table thesard hérite de Etudiant et Prof
create table thesard (
        annee date
) inherits (etudiant, prof) ;
```

- **Pas** encore le support des clés étrangères

Une clef étrangère pointant une colonne de la table mère ne pourra pas faire référence aux valeurs des tables filles.

exemple

```
-- Table absent
create table absent (
       id_personne int4  NOT NULL REFERENCES personne(id)
       ) ;

-- test
insert into prof(id,nom) values (1,'Tournesol') ;
insert into absent(id_personne) values (1) ;
```

```
ERREUR :  Une instruction insert ou update sur la table « absent
» viole la contrainte de clé étrangère «
absent_id_personne_fkey »
DETAIL :  La clé (id_personne)=(1) n'est pas présente dans la
table « personne ».
```

Index

Les algorithmes suivants sont supportés :

- `B-tree` (par défaut)
- `R-tree`
  R-tree est remplacé par GiST. Le terme reste par compatibilité.
- `Hash`
  Attention aux index `hash`. Leur modification n'est pas enregistrée dans les journaux de transactions, ce qui amène deux problèmes. En cas de crash du serveur, il est fréquemment nécessaire de les reconstruire (`REINDEX`). De plus, ils ne sont pas restaurés avec `PITR` et donc avec le `Log Shipping`. Par ailleurs, ils ne sont pour le moment que rarement plus performants que les index B-Tree.
- `GiST`
- `GIN` (version `8.2`)
  Pour une indexation standard , on utilise en général un arbre B
  Les index plus spécifiques (GIN, GIST) sont spécialisés pour les grands volumes de données complexes et multidimensionnelles : indexation textuelle, géométrique, ou géographique pas exemple.
  Plus d'informations :
  ⇒ http ://fr.wikipedia.org/wiki/Arbre_B
  ⇒ http ://en.wikipedia.org/wiki/R-tree
  ⇒ http ://fr.wikipedia.org/wiki/Table_de_hachage
  ⇒ http ://docs.postgresql.fr/current/textsearch-indexes.html

Write Ahead Logs, aka WAL

Technique standard de journalisation (*log*) de transactions :
- Changements : dans les journaux de transaction **puis** dans les fichiers de données
- Plus besoin de *flusher* les fichiers de données à chaque `COMMIT`

Les WAL sont une garantie contre les pertes de données.

Ainsi en cas de crash :

- PostgreSQL redémarre
- PostgreSQL vérifie s'il reste des données non intégrées aux fichiers de données dans les journaux (mode recovery)
- Si c'est le cas, ces données sont recopiées dans les fichiers de données afin de retrouver un état stable.

Avantages des WAL

- Nombre d'écritures sur disque réduit
- Un seul *sync* sur le fichier de transactions au lieu de potentiellement plusieurs sur les fichiers de données
- Le fichier de transactions est écrit de manière séquentielle
- Assure la cohérence des fichiers de données
- Permet la sauvegarde à chaud et la restauration à un temps dans le passé (`Point In Time Recovery`)

Point In Time Recovery, aka PITR (1/2)

En cas de *crash* disque :

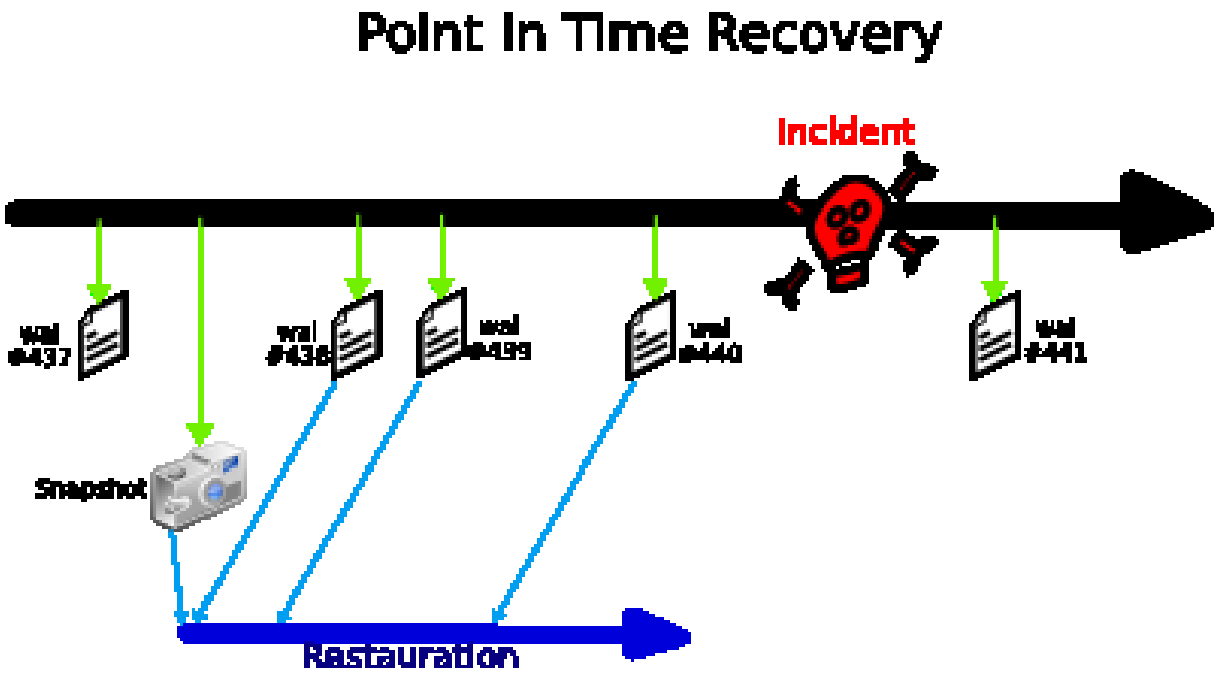
- Avant PostgreSQL 8 : restauration *dump* ou *fichiers* ou réplication
- À partir de PostgreSQL 8 : `PITR`

Point In Time Recovery, aka PITR (2/2)

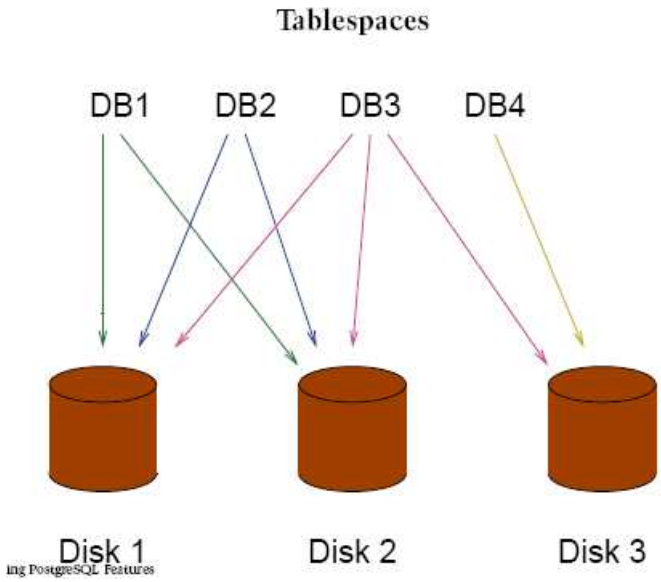
Avec le PITR, on peut désormais sauvegarder le serveur en continu :

- Les WAL contiennent tous les changements : on les sauvegarde...
- PITR basé sur une transmission continue des WAL vers une machine de secours
- Permet la restauration au moment du *crash* ou à un autre point dans le passé
- Permet aussi une restauration en continu sur un serveur en attente (fonctionnalité appelée Log Shipping)

Schéma :



Tablespaces



Avant PostgreSQL 8.0 : arrêt du serveur puis `ln -s` pour déplacer les fichiers de données ailleurs que dans le *cluster* PostgreSQL (`$PGDATA`).

À partir de la version 8.0, utilisation de `tablespaces`.

Les `tablespaces` peuvent contenir une base de données, un schéma, des tables ou index.

Tablespaces : avantages

- Granularité au niveau de l'objet
- Gain de performances via un contrôle de l'utilisation des disques
- Plus de flexibilité lorsqu'un disque arrive à saturation

Outils de la communauté

- Serveurs Web
- Listes de diffusion
- Forums / IRC
- Wiki
- Dalibo

Serveurs

- Site officiel : http ://www.postgresql.org/
- La doc : http ://www.postgresql.org/docs/
- Pour les développeurs externes : http ://pgfoundry.org/
- Actualité : http ://www.planetpostgresql.org/
- Documentation du code : http ://doxygen.postgresql.org/

`PG Foundry` est l'équivalent de `Source Forge` mais dédié à PostgreSQL.

Le site « Planet PostgreSQL » est un agrégateur de blogs des *core-hackers*, contributeurs, traducteurs ou simples utilisateurs de PostgreSQL.

Serveurs francophones

- Site officiel : http ://www.postgresql.fr/
- La doc : http ://docs.postgresql.fr/
- Actualité : http ://www.planet.postgresql.fr/
- Wiki : http ://wiki.postgresql.fr/

Le site PostgreSQLfr.org est le site de l'association des utilisateurs francophones du logiciel. La communauté francophone se charge de la traduction de toutes les documentations.

Listes de discussions / Listes d'annonces

- pgsql-announce
- pgsql-general
- pgsql-admin
- pgsql-sql
- pgsql-fr-generale
- pgsql-performance
- pgsql-advocacy

Pour s'inscrire ou consulter les archives :

http ://www.postgresql.org/community/lists/

Les mailing-lists sont les outils principaux de gouvernance du projet. Toute l'activité de la communauté (bugs, promotion, entraide, décisions) est accessible par ce canal.

Si vous avez une question ou un problème, la réponse se trouve probablement dans les archives ! Pourquoi ne pas utiliser un moteur de recherche spécifique :

http ://www.nabble.com/ http ://markmail.org/

N'hésitez pas à rejoindre ces listes.

Avant poser une question, consultez :

http ://www.linux-france.org/article/these/smart-questions/smart-questions-fr.html :

Forums / IRC

- Forums français : http ://forums.postgresql.fr/
  Un autre forum est disponible à l'adresse : http ://postgresql.developpez.com/
- IRC (anglophone) : #postgresql sur le réseau Freenode
  Des canaux de discussion spécifiques à certains projets connexes sont également disponibles, comme par exemple #slony
- IRC (européen) : #postgresql-eu sur le réseau Freenode
- IRC (francophone) : #postgresqlfr sur le réseau Freenode
  Le point d'entrée principal pour le réseau Freenode est le serveur : `irc.freenode.net`

La majorité des développeurs sont disponibles sur IRC et peuvent répondre à vos questions.

Attention ! vous devez poser votre question en public et ne pas solliciter de l'aide par message privé.

Wiki

- http ://wiki.postgresql.org

Le wiki est un nouvel outil de la communauté (2008). C'est un media très prometteur.

On peut y retrouver :

- L'avancée du développement
- De la documentation
- Des informations de promotion

Dalibo

Dalibo propose des services complémentaires :

- Annuaire
- Lettre d'information mensuelle
- Articles et modules de formations
- Base de connaissance

Dalibo maintient un annuaire web des sites et des documents relatifs à PostgreSQL :

http ://del.icio.us/dalibo

La lettre d'information retrace l'activité de PostgreSQL et vous avertit des dernières mises à jour. Pour vous abonner, rendez-vous sur :

http ://dalibo.com/-Contact-.html

Dalibo.org est le centre de ressource de la société, vous pourrez y trouver des documents, des tests, des tutoriels, des projets, des modules de formation...

http ://dalibo.org

Notre base de compétence est le plus grande source d'informations francophone après la documentation officielle.

https ://support.dalibo.com/kb/

Cette base de connaissance est en accès restreint. Envoyez un message à contact@dalibo.com pour obtenir une invitation.

Partie 5 : Les projets satellites

- Administration
- Réplication
- Pooling
- Monitoring
- PostGIS

La richesse de PostgreSQL réside dans la grande variété de projets « satellites » qui gravitent autour du projet principal.

pgAdmin

.

Site officiel : http ://www.pgadmin.org/

Version : 1.10.2

Licence : PostgreSQL (BSD/MIT)

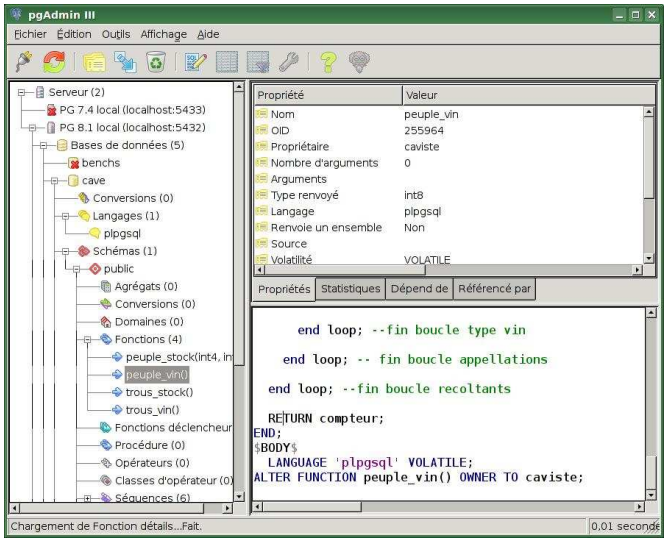
- Gestion graphique de l'administration des bases
- `EXPLAIN` graphique
- Gestion de `Slony`

Logiciel libre d'administration de la base de données PostgreSQL. Comprend une interface graphique d'administration, un outil de requêtage `SQL`, un éditeur de code procédural, un débogueur PL/psql, un éditeur des fichiers de configuration, une fenêtre de statut du serveur et bien plus encore.

`pgAdmin III` est conçu pour répondre à la plupart des besoins, depuis l'écriture de simples requêtes `SQL` jusqu'au développement de bases de données complexes. L'interface graphique supporte les fonctionnalités de PostgreSQL les plus récentes et facilite l'administration.

Il supporte toutes les versions de PostgreSQL supérieures à la version `7.3` ainsi que les versions commerciales de PostgreSQL comme `Pervasive Postgres`, `EnterpriseDB`, `Mammoth Replicator` et `SRA PowerGres`.

Disponible dans 17 langues et pour plusieurs systèmes d'exploitation, dont `MS Windows`, `Mac OSX`, `GNU/Linux` et `FreeBSD`.



PhpPgAdmin

Site officiel : http ://phppgadmin.sourceforge.net/

Version : 4.2.2

Licence : GPL

- Gestion graphique de l'administration des bases
- Interface web
- Gestion FTS

Le projet phpPgAdmin (ppa) poursuit sa prise en charge des différentes versions de PostgreSQL depuis la bien ancienne 7.0 à la toute dernière 8.3. Mais pas seulement. Les nouvelles fonctionnalités de la 8.3, tel que les types ENUM et le type UUID, la « recherche plein texte » (Full Text Search, durant le GSoC 2007 de ppa) et les paramétrages de coûts des procédures ont été intégrées. Les autres versions de PostgreSQL ont vu apparaître le support de fonctionnalités jusqu'alors absentes de ppa et que vous pourrez trouver dans les notes de version du projet.

Côté ergonomie, ppa a fait un petit pas (historique des requêtes utilisateur, quelques multi-actions, mise en page, etc) et continuera encore sur ce chemin. En parlant du futur, l'équipe de ppa supportant l'initiative goPHP5, la version à venir ne supportera officiellement plus php4. Cependant, le but étant d'avoir un projet fonctionnel sur un très large panel de configuration, ce support devrait débuter avec php 5.0, cette dernière étant toujours incluse dans certaines distributions qui la maintiennent toujours officiellement.

Slony

.

Site officiel : http ://www.slony.info

Version : 1.2.20

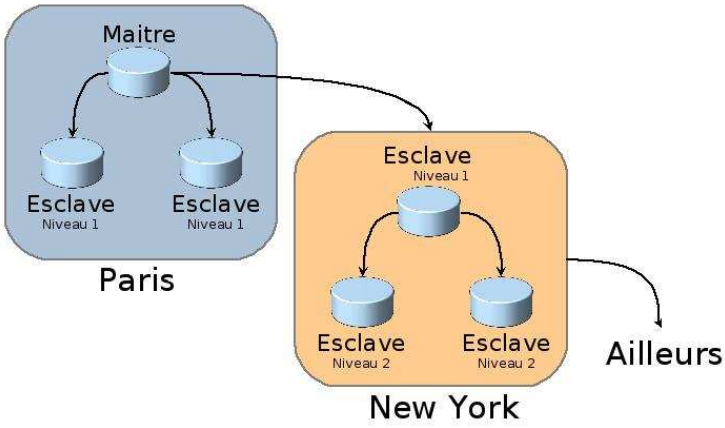
Licence : BSD

- Réplication 1 maître / plusieurs esclaves
- Agrégation : centralisation de plusieurs bases dans une seule.

`Slony-I` est un système de réplication maitre/esclaves supportant le *cascading*, le *failover* **et** le *fail-back*.

L'objectif de `Slony-I` est de pouvoir répliquer de grosses bases de données vers un nombre limité de systèmes esclaves.

`Slony-I` est une solution pour les *data-centers*, les serveurs de sauvegarde et plus généralement pour tous les systèmes de réplication dont les noeuds doivent être disponibles à un instant `t`.



Bucardo

Site officiel : http ://bucardo.org/

Version : 4.4.0

Licence : BSD

- Réplication Maître-Maître
- 2 noeuds uniquement

Bucardo est un système de réplication asynchrone pour PostgreSQL qui permet des opérations multi-maitre et maitre-esclave. Il a été developpé initialement par Greg Sabino Mullane pour la société End Point Corporation.

pgpool

Site officiel : http ://pgpool.projects.postgresql.org

Version : 3.4.1

Licence : BSD

- Point de connexion unique pour plusieurs serveurs
- Réduction de l'*overhead* de connexions
- Mécanismes *fail over* / *fail back*

`pgpool` est un outil de concentration (*pooling*) de connexion pour PostgreSQL. `pgpool` s'interpose entre les applications clientes (*frontends*) et les serveurs (*backends*). Chaque client peut se connecter à `pgpool` comme s'il était un serveur PostgreSQL standard. C'est-à-dire que le *pooling* est assuré de manière complètement transparente pour les clients.

`pgpool` crée un cache de connexions pour réduire l'*overhead* (i.e le coût initial incompressible) d'établissement des connexions.

`pgpool` peut également utiliser deux serveurs PostgreSQL, voire plus depuis la version 3, pour garantir la disponibilité du service en mettant en place un système de *fail over*. Si le premier serveur connait des problèmes, `pgpool` redirigera automatiquement les connexions vers un autre serveur.

PgBouncer

Site officiel : https ://developer.skype.com/SkypeGarage/DbProjects/PgBouncer

Version : 1.3.2

Licence : BSD

- très rapide
- très léger
- très prometteur
- développé par Skype

http ://dalibo.org/comparatif_des_outils_de_pooling_de_connexions

pgFouine

Site officiel : http ://pgfouine.projects.postgresql.org

Version : 1.2

Licence : GPL

- Analyse des journaux applicatifs PostgreSQL
- Recherche des requêtes consommatrices, longues ou fréquentes
- Analyse des journaux applicatifs du `VACUUM` depuis la dernière version

`pgFouine` est un analyseur des journaux applicatifs de PostgreSQL. Il permet de créer des rapports détaillés depuis ceux-ci. `pgFouine` est utilisé pour déterminer les requêtes à améliorer en priorité pour accélerer son application basée sur PostgreSQL .

`pgFouine` est écrit en `PHP` (mode client) orienté objet et est facilement extensible si vous avez besoin de rapports spécifiques.

`pgFouine` est conçu pour traiter de gros fichiers de logs avec une mémoire réduite.

Munin

.

Site officiel : http ://munin.projects.linpro.no

Version : 1.4.4

Licence : GPL

- Graphes & surveillance
- Plug-ins pour PostgreSQL
- Disponibles sur http ://www.dalibo.org/

Les plugins de dalibo ont été en grande partie repris par le projet `Munin`, qui les a ensuite adapté. On peut trouver ces plugins à l'adresse suivante :

http ://munin.projects.linpro.no/wiki/PluginCat#Postgresql

On remarquera l'impressionnant catalogue de *plugins* disponibles pour `Munin`.

pgsnap

Site officiel : http ://pgsnap.projects.postgresql.org/

Version : 0.5

Licence : BSD

- « Photographie » de votre SGBD à un instant t
- Très léger
- Très pratique
- Developpé par Guillaume Lelarge et Dalibo

pgsnap se base sur l'outil OraSnap d'Oracle. Cet outil crée un ensemble de fichiers HTML décrivant l'état d'une base de données Oracle.

La majorité des informations systèmes de PostgreSQL est disponible.

PostGIS

.

Site officiel : http ://postgis.refractions.net

Version : 1.5.1

Licence : GPL

- Module spatial pour PostgreSQL
- Conforme aux spécifications de l'OpenGIS Consortium
- Compatible avec MapServer

`PostGIS` ajoute le support d'objets géographiques à PostgreSQL. En fait, `PostGIS` transforme un serveur PostgreSQL en serveur de données spatiales, qui sera utilisé par un système d'information géographique (*SIG*), tout comme le `SDE` de la société `ESRI` ou bien l'extension `Oracle Spatial`. `PostGIS` se conforme aux directives du consortium `OpenGIS` et a été certifié par cet organisme comme tel, ce qui est la garantie du respect des standards par `PostGIS`.

`PostGIS` a été développé par la société `Refractions Research` comme une technologie Open-Source de base de données spatiale. Cette société continue à développer `PostGIS`, soutenue par une communauté active de contributeurs.

Parmi la liste des projets en cours, on trouve un support complet de la topologie, le support des fonctionnalités *raster*, les réseaux et la recherche de chemins dans ceux-ci, les surfaces tri-dimensionnelles, les courbes et bien d'autres fonctionnalités !

Avantages

- Licence BSD
  - ⇒ Coût nul
  - ⇒ Code source disponible
  - ⇒ Aucune contrainte de redistribution
- Robustesse
  - ⇒ Sécurité des données
  - ⇒ Reprise en cas de crash
  - ⇒ Résistance aux bogues applicatifs
- Souplesse, extensibilité
  - ⇒ Facilité de configuration de PostgreSQL
  - ⇒ Montée en puissance et en charge progressive
  - ⇒ Gestion des gros volumes de données
- Performances
  - ⇒ *Tuning* matériel
  - ⇒ *Tuning* logiciel
  - ⇒ Optimiseur statistique de requêtes évolué

Conclusion

- Un projet de grande ampleur
- Un SGBD complet
- Une communauté réactive et internationale
- Un large panel de projets complémentaires
- Une solution **stable**, **ouverte** , **performante** et **éprouvée**

Questions

N’hésitez pas, c’est le moment !