



# Présentation de PostgreSQL

**Formateur :** Guillaume Lelarge  
**Contact :** [guillaume.lelarge@dalibo.com](mailto:guillaume.lelarge@dalibo.com)  
**Date :** mai 2008

## Table des matières

|           |   |           |
|-----------|---|-----------|
| <b>1</b>  | <b>Introduction</b>                         | <b>6</b>  |
| <b>2</b>  | <b>Licence Creative Commons CC-BY-NC-SA</b> | <b>7</b>  |
| <b>3</b>  | <b>Partie 1 : Un peu d'histoire...</b>      | <b>8</b>  |
| <b>4</b>  | <b>Principes fondateurs</b>                 | <b>9</b>  |
| <b>5</b>  | <b>Origines</b>                             | <b>10</b> |
| <b>6</b>  | <b>Origines ( années 2000 )</b>             | <b>11</b> |
| <b>7</b>  | <b>Progression du projet</b>                | <b>12</b> |
| <b>8</b>  | <b>Communauté</b>                           | <b>13</b> |
| <b>9</b>  | <b>PostgreSQL Core Team</b>                 | <b>14</b> |
| <b>10</b> | <b>Contributeurs</b>                        | <b>15</b> |
| <b>11</b> | <b>Utilisateurs</b>                         | <b>16</b> |
| <b>12</b> | <b>Pourquoi participer ?</b>                | <b>17</b> |
| <b>13</b> | <b>PostgreSQL, un projet mondial</b>        | <b>18</b> |
| <b>14</b> | <b>Sponsors</b>                             | <b>19</b> |
| <b>15</b> | <b>Références</b>                           | <b>20</b> |
| <b>16</b> | <b>References</b>                           | <b>21</b> |
| <b>17</b> | <b>Limites</b>                              | <b>22</b> |
| <b>18</b> | <b>Roadmap</b>                              | <b>23</b> |

|           |  |           |
|-----------|--|-----------|
| <b>19</b> | <b>Partie 2 : Les versions</b>                 | <b>24</b> |
| <b>20</b> | <b>Historique</b>                              | <b>25</b> |
| <b>21</b> | <b>Versions courantes</b>                      | <b>26</b> |
| <b>22</b> | <b>Version 7.4</b>                             | <b>27</b> |
| <b>23</b> | <b>Version 8.0</b>                             | <b>28</b> |
| <b>24</b> | <b>Version 8.1</b>                             | <b>29</b> |
| <b>25</b> | <b>Version 8.2</b>                             | <b>30</b> |
| <b>26</b> | <b>Version 8.3</b>                             | <b>31</b> |
| <b>27</b> | <b>Version 8.4</b>                             | <b>32</b> |
| <b>28</b> | <b>Quelle version utiliser ?</b>               | <b>33</b> |
| <b>29</b> | <b>Partie 3 : Tour d'horizon technique</b>     | <b>34</b> |
| <b>30</b> | <b>Caractéristiques</b>                        | <b>35</b> |
| <b>31</b> | <b>Fonctionnalités : cœur</b>                  | <b>36</b> |
| <b>32</b> | <b>Fonctionnalités : développement</b>         | <b>37</b> |
| <b>33</b> | <b>Fonctionnalités : sécurité</b>              | <b>38</b> |
| <b>34</b> | <b>Fonctionnalités : SQL</b>                   | <b>39</b> |
| <b>35</b> | <b>Fonctionnalités : extensibilité</b>         | <b>40</b> |
| <b>36</b> | <b>Conformité SQL</b>                          | <b>41</b> |
| <b>37</b> | <b>ACID</b>                                    | <b>42</b> |
| <b>38</b> | <b>MultiVersion Concurrency Control (MVCC)</b> | <b>43</b> |
| <b>39</b> | <b>Transactions</b>                            | <b>44</b> |

|   |           |
|---|-----------|
| <b>40 Vues</b>                                      | <b>45</b> |
| <b>41 Schémas</b>                                   | <b>46</b> |
| <b>42 Contraintes</b>                               | <b>47</b> |
| <b>43 Triggers</b>                                  | <b>48</b> |
| <b>44 Héritage</b>                                  | <b>49</b> |
| <b>45 Index</b>                                     | <b>50</b> |
| <b>46 Write Ahead Logs, aka WAL</b>                 | <b>51</b> |
| <b>47 Avantages des WAL</b>                         | <b>52</b> |
| <b>48 Point In Time Recovery, aka PITR (1/2)</b>    | <b>53</b> |
| <b>49 Point In Time Recovery, aka PITR (2/2)</b>    | <b>54</b> |
| <b>50 Tablespaces</b>                               | <b>55</b> |
| <b>51 Tablespaces : avantages</b>                   | <b>56</b> |
| <b>52 Outils de la communauté</b>                   | <b>57</b> |
| <b>53 Serveurs</b>                                  | <b>58</b> |
| <b>54 Serveurs francophone</b>                      | <b>59</b> |
| <b>55 Listes de discussions / Listes d'annonces</b> | <b>60</b> |
| <b>56 Forums / IRC</b>                              | <b>61</b> |
| <b>57 Wiki</b>                                      | <b>62</b> |
| <b>58 Dalibo ?</b>                                  | <b>63</b> |
| <b>59 Partie 5 : Les projets satellittes</b>        | <b>64</b> |
| <b>60 pgAdmin</b>                                   | <b>65</b> |

|                      |           |
|----------------------|-----------|
| <b>61 PhpPgAdmin</b> | <b>66</b> |
| <b>62 Slony</b>      | <b>67</b> |
| <b>63 Bucardo</b>    | <b>68</b> |
| <b>64 pgpool</b>     | <b>69</b> |
| <b>65 PgBouncer</b>  | <b>70</b> |
| <b>66 pgFouine</b>   | <b>71</b> |
| <b>67 Munin</b>      | <b>72</b> |
| <b>68 pgsnap</b>     | <b>73</b> |
| <b>69 PostGIS</b>    | <b>74</b> |
| <b>70 Avantages</b>  | <b>75</b> |
| <b>71 Conclusion</b> | <b>76</b> |
| <b>72 Questions</b>  | <b>77</b> |

## Introduction

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

## Licence Creative Commons CC-BY-NC-SA

Cette formation (diapositives, manuels et travaux pratiques) est sous licence **CC-BY-NC-SA**.

Vous êtes libres de redistribuer et/ou modifier cette création selon les conditions suivantes :

- Paternité
- Pas d'utilisation commerciale
- Partage des conditions initiales à l'identique

Vous devez citer le nom de l'auteur original de la manière indiquée par l'auteur de l'œuvre ou le titulaire des droits qui vous confère cette autorisation (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'œuvre).

Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.

Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

À chaque réutilisation ou distribution de cette création, vous devez faire apparaître clairement au public les conditions contractuelles de sa mise à disposition. La meilleure manière de les indiquer est un lien vers cette page web.

Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits sur cette œuvre.

Rien dans ce contrat ne diminue ou ne restreint le droit moral de l'auteur ou des auteurs.

Le texte complet de la licence est disponible à cette adresse :

<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/legalcode>

## 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

<http://www.postgresql.jp>

- 2004 : 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.

<http://www.spi-inc.org>

- 2007 : Communauté italienne

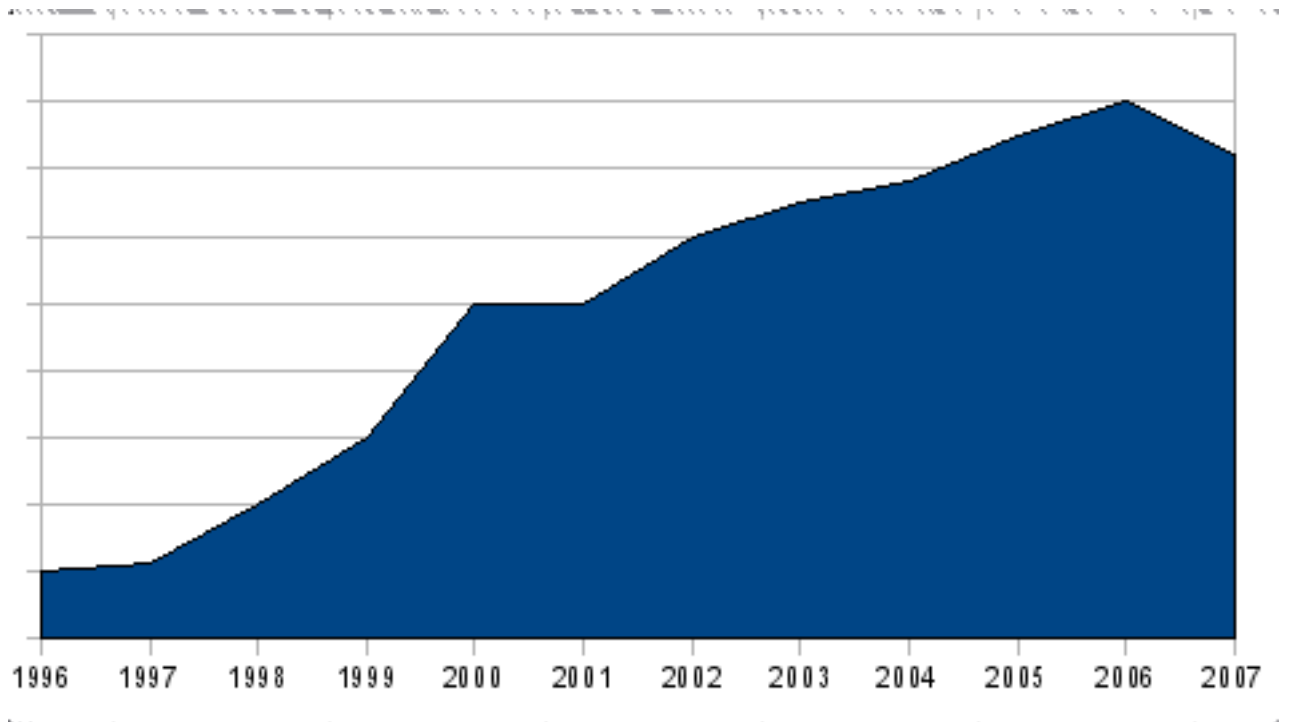
<http://www.itpug.org>

- 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.

## Progression du projet



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

Source :

<http://markmail.blogspot.com/2008/02/postgresql-more-traffic-than-mysql-and.html>

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

<http://www.vimeo.com/1081680>

## Communauté

La communauté PostgreSQL est structurée en trois cercles concentriques :

- 1er cercle : Les développeurs ( "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.

- 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'utilise.

## PostgreSQL Core Team



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

Détails sur quelques membres actuels ou passés de la *core team*

- **Josh Berkus**, ancien de Sun Microsystems, San Francisco (USA)
- **Peter Eisentraut**, Sun Microsystems, 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**, Afiliis 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\\_\(université\)](http://fr.wikipedia.org/wiki/Hacker_(université))

## Contributeurs



Actuellement, PostgreSQL compte une centaine de « contributeurs » qui se répartissent les tâches 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.

## Utilisateurs

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

Il est impossible de connaître précisément le nombre d'utilisateurs 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
- Traduire la documentation
- Participer aux forums d'entre-aide
- 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 manière 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.

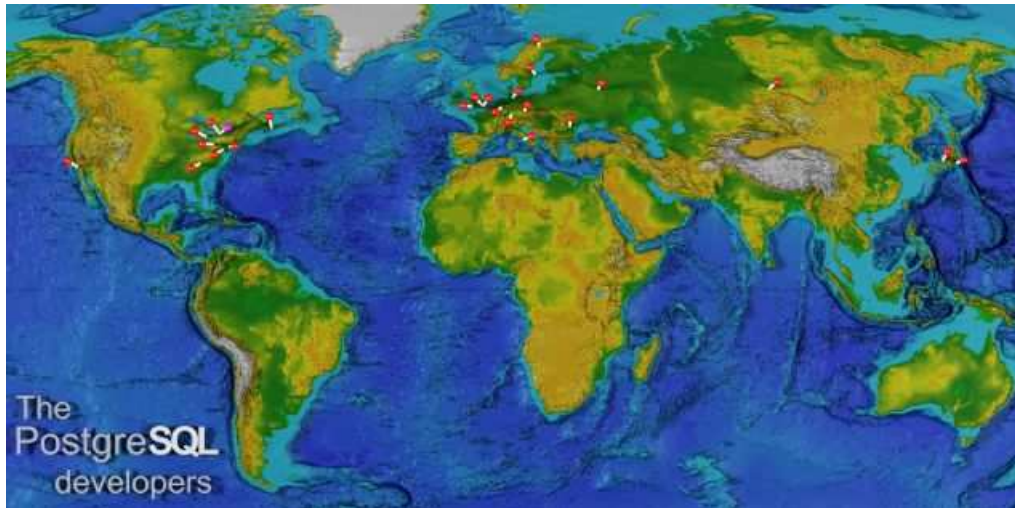
- Traduire et répondre aux questions

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'approfondir ses compétences.

- 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.

## 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 francophone.
- La communauté hispanophone est naissante
- Les développeurs du noyau ( "core hackers" ) vivent en Europe ou en Amérique du Nord

## Sponsors

- Sun Microsystems (Peter Eisentraut)
- Fujitsu
- Red Hat (Tom Lane)
- Skype ( projets 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.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>

## Références

- Yahoo
- Météo France
- RATP

## References

- Yahoo!

Le 21 mai 2008, à l'occasion du rendez-vous annuel PostgreSQL Conference, 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

Pour cela, 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://www.postgresqlfr.org/?q=node/1648>

## Limites

| <b>Limites</b>                    | <b>Valeurs</b> |
|-----------------------------------|----------------|
| 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)

<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*
- Patches en attente pour la version courante
- Patches en attente pour la version suivante

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.

**TODO list** : <http://www.postgresql.org/docs/faqs.TODO.html>

**Patches de la version courante** : <http://momjian.postgresql.org/cgi-bin/pgpatches>

**Patches de la version suivante** : [http://momjian.postgresql.org/cgi-bin/pgpatches\\_hold](http://momjian.postgresql.org/cgi-bin/pgpatches_hold)

Cependant les choses évoluent et un nouveau processus de développement est actuellement en cours d'expérimentation pour la version 8.4.x

Les développeurs principaux tente d'améliorer la transparence et de faciliter l'ajout de patches. Vous pouvez désormais suivre l'évolution des développement en cours sur le wiki officiel :

[http://wiki.postgresql.org/wiki/Development\\_information](http://wiki.postgresql.org/wiki/Development_information)

## Partie 2 : Les versions

- De 7.4 à 8.3
- 8.4 ?
- Quelle version choisir ?



## Historique

- 1996 : v1.0
- 1997 : v6.0
- 1998 : v7.0
- 2003 : v7.4
- 2005 : v8.0 et v8.1
- fin 2006 : v8.2
- février 2008 : v8.3
- mars 2009 : v8.4

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 (mai 2008) :

- version 7.4.19
- version 8.0.15
- version 8.1.11
- version 8.2.7
- version 8.3.1

La philosophie générale des développeurs du coeur 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 une fois par an.

Le support de la version 7.3 a été arrêté au début de l'année 2008.

## Version 7.4

- fin 2003
- « 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

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

## Version 8.0

- début 2005
  - Disponible en natif pour Windows<sup>©</sup><sup>®</sup> (NT/2000/XP/2003)  
Le port natif Windows de la version 8.0 est considéré comme une version beta dudit port.
  - Tablespaces
  - Savepoints
  - PITR
  - Améliorations de CHECKPOINT et VACUUM
- Attention!** Sous Windows<sup>©</sup><sup>®</sup>, PostgreSQL 8.0 nécessite 4 Go de mémoire RAM pour environ 50 utilisateurs concurrents.

## Version 8.1

- fin 2005
- Rôles
- Paramètres IN/OUT dans les fonctions
- Two-Phase Commit
- Autovacuum intégré
- Meilleures performances sur les machines multi-processeurs
- Meilleur partitionnement 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 *patches* ont été reversés au projet PostgreSQL, qui les inclut désormais en standard.

## Version 8.2

- fin 2006
- Compatibilité SQL2003
- Meilleur support de Windows<sup>®</sup> et des migrations depuis Oracle<sup>®</sup>
- 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.

## Version 8.3

- début 2008
- 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 de faire une revue et une correction de votre code avant la migration.

## Version 8.4

- 1er mars 2009 ?
- Simplification de l'héritage ?
- Fonctionnalités de data warehousing ?
- Vues modifiables ? tables en Read-Only ?

Un consensus semble se dégager au sein du groupes des développeurs pour respecter un rythme d'une version majeure par an.

Pour suivre l'avancement et la liste des fonctionnalités attendues dans la version 8.4, rendez-vous sur le wiki officiel :

<http://wiki.postgresql.org/wiki/ToDo> :WishlistFor84



## Quelle version utiliser ?

- 7.3 et inférieures : **Migrer immédiatement !**
- 7.4.19 : Mises à jour uniquement
- 8.0.15 : Release majeure
- 8.1.11 : Ajouts de fonctionnalités
- 8.2.7 : Recherche de performances, version Windows stable
- 8.3.1 : Nouveaux développements

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

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

## 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.

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

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](http://wiki.postgresql.org/wiki/PL_Matrix)

## Fonctionnalités : sécurité

- Filtrage IP
  - ⇒ Fichier `pg_hba.conf`
- Authentification
  - ⇒ Connexion par mots de passe chiffrés MD5, `identd`
  - ⇒ Support natif de LDAP
  - ⇒ Support natif de GSSAPI/SSPI
- Chiffrement de la connexion
  - ⇒ Support natif de SSL

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 )

## Fonctionnalités : SQL

- Excellent support du SQL ANSI
- Vues, règles, séquences, jointures, sous-requêtes, etc.
- Héritage
- Triggers, curseurs
- 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 agrégats

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 : ISO/IEC 9075 :2003 ou SQL :2003

Aucun SGBD ne supporte complètement SQL :2003 à ce jour

- PostgreSQL tente de s'en approcher au maximum, au fil des versions
- Une bonne partie de SQL :2003 est supportée (60% de 255 pré-requis), 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 jours en évitant les blocages trop contraignants (verrous sur UPDATE) et par là même apporte de meilleures performances.

MVCC maintient une version de chaque tuple, ainsi :

1. Chaque transaction voit son image de la base (appelée *snapshot*) telle qu'elle était lors du démarrage de la transaction, quoi que fassent les autres transactions.
2. La lecture ne bloque pas l'écriture.
3. L'écriture ne bloque pas la lecture.
4. L'écriture ne bloque les autres écritures que lors de la mise à jour de la **même version d'une ligne**.
5. MVCC a de plus permis la sauvegarde à chaud cohérente.

## 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

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*****
```

- **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](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 les 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) un INSERT, un UPDATE ou un DELETE
- COPY déclenche le trigger INSERT
- Par contre, TRUNCATE ne déclenche pas le trigger DELETE

Cette fonctionnalité a de fortes chances d'être implémentée pour 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).

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

Exemple :

```
-- Table personne
create table personne (
    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);
```

## Index

Les algorithmes suivants sont supportés :

- B-tree (par défaut)

- R-tree

R-tree est remplacé par GiST. Le terme reste par compatibilité.

- Hash

- GiST

- GIN (version 8.2)

Pour une indexation "standard" , on utilise en général un arbre B

Les index plus spécifique ( GIN, GIST ) sont spécialisés pour les grands volumes de textes.

Plus d'informations :

⇒ [http://fr.wikipedia.org/wiki/Arbre\\_B](http://fr.wikipedia.org/wiki/Arbre_B)

⇒ <http://en.wikipedia.org/wiki/R-tree>

⇒ [http://fr.wikipedia.org/wiki/Table\\_de\\_hachage](http://fr.wikipedia.org/wiki/Table_de_hachage)

⇒ <http://docs.postgresqlfr.org/8.3/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 WALs 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 transaction au lieu de potentiellement plusieurs sur les fichiers de données
- Le fichier de transaction 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 :

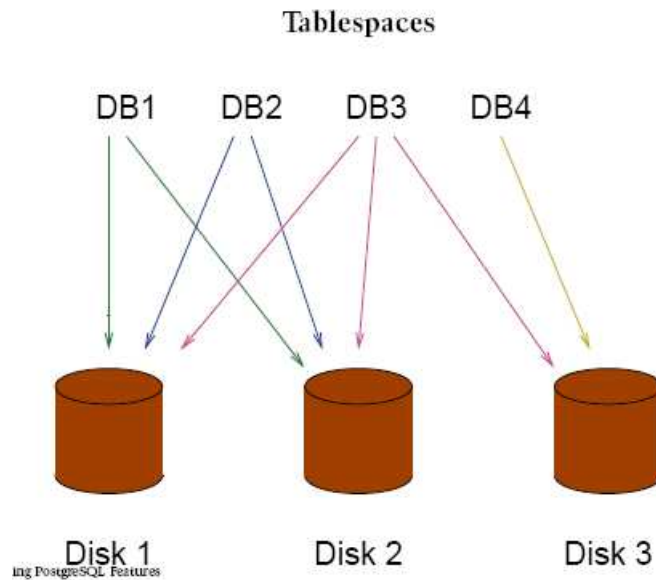
- Avant PostgreSQL 8 : restauration *dump* ou *fichiers* ou réplication
- À partir de PostgreSQL 8 : PITR

## 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)

## 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
- et 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/>

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.

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 francophone

- Site officiel : <http://www.postgresql.fr/>
- La doc : <http://doc.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

- pgsq-announce
- pgsq-general
- pgsq-admin
- pgsq-sql
- pgsq-fr-generale
- pgsq-performance
- pgsq-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.

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

Avant poser une question, consultez :

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

## Forums / IRC

- Forums français : <http://www.postgresqlfr.org/?q=forum>  
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 de développement
- De la documentation
- Des informations de promotion ('advocacy')

## 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](mailto: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 :** 1.8.2

**Licence :** Artistic License

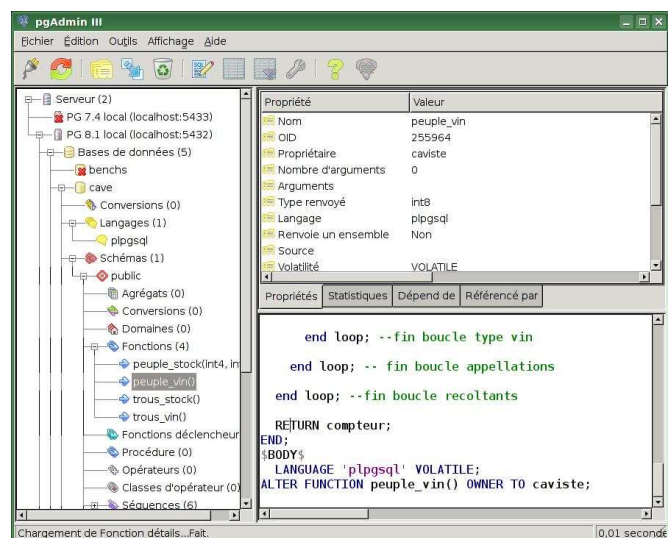
- Gestion graphique de l'administration des bases
- EXPLAIN graphique
- Gestion de Slony

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/pgsql, 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<sup>®</sup>, Mac OSX, GNU/Linux et FreeBSD.



## PhpPgAdmin

**Site officiel :** <http://phppgadmin.sourceforge.net/>

**Version :** 4.3

**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 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 débiter 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 :** 1.2.13

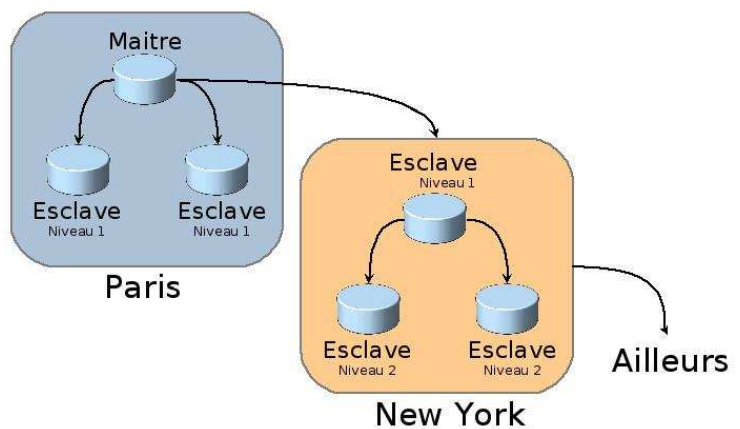
**Licence :** BSD

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

Slony-I est un système de réplication maître/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 :** 3.0.9

**Licence :** BSD

- Réplication Master-Master
- 2 noeuds uniquement

Bucardo est un système de réplication asynchrone pour PostgreSQL qui permet des opérations multi-maitre et maitre-esclave. Il a été développé 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 connaît des problèmes, pgpool redirigera automatiquement les connexions vers un autre serveur.

## PgBouncer

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

**Version :** 1.1.2

**Licence :** BSD

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

[http://dalibo.org/IMG/html/spooler\\_connexions.html](http://dalibo.org/IMG/html/spooler_connexions.html)

## pgFouine

**Site officiel :** <http://pgfouine.projects.postgresql.org>

**Version :** 1.0

**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 accélérer 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 :** 1.3.4

**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.dalibo.org>.

**Version :** 0.4

**Licence :** BSD

- « Photographie » de votre SGBD à un instant t
- Très léger
- Très pratique
- Développé 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 :** 1.3.3

**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 (S/G), 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 Refrations 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 de requêtes

## Conclusion

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

## Questions

N'hésitez pas, c'est le moment !