Module A1

Historique & communauté

24.04

Table des matières

Sur ce document

Formation	Module A1
Titre	Historique & communauté
Révision	24.04
PDF	https://dali.bo/a1_pdf
EPUB	https://dali.bo/a1_epub
HTML	https://dali.bo/a1_html
Slides	https://dali.bo/a1_slides

Vous trouverez en ligne les différentes versions complètes de ce document.

Chers lectrices & lecteurs,

Nos formations PostgreSQL sont issues de nombreuses années d'études, d'expérience de terrain et de passion pour les logiciels libres. Pour Dalibo, l'utilisation de PostgreSQL n'est pas une marque d'opportunisme commercial, mais l'expression d'un engagement de longue date. Le choix de l'Open Source est aussi le choix de l'implication dans la communauté du logiciel.

Au-delà du contenu technique en lui-même, notre intention est de transmettre les valeurs qui animent et unissent les développeurs de PostgreSQL depuis toujours : partage, ouverture, transparence, créativité, dynamisme… Le but premier de nos formations est de vous aider à mieux exploiter toute la puissance de PostgreSQL mais nous espérons également qu'elles vous inciteront à devenir un membre actif de la communauté en partageant à votre tour le savoir-faire que vous aurez acquis avec nous.

Nous mettons un point d'honneur à maintenir nos manuels à jour, avec des informations précises et des exemples détaillés. Toutefois malgré nos efforts et nos multiples relectures, il est probable que ce document contienne des oublis, des coquilles, des imprécisions ou des erreurs. Si vous constatez un souci, n'hésitez pas à le signaler via l'adresse formation@dalibo.com[1] !

À propos de DALIBO

DALIBO est le spécialiste français de PostgreSQL. Nous proposons du support, de la formation et du conseil depuis 2005.

Retrouvez toutes nos formations sur https://dalibo.com/formations

[1] mailto:formation@dalibo.com

Remerciements

Ce manuel de formation est une aventure collective qui se transmet au sein de notre société depuis des années. Nous remercions chaleureusement ici toutes les personnes qui ont contribué directement ou indirectement à cet ouvrage, notamment :

Jean-Paul Argudo, Alexandre Anriot, Carole Arnaud, Alexandre Baron, David Bidoc, Sharon Bonan, Franck Boudehen, Arnaud Bruniquel, Pierrick Chovelon, Damien Clochard, Christophe Courtois, Marc Cousin, Gilles Darold, Jehan-Guillaume de Rorthais, Ronan Dunklau, Vik Fearing, Stefan Fercot, Pierre Giraud, Nicolas Gollet, Dimitri Fontaine, Florent Jardin, Virginie Jourdan, Luc Lamarle, Denis Laxalde, Guillaume Lelarge, Alain Lesage, Benoit Lobréau, Jean-Louis Louër, Thibaut Madelaine, Adrien Nayrat, Alexandre Pereira, Flavie Perette, Robin Portigliatti, Thomas Reiss, Maël Rimbault, Julien Rouhaud, Stéphane Schildknecht, Julien Tachoires, Nicolas Thauvin, Be Hai Tran, Christophe Truffier, Cédric Villemain, Thibaud Walkowiak, Frédéric Yhuel.

Forme de ce manuel

Les versions PDF, EPUB ou HTML de ce document sont structurées autour des slides de nos formations. Le texte suivant chaque slide contient le cours et de nombreux détails qui ne peuvent être données à l'oral.

Licence Creative Commons CC-BY-NC-SA

Cela inclut les diapositives, les manuels eux-mêmes et les travaux pratiques. Cette formation peut également contenir quelques images et schémas dont la redistribution est soumise à des licences différentes qui sont alors précisées.

Marques déposées

PostgreSQL® Postgres® et le logo Slonik sont des marques déposées[3] par PostgreSQL Community Association of Canada.

Versions de PostgreSQL couvertes

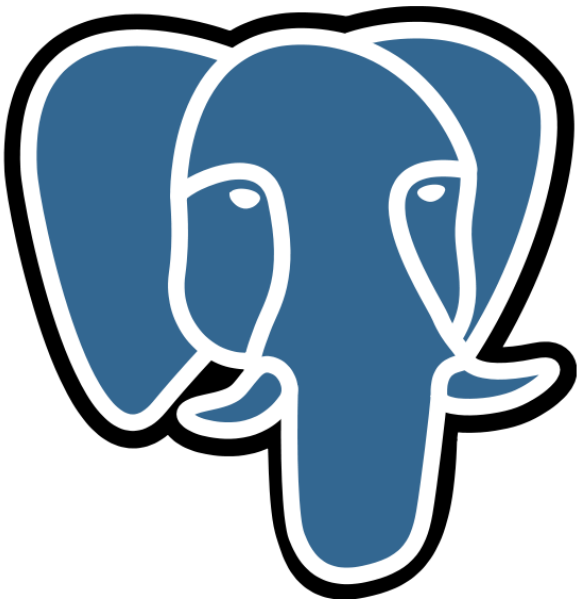
Ce document ne couvre que les versions supportées de PostgreSQL au moment de sa rédaction, soit les versions 12 à 16.

Sur les versions précédentes susceptibles d'être encore rencontrées en production, seuls quelques points très importants sont évoqués, en plus éventuellement de quelques éléments historiques.

Sauf précision contraire, le système d'exploitation utilisé est Linux.

[3] https://www.postgresql.org/about/policies/trademarks/

1/ PostgreSQL : historique & communauté



1.1 PRÉAMBULE

- Quelle histoire !
  - parmi les plus vieux logiciels libres
  - et les plus sophistiqués
- Souvent cité comme exemple
  - qualité du code
  - indépendance des développeurs
  - réactivité de la communauté

L'histoire de PostgreSQL est longue, riche et passionnante. Au côté des projets libres Apache et Linux, PostgreSQL est l'un des plus vieux logiciels libres en activité et fait partie des SGBD les plus sophistiqués à l'heure actuelle.

Au sein des différentes communautés libres, PostgreSQL est souvent cité comme exemple à différents niveaux :

- qualité du code ;
- indépendance des développeurs et gouvernance du projet ;
- réactivité de la communauté ;
- stabilité et puissance du logiciel.

Tous ces atouts font que PostgreSQL est désormais reconnu et adopté par des milliers de grandes sociétés de par le monde.

1.1.1 Au menu

- Origines et historique du projet
- Versions et feuille de route
- Projets satellites
- Sponsors et références
- La communauté

Cette première partie est un tour d'horizon pour découvrir les multiples facettes du système de gestion de base de données libre PostgreSQL.

Les deux premières parties expliquent la genèse du projet et détaillent les différences entre les versions successives du logiciel. PostgreSQL est un des plus vieux logiciels libres ! Comprendre son histoire permet de mieux réaliser le chemin parcouru et les raisons de son succès.

Nous verrons ensuite certains projets satellites et nous listerons plusieurs utilisateurs renommés et cas d'utilisations remarquables.

Enfin, nous terminerons par une découverte de la communauté.

1.2 UN PEU D'HISTOIRE...

- La licence
- L'origine du nom
- Les origines du projet
- Les principes

1.2.1 Licence

- Licence PostgreSQL
  - libre (BSD/MIT)
  - https://www.postgresql.org/about/licence/
- Droit, sans coûts de licence, de :
  - utiliser, copier, modifier, distribuer (et même revendre)
- Reconnue par l'Open Source Initiative
- Utilisée par un grand nombre de projets de l'écosystème

PostgreSQL est distribué sous une licence spécifique, combinant la licence BSD et la licence MIT. Cette licence spécifique est reconnue comme une licence libre par l'Open Source Initiative[1].

Cette licence vous donne le droit de distribuer PostgreSQL, de l'installer, de le modifier… et même de le vendre. Certaines sociétés, comme EnterpriseDB et PostgresPro, produisent leur version propriétaire de PostgreSQL de cette façon.

PostgreSQL n'est pas pour autant complètement gratuit : il peut y avoir des frais et du temps de formation, des projets de migration depuis d'autres bases, ou d'intégration des différents outils périphériques indispensables en production.

Cette licence a ensuite été reprise par de nombreux projets de la communauté : pgAdmin, pgCluu, pgstat, etc.

[1] https://opensource.org/licenses/PostgreSQL

1.2.2 PostgreSQL ?!?!

- 1985 : Michael Stonebraker recode Ingres
- post « ingres » ⇒ post~~in~~gres ⇒ postgres
- postgres ⇒ PostgreSQL

PostgreSQL a une origine universitaire.

L'origine du nom PostgreSQL remonte au système de gestion de base de données Ingres, développé à l'université de Berkeley par Michael Stonebraker. En 1985, il prend la décision de reprendre le développement à partir de zéro et nomme ce nouveau logiciel **Postgres**, comme raccourci de post-Ingres.

En 1995, avec l'ajout du support du langage SQL, Postgres fut renommé **Postgres95** puis **PostgreSQL**.

Aujourd'hui, le nom officiel est « PostgreSQL » (prononcé « post - gresse - Q - L »). Cependant, le nom « Postgres » reste accepté.

Pour aller plus loin :

- Fil de discussion sur les listes de discussion[2] ;
- Article sur le wiki officiel[3].

1.2.3 Principes fondateurs

- Sécurité des données (ACID[4])
- Respect des normes (ISO SQL)
- Portabilité
- Fonctionnalités intéressant le plus grand nombre
- Performances
  - si pas de péril pour les données
- Simplicité du code
- Documentation

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

La sécurité des données est un point essentiel. En premier lieu, un utilisateur doit être certain qu'à partir du moment où il a exécuté l'ordre `COMMIT` d'une transaction, les données modifiées relatives à cette transaction se trouvent bien sur disque et que même un crash ne pourra pas les faire disparaître. PostgreSQL est très attaché à ce concept et fait son possible pour forcer le système d'exploitation à ne pas conserver les données en cache, mais à les écrire sur disque dès l'arrivée d'un `COMMIT`.

L'intégrité des données, et le respect des contraintes fonctionnelles et techniques qui leur sont imposées, doivent également être garanties par le moteur à tout moment, quoi que fasse l'utilisateur. Par exemple, insérer 1000 caractères dans un champ contraint à 200 caractères maximum doit mener à une erreur explicite et non à l'insertion des 200 premiers caractères en oubliant les autres, comme cela s'est vu ailleurs. De même, un champ avec le type `date` ne contiendra jamais un 31 février, et un champ `NOT NULL` ne sera jamais vide. Tout ceci est formalisé par les propriétés (ACID[5]) que possèdent toute bonne base de données relationnelle.

Le respect des normes est un autre principe au cœur du projet. Les développeurs de PostgreSQL cherchent à coller à la norme SQL[6] le plus possible. PostgreSQL n'est pas compatible à cette norme à 100 %, aucun moteur ne l'est, mais il cherche à s'en approcher. Tout nouvel ajout d'une syntaxe ne sera accepté que si la syntaxe de la norme est ajoutée. Des extensions sont acceptées pour différentes raisons (performances, fonctionnalités en avance sur le comité de la norme, facilité de transition d'un moteur de bases de données à un autre) mais si une fonctionnalité existe dans la norme, une syntaxe différente ne peut être acceptée que si la syntaxe de la norme est elle-aussi présente.

La portabilité est importante : PostgreSQL tourne sur l'essentiel des systèmes d'exploitation : Linux (plate-forme à privilégier), macOS, les Unix propriétaires, Windows… Tout est fait pour que cela soit encore le cas dans le futur.

Ajouter des fonctionnalités est évidemment l'un des buts des développeurs de PostgreSQL. Cependant, comme il s'agit d'un projet libre, rien n'empêche un développeur de proposer une fonctionnalité, de la faire intégrer, puis de disparaître laissant aux autres la responsabilité de la corriger le cas échéant. Comme le nombre de développeurs de PostgreSQL est restreint, il est important que les fonctionnalités ajoutées soient vraiment utiles au plus grand nombre pour justifier le coût potentiel du débogage. Donc ne sont ajoutées dans PostgreSQL que ce qui est vraiment le cœur du moteur de bases de données et que ce qui sera utilisé vraiment par le plus grand nombre. Une fonctionnalité qui ne sert que une à deux personnes aura très peu de chances d'être intégrée. (Le système des extensions offre une élégante solution aux problèmes très spécifiques.)

Les performances ne viennent qu'après tout ça. En effet, rien ne sert d'avoir une modification du code qui permet de gagner énormément en performances si cela met en péril le stockage des données. Cependant, les performances de PostgreSQL sont excellentes et le moteur permet d'opérer des centaines de tables, des milliards de lignes pour plusieurs téraoctets de données, sur une seule instance, pour peu que la configuration matérielle soit correctement dimensionnée.

La simplicité du code est un point important. Le code est relu scrupuleusement par différents contributeurs pour s'assurer qu'il est facile à lire et à comprendre. En effet, cela facilitera le débogage plus tard si cela devient nécessaire.

[5] https://dali.bo/a2_html#ACID
[6] https://fr.wikipedia.org/wiki/Structured_Query_Language

Enfin, la documentation est là-aussi un point essentiel dans l'admission d'une nouvelle fonctionnalité. En effet, sans documentation, peu de personnes pourront connaître cette fonctionnalité. Très peu sauront exactement ce qu'elle est supposée faire, et il serait donc très difficile de déduire si un problème particulier est un manque actuel de cette fonctionnalité ou un bug.

Tous ces points sont vérifiés à chaque relecture d'un patch (nouvelle fonctionnalité ou correction).

1.2.4 Origines

- Années 1970 : Michael Stonebraker développe **Ingres** à Berkeley
- 1985 : **Postgres** succède à Ingres
- 1995 : Ajout du langage SQL
- 1996 : Libération du code : Postgres devient **PostgreSQL**
- 1996 : Création du PostgreSQL Global Development Group

L'histoire de PostgreSQL remonte au système de gestion de base de données Ingres[7], développé dès 1973 à l'Université de Berkeley (Californie) par Michael Stonebraker[8].

Lorsque ce dernier décide en 1985 de recommencer le développement de zéro, il nomme le logiciel Postgres, comme raccourci de post-Ingres. Des versions commencent à être diffusées en 1989, puis commercialisées.

Postgres utilise alors un langage dérivé de QUEL[9], hérité d'Ingres, nommé POSTQUEL[10]. En 1995, lors du remplacement par le langage SQL par Andrew Yu and Jolly Chen, deux étudiants de Berkeley, Postgres est renommé Postgres95.

En 1996, Bruce Momijan et Marc Fournier convainquent l'Université de Berkeley de libérer complètement le code source. Est alors fondé le PGDG (*PostgreSQL Development Group*), entité informelle — encore aujourd'hui — regroupant l'ensemble des contributeurs. Le développement continue donc hors tutelle académique (et sans son fondateur historique Michael Stonebraker) : PostgreSQL 6.0 est publié début 1997.

Plus d'informations :

- Page associée sur le site officiel[11].

[7] https://en.wikipedia.org/wiki/Ingres_(database)
[8] https://en.wikipedia.org/wiki/Michael_Stonebraker
[9] https://en.wikipedia.org/wiki/QUEL_query_languages
[10] La trace se retrouve encore dans le nom de la librairie C pour les clients, la **libpq**.

1.2.5 Apparition de la communauté internationale

- ~ 2000: Communauté japonaise (JPUG)
- 2004 : PostgreSQLFr
- 2006 : SPI
- 2007 : Communauté italienne
- 2008 : PostgreSQL Europe et US
- 2009 : Boom des PGDay
- 2011 : Postgres Community Association of Canada
- 2017 : Community Guidelines
- …et ça continue

Les années 2000 voient l'apparition de communautés locales organisées autour d'association ou de manière informelle. Chaque communauté organise la promotion, la diffusion d'information et l'entraide à son propre niveau.

En 2000 apparaît la communauté japonaise (JPUG). Elle dispose déjà d'un grand groupe, capable de réaliser des conférences chaque année, d'éditer des livres et des magazines. Elle compte, au dernier recensement connu, plus de 3000 membres.

En 2004 naît l'association française (loi 1901) appelée PostgreSQL Fr. Cette association a pour but de fournir un cadre légal pour pouvoir participer à certains événements comme Solutions Linux, les RMLL ou d'en organiser comme le pgDay.fr (qui a déjà eu lieu à Toulouse, Nantes, Lyon, Toulon, Marseille). Elle permet aussi de récolter des fonds pour aider à la promotion de PostgreSQL.

En 2006, le PGDG intègre Software in the Public Interest, Inc.(SPI)[12], une organisation à but non lucratif chargée de collecter et redistribuer des financements. Elle a été créée à l'initiative de Debian et dispose aussi de membres comme LibreOffice.org.

Jusque là, les événements liés à PostgreSQL apparaissaient plutôt en marge de manifestations, congrès, réunions… plus généralistes. 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. Des « sur-groupes » apparaissent aussi pour aider les groupes locaux : PGUS rassemble les différents groupes américains, plutôt organisés géographiquement, par État ou grande ville. De même, en Europe, est fondée PostgreSQL Europe, association chargée d'aider les utilisateurs de PostgreSQL souhaitant mettre en place des événements. Son principal travail est l'organisation d'un événement majeur en Europe tous les ans : pgconf.eu[13], d'abord à Paris en 2009, puis dans divers pays d'Europe jusque Milan en 2019. Cependant, elle aide aussi les communautés allemande, française et suédoise à monter leur propre événement (respectivement PGConf.DE[14], pgDay Paris[15]

[12] https://fr.wikipedia.org/wiki/Software_in_the_Public_Interest
[13] https://pgconf.eu/
[14] https://pgconf.de/
[15] https://pgday.paris/

et Nordic PGday[16]).

Dès 2010, nous dénombrons plus d'une conférence par mois consacrée uniquement à PostgreSQL dans le monde. Ce mouvement n'est pas prêt de s'arrêter :

- communauté japonaise[17] ;
- communauté francophone[18] ;
- communauté italienne[19] ;
- communauté européenne[20] ;
- communauté américaine (États-Unis)[21].

En 2011, l'association Postgres Community Association of Canada voit le jour[22]. Elle est créée par quelques membres de la *Core Team* pour gérer le nom déposé PostgreSQL, le logo, le nom de domaine sur Internet, etc.

Vu l'émergence de nombreuses communautés internationales, la communauté a décidé d'écrire quelques règles pour ces communautés. Il s'agit des *Community Guidelines*, apparues en 2017, et disponibles sur le site officiel[23].

1.2.6 Progression du code

- 1,6 millions de lignes
  - dont 1/4 de commentaires
  - le reste surtout en C
- Nombres de commit par mois :

[16] https://nordicpgday.org/
[17] https://www.postgresql.jp/
[18] https://www.postgresql.fr/
[19] https://www.itpug.org/
[20] https://www.postgresql.eu/
[21] https://www.postgresql.us/
[22] https://www.postgresql.org/message-id/4DC440BE.5040104%40agliodbs.com%3E
[23] https://www.postgresql.org/community/recognition/

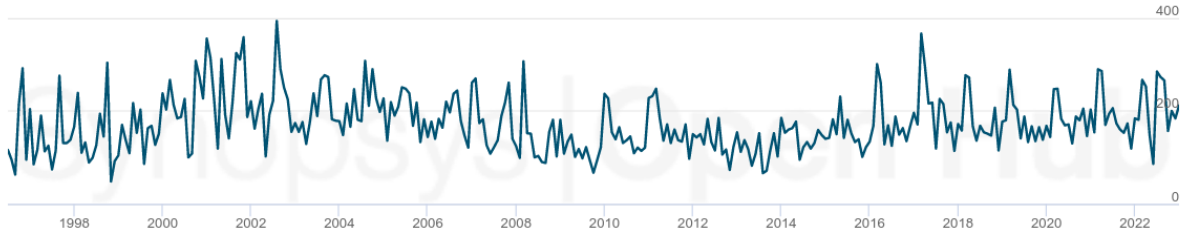


Figure 1/ .1: Évolution du nombre de commit dans le dépôt PostgreSQL

Le dépôt principal de PostgreSQL a été un dépôt CVS, passé depuis à git[24]. Il est en accès public en lecture.

Le graphe ci-dessus (source[25]) représente l'évolution du nombre de commit dans les sources de PostgreSQL. L'activité ne se dément pas. Le plus intéressant est certainement de noter que l'évolution est constante. Il n'y a pas de gros pic, ni dans un sens, ni dans l'autre.

Début 2023, PostgreSQL est composé d'1,6 millions de lignes de code, dont un quart de commentaires. Ce ratio montre que le code est très commenté, très documenté. Ceci fait qu'il est facile à lire, et donc pratique à déboguer. Et le ratio ne change pas au fil des ans. Le code est essentiellement en C, pour environ 200 développeurs actifs, à environ 200 commits par mois ces dernières années.

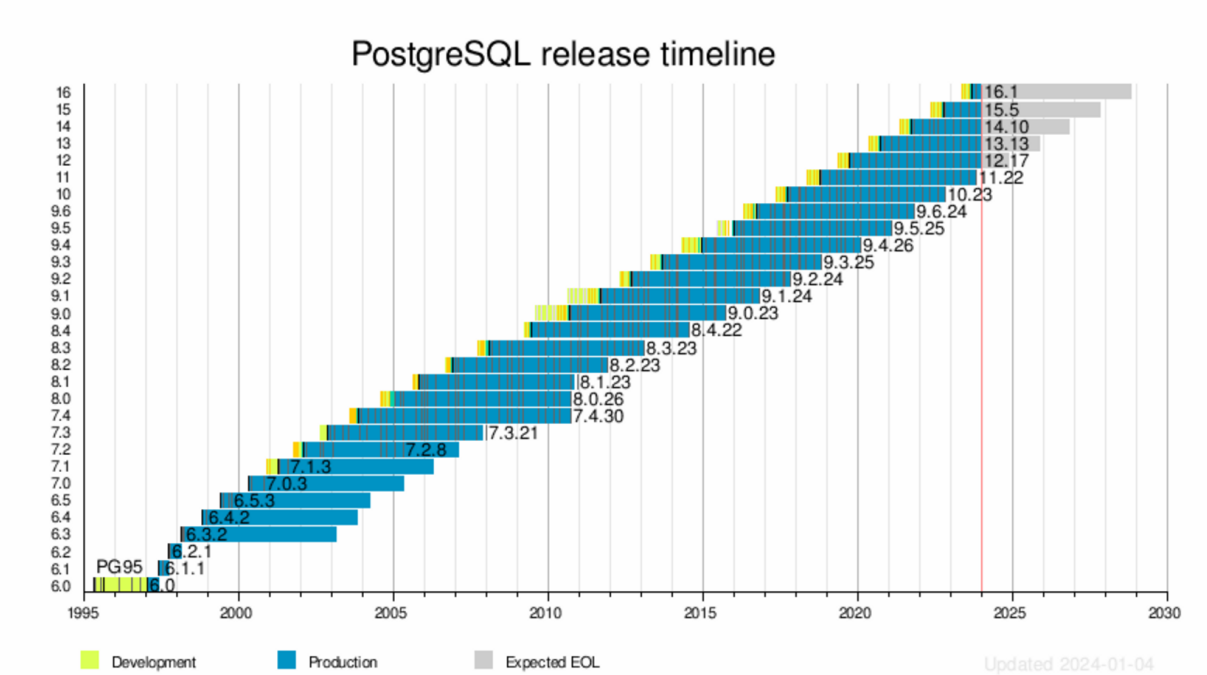
[24] https://git.postgresql.org/
[25] https://www.openhub.net/p/postgres/analyses/latest/

1.3 LES VERSIONS DE POSTGRESQL

Quelle version utiliser ?

- Historique
- Numérotation
- Mises à jour mineures et majeures
- Les versions courantes
- Quelle version en production ?
- Les forks & dérivés

1.3.1 Historique



Sources : page Wikipédia de PostgreSQL[26] et PostgreSQL Versioning Policy[27]

[26] https://en.wikipedia.org/wiki/PostgreSQL
[27] https://www.postgresql.org/support/versioning/

1.3.2 Versions & fonctionnalités

- 1996 : v6.0 -> première version publiée
- 2003 : v7.4 -> première version *réellement* stable
- 2005 : v8.0 -> arrivée sur Windows
- 2008 : v8.3 -> performances et fonctionnalités, organisation (commitfests)
- 2010 : v9.0 -> réplication physique
- 2016 : v9.6 -> parallélisation
- 2017 : v10 -> réplication logique, partitionnement déclaratif
- 2023 : v16 -> performances, fonctionnalités, administration…

La version 7.4 est la première version réellement stable. La gestion des journaux de transactions a été nettement améliorée, et de nombreuses optimisations ont été apportées au moteur.

La version 8.0 marque l'entrée tant attendue de PostgreSQL dans le marché des SGDB de haut niveau, en apportant des fonctionnalités telles que les tablespaces, les routines stockées en Java, le *Point In Time Recovery*, ainsi qu'une version native pour Windows.

La version 8.3 se focalise sur les performances et les nouvelles fonctionnalités. C'est aussi la version qui a causé un changement important dans l'organisation du développement pour encourager les contributions : gestion des commitfests, création de l'outil web associé, etc.

Les versions 9.x sont axées réplication physique. La 9.0 intègre un système de réplication asynchrone asymétrique. La version 9.1 ajoute une réplication synchrone et améliore de nombreux points sur la réplication (notamment pour la partie administration et supervision). La version 9.2 apporte la réplication en cascade. La 9.3 et la 9.4 ajoutent quelques améliorations supplémentaires. La version 9.4 intègre surtout les premières briques pour l'intégration de la réplication logique dans PostgreSQL. La version 9.6 apporte la parallélisation, ce qui était attendu par de nombreux utilisateurs.

La version 10 propose beaucoup de nouveautés, comme une amélioration nette de la parallélisation et du partitionnement (le partitionnement déclaratif complète l'ancien partitionnement par héritage), mais aussi l'ajout de la réplication logique.

Les améliorations des versions 11 à 16 sont plus incrémentales, et portent sur tous les plans. Le partitionnement déclaratif et la réplication logique sont progressivement améliorés, en performances comme en facilité de développement. Les performances s'améliorent encore grâce à la compilation *Just In Time*, la parallélisation de plus en plus d'opérations, les index couvrants, l'affinement des statistiques. La facilité d'administration s'améliore : nouvelles vues système, rôles supplémentaires pour réduire l'utilisation du superutilisateur, outillage de réplication, activation des sommes de contrôle sur une instance existante.

Il est toujours possible de télécharger les sources depuis la version 1.0 jusqu'à la version courante sur postgresql.org[28].

[28] https://www.postgresql.org/ftp/source/

1.3.3 Numérotation

- Version récentes (10+)
  - X : version majeure (10, 11, … 16)
  - X.Y : version mineure (14.8, 15.3)
- Avant la version 10 (toutes périmées !)
  - X.Y : version majeure (8.4, 9.6)
  - X.Y.Z : version mineure (9.6.24)

Une version majeure apporte de nouvelles fonctionnalités, des changements de comportement, etc. Une version majeure sort généralement tous les ans à l'automne. Une migration majeure peut se faire directement depuis n'importe quelle version précédente. Le numéro est incrémenté chaque année (version 12 en 2019, version 16 en 2023).

Une version mineure ne comporte que des corrections de bugs ou de failles de sécurité. Les publications de versions mineures sont plus fréquentes que celles de versions majeures, avec un rythme de sortie trimestriel, sauf bug majeur ou faille de sécurité. Chaque bug est corrigé dans toutes les versions stables actuellement maintenues par le projet. Le numéro d'une version mineure porte deux chiffres. Par exemple, en mai 2023 sont sorties les versions 15.3, 14.8, 13.11, 12.15 et 11.20.

Avant la version 10, les versions majeures annuelles portaient deux chiffres : 9.0 en 2010, 9.6 en 2016. Les mineures avaient un numéro de plus (par exemple 9.6.24). Cela a entraîné quelques confusions, d'où le changement de numérotation. Il va sans dire que ces versions sont totalement périmées et ne sont plus supportées, mais beaucoup continuent de fonctionner.

1.3.4 Mises à jour mineure

De M.m à M.m+n :

- En général chaque trimestre
- Et sans souci
  - *Release notes*
  - tests
  - mise à jour des binaires
  - redémarrage

Une mise à jour mineure consiste à mettre à jour vers une nouvelle version de la même branche majeure, par exemple de 14.8 à 14.9, ou de 16.0 à 16.1 (mais pas d'une version 14.x à une version 16.x). Les mises à jour des versions mineures sont cumulatives : vous pouvez mettre à jour une instance 15.0 en version 15.5 sans passer par les versions 15.1 à 15.4 intermédiaires.

En général, les mises à jour mineures se font sans souci et ne nécessitent que le remplacement des binaires et un redémarrage (et donc une courte interruption). Les fichiers de données conservent le même format. Des opérations supplémentaires sont possibles mais rarissimes. Mais comme pour toute mise à jour, il convient d'être prudent sur d'éventuels effets de bord. En particulier, il faudra lire les *Release Notes* et, si possible, effectuer les tests ailleurs qu'en production.

1.3.5 Versions courantes

- 1 version majeure par an
  - maintenue 5 ans
- Dernières mises à jour mineures[29] (au 9/11/2023) :
  - version 11.22[30] (dernière !)
  - version 12.17[31]
  - version 13.13[32]
  - version 14.10[33]
  - version 15.5[34]
  - version 16.1[35]
- Prochaine sortie de versions mineures prévue : 8 février 2024

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, en pratique une version stable majeure paraît tous les ans, habituellement à l'automne. Pour ne pas sacrifier la qualité des versions, toute fonctionnalité supposée insuffisamment stable est repoussée à la version suivante. Il est déjà arrivé que la sortie de la version majeure soit repoussée à cause de bugs inacceptables.

La tendance actuelle est de garantir un support pour chaque version majeure pendant une durée minimale de 5 ans. Ainsi ne sont plus supportées les versions 10 depuis novembre 2022 et 11 depuis novembre 2023. Il n'y aura pour elles plus aucune mise à jour mineure, donc plus de correction de bug ou de faille de sécurité. Le support de la dernière version majeure, la 16, devrait durer jusqu'en 2028.

Pour plus de détails :

- Politique de versionnement[36] ;
- Dates prévues des futures versions[37].

1.3.6 Versions 9.4 à 11

- `jsonb`
- Row Level Security
- Index BRIN, bloom
- Fonctions OLAP
- Parallélisation
- SQL/MED : accès distants
- Réplication logique
- Partitionnement déclaratif
- Réduction des inconvénients de MVCC
- JIT
- Index couvrants

Ces versions ne sont plus supportées !

La version 9.4 (décembre 2014) a apporté le type `jsonb`, binaire, facilitant la manipulation des objets en JSON.

La 9.5 parue en janvier 2016 apportait notamment les index BRIN et des possibilités OLAP plus avancées que `GROUP BY`. Pour plus de détails :

- Page officielle des nouveautés de la version 9.5[38] ;
- Workshop Dalibo sur la version 9.5[39].

En 9.6, la nouvelle fonctionnalité majeure est certainement la parallélisation de certaines parties de l'exécution d'une requête. Le `VACUUM FREEZE` devient beaucoup moins gênant.

- Page officielle des nouveautés de la version 9.6[40] ;
- Workshop Dalibo sur la version 9.6[41].

En version 10, les fonctionnalités majeures sont l'intégration de la réplication logique et le partitionnement déclaratif, longtemps attendus, améliorés dans les versions suivantes. Sont notables aussi les tables de transition ou les améliorations sur la parallélisation.

La version 10 a aussi été l'occasion de renommer plusieurs répertoires et fonctions système, et même des outils. Attention donc si vous rencontrez des requêtes ou des scripts adaptés aux versions précédentes. Entre autres :

- le répertoire `pg_xlog` est devenu `pg_wal` ;
- le répertoire `pg_clog` est devenu `pg_xact` ;
- dans les noms de fonctions, `xlog` a été remplacé par `wal` (par exemple `pg_switch_xlog` est devenue `pg_switch_wal`) ;
- toujours dans les fonctions, `location` a été remplacé par `lsn`.

Pour plus de détails :

- Page officielle des nouveautés de la version 10[42] ;
- Workshop Dalibo sur la version 10[43].

La version 11 (octobre 2018) améliore le partitionnement de la version 10, le parallélisme, la réplication logique… et de nombreux autres points. Elle comprend aussi une première version du JIT (*Just In Time compilation*) pour accélérer les requêtes les plus lourdes en CPU, ou encore les index couvrants.

Pour plus de détails, voir notre workshop sur la version 11[44].

[38] https://wiki.postgresql.org/wiki/What%27s_new_in_PostgreSQL_9.5
[39] https://kb.dalibo.com/conferences/nouveautes_de_postgresql_9.5
[40] https://wiki.postgresql.org/wiki/NewIn96
[41] https://github.com/dalibo/workshops/tree/master/fr
[42] https://wiki.postgresql.org/wiki/New_in_postgres_10
[43] https://dali.bo/workshop10_pdf
[44] https://dali.bo/workshop11_pdf

1.3.7 Version 12

- Octobre 2019 - Novembre 2024
- Amélioration du partitionnement déclaratif
- Amélioration des performances
  - sur la gestion des index
  - sur les CTE (option MATERIALIZED)
- Colonnes générées
- Nouvelles vues de visualisation de la progression des commandes
- Refonte de la configuration de la réplication

La version 12 est sortie le 3 octobre 2019. Elle améliore de nouveau le partitionnement et elle fait surtout un grand pas au niveau des performances et de la supervision.

Le fichier `recovery.conf` (pour la réplication et les restaurations physiques) disparaît. Il est maintenant intégré au fichier `postgresql.conf`. Une source fréquente de ralentissement disparaît, avec l'intégration des CTE (clauses `WITH`) dans la requête principale. Des colonnes d'une table peuvent être automatiquement générées à partir d'autres colonnes.

Pour plus de détails, voir notre workshop sur la version 12[45].

1.3.8 Version 13

- Septembre 2020 - Septembre 2025
- Améliorations :
  - partitionnement déclaratif
  - réplication logique
- Amélioration des performances :
  - index B-tree, objet statistique, tri et agrégat
- Amélioration de l'autovacuum et du VACUUM :
  - gestion complète des tables en insertion seule
  - traitement parallélisé des index lors d'un VACUUM
- Amélioration des sauvegardes :
  - génération d'un fichier manifeste, outil pg_verifybackup
- Nouvelles vues de progression de commandes :
  - pg_stat_progress_basebackup, pg_stat_progress_analyze

La version 13 est sortie le 24 septembre 2020. Elle est remplie de nombreuses petites améliorations sur différents domaines : partitionnement déclaratif, autovacuum, sauvegarde, etc. Les performances sont aussi améliorées grâce à un gros travail sur l'optimiseur, ou la réduction notable de la taille de certains index.

Pour plus de détails, voir notre workshop sur la version 13[46].

1.3.9 Version 14

- Septembre 2021 - Novembre 2026
- Nouvelles vues système & améliorations
  - `pg_stat_progress_copy`, `pg_stat_wal`, `pg_lock.waitstart`, `query_id`…
- Lecture asynchrone des tables distantes
- Paramétrage par défaut adapté aux machines plus récentes
- Améliorations diverses :
  - réplications physique et logique
  - quelques facilités de syntaxe (triggers, tableaux en PL/pgSQL)
- Performances :
  - connexions en lecture seule plus nombreuses
  - index…

La version 14 est remplie de nombreuses petites améliorations sur différents domaines listés ci-dessus.

Pour plus de détails, voir notre workshop sur la version 14[47].

1.3.10 Version 15

- Octobre 2022 - Novembre 2027
- Nombreuses améliorations incrémentales
  - dont en réplication logique
- Commande `MERGE`
- Performances :
  - `DISTINCT` parallélisable
  - pg_dump & sauvegardes, recovery, partitionnement
- Changements notables :
  - `public` n'est plus accessible en écriture à tous
  - sauvegarde PITR exclusive disparaît

La version 15 est également une mise à jour sans grande nouveauté fracassante, mais contenant de très nombreuses améliorations et optimisations sur de nombreux plans, comme par exemple la commande `MERGE` ou l'accélération du *recovery* sur une reprise de restauration.

Signalons deux changements de comportement importants : pour renforcer la sécurité, le schéma `public` n'est plus accessible en écriture par défaut à tous les utilisateurs ; et la sauvegarde physique en mode exclusif n'est plus disponible.

Pour plus de détails, voir notre workshop sur la version 15[48].

1.3.11 Version 16

- Septembre 2023 - Novembre 2028
- Plus de tris incrémentaux (`DISTINCT` …)
- Réplication logique depuis un secondaire
- Expressions régulières dans `pg_hba.conf`
- Vues systèmes améliorées : `pg_stat_io` …
- Compression lz4 ou zstd pour pg_dump
- Optimisation et améliorations diverses (parallélisation…)

La version 16 est parue le 14 septembre 2023. Sa version 16.1 est considérée comme bonne pour la production. Là encore, les améliorations sont incrémentales.

On notera la possibilité de rajouter des expressions régulières dans `pg_hba.conf` pour faciliter la gestion des accès. En réplication logique, un abonnement peut se faire auprès d'un serveur secondaire. La réplication logique peut devenir parallélisable. `pg_dump` acquiert des algorithmes de compression plus modernes. Le travail de parallélisation de nouveaux nœuds se poursuit. Une nouvelle vue de suivi des entrées-sorties apparaît : `pg_stat_io`.

Pour plus de détails :

- workshop Dalibo sur la version 16[49] ;
- blog Dalibo[50] ;
- présentation de Magnus Hagander au PGDay UK 2023[51]

1.3.12 Petit résumé

- Versions 7.x :
  - fondations
  - durabilité
- Versions 8.x :
  - fonctionnalités
  - performances
- Versions 9.x :
  - réplication physique
  - extensibilité
- Versions 10 à 16 :
  - réplication logique
  - parallélisation
  - partitionnement
- … et la 17 est en développement

Si nous essayons de voir cela avec de grosses mailles, les développements des versions 7 ciblaient les fondations d'un moteur de bases de données stable et durable. Ceux des versions 8 avaient pour but

[49] https://dali.bo/workshop16_pdf
[50] https://blog.dalibo.com/2023/09/15/release_postgresql_16.html
[51] https://www.hagander.net/talks/PostgreSQL%2016.pdf

de rattraper les gros acteurs du marché en fonctionnalités et en performances. Enfin, pour les versions 9, on est plutôt sur la réplication et l'extensibilité.

La version 10 se base principalement sur la parallélisation des opérations (développement mené principalement par EnterpriseDB) et la réplication logique (par 2ndQuadrant). Les versions 11 à 16 améliorent ces deux points, entre mille autres améliorations en différents points du moteur, notamment les performances et la facilité d'administration.

1.3.13 Quelle version utiliser en production ?

- 11 et inférieures
  - **Danger !**
  - planifier une migration urgemment !
- 12, 13, 14, 15, 16
  - mises à jour mineures uniquement
- 16
  - nouvelles installations et nouveaux développements
- Tableau comparatif des versions[52]

Si vous avez une version 11 ou inférieure, planifiez le plus rapidement possible une migration vers une version plus récente, comme la 15 ou la 16. La 10 n'est plus maintenue depuis 2022, la 11 ne le sera plus dès novembre 2023. Elles fonctionneront toujours aussi bien, mais il n'y aura plus de correction de bug, y compris pour les failles de sécurité ! Si vous utilisez ces versions ou des versions antérieures, il est impératif d'étudier une migration de version dès que possible.

Les versions 12 à 16 sont celles recommandées pour une production. Le plus important est d'appliquer les mises à jour correctives. Attention, la version 12 ne sera plus supportée dès novembre 2024.

La version 16 est officiellement stable. Cette version est donc celle conseillée pour les nouvelles installations en production. Par expérience, quand une version x.0 paraît à l'automne, elle est généralement stable. Nombre de DBA préfèrent prudemment attendre les premières mises à jour mineures (en novembre généralement) pour la mise en production. Cette prudence est à mettre en balance avec l'intérêt pour les nouvelles fonctionnalités. Pour plus de détails, voir le tableau comparatif des versions[53].

[53] https://www.postgresql.org/about/featurematrix

1.3.14 Versions dérivées / Forks

Entre de nombreux autres :

- Compatibilité Oracle :
  - EnterpriseDB
- Data warehouse :
  - Greenplum, Netezza
- Forks :
  - Amazon RedShift, Aurora…
- Extensions :
  - Citus
  - timescaledb
- Packages avec des outils & support
- Bases compatibles

Il existe de nombreuses versions dérivées de PostgreSQL. Elles sont en général destinées à des cas d'utilisation très spécifiques et offrent des fonctionnalités non proposées par la version communautaire. Leur code est souvent fermé et nécessite l'acquisition d'une licence payante. La licence de PostgreSQL permet cela, et le phénomène existait déjà dès les années 1990 avec divers produits commerciaux comme Illustra.

Modifier le code de PostgreSQL a plusieurs conséquences négatives. Certaines fonctionnalités de PostgreSQL peuvent être désactivées. Il est donc difficile de savoir ce qui est réellement utilisable. De plus, chaque nouvelle version mineure demande une adaptation de leur ajout de code. Chaque nouvelle version majeure demande une adaptation encore plus importante de leur code. C'est un énorme travail, qui n'apporte généralement pas suffisamment de plus-value à la société éditrice pour qu'elle le réalise. La seule société qui le fait de façon complète est EnterpriseDB, qui arrive à proposer des mises à jour régulièrement. Par contre, si on revient sur l'exemple de Greenplum, ils sont restés bloqués pendant un bon moment sur la version 8.0. Ils ont cherché à corriger cela. Fin 2021, Greenplum 6.8 est au niveau de la version 9.4[54], version considérée alors comme obsolète par la communauté depuis plus de deux ans. En janvier 2023, Greenplum 7.0 bêta n'est toujours parvenu qu'au niveau de PostgreSQL 12.12…

Rien ne dit non plus que la société ne va pas abandonner son fork. Par exemple, il a existé quelques forks créés lorsque PostgreSQL n'était pas disponible en natif sous Windows : ces forks ont majoritairement disparu lors de l'arrivée de la version 8.0, qui proposait exactement cette fonctionnalité dans la version communautaire.

[54] https://web.archive.org/web/20211018012643/https://docs.greenplum.org/6-8/security-guide/topics/preface.html

Il y a eu aussi quelques forks créés pour gérer la réplication. Là aussi, la majorité de ces forks ont été abandonnés (et leurs clients avec) quand PostgreSQL a commencé à proposer de la réplication en version 9.0. Cependant, tous n'ont pas été abandonnés, en tout cas immédiatement. Par exemple, Slony est resté très vivant parce qu'il proposait des fonctionnalités que PostgreSQL n'avait pas encore à l'époque (notamment la réplication entre versions majeures différentes, et la réplication partielle). Ces fonctionnalités étant arrivées avec PostgreSQL 10, Slony est en fort déclin (tout comme Londiste, qui a été plus ou moins abandonné quand Skype a été racheté par Microsoft, ou Bucardo qu'on ne voit actuellement nulle part, du moins en France).

Il faut donc bien comprendre qu'à partir du moment où un utilisateur choisit une version dérivée, il dépend fortement (voire uniquement) de la bonne volonté de la société éditrice pour continuer son produit, le mettre à jour avec les dernières corrections et les dernières nouveautés de la version communautaire. Pour éviter ce problème, certaines sociétés ont décidé de transformer leur fork en une extension. C'est beaucoup plus simple à maintenir et n'enferme pas leurs utilisateurs. C'est le cas par exemple de citusdata (racheté par Microsoft) pour son extension de *sharding* ; ou encore de TimescaleDB, avec leur extension spécialisée dans les séries temporelles.

Dans les exemples de fork dédiés aux entrepôts de données, les plus connus historiquement sont Greenplum, de Pivotal (racheté par WMWare), et Netezza, d'IBM. Autant Greenplum tente de se raccrocher au PostgreSQL communautaire toutes les quelques années, autant ce n'est pas le cas de Netezza, optimisé pour du matériel dédié, et qui a forké de PostgreSQL 7.2.

Amazon, avec notamment les versions Redshift[55] ou Aurora, a la particularité de modifier profondément PostgreSQL pour l'adapter à son infrastructure, mais ne diffuse pas ses modifications. Même si certaines incompatibilités sont listées, il est très difficile de savoir où ils en sont et l'impact qu'a leurs modifications.

EDB Postgres Advanced Server d'EnterpriseDB permet de faciliter la migration depuis Oracle. Son code est propriétaire et soumis à une licence payante. Certaines fonctionnalités finissent par atterrir dans le code communautaire (une fois qu'EnterpriseDB le souhaite et que la communauté a validé l'intérêt de cette fonctionnalité et sa possible intégration).

BDR, anciennement de 2nd Quadrant, maintenant EnterpriseDB, est un *fork* visant à fournir une version multimaître de PostgreSQL, mais le code a été refermé dans les dernières versions. Il est très difficile de savoir où ils en sont. Son utilisation implique de prendre le support chez eux.

La société russe Postgres Pro, tout comme EnterpriseDB, propose diverses fonctionnalités dans sa version propre, tout en proposant souvent leur inclusion dans la version communautaire — ce qui n'est pas automatique.

Face au leadership de PostgreSQL, une tendance récente pour certaines bases de données est de se revendiquer « compatibles PostgreSQL ». Certains éditeurs de solutions de bases de données distribuées propriétaires disent que leur produit peut remplacer PostgreSQL sans modification de code côté application. Il convient de rester critique et prudent face à cette affirmation, car ces produits n'ont parfois rien à voir avec PosgreSQL.

[55] https://www.stitchdata.com/blog/how-redshift-differs-from-postgresql/

Cet historique provient en partie de la liste exhaustive des « forks »[56], ainsi de que cette conférence de Josh Berkus[57] de 2009 et des références en bibliographie.

Sauf cas très précis, il est recommandé d'utiliser la version officielle, libre et gratuite. Vous savez exactement ce qu'elle propose et vous choisissez librement vos partenaires (pour les formations, pour le support, pour les audits, etc).

1.4 QUELQUES PROJETS SATELLITES

PostgreSQL n'est que le moteur ! Besoin d'outils pour :

- Administration
- Sauvegarde
- Supervision
- Migration
- SIG

PostgreSQL n'est qu'un moteur de bases de données. Quand vous l'installez, vous n'avez que ce moteur. Vous disposez de quelques outils en ligne de commande (détaillés dans nos modules « Outils graphiques et consoles » et « Tâches courantes ») mais aucun outil graphique n'est fourni.

Du fait de ce manque, certaines personnes ont décidé de développer ces outils graphiques. Ceci a abouti à une grande richesse grâce à la grande variété de projets « satellites » qui gravitent autour du projet principal.

Par choix, nous ne présenterons ici que des logiciels libres et gratuits. Pour chaque problématique, il existe aussi des solutions propriétaires. Ces solutions peuvent parfois apporter des fonctionnalités inédites. Il faut néanmoins considérer que l'offre de la communauté Open-Source répond à la plupart des besoins des utilisateurs de PostgreSQL.

1.4.1 Administration, Développement, Modélisation

Entre autres, dédiés ou pas :

- Administration :
  - pgAdmin4
  - temBoard
- Développement :
  - DBeaver
- Modélisation :
  - pgModeler

Il existe différents outils graphiques pour l'administration, le développement et la modélisation. Une

liste plus exhaustive est disponible sur le wiki PostgreSQL[58].

pgAdmin4[59] est un outil d'administration dédié à PostgreSQL, qui permet aussi de requêter. (La version 3 est considérée comme périmée.)

temBoard[60] est une console d'administration plus complète. temBoard intègre de la supervision, des tableaux de bord, la gestion des sessions en temps réel, du bloat, de la configuration et l'analyse des performances.

DBeaver[61] est un outil de requêtage courant, utilisable avec de nombreuses bases de données différentes, et adapté à PostgreSQL.

Pour la modélisation, pgModeler[62] est dédié à PostgreSQL. Il permet la modélisation, la rétro-ingénierie d'un schéma existant, la génération de scripts de migration.

1.4.2 Sauvegardes

- Export logique :
  - pg_back[63]
- Sauvegarde physique (PITR) :
  - pgBackRest[64], barman[65]

Les outils listés ci-dessus sont les outils principaux et que nous recommandons pour la réalisation des sauvegardes et la gestion de leur rétention.

Ils se basent sur les outils standards de PostgreSQL de sauvegarde physique ou logique.

[58] https://wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools
[59] https://www.pgadmin.org/
[60] https://labs.dalibo.com/temboard
[61] https://dbeaver.io/
[62] https://pgmodeler.io/

1.4.3 Supervision

- Nagios/Icinga2 :
  - check_pgactivity
  - check_postgres
- Prometheus : postgres_exporter
- PoWA

Pour ne citer que quelques projets libres et matures :

check_pgactivity[66] est une sonde Nagios pouvant récupérer un grand nombre de statistiques d'activités renseignées par PostgreSQL. Il faut de ce fait un serveur Nagios (ou un de ses nombreux forks ou surcharges) pour gérer les alertes et les graphes. Il existe aussi check_postgres[67].

postgres_exporter[68] est l'exporteur de métriques pour Prometheus.

PoWA[69] est composé d'une extension qui historise les statistiques récupérées par l'extension `pg_stat_statements` et d'une application web qui permet de récupérer les requêtes et leur statistiques facilement.

1.4.4 Audit

- pgBadger
- pgCluu

pgBadger[70] est l'outil de base pour les analyses (à posteriori) des traces de PostgreSQL, dont notamment les requêtes.

pgCluu[71] permet une analyse du système et de PostgreSQL.

[66] https://github.com/OPMDG/check_pgactivity
[67] https://bucardo.org/check_postgres/
[68] https://github.com/prometheus-community/postgres_exporter
[69] https://powa.readthedocs.io/en/latest/
[70] https://pgbadger.darold.net/
[71] https://pgcluu.darold.net/

1.4.5 Migration

- Oracle, MySQL : ora2pg
- MySQL, SQL Server : pgloader

Il existe de nombreux outils pour migrer vers PostgreSQL une base de données utilisant un autre moteur. Ce qui pose le plus problème en pratique est le code applicatif (procédures stockées).

Plusieurs outils libres ou propriétaires, plus ou moins efficaces, existent - ou ont existé. Citons les plus importants :

Ora2Pg[72], de Gilles Darold, convertit le schéma de données, migre les données, et tente même de convertir le code PL/SQL en PL/pgSQL. Il convertit aussi des bases MySQL.

pgloader[73], de Dimitri Fontaine, permet de migrer depuis MySQL, SQLite ou MS SQL Server, et importe les fichiers CSV, DBF (dBase) ou IXF (fichiers d'échange indépendants de la base).

Ces outils sont libres. Des sociétés vivant de la prestation de service autour de la migration ont également souvent développé les leurs.

1.4.6 PostGIS

[72] http://ora2pg.darold.net/
[73] https://pgloader.io/

- Projet indépendant, GPL, https://postgis.net/
- Module spatial pour PostgreSQL
  - Extension pour types géométriques/géographiques & outils
  - La référence des bases de données spatiales
  - « quelles sont les routes qui coupent le Rhône ? »
  - « quelles sont les villes adjacentes à Toulouse ? »
  - « quels sont les restaurants situés à moins de 3 km de la Nationale 12 ? »

PostGIS ajoute le support d'objets géographiques à PostgreSQL. C'est un projet totalement indépendant développé par la société Refractions Research sous licence GPL, soutenu par une communauté active, utilisée par des spécialistes du domaine géospatial (IGN, BRGM, AirBNB, Mappy, Openstreetmap, Agence de l'eau…), mais qui peut convenir pour des projets plus modestes.

Techniquement, c'est une extension transformant 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 permet d'écrire des requêtes de ce type :

```
SELECT restaurants.geom, restaurants.name FROM restaurants
  WHERE EXISTS (SELECT 1 FROM routes
                  WHERE ST_DWithin(restaurants.geom, routes.geom, 3000)
                AND route.name = 'Nationale 12')
```

PostGIS fournit les fonctions d'indexation qui permettent d'accéder rapidement aux objets géométriques, au moyen d'index GiST. La requête ci-dessus n'a évidemment pas besoin de parcourir tous les restaurants à la recherche de ceux correspondant aux critères de recherche.

La liste des fonctionnalités comprend le support des coordonnées géodésiques ; des projections et reprojections dans divers systèmes de coordonnées locaux (Lambert93 en France par exemple) ; des opérateurs d'analyse géométrique (enveloppe convexe, simplification…)

PostGIS est intégré aux principaux serveurs de carte, ETL, et outils de manipulation.

La version 3.0 apporte la gestion du parallélisme, un meilleur support de l'indexation SP-GiST et GiST, ainsi qu'un meilleur support du type GeoJSON.

1.5 SPONSORS & RÉFÉRENCES

- Sponsors[74]
- Références :
  - françaises
  - et internationales

Au-delà de ses qualités, PostgreSQL suscite toujours les mêmes questions récurrentes :

- qui finance les développements ? (et pourquoi ?)
- qui utilise PostgreSQL ?

1.5.1 Sponsors principaux

- Sociétés se consacrant à PostgreSQL :
  - Crunchy Data (USA) : Tom Lane, Stephen Frost, Joe Conway…
  - EnterpriseDB (USA) : Bruce Momjian, Robert Haas, Dave Page…
  - 2nd Quadrant (R.U.) : Simon Riggs, Peter Eisentraut…
    - racheté par EDB
  - PostgresPro (Russie) : Oleg Bartunov, Alexander Korotkov
  - Cybertec (Autriche), Dalibo (France), Redpill Linpro (Suède), Credativ (Allemagne)…
- Sociétés vendant un fork ou une extension :
  - Citusdata (Microsoft), Pivotal (VMWare), TimescaleDB

La liste des sponsors de PostgreSQL contribuant activement au développement figure sur la liste officielle des sponsors[75]. Ce qui suit n'est qu'un aperçu.

EnterpriseDB est une société américaine qui a décidé de fournir une version de PostgreSQL propriétaire fournissant une couche de compatibilité avec Oracle. Ils emploient plusieurs développeurs importants du projet PostgreSQL (dont trois font partie de la *Core Team*), et reversent un certain nombre

[75] https://www.postgresql.org/about/sponsors/

de leurs travaux au sein du moteur communautaire. Ils ont aussi un poids financier qui leur permet de sponsoriser la majorité des grands événements autour de PostgreSQL : PGEast et PGWest aux États-Unis, PGDay en Europe.

En 2020, EnterpriseDB rachète 2nd Quadrant, une société anglaise fondée par Simon Riggs, développeur PostgreSQL de longue date. 2nd Quadrant développe de nombreux outils autour de PostgreSQL comme pglogical, des versions dérivées comme Postgres-XL ou BDR, ou des outils annexes comme barman ou repmgr.

Crunchy Data offre sa propre version certifiée et finance de nombreux développements.

De nombreuses autres sociétés dédiées à PostgreSQL existent dans de nombreux pays. Parmi les sponsors officiels, nous pouvons compter Cybertec en Autriche ou Redpill Linpro en Suède. En Russie, PostgresPro maintient une version locale et reverse aussi de nombreuses contributions à la communauté.

En Europe francophone, Dalibo participe pleinement à la communauté. La société est Major Sponsor du projet PostgreSQL[76], ce qui indique un support de longue date. Elle développe et maintient plusieurs outils plébiscités par la communauté, comme autrefois Open PostgreSQL Monitoring (OPM) ou la sonde check_pgactivity[77], plus récemment la console d'administration temBoard[78], avec de nombreux autres projets en cours[79], et une participation active au développement de patchs pour PostgreSQL. Dalibo sponsorise également des événements comme les PGDay français et européens, ainsi que la communauté francophone.

Des sociétés comme Citusdata (racheté par Microsoft), Pivotal (VMWare) ou TimescaleDB proposent ou ont proposé leur version dérivée sous une forme ou une autre, mais « jouent le jeu » et participent au développement de la version communautaire, notamment en cherchant à ce que leur produit n'en diverge pas.

1.5.2 Autres sponsors

- Autres sociétés :
  - VMWare, Rackspace, Heroku, Conova, Red Hat, Microsoft
  - NTT (*streaming replication*), Fujitsu, NEC
- Cloud
  - nombreuses

[76] https://www.postgresql.org/about/sponsors/
[77] https://github.com/OPMDG/check_pgactivity
[78] https://labs.dalibo.com/temboard
[79] https://labs.dalibo.com/about

Contribuent également à PostgreSQL nombre de sociétés non centrées autour des bases de données.

NTT a financé de nombreux patchs pour PostgreSQL.

Fujitsu a participé à de nombreux développements aux débuts de PostgreSQL, et emploie Amit Kapila.

VMWare a longtemps employé le développeur finlandais Heikki Linnakangas, parti ensuite un temps chez Pivotal. VMWare emploie aussi Michael Paquier ou Julien Rouhaud.

Red Hat a longtemps employé Tom Lane à plein temps pour travailler sur PostgreSQL. Il a pu dédier une très grande partie de son temps de travail à ce projet, bien qu'il ait eu d'autres affectations au sein de Red Hat. Tom Lane a travaillé également chez SalesForce, ensuite il a rejoint Crunchy Data Solutions fin 2015.

Il y a déjà plus longtemps, Skype a offert un certain nombre d'outils très intéressants : pgBouncer (pooler de connexion), Londiste (réplication par trigger), etc. Ce sont des outils utilisés en interne et publiés sous licence BSD comme retour à la communauté. Malgré le rachat par Microsoft, certains sont encore utiles et maintenus.

Zalando est connu pour l'outil de haute disponibilité patroni.

De nombreuses sociétés liées au cloud figurent aussi parmi les sponsors, comme Conova (Autriche), Heroku ou Rackspace (États-Unis), ou les mastodontes Google, Amazon Web Services et, à nouveau, Microsoft.

1.5.3 Références

- Météo France
- IGN
- RATP, SNCF
- CNAF
- MAIF, MSA
- Le Bon Coin
- Air France-KLM
- Société Générale
- Carrefour, Leclerc, Leroy Merlin
- Instagram, Zalando, TripAdvisor
- Yandex
- CNES
- …et plein d'autres

Météo France utilise PostgreSQL depuis plus d'une décennie pour l'essentiel de ses bases, dont des

instances critiques de plusieurs téraoctets (témoignage sur postgresql.fr[80]).

L'IGN utilise PostGIS et PostgreSQL depuis 2006[81].

La RATP a fait ce choix depuis 2007 également[82].

La Caisse Nationale d'Allocations Familiales a remplacé ses mainframes par des instances PostgreSQL[83] dès 2010 (4 To et 1 milliard de requêtes par jour).

Instagram utilise PostgreSQL depuis le début[84].

Zalando a décrit plusieurs fois son infrastructure PostgreSQL[85] et annonçait en 2018[86] utiliser pas moins de 300 bases de données en interne et 650 instances dans un cloud AWS. Zalando contribue à la communauté, notamment par son outil de haute disponibilité patroni[87].

Le DBA de TripAdvisor témoigne de leur utilisation de PostgreSQL dans l'interview suivante[88].

Dès 2009, Leroy Merlin migrait vers PostgreSQL des milliers de logiciels de caisse[89].

Yandex, équivalent russe de Google a décrit en 2016 la migration des 300 To de données de Yandex.Mail depuis Oracle vers PostgreSQL[90].

La Société Générale a publié son outil de migration d'Oracle à PostgreSQL[91].

Autolib à Paris utilisait PostgreSQL. Le logiciel est encore utilisé dans les autres villes où le service continue. Ils ont décrit leur infrastructure au PG Day 2018 à Marseille[92].

De nombreuses autres sociétés participent au Groupe de Travail Inter-Entreprises de PostgreSQLFr[93] : Air France, Carrefour, Leclerc, le CNES, la MSA, la MAIF, PeopleDoc, EDF…

Cette liste ne comprend pas les innombrables sociétés qui n'ont pas communiqué sur le sujet. PostgreSQL étant un logiciel libre, il n'existe nulle part de dénombrement des instances actives.

[80] https://www.postgresql.fr/temoignages/meteo_france
[81] https://www.postgresql.fr/temoignages/ign
[82] https://www.journaldunet.com/solutions/dsi/1013631-la-ratp-integre-postgresql-a-son-systeme-d-information/
[83] https://www.silicon.fr/cnaf-debarrasse-mainframes-149897.html?inf_by=5bc488a1671db858728b4c35
[84] https://media.postgresql.org/sfpug/instagram_sfpug.pdf
[85] http://gotocon.com/dl/goto-berlin-2013/slides/HenningJacobs_and_ValentineGogichashvili_WhyZalandoTrustsInPostgreSQL.pdf
[86] https://www.postgresql.eu/events/pgconfeu2018/schedule/session/2135-highway-to-hell-or-stairway-to-cloud/
[87] https://jobs.zalando.com/tech/blog/zalandos-patroni-a-template-for-high-availability-postgresql/
[88] https://www.citusdata.com/blog/25-terry/285-matthew-kelly-tripadvisor-talks-about-pgconf-silicon-valley
[89] https://wiki.postgresql.org/images/6/63/Adeo_PGDay.pdf
[90] https://www.pgcon.org/2016/schedule/attachments/426_2016.05.19%20Yandex.Mail%20success%20story.pdf
[91] https://github.com/societe-generale/code2pg
[92] https://www.youtube.com/watch?v=vd8B7B-Zca8
[93] https://www.postgresql.fr/entreprises/accueil

1.5.4 Le Bon Coin

- Site de petites annonces
- 4è site le plus consulté en France (2017)
- 27 millions d'annonces en ligne, 800 000 nouvelles chaque jour
- Instance PostgreSQL principale : 3 To de volume, 3 To de RAM
- 20 serveurs secondaires

PostgreSQL tient la charge sur de grosses bases de données et des serveurs de grande taille.

Le Bon Coin privilégie des serveurs physiques dans ses propres datacenters.

Pour plus de détails et l'évolution de la configuration, voir les témoignages de ses directeurs technique[94] (témoignage de juin 2012) et infrastructure[95] (juin 2017), ou la conférence de son DBA Flavio Gurgel au pgDay Paris 2019[96].

Ce dernier s'appuie sur les outils classiques fournis par la communauté : pg_dump (pour archivage, car ses exports peuvent être facilement restaurés), barman, pg_upgrade.

[94] https://www.postgresql.fr/temoignages:le_bon_coin
[95] https://web.archive.org/web/20171222173630/https://www.kissmyfrogs.com/jean-louis-bergamo-leboncoin-ce-qui-a-ete-fait-maison-est-ultra-performant/
[96] https://www.postgresql.eu/events/pgdayparis2019/schedule/session/2376-large-databases-lots-of-servers

1.6 À LA RENCONTRE DE LA COMMUNAUTÉ

- Cartographie du projet
- Pourquoi participer
- Comment participer

1.6.1 PostgreSQL, un projet mondial

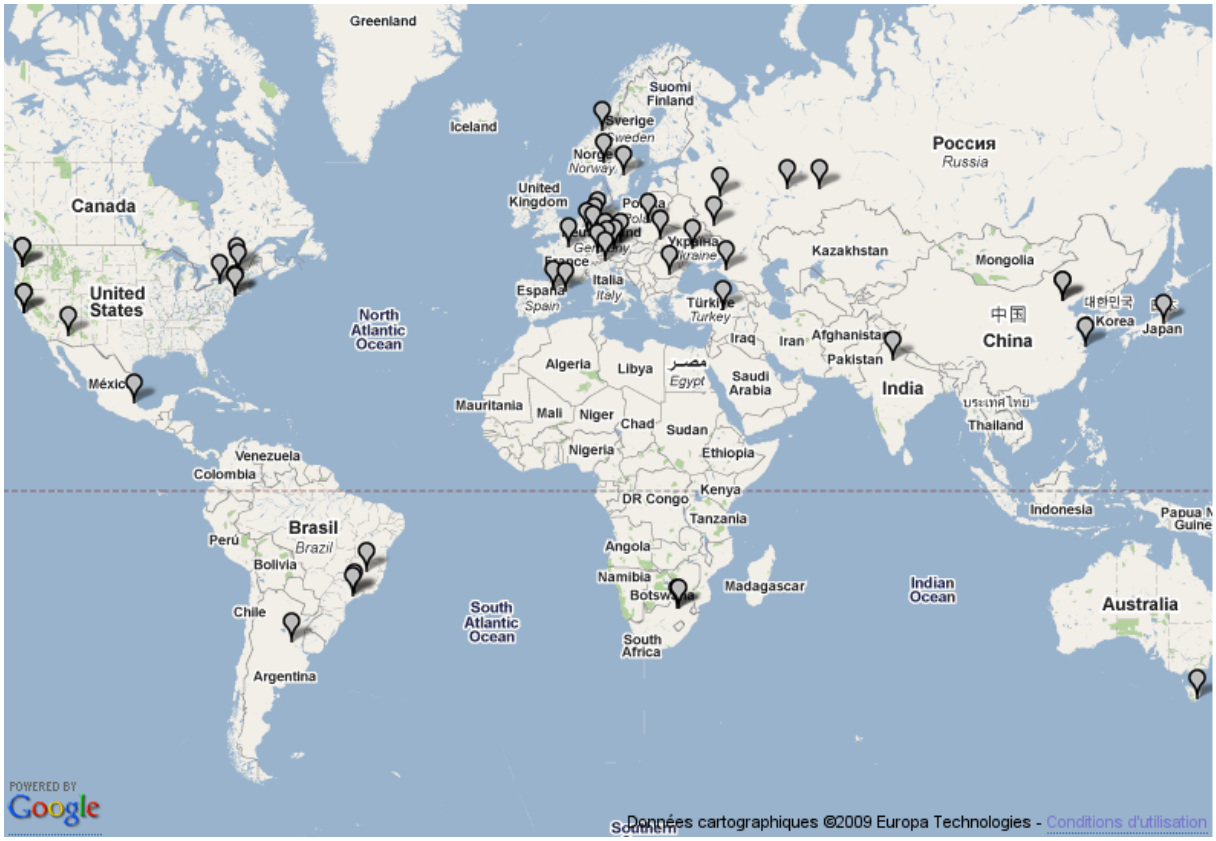


Figure 1/ .2: Carte des hackers

On le voit, PostgreSQL compte des contributeurs sur tous les continents.

Le projet est principalement anglophone. Les *core hackers* sont surtout répartis en Amérique, Europe, Asie (Japon surtout).

Il existe une très grande communauté au Japon, et de nombreux développeurs en Russie.

La communauté francophone est très dynamique, s'occupe beaucoup des outils, mais il n'y a que quelques développeurs réguliers du *core* francophones : Michael Paquier, Julien Rouhaud, Fabien Coelho…

La communauté hispanophone est naissante.

1.6.2 PostgreSQL Core Team

Figure 1/ .3: Core team

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

Le terme *hacker* peut porter à confusion, il s'agit ici de la définition « universitaire » : https://fr.wikipedia.org/wiki/Hacker_(programmation)

La *Core Team* est un ensemble de personnes doté d'un pouvoir assez limité. Ils ne doivent pas appartenir en majorité à la même société. Ils peuvent décider de la date de sortie d'une version. Ce sont les personnes qui sont immédiatement au courant des failles de sécurité du serveur PostgreSQL. Exceptionnellement, elles tranchent certains débats si un consensus ne peut être atteint dans la communauté. Tout le reste des décisions est pris par la communauté dans son ensemble après discussion, généralement sur la liste pgsql-hackers.

Les membres actuels de la *Core Team* sont[97] :

- **Tom Lane** (Crunchy Data, Pittsburgh, États-Unis) : certainement le développeur le plus aguerri avec la vision la plus globale, notamment sur l'optimiseur ;

[97] https://www.postgresql.org/community/contributors/

- **Bruce Momjian** (EnterpriseDB, Philadelphie, États-Unis) : a lancé le projet en 1995, écrit du code (pg_upgrade notamment) et s'est beaucoup occupé de la promotion ;
- **Magnus Hagander** (Redpill Linpro, Stockholm, Suède) : développeur, a participé notamment au portage Windows, à l'outil pg_basebackup, à l'administration des serveurs, président de PostgreSQL Europe ;
- **Andres Freund** (Microsoft, San Francisco, États-Unis) : contributeur depuis des années de nombreuses fonctionnalités (JIT, réplication logique, performances…) ;
- **Dave Page** (EnterpriseDB, Oxfordshire, Royaume-Uni) : leader du projet pgAdmin, version Windows, administration des serveurs, secrétaire de PostgreSQL Europe ;
- **Peter Eisentraut** (EnterpriseDB, Dresde, Allemagne) : développement du moteur (internationalisation, SQL/Med…), respect de la norme SQL, etc. ;
- **Jonathan Katz** (Crunchy Data, New York, États-Unis) : promotion du projet, modération, revues de patchs.

1.6.3 Contributeurs

Figure 1/ .4: Contributeurs

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

- développement des projets satellites (Slony, pgAdmin…) ;
- promotion du logiciel ;
- administration de l'infrastructure des serveurs ;

- rédaction de documentation ;
- conférences ;
- traductions ;
- organisation de groupes locaux.

Le *PGDG* a fêté son 10e anniversaire à Toronto en juillet 2006. Ce « PostgreSQL Anniversary Summit » a réuni pas moins de 80 membres actifs du projet. La photo ci-dessus a été prise à l'occasion.

PGCon2009 a réuni 180 membres actifs à Ottawa, et environ 220 en 2018 et 2019.

Voir la liste des contributeurs officiels[98].

1.6.4 Qui contribue du code ?

- Principalement des personnes payées par leur société
- 28 committers[99]
- En 2019, en code :
  - Tom Lane
  - Andres Freund
  - Peter Eisentraut
  - Nikita Glukhov
  - Álvaro Herrera
  - Michael Paquier
  - Robert Haas
  - …et beaucoup d'autres
- Commitfests[100] : tous les 2 mois

À l'automne 2021, on compte 28 *committers*, c'est-à-dire personnes pouvant écrire dans tout ou partie du dépôt de PostgreSQL. Il ne s'agit pas que de leur travail, mais pour une bonne partie de patchs d'autres contributeurs après discussion et validation des fonctionnalités mais aussi des standards propres à PostgreSQL, de la documentation, de la portabilité, de la simplicité, de la sécurité, etc. Ces autres contributeurs peuvent être potentiellement n'importe qui. En général, un patch est relu par plusieurs personnes avant d'être transmis à un *committer*.

Les discussions quant au développement ont lieu principalement (mais pas uniquement) sur la liste pgsql-hackers[101]. Les éventuels bugs sont transmis à la liste pgsql-bugs[102]. Puis les patchs en cours sont revus au moins tous les deux mois lors des Commitfests. Il n'y a pas de *bug tracker* car le fonctionnement actuel est jugé satisfaisant.

[98] https://www.postgresql.org/community/contributors/
[101] https://www.postgresql.org/list/pgsql-hackers/
[102] https://www.postgresql.org/list/pgsql-bugs/

Robert Haas publie chaque année une analyse sur les contributeurs de code et les participants aux discussions sur le développement de PostgreSQL sur la liste pgsql-hackers :

- 2020/2021 : http://rhaas.blogspot.com/2022/01/who-contributed-to-postgresql.html
- 2019 : http://rhaas.blogspot.com/2020/05/who-contributed-to-postgresql.html
- 2018 : http://rhaas.blogspot.com/2019/01/who-contributed-to-postgresql.html
- 2017 : http://rhaas.blogspot.com/2018/06/who-contributed-to-postgresql.html
- 2016 : http://rhaas.blogspot.com/2017/04/who-contributes-to-postgresql.html

1.6.5 Répartition des développeurs

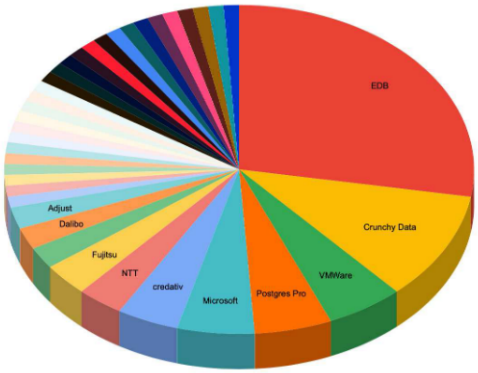


Figure 1/ .5: Répartition des développeurs

Voici une répartition des différentes sociétés qui ont contribué aux améliorations de la version 13. On y voit qu'un grand nombre de sociétés prend part à ce développement. La plus importante est EDB, mais même elle n'est responsable que d'un petit tiers des contributions.

(Source : Future Postgres Challenges[103], Bruce Momjian, 2021)

1.6.6 Utilisateurs

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

Il est impossible de connaître précisément le nombre d'utilisateurs de PostgreSQL. Toutefois ce nombre est en constante augmentation.

[103] https://momjian.us/main/writings/pgsql/challenges.pdf

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.

1.6.7 Pourquoi participer

- Rapidité des corrections de bugs
- Préparer les migrations / tester les nouvelles versions
- Augmenter la visibilité du projet
- Créer un réseau d'entraide

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

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 sur pgsql-bugs[104] ! Assurez-vous que vous pouvez le reproduire.

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.

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.

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.

1.6.8 Ressources web de la communauté

- Site officiel : https://www.postgresql.org/
- Actualité : https://planet.postgresql.org/
- Des extensions : https://pgxn.org/

[104] https://www.postgresql.org/list/pgsql-bugs/

Le site officiel de la communauté se trouve sur https://www.postgresql.org/. Ce site contient des informations sur PostgreSQL, la documentation des versions maintenues, les archives des listes de discussion, etc.

Le site « Planet PostgreSQL » est un agrégateur réunissant les blogs des *Core Hackers*, des contributeurs, des traducteurs et des utilisateurs de PostgreSQL.

Le site PGXN est l'équivalent pour PostgreSQL du CPAN de Perl, une collection en ligne de librairies et extensions accessibles depuis la ligne de commande.

1.6.9 Documentation officielle

- LA référence, même au quotidien
- Anglais : https://www.postgresql.org/docs/
- Français : https://docs.postgresql.fr/

La documentation officielle sur https://www.postgresql.org/docs/current est maintenue au même titre que le code du projet, et sert aussi au quotidien, pas uniquement pour des cas obscurs.

Elle est versionnée pour chaque version majeure.

La traduction française suit de près les mises à jour de la documentation officielle : https://docs.postgresql.fr/.

1.6.10 Serveurs francophones

- Site officiel : https://www.postgresql.fr/
- Documentation traduite : https://docs.postgresql.fr/
- Forum : https://forums.postgresql.fr/
- Actualité : https://planete.postgresql.fr/
- Association PostgreSQLFr : https://www.postgresql.fr/asso/accueil
- Groupe de Travail Inter-Entreprises (PGGTIE) : https://www.postgresql.fr/entreprises/accueil

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

1.6.11 Listes de discussions / Listes d'annonces

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

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. Les développeurs principaux du projets répondent parfois eux-mêmes. Si vous avez une question ou un problème, la réponse se trouve probablement dans les archives !

Pour s'inscrire ou consulter les archives : https://www.postgresql.org/list/.

Si vous pensez avoir trouvé un bug, vous pouvez le remonter sur la liste anglophone pgsql-bugs[105], par le formulaire dédié[106]. Pour faciliter la tâche de ceux qui tenteront de vous répondre, suivez bien les consignes sur les rapports de bug[107] : informations complètes, reproductibilité…

1.6.12 IRC

- Réseau LiberaChat
- IRC anglophone :
  - #postgresql
  - #postgresql-eu
- IRC francophone :
  - #postgresqlfr

Le point d'entrée principal pour le réseau LiberaChat est le serveur **irc.libera.chat**. La majorité des développeurs sont disponibles sur IRC et peuvent répondre à vos questions.

Des canaux de discussion spécifiques à certains projets connexes sont également disponibles, comme par exemple #slony.

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

1.6.13 Wiki

- https://wiki.postgresql.org/

Le wiki est un outil de la communauté qui met à disposition une véritable mine d'informations.

Au départ, le wiki avait pour but de récupérer les spécifications écrites par des développeurs pour les grosses fonctionnalités à développer à plusieurs. Cependant, peu de développeurs l'utilisent dans ce cadre. L'utilisation du wiki a changé en passant plus entre les mains des utilisateurs qui y intègrent un bon nombre de pages de documentation (parfois reprises dans la documentation officielle). Le wiki est aussi utilisé par les organisateurs d'événements pour y déposer les slides des conférences. Elle n'est pas exhaustive et, hélas, souffre fréquemment d'un manque de mises à jour.

1.6.14 L'avenir de PostgreSQL

- PostgreSQL est devenu la base de données de référence
- Grandes orientations :
  - réplication logique
  - meilleur parallélisme
  - gros volumes
- Prochaine version, la 17
- Stabilité économique
- De plus en plus de (gros) clients
- Le futur de PostgreSQL dépend de vous !

Le projet avance grâce à de plus en plus de contributions. Les grandes orientations actuelles sont :

- une réplication de plus en plus sophistiquée ;
- une gestion plus étendue du parallélisme ;
- une volumétrie acceptée de plus en plus importante ;
- etc.

PostgreSQL est là pour durer. Le nombre d'utilisateurs, de toutes tailles, augmente tous les jours. Il n'y a pas qu'une seule entreprise derrière ce projet. Il y en a plusieurs, petites et grosses sociétés, qui s'impliquent pour faire avancer le projet, avec des modèles économiques et des marchés différents, garants de la pérennité du projet.

1.7 CONCLUSION

- Un projet de grande ampleur
- Un SGBD complet
- Souplesse, extensibilité
- De belles références
- Une solution **stable**, **ouverte**, **performante** et **éprouvée**
- Pas de dépendance envers UN éditeur

Certes, la licence PostgreSQL implique un coût nul (pour l'acquisition de la licence), un code source disponible et aucune contrainte de redistribution. Toutefois, il serait erroné de réduire le succès de PostgreSQL à sa gratuité.

Beaucoup d'acteurs font le choix de leur SGBD sans se soucier de son prix. En l'occurrence, ce sont souvent les qualités intrinsèques de PostgreSQL qui séduisent :

- sécurité des données (reprise en cas de crash et résistance aux bogues applicatifs) ;
- facilité de configuration ;
- montée en puissance et en charge progressive ;
- gestion des gros volumes de données ;
- pas de dépendance envers un unique éditeur ou prestataire.

1.7.1 Bibliographie

- Documentation officielle (préface)
- Articles fondateurs de M. Stonebracker (1987)
- *Présentation du projet PostgreSQL* (Guillaume Lelarge, 2008)
- *Looking back at PostgreSQL* (J.M. Hellerstein, 2019)

Quelques références :

- Préface de la documentation officielle : 2. Bref historique de PostgreSQL[108]
- *The Design of POSTGRES*[109] , Michael Stonebraker & Lawrence A. Rowe, 1987
- Présentation du projet PostgreSQL[110], Guillaume Lelarge, RMLL 2008

[108] https://docs.postgresql.fr/current/history.html
[109] http://db.cs.berkeley.edu/papers/ERL-M85-95.pdf
[110] https://web.archive.org/web/20160322070704/2008.rmll.info/Presentation-de-PostgreSQL.html

- *Looking Back at PostgreSQL*[111] , Joseph M. Hellerstein, 2019

Iconographie : La photo initiale est le logo officiel de PostgreSQL[112].

1.7.2 Questions

N'hésitez pas, c'est le moment !

[111] https://arxiv.org/pdf/1901.01973.pdf
[112] https://www.postgresql.org/about/policies/trademarks/

1.8 QUIZ

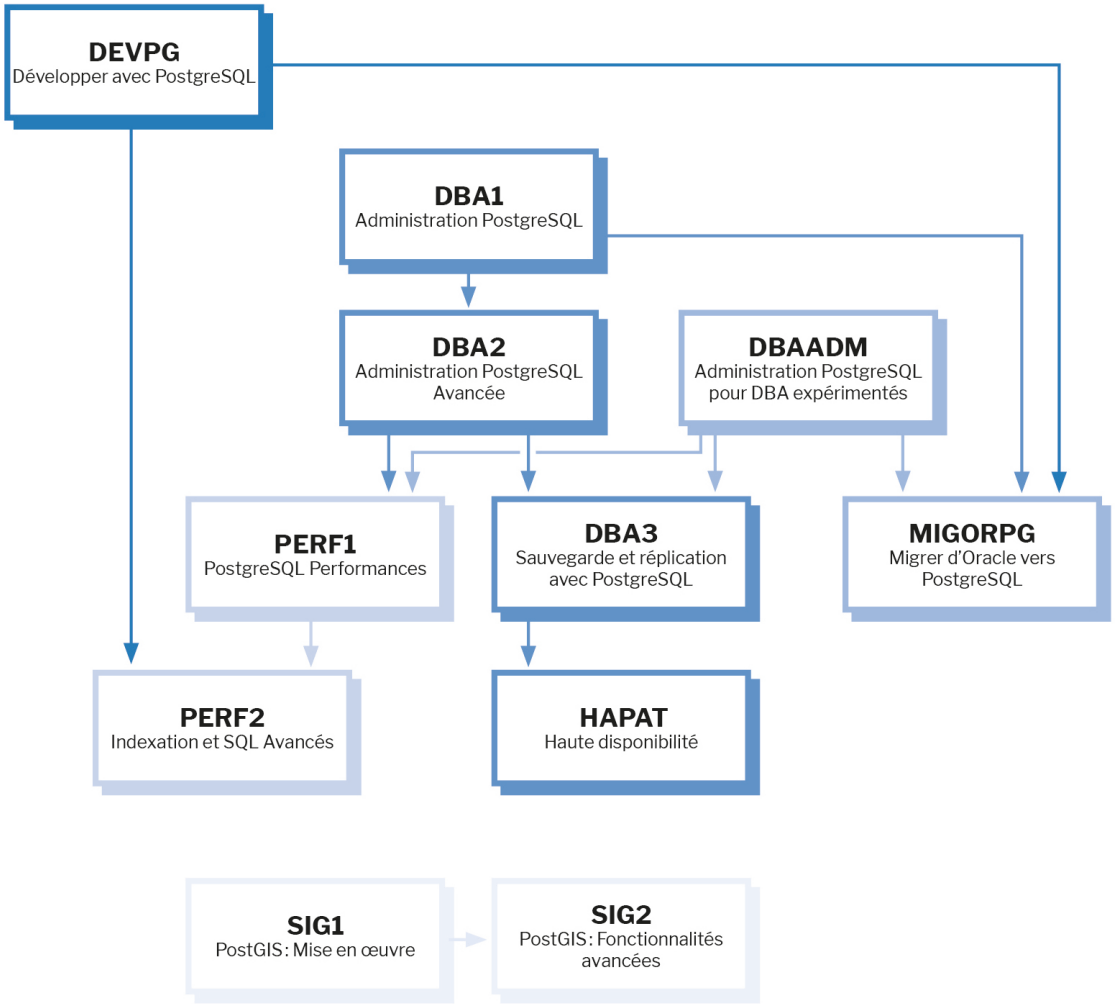
https://dali.bo/a1_quiz

Les formations Dalibo

Retrouvez nos formations et le calendrier sur https://dali.bo/formation

Pour toute information ou question, n'hésitez pas à nous écrire sur contact@dalibo.com.

Cursus des formations



Retrouvez nos formations dans leur dernière version :

- DBA1 : Administration PostgreSQL
  https://dali.bo/dba1
- DBA2 : Administration PostgreSQL avancé
  https://dali.bo/dba2
- DBA3 : Sauvegarde et réplication avec PostgreSQL
  https://dali.bo/dba3
- DEVPG : Développer avec PostgreSQL
  https://dali.bo/devpg
- PERF1 : PostgreSQL Performances
  https://dali.bo/perf1
- PERF2 : Indexation et SQL avancés
  https://dali.bo/perf2
- MIGORPG : Migrer d'Oracle à PostgreSQL
  https://dali.bo/migorpg
- HAPAT : Haute disponibilité avec PostgreSQL
  https://dali.bo/hapat

Les livres blancs

- Migrer d'Oracle à PostgreSQL
  https://dali.bo/dlb01
- Industrialiser PostgreSQL
  https://dali.bo/dlb02
- Bonnes pratiques de modélisation avec PostgreSQL
  https://dali.bo/dlb04
- Bonnes pratiques de développement avec PostgreSQL
  https://dali.bo/dlb05

Téléchargement gratuit

Les versions électroniques de nos publications sont disponibles gratuitement sous licence open source ou sous licence Creative Commons.