DALIBO

L'Expertise PostgreSQL

Organisation du projet PostgreSQL

CC BY NC SA

1 À propos des auteurs...

» Auteur : Guillaume Lelarge
» Société : DALIBO
» Date : Octobre 2010
» URL : https://support.dalibo.com/kb/conferences/organisation_projet_postgresql/

2 Licence

- Licence Creative Common BY-NC-SA
- 3 contraintes de partage :
  - Citer la source (dalibo)
  - Pas d'utilisation commerciale
  - Partager sous licence BY-NC-SA

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

Ceci est un résumé explicatif du Code Juridique. La version intégrale du contrat est disponible ici : http://creativecommons.org/licenses/by-nc-sa/2.0/fr/legalcode

3 Au menu

- Organisation dans l'histoire de PostgreSQL
  - Les problèmes rencontrés, les solutions apportées
- Résumé sur les outils utilisés
- Cycle de développement
  - Version majeure
  - Version mineure
- Petit résumé sur les différents types de contributeurs

Ce document tente de présenter l'organisation du projet PostgreSQL. Le fonctionnement interne est très intéressant pour tous ceux qui participent à des projets libres car ils peuvent s'en inspirer. Pour le monde de l'entreprise, le fonctionnement d'un projet libre est souvent assez mystérieux et beaucoup d'à priori sur les projets libres et opensource sont dûs en grande partie à cette méconnaissance.

Le but de cette présentation est donc de montrer que l'organisation d'un projet libre comme PostgreSQL n'a rien à envier à l'organisation disponible chez certaines entreprises et que même, dans bien des cas, cette organisation est bien plus productive.

4 Première partie

Évolution de l'organisation

La première partie de cette conférence va s'attacher à présenter l'organisation du projet PostgreSQL tout au long de son histoire. Cela va permettre d'indiquer les différents problèmes rencontrés et les solutions qui ont été acceptées.

5 PostgreSQL

- Serveur de bases de données libre (MIT/BSD)
- Respectueux des normes SQL
- Respectueux des données
- Une excellente portabilité
- Une grande communauté réactive et internationale

PostgreSQL est un serveur de bases de données libre. Il est disponible sous une licence libre proche de celles de BSD et MIT. C'est l'une des moins restrictives dans le monde informatique. En effet, cette licence garantit:

- l'accès au code source du logiciel ;
- le droit de modifier le code.

De plus, elle n'impose:

- aucune restriction d'utilisation ;
- aucune viralité de sa licence ;

- aucune contrainte de redistribution des modifications éventuelles ;
- aucune contrainte de publicité pour PostgreSQL et/ou sa communauté.

Ses caractéristiques les plus importantes au niveau technique sont son respect de la norme SQL et des données des utilisateurs. Tout est fait, lors du développement de PostgreSQL, pour respecter le plus possible ses deux caractéristiques.

Une autre caractéristique essentielle est la portabilité. PostgreSQL fonctionne sur un grand nombre d'architectures processeur (x86, x86_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc64, Alpha, ARM, MIPS, MIPSEL, M68K et PA-RISC) et sur plusieurs systèmes d'exploitation (Linux, Windows 2000 et ultérieurs, toutes les distributions de la famille BSD, Mac OS X, AIX, HP/UX, IRIX, Tru64 Unix, UnixWare, Solaris).

Sa caractéristique humaine la plus flagrante est sa communauté. Elle est importante et répartie dans de nombreux pays, sur tous les continents. Elle est très réactive. Il est très fréquent qu'une question reçoive une réponse dans l'heure qui suit. Les bugs sont pris très au sérieux et là-aussi ils sont rapidement disponibles après la découverte du bug.

6 Un peu d'histoire

- 1970: développement de Ingres
- 1985: développement de Postgres
- 1995: ajout du langage SQL
- 1996: libération du code
  - renommage en PostgreSQL
  - création du groupe PGDG

Dans les années 1970, Michael Stonebraker développe Ingres à l'université de Berkeley. Il continue à travailler sur ce projet jusqu'en 1985. Pendant toute cette période, il apprend, il améliore et surtout il constate un certain nombre de problèmes dans la conception d'Ingres. Tant et si bien qu'en 1985, il décide de développer un nouveau serveur de bases de données en n'utilisant rien du code de Ingres, mais en faisant profiter à ce nouveau projet de toute l'expérience qu'il a pu acquérir lors de son travail sur Ingres.

Postgres est ce nouveau projet. Le nom est une contraction de « Post-Ingres », l'après Ingres. Ce logiciel est toujours développé à l'université de Berkeley et n'en sort pas. Deux étudiants chinois vont ajouter le langage SQL en 1995 (avant cela, ils utilisaient le langage QUEL). D'autres

développements vont continuer mais il n'y a plus personne à la tête du projet.

En 1996, Bruce Momjian décide de reprendre le projet et convainc l'université de libérer le code. Après avoir testé les différents patchs qui traînaient, il décide d'en intégrer certains et de sortir la première version libre de Postgres. Le projet est renommé PostgreSQL et un groupe chargé de son développement est créé. Ce groupe s'appelle le PostgreSQL Global Development Group.

7 Mise en place du développement

- Licence PostgreSQL (MIT)
- Création d'un dépôt CVS
- Création des listes de discussion

PostgreSQL est un logiciel libre. Sa licence se rapproche beaucoup de la licence MIT.

Dès la reprise du projet, Bruce Momjian travaille avec Marc Fournier pour mettre en place une plateforme de développement. Cela passe tout d'abord par un dépôt des sources utilisant CVS, des listes de discussions, un serveur web.

8 Cycle de développement en 96

- Les patchs arrivent sur la liste de discussion
- et sont intégrés quand un commiter a pu vérifier le patch
- Peu de commiters

L'ouverture du code à la communauté du libre accroît assez rapidement le nombre de participants à son développement. Les patchs sont envoyés sur la liste de discussion des développeurs (nommée pgsql-hackers, dont les archives sont disponibles sur http://archives.postgresql.org/pgsql-hackers/). Bruce Momjian et quelques autres les intègrent dans le code source après vérification du patch.

9 Vérification d'un patch

- S'assurer de la sécurité des données
- S'assurer du respect des normes
- Tester les performances
- Vérifier la simplicité du code
- Relire la documentation

La vérification d'un patch est une opération en plusieurs étapes.

La première consiste à s'assurer que les données ne courent aucun risque: pas de corruption des données, un vidage du cache Linux en cas d'écriture sur disque, etc.

La deuxième étape concerne le respect des normes. Lorsqu'une nouvelle instruction SQL ou une nouvelle clause SQL est ajoutée, le premier soucis sera de vérifier dans la norme si cette instruction/clause est disponible. Si c'est le cas, le patch devra au moins respecter cette syntaxe. Il peut aussi en respecter une autre pour faciliter la migration d'un autre serveur de bases de données à PostgreSQL, mais il est tout d'abord tenu à respecter la syntaxe de la norme.

Les performances ne viennent qu'en troisième position. En effet, rien ne sert à avoir des performances si la sécurité des données n'est pas assurée et si la norme SQL n'est pas respectée. Néanmoins, le travail sur les performances est un point nécessaire pour les utilisateurs et on vérifiera donc qu'il n'y a pas de régressions de performances. Si le patch est supposé apporter de meilleures performances, ce sera aussi vérifié.

La quatrième étape est essentielle dans un projet libre. En effet, rien ne dit que le développeur d'un patch pour une nouvelle fonctionnalité sera toujours là, prêt à aider, en cas de découverte d'un bug dans sa fonctionnalité quelques mois après. Quand on sait qu'une version majeure PostgreSQL a une assurance de cinq ans de mises à jour correctives, la garantie d'avoir toujours le développeur au bout de quatre ans est encore plus mince. Donc il est essentiel pour les contributeurs de long terme de n'intégrer que les patchs dont ils comprennent le code et dont le code est facile à lire et à comprendre. Tout cela permet une meilleure maintenabilité du projet

tout entier.

La dernière étape permet de vérifier qu'une documentation est fournie pour décrire la fonctionnalité. C'est grâce à ce soucis permanent que la documentation de PostgreSQL (en anglais, et en français) est si complète.

10 La communauté grandit

- Les versions s'enchaînent
- Environ une tous les 10/12 mois
- La communauté des utilisateurs grandit notamment avec la version 8.0
- Celle des développeurs grandit elle-aussi

Une fois que la plateforme de développement a été mise en place, les versions se sont régulièrement enchaînées, au rythme de une par an. La communauté a grandi au fil des versions. Un grand cap a été passé lors de la sortie de la version 8.0. Elle combinait à la fois une version native Windows et des fonctionnalités attendues par les entreprises comme les tablespaces et le rejeu des journaux de transactions.

Avec les utilisateurs viennent aussi les contributeurs, et donc les développeurs. Le nombre de fonctionnalités par version accroît. Le nombre de plateformes supportés aussi. PostgreSQL devient un système de plus en plus complexe et complet.

11 Les problèmes commencent

- Plus de développeurs, avec une connaissance variée du code

- Nombreuses plateformes supportées

Le fait d'avoir un grand nombre de nouveaux développeurs n'a pas que des avantages. Les nouveaux connaissent moins le code et leur patch peut très bien fonctionner sur la partie qu'ils souhaitaient améliorer et avoir un impact très négatif sur une autre partie. Les anciens développeurs sont heureusement là pour vérifier les patchs, s'assurer des impacts qu'ils peuvent avoir, conseiller les nouveaux développeurs.

Le support de nouvelles plateformes apporte aussi un gros inconvénient. Un patch fonctionnant sur une plateforme peut très bien ne plus fonctionner sur un autre. Cela demande beaucoup de tests et la majorité des développeurs n'ont pas toutes les plateformes en question sous la main.

12 Solution apportée: la ferme de construction

- Composée d'un grand nombre de machines (40)
- Avec tous les systèmes supportés par PostgreSQL
- Avec différents compilateurs
- Avec différentes distributions
- Compilation avec différentes options

Une solution va être apportée par Andrew Dunstan. Récupérer un tas de machines, leur installer différents systèmes d'exploitation, avec différents compilateurs. Il crée un outil qu permet d'automatiser la compilation, l'installation et la vérification d'un certain nombre de tests. Cet outil est installé sur tous les nœuds de ce qu'on va appeler la ferme de construction. Il va aussi créer une application web qui permet de connaître l'état des derniers tests effectués par les nœuds de la ferme de construction. Voici deux copies d'écran de la ferme:

PostgreSQL BuildFarm

Home | Status | Members | Register | PGFoundry | PostgreSQL.org

PostgreSQL BuildFarm Status

Shown here is the latest status of each farm member for each branch it has reported on in the last 30 days.

Use the farm member link for history of that member on the relevant branch.

Legend	= cassert	= debug	= git	= integer-datetimes	= krb5	= nls	= openssl
	= pam	= perl	= python	[tcl] = tcl	= thread-safety	= vpath	= xml

Branch: HEAD			
Alias	System	Status	Flags
wombat	Gentoo 1.12.9 gcc 4.1.1 ppc64 PPC970MP	04:51:07 ago OK Config	
grebe	AIX 5.3 GCC 4.0.1 PPC	07:18:54 ago OK Config	
sloth	Slackware 13.0 gcc 4.3.3 i686	09:51:08 ago OK Config	
baiji	Vista Ultimate 6.0.6000 MSVC 2005 Pro 8.0.50727.867 x86	16:21:07 ago OK Config	
caracara	Ubuntu 6.06 GCC 4.0.3 IA-32	16:26:37 ago OK Config	
dawn_bat	Windows XP-Pro SP2 gcc 3.4.2 i386	16:51:05 ago OK Config	
koi	unixware 7.1.4 sco cc 4.2 bi-xeon HT	17:15:08 ago OK Config	

PostgreSQL BuildFarm

Home | Status | Members | Register | PGFoundry

PostgreSQL Build Farm Log

System Information	Farm member	Branch	OS	Compiler	Architecture	Owner
	caracara	HEAD	Ubuntu 6.06	GCC 4.0.3	IA-32	cbbrowne [a t] ca.afilias.info

Stage Logs						
	SCM-checkout	config	configure	make	check	make-contrib
	make-install	install-contrib	initdb-C	startdb-C-1	install-check-C	stopdb-C-1
	startdb-C-2	pl-install-check-C	stopdb-C-2	startdb-C-3	contrib-install-check-C	stopdb-C-3
	ecpg-check					

Configuration summary for system "caracara"

Status 'OK' on snapshot taken 2010-10-01 165432

```
This file was created by PostgreSQL configure 9.1devel, which was
generated by GNU Autoconf 2.63.  Invocation command line was

  $ ./configure --enable-cassert --enable-debug --enable-nls --enable-integer-datetimes \
        --with-perl --with-python --with-openssl --with-ldap \
        --with-libxml --with-libxslt --prefix=/opt/build-farm/BUILDS/HEAD/inst \
        --with-pgport=5699


hostname = cbbrowne-laptop.afilias-int.info
uname -m = i686
uname -r = 2.6.24-28-386
uname -s = Linux
uname -v = #1 Thu Sep 16 14:20:17 UTC 2010


PATH: /home/cbbrowne/bin
PATH: /usr/local/bin
PATH: /opt/dbs/pgsql74
PATH: /home/cbbrowne/bin
PATH: /usr/local/bin
PATH: /usr/local/sbin
PATH: /usr/local/bin
```

CC BY NC SA

13 De nouveaux problèmes apparaissent

- De plus en plus de patchs
- Mais un nombre de commiters qui évoluent peu
- Du coup, la liste d'attente des patchs grossit:
  - Les patchs sont vérifiés avec beaucoup de retard
  - Frustration des développeurs
  - La sortie des versions finales se trouve repoussée
  - Frustration des utilisateurs et des commiters

Bien qu'il y ait de plus en plus de développeurs, le nombre de commiters n'a pas augmenté. La vérification des patchs est devenue plus longue. De plus, la complexité des patchs s'est accrue. C'est généralement bon signe mais il faut toujours avoir des contributeurs d'assez longue date, prêt à s'investir dans la relecture des patchs, job important mais peu motivant. Du coup, beaucoup de développeurs avaient du mal à avoir des retours sur leur patch. Lorsqu'ils avaient un retour, c'était souvent bien après l'écriture du patch. Cela complique le travail du développeur. Il doit se rappeler de son patch, trouver du temps pour corriger les erreurs éventuelles. Tout ceci a fini par aboutir à une grande frustration des développeurs. Les commiters n'étaient pas en reste car ils avaient au final un énorme travail à fournir avant la sortie de la première beta d'une version finale. Quant aux utilisateurs, ils n'étaient pas non plus heureux, voyant que la sortie de la prochaine version était souvent repoussée. Des délais de deux à quatre mois ont été observés.

14 Première solution: le commitfest

- Dès le développement de la 8.3
- À chaque développement d'un patch, ajout d'une entrée dans le wiki
- Des relecteurs de patch travaillent tous les deux mois pour vider la queue de patchs
- La vérification des patchs est plus rapide
- Mais les gros patchs font peur

Lors du développement de la version 8.3, il a été décidé de découper le cycle de développement en plusieurs parties séparées chacune par deux mois. Chaque fois qu'un développeur avait terminé un patch, il l'envoyait sur la liste de discussion et ajoutait une entrée dans une page spécifique du wiki.

Au bout des deux mois, les contributeurs volontaires se répartissent les patchs. Ils travaillent le temps nécessaire pour vider la queue.

Le gros plus de cette solution, c'est de fournir un retour rapide au développeur qui ensuite a du temps pour corriger son patch si nécessaire. Ça a été aussi profitable aux commiters car leur travail a été allégé. En effet, la première relecture du patch fait par un non commiter permet déjà de faire un tri dans les patchs fournis. Du coup, les commiters n'ont plus qu'à vérifier les patchs qui sont vraiment dans un bon état.

Bien que cette solution a amélioré l'organisation du développement, il s'est avéré qu'un problème restait : les gros patchs ont souvent fait peur aux relecteurs qui les ont laissé à la fin, causant une fois encore un engorgement et un délai dans la sortie de la version.

15 Améliorations du commitfest

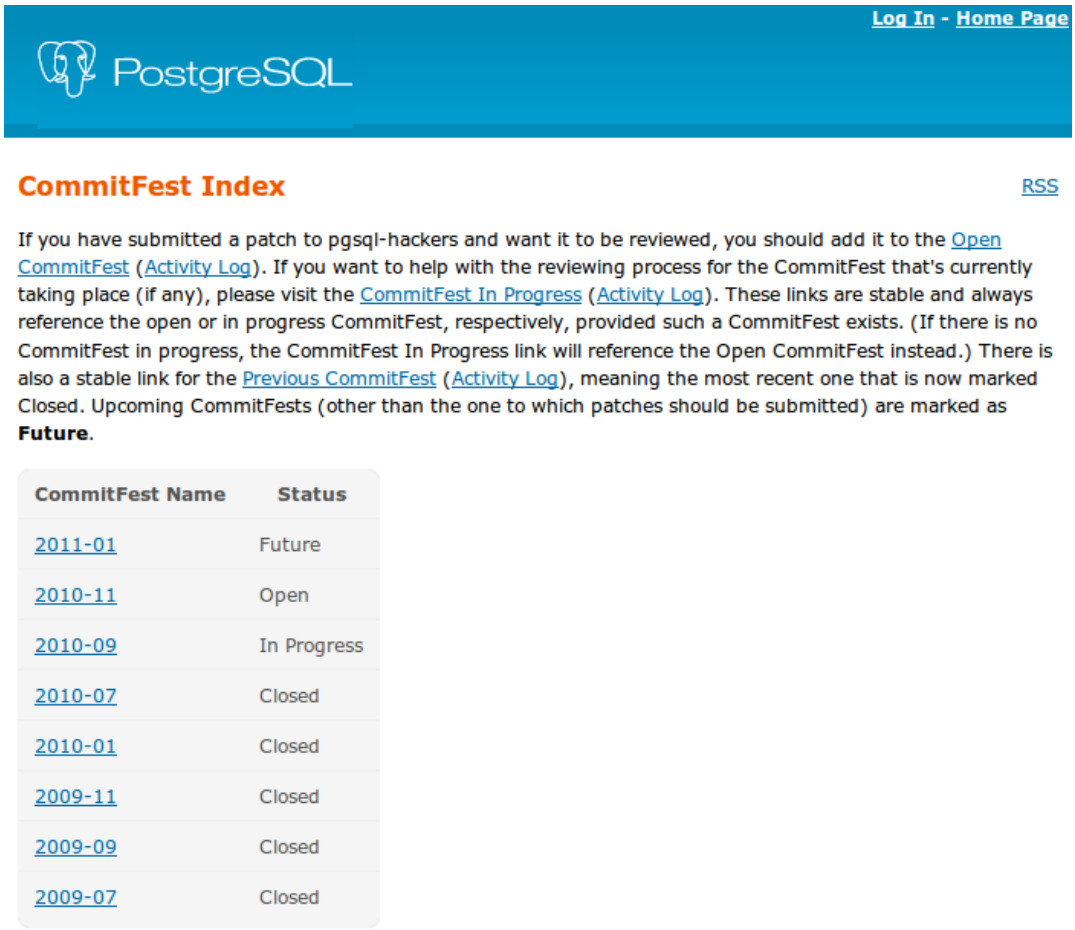
- Pour le développement de la 8.4
  - Ajout d'un responsable de commitfest
  - Création d'application web commitfest
- Pour le développement de la 9.0
  - Packaging de versions alpha

Le commitfest est une bonne solution, tout le monde en est convaincu à la fin du cycle de développement de la 8.3, mais des améliorations peuvent certainement être apportées à ce modèle d'organisation.

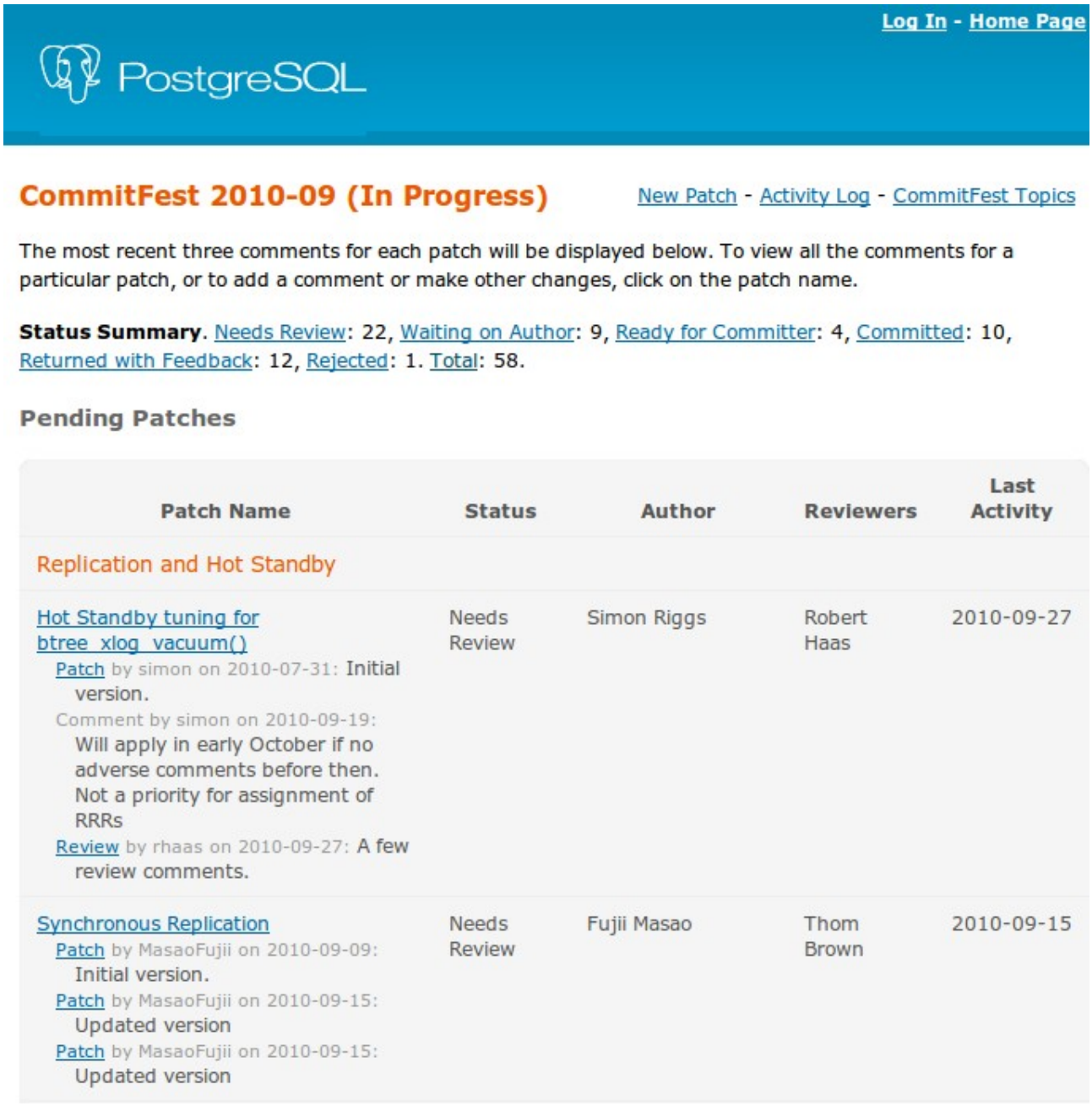
Pour le développement de la 8.4, un responsable de commitfest a été ajouté. Ce dernier a pour but de s'assurer que tous les patchs sont fournis aux relecteurs, que les relecteurs travaillent bien sur leur patchs, que les auteurs des patchs répondent bien aux questions des relecteurs. Bref, il est là pour aider au bon déroulement du commitfest. Plusieurs membres de la communauté se succèdent à ce poste pour que les compétences nécessaires à ce type de travail ne soient pas entre les mains d'un seul membre.

Toujours pour la 8.4, une application web est développée pour accueillir les informations sur les patchs. Elle remplace donc le wiki, pour le plus grand bonheur des développeurs et relecteurs. Elle facilite la visualisation de ce qu'il reste à faire, de ce qui a été fait.

Cette première copie d'écran montre la liste des commitfest, avec leur état (terminé, en cours, ouvert, futur) :



Les deux copies d'écran suivantes montre le détail sur un commit fest, avec un décompte des patchs suivant leur état, ainsi que la liste des patchs avec auteur, relecteur, état, etc.



Log In - Home Page

PostgreSQL

CommitFest 2010-09 (In Progress)

New Patch - Activity Log - CommitFest Topics

The most recent three comments for each patch will be displayed below. To view all the comments for a particular patch, or to add a comment or make other changes, click on the patch name.

Status Summary. Needs Review: 22, Waiting on Author: 9, Ready for Committer: 4, Committed: 10, Returned with Feedback: 12, Rejected: 1. Total: 58.

Pending Patches

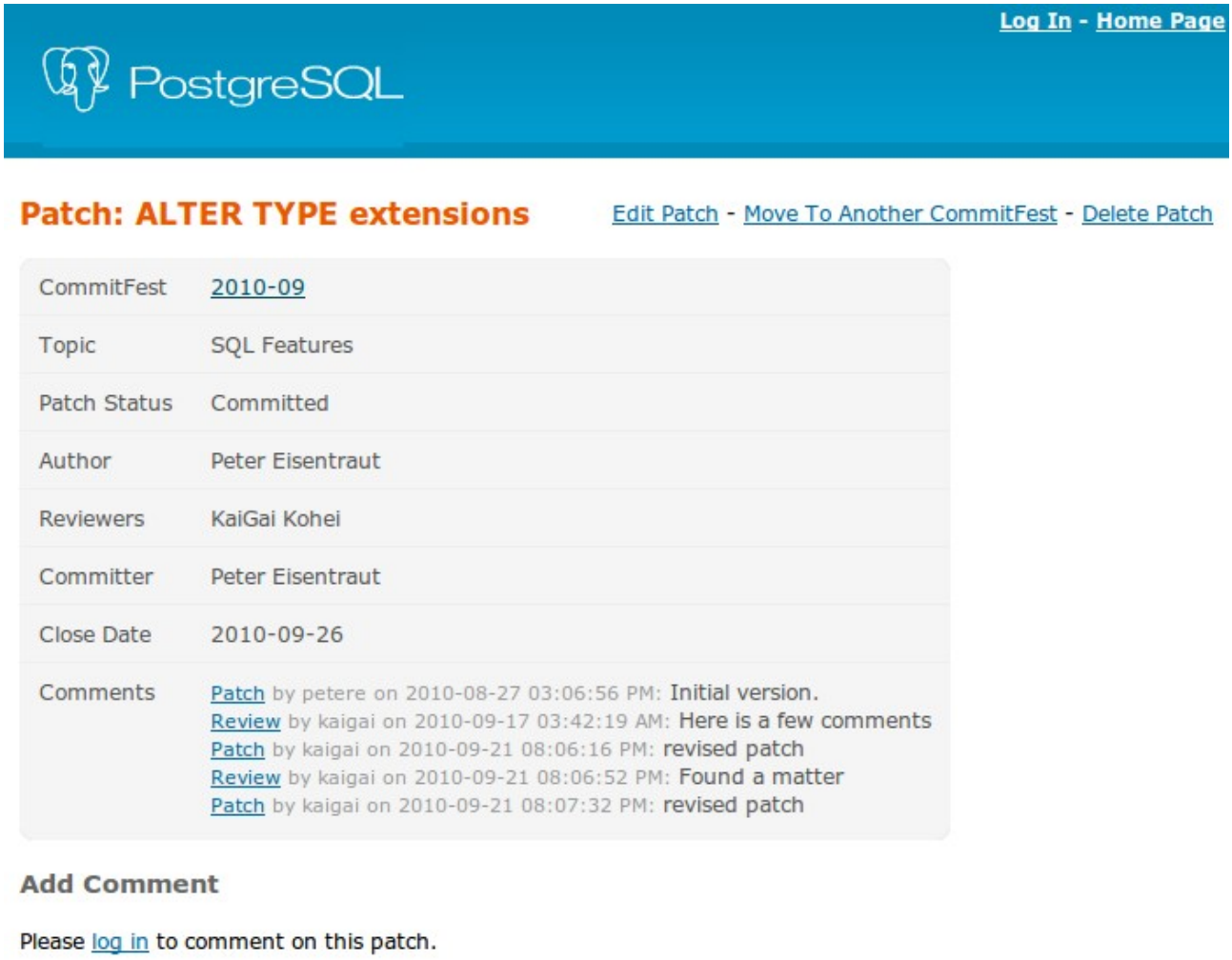
Patch Name	Status	Author	Reviewers	Last Activity
Replication and Hot Standby				
Hot Standby tuning for btree_xlog_vacuum() Patch by simon on 2010-07-31: Initial version. Comment by simon on 2010-09-19: Will apply in early October if no adverse comments before then. Not a priority for assignment of RRRs Review by rhaas on 2010-09-27: A few review comments.	Needs Review	Simon Riggs	Robert Haas	2010-09-27
Synchronous Replication Patch by MasaoFujii on 2010-09-09: Initial version. Patch by MasaoFujii on 2010-09-15: Updated version Patch by MasaoFujii on 2010-09-15: Updated version	Needs Review	Fujii Masao	Thom Brown	2010-09-15

CC BY NC SA

Committed Patches

Patch Name	Status	Author	Reviewers	Date Closed
utf8_to_unicode Patch by joeyadams on 2010-07-24: Initial version. Patch by joeyadams on 2010-08-13: Variation of initial patch with utf8_to_unicode in src/port	Committed	Joey Adams	Nobody	2010-08-18
Vacuum and analyze counters for pg_stat_*_tables Patch by mha on 2010-08-20: Initial version.	Committed	Magnus Hagander	Tom Lane	2010-08-21

Et voici le détail d'un patch :



Au début du développement de la version 9.0, il a aussi été décidé de créer des versions alpha. Ces versions permettent à des utilisateurs avancées de pouvoir tester certaines des nouvelles fonctionnalités sans avoir à récupérer les derniers sources à partir du dépôt.

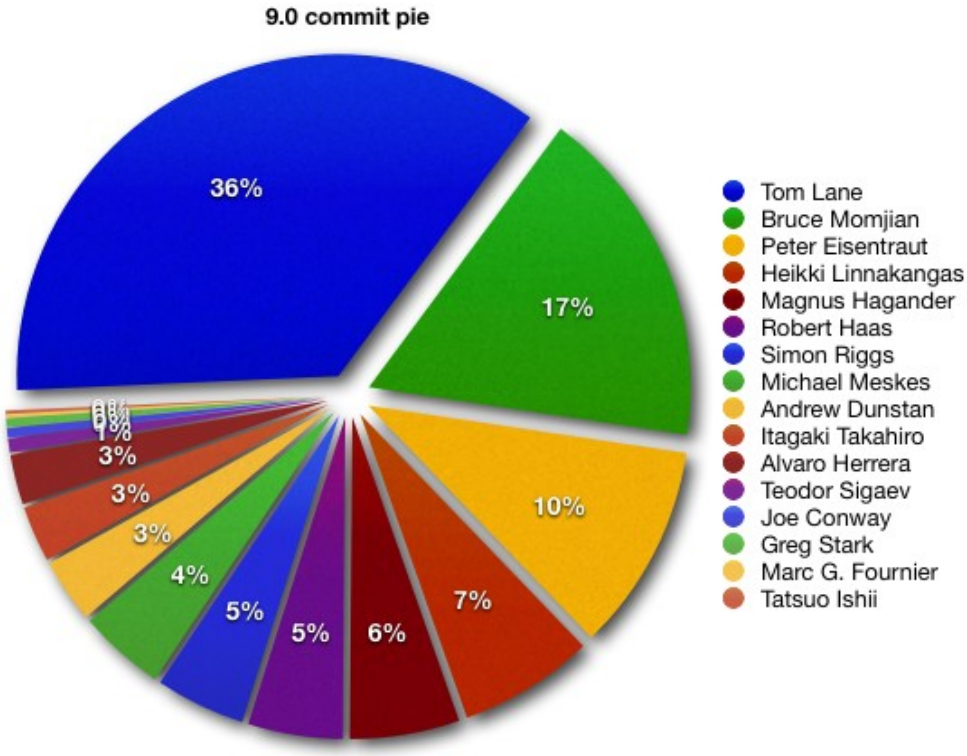
CC BY NC SA

16 Deuxième solution: plus de commiters

- 4 commiters ajoutés pendant le développement de la 9.0
- Un total de 19 commiters

Avec les nouveaux committers, le nombre de personnes pouvant modifier le code source de PostgreSQL arrive à 19 (http://wiki.postgresql.org/wiki/Committers). Cela a assez peu changé la répartition des commit dans le dépôt des sources. Néanmoins, cela permet d'alléger le travail aux anciens commiters.

Selena Deckelmann a fait une étude rapide sur les commiters pour les versions 8.4 et 9.0. Le détail se trouve sur son blog (http://www.chesnok.com/daily/2010/09/21/postgresql-9-0-contributions/), mais voici un graphe qui en est tiré :



CC BY NC SA

On peut facilement voir que Tom Lane et Bruce Momjian sont toujours les commiters les plus actifs du projet. Peter Eisentraut, Magnus Hagander et Heikki Linnakangas les suivent de plus loin. Les nouveaux commiters, comme Robert Haas, Simon Riggs et Itagaki Takahiro sont loin derrière.

Néanmoins, cela montre aussi que le projet est ouvert aux nouveaux contributeurs. Cela diminue aussi le risque pris par le projet de voir certains de ses contributeurs importants partir vers d'autres horizons.

17 Encore des problèmes

- Ferme de construction pour s'assurer d'un minimum de fonctionnalités
- Mais rien pour les performances

La ferme de construction a un avantage indéniable : prévenir très rapidement les développeurs si jamais un patch nouvellement enregistré dans le dépôt pose problème avec certains systèmes d'exploitation exotiques. Cependant, il existe un autre type de problème pour lequel la ferme de construction ne peut rien faire : les régressions de performance. Et cela commence à devenir un problème assez important.

18 Solution apportée: la ferme de performances

- Performance Farm
- Toujours en cours de mise en place

La solution assez logique à ce problème est de mettre en place une ferme de performances. Quelques contributeurs, dont Greg Smith, travaillent à préparer une ferme de ce type. Rien n'est encore fait.

19 Deuxième partie

Résumé sur les outils

Cette deuxième partie résume les outils actuellement utilisés par le projet PostgreSQL. Des outils pour les développeurs aux outils pour les testeurs.

20 Outils de compilation

- Langage C principalement
- Toolchain linux standard
- Toolchain Windows: Visual C++

PostgreSQL est écrit en langage C et essaie de respecter au mieux le C ISO/ANSI (au minimum compatible avec C89).

Les outils libres habituels sont utilisés sous Linux : make, gcc, gdb, etc. Ensuite tout dépend de la plateforme. Sous Windows, ce sera soit du mingw soit du Visual C++ 2008, ce dernier étant fortement conseillé.

CC BY NC SA

21 Outil pour le dépôt des sources

- CVS à l'origine et jusqu'en 2010
- Passerelle CVS-Git pendant deux ans
- Passage à Git mi-septembre 2010

Le projet PostgreSQL est très longtemps resté fidèle à son système de gestion de sources, CVS. Il a fallu une très grande patience et beaucoup de pédagogie à certains développeurs pour faire accepter le passage à Git.

Le passage à Git n'a pas été simple. Une première tentative a eu lieu en août 2010 et a dû être abandonné à cause de problèmes suite à la migration. Michael Haggerty et Max Bowsher ont énormément aidé à comprendre les problèmes de migration et ont fourni plusieurs patchs pour leur outil. Un grand nombre de tests ont eu lieu et ont permis la préparation d'une nouvelle tentative. Mi-septembre, une deuxième tentative a eu lieu et cette dernière a réussi.

Ce conservatisme observé chez les développeurs est essentiel au projet. Il permet d'éviter les technologies “du moment” pour se concentrer sur le travail réel. Toutes les questions de conservation de l'historique sont très importantes car c'est ce qui permet au projet de maintenir un grand nombre de versions sur plusieurs années.

22 Outil de suivi des bugs

- Aucun bugtracker pour PostgreSQL
- pgsql-bugs pour les bugs
- pgsql-* pour les demandes de fonctionnalités

PostgreSQL n'a jamais eu de bugtracker. Le problème soulevé par ces outils est une activité supplémentaire importante de validation et de tri des rapports.

Les bugs découverts peuvent être envoyés sur la liste pgsql-bugs ou sur toute autre liste.

Les demandes de fonctionnalités font plutôt partie des discussions normales. Les développeurs peuvent valider certaines demandes, plus ou moins explicites, en les enregistrant dans un document appelé TODO list (http://wiki.postgresql.org/wiki/Todo).

23 Outils de discussion

- Toute discussion passe par les listes
- pgsql-hackers pour le développement
- mais aussi les pgsql-*

L'âme de PostgreSQL, sa communauté, se retrouve en permanence sur les listes de discussion. Certains discutent aussi par IM (Jabber par exemple) ou par IRC, mais beaucoup plus rarement. Certains développeurs, y compris les plus importants, ne sont jamais sur IRC. Et comme les réunions physiques sont peu pratiques vu le côté international du projet, tout débat dans la communauté doit se faire sur les listes.

À noter quand même une réunion des développeurs par an, lors du pgCon, un événement PostgreSQL qui a lieu au Canada (http://www.pgcon.org/2010/). Le choix du Canada n'est pas innocent car il permet à plus de personnes de venir (à opposer par exemple aux États-Unis où l'accès au territoire est beaucoup plus contrôlé). Voici par exemple le compte-rendu de la dernière réunion: http://wiki.postgresql.org/wiki/PgCon_2010_Developer_Meeting.

Ce qui concerne le développement de nouvelles fonctionnalités ou les corrections de bugs est discuté sur la liste pgsql-hackers, mais certaines discussions ont aussi lieu sur les autres listes.

24 Outil commitfest

CC BY NC SA

- Application web commitfest.postgresql.org
- Écrit par Robert Haas
- Langage PHP, base de données PostgreSQL

Cet outil devient de plus en plus important pour le projet.

25 Outil BuildFarm

- Application web de présentation des résultats
- Client perl permettant de créer un nœud de compilation
- Écrit par Andrew Dunstan
- Maintenu par le même

Cette application existe depuis bien longtemps. Elle permet le test de la compilation, de l'installation et d'un ensemble d'actions sur de nombreuses plateformes et systèmes d'exploitation.

Le passage à Git a nécessité une mise à jour de l'outil qui s'est passé très correctement.

26 Troisième partie

Les différentes étapes d'une mise à jour

Les étapes d'une mise à jour sont scrupuleusement respectées par le projet. Les versions mineures comme les versions majeures sont construites avec la même rigueur, même si les étapes ne sont pas du tout identiques. L'objet de cette troisième partie est donc de détailler les différentes étapes suivies pour la mise en place d'une version majeure et d'une version mineure.

27 Deux types de versions

- Mineure/Corrective: correction de bugs et de failles de sécurité
- Majeure: nouvelles fonctionnalités

Le project PostgreSQL fait preuve d'une très grande retenue quant aux modifications des anciennes versions. Toute version stable impactée par un bug sera corrigée. Par contre, aucun ajout de fonctionnalité ne se fera sur les anciennes branches stables. Autrement dit, une version mineure ne contiendra que des corrections de bug et de faille de sécurité.

28 Correction d'un bug

- Validation qu'il s'agit bien d'un bug
- Correction sur la version indiquée
- Puis tests des différentes versions

- Et correction de celles-ci si nécessaire

Tout rapport de bug est traité. Tout d'abord, les développeurs vérifient qu'il s'agit bien d'un bug. Ensuite, ils corrigent le bug sur la version indiquée par l'utilisateur qui a rapporté le bug. Le bug est aussi testé sur les autres versions stables maintenues par le projet. Si nécessaire, des corrections sont apportées à ces versions.

29 Sortie de version mineure

- Lot de toutes les versions stables
- Tous les 2/3 mois
- Sauf si bug majeur ou faille de sécurité

Il n'y a pas à proprement parler de développement d'une version mineure. Cette dernière ne dispose que de correctifs. Donc son évolution dépend du retour des utilisateurs. Plusieurs bugs peuvent être corrigés sans qu'il y ait de nouvelles versions mineures officielles.

Au bout de 2 à 3 mois, les développeurs commencent à préparer la sortie d'une nouvelle suite de versions mineures. Dans le cas où un bug majeur est apparu ou si une faille de sécurité est découverte, des versions mineures sont sorties immédiatement et tous les utilisateurs sont priés de mettre à jour leur version.

30 Sortie de version majeure

- Tous les 12/15 mois
- Tout de suite après, création d'une branche
- Le développement de la prochaine majeure peut commencer
- sauf pour la 9.1 où le développement a commencé avant la sortie de la 9.0

Le travail pour une version majeure est bien plus complexe. Les développeurs travaillent sur des patchs pour des fonctionnalités qu'ils veulent avoir. La version de développement va passer par toute une série de commitfest, puis une version beta va sortir. D'autres sortiront jusqu'à arriver à la version RC, puis à la version finale. À ce moment-là, les développeurs créent une nouvelle branche et le développament de nouvelles fonctionnalités va pouvoir recommencer.

31 Cycle de vie d'un patch

- Discussion
  - Identification du problème
  - Recherche d'une solution
- Codage
- Patch soumis dans la queue
- Revue du patch
- Correction du patch si nécessaire
- Intégration du patch dans les sources

Avant tout codage, il est essentiel de discuter du problème qu'on cherche à éliminer ou de la nouvelle fonctionnalité qu'on cherche à implémenter. Cette discussion a généralement lieu sur pgsql-hackers, la liste de discussion des développeurs de PostgreSQL. Une fois que le problème a

été bien identifié, commence la recherche d'une solution. Toute cette partie peut prendre du temps, avec des pistes prometteuses qui finiront par être abandonnées et des pistes moins tentantes mais qui pourraient être la seule solution.

Ceci fait, le codage peut commencer. C'est généralement l'opération la moins longue, mais tout dépend évidemment du problème ou de la fonctionnalité.

Le développeur content de son patch l'envoie sur la liste de discussion pgsql-hackers puis le met sur le site des commitfest.

Lors du prochain commitfest, le patch est relu, rediscuté, voire amélioré. Si nécessaire, le relecteur le renverra au développeur pour corrections. Sinon, il intègrera la queue des patchs en attente d'un commiter pour intégrer le patch dans les sources.

32 Versions alpha

- Après chaque commitfest
- Permet aux utilisateurs de tester les nouveautés sans avoir à récupérer le Git
- Permet aux développeurs de savoir si un bug d'une certaine alpha a déjà été corrigé… ou pas

À la fin de chaque commitfest, un tag est ajouté dans le dépôt et un package des sources est créé. Aucun package binaire ne sera généré. L'idée est de permettre aux utilisateurs de tester une version tagguée, ce qui facilite ensuite le travail des développeurs sur les bugs rencontrés.

33 Feature freeze

- Peut être très long

- Vérification des derniers patchs
- Debugage intensif
- Début du travail pour les traducteurs
- et pour les promoteurs

Après 4 à 5 versions alpha, il est temps d'arrêter les développements de nouvelles fonctionnalités et de se concentrer sur la stabilisation de la version. Les derniers patchs sont vérifiés et intégrés dans la mesure du possible. Le debugage commence de façon intensive.

C'est aussi le moment où les traducteurs d'applications vont commencer à travailler. De même, le kit de presse commence à être rédigé.

34 Versions Beta

- Aucune fonctionnalité ajoutée
- Intégration des traductions
- Nécessite de créer un paquet des sources
- et de packager certaines versions binaires
- Effort commun des utilisateurs et des développeurs
  - Tests des fonctionnalités
  - Tests de la compilation
  - Test des performances

Une fois tous les patchs possibles intégrés et les bugs majeurs corrigés, une première version beta sort. Elle intègre les traductions déjà disponibles. Des paquets source et binaires (Windows et Mac principalement) sont disponibles.

Commence un gros effort commun des utilisateurs et des développeurs pour tester les

fonctionnalités, tester les performances et s'assurer qu'il n'y a pas de régressions.

35 Release Candidate

- Uniquement quand on est certain qu'il n'y a plus de bugs
- Dure assez peu

Après quelques versions beta, après une période d'accalmie au niveau des bugs, une version Release Candidate sort. La période des RC est généralement très courte et les RC sont peu nombreuses.

36 J-5: les packages

- Tag dans le dépôt
- Construction du paquet des sources
- Envoi de l'information aux différents packageurs
- Préparation des paquets binaires

Cinq jours avant la sortie officielle de la version majeure (ou des versions mineures), un tag est ajouté dans le dépôt. Un paquet des sources est créé et les packageurs peuvent travailler sur leurs paquets binaires. L'idée des cinq jours est justement de laisser du temps aux packageurs pour préparer leur paquets. Au bout du délai, l'annonce officielle sera envoyée.

37 Jour de la sortie

- Kit de presse
- Annonce sur le site web et sur les listes de discussion

Le jour de la sortie officielle, l'annonce est envoyée sur différents canaux :

- le site web
- la liste de discussion -announce
- le canal IRC

Et évidemment quelques blogs de contributeurs qui relaieront l'information.

Le kit de presse est mis en place sur le site web. Les contacts régionaux sont prêts à faire face aux demandes des journalistes.

38 Quatrième partie

Les contributeurs

C'est certainement le point le plus important. Avoir des contributeurs est essentiel au projet, ne serait-ce que pour connaître les bugs trouvés par les utilisateurs et les demandes de nouvelles fonctionnalités. Nous allons donc détailler dans cette quatrième partie les différents contributeurs majeurs, ainsi que les types de contributeurs.

CC BY NC SA

39 Core Team

De gauche à droite et de haut en bas…

- Tom Lane
- Bruce Momjian
- Marc Fournier
- Dave Page
- Josh Berkus
- Peter Eisentraut

La Core-Team est un groupe de six personnes, dont le pouvoir de décision est très limité. Ils ont une aire de travail très contrainte. Néanmoins, leur passé de contributeur leur donne un poids très important sur les décisions que la communauté devra prendre.

Tom Lane est le commiter le plus important au projet. Il est un des rares à connaître suffisamment le code de PostgreSQL pour savoir qu'un patch modifiant telle partie du code aura

aussi des implications dans telle autre partie du code. Cette connaissance lui donne un pouvoir assez important quant à la décision d'intégrer ou non un patch. Il travaille chez RedHat.

Bruce Momjian est la personne qui a permis la libération du code source de PostgreSQL. Il a beaucoup œuvré sur le code au début et encore un peu maintenant. Il est notamment le développeur de pg_upgrade, un outil de plus en plus important au sein de la communauté. Il travaille chez EntrepriseDB.

Marc Fournier (celui pour qui nous n'avons pas de photo) s'est beaucoup occupé de l'administration globale des serveurs. Actuellement, il s'occupe uniquement des serveurs mails. Il travaille dans la même branche chez Hub.org.

Dave Page est le développeur principal de pgAdmin, un outil d'administration de bases de données PostgreSQL. Il s'occupe aussi beaucoup de l'administration des serveurs utilisés par la communauté. Il travaille chez EntrepriseDB.

Josh Berkus s'occupe de la promotion du logiciel. Il code très peu. Il travaille chez PGX, une société d'expertise sur PostgreSQL qu'il a créé aux États-Unis.

Peter Eisentraut fait partie du groupe des commiters. Il a travaillé sur un grand nombre de patchs, notamment pour l'internationalisation des applications. Dernièrement, il travaillait sur le protocole SQL/Med et s'investissait de plus en plus avec le comité SQL. Il travaille chez F-Prot Antivirus.

40 Responsabilité de la Core Team

- Planification des sorties de version
- Point de contact pour les problèmes de sécurité
- Prise de décision quand la communauté ne parvient pas au consensus

Les responsabilités de la Core Team sont très précisément définies. Ils décident de la sortie des versions, et ils sont le principal point de contact pour les problèmes de sécurité (failles). Ils n'ont aucun autre pouvoir de décision (y compris sur le contenu des versions), sauf dans le cas où la communauté n'arriverait pas à obtenir un consensus pour un problème relativement important.

41 Contributeurs au niveau code

- Le développeur
- Le relecteur
- Le commiter

Après la Core Team, nous avons les contributeurs au niveau code. On peut les distinguer suivant trois types :

- le développeur, qui écrit un patch et le fournit à la communauté
- le relecteur, qui relit un patch et soit le renvoie au développeur pour qu'il le corrige, soit le passe au commiter
- le commiteur, qui fait une deuxième relecture du patch, quelques vérifications supplémentaires et l'enregistre, si tout va bien, dans le dépôt des sources

Évidemment, beaucoup de contributeurs ne se limitent pas à un seul type. Beaucoup de développeurs sont aussi relecteurs, c'est même encouragé par le projet.

42 Autres contributeurs

- Tout aussi important
  - équipe d'administration des serveurs
  - groupe de promotion
  - conférenciers
  - traducteurs

PostgreSQL, ce n'est pas que du code. C'est aussi de la documentation, c'est aussi des conférences, des articles dans la presse, et un tas de serveurs à administrer… pour que les développeurs puissent travailler tranquillement et que les utilisateurs connaissent bien PostgreSQL.

Les autres contributeurs sont donc importants, voire essentiels pour que le projet soit mené à terme. Il n'est pas étonnant par exemple que la documentation de PostgreSQL est très appréciée car elle fait l'objet d'une attention pratiquement aussi maniaque que celle apportée au code.

43 Contributeur d'un projet libre: un hobby ?

- Certains le font sur leur temps libre
- Mais la majorité sont financés
  - soit par leur entreprise
  - soit pour un développement particulier

L'un des gros mythes du logiciel libre est que les logiciels libres sont créés par des amateurs, des dilettantes qui font ça pendant leur temps libre.

Ce n'est pas totalement faux. Cependant, il faut bien voir que la majorité des développements des gros logiciels libres est souvent effectuée par des personnes payées pour. Tout simplement parce qu'une entreprise a besoin d'une fonctionnalité particulière et demande à un de ses employés de se charger de la coder, soit parce que cette entreprise est prête à financer un développeur pour travailler sur cette fonctionnalité. Évidemment, cela n'assure en rien que la fonctionnalité sera ajoutée car seule la communauté décidra de son ajout… ou pas.

Bien que la plupart des développeurs de PostgreSQL développent pendant leur temps de travail, beaucoup (et parfois les mêmes) le font aussi sur leur temps libre. Ce sont tout simplement des passionnés.

44 Conclusion

- PostgreSQL est un projet vivant
- qui tente de trouver des solutions aux problèmes posés par son organisation
- qui intègre tout type de contributeur
- et qui récolte le fruit de ses efforts grâce à de nombreux contributeurs et utilisateurs heureux
- Projet phrare

CC BY NC SA