



 **DALIBO**

## Les pires pratiques PostgreSQL

8 juin 2017

**Thomas REISS**, *Consultant*  
**Philippe BEAUDOIN**, *Consultant*

**L'expertise PostgreSQL**

# Les pires pratiques ?

- Les bonnes pratiques, c'est ennuyeux !
- Les pires pratiques, un nouveau concept ... à essayer
- Inspiré d'une conférence d'Ylia Kosmodemiansky (Data Egret), et de cas réels que nous avons rencontrés



## 0 – N'utilisez pas d'index



- Fonctionnellement, scan de table et scan d'index sont équivalents
- Postgres ne s'en sert pas
  - ★ Essayez d'insérer 10 lignes dans une table vide et faites un EXPLAIN
- Avec les disques modernes, les accès sont très rapides



- Nombreux types d'index : *BTREE*, *GIN*, *GIST*, *BRIN*, *BLOOM*,...



# 1 – Créez autant d'index que vous pouvez



- C'est l'optimiseur qui sélectionnera les bons
- Ça ne prend presque pas de place sur disque et dans les *shared\_buffers*
- Les mises à jour de tables sont à peine plus longues
- Si les perf ne sont pas bonnes, il doit manquer des index
- => créez en d'autres



- Index partiels, index fonctionnels



## 2 – Faites les jointures dans l'application



- Quelques *SELECT \* FROM une\_table* sont faciles à coder et à maintenir
- Pas de risque que le SGBD n'oublie de retourner une donnée
- Et vous n'avez plus qu'à coder dans votre langage favori :
  - ★ *Nested Loop Join, Hash Join, Merge Join*
  - ★ ... et un optimiseur pour choisir le bon algorithme à utiliser



- Exploitez la richesse du SQL : *CTE, windowing functions, LATERAL, ...*
- Sans mettre toute l'application dans la base de données



### 3 – Soyez « agile », soyez « schema-less »



- Ne perdez plus de temps à faire un design de base
- Une seule table est suffisante pour contenir toutes les données
- ... avec 2 colonnes : *id bigserial*, *le\_rest jsonb*
- Une fois le bon index créé, vous pouvez y accéder comme dans des tables bien structurées



- Mixer le « très structuré » et le « peu structuré », en gardant les propriétés ACID
- Pour certaines données, utilisez des structures adaptées : *jsonb*, *hstore*



## 4 – Utilisez le type TEXT pour toutes vos colonnes



- Toute donnée peut se représenter comme du texte
- C'est très drôle de recoder une validation de date ou d'adresse ip dans les applications, en traitant tous les cas tordus
- Vous allez adorer les « 12-31-2016 03:01AM » traduit en
- ... « 15:01 12 of undef 2016 »



- Le typage de PostgreSQL est puissant
- Ex : remplacez 2 dates de début et fin par un *RANGE* et bénéficiez des contraintes d'exclusion



## 5 – Bannissez les Foreign Keys



- Vous n'avez pas confiance dans la capacité de vos applications à maintenir la cohérence des données ?
- Il faut laisser un peu de travail à vos équipes en charge de la qualité des données



- Pas que les clés étrangères :
  - ★ Les classiques : *PRIMARY KEY*, *NOT NULL*, *UNIQUE*
  - ★ Mais aussi : *CHECK*, *EXCLUSION*





## 6 – Fabriquez votre version améliorée de Postgres



- Postgres n'est pas parfait et ... vous êtes très intelligent
- La soumission des patchs à la communauté est long et fastidieux. Mettez vos améliorations directement en production.
- Il est très facile de garder la compatibilité avec les versions officielles



- Vous pouvez avoir confiance dans la communauté
- Mettez à jour vos versions
- Participez...



## 7 – Développez vos outils d'administration

- Les outils graphiques, ça fait mal aux yeux
- Quelques bonnes *regex* pour exploiter les log ?
- Pour la supervision, mieux vaut taper ses propres commandes `psql`
  - ★... toutes les 5 mn
  - ★... et sur chaque instance
  - ★... et demander à un collègue pour la nuit

- Bâissez votre socle d'administration à partir des nombreux logiciels OS disponibles



## 8 – Besoin de place ? Faites le ménage



- Les répertoires dont vous ne savez pas ce qu'il y a dedans sont certainement inutiles
- `rm -Rf pg_clog/* pg_xlog/*`
- Et s'il y a des messages d'erreur ensuite : `pg_resetxlog`



- Celle là, c'est la pire !
- Consultez les logs (archivage en panne, autre chose ?)
- Linux réserve 5 % de l'espace disque d'un FS pour root



## 9 – Désactivez l'autovacuum

- Les process de l'*autovacuum* prennent des ressources
- Facile à désactiver
- Ce n'est pas un problème d'avoir 100 Go de données dans une base de 1To
- Les To de mémoire ne sont pas chers et les I/O sont de plus en plus performantes
- Un moyen sûr d'aller vers le **BIG DATA**

- Eviter les très longues transactions
- Laissez faire *autovacuum*
- Sur des batchs, *VACUUM ANALYZE* fait partie du traitement



## 10 – Vous voulez vraiment sauvegarder vos bases ?



- Utilisez de la réplication à la place des sauvegardes
- *pg\_dump* pour les grosses bases
- Écrivez votre propre script, en combinant tous les outils externes que vous connaissez
- Inutile de tester les restaurations



- Des bons produits OS disponibles : *Pitrery, Barman, pgBackRest,...*



## On est bien d'accord ...

- Il s'agissait des PIRES pratiques
- Donc à ne PAS suivre

*Toute ressemblance avec des situations réelles n'est  
NI fortuite NI involontaire...*





Eclatez-vous avec PostgreSQL...