Slony-I

enterprise-level replication system

Réplications sophistiqués avec Slony

À propos des auteurs

- Auteur : Guillaume Lelarge
- Société : DALIBO
- Date : Février 2012
- URL: http://dalibo.org

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

Qu'est-ce que Slony?

- Système de réplication
- Asymétrique
  - Un maître, plusieurs esclaves
- Asynchrone
  - Les données modifiées sont envoyées aux esclaves avec un certain retard
- Basé sur les triggers
- Licence BSD

Pourquoi utiliser Slony ?

- Surtout maintenant qu'il y a la réplication interne !
- La réplication interne de PostgreSQL est
  - Fiable
  - Simple à mettre en place
  - Facile à superviser
  - Simple à comprendre
  - ... idéale ?

Quelques raisons pour utiliser Slony

- Trois raisons principales
  - Différentes versions de PostgreSQL sur le maître et les esclaves
  - Réplication d'une partie seulement d'une instance
  - Autres objets sur l'esclave
- Autres raisons
  - Réplication en cascade
  - Switchover bien plus simple
- Bref, Slony utile pour les cas plus sophistiqués

Quelques exemples

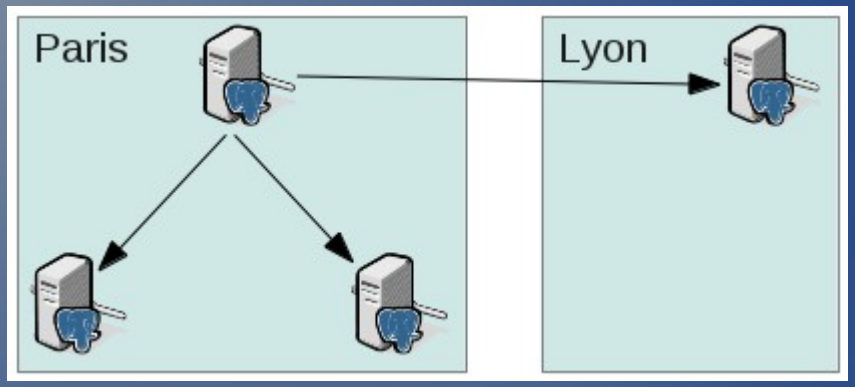
- Cascade
- Réplication croisée
- Répartition de charge
- Avec des objets supplémentaires sur l'esclave

Cascade

- Un centre à Paris, déjà redondé
  - Un maître
  - Deux esclaves
- Un nouveau centre à Lyon

Cascade – une solution

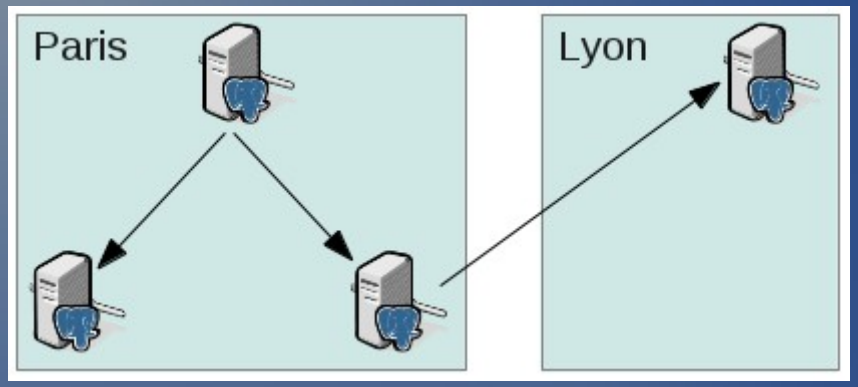
- Ajouter un esclave au maître



- Mais
  - charge supplémentaire au niveau du maître

Cascade – une meilleure solution

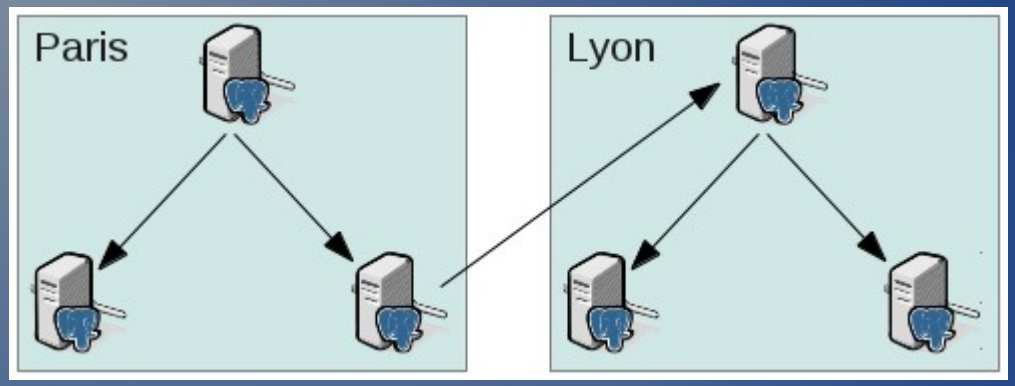
- Ajouter un esclave à l'esclave



- Décharge le maître
- Ne pose pas de problème pour un failover ou un switchover

Cascade – pour aller plus loin

- Ajoutons des esclaves aux esclaves

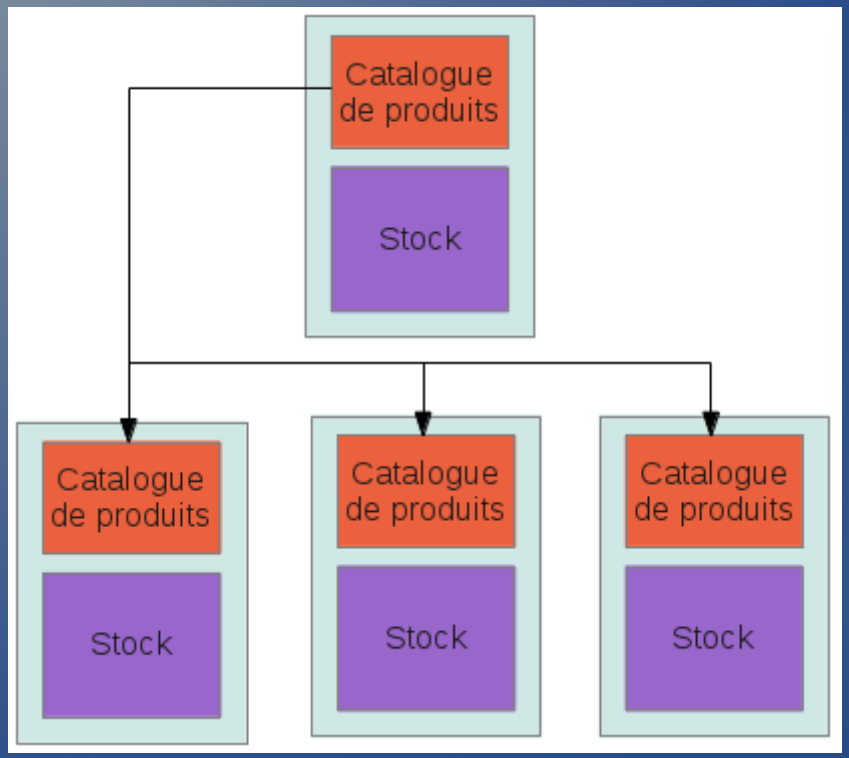


Réplication croisée

- Grande chaîne de magasins
- La maison mère gère le catalogue
  - Nouveaux produits
  - Changement de prix
- Les magasins gèrent leur stocks
- Les magasins ont besoin rapidement du changement tarifaire de la maison mère
- La maison mère a besoin de statistiques sur les stocks des magasins

Réplication croisée - catalogue

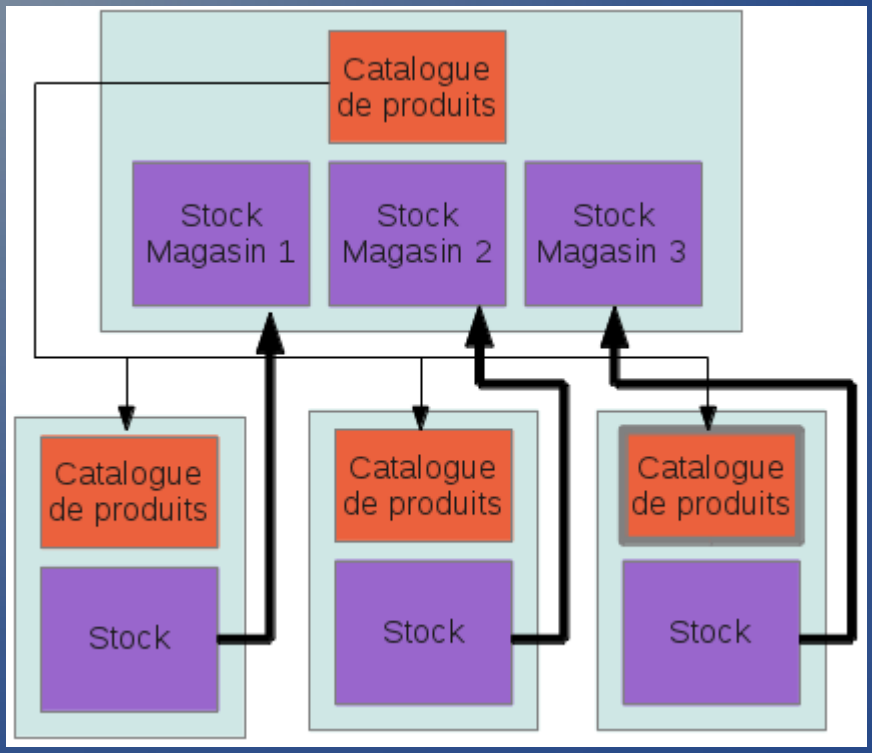
- Première action : partager le catalogue



- Toute modification arrive rapidement aux magasins

Réplication croisée - stocks

- Deuxième action : partager les stocks



- Tout changement dans les stocks arrivent pratiquement immédiatement au central

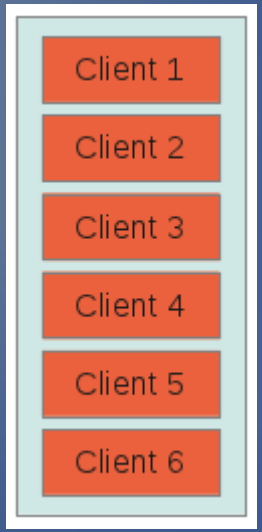
Répartition de charge

- Grand nombre de bases
- Plus ou moins volumineuses
- Plus ou moins chargés
- Toutes sur un serveur surchargé

- Répartition de charge demandée
- Mais sans pgPool

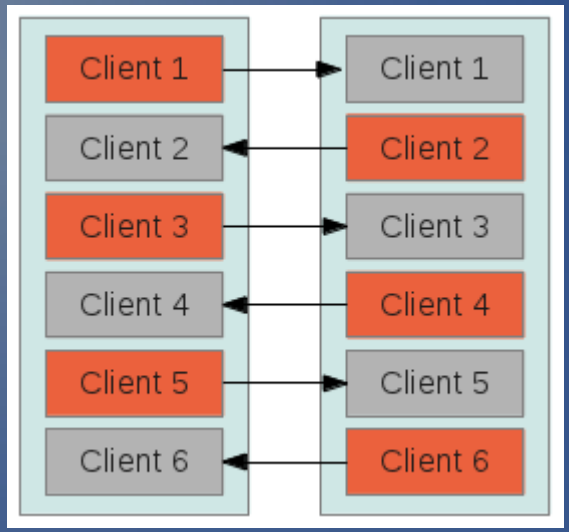
Répartition de charge – état initial

- Un serveur avec six bases de clients



Répartition de charge – 2 serveurs

- Ajouter un serveur



- 2 avantages
  - Répartition de charge
  - Et deuxième serveur disponible

Répartition de charge – autres avantages

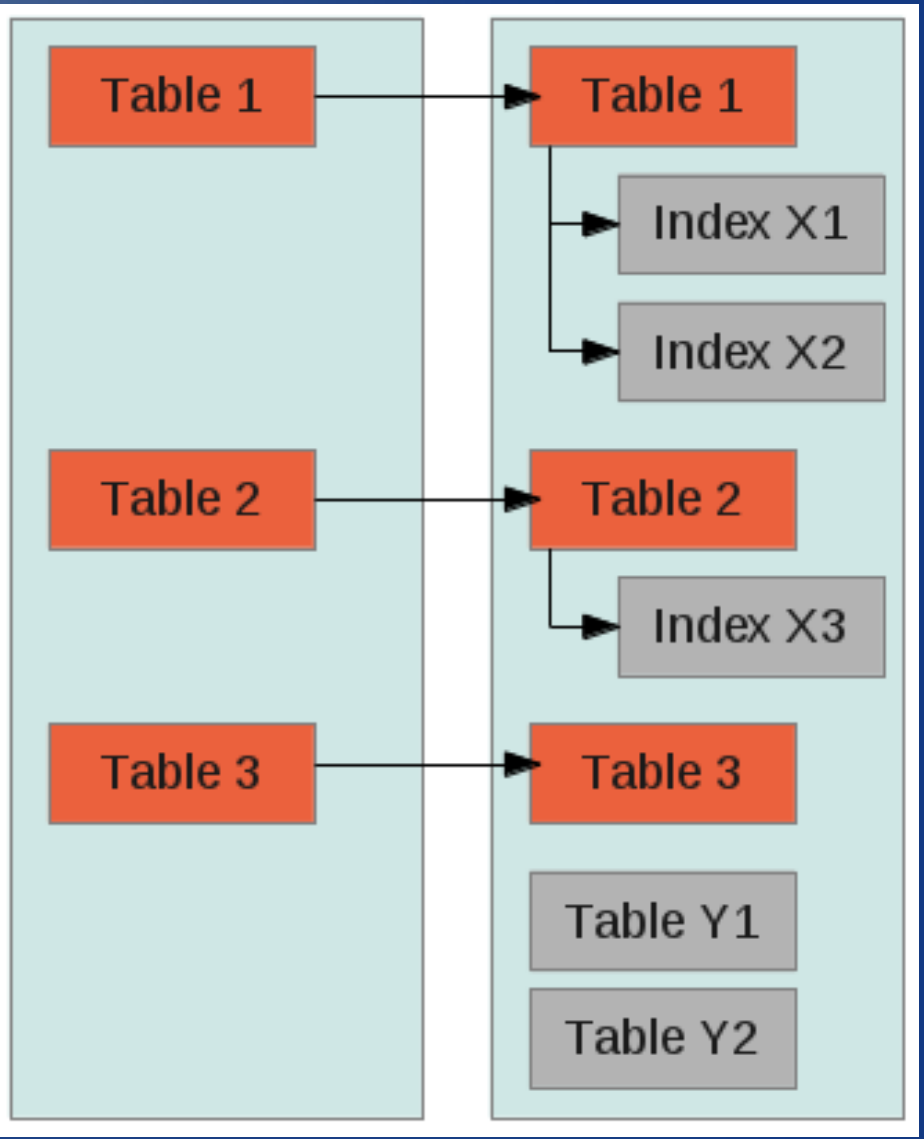
- Facilite les mises à jour majeures
- Facilite la maintenance d'un serveur
- L'intégration de pgBouncer améliore encore le système
  - Bascule plus simple
  - Avec moins de configuration

Autres objets sur l'esclave

- BI/Reporting
  - Ajout de tables de travail
  - Ajout d'index
- Statistiques
  - Ajout de trigger de calcul

Autres objets - exemple

- Réplication de trois tables du maître
- 3 index specifiques et deux tables de travail sur l'esclave



Conclusion

- Réplication interne de PostgreSQL
  - Parfaite pour des systèmes simples
- Réplication Slony à privilégier pour les systèmes complexes
  - Cascade
  - Réplication croisée
  - Ajout d'objets sur l'esclave
  - Etc.

Ce document a pour but de montrer les avantages de Slony en tant que système de réplication pour PostgreSQL.

Depuis l'implémentation de la réplication interne, beaucoup ont cru que Slony allait disparaître. Il est vrai que ce système est moins utilisé, au profit principalement de la réplication interne.

Néanmoins, de nombreux cas ne sont pas (encore ?) gérés par la réplication interne. Ce document va justement montrer quelques cas intéressants, profitant des fonctionnalités uniques de Slony.

À propos des auteurs

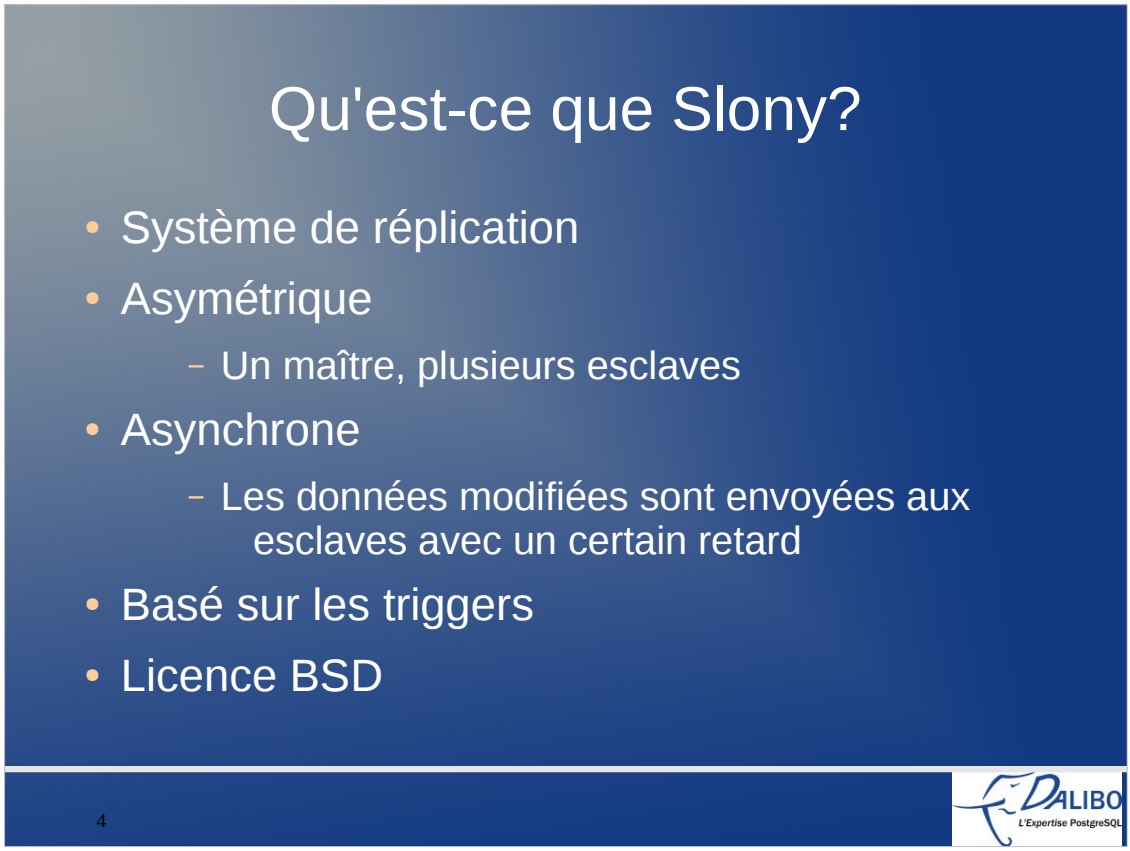
- Auteur : Guillaume Lelarge
- Société : DALIBO
- Date : Février 2012
- URL: http://dalibo.org

Ce document vous est fourni par la société Dalibo, spécialisé sur PostgreSQL. Guillaume Lelarge a écrit ce document en janvier 2012, pour donner cette conférence lors de la PostgreSQL Session #3 le 2 février 2012.

Ce document est disponible sur le site http://dalibo.org où se trouvent aussi de nombreux autres documents (articles de journaux, documents de conférences, etc.) disponibles librement.

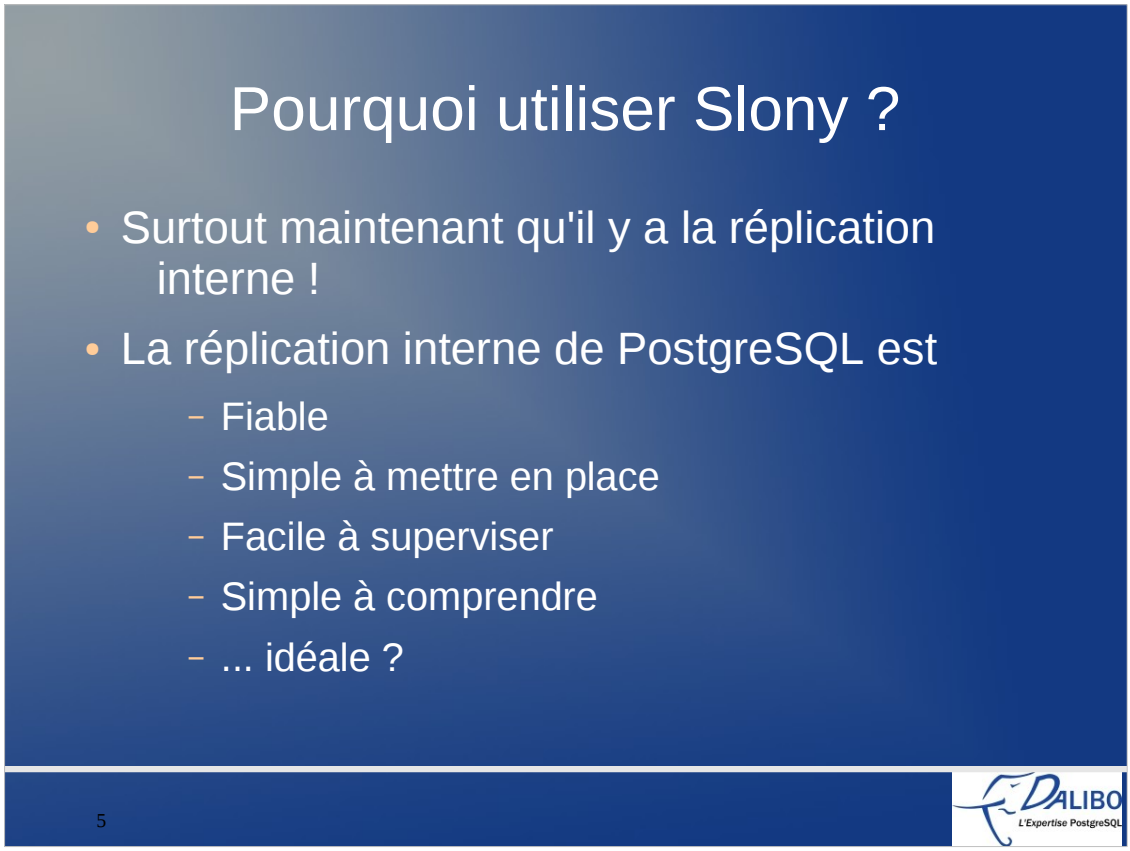
Licence

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



Slony est certainement le système de réplication le plus ancien pour PostgreSQL. Il a été écrit par la société Afilias pour ses besoins internes. Afilias a décidé de libérer le code source de ce produit très rapidement. La licence choisie est la licence BSD, comme PostgreSQL.

Techniquement, ce système permet de faire de la réplication asynchrone asymétrique. Il y a donc un seul maître et un ou plusieurs esclaves. Les changements de données sont capturées par des triggers qui conservent les informations le temps que tous les esclaves aient pu les récupérer. C'est donc un système qui résiste assez bien aux coupures de connexions entre le maître et les esclaves. C'est aussi un système robuste et performant.



Depuis que la réplication a été ajoutée aux déjà nombreuses fonctionnalités de PostgreSQL, la question qui est rapidement survenue est : que va-t-il arriver aux autres systèmes de réplication ?

En effet, la réplication interne est particulièrement simple à mettre en place. Depuis la version 9.1, il est même assez facile de la superviser et de l'administrer. Son fonctionnement est simple à comprendre.

Du coup, quel intérêt présentent les autres systèmes de réplication comme Slony ?

La réplication interne paraît idéale... Elle l'est. Pour les systèmes simples.

Quelques raisons pour utiliser Slony

- Trois raisons principales
  - Différentes versions de PostgreSQL sur le maître et les esclaves
  - Réplication d'une partie seulement d'une instance
  - Autres objets sur l'esclave
- Autres raisons
  - Réplication en cascade
  - Switchover bien plus simple
- Bref, Slony utile pour les cas plus sophistiqués

Plusieurs raisons amènent les administrateurs de bases de données à choisir Slony.

Il est possible d'utiliser un maître avec une certaine version majeure de PostgreSQL et un esclave avec une autre. Ceci est tout simplement impossible avec la réplication interne. Cela a même fait la réputation de Slony pour la mise à jour de version majeure sur des grosses volumétries.

Il est aussi possible de ne répliquer qu'une partie de la base. La réplication interne de PostgreSQL s'occupe de l'instance complète. Celle de Slony permet de choisir table par table ce qui va être répliqué.

Il est même possible de créer des objets supplémentaires sur l'esclave. Très utile pour les utilisateurs de systèmes de reporting.

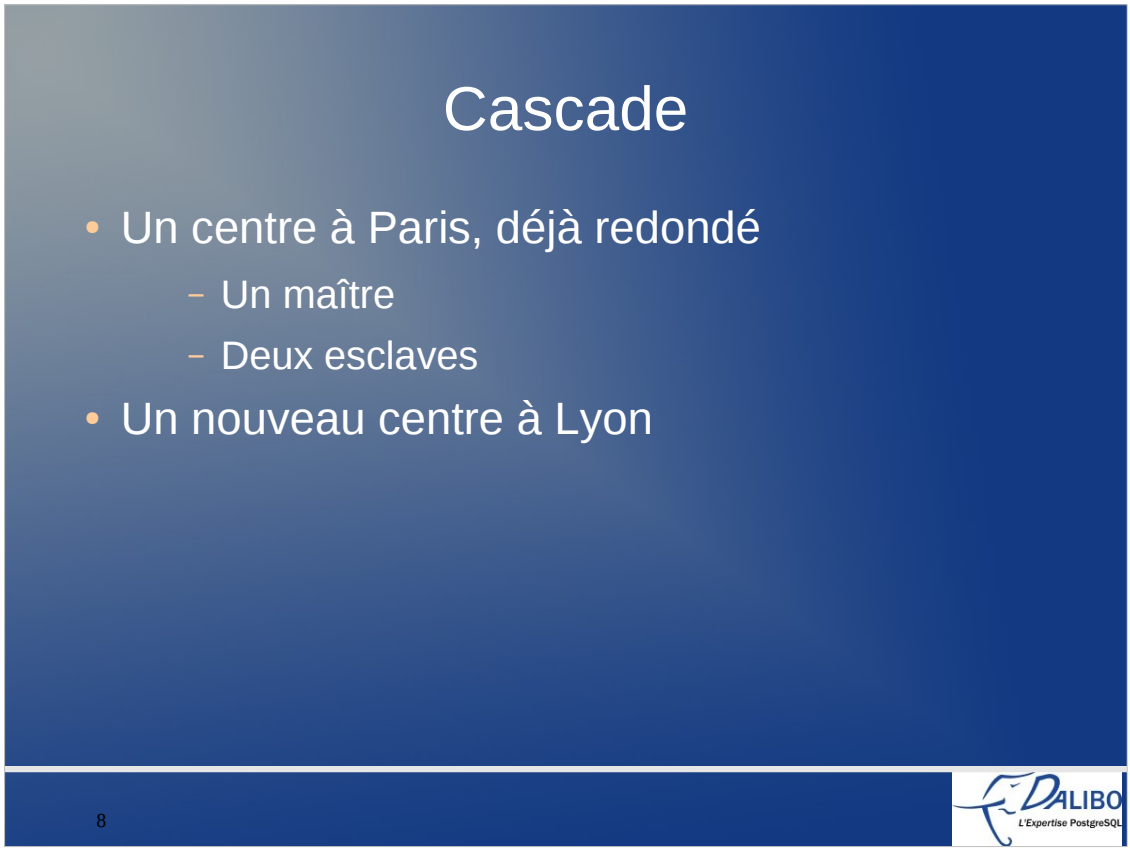
Il existe quelques autres raisons, moins fréquemment mises en avant: la réplication en cascade, qui permet de décharger le maître du travail de réplication, un switchover beaucoup plus simple et rapide.

Autrement dit, la réplication interne de PostgreSQL satisfera la plupart des personnes qui ont des besoins simples en terme de réplication. Et slony sauvera ceux qui ont des besoins plus complexes.

Quelques exemples

- Cascade
- Réplication croisée
- Répartition de charge
- Avec des objets supplémentaires sur l'esclave

Nous allons donc montrer maintenant quelques exemples de cas complexes de réplication.
Ce sont des exemples que nous avons rencontré réellement chez nos clients, que nous avons aidé à mettre en place.

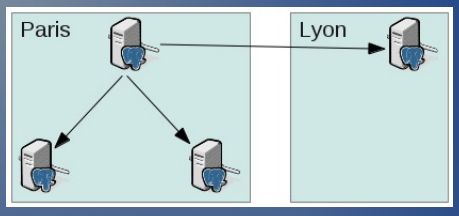


Supposons que nous avons un centre à Paris. Ce centre comporte un serveur de bases de données. Il réplique les données qu'il contient vers deux serveurs esclaves locaux.

L'entreprise grandit et une succursale va s'ouvrir à Lyon. Pour accéder plus rapidement aux données, un serveur est installé dans les locaux de Lyon. Il faut que ce dernier récupère les données de Paris.

Cascade – une solution

- Ajouter un esclave au maître

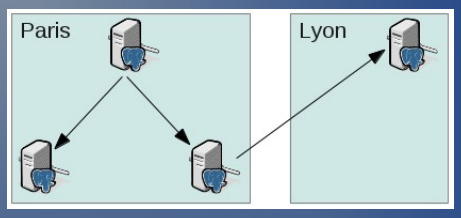


- Mais
  - charge supplémentaire au niveau du maître

La solution la plus simple revient à le configurer en tant que serveur esclave du serveur maître à Paris.
Cependant, ce n'est pas forcément la solution la plus raisonnable d'un point de vue des performances. Le serveur maître de Paris se charge déjà des écritures demandées par les clients et de la réplication vers les deux serveurs esclaves locaux. Ses performances vont donc forcément diminuer s'il doit en plus se charger d'envoyer les données de réplication à un troisième esclave.

Cascade – une meilleure solution

- Ajouter un esclave à l'esclave

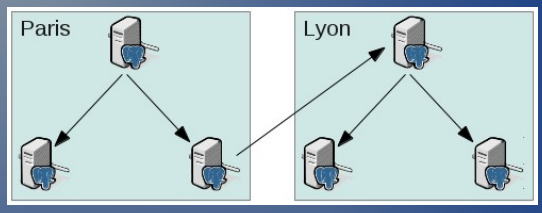


- Décharge le maître
- Ne pose pas de problème pour un failover ou un switchover

Une meilleure solution serait plutôt d'utiliser un des esclaves de Paris pour l'envoi des données de réplication vers l'esclave de Lyon.
Ainsi, le maître ne subit pas de charge supplémentaire avec ce nouvel esclave.
Il est à noter que cela ne gênera en rien les opérations de failover et de switchover. Ces dernières sont toujours aussi simples à réaliser et permettra de conserver la réplication vers l'esclave de Lyon.

Cascade – pour aller plus loin

- Ajoutons des esclaves aux esclaves



La cascade peut être poussée encore plus loin. Il peut être intéressant, pour des raisons de disponibilités et de performances, d'ajouter d'autres serveurs esclaves à Lyon

Mais plutôt que l'envoi des données de réplication se fasse à partir de Paris (ce qui risque d'être particulièrement lent), il est préférable que le premier esclave se charge d'envoyer les données de réplication aux autres esclaves de Lyon.

Cela permet encore une fois de décharger le serveur maître, et de ne pas augmenter l'utilisation de la bande passante réseau.

Imaginons maintenant une grande chaîne de magasins. La maison mère gère le catalogue des produits : elle ajoute les nouveaux produits en vente, supprime les produits qui ne sont plus disponibles, met à jour les prix des produits, etc.

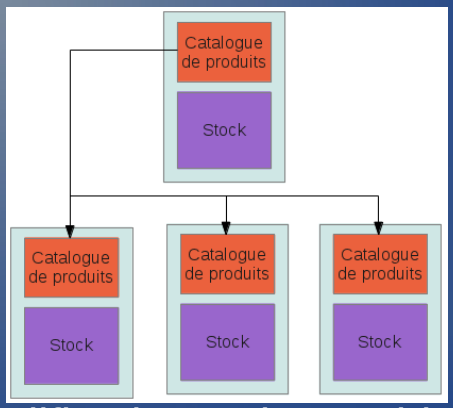
Les magasins, eux, gèrent leur stock, en fonction des produits proposés par la maison mère.

Les magasins ont besoin de connaître le plus rapidement possible les changements tarifaires de la maison mère. Et en même temps, la maison mère a besoin de connaître les stocks des magasins pour calculer des statistiques et établir la liste des produits les plus intéressants.

Ils utilisent une application interne qui travaille à partir d'une base de données contenant le catalogue et les stocks.

Réplication croisée - catalogue

- Première action : partager le catalogue



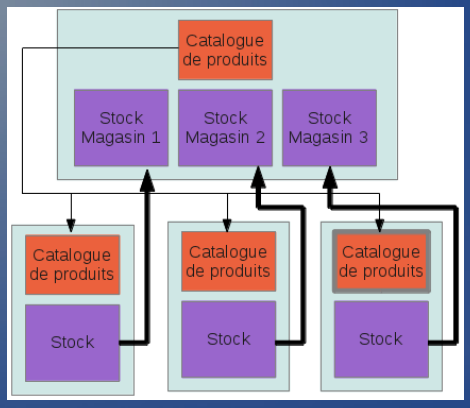
- Toute modification arrive rapidement aux magasins

La première action à entreprendre est de partager le catalogue. La façon la plus simple de le faire revient à mettre en place une réplication Slony entre le serveur de la maison mère et ceux des magasins. La maison mère devant faire des modifications sur le catalogue, son serveur est le maître des tables concernées. Tous les magasins sont configurés en serveur pour ces tables.

Ainsi, toute modification effectuée par la maison mère arrive très rapidement dans les différents magasins.

Réplication croisée - stocks

- Deuxième action : partager les stocks



- Tout changement dans les stocks arrivent pratiquement immédiatement au central

Reste à s'occuper des stocks. Chaque serveur des magasins est configuré comme maître du schéma du stock. L'esclave dans ce cas est le serveur de la maison mère.

Là-aussi, toute modification dans le stock est pratiquement immédiatement transmise au serveur de la maison mère.

Parlons maintenant d'un tout autre cas. Supposons que nous sommes une société d'hébergement de bases de données.

Nous n'avons pour l'instant qu'un seul serveur qui contient toutes les bases de nos clients. Certaines bases sont très volumineuses, d'autres beaucoup moins. Certaines sont très utilisées, d'autres non.

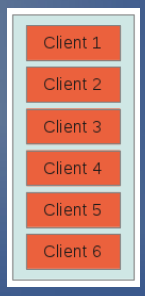
Le résultat est un serveur rapidement surchargé de demandes, et sans serveur de bascule.

Une solution simple pour décharger le serveur serait de mettre en place un deuxième serveur et d'y placer certaines des bases du premier serveur.

Mais s'il faut en plus avoir un serveur de bascule, on ne passera pas de un à deux serveurs, mais de un à quatre. Cela n'a pas la même chose en terme de finances et d'administration. Là-aussi, Slony peut aider à résoudre ce problème.

Répartition de charge – état initial

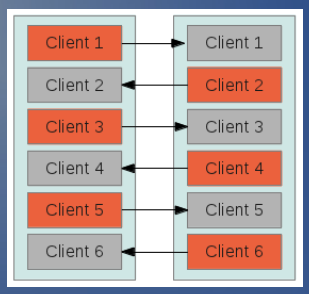
- Un serveur avec six bases de clients



L'idée initiale est donc le suivant : un serveur contenant six bases de données de clients. Le but est de décharger ce serveur de tout le travail réclamé par les clients de ces six bases.

Répartition de charge – 2 serveurs

- Ajouter un serveur

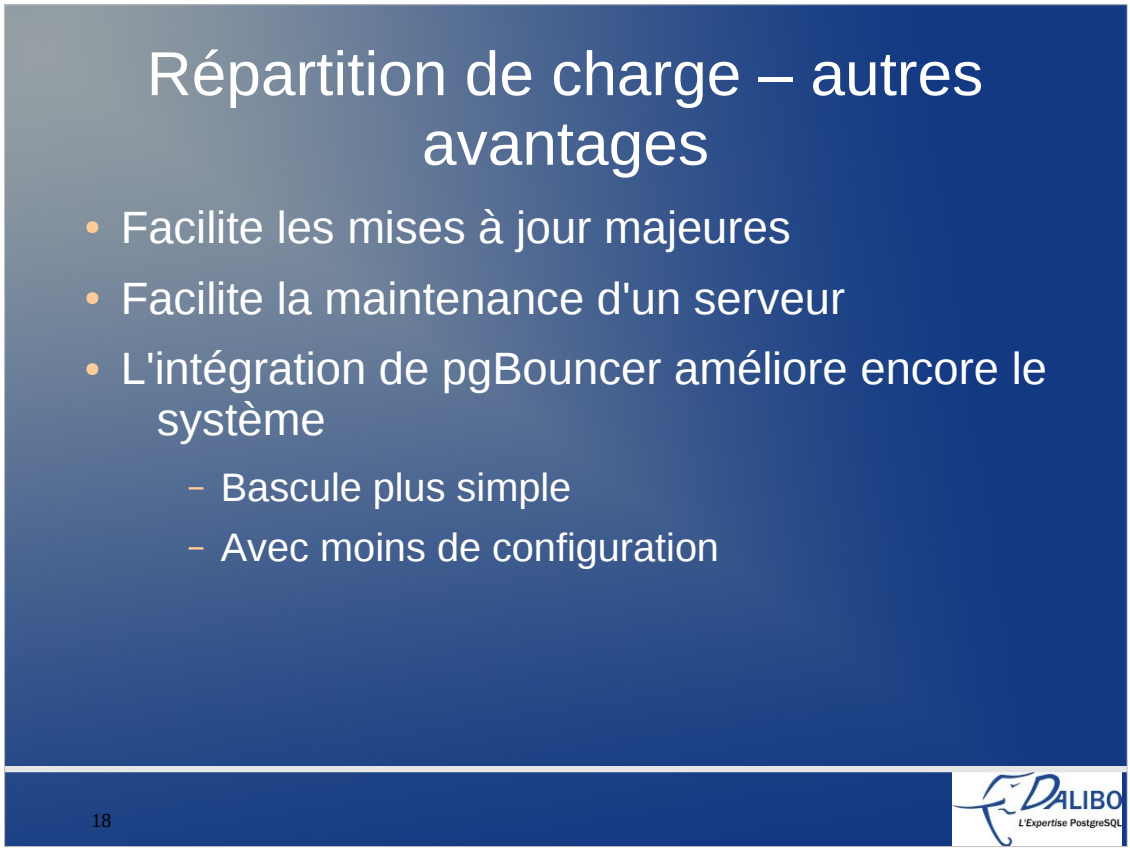


- 2 avantages
  - Répartition de charge
  - Et deuxième serveur disponible

Avec Slony, il est possible d'ajouter un nouveau serveur. Ce nouveau serveur récupèrera la moitié des bases. Les bases qui sont sur le serveur 1 seront répliquées sur le serveur 2, et inversement.

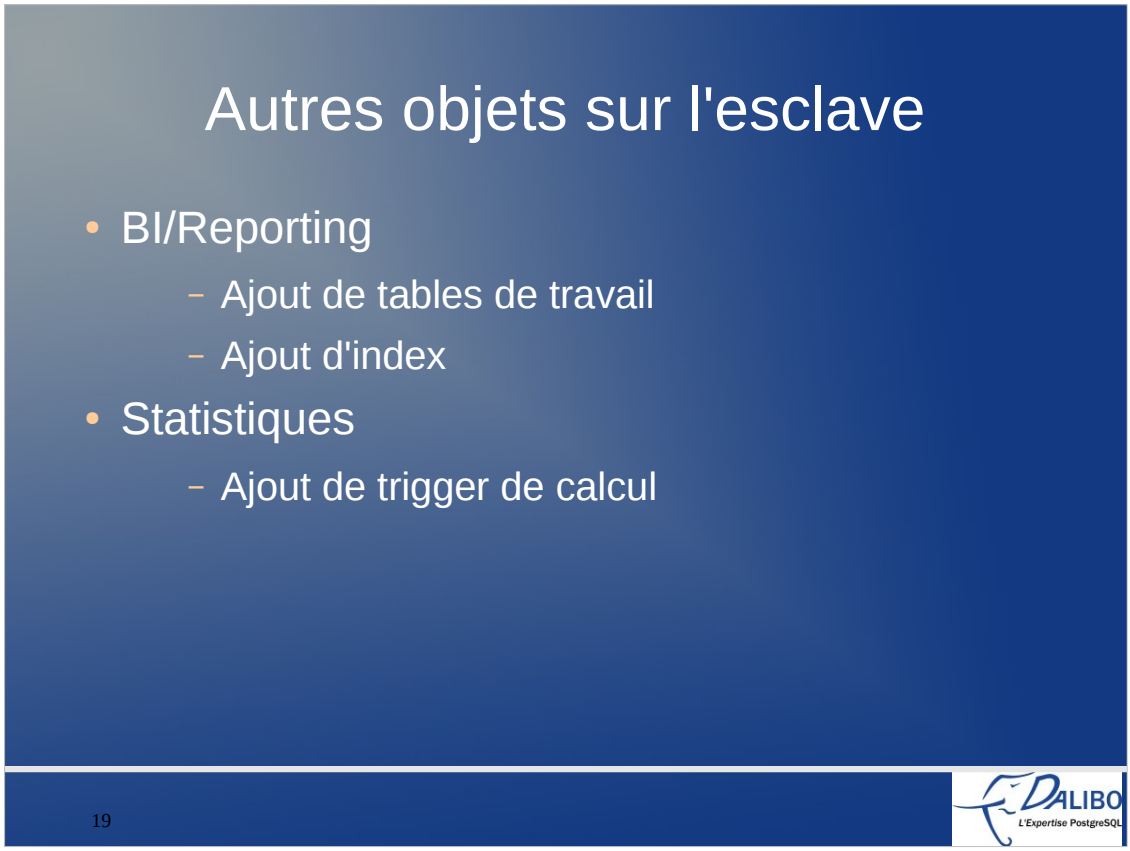
Cela présente deux avantages. Tout d'abord, nous diminuons la charge de chaque serveur, vu qu'ils n'ont plus qu'à s'occuper de l'exécution des requêtes pour la moitié des bases et de l'application de la réplication pour l'autre moitié. Ensuite, nous avons deux serveurs qui se redondent. Autrement dit, si le serveur 1 tombe, le serveur 2 est là pour fonctionner (mais en mode dégradé au niveau des performances).

De plus, si on veut de nouveau diminuer la charge, il suffit d'introduire un troisième serveur et de répartir les bases entre les trois serveurs. Etc.



Dans les autres avantages, on peut noter la facilité des opérations de maintenance. Il suffit de basculer les bases sur un serveur pour pouvoir procéder à un changement de matériel, à un redémarrage (par exemple pour utiliser un noyau ou un pilote mis à jour) sur l'autre serveur. Après redémarrage de ce serveur, il suffit de nouveau de basculer les bases pour retrouver les performances attendues par les clients.

La mise en place d'un outil comme pgBouncer facilite grandement ce travail. Il peut bloquer les connexions pendant un redémarrage, et rediriger les connexions des clients d'un serveur vers un autre. Les bascules sont donc plus simples à faire, en nécessitant moins de configuration.



Le dernier exemple que nous allons prendre concerne la possibilité d'ajouter des objets sur un esclave.

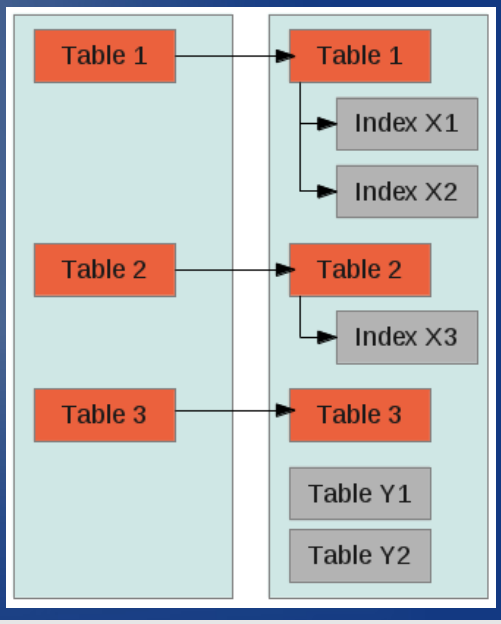
Lors de l'utilisation d'un outil de reporting, les requêtes exécutées sont souvent très gourmandes en performance. Les exécuter sur un serveur esclave permet de décharger le maître qui reçoit déjà le travail de tous les opérateurs chargés d'ajouter et de mettre à jour les données.

Cependant, les requêtes de reporting sont très différentes des requêtes standards des opérateurs. Des index spécifiques sont souvent nécessaires. Il est donc intéressant de pouvoir créer ces index uniquement sur l'esclave pour que le maître ne paie pas le temps de mise à jour de ces index dont lui n'a aucunement besoin.

De plus, certains outils de reporting ont besoin de tables de travail (temporaires ou non).

Autres objets - exemple

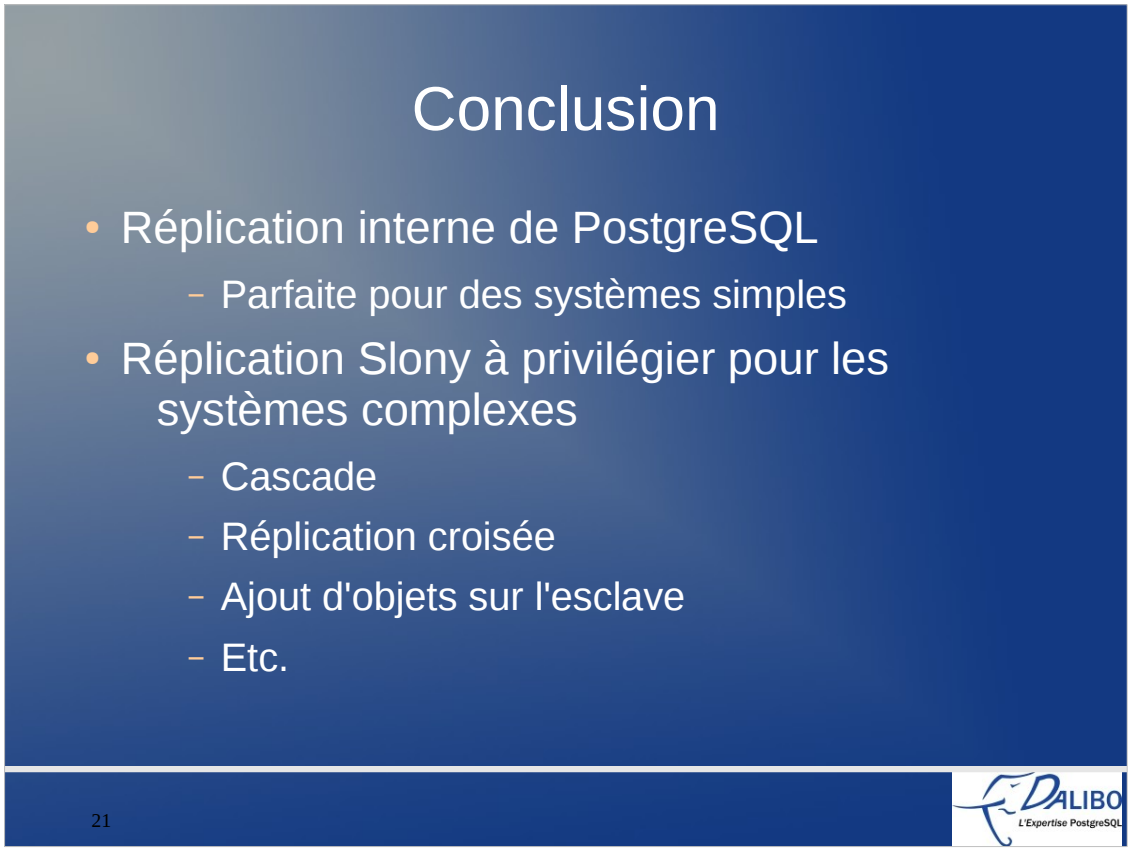
- Réplication de trois tables du maître
- 3 index specifiques et deux tables de travail sur l'esclave



Parfois, il est aussi intéressant d'ajouter des triggers qui vont précalculer des données qui seront utilisées par les requêtes de reporting.

Tout ça fait qu'il est très agréable de pouvoir créer des objets supplémentaires sur les esclaves. Slony permet cela. En fait, il ne fait qu'interdire l'ajout, la modification et la suppression de données des tables dont le serveur est configuré en esclave.

L'exemple montre ici un serveur maître contenant trois tables répliquées vers un serveur esclave. Ce serveur esclave dispose en plus de deux autres tables et de trois index supplémentaires.



Ce document a montré que la réplication Slony reste intéressante dans le cadre de certaines utilisations plus complexes que ce que la réplication interne peut gérer : réplication en cascade, réplication croisée, ajout d'objets, réplication entre serveurs de version majeure différente, etc.

Slony n'est donc pas à mettre au rebut par principe. Il est essentiel, lors de la réflexion sur la mise en place d'une solution de réplication d'utiliser celle qui sera la plus simple mais qui répond aussi aux critères demandées par l'entreprise. Dans beaucoup de cas, la réplication interne suffira et il est donc conseillé dans ce cas de l'utiliser. Dans les autres cas, Slony est un des systèmes à considérer avec d'autres comme Londiste et Bucardo.