Insérer beaucoup de tuples COPY ou INSERT ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 septembre 2010. Dernière mise à jour le 10 septembre 2010

https://www.bortzmeyer.org/copy-and-insert.html

Tiens, un petit problème de performance avec un SGBDR. Pour insérer un grand nombre de tuples, est-ce plus rapide avec une boucle sur le traditionnel `INSERT` ou bien avec une autre commande SQL, `COPY` ?

Le problème vient de DNSmezzo <http://www.dnsmezzo.net/> qui doit parfois insérer plusieurs millions de tuples en une seule exécution (chaque tuple représentant une requête ou une réponse DNS). Cela prend du temps et cela s'aggrave avec le remplissage de la base. La documentation de PostgreSQL suggère d'utiliser COPY plutôt qu'INSERT <http://www.postgresql.org/docs/current/interactive/populate.html#POPULATE-COPY-FROM>. Qu'y gagne t-on exactement ?

Les tests ont été effectués avec PostgreSQL 8.3 sur un PC/Debian. Deux programmes en C, (en ligne sur https://www.bortzmeyer.org/files/speed-insert.c) et (en ligne sur https://www.bortzmeyer.org/files/speed-copy.c) ont été comparés, pour une table composée de trois champs (dont deux seulement sont spécifiés lors de l'insertion) :

```
CREATE TABLE Shadoks (id SERIAL, name TEXT, value INTEGER);
```

Voici d'abord l'insertion d'une table sans index, pour dix millions de tuples :

```
% time ./speed-insert dbname=essais 10000000
./insert dbname=essais 10000000  53.35s user 53.72s system 15% cpu 11:49.85 total

% time ./speed-copy dbname=essais 10000000
./copy dbname=essais 10000000  8.61s user 0.14s system 11% cpu 1:15.79 total
```

Bref, `COPY` est dix fois plus rapide. (Si vous regardez le source de `speed-copy`, vous verrez une constante `INCREMENT` dont la valeur a été déterminée empiriquement. Mettez-la trop bas, à 10 par exemple, et les performances chutent.)

Et avec des index ? Notons que la documentation de PostgreSQL, citée plus haut, recommande de les débrayer avant d'insérer en masse et de les rétablir ensuite. Créons le :

```
CREATE INDEX name_idx on Shadoks(name);
```

Et, en effet, les performances chutent dramatiquement : même avec `COPY`, au bout de quinze minutes, j'ai interrompu le programme. Utiliser la méthode suggérée par la documentation est bien plus rapide :

```
% time sh -c '(psql -c "DROP INDEX name_idx" essais ;  \
            ./copy dbname=essais 10000000; \
            psql -c "CREATE INDEX name_idx on Shadoks(name)" essais)'
DROP INDEX
CREATE INDEX
sh -c   8.56s user 0.14s system 3% cpu 4:01.39 total
```

Il existe une troisième solution à la question de la « copie en masse » mais elle nécessite l'ajout d'un logiciel supplémentaire, pgbulkload <http://pgbulkload.projects.postgresql.org/>. Je ne l'ai pas testé mais, questions performances, les auteurs annoncent des résultats intéressants. Mais attention : rien n'est gratuit. Ces gains de performance reposent essentiellement sur l'absence de support des "*triggers*" et des "*rules*", et surtout il ne valide pas les données en entrée. Du coup ça n'est utilisable que pour restaurer une sauvegarde de confiance. Dans tous les cas, ces copies rapides (`COPY` ou pgbulkload) peuvent poser des problèmes de compatibilité avec d'éventuels mécanismes de réplication qui ont été mis en place donc lisez bien la documentation de votre système de réplication, si vous en avez un. D'autre part, les techniques « rapides » (tout ce qui n'est pas `INSERT`) peuvent poser des problèmes en cas de parallélisme (voir un exemple avec COPY <http://archives.postgresql.org/pgsql-fr-generale/2006-09/msg00027.php>).

Merci à Malek Shabou et Dimitri Fontaine pour leurs conseils et informations. Un bon article sur la question est « "*How to insert data to database - as fast as possible*" <http://www.depesz.com/index.php/2007/07/05/how-to-insert-data-to-database-as-fast-as-possible/> ».