≪ It's time for a complete rewrite ≫ des SGBD ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 novembre 2007

https://www.bortzmeyer.org/complete-rewrite.html

Faut-il abandonner les moteurs de bases de données relationnelles, qui ont été au cœur de l'informatique depuis des dizaines d'années ? Oui, disent sans hésiter les auteurs de cet article, parmi lesquels Michael Stonebraker, un pionnier du domaine et un des responsables du projet qui a donné PostgreSQL.

Les SGBD relationnels sont partout. Initialement très chers, réservés aux "*mainframes*" et limités à l'informatique de gestion, ils sont présents désormais derrière presque chaque Blog, chaque site Web. Peu de sites Unix n'en ont pas. Mais il est temps de les jeter, disent Stonebraker et ses collègues, dans un article au titre qui réveille ≪ "*The end of an Architectural Era (it's Time for a Complete Rewrite)*" <http://web.mit.edu/dna/www/vldb07hstore.pdf> ≫.

Que disent-ils ? Que les SGBD actuels sont trop généralistes et qu'on pourrait gagner beaucoup en performances en se limitant à des secteurs spécialisés. Par exemple, une grande force de SQL est la possibilité d'écrire des requêtes qui n'avaient pas été prévues lors de la création de la base, par exemple à des fins de "*data mining*". Mais cette possibilité coûte cher alors que beaucoup de bases fonctionnent au contraire en OLTP où les requêtes sont connues à l'avance. Un exemple d'environnement OLTP est un site Web de commerce électronique où les requêtes sont codées dans l'application, donc prévisibles. Ne pourrait-on pas optimiser pour cette utilisation OLTP, laissant un autre modèle de base de données répondre aux questions ouvertes ? On le peut, montrent les auteurs, qui annoncent un facteur 80 de gain de performances, avec la base qu'ils ont créé, H-store.

De même, la concurrence entre deux sessions simultanées coûte cher. Dans un environnement OLTP typique, les transactions sont courtes, alors pourquoi ne pas les traiter en série plutôt qu'en parallèle, éliminant ainsi une grande part de la complexité du SGBD ?

Stonebraker et ses collègues ont plein d'autres idées. Par exemple généraliser les procédures stockées (pour lesquelles ils envisagent Ruby) afin d'éviter les allers-retours entre le client et le serveur. Ou préférer des réseaux de machines pas chères ne partageant pas de ressources à des gros serveurs multiprocesseurs, un peu ce que fait Google avec MapReduce. Je vous laisse les découvrir dans l'article. Personnellement, j'apprécie beaucoup la souplesse de SQL et j'espère qu'on me laissera toujours écrire des requêtes SQL compliquées comme dans l'exemple dont les auteurs de moquent, ≪ Quels sont les employés qui gagnent plus que leur supérieur ? ≫, en laissant entendre que les exercices SQL classiques de l'enseignement ne se rencontrent jamais dans la réalité.

On peut suivre le débat dans plusieurs articles publiés dans le blog "*The database column*" <http://www.databasecolumn.com/> (merci à Erwan Azur pour le lien).