DBFrequentQueries : Extraction de requêtes fréquentes

Lucie Copin*, Nicolas Pecheur*, Anne Laurent***, Yudi Augusta**, Budi Sentana**,
Dominique Laurent****, Tao-Yuan Jen ****
*Univ. Montpellier 2 - Polytech'Montpellier, lucie.copin@gmail.com
**STIKOM-Bali, yudi@stikom-bali.ac.id
***Univ. Montpellier 2 - LIRMM - CNRS, laurent@lirmm.fr
****Univ. Cergy-Pontoise - ETIS - CNRS, jen@u-cergy.fr

L'extraction de requêtes fréquentes dans une base de données (*i.e.,* les requêtes dont la réponse contient un nombre de n-uplets supérieur à un seuil donné *min-sup*) est un problème difficile. L'implémentation d'outils efficaces est délicate à mettre en œuvre car le nombre de requêtes à considérer est exponentiel par rapport à la taille de la base.
L'approche définie dans [2] montre que ce nombre peut être réduit en considérant une relation d'équivalence entre requêtes qui prend en compte les dépendances fonctionnelles. Dans cette démonstration, nous présentons un outil montrant que cette approche permet l'extraction *effective* de toutes les requêtes projection-sélection fréquentes sur une table donnée Δ vue comme la jointure des tables de dimension et de la table de faits d'un schéma étoile. L'algorithme utilisé suit le principe de l'algorithme Apriori [1].

L'outil : DBFrequentQueries

L'outil DBFrequentQueries prend en entrée : (i) la table Δ, (ii) une table décrivant le schéma étoile et (iii) le seuil de support *min-sup*. Le résultat produit par DBFrequentQueries est l'ensemble des requêtes fréquentes ainsi que le temps total d'exécution et le nombre de classes de requêtes fréquentes. Les tables traitées sont dans une base Mysql, qui doit être fournie à l'interface. DBFrequentQueries est développé en Java, et est donc multi-plateforme.

Résultats

Les tests ont été menés selon deux objectifs : d'une part mesurer les performances de l'application (et donc de l'algorithme) en termes de temps et de mémoire, et d'autre part analyser distinctement le coût des trois étapes de l'algorithme. En termes de performance, on observe une augmentation du temps de traitement linéaire selon la taille de la table, et exponentielle selon le nombre d'attributs. Les différents cas ainsi que la stratégie test seront exposés au cours de la démo.

Références

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo. Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining*, pp. 309-328, AAAI-MIT Press, 1996.

[2] T.-Y. Jen, D. Laurent, N. Spyratos. Mining All Frequent Projection-Selection Queries from a Relational Table. *Int. Conference on Extending Database Technology* (EDBT08), pp. 368-379, ACM Press, 2008.