Vers l'auto-administration des entrepôts de données

Kamel Aouiche — Jérôme Darmont[*] — Le Gruenwald[**]

[*] *ERIC/BDD, Université Lyon 2*
5 avenue Pierre Mendès-France
69676 Bron Cedex, France
{kaouiche, jdarmont}@eric.univ-lyon2.fr

[**] *School of Computer Science*
University of Oklahoma
Norman, OK 73019, USA
ggruenwald@ou.edu

RÉSUMÉ. *Avec le développement des bases de données en général et des entrepôts de données (data warehouses) en particulier, il est devenu primordial de réduire la fonction d'administration de base de données. L'idée d'utiliser des techniques de fouille de données (data mining) pour extraire des connaissances utiles des données elles-mêmes pour leur administration est avancée depuis quelques années. Pourtant, peu de travaux de recherche ont été entrepris. L'objectif de cette étude est de rechercher une façon d'extraire des données stockées des connaissances utilisables pour appliquer de manière automatique des techniques d'optimisation des performances, et plus particulièrement d'indexation. Nous avons réalisé un outil qui effectue une recherche de motifs fréquents sur une charge donnée afin de calculer une configuration d'index permettant d'optimiser le temps d'accès aux données. Les expérimentations que nous avons menées ont montré que les configurations d'index générées par notre outil permettent des gains de performance de l'ordre de 15% à 25% sur une base et un entrepôt de données tests.*

ABSTRACT. *With the wide development of databases in general and data warehouses in particular, it is important to reduce the tasks that a database administrator must perform manually. The idea of using data mining techniques to extract useful knowledge for administration from the data themselves has existed for some years. However, little research has been achieved. The aim of this study is to search for a way of extracting useful knowledge from stored data to automatically apply performance optimization techniques, and more particularly indexing techniques. We have designed a tool that extracts frequent itemsets from a given workload to compute an index configuration that helps optimizing data access time. The experiments we performed showed that the index configurations generated by our tool allowed performance gains of 15% to 25% on a test database and a test data warehouse.*

MOTS-CLÉS : *Bases et entrepôts de données, Auto-indexation, Data mining, Motifs fréquents.*

KEYWORDS: *Databases and data warehouses, Auto-indexing, Data mining, Frequent itemsets.*

1re soumission à *XXXVèmes Journées de Statistique, Session spéciale EFD*, le 10 février 2003.

L'utilisation courante de bases de données requiert un administrateur qui a pour rôle principal la gestion des données au niveau logique (définition de schéma) et physique (fichiers et disques de stockage), ainsi que l'optimisation des performances de l'accès aux données. Avec le déploiement à grande échelle des systèmes de gestion de bases de données (SGBD), minimiser la fonction d'administration est devenu indispensable [WEI 02].

L'une des tâches importantes d'un administrateur est la sélection d'une structure physique appropriée pouvant améliorer les performances du système en minimisant les temps d'accès aux données [FIN 88]. Les index sont des structures physiques permettant un accès direct aux données. Le travail d'optimisation des performances de l'administrateur se porte en grande partie sur la sélection d'index et de vues matérialisées [GUP 99, AGR 01]. Ces structures jouent un rôle particulièrement important dans les bases de données décisionnelles (BDD) tels que les entrepôts de données, qui présentent une volumétrie très importante et sont interrogés par des requêtes complexes.

Depuis quelques années, l'idée est avancée d'utiliser les techniques de fouille de données (*data mining*) pour extraire des connaissances utiles des données elles-même pour leur administration [CHA 98a]. Cependant, peu de travaux de recherches ont été entrepris dans ce domaine jusqu'ici. C'est pourquoi nous avons conçu et réalisé un outil qui utilise la fouille de données pour proposer une sélection (configuration) d'index pertinente.

Partant de l'hypothèse que l'utilité d'un index est fortement corrélée à la fréquence de l'utilisation des attributs correspondants dans l'ensemble des requêtes d'une charge donnée, la recherche de motifs fréquents [AGR 93] nous a semblé appropriée pour mettre en évidence cette corrélation et faciliter le choix des index à créer. L'outil que nous présentons dans cet article exploite le journal des transactions (ensemble de requêtes résolues par le SGBD) pour proposer une configuration d'index.

Ce contexte met en relation les requêtes de la charge en entrée avec l'ensemble des attributs susceptibles d'être indexés. Les motifs fréquents obtenus en sortie sont des ensembles d'attributs formant une configuration d'index candidats. Diverses stratégies peuvent finalement être appliquées pour sélectionner les index à construire effectivement parmi ceux de cette configuration.

L'utilisation de l'algorithme d'extraction de fréquents Close [PAS 99a, PAS 99b] nous permet de générer des index mono-attribut et multi-attributs à la volée, sans avoir à mettre en œuvre un processus itératif permettant de créer successivement des index multi-attributs de plus en plus gros à partir d'un ensemble d'index mono-attribut, comme c'est le cas, par exemple, dans l'outil IST développé par Microsoft [CHA 97, CHA 98b, AGR 00]

Nos premiers résultats expérimentaux montrent que notre technique permet effectivement d'améliorer le temps de réponse de 20% à 25% pour une charge décisionnelle appliquée à une base de données relationnelle (banc d'essais TPC-R [Tra99]). Nous avons par ailleurs proposé deux stratégies effectuer une sélection parmi les index candidats : la première crée systématiquement tous les index candidats et la deuxième

ne crée que les index associés à des tables dites volumineuses. La deuxième stratégie apporte une meilleure amélioration car elle propose un compromis entre l'espace occupé par les index (le nombre d'index créés est limité à ceux qui sont définis sur des attributs de tables volumineuses) et l'intérêt de la création d'un index (il est peu intéressant de créer un index sur une petite table).

Nous avons également réalisé des tests sur un petit magasin de données d'accidentologie auquel nous avons appliqué une charge décisionnelle ad hoc [AOU 02]. Le gain en temps de réponse, de l'ordre de 14%, est moins important que dans le cas de TPC-R. Cela peut être expliqué par le fait que les index créés par défaut par SQL Server sont des variantes des B-arbres et non des index *bitmap* et des index de jointure en étoile, qui seraient plus adaptés pour un entrepôt de données [O'N 95, O'N 97].

Notre travail démontre que l'idée d'utiliser des techniques de fouille de données pour l'auto-administration des SGBD est prometteuse. Il n'est cependant qu'une première approche et ouvre de nombreuses perspectives de recherche. Une première voie consisterait à améliorer la sélection des index en concevant des stratégies plus élaborées que l'utilisation exhaustive d'une configuration ou l'exploitation de renseignements relativement basiques concernant la taille des tables. Un modèle de coût plus fin au regard des caractéristiques des tables (autres que la taille), ou encore une stratégie de pondération des requêtes de la charge (par type de requête : sélection ou mise à jour), pourraient nous aider dans cette optique. L'utilisation d'autres méthodes de fouille de données non-supervisées telles que le regroupement (*clustering*) pourraient également fournir des ensembles de fréquents moins volumineux.

Par ailleurs, il paraît indispensable de continuer à tester notre méthode pour mieux évaluer la surcharge qu'elle engendre pour le système, que ce soit en terme de génération des index ou de leur maintenance. Il est notamment nécessaire de l'appliquer pour des entrepôts de données de grande taille et en tirant partie d'index adaptés. Il serait également très intéressant de la comparer de manière plus systématique avec l'outil IST, que ce soit par des calculs de complexité des heuristiques de génération de configurations d'index (surcharge) ou des expérimentations visant à évaluer la qualité de ces configurations (gain en temps de réponse et surcharge due à la maintenance des index).

Finalement, étendre ou coupler notre approche à d'autres techniques d'optimisation des performances (vues matérialisées, gestion de cache, regroupement physique, etc.) constitue également une voie de recherche prometteuse. En effet, dans le contexte des entrepôts de données, c'est principalement en conjonction avec d'autres structures physiques (principalement les vues matérialisées) que l'indexation permet d'obtenir des gains de performance significatifs [GUP 99, AGR 00, AGR 01].

Bibliographie

[AGR 93] AGRAWAL R., IMIELINSKI T., SWAMI A. N., « Mining Association Rules between Sets of Items in Large Databases », *SIGMOD Record*, vol. 22, n° 2, 1993, p. 207–216.

[AGR 00] AGRAWAL S., CHAUDHURI S., NARASAYYA V. R., « Automated Selection of Materialized Views and Indexes in SQL Databases », *26th International Conference on Very Large Data Bases (VLDB 2000), Cairo, Egypt*, 2000, p. 496–505.

[AGR 01] AGRAWAL S., CHAUDHURI S., NARASAYYA V. R., « Materialized View and Index Selection Tool for Microsoft SQL Server 2000 », *2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, USA*, , 2001.

[AOU 02] AOUICHE K., « Magasin de données accidentologie : schéma et charge », http ://bdd.univ-lyon2.fr/download/charge-accidentologie.pdf, 2002.

[CHA 97] CHAUDHURI S., NARASAYYA V. R., « An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server », *23rd International Conference on Very Large Data Bases (VLDB 1997), Athens, Greece*, 1997, p. 146–155.

[CHA 98a] CHAUDHURI S., « Data Mining and Database Systems : Where is the Intersection ? », *Data Engineering Bulletin*, vol. 21, n° 1, 1998, p. 4–8.

[CHA 98b] CHAUDHURI S., NARASAYYA V. R., « AutoAdmin 'What-if' Index Analysis Utility », *1998 ACM SIGMOD International Conference on Management of Data, Seattle, USA*, 1998, p. 367–378.

[FIN 88] FINKELSTEIN S. J., SCHKOLNICK M., TIBERIO P., « Physical Database Design for Relational Databases », *TODS*, vol. 13, n° 1, 1988, p. 91–128.

[GUP 99] GUPTA H., « Selection and maintenance of views in a data warehouse », PhD thesis, Stanford University, 1999.

[O'N 95] O'NEIL P., GRAEFE G., « Multi-table joins through bitmapped join indices », *SIGMOD Record*, vol. 24, n° 3, 1995, p. 8–11.

[O'N 97] O'NEIL P., QUASS D., « Improved Query Performance with Variant Indexes », *SIGMOD Record*, vol. 26, n° 2, 1997, p. 38–49.

[PAS 99a] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Discovering Frequent Closed Itemsets for Association Rules », *7th International Conference on Database Theory (ICDT 1999), Jerusalem, Israel*, vol. 1540 de *LNCS*, 1999, p. 398–416.

[PAS 99b] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Efficient mining of association rules using closed itemset lattices », *Information Systems*, vol. 24, n° 1, 1999, p. 25–46.

[Tra99] Transaction Processing Council, « TPC Benchmark R Standard Specification », 1999.

[WEI 02] WEIKUM G., MONKEBERG A., HASSE C., ZABBACK P., « Self-tuning Database Technology and Information Services : from Wishful Thinking to Viable Engineering », *28th International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China*, 2002.