Expérimentations avec le banc d'essais pour entrepôts de données DWEB

Jérôme Darmont
Université de Lyon (ERIC Lyon 2)
5 avenue Pierre Mendès-France
69676 Bron Cedex
jerome.darmont@univ-lyon2.fr

Résumé : La performance est un problème crucial pour les utilisateurs et les concepteurs de bases de données. Elle est généralement évaluée de manière expérimentale à l'aide de bancs d'essais (*benchmarks*). Dans le contexte particulier des entrepôts de données et de l'analyse en ligne OLAP (*On Line Analytical Processing*), il existe peu de bancs d'essais. Le but de cet article est de présenter les modèles sous-jacents et le fonctionnement du banc d'essais DWEB (*Data Warehouse Engineering Benchmark*), qui permet de générer des entrepôts de données synthétiques ad hoc, les charges de requêtes d'alimentation et d'analyse associées et d'exécuter les secondes sur les premiers pour mesurer le temps de réponse du système.

Mots-clés : Entrepôts de données, OLAP, évaluation de performance, banc d'essais.

1. Introduction

La performance est un problème crucial pour les utilisateurs et les concepteurs de bases de données. Elle est généralement évaluée expérimentalement à l'aide de bancs d'essais (*benchmarks*, dans la terminologie anglo-saxonne) qui permettent, par exemple, de tester des choix architecturaux, de comparer des technologies ou de régler un système (*tuning*).

Dans le contexte particulier des entrepôts de données et de l'analyse en ligne OLAP (*On Line Analytical Processing*), le banc d'essais APB-1 [8] a été très populaire à la fin des années 1990, mais sa grande simplicité le rend inapproprié pour de nombreux types d'expériences [10]. Le *Transaction Performance Processing Council* (TPC), qui spécifie des bancs d'essais standards dans le monde relationnel, dispose également d'un banc d'essais décisionnel, TPC-H [11], mais ce dernier de modélise ni une base de données multidimensionnelle typique des entrepôts, ni une charge de requêtes OLAP. Il est d'ailleurs implicitement considéré comme obsolète par le TPC, et en cours de remplacement. Son successeur, TPC-DS [12], modélise explicitement un entrepôt de données et une charge de requêtes décisionnelles. Très complet, voire complexe, il n'est cependant actuellement disponible que sous forme de *draft*.

Tous les bancs d'essais pour entrepôts de données existants présentent le défaut d'être peu paramétrables. Par exemple, les bancs d'essais du TPC ne varient que selon un facteur d'échelle qui fixe la taille de la base. Or, les architectures d'entrepôts et les requêtes décisionnelles associées dépendent énormément du domaine d'application. Il est donc important pour les concepteurs qui souhaitent évaluer l'impact de choix architecturaux ou de techniques d'optimisation de pouvoir tester leurs performances dans des conditions expérimentales variées, c'est-à-dire avec différentes configurations d'entrepôts et de charge.

C'est pourquoi nous avons proposé un banc d'essais qui permet de générer des entrepôts synthétiques ad hoc, ainsi que les charges de requêtes d'alimentation et d'analyse associées : DWEB (*Data Warehouse Engineering Benchmark*) [1, 2, 3, 4]. DWEB est un logiciel portable écrit en Java qui est distribué gratuitement[1] selon les termes de la licence *Creative Commons*.

DWEB, qui permet en premier lieu d'évaluer l'impact de différents choix architecturaux ou de techniques d'optimisation sur les performances d'un système donné, est plutôt destiné aux concepteurs et aux chercheurs. Il est donc plus complémentaire que concurrent de TPC-DS, qui vise à comparer les performances de plusieurs systèmes dans des conditions expérimentales fixées et s'adresse aux utilisateurs.

[1] http://eric.univ-lyon2.fr/~jdarmont/?page=logiciels

Nous présentons dans cet article un résumé des spécifications de DWEB (Section 2) ainsi qu'un scénario classique d'utilisation de notre banc d'essais (Section 3). Nous concluons finalement en Section 4.

2. Spécifications de DWEB

Les principaux composants d'un banc d'essais sont ses modèles de base de données et de charge. Nous présentons également dans cette section le processus d'ETL (*Extract, Transform, Load*) mis en œuvre dans DWEB, ainsi que le protocole d'exécution de notre banc d'essais.

2.1. Métamodèle d'entrepôt

Notre objectif avec DWEB est de pouvoir modéliser les grands types de schémas multidimensionnels qui sont populaires dans les environnements ROLAP (*Relational OLAP*), à savoir les schémas en étoile, en flocon de neige (avec des dimensions hiérarchiques) et en constellation (avec des tables de faits multiples et des dimensions partagées) [6, 7]. Pour cela, nous utilisons un métamodèle d'entrepôt de données (Figure 1) qui permet d'instancier ces différents schémas. Notre métamodèle peut par ailleurs être vu lui-même comme une instance du CWM (*Common Warehouse Metamodel*) [9].

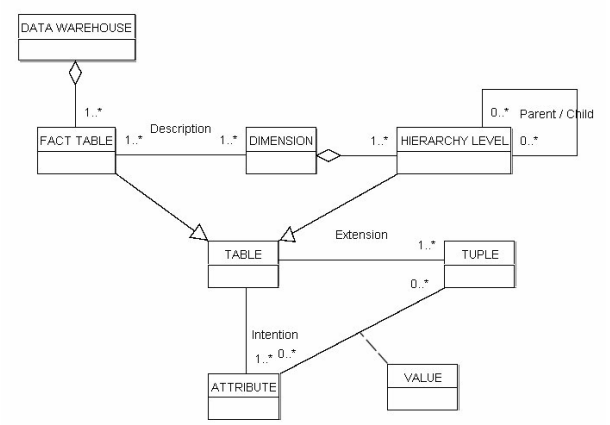

Figure 1 : Métamodèle d'entrepôt DWEB

La génération d'entrepôt est totalement configurable, à l'aide d'un ensemble complet de paramètres dits de bas niveau qui permettent de spécifier finement les caractéristiques de chaque table de faits, niveau hiérarchique de dimension, etc. Cependant, comme ce paramétrage très fin peut s'avérer fastidieux et compliqué à maîtriser pour des schémas même de taille moyenne, nous proposons un ensemble limité et plus gérable de paramètres de haut niveau qui permettent de calculer l'ensemble complet des paramètres de bas niveau (ce sont des valeurs moyennes des paramètres de bas niveau). Les paramètres de haut niveau incluent, par exemple, le nombre moyen de tables de faits, la densité moyenne des faits, le nombre moyen de dimensions par table de faits, le nombre moyen de niveaux hiérarchiques dans les dimensions, etc.

2.2. Modèle de charge

Le modèle de charge de DWEB représente deux classes de requêtes différentes : des requêtes purement décisionnelles, qui comprennent les opérations OLAP classiques telles que la construction de cube, le forage vers le haut ou vers le bas (*roll-up* ou *drill-down*, respectivement) et la projection/sélection (*slice and dice*) ; et des requêtes dites d'extraction qui effectuent des jointures sur les tables de l'entrepôt. Ce modèle de charge permet de générer des requêtes à la norme SQL-99, en fonction d'un ensemble de paramètres tels que le nombre total de requêtes dans la charge, la proportion respective de requêtes décisionnelles et de requêtes d'extraction, le nombre d'attributs et de restriction dans une requête, etc.

2.3. Processus d'ETL

Puisque DWEB est un logiciel indépendant qui génère des données et des charges, nous avons choisi de ne pas inclure de phase d'extraction dans sa phase d'ETL. Les mises à jour sont effectuées directement dans la base de données pour conserver une certaine simplicité d'usage à l'outil et minimiser la gestion de fichiers externes. Ces mises à jour pourraient cependant être enregistrées dans des fichiers avant d'être appliquées pour simuler l'extraction.

Nous n'avons pas non plus inclu de transformation dans le processus d'ETL, malgré l'importance de cette phase. Cependant, dans un banc d'essais, les transformations sont simulées en consommant du temps processeur (c'est la tactique adoptée par TPC-DS). Dans DWEB, nous considérons que le temps de traitement des tests divers effectués lors des procédures d'insertion et de modification de don-

nées est équivalent à ces transformations simulées.

Nous nous focalisons donc dans cette section sur la partie chargement des données dans l'entrepôt. Nous avons dans ce cadre identifié quatre cas d'utilisation et conçu les stratégies de rafraîchissement correspondantes : insertions dans les tables de faits ; insertions dans les dimensions ; modifications dans les tables de faits ; modifications dans les dimensions. Comme les données sont le plus souvent historisées dans un entrepôt, nous n'avons pas considéré le cas des suppressions. Ce modèle d'ETL est également paramétré, avec le taux de rafraîchissement global, les taux de rafraîchissement respectifs des tables de faits et des dimensions et la proportion d'insertions et de modifications.

2.4. Protocole d'exécution

Le protocole d'exécution de DWEB est en fait une variante de celui de TPC-DS, qui est lui même classique pour un banc d'essais. Il est subdivisé en deux parties : un test de chargement qui consiste à alimenter l'entrepôt en données ; un test de performance qui évalue la réponse du système. Ce dernier est également subdivisé en deux étapes : une exécution à froid pendant laquelle la charge est appliquée une fois sur l'entrepôt test ; des exécutions à chaud répliquées un certain nombre (paramètre) de fois et qui incluent chacune un rafraîchissement de l'entrepôt et une exécution de la charge.

L'unique mesure de performance retenue dans DWEB est actuellement le temps de réponse. Il est calculé séparément pour les rafraîchissements et les exécutions de la charge de requêtes, de manière à ce que chaque temps de réponse atomique puisse être affiché et/ou réutilisé dans une mesure de performance composite. Les temps de réponses totaux, moyens, minimum et maximum, ainsi que les écarts-types, sont également calculés automatiquement.

3. Scénario de démonstration de DWEB

L'objectif de cette démonstration est d'illustrer la généricité de DWEB en générant différentes configurations d'entrepôts et de charges sur une même configuration matérielle. Comparer des systèmes est également possible avec DWEB, mais TPC-DS serait plus adapté pour cela. La démonstration se déroulera en quatre étapes (successivement décrites dans les sections suivantes) qui forment le cas d'utilisation typique du banc d'essais. Elles pourront être réitérées autant de fois que nécessaire pour décrire aux participants de façon interactive les subtilités du paramétrage, et notamment pour « descendre » au niveau des paramètres de bas niveau si nécessaire.

L'interface graphique principale de DWEB est représentée dans la Figure 2. Elle est divisée en trois sections/panneaux : *Database connection*, qui permet la connexion tout à tout Système de Gestion de Bases de Données Relationnel (SGBDR) via JDBC (*Java DataBase Connectivity*) ; *Action*, qui permet de fixer la valeur des paramètres et de lancer les tests ; *Information*, qui affiche des messages lorsqu'un événement ou une erreur se produit. L'utilisation de DWEB à l'aide du panneau *Action* se déroule en quatre étapes.

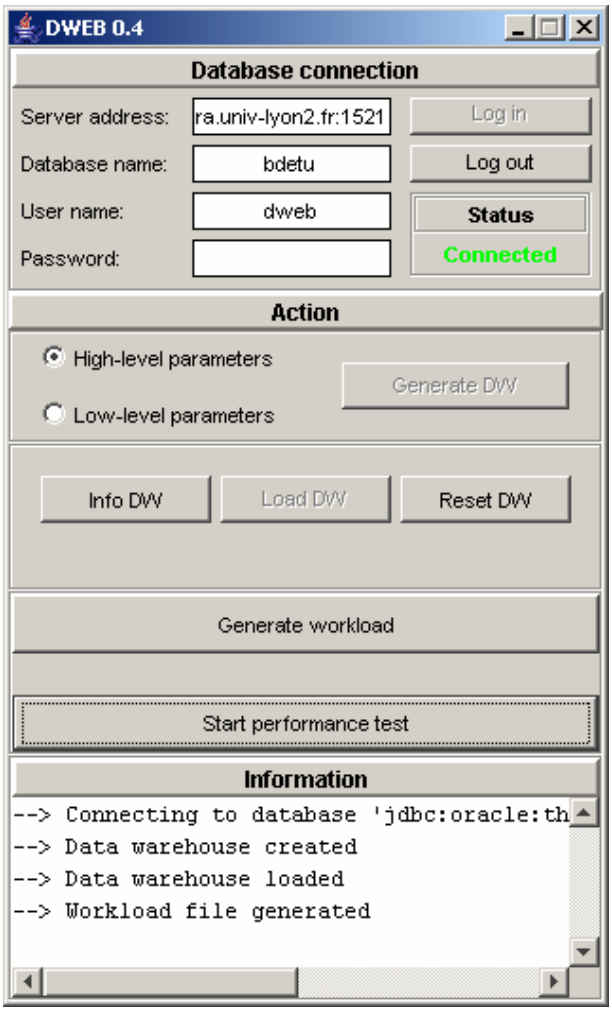

Figure 2 : Interface de DWEB

3.1. Génération de l'entrepôt

Cliquer sur le bouton de commande *Generate DW* permet de fixer la valeur de l'ensemble complet de paramètres de bas niveau de DWEB ou bien seulement des paramètres de haut niveau (Section 2.1). Nous recommandons ces derniers pour la plupart des tests de performance (Figure 3). La structure (vide) de l'entrepôt est ensuite créée automatiquement.

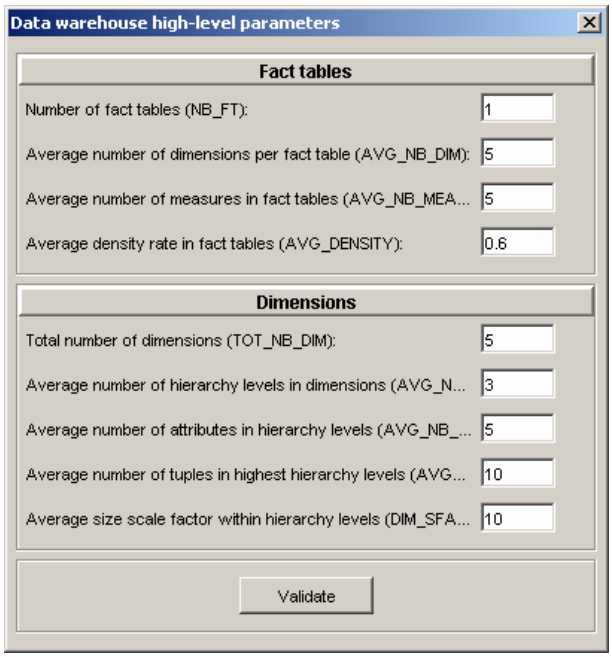

Figure 3 : Paramétrage de l'entrepôt

3.2. Test de chargement

La partie inférieure du panneau *Action* contient trois boutons de commande. Comme les paramètres de DWEB peuvent sembler abstraits au premier abord, le bouton de commande *Info DW* permet de donner une estimation de la taille de l'entrepôt en mégaoctets avant qu'il soit effectivement chargé. Ainsi, les utilisateurs peuvent ajuster les paramètres pour représenter au mieux le type d'entrepôt dont ils ont besoin.

Le bouton de commande *Load DW* permet de lancer le test de chargement (Section 2.4), dont le statut d'avancement est indiqué à l'utilisateur (Figure 4), qui peut interrompre le processus à tout moment. Lorsque le chargement est terminé, le temps de chargement est affiché.

3.3. Génération de la charge

Le bouton de commande *Generate workload* permet dans un premier temps de fixer les paramètres de la charge (Section 2.2 ; Figure 5), puis de sauvegarder les requêtes qui la composent dans un fichier externe de manière à ce qu'elle puisse être réutilisée dans des expériences ultérieures.

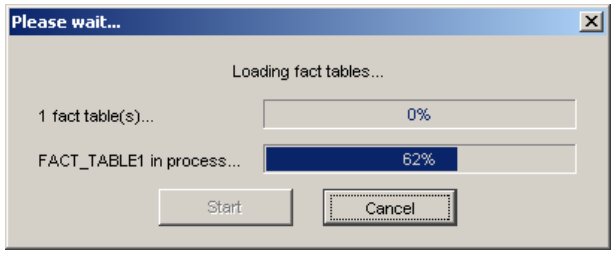

Figure 4 : Test de chargement en cours

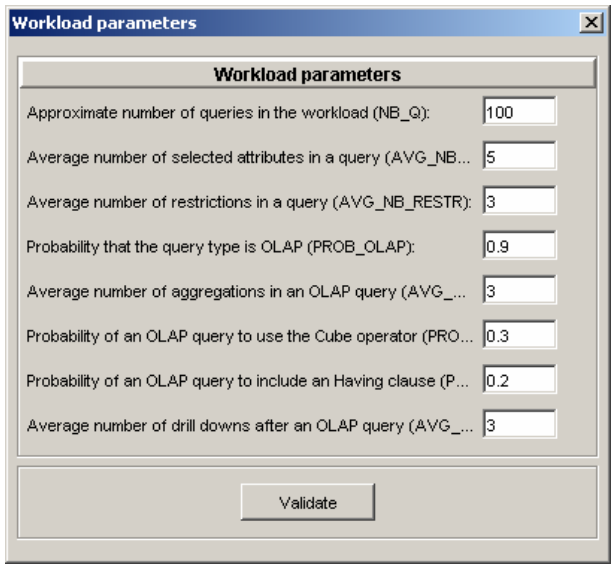

Figure 5 : Paramétrage de la charge

3.4. Test de performance

Finalement, le bouton de commande *Start performance test* permet de fixer les paramètres du processus d'ETL et du protocole d'exécution du banc d'essais (Sections 2.3 et 2.4, respectivement). Ils sont ensuite récapitulés dans la fenêtre du test de performance (Figure 6), qui permet de lancer l'exécution. Le temps de réponse de chaque rafraîchissement et exécution de la charge est affiché dans la fenêtre de résultats, et simultanément sauvegardé dans un fichier CSV (*Comma-Separated Values*) qui peut ensuite être traité à l'aide d'un tableur ou de toute autre application d'analyse. Les temps total, moyen, minimum et maximum de l'exécution à chaud, ainsi que l'écart-type, sont calculés pour les rafraîchissements, les exécutions de la charge et globalement (rafraîchissements + charge de requêtes). Les tests de performance peuvent être

réitérés autant de fois que nécessaire, en générant une nouvelle charge à chaque fois ou non.

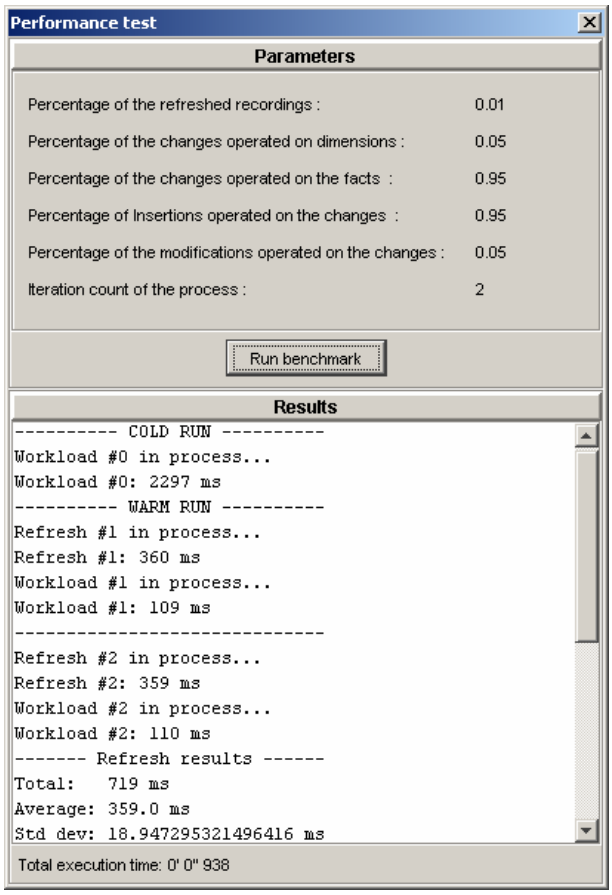

Figure 6 : Test de performance

4. Conclusion

Nous avons conçu DWEB avec en tête les critères établis par Jim Gray pour caractériser un « bon » banc d'essais [5]. Ces critères sont la pertinence (un banc d'essais doit répondre à des besoins raisonnablement divers) ; la portabilité (un banc d'essais doit pouvoir s'exécuter sur différents systèmes) ; la « scalabilité » ; la simplicité (un banc d'essais doit demeurer compréhensible pour être utilisé).

Nous avons par ailleurs proposé d'étendre le critère de scalabilité à l'adaptabilité : un banc d'essais doit proposer différentes configurations de bases de données et de charges pour permettre de mener des tests dans des conditions expérimentales variées.

DWEB a clairement été conçu pour optimiser les critères de pertinence et d'adaptabilité. Or, l'adaptabilité est orthogonale à la simplicité, qui demeure pourtant très importante. Le compromis que nous avons trouvé entre ces deux critères repose principalement dans l'établissement des paramètres de haut niveau qui régissent la génération de l'entrepôt de données (Section 2.1). Finalement, le choix du langage Java garantit la portabilité de DWEB.

Références

[1] J. Darmont, "Data warehouse benchmarking with DWEB", In D. Taniar, Ed., "Progressive Methods in Data Warehousing and Business Intelligence: Concepts and Competitive Analytics", *Advances in Data Warehousing and Mining*, Vol. 3, IGI Publishing, 2008.

[2] J. Darmont, F. Bentayeb, O. Boussaïd, "Conception d'un banc d'essais décisionnel", *20èmes Journées Bases de Données Avancées (BDA 04), Montpellier*, 2004, 493-511.

[3] J. Darmont, F. Bentayeb, O. Boussaïd, "DWEB: A Data Warehouse Engineering Benchmark", *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, 2005; *LNCS*, Vol. 3589, 85-94.

[4] J. Darmont, F. Bentayeb, O. Boussaïd, "Benchmarking Data Warehouses", *International Journal of Business Intelligence and Data Mining*, Vol. 2, No. 1, 2007, 79-104.

[5] J. Gray, Ed., *The Benchmark Handbook for Database and Transaction Processing Systems*, 2nd edition, Morgan Kaufmann, 1993.

[6] W.H. Inmon, *Building the Data Warehouse*, 3rd edition, Wiley, 2002.

[7] R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, 2nd edition, Wiley, 2002.

[8] OLAP Council, *APB-1 OLAP Benchmark Release II*, www.olapcouncil.org, 1998.

[9] Object Management Group, *Common Warehouse Metamodel (CWM) Specification version 1.1*, www.omg.org, 2003.

[10] E. Thomsen, "Comparing different approaches to OLAP calculations as revealed in benchmarks", *Intelligence Enterprise's Database Programming & Design*, www.dbpd.com/vault/9805desc.htm, 1998.

[11] Transaction Processing Performance Council, *TPC Benchmark H Standard Specification Revision 2.6.1*, www.tpc.org, 2006.

[12] Transaction Processing Performance Council, *TPC Benchmark DS Standard Specification Draft version 52*, www.tpc.org, 2007.