Université Lumière Lyon 2
École Doctorale de Sciences Cognitives

Habilitation à Diriger des Recherches
Spécialité : Informatique

présentée par

Jérôme Darmont
Maître de conférences

Laboratoire ERIC

Optimisation et évaluation de performance pour l'aide à la conception et à l'administration des entrepôts de données complexes

Mémoire scientifique

Soutenue publiquement le 23 novembre 2006 devant le jury :

Pr Danielle Boulanger	Université Jean Moulin Lyon 3	(rapporteur)
Pr Claude Chrisment	Université Paul Sabatier Toulouse III	(rapporteur)
Pr Georges Gardarin	Université de Versailles Saint-Quentin	(rapporteur)
Pr Stefano Spaccapietra	École Polytechnique Fédérale de Lausanne	(examinateur)
Pr Abdelkader Zighed	Université Lumière Lyon 2	(coordinateur)

Résumé

Les entrepôts de données forment le socle des systèmes décisionnels. Ils permettent d'intégrer les données de production d'une entreprise ou d'un organisme et sont le support de l'analyse multidimensionnelle en ligne (OLAP) ou de la fouille de données. Avec l'exploitation de plus en plus courante de données complexes dans le cadre des processus décisionnels, de nouvelles approches d'entreposage, qui exploitent notamment le langage XML, sont développées. Dans ce contexte, le problème de la performance des entrepôts de données demeure plus que jamais un enjeu crucial.

Le travail présenté dans ce mémoire vise à proposer des solutions innovantes au niveau de l'optimisation et de l'évaluation des performances des entrepôts de données. Nous avons en effet conçu une approche générique dont l'objectif est de proposer automatiquement à l'administrateur d'un entrepôt des solutions permettant d'optimiser les temps d'accès aux données. Le principe de cette approche est d'appliquer des techniques de fouille de données sur une charge (ensemble de requêtes) représentative de l'utilisation de l'entrepôt de données afin de déduire une configuration quasi-optimale d'index et/ou de vues matérialisées. Des modèles de coût permettent ensuite de sélectionner parmi ces structures de données les plus efficaces en terme de rapport gain de performance/surcharge.

Par ailleurs, l'évaluation de performance peut venir en appui de la conception des entrepôts de données. Ainsi, afin de valider notre approche de manière expérimentale, nous avons également conçu plusieurs bancs d'essais génériques. Le principe directeur qui a présidé à leur élaboration est l'adaptabilité. En effet, pour comparer l'efficacité de différentes techniques d'optimisation des performances, il est nécessaire de les tester dans différents environnements, sur différentes configurations de bases de données et de charges, etc. La possibilité d'évaluer l'impact de différents choix d'architecture est aussi une aide appréciable dans la conception des entrepôts de données. Nos bancs d'essais permettent donc de générer diverses configurations d'entrepôts de données, ainsi que des charges décisionnelles qui s'y appliquent.

Finalement, nos solutions d'optimisation et d'évaluation des performances ont été mises en œuvre dans les contextes des entrepôts de données relationnels et XML.

Mots clés : Bancs d'essais, Données complexes, Entrepôts de données, Évaluation de performance, Fouille de données, Index, OLAP, Optimisation de performance, Vues matérialisées, XML.

Abstract

Data warehouses form the basis of decision-support systems. They help integrating the production data of companies or organizations and support multidimensional on-line analysis (OLAP) or data mining. Complex data are now more and more casually exploited within decision-support processes, hence new data warehousing approaches are developed, some of which exploit the XML language. In this context, data warehouse performance remains as much as ever a crucial issue.

In this thesis, we aim at proposing novel solutions for optimizing and evaluating data warehouse performance. We have indeed designed a generic approach whose objective is to automatically propose solutions to data warehouse administrators for optimizing data access times. The principle of this approach is to apply data mining techniques on a workload (set of queries) that is representative of data warehouse usage in order to deduce a quasi-optimal configuration of indices and/or materialized views. Then, cost models help selecting among these data structures those that are the most efficient in terms of performance gain/overhead ratio.

Besides, performance evaluation may help supporting data warehouse design. Thus, in order to experimentally validate our approach, we have also designed several generic benchmarks. Their main design principle is adaptability. In order to compare the efficiency of different performance optimization techniques, it is indeed necessary to test them in various environments, on different database and workload configurations, etc. The ability to assess the impact of different architecture choices is also a valuable help in data warehouse design. Our benchmarks thus allow the generation of various data warehouse configurations, as well as associated decision-support workloads.

Eventually, our performance optimization and evaluation solutions have been implemented in both the contexts of relational and XML data warehouses.

Keywords: Benchmarks, Complex data, Data mining, Data warehouses, Indices, Materialized views, OLAP, Performance evaluation, Performance optimization, XML.

À Maëlle.

Remerciements

J'exprime tout d'abord tous mes remerciements à Madame Danielle Boulanger, Monsieur Claude Chrisment et Monsieur Georges Gardarin, qui m'ont fait l'honneur de porter intérêt à mon travail et d'en être les rapporteurs. Je remercie également de tout cœur Monsieur Stefano Spaccapietra d'avoir accepté de faire partie de mon jury et Monsieur Djamel Zighed d'avoir joué le rôle de coordinateur de mon HDR.

Je tiens également à exprimer ma gratitude envers tous mes collègues du laboratoire ERIC, permanents, doctorants et administratifs, et notamment à son directeur, Nicolas Nicoloyannis, pour ses encouragements constants. Je remercie également Stéphane Lallich et Jean-Hugues Chauchat pour leurs conseils avisés. Des pensées toutes particulières vont aux membres du pôle BDD, dont le dynamisme collectif a grandement facilité mon travail d'HDR : Fadila Bentayeb, Omar Boussaïd, Nouria Harbi, Sabine Loudcher, Riadh Ben Messaoud, Cécile Favre, Nora Maiz ; et bien sûr Kamel Aouiche, Hadj Mahboubi et Jean-Christian Ralaivao, dont j'ai ou j'ai eu le réel plaisir d'encadrer les travaux de thèse.

Enfin, je remercie mon épouse Anne-Gaëlle, pour son soutien et ses encouragements permanents, ainsi que toute ma famille et mes amis, qui ont constamment suivi l'évolution de mes complexes (vues de l'extérieur) activités universitaires.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte du travail

1.1.1 Avènement des données complexes

La technologie des entrepôts de données (*data warehouses*, dans la terminologie anglo-saxonne) et de l'analyse multidimensionnelle en ligne OLAP (*On-Line Analytical Processing*) développe des outils décisionnels qui permettent d'étudier, par exemple, le comportement de consommateurs, de produits, de sociétés ; d'effectuer une veille concurrentielle ou technologique, etc. Pour cela, ils intègrent traditionnellement des données dites de production dans une base de données centralisée à vocation décisionnelle (l'entrepôt), où elles sont agrégées, historisées et structurées de manière à en permettre et à en optimiser l'analyse en ligne. Par ailleurs, les entrepôts de données peuvent également servir de socle à un processus d'Extraction de Connaissances à partir des Données (ECD), plus précisément en tant que source de données pour des méthodes de fouille (*data mining*), grâce à des vues extraites de l'entrepôt et formatées sous la forme de tableaux attributs-valeurs exploitables par les algorithmes de fouille.

Dans la grande majorité des cas, les activités étudiées sont matérialisées sous la forme de données numériques et symboliques. Or, les données exploitées dans le cadre des processus décisionnels sont de plus en plus diverses et hétérogènes. En effet, le besoin d'exploiter des données qui ne sont ni numériques ni symboliques se fait jour dans de nombreux domaines : gestion de la relation client, marketing, veille concurrentielle, médecine... L'avènement du Web et la profusion de données multimédias ont en grande partie contribué à l'émergence de ces données, que nous qualifions de complexes. Par exemple, l'analyse de données médicales relatives à des sportifs de haut niveau nous a menés à construire un entrepôt qui permet d'intégrer et d'exploiter conjointement différentes informations disponibles sous des formes variées : dossiers de patients (base de données classique), antécédents médicaux (textes), radiographies, écographies (documents multimédia), électrocardiogrammes (séries temporelles), diagnostics de médecins (textes ou enregistrements audios), etc. [DO06, DO07]

Un domaine de recherche centré sur les données complexes, notamment sur leur entreposage et leur fouille, émerge depuis quelques années. Il est relayé par des ateliers adossés à de grandes conférences nationales [GT04, GM05a, BT06] et internationales [KP04, BKM05, ZTR05] et est l'objet de premiers ouvrages de référence [ZSD03, DB06]. Cependant, le concept de donnée complexe varie selon les chercheurs, y compris au sein d'une même communauté. Aussi proposons-nous une définition, en nous basant sur les travaux précités. Dans un premier temps, des données sont ainsi qualifiées de complexes si elles sont [DBRA05] :

1. *multiformats* : l'information est portée par des données de types différents (données numériques, symboliques, textes, images, sons, vidéos...) ;
   et/ou
2. *multistructures* : les données peuvent être structurées, non structurées ou semi-structurées (bases de données relationnelles, collection de documents XML...) ;
   et/ou
3. *multisources* : les données proviennent de plusieurs origines différentes (bases de données réparties, Web...) ;
   et/ou
4. *multimodales* : un même phénomène est décrit par plusieurs canaux ou points de vue (radiographies et diagnostic audio d'un médecin pour évaluer l'état de santé d'un patient, données exprimées dans des échelles ou des langues différentes...) ;
   et/ou
5. *multiversions* : les données sont évolutives en termes de définition ou de valeur (bases de données temporelles, recensements périodiques dont les critères évoluent...).

Cependant, cette première définition est insuffisante et ne couvre pas totalement la grande variété des données complexes. Elle pourrait par contre être envisagée comme un axe de complexité parmi d'autres, comme des axes liés à la sémantique des données ou à leur traitement (Figure 1.1). La volumétrie des données pourrait également constituer un tel axe. Bien que le volume des données ne soit pas une expression de complexité intrinsèque s'il est vu en termes de n-uplets d'une base de données, il devient un problème difficile en statistique ou en fouille de données lorsque c'est le nombre d'attributs qui augmente.

En conclusion, nous définissons ici un cadre permettant d'indentifier ce que nous appelons des données complexes. Cependant, cette définition ne pouvant être exhaustive, nous la laissons aussi ouverte que possible à d'autres axes de complexité.

1.1.2 Entrepôts de données complexes et performance

Un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles, historisées et organisées pour supporter un processus d'aide à la décision [Inm02]. Son objectif principal est de supporter des requêtes de type OLAP [Tho04]. Un entrepôt de données présente une modélisation dite dimensionnelle qui se compose classiquement

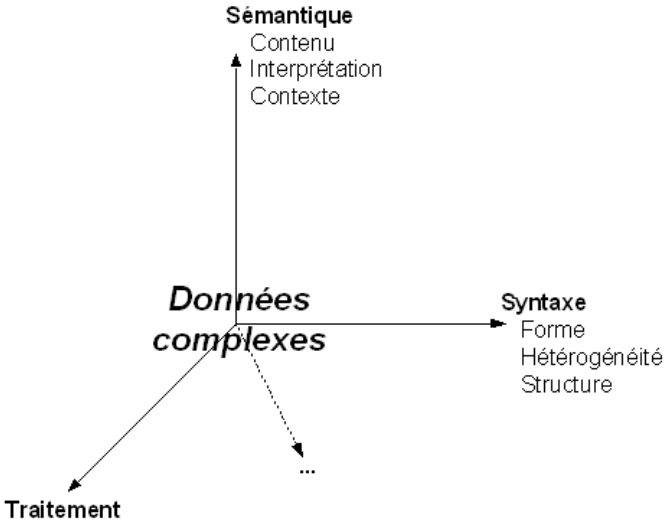

FIG. 1.1 – Axes de complexité des données

d'une table des faits centrale et d'un ensemble de tables de dimensions. Cette modélisation conceptuelle a pour objectif d'observer les faits à travers des mesures, en fonction des dimensions qui représentent les axes d'analyse. Ce modèle est qualifié de modèle en étoile.

L'émergence des données complexes impose une nouvelle approche des processus décisionnels. En effet, les architectures classiques d'entrepôts de données [Inm02, KR02] ont montré leur utilité et leur efficacité lorsque les données sont « simples » (numériques ou symboliques). En revanche, elles doivent être complètement reconsidérées lorsque les données sont complexes. Par exemple, le concept d'un entrepôt centralisé n'est plus nécessairement pertinent dans tous les cas de figure. En effet, la spécificité des données complexes, ainsi que leurs lieux de stockage physique, peuvent amener à envisager de nouvelles solutions basées sur l'entreposage virtuel [BCH99] ou les approches orientées architecture. L'intégration des données grâce à des systèmes de médiation [BBD03] peut également être considérée comme une nouvelle façon d'aborder le processus d'extraction, transformation et chargement (*Extract, Transform, Load* ou ETL dans la terminologie anglo-saxonne [KC04]). De plus, la technologie OLAP doit désormais dépasser sa vocation initiale pour permettre des analyses plus efficaces. Le couplage d'OLAP et de la fouille de données est également un nouveau problème posé par la prise en compte des données complexes [MBR04]. L'intrégration de méthodes de fouille directement dans les SGBD supports des entrepôts [BD02, UBDB04, BDU04, BDFU07], plutôt que leur usage « externe » habituel, pourrait contribuer à atteindre cet objectif.

Néanmoins, les principes de base de l'entreposage demeurent valides pour les données complexes. Les faits à observer sont toujours définis par des mesures. Par exemple, l'état de santé d'un patient pourrait être un fait matérialisé par les résultats d'une série d'examens cliniques [DO06, DO07]. Les mesures sont classiquement numériques et additives, mais cela n'est plus strictement nécessaire dans le contexte des données complexes. On peut par exemple imaginer qu'une radiographie des poumons soit une mesure. Comme dans l'ap-

proche classique, les mesures demeurent toutefois les indicateurs de l'analyse. De même, les faits demeurent décrits par des dimensions qui permettent de les envisager sous différentes perspectives ou axes d'analyse. Dans notre exemple, les dimensions pourraient être le patient, le temps, le centre de soins où sont effectués les examens cliniques, etc. Pour finir, les grands volumes de données à stocker ainsi que leur historisation plaident également en faveur d'une approche d'entreposage [DBB02, BBDR03, DBB+03, BTBD06, BDBL07].

Dans ce contexte, le problème de la performance des systèmes décisionnels est un enjeu crucial et qui demeure actuel dans les entrepôts de données classiques. L'analyse en ligne nécessite en effet, par définition, des temps de réponse courts, alors que les entrepôts stockent des volumes de données susceptibles d'être énormes. Or, la taille des mémoires centrales n'a pas augmenté dans la même proportion que celle des disques, en raison de contraintes techniques et de coût. En attendant le développement de mémoires *flash* de très grande capacité susceptibles de remplacer les disques magnétiques actuels, le ratio entre les temps d'accès sur mémoire secondaire et primaire demeure très important. Les caches mémoires étant nécessairement de taille limitée, il est indispensable de recourir à des techniques d'optimisation des performances pour garantir un bon temps de réponse aux requêtes décisionnelles. Les architectures d'entrepôts pouvant être différentes, ce problème se pose à nouveau pour les données complexes, notamment lorsqu'elles sont structurées sous la forme de documents XML (*eXtensible Markup Language* [BPSM+04]) [MDB01, DBB+03, BTBD06, BDBL07], l'efficacité des systèmes de gestion de bases de données (SGBD) XML étant encore relativement faible.

Afin de déterminer la performance de ces techniques d'optimisation, il est nécessaire de disposer d'outils d'évaluation des performances. En effet, si, d'un côté, ces méthodes permettent des gains en temps de réponse, d'un autre, elles engendrent invariablement une surcharge de traitement et de stockage pour le système, soit lors de leur utilisation en temps réel (optimisation de requêtes, par exemple), soit lors de la maintenance de structures de données physiques comme des index ou des vues matérialisées. Il est donc primordial d'évaluer le gain de performance effectif en tenant compte de ces contraintes. De plus, il est intéressant de pouvoir comparer entre elles différentes stratégies. Il est ainsi possible d'apprécier quelle méthode est la meilleure dans des conditions données.

Par ailleurs, l'évaluation de performance peut aussi venir en appui de la conception des entrepôts de données. En effet, différents choix d'architectures sont généralement possibles, qui sont cruciaux et très dépendants du domaine d'application et des objectifs d'analyse finaux. Ils présentent différents défauts et avantages qu'il faut prendre en compte *a priori* et qui influencent grandement le temps de réponse des requêtes décisionnelles. Par exemple, dans un environnement OLAP relationnel (*Relational OLAP* ou ROLAP), un schéma en flocon de neige (où les dimensions sont structurées en niveaux de hiérarchie correspondants à des granularités différentes, par exemple ville-région-pays dans une dimension géographique) permet d'affiner les analyses lorsque la plupart des requêtes s'appliquent aux plus hauts niveaux hiérarchiques, mais induit de nombreuses opérations de jointure coûteuses entre les niveaux hiérarchiques qui n'existeraient pas au sein d'un schéma en étoile simple,

sans hiérarchie.

Pour terminer, 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 développement des bases de données en général et des entrepôts de données en particulier, il est devenu indispensable de réduire la fonction d'administration des SGBD [CN98, WMHZ02] ou du moins de fournir des outils d'aide pour l'administrateur en automatisant ses tâches au maximum. Ces fonctionnalités sont particulièrement critiques dans le cas des entrepôts de données complexes, du fait de l'hétérogénéité même des données stockées.

En conclusion, il apparaît donc qu'il existe un besoin fort d'outils d'aide à la conception et à l'administration des entrepôts de données en général et plus particulièrement des entrepôts de données complexes.

1.2 Contributions et organisation du mémoire

Ce travail vise à proposer des solutions au niveau de l'optimisation et de l'évaluation des performances des bases de données en général et des entrepôts en particulier, notre objectif final étant de traiter la problématique des données complexes. Pour cela, nous avons procédé par étapes en traitant en premier lieu le cas de bases et entrepôts de données classiques. Nous avons ensuite appliqué ou étendu les principes proposés dans un premier temps au contexte des données complexes structurées sous la forme de documents XML.

Les avantages de l'utilisation du langage XML pour représenter des données complexes sont en effet nombreux [DBB+03, DBRA05, BTBD06, BDBL07]. Tout d'abord, XML permet d'encapsuler les données elles-mêmes ainsi que leur schéma, soit implicitement, soit à l'aide d'une définition de schéma. Ce type de représentation se retrouve au sein des entrepôts de données où des métadonnées décrivant les données sont également stockées et exploitées. XML est donc particulièrement adapté à nos besoins. De plus, nous bénéficions de la flexibilité, de l'extensibilité et de la richesse du modèle de données semi-structurées. Les possibilités de stockage de documents XML sont multiples, que ce soit dans des SGBD relationnels dits « compatibles XML » (comme Oracle, SQL Server ou DB2), après un *mapping* [KKRSR00] ou dans des SGBD natifs XML tels que Lore [MAG+97] ou plus récemment eXist [Mei02] et X-Hive [Wal05]. Finalement, les langages d'interrogation XML tels que XQuery [BCF+05] permettent de formuler des requêtes d'analyse qui sont difficiles à exprimer dans un système relationnel, comme des agrégations sur fenêtre mobile ou des opérations de *rollup* suivant des hiérarchies dynamiques [BCC+04, BCC+05].

Ce mémoire est organisé en deux grandes parties. La première (Chapitre 2) est consacrée à l'optimisation des performances des entrepôts de données. Ce domaine recouvre principalement les techniques relevant de l'indexation, de la matérialisation des vues et du partitionnement, qui sont principalement héritées des bases de données relationnelles [Bel00]. Cependant, les techniques d'indexation utilisées dans les bases de données de

production de type OLTP (*On-Line Transaction Processing*) ne sont pas adaptées aux entrepôts de données. En effet, une requête OLTP typique n'accède qu'à un nombre relativement peu élevé de n-uplets, tandis que les requêtes OLAP effectuent des opérations complexes (jointures et agrégations) sur de gros volumes de données. Des statégies d'indexation spécifiques ont donc dû être adaptées ou développées pour les entrepôts.

Les vues matérialisées sont des relations qui contiennent le résultat d'une requête. Elles permettent de précalculer des opérations complexes et de les stocker dans la base de données. Ainsi, des requêtes simples nécessitant uniquement un accès aux vues matérialisées, et non plus aux données sources, peuvent être exécutées rapidement. Les problèmes majeurs liés à la matérialisation de vues sont leur sélection et leur maintenance.

Finalement, le partitionnement consiste à diviser une relation en plusieurs fragments, disjoints ou non. La fragmentation peut être verticale (obtenue par projection), horizontale (obtenue par restriction) ou mixte. Elle favorise alors le traitement des requêtes de projection ou de restriction, respectivement. Le groupement (*clustering* dans la terminologie anglo-saxonne) peut également être vu comme un type de partitionnement au niveau physique. Il consiste à stocker des données logiquement liées entre elles (c'est-à-dire, susceptibles d'être fréquemment utilisées conjointement) proches les unes des autres en mémoire secondaire, typiquement dans des pages disques contiguës [Dar99]. Ainsi, quand un objet (au sens général du terme) est chargé en mémoire primaire, tous ses objets liés le sont en même temps en un minimum d'entrées/sorties. Dans les entrepôts de données, cette technique a été utilisée pour grouper les n-uplets de la table de faits [JLS99].

Dans ce contexte, nous avons conçu une approche générique dont l'objectif est de proposer à l'administrateur de bases de données, de la façon la plus automatique possible, des configurations d'index et de vues matérialisées permettant d'optimiser le temps d'accès aux données [AD07]. Le principe de cette approche est d'appliquer des techniques de fouille de données sur une charge (ensemble de requêtes) représentative de l'utilisation de la base ou de l'entrepôt, afin d'extraire des motifs fréquents ou une classification permettant de déduire des index et/ou des vues matérialisées. Cette approche a été successivement appliquée aux bases de données [Aou02, ADG03a, ADG03b], aux entrepôts de données classiques [ADG03c, ADG03d, ADB04, Mai05, ADBB05b, Aou05, MAD06c, AJD06] et aux entrepôts de données complexes [ADBB05a, Aou05] et XML en particulier [Mah05, Aou05, MAD06a].

La seconde partie du mémoire (Chapitre 3) est consacrée à l'évaluation des performances des bases et entrepôts de données. Trois approches se dégagent dans ce domaine. Tout d'abord, l'analyse mathématique est le plus souvent utilisée pour déterminer de façon exacte la complexité ou le coût (grâce à des modèles de coûts) d'algorithmes spécifiques. Elle possède cependant des limites lorsqu'il faut appréhender le comportement global d'un système. En effet, ce comportement est généralement régi par un tel nombre de paramètres qu'il est nécessaire de recourir à des hypothèses simplificatrices qui font diverger les résultats théoriques de ceux observés dans la réalité. Une deuxième approche est la simulation, qui permet de réaliser des études de performance *a priori*. Très répandue dans les pays

anglo-saxons, elle rencontre plus de méfiance en France car elle demeure tributaire de la qualité des modèles de simulation utilisés, qui sont susceptibles de souffrir des mêmes défauts que les modèles mathématiques. Finalement, la manière la plus commune d'évaluer les performances d'un SGBD s'effectue par expérimentation « grandeur nature » à l'aide de bancs d'essais (ou *benchmarks*, dans la terminologie anglo-saxonne). Ce type d'outils est généralement constitué d'une base de données sur laquelle est exécutée une charge prédéfinie. Divers critères de performance sont ensuite mesurés directement lors de l'exécution du banc d'essais.

Nous avons conçu plusieurs bancs d'essais pour évaluer les performances des bases de données en général, des entrepôts de données et des entrepôts de données XML. Ces outils nous ont permis de tester l'efficacité de nos propositions en matière d'optimisation des performances. Le principe directeur qui a présidé à leur élaboration est la paramétrabilité. En effet, pour comparer l'efficacité de différentes techniques d'optimisation des performances, il est nécessaire de les tester dans différents environnements, sur différentes configurations de bases de données et de charges, etc. L'objectif d'aide à la conception nécessite également de pouvoir tester différents choix d'architecture afin de sélectionner celui qui correspond le mieux à des besoins donnés. Dans cet esprit, nous avons conçu trois bancs d'essais.

Le premier est en fait plutôt une plate-forme générique pour l'évaluation du comportement dynamique des applications de bases de données [HD03a, HD03b, HD04, HD05, HD06]. Baptisé DEF (*Dynamic Evaluation Framework*), il permet de simuler les changements dans les séquences d'accès aux données, alors que les autres bancs d'essais sont dotés de charges qui accèdent toujours aux mêmes données dans le même ordre. Notre deuxième proposition, DWEB (*Data Warehouse Engineering Benchmark*), est actuellement un des rares bancs d'essais décisionnels opérationnels permettant d'évaluer les performances des entrepôts de données [DBB04, DBB05, DBB07]. Il permet notamment de générer différents entrepôts de données synthétiques *ad-hoc*, ainsi que des charges de requêtes associées. Finalement, notre troisième banc d'essais, XDW-Bench (*XML Data Warehouse Benchmark*), est dédié aux entrepôts de données XML [Dio05]. Plus simple que DWEB pour des raisons techniques (Section 3.4), il permet d'exécuter une charge de requêtes décisionnelles exprimées à l'aide du langage XQuery sur un entrepôt de données modélisé sous la forme de documents XML.

Chapitre 2

Optimisation automatique des performances des entrepôts de données

L'objectif de ce chapitre est tout d'abord de présenter et de discuter l'existant en matière de techniques de sélection d'index et de vues matérialisées dans le cadre spécifique des entrepôts de données. Nous motivons et présentons ensuite l'approche générique basée sur la fouille de données que nous avons conçue pour assister, de la façon la plus automatique possible, les administrateurs d'entrepôts de données dans leur tâche d'optimisation des performances. Nous illustrons finalement cette approche par trois applications :

- sélection automatique d'index ;
- sélection automatique de vues matérialisées ;
- sélection conjointe des index et des vues matélialisées.

Ce travail a fait l'objet de la thèse de doctorat de Kamel Aouiche, préparée au sein du laboratoire ERIC dans le cadre de l'Action Concertée Incitative « Terrains, Techniques, Théories » Nomex-CLAPI (Corpus de Langues Parlées en Interaction [ABBD03]), co-encadrée par Djamel Abdelkader Zighed et moi-même, et soutenue en 2005. Il a en partie été mené en collaboration avec le Pr. Le Gruenwald, de l'Université d'Oklahoma, USA.

2.1 Optimisation des performances des entrepôts de données

Nous présentons dans cette section les approches existantes de sélection d'index et de vues matérialisées, de façon globale. Le lecteur intéressé pourra se reporter à la thèse de doctorat de Kamel Aouiche pour plus de détails [Aou05].

2.1.1 Formalisation du problème de sélection d'index et de vues matérialisées

Soient I_C et V_C deux ensembles d'index et de vues matérialisées, respectivement, qui sont qualifiés de candidats et susceptibles de réduire le coût d'exécution d'un ensemble de requêtes Q, généralement supposé représentatif de la charge du système. Posons $O_C = I_C \cup V_C$. Soit S l'espace disque alloué par l'administrateur de l'entrepôt de données à la création des objets (index ou vues) de l'ensemble O_C. Le problème de sélection conjointe d'index et de vues matérialisées consiste à construire une configuration d'objets $O \subseteq O_C$ minimisant le coût d'exécution de Q, sous contrainte d'espace de stockage. Ce problème NP-complet [Com78, Gup99] peut se formaliser comme suit :

- $coût(Q, O) = min(coût(Q, \vartheta)) \; \forall \; \vartheta \subseteq O_C$;
- $\sum_{o \in O} taille(o) \leq S$, où $taille(o)$ est l'espace disque occupé par l'objet o.

2.1.2 Sélection d'index

Les heuristiques de sélection d'index proposées dans la littérature pour approcher une solution optimale se distinguent principalement selon les caractéristiques suivantes :

- la méthode de sélection de l'ensemble d'index candidats I_C,
- la méthode de sélection d'une configuration finale d'index I,
- les modèles de coûts utilisés pour cette seconde sélection.

Sélection de l'ensemble d'index candidats

Les premières approches de sélection d'index qui ont été proposées dans la littérature font appel à l'administrateur de l'entrepôt de données pour constituer manuellement l'ensemble d'index candidats [Wha85, FON92, CBC93, Gun99, KLT03]. Cette tactique a l'avantage de faire appel à l'expertise humaine, mais la tendance dans les travaux plus récents est de recourir à une approche automatique, plus susceptible d'être déployée à grande échelle. Pour cela, une analyse syntaxique des requêtes de la charge Q est réalisée afin d'identifier les attributs intéressants à indexer, qui sont principalement présents dans les clauses `Where` et `Group by` des requêtes SQL [CD97, VZZ$^+$00, GRS02, FR03].

Sélection de la configuration finale d'index

Construction gloutonne. La construction de la configuration finale d'index est le plus souvent assurée par un algorithme glouton. Des méthodes ascendantes partent d'une configuration d'index vide et y ajoutent itérativement des index qui minimisent le coût de la charge jusqu'à ce qu'il ne diminue plus [CN97, VZZ$^+$00, GRS02]. Au contraire, les méthodes descendantes ont comme point de départ l'ensemble complet des index candidats. À chaque itération, des index sont élagués jusqu'à ce que le coût de la charge ne diminue plus [FON92]. Finalement, certaines méthodes adoptent une approche mixte ascendante et descendante [Wha85, CBC93, Gun99].

Sac à dos. Dans certains travaux, le problème de sélection d'index a été assimilé au problème du sac à dos [ISR83, Gun99, FR03]. Les index sont alors assimilés à des objets, l'espace de stockage des index aux poids des objets, le coût de la charge en présence d'un index au bénéfice de l'objet correspondant, et l'espace disque alloué au stockage de la configuration d'index finale à la taille du sac à dos.

Algorithmes génétiques. Les algorithmes génétiques ont également été adaptés au problème de sélection d'index [KLT03]. La population de départ est alors l'ensemble des index candidats et la construction combinatoire des configurations d'index est classiquement réalisée à l'aide des opérateurs génétiques de croisement, de mutation et de sélection.

Modèles de coûts

Les méthodes de sélection d'index exploitent deux types de modèles de coût : soit une fonction mathématique *ad-hoc* [Wha85, CBC93, Gun99, KLT03, FR03], soit l'optimiseur de requêtes du SGBD [FON92, CN97, VZZ+00, GRS02]. L'utilisation d'une fonction *ad-hoc* a l'avantage de la rapidité, mais implique des hypothèses simplificatrices parfois trop irréalistes. Faire appel à l'optimiseur de requête est plus fiable, mais rend la sélection d'index dépendante d'un SGBD donné et s'avère plus gourmande en temps de calcul.

2.1.3 Sélection de vues matérialisées

Deux grandes classes de travaux ont abordé le problème de la sélection de vues matérialisées. La première privilégie l'optimisation des performances de l'interrogation de l'entrepôt et fonctionne sous contrainte d'espace de stockage des vues matérialisées. La seconde vise à ne pas engendrer de temps de maintenance des vues matérialisées trop important [GM99]. Elle est employée dans le cas où les rafraîchissements de l'entrepôt sont conséquents ou leur fréquence élevée. Certaines approches essayent de résoudre simultanément ces deux problèmes [dSS99], mais identifient de nombreux cas d'incompatibilité des objectifs qui rendent le calcul d'une solution impossible. Pour notre part, nous nous intéressons plus particulièrement à la première de ces problématiques, à laquelle la formalisation de la Section 2.1.1 fait référence.

Les heuristiques proposées dans la littérature pour approcher une solution optimale ont un principe très proche de celles qui ont été développées pour la sélection d'index (Section 2.1.2). Dans une première étape, elles construisent l'ensemble de vues matérialisées candidates V_C à partir de l'ensemble Q de requêtes extraites de la charge du système. Cet ensemble est structuré de manière à prendre en compte les relations susceptibles d'exister entre les vues candidates. Les premières différences entre les solutions existantes apparaîssent au niveau des structures de données qui représentent ces relations : treillis [HRU96, BPT97, URT99, SDN00, NT02], graphe [dSS99, GR00, TX04, GM05b], plan d'exécution des requêtes [ZYY01, VVK02, BB03b], ondelettes [SLJ04] et structures similaires [SDRK02], ou résultat d'analyse syntaxique de la charge [ACN00].

Dans une seconde étape, les algorithmes sélectionnent des vues matérialisées dans l'ensemble V_C à l'aide d'un algorithme glouton [HRU96, BPT97, URT99, ACN00, SDN00, NT02, VVK02, GM05b], de méthodes issues de la recherche opérationnelle (sac à dos [BB03b] ou méthode hongroise [dSS99]) ou d'un algorithme génétique [ZYY01]. Ces algorithmes exploitent des modèles de coût *ad-hoc* [HRU96, BPT97, URT99, ZYY01, NT02, VVK02, BB03b, SLJ04, GM05b] ou font appel à l'optimiseur de requêtes du SGBD [ACN00].

Notons finalement que tous ces algorithmes sont statiques. Si la charge du système évolue, il est alors nécessaire de réexécuter une sélection de vues matérialisées, à moins de recourir à une stratégie dynamique comme DynaMat [KR99]. Ce système rafraîchit la configuration de vues matérialisées si sa taille dépasse la taille de l'espace de stockage qui lui est allouée. Pour cela, il emploie divers critères de remplacement (par exemple, les vues les moins utilisées sont supprimées). Une approche hybride améliore ce principe et propose d'exploiter conjointement un ensemble statique de vues persistantes utilisées par plusieurs séquences de requêtes et de maintenance, et un ensemble dynamique de vues agrégées et de taille plus réduite, accessibles et remplaçables à la volée [SRR06]. Ces approches se focalisent cependant sur la performance du rafraîchissement des vues matérialisées et non sur celle des requêtes d'interrogation, ce qui constitue le problème que nous traitons plus particulièrement.

2.1.4 Sélection simultanée d'index et de vues matérialisées

Les travaux présentés dans les Sections 2.1.2 et 2.1.3 traitent des problèmes de sélection d'index et de vues matérialisées de façon isolée. Or, ces structures de données partagent des caractéristiques fondamentalement similaires : objectif identique (réduction du temps de réponse des requêtes), utilisation du même espace disque, surcharge de maintenance pour le système [ACN00]. Elles peuvent de plus entrer en interaction, la présence d'un index pouvant par exemple rendre une vue matérialisée plus efficace. Néanmoins, les travaux sur la sélection conjointe des index et des vues matérialisées sont peu nombreux dans la littérature.

La première approche qui a traité de ce problème s'est limitée à la sélection d'index sur des vues matérialisées présélectionnées (elle ne prend en effet pas en compte les index sur l'entrepôt initial) [Gup99]. Pour cela, elle construit une configuration d'objets (vues et index associés) de façon itérative, en lui ajoutant à chaque étape l'ensemble (vue + index) ou l'index (sur une vue de la configuration) qui minimise le temps de réponse d'une charge donnée.

Agrawal *et al.* ont élaboré trois approches pour la sélection conjointe d'index sur des vues matérialisées [ACN00]. La première, baptisée MVFIRST, consiste à sélectionner les vues matérialisées dans un premier temps, puis les index, en prenant en compte les vues déjà créées. La deuxième approche, INDFIRST, effectue la sélection dans l'ordre inverse. Finalement, l'approche *joint enumeration* traite la sélection des index et des vues en une

seule étape. Elle est annoncée comme la plus efficace par les auteurs, mais ils ne donnent malheureusement aucun détail sur son fonctionnement.

Bellatreche *et al.* ont abordé ce problème sous l'angle de la distribution de l'espace de stockage entre les index et les vues matérialisées [BKS00]. Cet espace S est subdivisé en deux sous-ensembles S_I et S_V, qui ne stockent que des index et des vues, respectivement. Une configuration initiale est construite par sélection séparée d'index et de vues matérialisées respectivement stockés dans S_I et S_V. L'approche reconsidère ensuite itérativement cette solution initiale en mettant en compétition deux agents qui « volent » de l'espace aux index pour créer de nouvelles vues matérialisées et *vice versa*. L'algorithme s'arrête lorsque le coût d'une charge de requêtes donnée ne peut plus être réduit grâce à ces opérations.

Rizzi et Saltarelli ont proposé une approche qui détermine *a priori* le bon compromis entre l'espace de stockage alloué aux index et aux vues matérialisées, respectivement [RS03]. Pour cela, ils se basent sur une analyse des requêtes de la charge et plus particulièrement du niveau d'agrégation et de la sélectivité des attributs présents dans les clause `Where`, `Group by` et `Having` des requêtes SQL. La Figure 2.1 montre les cas identifiés par les auteurs, qui proposent un modèle mathématique pour déterminer un compromis lorsque la solution ne peut pas être déterminée directement.

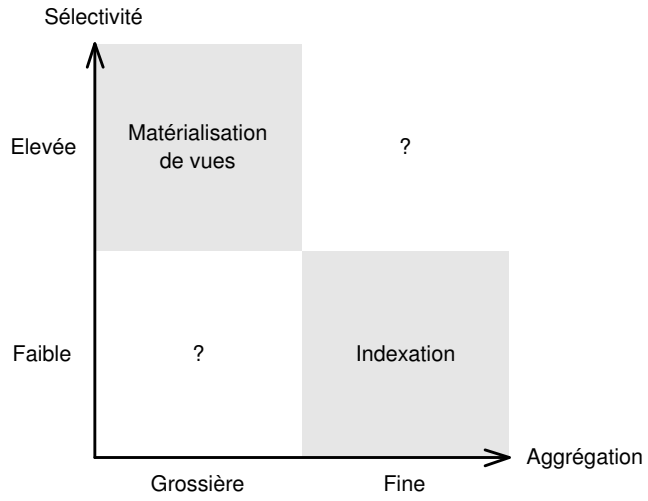

Fig. 2.1 – Recommandation de technique d'optimisation de Rizzi et Saltarelli

Finalement, Bruno et Chaudhuri ont récemment travaillé sur l'affinage de la conception physique des bases de données relationnelles [BC06], dont l'objectif est d'améliorer automatiquement la conception physique d'un expert, afin de prendre en compte les contraintes primordiales qu'elle est souvent susceptible de violer. Dans cette optique, ils ont proposé une architecture de transformation basée sur deux primitives de fusion et de réduction, qui permet de traiter les index et les vues matérialisées de manière unifiée.

2.1.5 Discussion

Les travaux existants concernant la sélection d'index et de vues matérialisés sont très nombreux et variés dans le domaine des bases de données relationnelles et également bien développés dans celui des entrepôts de données. Nous avons cependant identifié deux points principaux qui sont susceptibles d'être améliorés dans ces approches.

Sélection des objets candidats

La sélection des objets (index et vues matérialisées) candidats est rarement le cœur des approches existantes qui, pour la plupart, présentent un problème de passage à l'échelle à ce niveau. En effet, de nombreuses stratégies de sélection d'index reposent sur l'expertise de l'administrateur de l'entrepôt pour proposer une configuration initiale. Étant données la taille et la complexité de la majorité des entrepôts de données, une approche automatique est indispensable pour déployer ces méthodes sur une échelle réelle. C'est d'ailleurs l'option retenue par les travaux les plus récents, qui construisent la configuration d'index initiale à partir de la charge de travail du système.

Pour la sélection de vues matérialisées, diverses structures de données ont été proposées (treillis, graphes, ondelettes...) pour modéliser les relations entre vues. Toutes présentent également des limitations pour le passage à l'échelle. Par exemple, le parcours d'un treillis de vues candidates est très coûteux lorsque le cube de données en entrée est de forte dimensionnalité. De même, la construction de graphes de vues est d'autant plus difficile que la charge considérée en entrée est volumineuse. Il est donc nécessaire d'évaluer avec soin la complexité d'une stratégie avant de l'adopter et d'optimiser les structures de données employées.

Prise en compte des interactions entre objets

Aucune des approches présentées à la Section 2.1 ne prend en compte les interactions qui peuvent exister entre index, entre vues matérialisées et entre index et vues, y compris les méthodes de sélection simultanée de ces objets. En effet, les travaux existants, et notamment ceux qui assimilent le problème de sélection à celui du sac à dos ou qui utilisent des algorithmes génétiques, calculent le coût ou le bénéfice associé à un objet (index ou vue matérialisée) une seule fois avant de l'injecter dans leur algorithme. Or, l'intérêt de sélectionner un objet peut varier d'une itération à une autre si un objet en interaction avec le premier a été préalablement sélectionné. Il est donc indispensable de recalculer ce coût ou ce bénéfice de façon dynamique avant d'opérer une sélection.

La solution qui s'en rapproche le plus est celle de Bellatreche *et al.* [BKS00]. Cependant, les politiques de remplacement d'objets dans les espaces disque réservés aux index et aux vues qu'elle intègre ne reflètent pas réellement l'interaction index-vues matérialisées. En effet, elle considère la fréquence d'utilisation conjointe de ces structures dans les requêtes et non le bénéfice qu'elles apportent les unes par rapport aux autres.

2.2 Approche automatique d'optimisation des performances des entrepôts de données

2.2.1 Motivation

Notre objectif dans ce travail est de résoudre les problèmes évoqués à la Section 2.1.5. En premier lieu, afin d'assurer le passage à l'échelle de la phase de sélection des objets candidats (index et vues matérialisées), il est nécessaire de mettre en œuvre une démarche automatique. Cela passe en général par l'analyse syntaxique de la charge de requêtes supportée par le système, qui permet de repérer les attributs susceptibles d'être le support d'index ou de vues matérialisées. Ces attributs sont ensuite combinés de manière systématique pour proposer des index multi-attributs ou des treillis ou graphes de vues exhaustifs. Or, cette stratégie conduit ensuite à considérer, lors de la phase de sélection, des objets non pertinents, c'est-à-dire des objets qui sont bien présents dans la charge, mais qui ne présentent pas d'intérêt du point de vue de l'indexation ou de la matérialisation.

Afin d'éliminer d'emblée ces objets non pertinents, nous proposons d'exploiter des techniques de fouille de données pour extraire directement de la charge une configuration d'objets candidats pertinents. Notre idée est de faire ressortir les co-occurrences et les similitudes entre objets de la charge. Dans le cas de l'indexation, nous nous basons par exemple sur l'intuition que l'importance d'un attribut à indexer est fortement corrélée avec sa fréquence d'apparition dans la charge. Pour les vues matérialisées, dégager des classes de requêtes similaires permet également de construire des vues susceptibles de répondre à toutes les requêtes d'une classe donnée.

Sur la base d'un ensemble le plus réduit possible d'objets candidats tous pertinents, il faut ensuite avoir recours à un algorithme d'optimisation (glouton, du sac à dos ou génétique, typiquement) pour construire une configuration quasi-optimale d'objets. Cependant, afin de prendre en compte les interactions entre index et vues matérialisées, il est nécessaire de modifier les algorithmes présentés à la Section 2.1. Dans une itération donnée, le coût d'un objet dépend en effet des objets sélectionnés lors des itérations précédentes. Il doit donc être recalculé à chaque étape. Par soucis de simplicité, nous avons mis en œuvre cette proposition au sein d'un algorithme glouton.

2.2.2 Principe général

Notre approche automatique d'optimisation des performances des entrepôts de données (Figure 2.2) se base non seulement sur des informations extraites des données entreposées (statistiques telles que la sélectivité des attributs, par exemple) ou de la charge appliquée à l'entrepôt, mais aussi sur des connaissances. Celles-ci relèvent des métadonnées classiquement associées aux entrepôts (nous exploitons notamment le schéma de la base), ainsi que d'une certaine expertise d'administration, formalisée par des modèles de coûts (bénéfice apporté par un index et coût de maintenance, par exemple) ou des règles. Notre approche procède en deux principales étapes, qui sont pilotées par ces connaissances.

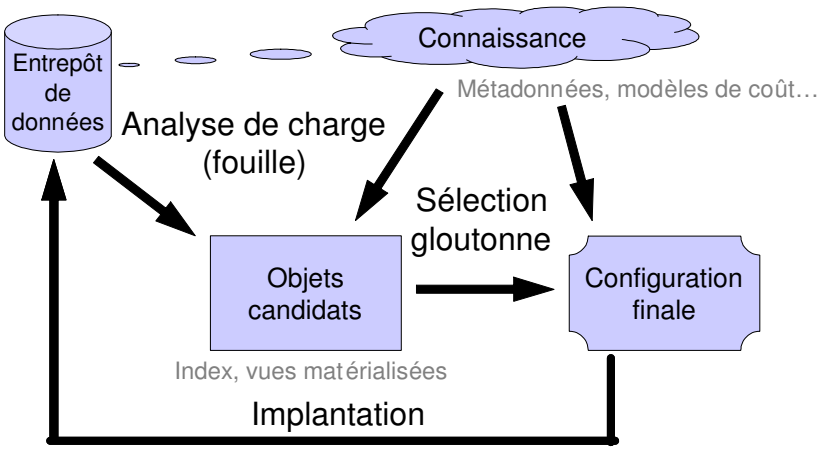

FIG. 2.2 – Principe de notre approche automatique d'optimisation des performances

La première étape est la constitution d'une configuration d'objets candidats. Elle consiste en l'analyse syntaxique d'une charge en entrée, qui fournit des attributs susceptibles d'être utiles à des stratégies d'indexation ou de matérialisation de vues. L'application de règles issues de connaissances liées à l'administration peut d'ores et déjà réduire la taille de cet ensemble d'attributs. Par exemple, un attribut à faible sélectivité comme le sexe d'une personne, qui ne peut prendre que deux valeurs, n'est pas intéressant à indexer. Cet ensemble d'attributs est ensuite structuré en tableau attributs-valeurs susceptible d'être traité par un algorithme de fouille de données qui fournit en sortie la configuration d'objets candidats.

Il serait impossible d'exploiter tous les objets candidats, l'espace disque qui leur est alloué étant en effet contraint. La deuxième étape de notre processus est donc une sélection gloutonne d'une configuration finale d'objets parmi les objets candidats. Cet algorithme exploite des modèles de coûts que nous avons développés pour exprimer, par exemple, le bénéfice apporté par un index ou une vue matérialisée et leurs coûts de stockage et de maintenance (Section 2.3). Enfin, la dernière étape de notre approche consiste à implanter la configuration d'objets finale dans l'entrepôt de données.

Notons que nous avons conçu cette démarche de façon modulaire, afin qu'elle soit la plus générique possible. La mise en œuvre des deux étapes principales nous a en effet amenés à faire des choix, mais d'autres options sont faciles à prendre en compte. Par exemple, la technique de fouille de données que nous avons sélectionnée pour la construction d'une configuration d'index candidats est l'extraction de motifs fréquents, mais une autre voie explorée à l'Université d'Oklahoma a tiré parti de la classification [ZSG04]. Nous avons d'ailleurs également employé la classification pour la sélection des vues matérialisées. De même, d'autres algorithmes d'optimisation peuvent être substitués à la stratégie gloutonne que nous avons adoptée pour constituer la configuration finale d'index. Nos modèles de coût peuvent aussi aisément être remplacés par d'autres si nécessaire, ou par des appels à un optimiseur de requêtes, si ce dernier est accessible sur le SGBD hôte.

2.2.3 Sélection des objets candidats

La charge de travail du système est typiquement accessible depuis le journal des transactions du SGBD hôte. Une charge donnée est supposée représentative si elle a été mesurée pendant une période de temps jugée suffisante par l'administrateur de l'entrepôt pour pouvoir anticiper les transactions à venir.

Comme nous nous intéressons plus particulièrement à la performance des requêtes décisionnelles et non à la maintenance de l'entrepôt, nous ne traitons que des extraits de charges de requêtes d'interrogation. Typiquement, ces requêtes sont composées d'opérations de jointure entre la table de faits et les dimensions, de prédicats de restriction et d'opérations d'agrégation et de regroupement. Plus formellement, une requête décisionnelle q peut s'exprimer comme suit en algèbre relationnelle :

$q = \Pi_{G,M}(\sigma_R(F \bowtie D_1 \bowtie D_2 \bowtie ... \bowtie D_d))$,

où G est l'ensemble des attributs issus des dimensions D_i présents dans la clause de regroupement de la requête, M est un ensemble de mesures agrégées de la table de faits F et R une conjonction de prédicats sur les attributs des dimensions.

Les attributs susceptibles d'être des supports d'index ou de vues matérialisées appartiennent aux ensembles G et R [CD97, VZZ+00, GRS02, FR03]. Nous les référençons dans une matrice binaire dite « requêtes-attributs » dont les lignes représentent les requêtes q_i de la charge et les colonnes les attributs représentatifs a_j. Le terme général m_{ij} de cette matrice est égal à un si l'attribut a_j est présent dans la requête q_i et à zéro sinon. Un exemple simplifié de matrice « requêtes-attributs » basé sur l'extrait de charge fournie à la Figure 2.3 est donné dans le Tableau 2.1.

```
q1  SELECT F.a1, SUM(F.a2) FROM F, D1
    WHERE F.a1 = D1.a3 AND D1.a4 < 2000
    GROUP BY F.a1
q2  SELECT F.a1, F.a5, AVG(F.a6) FROM F, D1, D2
    WHERE F.a1 = D1.a3 AND F.a5 = D2.a7 AND D2.a8 = 'ABC'
    GROUP BY F.a1, F.a5
q3  SELECT F.a1, F.a9, SUM(F.a2) FROM F, D1, D3
    WHERE F.a1 = D1.a3 AND F.a9 = D3.a10
    GROUP BY F.a1, F.a9
```

FIG. 2.3 – Exemple de charge

	a_1	a_3	a_4	a_5	a_7	a_8	a_9	a_{10}
q_1	1	1	1	0	0	0	0	0
q_2	1	1	0	1	1	1	0	0
q_3	1	1	0	0	0	0	1	1

TAB. 2.1 – Exemple de matrice requêtes-attributs

Cette structure de données est directement assimilable aux tableaux individus-variables exploités par les algorithmes de fouille, les individus étant ici les requêtes et les variables,

les attributs. L'application d'une technique de fouille de données sur la matrice « requêtes-attributs » permet d'obtenir un ensemble d'objets (index et vues matérialisés) candidats O_c.

2.2.4 Construction de la configuration d'objets finale

Notre algorithme de construction d'une configuration finale d'index et de vues matérialisées (Figure 2.4) se base sur une recherche gloutonne ascendante dans l'ensemble des objets candidats O_c fourni en entrée. Il part d'une configuration d'objets finale O vide, puis y ajoute à chaque itération l'objet o de O_c qui maximise la fonction objectif f_O. Pour tout objet $o \in O_c$, la valeur de $f_O(o)$ dépend des objets déjà sélectionnés dans O. Elle doit donc être recalculée à chaque itération. C'est ce qui permet de prendre en compte les interactions entre index et vues matérialisées. L'algorithme se termine quand la fonction objectif f_O ne peut plus être améliorée, qu'il n'y a plus d'objets candidats dans O_c ou que l'espace disque S alloué par l'administrateur aux index et aux vues matérialisées est saturé.

```
O = ∅
t_O = 0
Repeat
    o_max = ∅
    f_max = 0
    For each o ∈ O_c do
        If f_O(o) > f_max then
            o_max = o
            f_max = f_O(o)
        End if
    End for
    If f_O(o_max) > 0 then
        O = O ∪ {o_max}
        t_O = t_O + taille(o_max)
        O_c = O_c \ {o_max}
    End if
Until f_O(o_max) ≤ 0 or O_c = ∅ or t_O ≥ S
```

FIG. 2.4 – Algorithme de construction de la configuration finale d'objets

Pour une charge Q et une configuration d'objets O données, la fonction objectif f_O s'exprime de manière générale comme suit :

$f_O(o) = \alpha_o \; bénéfice_O(o) - \beta_o \; maintenance(o)$.

Généralement, $bénéfice_O(o) = \frac{coût(Q,O) - coût(Q,O \cup \{o\})}{taille(o)}$. Cependant, la prise en compte des interactions entre index et vues matérialisées complique le calcul de cette fonction (Section 2.3.3). Les modèles de coût développés dans la Section 2.3 permettent de valoriser les fonctions *coût* et *maintenance*.

Le coefficient α_o permet de pondérer le bénéfice. Il est généralement égal à un, mais peut aussi servir à privilégier les index qui évitent des opérations de jointure (Section 2.3.1).

Finalement, le coefficient $\beta_o = |Q|p(o)$ est un estimateur du nombre de mises à jour de

l'objet o. La probabilité $p(o)$ de mise à jour de l'objet o est égale à $\frac{1}{|O|}\frac{\%rafra\hat{i}chissement}{\%interrogation}$, où $\frac{\%rafra\hat{i}chissement}{\%interrogation}$ représente la proportion de mises à jour de l'entrepôt de données par rapport aux interrogations.

2.3 Applications

Cette section présente trois instanciations de notre approche d'optimisation automatique des performances des entrepôts de données : la sélection automatique d'index, la sélection automatique de vues matérialisées et la sélection conjointe des index et des vues matérialisées. Nous insistons particulièrement, pour chacune de ces applications, sur ses spécificités en termes de sélection des objets candidats (méthode de fouille employée, notamment) et de modèles de coût exploités dans la construction de la configuration finale d'objets.

2.3.1 Sélection automatique d'index

Dans cette application, nous travaillons sur l'optimisation de l'exécution des opérations de jointure d'une charge de requêtes décisionnelles. Pour cela, nous proposons une sélection d'index basée sur l'extraction des attributs fréquents de la charge qui sont susceptibles d'être des supports d'index.

Nous avons dans un premier temps travaillé sur des index classiques de type B-arbre [Aou02, ADG03a, ADG03b, ADG03c, ADG03d, ADB04]. Nous nous focalisons ici sur la sélection d'index *bitmap* de jointure [ADBB05a, ADBB05b, Aou05]. Ces structures de données [OG95] sont particulièrement adaptées au contexte des entrepôts de données. En effet, non seulement ils rendent efficaces les opérations logiques et de comptage sur les *bitmaps* (qui opèrent directement en mémoire vive), mais ils permettent aussi de précalculer les jointures au moment de la création des index. De plus, l'espace nécessaire au stockage des *bitmaps* est réduit, notamment quand la cardinalité des attributs indexés est relativement faible, ce qui est en général le cas dans les dimensions d'un entrepôt.

Sélection des index candidats

Au moment de la construction du contexte d'extraction (matrice requêtes-attributs) qui sera exploité par un algorithme de fouille de données pour sélectionner les index candidats, nous utilisons des connaissances relatives à l'administration des SGBD et à l'optimisation des performances à la manière de Feldman et Reouven [FR03]. Cette présélection d'attributs permet de réduire l'espace de recherche de l'algorithme de fouille et, mécaniquement, d'en améliorer le temps de réponse.

La connaissance est formalisée sous la forme de règles de type « si-alors » comme, par exemple : « si un prédicat est de la forme $attribut \neq valeur$, alors $attribut$ ne doit pas être sélectionné ». En effet, un tel prédicat n'exploiterait pas un index défini sur $attribut$, toutes ses valeurs étant balayées sauf $valeur$.

Nous nous basons pour la sélection finale des index candidats sur l'intuition que l'importance d'un attribut à indexer est fortement corrélée avec sa fréquence d'apparition dans la charge. La recherche de motifs fréquents [AS94] apparaît donc comme une solution naturelle pour faire ressortir ces attributs. De nombreux algorithmes d'extraction de motifs fréquents sont disponibles dans la littérature. Nous avons sélectionné Close [PBTL99], qui présente différents avantages dans notre contexte.

Tout d'abord, Close permet de traiter des charges volumineuses. Grâce à l'exploitation des opérateurs de fermeture de Galois, il réduit en effet le nombre d'accès au contexte d'extraction de connaissance lors de la recherche des motifs fréquents [Pas00]. Il se révèle également performant lorsque le contexte d'extraction est dense, ce qui est le cas dans notre contexte, où des suites de requêtes forment souvent un enchaînement logique. Finalement, les motifs fermés fréquents découverts par Close sont moins nombreux que l'ensemble des motifs fréquents (qui peuvent néanmoins être générés à partir des fermés). Cela permet à la fois de réduire les temps de calcul et d'éviter la prolifération d'index candidats inutiles.

Modèles de coût

Coût d'accès aux données à travers un index *bitmap* de jointure. L'accès aux données s'effectue en deux temps : parcours des *bitmaps* de l'index, puis lecture des n-uplets. Si l'accès aux *bitmaps* est direct et que les données sont uniformément distribuées (ce qui est une hypothèse raisonnable [CBC93]), le coût de parcours de l'index est $d\frac{|A||F|}{8S_p}$. d est le nombre de prédicats appliqués sur l'attribut indexé A. F est la table de faits. S_p est la taille d'une page disque. $\frac{|A||F|}{8}$ représente la taille de l'index *bitmap* [WB98].

Le nombre de n-uplets lus par une requête utilisant d *bitmaps* est $d\frac{|F|}{|A|}$ si les données sont uniformément distribuées. Le nombre d'entrées/sorties correspondant est alors égal à $p_F(1 - e^{-\frac{d|F|}{p_F|A|}})$ [OQ97], où p_F est le nombre de pages disque nécessaires pour stocker F. Finalement,

$coût = d\frac{|A||F|}{8S_p} + p_F(1 - e^{-\frac{d|F|}{p_F|A|}})$.

Si l'accès aux *bitmaps* de l'index s'effectue à travers un B-arbre, comme c'est le cas dans le SGBD Oracle, par exemple, il faut prendre en compte le coût de descente du B-arbre : $log_m|A| - 1$, où m est l'ordre du B-arbre. Le coût de parcours des nœuds feuilles du B-arbre est ensuite $\frac{|A|}{m-1}$ au pire. Cependant, celui de l'index bitmap se réduit à $d\frac{|F|}{8S_p}$. Alors,

$coût = log_m|A| - 1 + d\frac{|F|}{8S_p} + p_F(1 - e^{-\frac{d|F|}{p_F|A|}})$.

Coût de maintenance d'un index *bitmap* de jointure. Soit un index *bitmap* de jointure défini sur l'attribut A de la dimension D. Lors de l'insertion d'un n-uplet dans la table de faits F, D doit être parcourue pour trouver le n-uplet qui doit être joint avec celui qui est inséré dans F : p_D pages sont lues. Il faut ensuite mettre à jour les *bitmaps* de l'index. Au pire, ils sont tous parcourus et $\frac{|A||F|}{8S_p}$ pages sont lues. Donc,

$maintenance_F = p_D + \frac{|A||F|}{8S_p}$.

Lors de l'insertion d'un n-uplet dans une dimension D, la mise à jour peut être sans expansion du domaine, auquel cas un bit correspondant au n-uplet inséré est ajouté à chaque *bitmap* ; ou avec expansion du domaine, auquel cas un nouveau *bitmap* doit être créé. Alors,

$maintenance_D = p_F + (1 + \xi)\frac{|A||F|}{8S_p}$, où $\xi = 1$ en cas d'expansion du domaine et $\xi = 0$ sinon.

Bilan

L'originalité de notre approche provient principalement de l'utilisation de l'extraction de motifs fréquents pour sélectionner les index candidats les plus pertinents. Elle présente cependant un autre avantage. Les rares approches qui permettent de sélectionner des index multi-attributs exploitent un processus itératif pour les construire : index mono-attribut à la première itération, index sur deux attributs à la deuxième et ainsi de suite [CD97].

Dans notre approche, les motifs fréquents, qui sont des ensembles d'attributs de taille variable, permettent de proposer directement des index candidats multi-attributs. De plus, ces index candidats sont *a priori* pertinents, tandis que des combinaisons générées à partir d'index candidats de taille inférieure ne le sont pas toutes nécessairement. Notre approche évite donc un traitement et fournit un ensemble de candidats pertinents plus réduit.

Par ailleurs, la plupart des techniques de sélection d'index de la Section 2.1.2 n'exploitent que les index en B-arbre. Si ce type d'index est largement répandu dans les SGBD, il n'est pas le plus adapté pour indexer des données volumineuses et des attributs de faible cardinalité. Dans le contexte des entrepôts de données, les index *bitmaps* de jointure que nous avons privilégiés sont plus efficaces.

Finalement, les expériences que nous avons menées ont montré que les configurations d'index sélectionnées à l'aide de notre approche permettent un gain de performance d'environ 30 % en moyenne. Elles ont également indiqué que l'application de nos modèles de coût réduit nettement le nombre d'index sélectionnés, mais sans dégrader les performances de façon significative. Cela garantit un gain substantiel en terme d'espace de stockage (40 % en moyenne) et une diminution du coût de maintenance des index.

2.3.2 Sélection automatique de vues matérialisées

Dans cette application, nous proposons une stratégie de sélection de vues matérialisées exploitant la classification non supervisée des requêtes de la charge Q. En effet, plusieurs requêtes de syntaxe similaire ont une forte probabilité d'être résolues à partir d'une même vue matérialisée. Il s'agit alors de construire des classes de requêtes de Q similaires. Comme le nombre de classes est *a priori* inconnu, nous avons opté pour une méthode de classification non supervisée.

Le principe de cette approche est similaire à celui de la compression de charge de requêtes SQL [CGN02], une technique qui a été proposée dans le cadre des bases de données relationnelles pour optimiser, par exemple, la sélection d'index ou la réponse approximative

à des requêtes d'agrégation. Nous avons pour notre part mis cette idée en œuvre dans les contextes des entrepôts de données relationnels (avec une charge de requêtes décisionnelles exprimées en SQL [Aou05, AJD06]) et XML (avec une charge de requêtes décisionnelles exprimées en XQuery [Mah05, MAD06a]).

Sélection des vues matérialisées candidates

Similarité et dissimilarité entre requêtes. Afin d'opérer une classification et de nous assurer que les classes de requêtes sont homogènes, il nous faut définir des mesures de similarité et de dissimilarité entre requêtes. Soit M une matrice requêtes-attributs de terme général m_{ij} définie sur l'ensemble de requêtes $Q = \{q_i, i = 1..n\}$ et l'ensemble d'attributs $A = \{a_j, j = 1..l\}$. Nous définissons la similarité et la dissimilarité élémentaires entre deux requêtes q_i et $q_{i'}$ suivant l'attribut a_j comme suit.

$$\partial_{sim/a_j}(q_i, q_{i'}) = \begin{cases} 1 \text{ si } m_{ij} = m_{i'j} = 1 \\ 0 \text{ sinon} \end{cases}$$

$$\partial_{dissim/a_j}(q_i, q_{i'}) = \begin{cases} 1 \text{ si } m_{ij} \neq m_{i'j} \\ 0 \text{ sinon} \end{cases}$$

Notons que ces définitions ne sont pas symétriques. En effet, l'absence d'un attribut donné dans deux requêtes ne constitue pas un élément de similarité, au contraire de sa présence. Nous étendons maintenant ces définitions sur l'ensemble d'attributs A de manière à obtenir la similarité et la dissimilarité globales entre deux requêtes q_i et $q_{i'}$.

$$sim(q_i, q_{i'}) = \sum_{j=1}^{l} \partial_{sim/a_j}(q_i, q_{i'})$$

$$dissim(q_i, q_{i'}) = \sum_{j=1}^{l} \partial_{dissim/a_j}(q_i, q_{i'})$$

Classification des requêtes. L'objectif de la classification est de construire une partition naturelle des requêtes qui reflète leur structure interne. Pour cela, les objets de même classe doivent présenter une forte similarité, et les objets de classes différentes une forte dissimilarité. Soit une partition $P = \{C_k, k = 1..p\}$ constituée de p classes (ensembles de requêtes). Nous définissons la similarité inter-classes entre deux classes C_a et C_b distinctes de P, ainsi que la dissimilarité intra-classe au sein d'une classe C_a de P, comme suit.

$$Sim(C_a, C_b) = \sum_{q_i \in C_a, q_{i'} \in C_b,} sim(q_i, q_{i'})$$

$$Dissim(C_a) = \sum_{q_i \in C_a, q_{i'} \in C_a, i<i'} dissim(q_i, q_{i'})$$

Finalement, nous définissons sur P la mesure de qualité de classification $Q(P)$, qui permet de capturer l'aspect naturel de la partition. $Q(P)$ possède en effet des valeurs faibles pour des partitions présentant une forte homogénéité intra-classe et une forte disparité inter-classes. $Q(P)$ doit être minimisée.

$$Q(P) = \sum_{a=1..p, b=1..p, a<b} Sim(C_a, C_b) + \sum_{a=1}^{z} Dissim(C_a)$$

Pour réaliser la classification, nous avons choisi l'algorithme Kerouac [JN03] car il présente des caractéristiques intéressantes dans notre contexte. En effet, il permet non seulement de prendre facilement en compte notre mesure de qualité $Q(P)$, mais aussi d'intégrer des contraintes dans le processus de classification. Ainsi, il est possible de satisfaire une précondition du processus de fusion des vues matérialisées (voir paragraphe suivant) : les requêtes d'une classe donnée doivent présenter les mêmes conditions de jointure.

Fusion des vues candidates. Le résultat de la classification est un ensemble de classes de requêtes similaires. Notre objectif est d'associer à chaque classe le nombre le plus réduit possible de vues matérialisées permettant de couvrir toutes les requêtes de la classe. Pour cela, nous considérons chacune de ces requêtes comme une vue potentielle et opérons un processus de fusion pour en diminuer le nombre. Nous fusionnons les vues potentielles à l'aide d'un algorithme très similaire à celui proposé par Agrawal *et al.*, qui exploite une approche itérative par niveau [ACN01]. Cependant, dans notre contexte, il se révèle plus efficace car il est effectué sur un nombre limité de vues dans chaque classe et non sur l'ensemble des vues candidates dérivées de la charge. Le résultat de la fusion appliquée aux classes obtenues à l'étape précédente fournit les vues matérialisées candidates.

Modèle de coût

Dans la plupart des modèles de coût proposés dans la littérature pour les entrepôts de données, le coût d'une requête q est supposé proportionnel à la taille en n-uplets de la vue matérialisée exploitée par q [GR98]. Il en est de même pour le coût de maintenance de la vue. Nous réutilisons donc un modèle qui estime la taille d'une vue matérialisée donnée. Proposé par Golfarelli et Rizzi [GR98], il exploite la formule de Yao [Yao77] pour estimer le nombre de n-uplets $|V|$ d'une vue V composée de k attributs $a_1, a_2, ..., a_k$ et basée sur une table de faits F et d dimensions $D_1, D_2, ..., D_d$:

$$|V| = taillemax(V) \times \left(1 - \prod_{i=1}^{|F|} \frac{taillemax(F) \times (1 - \frac{1}{taillemax(V)}) - i + 1}{taillemax(F) - i + 1}\right)$$

où $taillemax(V) = \prod_{i=1}^{k} |a_i|$ et $taillemax(F) = \prod_{i=1}^{d} |D_i|$.

Lorsque le ratio $\frac{taillemax(F)}{taillemax(V)}$ est suffisamment élevé, la formule de Cardenas [Car75] permet d'obtenir une bonne approximation :

$$|V| = taillemax(V) \times \left(1 - \left(1 - \frac{1}{taillemax(V)}\right)^{|F|}\right).$$

La taille en octets de V est alors $taille(V) = |V| \times \sum_{i=1}^{n} taille(d_i)$, où $taille(d_i)$ est la taille en octets de la dimension d_i de V (qui peut être directement obtenue à partir des métadonnées de l'entrepôt) et n le nombre de dimensions de V.

Les formules de Yao et Cardenas se basent sur l'hypothèse que les données sont uniformément distribuées et tendent à surestimer la taille des vues. Elles ont cependant l'avantage d'être simples à implémenter et rapides à calculer. D'autres méthodes plus précises

exploitent l'échantillonnage de données et des lois statistiques [SDNR96, CM99, NT01], mais elles sont plus difficiles à mettre en œuvre.

Pour terminer, notons que ce modèle de coût est très facile à adapter dans le cas XML en mettant en équivalence relations et documents XML, d'une part, et n-uplets et éléments XML, d'autre part. La seule véritable différence réside dans le calcul de $taille(d_i)$, mais cette dernière est également obtenue à partir des métadonnées de l'entrepôt dans le cas XML.

Bilan

L'apport principal de notre approche de sélection de vues matérialisées se situe au niveau de la sélection des vues candidates. La plupart des méthodes antérieures construisent en effet un treillis ou un graphe de toutes les vues syntaxiquement correctes pour une charge donnée. Or, en pratique, ces structures de données sont complexes à construire et à parcourir. Notre appel à un algorithme de classification permet de réduire considérablement le nombre de vues matérialisées candidates en proposant quelques vues par classe (une seule dans le meilleur cas, grâce au processus de fusion décrit à la Section 2.3.2) et non plus une vue par requête de la charge. Cette réduction de dimensionnalité permet d'améliorer l'efficacité de la sélection d'une configuration finale de vues matérialisées et offre une possibilité réelle de passage à l'échelle. L'idée d'opérer une fusion des vues candidates avait déjà été avancée par Agrawal *et al.* [ACN00, ACN01], mais nous gagnons beaucoup en efficacité en ne l'appliquant qu'aux vues d'une classe donnée et non à l'ensemble complet des vues candidates.

Par ailleurs, les expériences que nous avons menées ont montré que les vues sélectionnées à l'aide de notre approche améliorent les performances de façon substantielle. À titre d'exemple, dans le contexte relationnel, si S_V est l'espace nécessaire pour stocker toutes les vues matérialisées candidates obtenues à l'issue de la classification, le gain de performance est de 68,9 % en moyenne lorsque la contrainte d'espace de stockage est fixée à 35,4 % de S_V et de 94,9 % en moyenne lorsqu'elle est fixée à 100 %. De plus, nous avons montré la pertinence des vues matérialisées sélectionnées par notre approche en calculant le taux de couverture des requêtes par ces vues, c'est-à-dire la proportion de requêtes résolues en utilisant les vues. Lorsque la contrainte d'espace de stockage est très forte (0,05 % de S_V), le taux de couverture est déjà de 23 % en moyenne. Il atteint 100 % lorsque la contrainte d'espace est relachée. Finalement, nos premières expériences dans le contexte XML ont également donné des résultats très encourageants. L'exploitation des vues XML proposées par notre stratégie en lieu et place de l'entrepôt XML original a permis de rendre l'exécution d'une charge décisionnelle exprimée en XQuery 24 700 fois plus rapide en moyenne.

2.3.3 Sélection conjointe des index et des vues matérialisées

Cette dernière application a pour objectif de réellement prendre en compte les interactions entre index et vues matérialisées et d'optimiser le partage de l'espace de stockage al-

loué à ces structures de données. Elle s'appuie sur la structure modulaire de notre approche et prend en entrée un ensemble d'objets (index et vues matérialisées) candidats obtenus avec un algorithme de sélection quelconque, comme ceux que nous avons proposés aux Sections 2.3.1 et 2.3.2. Elle met ensuite en œuvre des structures de données et des modèles de coût spécifiques qui permettent d'atteindre nos objectifs [Mai05, Aou05, MAD06c].

Sélection des objets candidats

Objets en entrée. L'application isolée d'algorithmes de sélection d'index et de vues matérialisées fournit dans un premier temps un ensemble de vues candidates et un ensemble d'index candidats définis sur l'entrepôt de données étudié. Mais, une vue matérialisée étant physiquement stockée sur disque sous la forme d'une table, il est également possible de l'indexer pour optimiser plus avant les temps d'accès aux données. Nous considérons donc un troisième ensemble d'objets en entrée de notre algorithme de sélection simultanée : les index candidats définis sur les vues matérialisées candidates.

Structure de données spécifique. Afin de modéliser les relations entre les vues matérialisées candidates et les index candidats, nous introduisons une nouvelle matrice VI dite « vues-index » qui identifie les index candidats construits sur des vues matérialisées candidates. Les lignes de la matrice VI représentent les vues matérialisées candidates et les colonnes les index candidats sur ces vues. Le terme général VI_{vi} de cette matrice est égal à un si la vue v exploite l'index i et à zéro sinon.

Modèles de coût

Nous avons déjà présenté dans les Sections 2.3.1 et 2.3.2 les modèles de coût relatifs aux index *bitmap* de jointure et aux vues matérialisées, respectivement. Les index portant sur les vues matérialisées étant en général de type B-arbres ou leurs dérivés, nous rappelons ici les modèles de coût relatifs à ces index.

Coût d'accès aux données à travers un index en B-arbre. L'accès aux données à travers un index se décompose en deux parties : le parcours de l'index pour trouver les valeurs de clés correspondant à la requête (coût $C_{parcours}$), puis la recherche de ces identifiants dans la base de données (coût $C_{recherche}$). Soient q une requête, ζ un ensemble d'index, SNA_q l'ensemble des attributs présents dans la clause de restriction (`Where` en SQL) de la requête q, BF_a le facteur de bloc de l'index construit sur un attribut a (le nombre moyen de couples $(clé, identifiant)$ par page disque), SF_a le facteur de sélectivité de l'attribut a et v la vue matérialisée accédée. Alors :

$$C_{parcours} = \sum_{a \in (\zeta \cap SNA_q)} \lceil log_{BF_a} |v| \rceil + \left\lceil \frac{SF_a |v|}{BF_a} \right\rceil - 1.$$

Le nombre d'identifiants à rechercher ensuite est $N = |v| \prod_{a \in (\zeta \cap SNA_q)} SF_a$, d'où on peut déduire grâce à la formule de Cardenas [Car75] le nombre de pages disques à accéder :

$C_{recherche} = S_p \left(1 - (1 - \frac{1}{S_p})^N\right)$, où S_p est la taille d'une page disque.

Finalement, le coût d'accès aux données à travers un index en B-arbre est $coût = C_{parcours} + C_{recherche}$.

Coût de maintenance d'un index en B-arbre. Classiquement, ce coût s'exprime comme suit :

$$maintenance = \sum_{op \in \{aj, sup, mod\}} \sum f_{op} \sum_{a \in A_{op}} C_{op}(a)\,;$$

où f_{aj}, f_{sup} et f_{mod} sont respectivement les fréquences d'ajout, de suppression et de modification et $C_{aj}(a)$, $C_{sup}(a)$ et $C_{mod}(a)$ sont respectivement les coûts de maintenance liés à une opération d'ajout, de suppression et de modification de l'attribut a. A_{op} est l'ensemble d'attributs considéré. $A_{aj} = A_{sup} = \zeta$, où ζ est l'ensemble d'index à maintenir. $A_{mod} = \zeta \cap SNA_{mod}$, où SNA_{mod} est l'ensemble des attributs à mettre à jour. Finalement, les coûts de maintenance s'expriment de la façon suivante [Wha85] :

$C_{aj}(a) = C_{sup}(a) = \lceil log_{BF_a}|v| \rceil$ et $C_{mod}(a) = \lceil log_{BF_a}|v| \rceil + \left\lceil \frac{|v|SF_a}{2BF_a} \right\rceil - 1$.

Bénéfice de matérialisation et d'indexation. Dans le cas général (Section 2.2.4), le bénéfice apporté par la sélection d'un objet o est défini comme la différence entre le coût d'exécution des requêtes de la charge Q avant et après insertion de o dans la configuration finale d'objets O. La prise en compte des relations entre index et vues matérialisées implique une redéfinition de la fonction de calcul du bénéfice. Soient $i \in O_C$ et $v \in O_C$ un index et une vue candidats, respectivement. L'ajout de i ou v à O peuvent conduire aux cas de bénéfices énumérés dans les Tableaux 2.2 et 2.3, respectivement, selon les interactions entre i et v.

	$VI_{vi} = 1$	$VI_{vi} = 0$
$v \in O$	min(matérialisation, indexation v)	indexation
$v \notin O$	—	indexation

Tab. 2.2 – Bénéfice apporté par un index i

	$VI_{vi} = 1$	$VI_{vi} = 0$
$i \in O$	—	matérialisation
$i \notin O$	min(indexation, matérialisation v)	matérialisation

Tab. 2.3 – Bénéfice apporté par une vue v

Les bénéfices d'indexation et de matérialisation pour Q apportés par l'ajout d'un index i ou d'une vue v à O, respectivement, s'expriment finalement comme suit.

$$bénéfice_O(i) = \begin{cases} \frac{coût(Q,O) - coût(Q, O \cup \{i\})}{taille(i)} \text{ si } VI_{vi} = 0 \;\forall\; v \in V \;(V \subseteq O) \\ \frac{coût(Q,O) - coût(Q, O \cup \{i\} \cup V')}{taille(i) + \sum v' \in V' taille(v')} \text{ si } V' = \{v \in V, VI_{vi} = 1\} \neq \emptyset \\ 0 \text{ sinon} \end{cases}$$

$$bénéfice_O(v) = \begin{cases} \frac{coût(Q,O) - coût(Q,O \cup \{v\})}{taille(v)} \text{ si } VI_{vi} = 0 \ \forall \ i \in I \ (I \subseteq O) \\ \frac{coût(Q,O) - coût(Q,O \cup \{v\} \cup I')}{taille(v) + \sum i' \in I' taille(i')} \text{ si } I' = \{i \in I, VI_{vi} = 1\} \neq \emptyset \\ 0 \text{ sinon} \end{cases}$$

Bilan

L'aboutissement de notre approche de sélection simultanée d'index et de vues matérialisées permet de prendre réellement en compte les interactions entre ces structures de données. En effet, les approches existantes (Section 2.1.4) les considèrent généralement comme des objets disjoints, dont le bénéfice et le coût de maintenance sont fixes et indépendants des objets déjà sélectionnés (en ce qui concerne le bénéfice). Par ailleurs, peu d'entre elles considèrent l'indexation des vues matérialisées sélectionnées.

L'approche de Bellatreche *et al.*, qui se rapproche le plus de la nôtre, part d'une solution initiale composée d'index et de vues sélectionnés d'une manière isolée sous contrainte d'espace de stockage. Or, prendre cette contrainte en compte *a priori* peut éliminer des solutions qui peuvent devenir pertinentes dans les itérations suivantes du processus de sélection. C'est pourquoi nous n'introduisons la contrainte d'espace de stockage qu'*a posteriori*, au sein de l'algorithme de sélection. Par ailleurs, les politiques de remplacement d'objets dans les espaces disque réservés aux index et aux vues exploitent la fréquence d'utilisation de ces objets dans les requêtes, et non le bénéfice apporté par leur utilisation conjointe.

Pour terminer, nos résultats expérimentaux ont montré que la sélection simultanée des index et des vues matérialisées est plus efficace que leur sélection isolée, notamment lorsque l'espace disque alloué à ces structures de données est suffisamment grand. En revanche, nos expériences ont également confirmé l'intuition que si cet espace est plus réduit, il est préférable de privilégier les index, dont la taille est en général significativement moins importante que celle des vues matérialisées.

2.4 Conclusion

Nous avons présenté dans ce chapitre une approche tendant à l'optimisation automatique des performances des entrepôts de données. Notre contribution dans ce domaine porte principalement sur l'exploitation de connaissances concernant l'entrepôt et son utilisation, qui sont soit la formalisation d'une expertise, soit automatiquement extraites à l'aide de techniques de fouille de données. Cette approche nous a permis de réduire la dimensionnalité du problème de sélection d'index et de vues matérialisées en établissant une configuration d'objets candidats réduite et pertinente. Nous avons également pris en compte explicitement les interactions entre index et vues matérialisées, de manière à proposer une configuration d'objet finale qui soit la plus proche possible de l'optimal.

Nous avons par ailleurs conçu notre approche de manière générique. Les techniques de fouille que nous exploitons et les modèles de coût que nous proposons ne sont en

effet liés à aucun système en particulier. Ils peuvent être appliqués sur n'importe quel SGBD hôte. Nos stratégies de sélection d'index et de vues matérialisées sont également modulaires : chaque étape (sélection des objets candidats, calculs de coûts...) exploite des outils interchangeables. Les techniques de fouille que nous préconisons et les modèles de coût que nous proposons peuvent par exemple être aisément remplacés par d'autres.

Nous avons aussi systématiquement cherché à démontrer l'efficacité de nos propositions en mettant en œuvre un processus d'expérimentation qui a impliqué leur implémentation sur des systèmes existants (Oracle, SQL Server, MySQL, eXist, X-Hive). Cependant, nous n'avons jusqu'à présent pas été en mesure de comparer *in situ* nos propositions aux approches existantes. En effet, celles qui sont développées par les grands éditeurs de SGBD [ACK+04, DDD+04, ZRL+04] nécessitent l'acquisition du système correspondant. Les algorithmes disponibles dans ces assistants sont également implantés sous forme de « boites noires » parfois difficiles à maîtriser. Par ailleurs, les propositions de recherche décrites dans la littérature ne sont pas toujours disponibles sous forme de code source ou exécutable (et dans ce cas, elles fonctionnent sur un système précis). Il faut souvent les réimplémenter soi-même. Il nous faudra cependant surmonter ces difficultés pour achever la validation de nos solutions.

Pour terminer, la principale évolution possible pour notre travail réside dans l'amélioration de l'automaticité de nos solutions. Ces dernières sont en effet des cas d'optimisation statique. Si la charge de requêtes en entrée évolue de façon significative, il faut les relancer intégralement pour maintenir le niveau de performance. Or, les approches dynamiques de sélection de vues matérialisées proposées pour optimiser le temps de rafraîchissement [KR99, SRR06] s'avèrent plus performantes que les approches statiques. Il nous faut donc travailler dans ce sens dans notre contexte, où nous cherchons à optimiser le temps de réponse des requêtes.

Pour cela, notre piste principale est d'exploiter le caractère modulaire de nos solutions en remplaçant les techniques de fouille de données que nous avons utilisées par des algorithmes incrémentaux de recherche de motifs fréquents [VMGM02, OPW+04, LKH05] ou de classification [JMF99]. Les travaux traitant de la détection de sessions basés sur des calculs d'entropie [YHA05] pourraient également être exploités pour détecter les moments où il est nécessaire de relancer une sélection (incrémentale) d'index et de vues matérialisées.

Chapitre 3

Évaluation des performances des bases et entrepôts de données

L'objectif de ce chapitre est, dans un premier temps, de présenter et de discuter l'existant en matière de bancs d'essais pour SGBD, ainsi que les grands principes qui sous-tendent ces outils. Nous motivons et détaillons ensuite trois de nos contributions dans ce domaine :

- DEF (*Dynamic Evaluation Framework*), une plateforme générique pour l'évaluation du comportement dynamique des applications de bases de données (objets et relationelles-objets, notamment) ;
- DWEB (*Data Warehouse Engineering Benchmark*), un des seuls bancs d'essais pour entrepôts de données actuellement opérationnels ;
- XDW-Bench (*XML Data Warehouse Benchmark*), le premier banc d'essais pour entrepôts de données XML.

3.1 Bancs d'essais pour bases de données

3.1.1 Définitions et généralités

Mettre en œuvre un banc d'essais consiste à effectuer une série de tests sur un SGBD existant dans le but d'estimer ses performances dans des conditions données. Les bancs d'essais sont généralement utilisés pour comparer les performances globales des SGBD, mais aussi pour illustrer les avantages d'un système ou d'un autre dans une situation donnée ou pour déterminer une configuration matérielle ou logicielle (nombre de disques, taille du cache...) pour un SGBD et/ou une application donnés. Ils peuvent donc être utilisés à la fois par les utilisateurs et les concepteurs de systèmes ou de bases de données, pour atteindre dans ces deux cas des objectifs différents.

Typiquement, un banc d'essais est composé de deux éléments principaux :

- une base de données (schéma conceptuel et méthode de génération ou extension de la base) ;

– une charge, c'est-à-dire un ensemble de requêtes (d'interrogation et de mise à jour) à lancer sur la base de données, ainsi qu'un protocole détaillant le déroulement de leur exécution.

Certains bancs d'essais spécifient en outre un ensemble de mesures simples ou composites qui permettent d'exprimer les performances du système. Parmi les mesures les plus simples et les plus couramment utilisées, nous pouvons citer à titre d'exemples le temps de réponse, le débit, le nombre d'objets accédés, le nombre d'entrées/sorties effectuées, l'espace mémoire ou disque occupé, etc.

Jim Gray définit quatre critères primordiaux permettant de spécifier un « bon » banc d'essais [Gra93] :

– la *pertinence* : le banc d'essais doit répondre aux besoins du plus grand nombre d'utilisateurs possible ;
– la *portabilité* : le banc d'essais doit être facile à implémenter sur différents systèmes ;
– le *passage à l'échelle* : il doit être possible d'étudier des bases de données de tailles diverses et d'augmenter l'échelle du banc d'essais ;
– la *simplicité* : le banc d'essais doit être compréhensible sous peine de demeurer inutilisé.

Notons que la pertinence et la simplicité forment des objectifs orthogonaux. En effet, augmenter la pertinence d'un banc d'essais pour différents utilisateurs se traduit souvent par une démarche tendant à la généricité et à une grande paramétrabilité. Cela peut rendre le banc d'essais difficile à appréhender et à utiliser.

À l'inverse, ne miser que sur la simplicité conduit à concevoir des bancs d'essais très ciblés et difficilement réutilisables dans d'autres contextes que celui dans lequel ils ont été créés. Par exemple, un banc d'essais dont la charge ne serait constitué que de requêtes d'interrogation serait mal adapté à l'évaluation de techniques d'indexation, pour lesquelles il faut non seulement mesurer les gains en temps de réponse, mais aussi la surcharge engendrée par la maintenance des index lors d'opérations de mise à jour. Étendre la charge devient alors indispensable pour assurer la pertinence du banc d'essais.

En conclusion, la conception d'un banc d'essais implique d'optimiser les critères énoncés par Jim Gray et notamment de trouver le meilleur compromis entre pertinence et simplicité. Dans les Sections 3.2, 3.3 et 3.4 de ce chapitre, nous nous attachons à proposer des solutions visant à augmenter la pertinence des bancs d'essais, tout en garantissant la plus grande simplicité possible.

Nous présentons dans les sections suivantes un panorama des bancs d'essais utilisés pour l'évaluation des performances de différents types de SGBD : relationnels, objets/relationnels-objets, XML et entrepôts de données [Dar08].

3.1.2 Bancs d'essais relationnels

Dans le monde des SGBD relationnels, le *Transaction Processing Performance Council* (TPC) joue un rôle prépondérant en matière de bancs d'essais. Cet institut à but non

lucratif fondé en 1988 a pour mission de définir des bancs d'essais standards, de vérifier leur application correcte par les usagers qui souhaitent voir leur produit évalué et de publier régulièrement les résultats de ces tests de performance. Au cours de son histoire, le TPC a spécifié huit bancs d'essais, dont quatre sont actuellement en usage. Ils partagent tous les variantes d'une base de données commerciale classique du type *client-commande-produit-fournisseur* et sont uniquement paramétrés par un facteur d'échelle qui détermine la taille de la base de données (de 1 à 100 000 Go, par exemple). Ces bancs d'essais sont :

- TPC-C, pour les bases de données transactionnelles [TPC05b],
- TPC-H, pour les applications décisionnelles (Section 3.1.5) [TPC05d],
- TPC-W, pour le commerce électronique [TPC02],
- TPC-App, pour les services Web [TPC05a].

En usage depuis 1992, TPC-C est spécifiquement dédié aux applications OLTP. Plus complexe que son prédécesseur TPC-A, il propose une base de données élaborée (neuf types de tables de taille et de structure variées), ainsi qu'une charge de transactions de complexités différentes, exécutées de manière concurrente. La mesure de performance adoptée par TPC-C est le débit en termes de transactions.

Il existe actuellement peu d'alternatives crédibles à TPC-C. Nous pouvons cependant citer OSDB (*The Open Source Database Benchmark*), fruit d'un projet démarré en 2001 dans la communauté du logiciel libre [Sou05]. OSDB étend et clarifie les spécifications d'un banc d'essais plus ancien : AS^{3}AP [TOB93]. La particularité d'OSDB est qu'il est disponible sous forme de code source libre en langage C, ce qui élimine toute ambiguïté relative à l'utilisation du langage naturel dans les spécifications. En revanche, OSDB demeure un projet en cours et la documentation du banc d'essais est extrêmement sommaire. La base de données d'AS^{3}AP est simple. Elle est composée de quatre relations dont la taille peut varier de 1 Mo à 100 Go. La charge est constituée de requêtes variées exécutées de manière concurrente. Les mesures de performance adoptées par OSDB sont le temps de réponse et le débit.

3.1.3 Bancs d'essais objets et relationnels-objets

Il n'existe pas de banc d'essais standard pour les SGBD orientés objets [Dar05]. Cependant, les plus souvent cités et utilisés, OO1 [Cat91], HyperModel [ABM$^+$90] et surtout OO7 [CDN93] font office de standards de fait. Tous trois sont principalement ciblés sur les applications d'ingénierie (Conception Assistée par Ordinateur, Ateliers de Génie Logiciel, etc.). Leur spectre s'étend d'OO1, qui présente un schéma très simple (deux classes) et seulement trois opérations, à OO7, qui est plus générique et propose un schéma complexe et paramétrable (dix classes), ainsi qu'une gamme d'opérations étendue (quinze opérations complexes). Cependant, même le plus abouti de ces bancs d'essais, OO7, n'est pas suffisamment générique pour modéliser d'autres types d'applications que des applications d'ingénierie, comme par exemple des applications financières, multimédia ou de télécommunication [TNL95]. De plus, sa complexité le rend difficile à implémenter. Pour contourner

ces limitations, le banc d'essais OCB a été proposé [DS00]. Totalement paramétrable, cet outil a pour ambition d'être réellement générique, sans pour autant trop sacrifier à l'objectif de simplicité de Gray. OCB est notamment capable de simuler le fonctionnement des bancs d'essais orientés objets antérieurs.

Par ailleurs, des ensembles de charges ont été proposés pour mesurer les performances des SGBD orientés objets [CFLS91, FCL93]. Ce ne sont pas des bancs d'essais à part entière, mais ils présentent des caractéristiques intéressantes. Ces charges synthétiques opèrent par exemple à la granularité de la page au lieu de celle de l'objet. Elles intègrent également la notion de régions « chaudes » et « froides » dans la base de données, c'est-à-dire des ensembles d'objets plus ou moins fréquemment utilisés que la moyenne, afin de modéliser plus fidèlement le comportement d'applications réelles.

Finalement, les bancs d'essais relationnels-objets, tels que BUCKY [CDN+97] et BORD [LKK00], sont orientés requêtes et uniquement dédiés aux systèmes relationnels-objets. Par exemple, BUCKY ne propose que des opérations spécifiques de ces systèmes, considérant que la navigation typique parmi des objets est déjà traitée par les bancs d'essais objets. Ces bancs d'essais se concentrent donc sur des requêtes impliquant des identifiants d'objets, l'héritage, les jointures, les références de classes et d'objets, les attributs multivalués, l'« applatissement » des requêtes, les méthodes d'objets et les types de données abstraits. Les schémas des bases de données de ces bancs d'essais sont figés.

3.1.4 Bancs d'essais XML

En l'absence de modèle standard, les solutions de stockage de documents XML développées depuis la fin des années 90 présentent des différences significatives, tant au niveau conceptuel qu'au niveau des fonctionnalités. Le besoin de comparer ces solutions, notamment en terme de performance, a amené à la conception de plusieurs bancs d'essais aux objectifs variés [LYW+05].

X-Mach1 [BR01], XMark [SWK+02], XOO7 (une extension XML d'OO7) [BLL+02] et XBench [YOK04] sont des bancs d'essais dits d'application dont l'objectif est d'évaluer les performances globales d'un SGBD natif ou compatible XML, et plus particulièrement du processeur de requêtes. Chacun met en œuvre une base de données XML mixte, à la fois orientée données (données structurées) et documents (en général des textes générés aléatoirement à partir d'un dictionnaire). Cependant, hormis pour XBench qui propose une base réellement mixte, leur orientation est plus particulièrement marquée vers les données (XMark, XOO7) ou les documents (X-Mach1). Ces bancs d'essais diffèrent également par :

- la nature fixe ou flexible du schéma XML (une ou plusieurs DTD[1] ou schémas XML) ;
- le nombre de documents XML utilisés pour modéliser la base de données au niveau physique (un ou plusieurs) ;
- la prise en compte ou non d'opérations de mise à jour dans leur charge.

[1] Définition de Type de Document ou *Document Type Definition*

Notons également que seul XBench permet d'évaluer toutes les fonctionnalités offertes par le langage XQuery.

Par ailleurs, des micro-bancs d'essais (le *Michigan Benchmark* [RPJAK02], ainsi nommé en référence au banc d'essais relationnel *Wisconsin Benchmark* [BDT83], et MemBeR [AMM05]) ont été proposé pour évaluer les performances individuelles d'opérations basiques comme les projections, restrictions, jointures et agrégations, plutôt que l'exécution de requêtes plus complexes. Ces bancs d'essais s'adressent particulièrement aux développeurs de solutions de stockage de documents XML, qui peuvent isoler des points critiques à améliorer, qu'aux utilisateurs soucieux de comparer différents systèmes. MemBeR propose de plus une méthodologie de construction de micro-bases de données, ce qui permet à l'utilisateur d'ajouter des jeux de données et de requêtes spécifiques à une tâche d'évaluation de performance donnée.

Finalement, des générateurs de données XML synthétiques ont également été proposés [DL99, ANZ01, BMKL02]. Ils ne proposent pas de charge et ne peuvent donc être qualifiés de bancs d'essais, mais présentent des caractéristiques intéressantes, notamment des paramètres de génération des données et des modèles de bases XML, qui permettent par exemple de reproduire les données de XMark et de TPC-H.

3.1.5 Bancs d'essais décisionnels

À notre connaissance, il existe très peu de bancs d'essais décisionnels en dehors de ceux du TPC et leurs spécifications sont rarement publiées dans leur intégralité [Dem95]. Cependant, parmi ceux-ci, le plus connu est APB-1, qui a été publié en 1998 par l'*OLAP council*, une association fondée par quatre éditeurs de solutions OLAP, mais désormais inactive. APB-1 a été assez intensivement utilisé à la fin des années 90. Son schéma d'entrepôt est structuré autour de quatre dimensions : Client, Produit, Canal et Temps. Sa charge de dix requêtes est destinée à la prédiction de ventes. APB-1 est assez simple et s'est révélé limité pour évaluer les spécificités d'activités et de fonctions différentes [Tho98]. Finalement, des jeux de données OLAP sont également disponibles en ligne [LRA00], mais ce ne sont que des bases de données et non des bancs d'essais complets (aucune charge n'est proposée, notamment).

Dans la suite de cette section, nous nous focalisons donc plus particulièrement sur les bancs d'essais décisionnels du TPC. TPC-D [Bal93, Bha96, TPC98] a fait son apparition dans le milieu des années 90 et forme la base de TPC-H et TPC-R [PF00, TPC03, TPC05d], qui l'ont remplacé depuis. TPC-H et TPC-R sont en fait identiques, seul leur mode d'utilisation les différencie. TPC-H est conçu pour le requêtage *ad-hoc* (les requêtes ne sont pas connues à l'avance et toute optimisation préalable est interdite), tandis que TPC-R a une vocation de *reporting* (les requêtes sont connues à l'avance et des optimisations adéquates sont possibles). Seul TPC-H est encore supporté par le TPC. TPC-H et TPC-R exploitent le même schéma de base de données que TPC-D : un modèle *clients-commandes-produits-fournisseurs* classique (représenté par un diagramme de classes UML dans la Figure 3.1) ;

ainsi que la charge de TPC-D enrichie de cinq nouvelles requêtes.

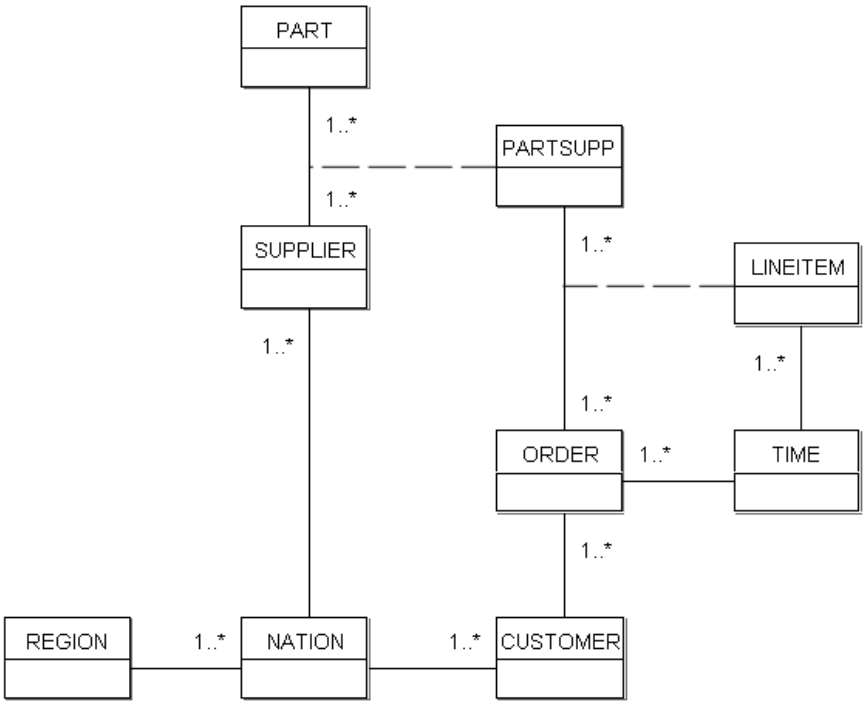

FIG. 3.1 – Schéma de la base de données de TPC-D, TPC-H et TPC-R

Plus précisément, cette charge est constituée de vingt-deux requêtes décisionnelles paramétrées écrites en SQL-92, numérotées de Q1 à Q22 et de deux fonctions de rafraîchissement RF1 et RF2 qui ajoutent et suppriment des n-uplets dans les tables ORDER et LINEITEM. Les paramètres des requêtes sont instanciés aléatoirement en suivant une loi uniforme. Finalement, le protocole d'exécution de TPC-H ou TPC-R est le suivant :

1. un test de chargement ;
2. un test de performance (exécuté deux fois), lui même subdivisé en un test de puissance et un test de débit.

Trois mesures principales permettent de décrire les résultats obtenus en termes de puissance, de débit et d'une composition de ces deux critères. La puissance et le débit sont respectivement la moyenne géométrique et arithmétique de la taille de la base de données divisée par le temps d'exécution de la charge.

TPC-DS, qui est actuellement en cours de développement [PSKL02, TPC05c], modélise plus clairement un entrepôt de données. Il est le successeur annoncé de TPC-H et TPC-R. Le schéma de la base de données de TPC-DS, dont les tables de faits sont représentées dans la Figure 3.2, décrit les fonctions décisionnelles d'un détaillant sous la forme de plusieurs schémas en flocon de neige. Les ventes par catalogue et par le Web sont interconnectées tandis que la gestion des stocks est indépendante. Ce modèle comprend également quinze dimensions partagées par les tables de faits. Il s'agit donc d'un schéma en constellation.

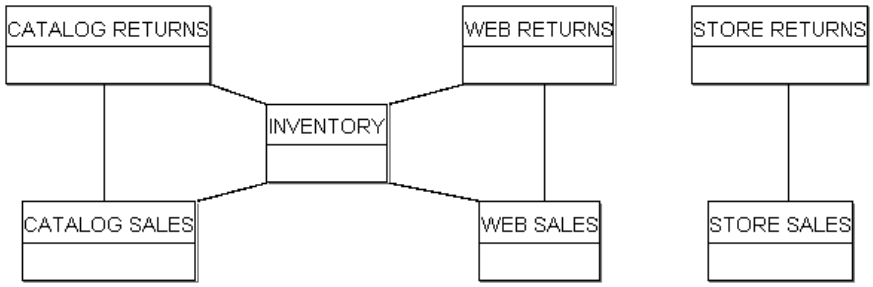

FIG. 3.2 – Schéma de l'entrepôt de données de TPC-DS (tables de faits)

La charge de TPC-DS est constituée de quatre classes de requêtes : requêtes de *reporting*, requêtes décisionnelles *ad-hoc*, requêtes interactives d'analyse en ligne (OLAP), requêtes d'extraction. Des modèles de requêtes écrits en SQL-99 (et comprenant donc des extensions OLAP) permettent de générer un ensemble d'environ cinq cents requêtes. Ces modèles sont instanciés aléatoirement selon des distributions non-uniformes. Le processus de maintenance de l'entrepôt de données comprend une phase d'ETL (*Extract, Transform, Load*) complète et un traitement spécifique des dimensions. Par exemple, les dimensions historisées conservent les anciens n-uplets quand de nouveaux sont ajoutés, tandis que les dimensions non-historisées ne conservent pas les anciennes données. Finalement, le modèle d'exécution de TPC-DS est divisé en quatre étapes :

1. un test de chargement,
2. une exécution des requêtes,
3. une phase de maintenance des données,
4. une seconde exécution des requêtes.

Une seule mesure (de débit) est proposée. Elle prend en compte l'exécution des requêtes et la phase de maintenance.

Comme pour tous les bancs d'essais du TPC, le passsage a l'échelle dans TPC-H et TPC-DS est réalisé grâce à un paramètre (*Scale Factor – SF*) qui permet de définir leur base de données en terme de taille (de 1 à 100 000 Go). Leur schéma demeure fixe. La charge est également invariante dans TPC-H. En revanche, le nombre de requêtes générées dépend de SF dans TPC-DS.

3.1.6 Discussion

La grande majorité des bancs d'essais existants vise à comparer les performances de différents systèmes dans des conditions expérimentales données. Cela permet aux éditeurs de SGBD d'étalonner leurs produits par rapport à la concurrence et aux utilisateurs de procéder à des choix de logiciels souvent stratégiques et coûteux en toute connaissance de cause. Ces bancs d'essais présentent invariablement un schéma de base de données et une charge fixes. L'objectif de passage à l'échelle de Gray est atteint grâce à un nombre réduits

de paramètres, qui permettent principalement de faire varier la taille de la base de données dans des proportions prédéterminées. C'est notamment le cas du paramètre d'échelle SF unique employé dans tous les bancs d'essais du TPC.

Cette solution a l'avantage de la simplicité (toujours au sens des critères définis par Jim Gray), mais la pertinence d'un tel banc d'essais est inévitablement réduite au cas explicitement modélisé par ses concepteurs. Par exemple, l'application type *client-commande-produit-fournisseur* adoptée par le TPC se révèle souvent inadaptée dans les domaines d'application autres que la gestion. Cela conduit souvent les utilisateurs de bancs d'essais à concevoir des variantes plus ou moins élaborées d'outils standards quand ils estiment ces derniers insuffisamment génériques pour satisfaire des besoins particuliers.

Or, ces utilisateurs ne sont généralement pas confrontés à un choix de logiciels. Ce sont plutôt des concepteurs et leurs besoins sont sensiblement différents. Ils cherchent en effet principalement à évaluer l'impact de choix architecturaux ou de techniques d'optimisation de performance au sein d'un système donné ou d'une famille de systèmes. Dans ce contexte, multiplier les jeux d'essais et les cas d'utilisation différents est indispensable et un banc d'essais monolithique a une pertinence réduite.

Même si les outils que nous développons peuvent être employés pour des comparaisons de performance classiques, nous nous situons essentiellement dans ce cadre. Notre objectif est en effet d'augmenter la pertinence des bancs d'essais pour les concepteurs de systèmes. Pour cela, nous proposons d'étendre le critère de passage à l'échelle de Gray à celui d'*adaptabilité*. Il s'agit alors pour un outil d'évaluation de performance de proposer différentes configurations de bases de données ou de charges qui permettent de mener des tests dans des conditions variées. Un tel outil peut alors être envisagé comme un générateur de bancs d'essais. Possiblement par abus de langage, nous les qualifions cependant de bancs d'essais paramétrables ou génériques.

Par ailleurs, tendre vers une bonne adaptabilité se fait mécaniquement au détriment de la simplicité du banc d'essais. Cependant, ce critère est très important et ne doit pas être négligé dans la conception d'un outil générique. Il est donc nécessaire de trouver des moyens d'assurer une bonne adaptabilité sans trop sacrifier la simplicité ; en résumé, de trouver un compromis satisfaisant entre ces deux critères.

3.2 DEF

DEF (*Dynamic Evaluation Framework*) est le fruit d'une collaboration avec le Dr. Zhen He, de l'Université de La Trobe, Australie. Cet outil d'évaluation des performances des SGBD est né du constat suivant : la plupart des applications n'accèdent pas de manière répétitive au même ensemble d'objets, dans le même ordre. La capacité d'adaptation d'un système aux évolutions de séquence d'accès est donc critique pour obtenir de bonnes performances. Optimiser une base de données pour bien répondre à une séquence d'accès particulière peut d'ailleurs entraîner des dégradations de performance importantes lorsque d'autres séquences d'accès sont employées. De plus, l'étude des performances d'un sys-

tème sur une seule trace donnée fournit peu d'indications aux concepteurs d'un système, qui ont besoin de cerner et d'optimiser le comportement des composants de leur système dans différents cas d'utilisation.

Or, aucun des bancs d'essais existants ne permet à notre connaissance de modifier en cours de test les séquences d'accès aux objets. Au contraire, les bancs d'essais usuels comme ceux du TPC proposent des outils d'évaluation très standardisés. Nous avons donc proposé une modélisation du comportement dynamique d'une application [HD05, HD06]. Plus précisément, DEF est une plateforme qui permet à ses utilisateurs d'évaluer les performances de SGBD selon différents styles de séquences d'accès. De façon anecdotique, du strict point de vue de l'évaluation des performances, nous pourrions d'ailleurs considérer cet accès dynamique aux données comme un axe de complexité (Figure 1.1).

DEF décrit un ensemble de protocoles qui définissent à leur tour un ensemble de modèles d'évolution de séquence d'accès. Notre plateforme ne comprend certainement pas tous les styles d'accès possibles. Cependant, nous l'avons conçue pour être totalement extensible avec de nouveaux modèles d'évolution. DEF est également une plateforme générique qui peut être instanciée pour différentes familles de SGBD (relationnels, objets, relationnels-objets ou encore XML). En particulier, elle est conçue comme une surcouche de tout banc d'essais existant, de manière à réutiliser les standards et/ou les travaux antérieurs du domaine. Nous avons d'ailleurs illustré cette possibilité en exploitant le banc d'essais orienté objet OCB pour proposer une instance de DEF baptisée DoEF (*Dynamic object Evaluation Framework*) [HD03a, HD03b, HD04].

Nous présentons dans cette section le principe de DEF, ainsi que son instance orientée objet DoEF. Notons que nous utilisons les termes Systèmes de Gestion de Bases de Données à Objets (SGBDO) pour désigner indifféremment les systèmes orientés objets et relationnels-objets. La plupart des SGBD multimédias et compatibles XML sont des SGBDO.

3.2.1 Spécification de DEF

Contexte dynamique

Afin d'illustrer notre propos, nous commençons par donner un exemple de scénario que notre plate-forme peut simuler. Supposons que nous modélisons une librairie en ligne dans laquelle certains types de livres se vendent bien à des périodes données. Par exemple, les guides touristiques sur l'Australie ont pu être populaires lors des Jeux Olympiques de 2000. Mais une fois l'événement terminé, ces livres se sont soudainement ou graduellement moins bien vendus.

Une fois un livre sélectionné, des informations le concernant peuvent être demandées, comme un résumé, une photo de la couverture, des extraits, des critiques, etc. Dans un SGBDO, cette information est stockée sous la forme d'objets référencés par l'objet (le livre) sélectionné. Accéder à ces informations revient donc à naviguer dans un graphe d'objets dont la racine est l'objet initialement sélectionné (ici, un livre). Après avoir consulté les

informations concernant un livre, un utilisateur peut ensuite choisir un autre ouvrage du même auteur, qui devient alors la racine d'un nouveau graphe de navigation.

Nous décrivons ici les cinq principales étapes de notre démarche et les illustrons à l'aide de l'exemple que nous venons de présenter.

1. **Paramétrage des H-régions :** nous divisons tout d'abord la base de données en régions à probabilité d'accès homogène (appelées H-régions). Dans notre exemple, chaque H-région représente un ensemble de livres différent, chacun de ces ensembles possédant une probabilité d'accès propre.
2. **Spécification de la charge :** les H-régions permettent d'assigner des probabilités d'accès aux objets, mais pas de déterminer les actions à effectuer une fois un objet sélectionné. Nous appelons ces objets sélectionnés *racines de la charge* ou simplement racines. Dans cette étape, nous sélectionnons le type d'opération à effectuer sur une racine parmi ceux proposés dans OCB. Dans notre exemple, la charge est un parcours de graphe d'objets qui va de l'ouvrage sélectionné à l'information requise (par exemple, un extrait du livre).
3. **Spécification du protocole régional :** les protocoles régionaux exploitent les H-régions pour accomplir l'évolution de séquence d'accès. Différents modèles d'évolution de séquence d'accès peuvent être obtenus en faisant varier le paramétrage des H-régions au cours du temps. Par exemple, un protocole régional peut tout d'abord affecter une grande probabilité d'accès à une H-région donnée, tandis que les autres H-régions ont une faible probabilité d'accès. Après un certain temps, une H-région différente hérite de la grande probabilité d'accès. Dans notre exemple de librairie en ligne, cela modélise la baisse d'intérêt pour les guides touristiques sur l'Australie après la fin des Jeux Olympiques.
4. **Spécification du protocole de dépendance :** les protocoles de dépendance nous permettent de spécifier une relation entre la racine courante et la racine suivante. Dans notre exemple, cela peut modéliser un client qui choisit de consulter un ouvrage écrit par le même auteur que l'ouvrage qu'il a sélectionné précédemment.
5. **Intégration du protocole régional et du protocole de dépendance :** dans cette étape, nous intégrons les protocoles régionaux et de dépendance afin de modéliser des évolutions de dépendance entre des racines successives. Par exemple, un client de notre librairie en ligne peut sélectionner un livre qui l'intéresse, puis découvrir une liste d'ouvrages du même auteur qui font partie des meilleures ventes actuelles. Le client sélectionne alors l'un des livres (protocole de dépendance). L'ensemble des livres du même auteur, qui font partie des meilleures ventes *actuelles*, est lui susceptible de changer avec le temps (protocole régional).

H-régions

Les H-régions sont des sous-ensembles de la base de données de probabilité d'accès homogène. Les paramètres qui les définissent sont détaillés ci-dessous.

- *HR_SIZE* : taille de la H-région (fraction de la taille de la base).
- *INIT_PROB_W* : poids initial de la H-région. La probabilité d'accès est égale à ce poids divisé par la somme des poids de toutes les H-régions.
- *LOWEST_PROB_W* : poids minimum pour la H-région.
- *HIGHEST_PROB_W* : poids maximum pour la H-région.
- *PROB_W_INCR_SIZE* : incrément par lequel le poids est augmenté ou diminué lorsqu'une évolution survient.
- *OBJECT_ASSIGN_METHOD* : détermine la manière dont les objets sont assignés à une H-région. La sélection *aléatoire* permet de les choisir au hasard dans la base de données. La sélection *par classe* trie tout d'abord les objets selon leur identifiant de classe, avant de sélectionner les N premiers, N étant le nombre d'objets alloués à la H-région.
- *INIT_DIR* : direction initiale dans laquelle évolue l'incrément de poids (haut ou bas).

Protocoles régionaux

Les protocoles régionaux simulent les évolutions de séquence d'accès en initialisant tout d'abord les paramètres de toutes les H-régions. Ces paramètres sont ensuite modifiés périodiquement de manière prédéterminée. Nous présentons dans les paragraphes suivants trois modèles d'évolution régionale.

Fenêtre mobile. Ce protocole régional simule des changements brusques dans la séquence d'accès. Dans notre exemple, cela correspond à un livre qui devient populaire tout d'un coup (suite à une promotion télévisée, par exemple). Une fois l'événement passé, l'ouvrage retrouve sa cote de popularité initiale très rapidement. Ce modèle d'évolution est obtenu en déplaçant une fenêtre dans la base de données. Les objets de la fenêtre présentent une probabilité d'être sélectionnés comme racine très supérieure à celle des autres objets de la base. Nous atteignons cet objectif en divisant la base de données en N H-régions de tailles égales. Une de ces H-régions est choisie pour être la première région « chaude » (celle qui a la plus haute probabilité d'accès). Après un certain nombre de sélections de racine successives, une nouvelle H-région devient la région chaude.

- La base de données est divisée en N H-régions de tailles égales.
- Le paramètre *INIT_PROB_W* d'une H-région est fixé à *HIGHEST_PROB_W* (région chaude). La valeur de *INIT_PROB_W* pour les autres H-régions est fixée à *LOWEST_PROB_W*.
- Pour chaque H-région, *PROB_INCR_SIZE* est égal à *HIGHEST_PROB_W* - *LOWEST_PROB_W*. Toutes les H-régions doivent avoir les mêmes *LOWEST_PROB_W* et *PROB_W_INCR_SIZE*.
- Soit un paramètre H défini par l'utilisateur, qui reflète la vitesse d'évolution de séquence d'accès.
- Le paramètre *INIT_DIR* de toutes les H-régions est fixé vers le bas. La fenêtre est

au départ placée dans la région chaude. Après 1 / *H* sélections de racines, la fenêtre se déplace d'une H-région à une autre. La région qu'elle quitte a son paramètre *INIT_DIR* réinitialisé vers le bas, tandis que celle sur laquelle elle arrive a son paramètre *INIT_DIR* réinitialisé vers le haut. Les poids des H-régions sont ensuite incrémentés ou décrémentés, selon la valeur de *INIT_DIR*.

Fenêtre mobile graduelle. Ce protocole est similaire au précédent, mais la région chaude « se refroidit » graduellement et non brutalement. Les régions froides « se réchauffent » également graduellement au fur et à mesure que la fenêtre passe au-dessus. Cela permet de tester la faculté d'un système ou d'un algorithme à s'adapter à des types d'évolution plus doux. Dans notre exemple, ce modèle d'évolution peut décrire la baisse d'intérêt graduel pour les guides touristiques sur l'Australie après les Jeux Olympiques. Simultanément, les guides touristiques pour d'autres destinations peuvent rencontrer plus de succès.

Ce protocole est défini de la même manière que le précédent, à deux exceptions près. Premièrement, le paramètre *PROB_INCR_SIZE* est fixé par l'utilisateur et non plus calculé. Sa valeur détermine l'intensité avec laquelle la séquence d'accès change à chaque itération. Deuxièmement, l'évolution des probabilités d'accès à une H-région change. Lorsque la fenêtre arrive sur une H-région, la valeur de son paramètre *INIT_DIR* est inversée. Lorsque la fenêtre quitte une H-région, elle reste en revanche inchangée. Cela permet à la H-région de continuer à se « réchauffer » ou à se « refroidir » graduellement.

Cycles d'évolution. Ce modèle d'évolution décrit, par exemple, le comportement des clients d'une banque, qui tendraient à appartenir à une catégorie (comportementale, socio-professionnelle...) le matin et à une autre catégorie l'après-midi. Réitéré, ce processus crée un cycle d'évolution.

- La base de données est divisée en trois H-régions. Les deux premières représentent l'ensemble d'objets qui subit le cycle, la dernière la partie de la base qui demeure inchangée. Les valeurs du paramètre *HR_SIZE* des deux premières H-régions doivent être égales et sont spécifiées par l'utilisateur. Celle de la troisième H-région correspond à la taille du reste de la base.
- Les valeurs de *LOWEST_PROB_W* et *HIGHEST_PROB_W* pour les deux premières H-régions sont fixées de manière à refléter les valeurs extrêmes du cycle.
- La valeur de *PROB_INCR_SIZE* est égale à *HIGHEST_PROB_W* - *LOWEST_PROB_W* pour les deux premières H-régions, à zéro pour la dernière.
- La valeur de *INIT_PROB_W* est fixée à *HIGHEST_PROB_W* pour la première H-région et à *LOWEST_PROB_W* pour la deuxième.
- La valeur de *INIT_DIR* est dirigée vers le bas pour la région « chaude » et vers le haut pour la région « froide ».
- Un paramètre *H* est de nouveau employé pour faire varier la vitesse d'évolution des séquences d'accès.

3.2.2 DoEF

DoEF est une surcouche du banc d'essais OCB et réutilise les mêmes base de données et opérations. Nous ne détaillons pas OCB ici, mais le lecteur intéressé peut se référer à [DS00] pour une description complète. Par ailleurs, comme le modèle générique d'OCB peut être implémenté dans un système relationnel-objet et que la plupart de ses opérations demeurent pertinentes, DoEF peut donc également être utilisé dans un contexte relationnel-objet.

Nous présentons dans la suite de cette section les parties de DEF qui sont spécifiques au contexte orienté objet et plus particulièrement la spécification des protocoles de dépendance et leur intégration avec les protocoles régionaux.

Protocoles de dépendance

Il existe de nombreux scénarios au cours desquels une personne exécute une requête, puis décide d'en exécuter une autre en se basant sur les résultats de la première, établissant ainsi une dépendance entre les deux requêtes. Nous avons spécifié dans cet article quatre protocoles de dépendance.

Sélection aléatoire. Cette méthode utilise simplement une fonction aléatoire pour sélectionner la racine courante. Ce protocole modélise une personne qui lance une toute nouvelle requête après avoir exécuté la précédente.

$r_i = RAND1()$, où r_i est l'identifiant du i^{eme} objet racine. La fonction $RAND1()$ ne suit pas nécessairement une loi uniforme.

Sélection par référence. La racine courante est choisie parmi les objets référencés par la racine précédente. Dans notre exemple, ce scénario correspond à une personne qui termine la consultation d'un livre, puis recherche l'ouvrage suivant dans la même série ou collection.

$r_{i+1} = RAND2(RefSet(r_i, D))$, où $RefSet(r_i, D)$ est une fonction qui retourne l'ensemble des objets référencés par la i^{eme} racine. Nous avons utilisé deux types de références. Les références structurelles (S-références) sont simplement celles qui sont issues du graphe d'objets. Nous avons de plus introduit les D-références dans le seul but d'établir des dépendances entre les racines de parcours. Le paramètre D est utilisé pour spécifier le nombre de D-références par objet.

Sélection par objets parcourus. La racine courante est sélectionnée dans l'ensemble d'objets retournés par la requête précédente. Par exemple, un client demande une liste de livres avec leurs auteurs et leurs éditeurs, puis décide ensuite de lire un extrait de l'un des ouvrages.

$r_{i+1} = RAND3(TraversedSet(r_i, C))$, où $TraversedSet(r_i, C)$ retourne l'ensemble des objets référencés pendant le parcours effectué à partir de la i^{eme} racine. Le paramètre C

sert à limiter le nombre d'objets retournés par $TraversedSet(r_i, C)$. C'est une fraction de la cardinalité de l'ensemble des objets retournés. Ainsi, le degré de localité des objets renvoyés par $TraversedSet(r_i, C)$ peut être contrôlé (plus C est petit, plus le degré de localité est grand).

Sélection par classe. La racine courante doit appartenir à la même classe que la racine précédente. De plus, la sélection de la racine est réduite à un sous-ensemble des objets de cette classe qui dépend de la racine précédente. Par exemple, un client de notre librairie en ligne choisit un livre du même auteur que l'ouvrage qu'il vient de consulter. Dans ce cas, la fonction de sélection par classe retourne les livres écrits par le même auteur.

$r_{i+1} = RAND4(f(r_i, Class(r_i), U))$, où $Class(r_i)$ retourne la classe de la i^{eme} racine. Le paramètre U est défini par l'utilisateur et indique la cardinalité de l'ensemble d'objets retourné par la fonction $f()$. C'est en fait une fraction de la cardinalité totale de la classe. Il peut être utilisé pour faire varier le degré de localité entre les objets retournés par $f()$. $f()$ doit être injective.

Sélection hybride. Ce type de sélection permet d'utiliser une combinaison des protocoles de dépendance décrits ci-dessus. Cette possibilité est importante, car elle peut simuler par exemple un utilisateur qui initie une requête totalement nouvelle après avoir suivi des dépendances. La sélection hybride est implémentée en deux phases. La première *phase aléatoire* utilise la sélection aléatoire pour identifier une première racine. Dans la seconde *phase de dépendance*, un des protocoles de dépendance ci-dessus est appliqué pour sélectionner la racine suivante. La seconde phase est répétée R fois avant que la première ne survienne à nouveau. Les deux phases sont réitérées de manière continue.

La probabilité de sélectionner un protocole de dépendance donné dans la *phase de dépendance* est spécifiée à l'aide des paramètres suivants : *RANDOM_DEP_PROB* (sélection aléatoire), *SREF_DEP_PROB* (sélection par référence sur les S-références), *DREF_DEP_PROB* (sélection par référence sur les D-références), *TRAVERSED_DEP_PROB* (sélection par objets parcourus) et *CLASS_DEP_PROB* (sélection par classe).

Intégration des protocoles régionaux et de dépendance

Les protocoles de dépendance modélisent le comportement des utilisateurs. Comme ce dernier peut changer au cours du temps, ces protocoles doivent également être capables d'évoluer. En intégrant les protocoles régionaux et de dépendance, nous parvenons à simuler des évolutions de dépendance entre des sélections successives de racines. Ceci est facilement accompli en exploitant la propriété des protocoles de dépendance de retourner un ensemble d'objets racines candidats en fonction d'une racine précédente donnée. Jusqu'à présent, la racine courante était sélectionnée dans cet ensemble à l'aide d'une fonction aléatoire. Au lieu de procéder ainsi, nous avons partitionné l'ensemble des racines candidates en H-régions et appliqué les protocoles régionaux à ces H-régions. Lorsque le

protocole de dépendance « sélection par objets parcourus » est utilisé, la propriété suivante doit être vérifiée : pour un même objet racine, le même ensemble d'objets doit être parcouru. Ainsi, une racine donnée donne toujours lieu au même parcours.

3.2.3 Bilan

Notre objectif en concevant DEF était de spécifier une plateforme d'évaluation de performance qui permette aux concepteurs et aux utilisateurs de SGBDO de tester un système donné à l'aide d'une charge dite dynamique. En effet, si la plupart des applications réelles présentent des évolutions dans leurs séquences d'accès aux données, les bancs d'essais actuels ne modélisent pas ce type de comportement. Nous avons donc cherché à augmenter leur pertinence en en proposant une surcouche dynamique.

Par ailleurs, nous avons conçu DEF comme une plateforme ouverte et extensible, selon deux axes. Premièrement, nous avons pris grand soin de garantir l'intégration facile de nouveaux modèles d'évolution de séquence d'accès, principalement grâce à la définition des H-régions. Deuxièmement, bien que nous n'ayons instancié notre plateforme que dans un environnement orienté objet avec DoEF, nous pourrions appliquer les concepts que nous avons développés à d'autres familles de SGBD. Instancier DEF pour les bases de données relationnelles-objets serait par exemple relativement facile. En effet, le banc d'essais OCB peut être assez aisément adapté au modèle relationnel-objet (bien que des extensions soient clairement nécessaires, comme les types de données abstraits et les tables emboîtées, par exemple). DoEF, en tant que surcouche d'OCB, peut donc également être utilisé dans un contexte relationnel-objet. Réutiliser les bancs d'essais BUCKY ou BORD (Section 3.1.3) est également envisageable.

Finalement, les résultats des expérimentations que nous avons menées ont démontré que DoEF permettait d'atteindre notre objectif d'observation des performances des SGBDO (identifier les composants qui forment des goulots d'étranglement, par exemple) lorsque les séquences d'accès aux données varient.

Nous avons en effet comparé les performances de trois algorithmes de regroupement dynamique d'objets : DSTC [BS96], DRO [DFR+00] et OPCF [HMB00], dont la composante dynamique ne pouvait être prise en compte par les bancs d'essais existants. Nos expérimentations nous ont permis de montrer que ces algorithmes peuvent s'adapter à des évolutions lentes des séquences d'accès, mais que leurs performances s'effondrent lorsque les changements sont rapides. De plus, nous avons pu proposer une stratégie de regroupement dite prudente et flexible pour prendre en compte de telles évolutions.

Nous avons également étudié le comportement de deux gestionnaires d'objets persistants : SHORE [CDF+94] et Platypus [HBKZ00]. La variation des séquences d'accès aux données nous a dans ce cas permis de mettre en évidence des problèmes de pagination disque de Platypus qui n'étaient pas apparus lors des tests menés par ses concepteurs à l'aide de bancs d'essais classiques.

3.3 DWEB

DWEB (*Data Warehouse Engineering Benchmark*) a été développé au sein du laboratoire ERIC, notamment dans l'optique d'évaluer les performances des stratégies automatiques d'optimisation des performances des entrepôts de données que nous avons présentées au Chapitre 2. Nous positionnons d'ailleurs ce banc d'essais essentiellement comme un outil à destination des concepteurs d'entrepôts de données et des chercheurs (Section 3.1.6).

DWEB permet la génération automatique d'entrepôts de données synthétiques *ad-hocs* (modélisés en étoile, en flocon de neige ou en constellation) et des charges de travail (ensembles de requêtes décisionnelles) associées [DBB04, DBB05, DBB07]. Cette génération est pilotée par un ensemble de paramètres qui permettent à l'utilisateur de sélectionner précisément la configuration d'entrepôt et de charge voulue.

Nous avons à l'origine conçu DWEB car nos besoins en matière d'évaluation des performances n'étaient pas satisfaits par les bancs d'essais décisionnels existants (Section 3.1.5). En effet, les schémas d'APB-1, TPC-H et TPC-DS sont fixes, alors que nos techniques d'optimisation des performances nécessitent d'être testées sur des configurations variées d'entrepôts de données. Les utilisateurs de ces bancs d'essais n'ont également aucun contrôle sur la définition de la charge. Par exemple, le nombre de requêtes générées dépend directement de l'unique paramètre SF dans TPC-DS. Ces bancs d'essais ne respectent donc pas le critère d'adaptabilité que nous avons défini à la Section 3.1.6.

Par ailleurs, leur utilisation pose des problèmes pratiques. APB-1 n'est plus supporté et s'avère difficile à se procurer. Le schéma de la base de données de TPC-H, qui est hérité du banc d'essais plus ancien TPC-D, n'est pas dimensionnel (ce n'est pas un schéma en étoile ni un de ses dérivés). Sa charge, bien qu'orientée décision, n'inclut pas non plus explicitement de requêtes OLAP. Finalement, la publication des spécifications complètes de TPC-DS n'est pas encore effective à ce jour, aussi est-il actuellement impossible d'exploiter ce banc d'essais.

Nous détaillons dans les sections suivantes les deux éléments principaux du banc d'essais DWEB : son modèle de base de données et son modèle de charge.

3.3.1 Base de données de DWEB

Schéma

Notre objectif avec DWEB est de pouvoir modéliser différents types d'architecture d'entrepôts de données [Inm02, KR02] au sein d'un environnement ROLAP (*Relational OLAP*) :

- des schémas en étoile classiques,
- des schémas en flocon de neige (avec des dimensions hiérarchisées),
- des schémas en constellation (avec plusieurs tables de faits et des dimensions partagées).

Afin d'atteindre ce but, nous proposons un métamodèle d'entrepôt de données (représenté par un diagramme de classes UML dans la Figure 3.3) susceptible d'être instancié en ces différents schémas. Nous considérons ce métamodèle comme un intermédiaire entre le métamodèle multidimensionnel du standard CWM (*Common Warehouse Metamodel*) [Obj03, PCTM03] et le modèle final du banc d'essais. Notre métamodèle est en fait une instance de celui de CWM, qui pourrait en fait être qualifié de méta-métamodèle dans notre contexte.

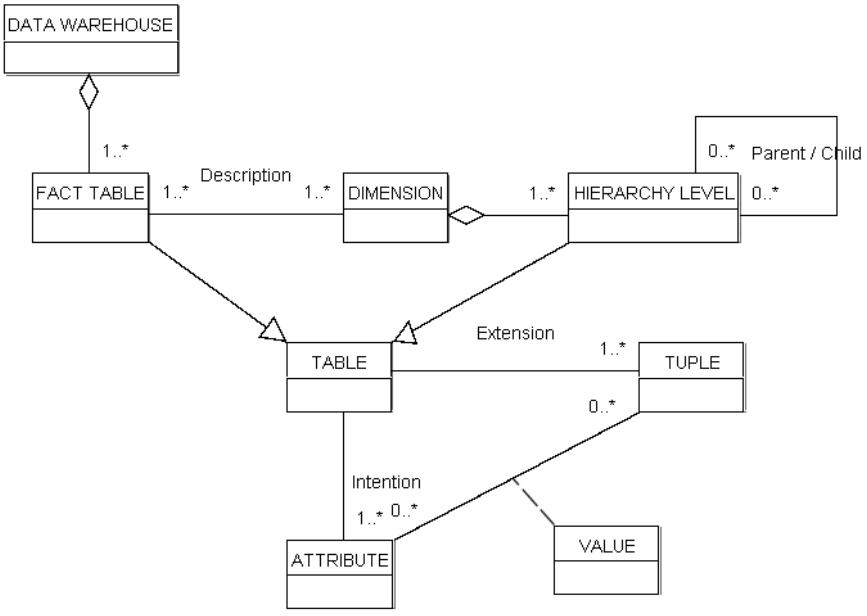

FIG. 3.3 – Métaschéma de l'entrepôt de données de DWEB

La partie supérieure de la Figure 3.3 décrit un entrepôt de données (ou un magasin de données si ce dernier est perçu comme un petit entrepôt dédié) constitué d'une ou plusieurs tables de faits qui sont chacune décrite par plusieurs dimensions. Chaque dimension peut également décrire plusieurs tables de faits (dimensions partagées) et peut contenir une ou plusieurs hiérarchies composées de différents niveaux. Il est possible de n'avoir qu'un seul niveau, auquel cas la dimension n'est pas hiérarchisée.

Les tables de faits et les niveaux hiérarchiques des dimensions sont tous des tables relationnelles, qui sont modélisées dans la partie inférieure de la Figure 3.3. Classiquement, une table ou relation est définie en intention par ses attributs et en extension par ses n-uplets. À l'intersection d'un attribut et d'un n-uplet donnés se trouve la valeur de cet attribut dans ce n-uplet.

Notre métamodèle est relativement simple. Il est en effet suffisant pour modéliser les schémas d'entrepôts de données que nous visons (en étoile, en flocon de neige, en constellation), mais demeure limité et ne prend pas en compte certaines particularités qui existent dans les entrepôts de données réels, comme des associations plusieurs à plusieurs entre une table de faits et une dimension, ou des niveaux hiérarchiques partagés par plusieurs

dimensions. C'est un choix délibéré de ne pas compliquer le métamodèle à ce stade, mais ce dernier pourrait bien sûr être étendu par la suite.

3.3.2 Paramétrage

Afin de maximiser la pertinence de notre banc d'essais tout en ne sacrifiant pas trop à sa simplicité, nous proposons de subdiviser les paramètres en deux sous-ensembles :

- un sous-ensemble détaillé de paramètres de bas niveau qui permet à un utilisateur avancé de controler totalement la génération de l'entrepôt de données (Tableau 3.1) ;
- une couche supérieure contenant beaucoup moins de paramètres facilement compréhensibles et instanciables (Tableau 3.2). Ces paramètres de haut niveau sont plus précisément les valeurs moyennes des paramètres de bas niveau. Lors de la génération de la base de données, ils sont exploités par des fonctions aléatoires (qui suivent une distribution gaussienne) pour fixer la valeur des paramètres de bas niveau. Finalement, bien que le nombre de paramètres de bas niveau puisse augmenter de façon significative si le schéma grossit, le nombre de paramètres de haut niveau, lui, demeure constant et raisonnable (moins de dix paramètres).

Les utilisateurs peuvent alors choisir d'initialiser l'ensemble complet de paramètres de bas niveau ou bien seulement les paramètres de haut niveau, pour lesquels nous proposons des valeurs par défaut correspondant à un schéma en flocon de neige.

Nom du paramètre	**Signification**
NB_FT	Nombre de tables de faits
$NB\_DIM(f)$	Nombre de dimensions décrivant la table de faits f
TOT_NB_DIM	Nombre total de dimensions
$NB\_MEAS(f)$	Nombre de mesures dans la table de faits f
$DENSITY(f)$	Densité de la table de faits f
$NB\_LEVELS(d)$	Nombre de niveaux hiérarchiques dans la dimension d
$NB\_ATT(d,h)$	Nombre d'attributs dans le niveau hiérarchique h de la dimension d
$HHLEVEL\_SIZE(d)$	Nombre de n-uplets dans le plus haut niveau hiérarchique de la dimension d
$DIM\_SFACTOR(d)$	Facteur d'échelle pour la taille des niveaux hiérarchiques de la dimension d

TAB. 3.1 – Paramètres de bas niveau de l'entrepôt de données de DWEB

Remarques :

- Puisque des dimensions partagées sont possibles, $TOT\_NB\_DIM \leq \sum_{i=1}^{NB\_FT} NB\_DIM(i)$.
- La cardinalité d'une table de faits est habituellement inférieure ou égale au produit des cardinalités de ses dimensions. C'est pourquoi nous introduisons la notion de densité. Une densité de 1 indique que toutes les combinaisons possibles des clés primaires des dimensions sont présentes dans la table de faits. Quand la densité

Nom du paramètre	Signification	Défaut
AVG_NB_FT	Nombre moyen de tables de faits	1
AVG_NB_DIM	Nombre moyen de dimensions par table de fait	5
$AVG\_TOT\_NB\_DIM$	Nombre moyen total de dimensions	5
AVG_NB_MEAS	Nombre moyen de mesures dans les tables de faits	5
$AVG\_DENSITY$	Densité moyenne des tables de faits	0,6
AVG_NB_LEVELS	Nombre moyen de niveaux hiérarchiques dans les dimensions	3
AVG_NB_ATT	Nombre moyen d'attributs dans les niveaux hiérarchiques	5
$AVG\_HHLEVEL\_SIZE$	Nombre moyen de n-uplets dans les plus hauts niveaux hiérarchiques	10
$DIM\_SFACTOR$	Facteur d'échelle de taille moyen au sein des niveaux hiérarchiques	10

TAB. 3.2 – Paramètres de haut niveau de l'entrepôt de données de DWEB

diminue, nous éliminons progressivement certaines de ces combinaisons (voir section suivante).

– Au sein d'une dimension, un niveau hiérarchique donné a normalement une cardinalité plus grande que le niveau suivant. Par exemple, dans une hiérachie de type *villes-régions-pays*, le nombre de villes doit être plus grand que le nombre de régions, qui doit à son tour être supérieur au nombre de pays. De plus, il existe souvent un facteur d'échelle significatif entre ces cardinalités (par exemple, mille villes, cent régions, dix pays). C'est pourquoi nous définissons la cardinalité des niveaux hiérarchiques en assignant une cardinalité « de départ » au plus haut niveau de la hiérarchie ($HHLEVEL\_SIZE$). Nous la multiplions ensuite par un facteur d'échelle prédéfini ($DIM\_SFACTOR$) pour chaque niveau hiérarchique inférieur.

Algorithme de génération

L'instanciation du métaschéma de DWEB en un schéma d'entrepôt de données réel s'effectue en deux étapes :

1. construction des dimensions et de leurs hiérarchies ;
2. construction des tables de faits.

L'algorithme correspondant à ces deux étapes est présenté dans les Figures 3.4 et 3.5, respectivement. Chacune d'entre elles est constituée, pour chaque dimension ou table de fait, de la génération de son intention, puis de son extension. De plus, les hiérarchies des dimensions doivent être gérées. Il faut noter qu'elles sont générées en démarrant du plus haut niveau hiérarchique. Par exemple, pour notre hiérarchie *villes-régions-pays*, nous construisons le niveau *pays* en premier, puis le niveau *région* et enfin le niveau *ville*. Ainsi,

les n-uplets d'un niveau hiérarchique donné peuvent référencer ceux du niveau supérieur (qui sont déjà créés) à l'aide d'une clé étrangère.

```
For i = 1 to TOT_NB_DIM do
    previous_ptr = NIL
    size = HHLEVEL_SIZE(i)
    For j = 1 to NB_LEVELS(i) do
       // Intention
       hl = New(Hierarchy_level)
       hl.intention = Primary_key()
       For k = 1 to NB_ATT(i,j) do
          hl.intention = hl.intention ∪ String_descriptor()
       End for
       // Gestion des hiérarchies
       hl.child = previous_ptr
       hl.parent = NIL
       If previous_ptr ≠ NIL then
          previous_ptr.parent = hl
          hl.intention = hl.intention
             ∪ previous_ptr.intention.primary_key // Clé étr.
       End if
       // Extension
       hl.extension = ∅
       For k = 1 to size do
          new_tuple = Integer_primary_key()
          For l = 1 to NB_ATT(i,j) do
             new_tuple = new_tuple ∪ Random_string()
          End for
          If previous_ptr ≠ NIL then
             new_tuple = new_tuple ∪ Random_key(previous_ptr)
          End if
          hl.extension = hl.extension ∪ new_tuple
       End for
       previous_ptr = hl
       size = size * DIM_SFACTOR(i)
    End for
    dim(i) = hl // Premier (plus bas) niveau de la hiérarchie
End for
```

FIG. 3.4 – Algorithme de génération des dimensions de DWEB

Nous utilisons trois classes principales de fonctions et une procédure dans ces algorithmes.

1. `Primary_key()`, `String_descriptor()` et `Float_measure()` renvoient des noms pour les clés primaires, les descripteurs des niveaux hiérarchiques et les mesures des tables de faits, respectivement. Ces noms sont étiquetés séquentiellement et préfixés par le nom de la table (par exemple, DIM1_1_DESCR1, DIM1_1_DESCR2...).

2. `Integer_primary_key()`, `Random_key()`, `Random_string()` et `Random_float()` renvoient des entiers séquentiels par rapport à une table donnée (aucun doublon n'est permis), des instances aléatoires de la clé primaire de la table spécifiée (valeurs aléatoires pour une clé étrangère), des chaînes de caractères aléatoires de taille fixe (20

```
For i = 1 to NB_FT do
    // Intention
    ft(i).intention = ⊘
    For j = 1 to NB_DIM(i) do
       j = Random_dimension(ft(i))
       ft(i).intention = ft(i).intention ∪
ft(i).dim(j).primary_key
    End for
    For j = 1 to NB_MEAS(i) do
       ft(i).intention = ft(i).intention ∪ Float_measure()
    End for
    // Extension
    ft(i).extension = ⊘
    For j = 1 to NB_DIM(i) do // Produit cartésien
       ft(i).extension = ft(i).extension ×
ft(i).dim(j).primary_key
    End for
    to_delete = DENSITY(i) * |ft(i).extension|
    For j = 1 to to_delete do
       Random_delete(ft(i).extension)
    End for
    For j = 1 to |ft(i).extension| do
    // |ft(i).extension| est supposée à jour
       For k = 1 to NB_MEAS(i) do
          ft(i).extension.tuple(j).measure(k) = Random_float()
       End for
    End for
End for
```

FIG. 3.5 – Algorithme de génération des tables de faits de DWEB

caractères) sélectionnées dans un référentiel préalablement construit et préfixées par le nom de l'attribut correspondant, ainsi que des nombres aléatoires réels simple précision, respectivement.

3. `Random_dimension()` renvoie une dimension sélectionnée parmi les dimensions existantes qui ne décrivent pas déjà la table de faits passée en argument.
4. `Random_delete()` efface aléatoirement un n-uplet dans l'extension d'une table.

A l'exception de la procédure `Random_delete()`, dans laquelle nous utilisons une distribution aléatoire uniforme, nous employons des distributions aléatoires gaussiennes.

Remarque : La manière dont la densité est gérée dans la Figure 3.5 est clairement non-optimale. Nous choisissons de présenter l'algorithme de cette manière afin de le rendre plus clair, mais notre implémentation ne crée pas tous les n-uplets du résultat du produit cartésien avant d'en effacer certains. Nous générons directement le bon nombre de n-uplets en utilisant la densité comme une probabilité de création de chaque n-uplet.

3.3.3 Charge de DWEB

Dans un banc d'essais pour entrepôts de données, la charge peut être subdivisée en :
– une charge de requêtes décisionnelles (principalement des requêtes OLAP) ;

– le processus d'ETL (chargement et rafraîchissement des données).

TPC-DS gère ces deux aspects. Puisque la base de données de DWEB compose la partie la plus originale de notre travail, nous nous inspirons donc de la définition de charge de TPC-DS, tout en exploitant d'autres sources d'informations concernant la performance des entrepôts de données [BMC00, Gre04]. Néanmoins, il nous est nécessaire d'adapter les propositions de TPC-DS à la nature variable des entrepôts de données de DWEB. De plus, nous voulons satisfaire le critère de simplicité de Gray, aussi proposons-nous au final une charge plus simple que celle de TPC-DS.

Par ailleurs, nous nous concentrons dans un premier temps sur la définition d'un modèle de requêtes. La modélisation du processus d'ETL complet est une tâche complexe sur laquelle nous comptons revenir plus tard. Nous considérons pour l'instant que les spécifications actuelles de DWEB permettent une évaluation grossière d'un chargement d'entrepôt. En effet, la base de données de DWEB peut être générée sous forme de fichiers plats, puis ensuite chargée dans un entrepôt en utilisant les outils fournis par le système.

Modèle de requêtes

La charge de DWEB modélise deux types de requêtes :

– des requêtes purement décisionnelles contenant les opérations OLAP usuelles telles que cube, agrégation (*roll-up*), forage (*drill down*) et projection/sélection (*slice and dice*) ;

– des requêtes d'extraction.

Nous définissons notre modèle de requêtes générique (Figure 3.6) à l'aide d'une grammaire qui est un sous-ensemble du standard SQL-99, qui introduit des outils analytiques (indispensables dans un contexte décisionnel) dans les requêtes relationnelles. Cela nous permet de générer des requêtes SQL analytiques dynamiquement.

Paramétrage

Comme les paramètres de la base de données de DWEB (Section 3.3.2), ceux qui définissent sa charge (Tableau 3.3) ont été conçus pour répondre au critère de simplicité de Gray. Ils déterminent la manière dont le modèle de requêtes de la Figure 3.6 s'instancie. Nous ne disposons ici que d'un nombre limité de paramètres de haut niveau (huit paramètres, puisque $PROB\_EXTRACT$ et $PROB\_ROLLUP$ sont calculés à partir de $PROB\_$
$OLAP$ et $PROB\_CUBE$, respectivement). Il n'est en effet pas envisageable d'entrer plus dans le détail si la taille de la charge atteint cinq cents requêtes comme c'est le cas dans TPC-DS, par exemple.

Remarque : NB_Q n'est qu'une approximation du nombre de requêtes parce que le nombre de forages (*drill downs*) effectués après une requête OLAP est variable. Nous ne pouvons donc arrêter le processus de génération que lorsque nous avons effectivement produit au moins NB_Q requêtes.

Query ::-	
Select	![<Attribute Clause> | <Aggregate Clause> | [<Attribute Clause>, <Aggregate Clause>]]
From	!<Table Clause> [<Where Clause> || [<Group by Clause> *<Having Clause>]]
Attribute Clause ::-	*Attribute Name* [[, <Attribute Clause>] | ⊥]
Aggregate Clause ::-	![*Aggregate Function Name* (*Attribute Name*)] [**As** *Alias*] [[, <Aggregate Clause>] | ⊥]
Table Clause ::-	*Table Name* [[, <Table Clause>] |⊥]
Where Clause ::-	**Where** ![<Condition Clause> | <Join Clause> | [<Condition Clause> **And** <Join Clause>]]
Condition Clause ::-	![*Attribute Name* <Comparison Operator> <Operand Clause>] [[<Logical Operator> <Condition Clause>] | ⊥]
Operand Clause ::-	[*Attribute Name* | *Attribute Value* | *Attribute Value List*]
Join Clause ::-	![*Attribute Name* i = *Attribute Name* j] [[**And** <Join Clause>] |⊥]
Group by Clause ::-	**Group by** [**Cube** | **Rollup**] <Attribute Clause>
Having Clause ::-	[*Alias* | *Aggregate Function Name* (*Attribute Name*)] <Comparison Operator> [*Attribute Value* | *Attribute Value List*]
Key:	Les crochets [et] sont des délimiteurs. !<A>: A est requis. *<A>: A est optionnel. <A || B>: A ou B. <A | B>: A ou exclusif B. ⊥: clause vide. Les éléments du langage SQL sont indiqués en gras.

FIG. 3.6 – Modèle de requêtes de DWEB

Nom du paramètre	**Signification**	**Défaut**
NB_Q	Nombre approximatif de requêtes dans la charge	100
AVG_NB_ATT	Nombre moyen d'attributs sélectionnés dans une requête	5
AVG_NB_RESTR	Nombre moyen de restrictions dans une requête	3
$PROB\_OLAP$	Probabilité que le type de requête soit OLAP	0,9
$PROB\_EXTRACT$	Probabilité que la requête soit une requête d'extraction	$1 - PROB\_OLAP$
AVG_NB_AGGREG	Nombre moyen d'agrégations dans une requête OLAP	3
$PROB\_CUBE$	Probabilité qu'une requête OLAP utilise l'opérateur *Cube*	0,3
$PROB\_ROLLUP$	Probabilité qu'une requête OLAP utilise l'opérateur *Rollup*	$1 - PROB\_CUBE$
$PROB\_HAVING$	Probabilité qu'une requête OLAP contienne une clause *Having*	0,2
AVG_NB_DD	Nombre moyen de forages (*drill downs*) après une requête OLAP	3

TAB. 3.3 – Paramètres de la charge de DWEB

Algorithme de génération

L'algorithme de génération de la charge de DWEB est présenté dans les Figures 3.7 et 3.8. Son objectif est de générer un ensemble de requêtes SQL-99 qui puisse être directement exécuté sur l'entrepôt de données synthétique défini à la Section 3.3.1. Il est subdivisé en deux étapes :

1. génération d'une requête initiale qui peut être soit une requête OLAP, soit une requête d'extraction ;
2. si la requête initiale était de type OLAP, exécution d'un certain nombre de forages basés sur cette première requête. Plus précisément, à chaque fois qu'un nouveau forage est exécuté, un attribut d'un niveau inférieur de la hiérarchie d'une dimension est ajouté à la clause *Attribute Clause* de la requête précédente.

La première étape est elle-même subdivisée en trois sous-étapes :

1. les clauses `Select`, `From` et `Where` d'une requête sont générées simultanément en sélectionnant aléatoirement une table de faits et des dimensions (y compris un niveau hiérarchique dans chacune de ces dimensions) ;
2. la clause `Where` est complétée par des conditions supplémentaires ;
3. finalement, le choix du type de requête est effectué (requête OLAP ou requête d'extraction). Dans le second cas, la requête est terminée. Dans le premier, des fonctions d'agrégat appliquées à des mesures de la table de faits sont ajoutées à la requête,

ainsi qu'une clause `Group by` pouvant inclure les opérateurs `Cube` ou `Rollup`. Il est également possible d'ajouter une clause `Having`. La fonction d'agrégat que nous appliquons sur les mesures est toujours une somme car c'est l'opération la plus courante dans les cubes. De plus, les autres fonctions d'agrégat ont des complexités similaires en temps de calcul. Elles n'apporteraient donc pas plus d'information dans le cadre d'une étude de performance.

Nous utilisons trois classes de fonctions et une procédure dans cet algorithme.

1. `Random_string()` et `Random_float()` sont les mêmes fonctions que celles décrites dans la Section 3.3.2. Cependant, nous introduisons ici la possibilité pour `Random_float()` d'utiliser une distribution aléatoire soit uniforme, soit gaussienne. Cela dépend des paramètres passés à la fonction : distribution uniforme pour un intervalle de valeurs, distribution gaussienne pour une valeur moyenne. Finalement, nous introduisons la fonction `Random_int()` qui se comporte exactement comme `Random_float()`, mais qui retourne des valeurs entières.
2. `Random_FT()` et `Random_dimension()` permettent de sélectionner une table de faits et une dimension qui décrit une table de faits donnée, respectivement. Elles utilisent toutes deux une distribution aléatoire gaussienne. `Random_dimension()` est également déjà décrite dans la Section 3.3.2.
3. `Random_attribute()` et `Random_measure()` ont des comportements très similaires. Elles retournent un attribut ou une mesure, respectivement, appartenant à l'intention d'une table ou à une liste d'attributs. Elles utilisent toutes deux une distribution aléatoire gaussienne.
4. `Gen_query()` est la procédure qui est chargée de générer effectivement le code SQL-99 des requêtes de la charge, d'après les paramètres nécessaires pour instancier notre modèle de requêtes.

3.3.4 Bilan

DWEB a été conçu pour aider les concepteurs d'entrepôts de données à choisir entre différentes architectures et/ou techniques d'optimisation des performances. Pour aller dans le sens de la pertinence et de l'adaptabilité, DWEB permet la génération d'entrepôts de données synthétiques variés et des charges décisionnelles associées. En effet, les schémas d'entrepôt les plus courants (en étoile, en constellation et en flocon de neige), ainsi que les opérateurs décisionnels les plus utilisés (*cube*, *rollup*, *drill-down*...) sont supportés par notre outil. Ces fonctionnalités sont pilotées par un ensemble complet de paramètres de bas niveau, mais nous avons proposé une série de paramètres de haut niveau en nombre limité afin de ne pas trop sacrifier au critère de simplicité de Gray. Finalement, nous avons opté pour une implémentation en langage Java de DWEB afin de satisfaire au critère de portabilité.

Par ailleurs, nous avons illustré quelques utilisations possibles de DWEB en évaluant l'efficacité de plusieurs techniques d'indexation sur différentes configurations d'entrepôts

```
n = 0
While n < NB_Q do
    // Etape 1 : Requête initiale
    // Etape 1.2 : Clauses Select, From et Where
    i = Random_FT() // Sélection de la table de faits
    attribute_list = ∅
    table_list = ft(i)
    condition_list = ∅
    For k = 1 to Random_int(AVG_NB_ATT) do
       j = Random_dimension(ft(i)) // Sélection d'une dimension
       l = Random_int(1, ft(i).dim(j).nb_levels)
       // Positionnement au niveau hiérarchique l
       hl = ft(i).dim(j) // Niveau hiérarchique courant
       m = 1 // Compteur de niveau
       fk = ft(i).intention.primary_key.element(j)
       // (clé étrangère sur ft(i).dim(j).primary_key)
       While m < l and hl.child ≠ NIL do
          // Construction de la jointure
          table_list = table_list ∪ hl
          condition_list = condition_list
            ∪ (fk = hl.intention.primary_key)
          // Niveau suivant
          fk = hl.intention.foreign_key
          m = m + 1
          hl = hl.child
       End while
       attribute_list = attribute_list
         ∪ Random_attribute(hl.intention)
    End for
    // Etape 1.2 : Compléter la clause Where
    For k = 1 to Random_int(AVG_NB_RESTR) do
       condition_list = condition_list
         ∪ (Random_attribute(attribute_list) = Random_string())
    End for
    // Etape 1.3 : Sélection requête OLAP ou requête d'extraction
    p1 = Random_float(0,1)
    If p1 ≤ PROB_OLAP then // Requête OLAP
       // Agrégat
       aggregate_list = ∅
       For k = 1 to Random_int(AVG_NB_AGGREG) do
          aggregate_list = aggregate_list
             ∪ Random_measure(ft(i).intention)
       End for
../..
```

FIG. 3.7 – Algorithme de génération de la charge de DWEB – Partie 1

```
../..
      // Clause Group by
      group_by_list = attribute_list
      p2 = Random_float(0,1)
      If p2 ≤ PROB_CUBE then
         group_by_operator = CUBE
      Else
         group_by_operator = ROLLUP
      End if
      // Clause Having
      p3 = Random_float(0,1)
      If p3 ≤ PROB_HAVING then
         having_clause = (Random_attribute(aggregate_list), ≥,
            Random_float())
      Else
         having_clause = ∅
      End if
   Else // Requête d'extraction
      group_by_list = ∅
      group_by_operator = ∅
      having_clause = ∅
   End if
   // Génération de la requête SQL
   Gen_query(attribute_list, aggregate_list, table_list,
      condition_list, group_by_list, group_by_operator,
      having_clause)
   n = n + 1
   // Etape 2 : Eventuelles requêtes DRILL DOWN
   If p1 ≤ PROB_OLAP then
      k = 0
      While k < Random_int(AVG_NB_DD) and hl.parent ≠ NIL do
         k = k +1
         hl = hl.parent
         att = Random_attribute(hl.intention)
         attribute_list = attribute_list ∪ att
         group_by_list = group_by_list ∪ att
         Gen_query(attribute_list, aggregate_list, table_list,
            condition_list, group_by_list, group_by_operator,
            having_clause)
      End while
      n = n + k
   End if
End while
```

FIG. 3.8 – Algorithme de génération de la charge de DWEB – Partie 2

[DBB05, DBB07]. Ces expériences n'étaient pas novatrices en elles-mêmes, mais elles nous ont permis de montrer la pertinence de notre outil en retrouvant des résultats précédemment publiés en ce qui concerne les index *bitmap* de jointure [OG95] et les index de jointure en étoile [BKM02]. Nous avons ainsi pu souligner à nouveau la nature critique des choix d'indexation dans un entrepôt de données. De plus, comme ces choix dépendent de l'architecture de l'entrepôt et des données qui y sont stockées, nous avons montré l'utilité de DWEB dans ce contexte, alors que les bancs d'essais « mono-schéma » ne s'appliquent pas.

À notre connaissance, DWEB est le plus abouti des bancs d'essais décisionnels actuellement opérationnels. Il est cependant largement perfectible à différents niveaux. Tout d'abord, nous avons dans un premier temps délibérément simplifié notre métamodèle d'entrepôt et notre modèle de requêtes. Des extensions pourraient les rendre plus pertinents, en regard d'entrepôts de données réels. Par exemple, le métamodèle d'entrepôt pourrait permettre des associations de type « plusieurs à plusieurs » entre les dimensions et la table de faits ou encore le partage de niveaux hiérarchiques par plusieurs dimensions. Les requêtes imbriquées qui sont courantes dans les opérations OLAP pourraient également être intégrées dans notre modèle de requêtes. Finalement, il serait très intéressant, bien que complexe, d'inclure réellement le processus d'ETL dans notre modèle de charge. Les spécifications de TPC-DS ainsi que d'autres études pourraient nous y aider [LR98].

Par ailleurs, l'ensemble des paramètres de DWEB correspond à la fois aux modèles que nous avons développés et à l'utilisation probable de notre banc d'essais. Une validation formelle pourrait cependant nous permettre de sélectionner les paramètres les plus pertinents, voire d'en introduire de nouveaux, comme la cardinalité du domaine des attributs des dimensions ou le facteur de sélectivité des restrictions dans les requêtes. Des expérimentations complémentaires devraient nous aider dans ce sens, ainsi que nous permettre de proposer des valeurs par défaut basées sur l'expérience.

Finalement, nous avons pris pour hypothèse en concevant DWEB qu'il était aisé de définir un protocole d'exécution et des mesures de performance, par exemple en réemployant ceux de TPC-DS. Cette tâche doit néanmoins être accomplie. Nous n'avons par exemple utilisé que le temps de réponse comme mesure dans nos expérimentations, mais d'autres doivent être envisagées, comme celles qui permettent de mesurer la qualité du modèle conceptuel d'un entrepôt [SCP03a, SCP03b, SCT+04a, SCT+04b]. La validation formelle de ces mesures améliorerait également la qualité de notre banc d'essais.

3.4 XDW-Bench

Étant donnée la récence du développement des SGBD natifs XML, un important travail d'optimisation des performances a été entrepris pour les amener à devenir une alternative crédible aux SGBD relationnels dits compatibles XML. Divers outils d'évaluation des performances ont été proposés pour accompagner cet effort (Section 3.1.4), mais aucun d'entre eux n'a jusqu'ici été consacré à une approche décisionnelle.

XDW-Bench (*XML Data Warehouse Benchmark*) est à notre connaissance le premier banc d'essais pour entrepôts de données XML. Son développement a fait l'objet du stage de master recherche d'Antoine Diouf, que j'ai encadré au sein du laboratoire ERIC en 2005 [Dio05]. Notre objectif en concevant cet outil a été de proposer un premier cadre d'évaluation des performances pour les entrepôts de données XML, pour lesquelles de nombreuses architectures ont été proposées dans la littérature récemment (Section 3.4.1).

Notre travail sur XDW-Bench est encore en cours et procède par étapes. La première version du banc d'essais est volontairement plus simple que DWEB (Section 3.3), et ce pour plusieurs raisons. Premièrement, plusieurs architectures d'entrepôts de données XML concurrentes ont été proposées. Si elles convergent toutes vers une traduction logique XML des modèles classiques en étoile et en flocon de neige, elles présentent néanmoins des différences et la stabilité des modèles d'entrepôts XML n'est actuellement pas assurée. Nous avons donc avancé le principe d'un banc d'essais pour entrepôts XML et, tout en opérant un choix d'architecture, nous avons favorisé l'évolutivité de notre solution en privilégiant le critère de simplicité de Gray.

Par ailleurs, le comportement des SGBD natifs XML actuellement disponibles est très sensible à la volumétrie des données, notamment lorsque de nombreux documents XML de grande taille sont stockés, comme c'est le cas dans les entrepôts. Il nous a donc paru prématuré de permettre la génération d'entrepôts et de charges décisionnelles trop élaborés. Finalement, l'objet de notre travail est de parvenir à une deuxième version de XDW-Bench qui intègre des données complexes, de manière à évaluer tout le profit d'une approche XML pour l'entreposage de données complexes (Section 1.2). Il nous a paru plus raisonnable de partir sur une base simple avant d'en arriver à ce stade.

Nous évoquons tout d'abord dans la suite de cette section les modèles d'entrepôts de données XML existants et détaillons celui que nous avons sélectionné, avant de présenter les spécifications de la première version de XDW-Bench (c'est-à-dire sans introduction de données complexes).

3.4.1 Modèles d'entrepôts de données XML existants

Les entrepôts XML sont nés du besoin d'exploiter des données stockées sous forme de documents XML, qui sont de plus en plus courantes, dans une optique décisionnelle. Plutôt que de passer par un stockage relationnel afin d'exploiter les outils ROLAP existants, avec tous les coûts de migration et de maintenance de deux instances des données que cela implique, il s'agit de conserver les données XML dans leur format natif, d'en proposer une modélisation dimensionnelle et de permettre leur interrogation à l'aide d'un langage de requêtes XML.

Les approches d'entreposage de données XML proposées dans la littérature peuvent être classées en deux grandes familles. Les premières sont orientées besoins et sont plutôt employées lorsque les spécifications de l'entrepôt sont peu susceptibles d'évoluer. Dans le système DAWAX [BB03a], l'entrepôt de données XML est envisagé comme une collection

de vues matérialisées représentées par des documents XML, qui constitue une interface uniforme, via un système de médiation, pour interroger les données. Nassis *et al.* et Rajugan *et al.* exploitent également une approche basée sur des vues définies par l'utilisateur pour proposer des méthodes de conception et de construction d'un espace de stockage XML (*XML repository*) qui représente le contexte d'analyse de l'entrepôt [NRDR05, RCD05]. Zhang *et al.* automatisent ce type de processus en se basant sur des motifs fréquents extraits des requêtes décisionnelles des utilisateurs pour matérialiser l'entrepôt XML [ZWLZ05]. Vrdoljak *et al.* exploitent de leur côté le schéma XML des sources de données pour concevoir des entrepôts Web [VBR03].

Finalement, l'approche X-Warehousing développée au laboratoire ERIC consiste en une méthodologie pour l'entreposage des données complexes [BMCA06a, BMCA06b]. Elle exploite également des schémas XML pour modéliser les besoins d'analyse des utilisateurs. Ils sont ensuite comparés avec ceux des données disponibles pour calculer des cubes XML.

Les autres approches existantes sont plus explicitement basées sur les modèles logiques classiques d'entrepôts de données. Par exemple, Pokorný modélise un schéma en étoile en XML en définissant les hiérarchies des dimensions comme des ensembles logiquement interconnectés de données XML, et les faits comme des éléments XML [Pok01, Pok02]. Park *et al.* proposent un modèle multidimensionnel dans lequel chaque fait est décrit par un document XML spécifique, tandis que les dimensions sont constituées de plusieurs documents XML [PHS05]. Rusu *et al.* travaillent de leur côté sur l'exploitation de requêtes XQuery pour générer les documents XML représentant les faits et les dimensions [RRT04, RRT05]. Finalement, Hümmer *et al.* proposent XCube, un ensemble de métamodèles logiques qui permet de décrire une structure multidimensionnelle (faits et dimensions) dans le but d'intégrer plusieurs entrepôts de données dans un entrepôt global, fédéré ou virtuel [HBH03].

La différence fondamentale entre ces deux familles d'approches réside dans leur manière d'envisager les axes d'analyse (dimensions). La première se focalise sur les faits et l'entrepôt est un ensemble de documents XML représentant ces faits à analyser. Les dimensions sont soit intégrées aux faits, soit ne sont pas matérialisées du tout (dimensions virtuelles). La seconde famille d'approches est plus fidèle à la représentation dimensionnelle classique des entrepôts de données et modélise clairement les dimensions. Néanmoins, toutes ces propositions convergent plus ou moins explicitement vers un modèle d'entrepôt de données XML dérivé du modèle en étoile. Elles diffèrent principalement par le nombre de documents XML utilisés pour représenter les faits et les dimensions.

3.4.2 Spécification de XDW-Bench

Choix de modèle

Nous avons sélectionné les métamodèles XCube [HBH03] pour mettre en œuvre notre banc d'essais, en raison de leur simplicité et de leur flexibilité. Il est en effet possible de les décliner facilement en différents modèles logiques d'entrepôts XML (en étoile, en flocon de neige, en constellation), ce qui va nous permettre de faire évoluer aisément notre

modèle dans le temps. De plus, ils conviennent bien à l'objectif de simplicité de Gray. Enfin, comme les différents modèles d'entrepôts XML proposés jusqu'ici ne sont pas extrêmement différents, ce n'est pas un choix très contraignant et il serait facile de revenir sur ce choix si un standard émergeait dans les années à venir.

Les trois métamodèles de base de XCube sont XCubeSchema, qui décrit le schéma de l'entrepôt de données (métadonnées) ; XCubeDimension, qui explicite la structure hiérarchique des dimensions ; et XCubeFacts, qui décrit les faits à observer.

XCubeSchema est le format pivot qui permet de définir la structure multidimensionnelle de l'entrepôt et d'expliciter les liens entre mesures et dimensions. La structure d'un document XCubeSchema est présentée dans la Figure 3.9. Les lignes pointillées représentent la relation parent-enfant et les astérisques indiquent une cardinalité « zéro à plusieurs ». L'élément `cubeSchema` recense les faits et les dimensions, tandis que l'élément `classSchema` décrit les niveaux hiérarchiques des dimensions. Le sous-élément `rollUp` de `classSchema` permet de définir récursivement la hiérarchie complète. La connexion entre chaque dimension et ses niveaux hiérarchiques passe par la correspondance entre les éléments `cubeSchema/dimension` et `classSchema/classLevel`.

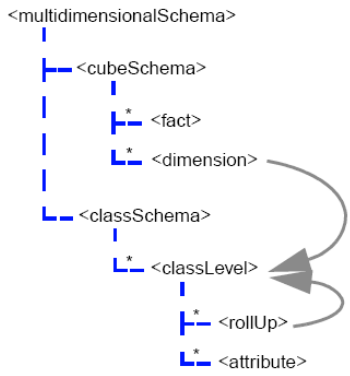

FIG. 3.9 – Métamodèle XCubeSchema

XCubeDimension permet d'instancier les hiérarchies de dimensions spécifiées dans XCubeSchema (Figure 3.10). Les éléments `units` de ce métamodèle servent à référencer les éléments correspondants dans XCubeSchema, tandis que les éléments `classification` définissent, pour chaque niveau hiérarchique d'une dimension, des nœuds (éléments `node`) contenant la valeur des attributs ainsi qu'un lien vers l'élément correspondant au niveau supérieur de la classification.

Finalement, XCubeFact représente les faits (Figure 3.11) sous la forme de cellules (éléments `cell`) contenant les références des dimensions et de leurs nœuds dans la classification, ainsi que les valeurs des mesures correspondantes.

Base de données de XDW-Bench

Schéma. Afin de disposer d'une base reconnue, et comme il n'existe à notre connaissance aucun banc d'essais pour entrepôts de données XML, nous nous sommes appuyés sur le

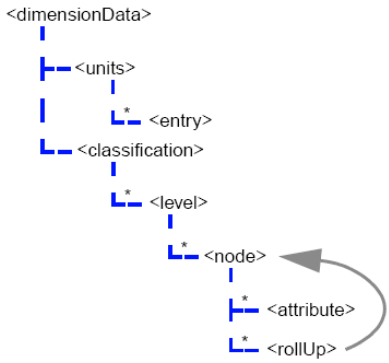

FIG. 3.10 – Métamodèle XCubeDimension

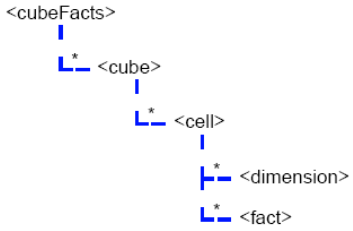

FIG. 3.11 – Métamodèle XCubeFact

banc d'essais décisionnel relationnel TPC-H (Section 3.1.5) pour concevoir le modèle de base de données de XDW-Bench. Nous avons sélectionné TPC-H pour la simplicité de son modèle, en opposition à celui de TPC-DS, dont la structure en constellation est plus difficile à mettre en œuvre et moins précisément documentée.

Nous avons donc simplifié TPC-H et remodélisé ses données dans un schéma dimensionnel en flocon de neige. Ce modèle conceptuel est présenté sous la forme d'un diagramme de classes UML dans la Figure 3.12. Il représente le cas d'école classique de faits de ventes décrits par une dimension « objets vendus », une dimension géographique (lieux de vente) et une dimension temporelle (dates de vente). Chacune des dimensions est hiérarchisée. Ce schéma se traduit directement dans le modèle logique XCubeSchema des Figures 3.13 et 3.14.

Instanciation du schéma. L'extension de cet entrepôt de données a également été tirée en grande partie de TPC-H. À partir du logiciel `dbgen` fourni par le TPC, nous mettons en œuvre un processus d'ETL qui nous permet d'alimenter directement les dimensions de l'entrepôt, comme l'illustre la Figure 3.15. Les faits sont ensuite générés aléatoirement à l'aide de l'algorithme simple de la Figure 3.16, qui fait appel à la notion de densité définie dans le cadre de DWEB (Section 3.3.2). Rappelons qu'une densité de 1 indique que toutes les combinaisons possibles des identifiants des dimensions sont présentes au niveau des faits. Deux exemples de faits sont fournis dans la Figure 3.17.

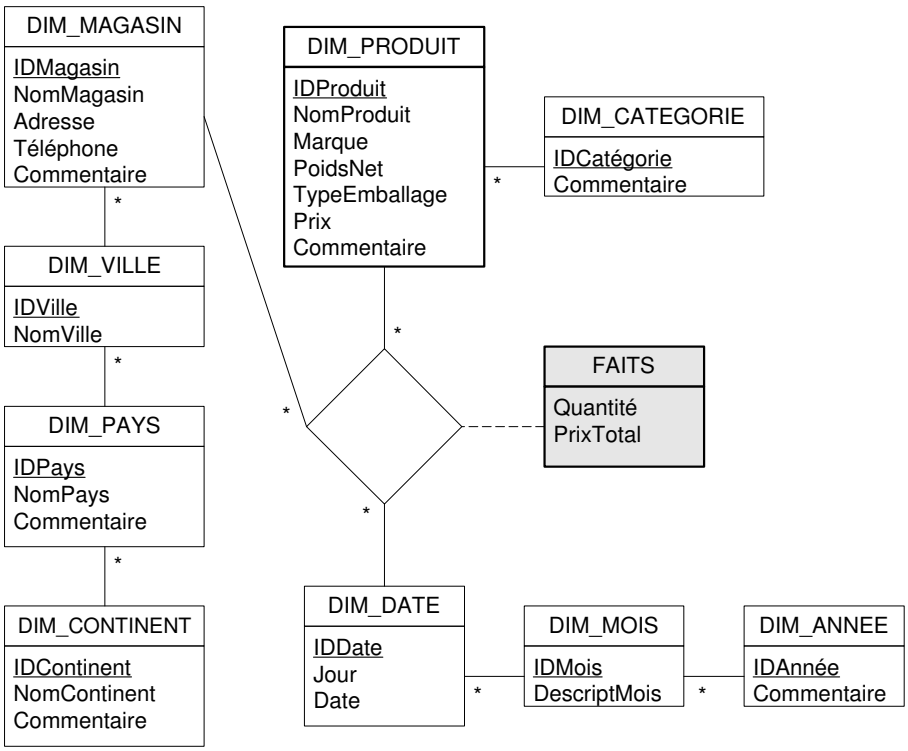

FIG. 3.12 – Schéma conceptuel UML de l'entrepôt XDW-Bench

Paramétrage. Les seuls paramètres définis dans XDW-Bench permettent de contrôler la taille de l'entrepôt de données généré. Comme XDW-Bench est dérivé de TPC-H, la taille T de l'entrepôt dépend directement du facteur d'échelle SF de ce banc d'essais. Elle se calcule de la manière suivante : $T = T_{dim} + T_{faits}$, où T_{dim} est la taille des dimensions, qui ne varie pas lorsque SF est fixé, et T_{faits} est la taille des faits, qui varie en fonction de la densité. Soient $D = \{DIM\_PRODUIT, DIM\_MAGASIN, DIM\_DATE\}$ l'ensemble des dimensions, $|d|_{SF}$ la taille de la dimension d (en nombre de lignes), $taille\_node(d)$ la taille moyenne d'un élément `node` de la dimension d dans le document XCubeDimension et *taille_cell* la taille moyenne d'un élément `cell` dans le document XCubeFact. Alors :

$$T_{dim} = \sum_{d \in D} |d|_{SF} \times taille\_node(d)\,;$$

$$T_{faits} = \prod_{d \in D} |d|_{SF} \times taille\_cell \times densité.$$

Le Tableau 3.4 fournit à titre d'exemple la valeur maximum de T_{faits} ($densité = 1$) pour $SF = 1$, en considérant une période d'observation d'un an (200 jours ouvrables environ) et sachant que $taille\_cell \simeq 220$ octets. Cette taille T_{faits} de pratiquement 82 Go est largement suffisante dans notre contexte, nos expériences ayant montré que certains SGBD natifs XML ne pouvaient pas gérer de documents XCubeFact plus volumineux que quelques dizaines de méga-octets. L'utilisateur n'a donc que la densité à régler en fonction de ses besoins. Le cas échéant, il serait néamoins facile d'augmenter le facteur d'échelle SF jusqu'au maximum autorisé dans TPC-H (10 000) pour proposer des entrepôts XML plus volumineux.

```
<multidimensionalSchema version="0.5">
    <cubeSchema id="XDW-Bench"> <!-- Faits et dimensions -->
       <fact id="Quantité" />
       <fact id="PrixTotal" />
       <dimension id="Produit" granularity="DIM_PRODUIT" />
       <dimension id="Magasin" granularity="DIM_MAGASIN" />
       <dimension id="Temps" granularity="DIM_DATE" />
    </cubeSchema>
    <classSchema>
       <!-- Hiérarchie de la dimension Produit -->
       <classLevel id="DIM_PRODUIT">
          <attribute id="IDProduit" />
          <attribute id="NomProduit" />
          <attribute id="Marque" />
          <attribute id="PoidsNet" />
          <attribute id="TypeEmballage" />
          <attribute id="Prix" />
          <attribute id="Commentaire" />
          <rollUp toLevel="DIM_CATEGORIE" />
       </classLevel>
       <classLevel id="DIM_CATEGORIE">
          <attribute id="IDCatégorie" />
          <attribute id="Commentaire" />
       </classLevel>
       <!-- Hiérarchie de la dimension Magasin -->
       <classLevel id="DIM_MAGASIN">
          <attribute id="IDMagasin" />
          <attribute id="NomMagasin" />
          <attribute id="Adresse" />
          <attribute id="Téléphone" />
          <attribute id="Commentaire" />
          <rollUp toLevel="DIM_VILLE" />
       </classLevel>
       <classLevel id="DIM_VILLE">
          <attribute id="IDVille" />
          <attribute id="NomVille" />
          <rollUp toLevel="DIM_PAYS" />
       </classLevel>
<!-- ../.. -->
```

FIG. 3.13 – Schéma logique XCubeSchema de l'entrepôt XDW-Bench – Partie 1

```
<!-- ../.. -->
      <classLevel id="DIM_PAYS">
         <attribute id="IDPays" />
         <attribute id="NomPays" />
         <attribute id="Commentaire" />
         <rollUp toLevel ="DIM_CONTINENT" />
      </classLevel>
      <classLevel id ="DIM_CONTINENT">
         <attribute id="IDContinent" />
         <attribute id="NomContinent" />
         <attribute id="Commentaire" />
      </classLevel>
      <!-- Hiérarchie de la dimension Temps -->
      <classLevel id="DIM_DATE">
         <attribute id="IDDate" />
         <attribute id="Jour" />
         <attribute id="Date" />
         <rollUp toLevel="DIM_MOIS" />
      </classLevel>
      <classLevel id="DIM_MOIS">
         <attribute id="IDMois" />
         <attribute id="DescriptMois" />
         <rollUp toLevel="DIM_ANNEE" />
      </classLevel>
      <classLevel id="DIM_ANNEE">
         <attribute id="IDAnnee" />
         <attribute id="Commentaire" />
      </classLevel>
   </classSchema>
</multidimensionalSchema>
```

FIG. 3.14 – Schéma logique XCubeSchema de l'entrepôt XDW-Bench – Partie 2

```
<dimensionData version="0.5">
  <classification>
   <level id="DIM_CATEGORIE">
    <node id="ECONOMY ANODIZED">
     <attribute id="Commentaire" value="ironic courts gro" />
    </node>
    <node id="PROMO BURNISHED">
     <attribute id="Commentaire" value="quickly bo" />
    </node>
   </level>
   <level id="DIM_PRODUIT">
    <node id="1">
     <attribute id="NomProduit" value="goldenrod lace" />
     <attribute id="Marque" value="Brand#13" />
     <attribute id="PoidsNet" value="7" />
     <attribute id="TypeEmballage" value="JUMBO PKG" />
     <attribute id="Prix" value="901.00" />
     <attribute id="Commentaire" value="final deposits s" />
     <rollUp toNode="PROMO BURNISHED" level="DIM_CATEGORIE" />
    </node>
    <node id="10">
     <attribute id="NomProduit" value="floral moccasin" />
     <attribute id="Marque" value="Brand#54" />
     <attribute id="PoidsNet" value="44" />
     <attribute id="TypeEmballage" value="LG CAN" />
     <attribute id="Prix" value="910.01" />
     <attribute id="Commentaire" value="bold, ironic" />
     <rollUp toNode="ECONOMY ANODIZED" level="DIM_CATEGORIE" />
    </node>
   </level>
  </classification>
</dimensionData>
```

FIG. 3.15 – Fragment du document XCubeDimension de l'entrepôt XDW-Bench

```
For each p ∈ DIM_PRODUIT do
    For each m ∈ DIM_MAGASIN do
       For each d ∈ DIM_DATE do
          If Random(0, 1) ≤ Densité then
             Quantité = Random(1, 10000)
             PrixTotal = Quantité × p.Prix
             Création_Fait(p, m, d, Quantité, PrixTotal)
          End if
       End for
    End for
End for
```

FIG. 3.16 – Algorithme de génération des faits de XDW-Bench

```
<cubeFacts version="0.5">
    <cube id="XDW-Bench">
       <cell>
          <dimension id="Produit" node="2" />
          <dimension id="Magasin" node="9" />
          <dimension id="Temps" node="124" />
          <fact id="Quantité" value="9" />
          <fact id="PrixTotal" value="1800" />
       </cell>
       <cell>
          <dimension id="Produit" node="2" />
          <dimension id="Magasin" node="11 "/>
          <dimension id="Temps" node="96" />
          <fact id="Quantité" value="200" />
          <fact id="PrixTotal" value="40000" />
       </cell>
    </cube>
</cubeFacts>
```

FIG. 3.17 – Fragment du document XCubeFact d'un entrepôt XDW-Bench

|DIM_PRODUIT|	|DIM_MAGASIN|	|DIM_DATE|	$Tmax_{faits}$ (Go)
200 000	10 000	200	81956

TAB. 3.4 – Taille du document XCubeFact de XDW-Bench pour $SF = 1$

Charge de XDW-Bench

Le langage XQuery s'affirme en tant que standard d'interrogation de données XML et permet de formuler des requêtes complexes. C'est donc le choix le plus naturel pour formuler une charge décisionnelle sur un entrepôt de données XML. Il présente cependant des limitations, car il n'inclut pas les opérations de regroupement multiples du type « moyenne des quantités par produit et par magasin » et ne permet donc pas *a fortiori* la formulation de requêtes OLAP explicites. La charge que nous proposons dans cette section n'est donc pas exprimée sous forme de requêtes XQuery, mais sous forme de spécifications que l'utilisateur doit mettre en œuvre sur le système cible qu'il souhaite tester. En général, les SGBD natifs XML proposent des API (*Application Programming Interfaces*) qui permettent d'implémenter aisément ce type de fonctionnalités.

Nous nous sommes de nouveau inspirés de TPC-H pour concevoir la charge de XDW-Bench, qui est actuellement constituée de 18 requêtes d'interrogation étiquetées $Q1$ à $Q18$. Ces requêtes exploitent le schéma multidimensionnel de XDW-Bench à l'aide de jointures, d'opérations d'agrégation, d'appels à des fonctions, de regroupements et de sous-requêtes. Leur spécification est fournie dans le Tableau 3.5. Nous préconisons d'exploiter ces requêtes en appliquant le protocole d'exécution de TPC-H (Section 3.1.5).

Remarque : Les mises à jour étant diversement prises en charge par les SGBD natifs XML et XQuery n'étant actuellement pas doté d'une syntaxe pour les effectuer, nous n'avons pour l'instant pas inclus de fonction de rafraîchissement dans XDW-Bench.

Requête	**Spécification**
$Q1$	Appel à une fonction dateOfIdentifier(id) qui retourne une structure `<date>` contenant des informations relatives à l'identificateur IDDate de la dimension DIM_DATE
$Q2$	Nombre de ventes par produit, par mois et par jour
$Q3$	Nombre de ventes par produit et par magasin
$Q4$	Nombre de ventes d'un produit donné, par continent, par pays et par ville
$Q5$	Total des quantités vendues, par produit et par ville
$Q6$	Somme des prix totaux, par produit et par ville
$Q7$	Valeurs des ventes (quantité et prix total) par ville
$Q8$	Moyenne des quantités vendues par magasin et par produit
$Q9$	Somme des prix totaux par magasin et par produit
$Q10$	Somme des prix totaux par année et par catégorie de produit
$Q11$	Somme des prix totaux par catégorie de produit et par magasin
$Q12$	Somme des prix totaux par magasin et par catégorie de produit
$Q13$	Total des totaux calculés par $Q11$
$Q14$	Total des totaux calculés par $Q12$
$Q15$	Somme totale des prix totaux par magasin et par catégorie de produit
$Q16$	Cinq produits les moins vendus en quantité, par ville
$Q17$	Cinq produits les plus vendus en quantité, par continent
$Q18$	Meilleures ventes (en prix total) par magasin

TAB. 3.5 – Spécification de la charge décisionnelle de XDW-Bench

3.4.3 Bilan

XDW-Bench est une première proposition de banc d'essais décisionnel XML. Le contexte non stabilisé des entrepôts de données XML nous a amenés à privilégier le critère de simplicité de Gray au cours de sa conception et de son développement, de manière à pouvoir aisément modifier notre outil en cas d'évolution de la recherche dans ce domaine ou d'émergence d'un standard. XDW-Bench est donc actuellement relativement sommaire, mais il est toujours en développement. De nombreuses améliorations sont en effet prévues.

Tout d'abord, le schéma de base de l'entrepôt va certainement être amené à évoluer vers une version plus développée en termes de dimensions. Nous envisageons également de proposer, de manière optionnelle, une architecture en constellation contenant plusieurs documents XCubeFact. La charge devra bien sûr évoluer en conséquence pour exploiter ces nouveautés.

Malgré les limitations du langage XQuery pour formuler des requêtes décisionnelles et de mise à jour, il va devenir le standard d'interrogation des données XML. Il serait donc judicieux de formuler les spécifications de la charge dans ce langage en comptant sur le développement d'extensions déjà identifiées, notamment en ce qui concerne les requêtes

de regroupement et les opérateurs OLAP [PAKJ+02, BC04, BCC+04, DPX04, BCC+05]. Il sera également important d'inclure le rafraîchissement de l'entrepôt XML dans notre charge et de concevoir un protocole d'exécution de XDW-Bench qui soit spécifique aux systèmes XML.

Nous travaillons également à une deuxième version de XDW-Bench qui intègre des données complexes. Plus précisément, nous adaptons les données médicales utilisées pour illustrer la méthodologie X-Warehousing [BMCA06a, BMCA06b] (ensemble de dossiers patients incluant des mammographies) au contexte du banc d'essais. Une charge décisionnelle adaptée visant à l'aide au diagnostic est également développée.

Pour terminer, XDW-Bench a démontré, bien qu'il ne soit pas encore finalisé, son utilité pour valider de façon expérimentale une stratégie d'indexation des entrepôts de données XML que nous avons conçue [MAD06b], ainsi que notre stratégie de sélection de vues matérialisées dans le contexte XML (Section 2.3.2).

3.5 Conclusion

Nous avons présenté dans ce chapitre trois outils d'évaluation des performances pour les bases de données (DEF), les entrepôts de données relationnels (DWEB) et les entrepôts de données XML (XDW-Bench). À notre connaissance, peu d'autres bancs d'essais partagent leurs caractéristiques, notamment dans le domaine des entrepôts de données. De plus, les bancs d'essais les plus couramment publiés visent à comparer les performances de différents systèmes pour permettre à leurs utilisateurs de les sélectionner en disposant d'informations quantitatives. Par contraste, nous proposons des outils plutôt destinés aux concepteurs de systèmes ou d'entrepôts de données, qui leur permettent de comparer différentes architectures ou solutions logicielles.

Nous avons donc concentré nos efforts sur la pertinence et l'adaptabilité de nos bancs d'essais. Même XDW-Bench, qui est le plus simple, présente des caractéristiques décisionnelles inédites dans les bancs d'essais XML existants. Bien que cette approche passe le plus souvent par la mise en œuvre d'un paramétrage fin, nous avons eu le souci de ne pas trop sacrifier à l'impératif de simplicité en proposant des outils de configuration permettant d'assister l'utilisateur, notamment dans DWEB.

Par ailleurs, nous avons systématiquement publié les spécifications complètes de nos bancs d'essais, de manière à ce qu'ils puissent être utilisés le plus facilement possible. Leur code est également libre et modifiable par quiconque le souhaiterait. Hormis pour XDW-Bench qui n'est pas encore finalisé, il est disponible gratuitement en ligne[2].

Pour terminer, la principale évolution prévue pour ces travaux est l'introduction de données complexes dans nos bancs d'essais. La variété des données complexes étant immense, nous pensons partir de jeux d'essais concrets développés au laboratoire ERIC pour concevoir des outils les plus génériques possibles, mais sans pour autant prétendre à l'uni-

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

versalité. Nous nous basons particulièrement sur trois applications. Les deux premières présentent des similarités. Elles relèvent toutes deux du domaine de la médecine et nécessitent le stockage de données variées : vues relationnelles (dossiers de patients), textes (antécédents médicaux, diagnostics antérieurs), images médicales (radiographies, mammographies), etc. Cependant, nous développons pour l'une d'entre elles un entrepôt de données relationnel [DO06, DO07] (dans le cadre du projet MAP — Médecine d'Anticipation Personnalisée — mené en partenariat avec un médecin du sport, la Région Rhône-Alpes et l'incubateur d'entreprises CREALYS) et pour l'autre un entrepôt de données XML [BMCA06a, BMCA06b], ce qui nous permettra de proposer des architectures fondamentalement différentes dans nos bancs d'essais.

La troisième application exploite les données du projet CLAPI [ABBD03] (qui a été mené en collaboration avec le laboratoire de linguistique ICAR commun à l'ENS Lettre et Sciences Humaines de Lyon et l'Université Lyon 2), et plus précisément les enregistrements audio et vidéo d'interactions et leurs transcriptions codifiées stockées au format XML. Ces différentes applications devraient nous permettre de concevoir une famille de bancs d'essais pour entrepôts de données complexes, tant au niveau de la base de données que de la charge décisionnelle qui lui est appliquée.

Chapitre 4

Conclusion et perspectives

4.1 Bilan général

Les travaux présentés dans ce mémoire ont été menés au sein du laboratoire ERIC, dont l'activité de recherche est principalement dédiée à l'Extraction de Connaissances à partir des Données (ECD). Ils relèvent plus particulièrement du pôle « bases de données décisionnelles », qui a été créé à ERIC fin 2000 avec l'objectif de travailler sur le volet bases de données du processus d'ECD. Cela inclut principalement le domaine des entrepôts de données et la manière dont leurs principes et technologies peuvent enrichir l'ECD et plus particulièrement la phase de fouille de données, et *vice versa*.

Notre recherche s'appuie donc sur deux domaines clairement identifiés, bien que voisins et en intersection : les bases de données et la fouille de données. Dans ce cadre, nous avons eu la volonté d'inscrire nos travaux dans la démarche préconisée par le laboratoire ERIC : recherche théorique, développement de prototypes logiciels et applications à des cas concrets.

Dans le domaine des entrepôts de données, nous nous sommes plus particulièrement intéressés au problème de la performance. En nous basant sur l'intuition que des connaissances sur les données entreposées et leur usage peuvent contribuer à optimiser leur accès, nous avons proposé des solutions d'indexation et de matérialisation de vues qui font appel à des techniques de fouille de données. Cette originalité permet de rendre le passage à l'échelle plus aisé dans ce type d'approches et de tendre vers l'automatisation des tâches d'optimisation de performance, qui constituent une part importante du travail d'administration d'un entrepôt.

Par ailleurs, le développement de ces techniques et outils inclut une phase de validation qui est le plus souvent expérimentale. Face à l'inadéquation à nos besoins des outils d'évaluation de performance existants dans le domaine des entrepôts de données, nous avons conçu et développé des bancs d'essais, que nous avons voulus les plus génériques possibles, afin de permettre la comparaison de différentes architectures ou solutions logicielles. Ces propositions sont inédites, notamment XDW-Bench qui est actuellement à notre connaissance le seul banc d'essais pour entrepôts de données XML.

Ces deux familles de contributions de recherche ont donné lieu au développement de plusieurs prototypes logiciels qui sont librement accessibles en ligne. Elles ont également

été appliquées (au moins en partie) dans le cadre de partenariats, notamment avec le laboratoire de linguistique ICAR (ACI Nomex-CLAPI) et le Docteur Ferret, médecin du sport (projet MAP).

4.2 Perspectives de recherche

Le laboratoire ERIC est un acteur de la recherche dans le domaine des données complexes depuis l'émergence de ce nouveau champ, il y a quelques années. Il travaille notamment à la conception de solutions pour entreposer et exploiter des données complexes, par l'analyse en ligne (OLAP) et/ou la fouille de données. Les travaux présentés dans ce mémoire convergent vers l'optimisation et l'évaluation de performance des entrepôts de données complexes, plus particulièrement les techniques et outils que nous proposons dans le contexte XML, qui est le formalisme que nous utilisons pour représenter les données complexes. Cependant, il reste, au-delà du formalisme, à réellement prendre en compte la complexité des données dans nos approches. Les grandes perspectives qui s'offrent à nous dans ce cadre sont de deux ordres.

Premièrement, de part sa vocation initiale pour l'échange de données semi-structurées, le langage XML offre une grande flexibilité de représentation des données complexes et des possibilités très intéressantes de structuration, de modélisation, de stockage et d'interrogation de ces données. Il se confirme donc comme le choix pivot des solutions d'entreposage de données que nous proposons.

Les SGBD natifs XML actuels présentent à première vue des performances nettement inférieures à celles des SGBD relationnels. Cependant, stocker des données XML, notamment si elles présentent des structures complexes et irrégulières, dans une base relationnelle n'est pas trivial. Leur interrogation nécessite également des opérations coûteuses pour traduire les requêtes du langage XQuery en SQL, puis les résultats obtenus sous forme de relations en documents XML. Dans ce contexte, stocker des données XML et *a fortiori* des données complexes représentées en XML présente l'avantage de la simplicité. De plus, des tests préliminaires que nous avons menés montrent qu'en mettant en œuvre des techniques d'optimisation adéquates, les SGBD natifs XML peuvent se révéler plus efficaces que les SGBD relationnels pour ce type d'applications. Il reste néanmoins d'importants efforts de recherche à fournir pour optimiser le traitement des données complexes dans les SGBD natifs XML.

Par ailleurs, ces systèmes présentent également des limitations au niveau de la volumétrie : il leur est souvent difficile de gérer et de permettre l'interrogation de documents XML de grande taille. Or, dans le contexte de l'entreposage de données, les faits sont typiquement disponibles en très grand nombre pour permettre les analyses les plus exhaustives possibles. Les partitionner et les répartir sur différents sites pourrait permettre de contourner cette limitation. Diverses pistes s'ouvrent alors pour exploiter un tel entrepôt réparti.

L'une d'elle est une variante de l'entreposage de données virtuel, qui consiste à l'origine

à baser un système décisionnel directement sur les données de production, ce qui perturbe leur traitement. Ce concept refait néanmoins surface par le biais de l'EII (*Enterprise Information Integration*) [Eck03]. Par ailleurs, il ne s'agirait pas ici d'exploiter les données de production, mais des données redondantes à vocation décisionnelle stockées de manière répartie. Le principe d'une telle approche serait de permettre le calcul de cubes de données à la volée, en fonction des besoins des utilisateurs.

Une autre alternative, qui n'est d'ailleurs pas nécessairement concurrente, serait de répartir l'entrepôt de données complexes XML sur une grille de calcul pour bénéficier des possibilités de répartition intelligente des données et de traitements parallèles de ce type d'architecture [WMT05a, WMT05b]. Cela permettrait de plus à des utilisateurs ayant des besoins spécifiques en termes de données, d'axes d'analyse et de niveaux de détails des résultats, de se connecter à n'importe quel point d'accès à la grille.

Finalement, l'aspect standard du Web du langage XML pourrait également être mis à profit dans ce contexte. Les services Web utilisent en effet des standards et des protocoles de transport basés sur XML pour échanger des données [TP02, FF03]. Faire appel à ces technologies pourrait nous permettre de concevoir un environnement d'entreposage de données complexes « tout XML ». Ces travaux font l'objet de la thèse de doctorat de Hadj Mahboubi, initiée en 2005.

Le deuxième grand point susceptible de nous aider dans notre tâche d'optimisation des entrepôts de données complexes est l'extraction de connaissances de ces données elles-mêmes. Typiquement, elles sont exploitées sous la forme de caractéristiques de bas niveau (histogrammes de couleur et de texture pour des images, par exemple) qui ne sont pas toujours suffisantes. Disposer d'informations sémantiques (identification des objets d'une image ou des relations qui les lient, par exemple) automatiquement extraites ou fournies par un expert serait beaucoup plus pertinent pour l'indexation, entre autres. Dans le contexte des entrepôts de données, ces informations peuvent être représentées sous la forme de métadonnées ou de connaissances liées au domaine d'application, par exemple représentées dans des ontologies.

Ici encore, XML s'impose comme un format privilégié pour structurer ces informations sémantiques et faciliter la communication entre les différentes composantes de l'entrepôt [DBRA05], grâce aux technologies développées dans le cadre du Web sémantique, comme RDF-S (*Resource Description Framework Schema* [BG04]) ou OWL (*Web Ontology Language* [MH04]). Le développement de services Web sémantiques permettrait d'intégrer ces éléments dans notre plateforme d'entreposage XML de données complexes.

Pour terminer, nous allons également continuer d'entretenir et développer nos partenariats pour réaliser des applications, qui deviennent souvent elles-mêmes le terreau de nouvelles recherches. C'est en ce sens que nous avons déposé auprès de l'Agence Nationale de la Recherche un projet « jeunes chercheuses et jeunes chercheurs » conjoint avec le laboratoire ICAR sur le thème de l'analyse de la grammaire et du lexique des tours de parole en interaction grâce à un entrepôt de données complexes XML (NOFIXware).

Bibliographie

[ABBD03] K. Aouiche, F. Bentayeb, O. Boussaïd et J. Darmont : Conception informatique d'une base de données multimédia de corpus linguistiques oraux : l'exemple de CLAPI 2. *36ème Colloque International de la Societas Linguistica Europaea, Lyon, France*, pages 11–12, Septembre 2003.

[ABM+90] T.L. Anderson, A.J. Berre, M. Mallison, H.H. Porter et B. Schneider : The HyperModel Benchmark. *International Conference on Extending Database Technology (EDBT 90), Venice, Italy*, volume 416 de *LNCS*, pages 317–331, 1990.

[ACK+04] S. Agrawal, S. Chaudhuri, L. Kollár, A.P. Marathe, V.R. Narasayya et M. Syamala : Database Tuning Advisor for Microsoft SQL Server 2005. *30th International Conference on Very Large Data Bases (VLDB 04), Toronto, Canada*, pages 1110–1121, August-September 2004.

[ACN00] S. Agrawal, S. Chaudhuri et V.R. Narasayya : Automated Selection of Materialized Views and Indexes in SQL Databases. *26th International Conference on Very Large Data Bases (VLDB 00), Cairo, Egypt*, pages 496–505, 2000.

[ACN01] S. Agrawal, S. Chaudhuri et V.R. Narasayya : Materialized View and Index Selection Tool for Microsoft SQL Server 2000. *ACM SIGMOD International Conference on Management of Data (SIGMOD 01), Santa Barbara, USA*, page 608, 2001.

[AD07] K. Aouiche et J. Darmont : Index and Materialized View Selection in Data Warehouses. *Encyclopedia of Database Technologies and Applications, Second Edition*. Idea Group Publishing, 2007.

[ADB04] K. Aouiche, J. Darmont et O. Boussaïd : Sélection automatique d'index dans les entrepôts de données. *1er atelier Fouille de Données Complexes dans un processus d'extraction des connaissances, EGC 04, Clermont-Ferrand*, pages 91–102, Janvier 2004.

[ADBB05a] K. Aouiche, J. Darmont, O. Boussaïd et F. Bentayeb : Auto-administration des entrepôts de données complexes. *Revue des Nouvelles Technologies de l'Information*, E-4:47–70, Septembre 2005.

[ADBB05b] K. Aouiche, J. Darmont, O. Boussaïd et F. Bentayeb : Automatic selection of bitmap join indexes in data warehouses. *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, volume 3589 de *LNCS*, pages 64–73, August 2005.

[ADG03a] K. Aouiche, J. Darmont et L. Gruenwald : Extraction de motifs fréquents pour l'auto-administration des bases de données. *Journées francophones d'Extraction et de Gestion des Connaissances (EGC 03), Lyon*, volume 17(1-2-3) de *RSTI*, page 547, Janvier 2003.

[ADG03b] K. Aouiche, J. Darmont et L. Gruenwald : Frequent itemsets mining for database auto-administration. *7th International Database Engineering and Application Symposium (IDEAS 03), Hong Kong, China*, pages 98–103, July 2003.

[ADG03c] K. Aouiche, J. Darmont et L. Gruenwald : Vers l'auto-administration des entrepôts de données. *XXXVèmes Journées de Statistique, Session spéciale Entreposage et Fouille de Données, Lyon*, pages 105–108, Juin 2003.

[ADG03d] K. Aouiche, J. Darmont et L. Gruenwald : Vers l'auto-administration des entrepôts de données. *Revue des Nouvelles Technologies de l'Information*, (1):1–12, 2003.

[AJD06] K. Aouiche, P. Jouve et J. Darmont : Clustering-Based Materialized View Selection in Data Warehouses. *10th East-European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, volume 4152 de *LNCS*, pages 81–95, September 2006.

[AMM05] L. Afanasiev, I. Manolescu et P. Michiels : MemBeR: A Micro-benchmark Repository for XQuery. *3rd International XML Database Symposium (XSym 05), Trondheim, Norway*, volume 3671 de *LNCS*, pages 144–161, August 2005.

[ANZ01] A. Aboulnaga, J. Naughton et C. Zhang : Generating Synthetic Complex-Structured XML Data. *International Workshop on the Web and Databases (WebDB 01), Santa Barbara, USA*, pages 79–84, 2001.

[Aou02] K. Aouiche : *Techniques d'ECD pour l'auto-administration des bases de données*. Mémoire de master recherche, Université Lumière Lyon 2, 2002.

[Aou05] K. Aouiche : *Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données*. Thèse de doctorat, Université Lumière Lyon 2, 2005.

[AS94] R. Agrawal et R. Srikant : Fast algorithms for mining association rules. *20th International Conference on Very Large Data Bases (VLDB 94), Santiago de Chile, Chile*, pages 487–499, 1994.

[Bal93] C. Ballinger : TPC-D: Benchmarking for Decision Support. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, 1993.

[BB03a] X. Baril et Z. Bellahsène : Designing and Managing an XML Warehouse, pages 455–473. *XML Data Management: Native XML and XML-enabled Database Systems*. Addison Wesley, 2003.

[BB03b] X. Baril et Z. Bellahsène : Selection of Materialized Views: a Cost-Based Approach. *15th International Conference on Advanced Information Systems Engineering (CAiSE 03), Klagenfurt, Austria*, pages 665–680, 2003.

[BBD03] O. Boussaïd, F. Bentayeb et J. Darmont : A Multi-Agent System-Based ETL Approach for Complex Data. *10th ISPE International Conference on Concurrent Engineering: Research and Applications (CE 03), Madeira, Portugal*, pages 49–52, July 2003.

[BBDR03] O. Boussaïd, F. Bentayeb, J. Darmont et S. Rabaseda : Vers l'entreposage des données complexes : structuration, intégration et analyse. *Ingénierie des Systèmes d'Information (RSTI série ISI)*, 8(5-6):79–107, 2003.

[BC04] V. Borkar et M. Carey : Extending XQuery for Grouping, Duplicate Elimination, and Outer Joins. *XML 2004, Washington DC, USA*, November 2004.

[BC06] N. Bruno et S. Chaudhuri : Physical Design Refinement: The "Merge-Reduce" Approach. *10th International Conference on Extending Database Technology (EDBT 06), Munich, Germany*, volume 3896 de *LNCS*, pages 386–404, March 2006.

[BCC+04] K.S. Beyer, R. Cochrane, L.S. Colby, F. Özcan et H. Pirahesh : XQuery for Analytics: Challenges and Requirements. *1st International Workshop on XQuery Implementation, Experience and Perspectives (<XIME-P/> 04), Paris, France*, pages 3–8, 2004.

[BCC+05] K.S. Beyer, D.D. Chamberlin, L.S. Colby, F. Özcan, H. Pirahesh et Y. Xu : Extending XQuery for Analytics. *2005 ACM SIGMOD International Conference on Management of Data (SIGMOD 05), Baltimore, USA*, pages 503–514, 2005.

[BCF+05] S. Boag, D. Chamberlin, M.F. Fernández, D. Florescu, J. Robie et J. Siméon : *XQuery 1.0: An XML Query Language*. World Wide Web consortium, April 2005.

[BCH99] D. Belanger, K. Church et A. Hume : Virtual data warehousing, data publishing, and call detail. *International Workshop on Databases in Telecommunications, Edinburgh, Scotland*, volume 1819 de *LNCS*, pages 106–117, 1999.

[BD02] F. Bentayeb et J. Darmont : Decision tree modeling with relational views. *XIIIth International Symposium on Methodologies for Intelligent Systems (ISMIS 02), Lyon, France*, volume 2366 de *LNAI*, pages 423–431, June 2002.

[BDBL07] O. Boussaïd, J. Darmont, F. Bentayeb et S. Loudcher : Warehousing complex data from the Web. *International Journal of Web Engineering and Technology*, 2007.

[BDFU07] F. Bentayeb, J. Darmont, C. Favre et C. Udréa : Efficient On-Line Mining of Large Databases. *International Journal of Business Information Systems*, 4(2), 2007.

[BDT83] D. Bitton, D.J. DeWitt et C. Turbyfill : Benchmarking Database Systems A Systematic Approach. *9th International Conference on Very Large Data Bases (VLDB 83), Florence, Italy*, pages 8–19, 1983.

[BDU04] F. Bentayeb, J. Darmont et C. Udréa : Efficient Integration of Data Mining Techniques in Database Management Systems. *8th International Database Engineering and Applications Symposium (IDEAS 04), Coimbra, Portugal*, pages 59–67, July 2004.

[Bel00] L. Bellatreche : *Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données*. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, décembre 2000.

[BG04] D. Brickley et R.V. Guha : *RDF Vocabulary Description Language 1.0: RDF Schema*. World Wide Web consortium, February 2004.

[Bha96] R. Bhashyam : TCP-D: The Challenges, Issues and Results. *SIGMOD Record*, 25(4):89–93, December 1996.

[BKM02] L. Bellatreche, K. Karlapalem et M. Mohania : Some issues in design of data warehousing systems, pages 22–76. *Data warehousing and web engineering*. IRM Press, 2002.

[BKM05] F. Bouali, L. Khan et F. Masséglia, éditeurs. *Sixth International Workshop on Multimedia Data Mining (MDM/KDD 05), Chicago, USA*, August 2005.

[BKS00] L. Bellatreche, K. Karlapalem et M. Schneider : On efficient storage space distribution among materialized views and indices in data warehousing environments. *9th International Conference on Information and Knowledge Management (CIKM 00), McLean, USA*, pages 397–404, 2000.

[BLL+02] S. Bressan, M.L. Lee, Y.G. Li, Z. Lacroix et U. Nambiar : The XOO7 benchmark. *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT, Hong Kong, China*, volume 2590 de *LNCS*, pages 146–147, 2002.

[BMC00] BMC Software : Performance Management of a Data Warehouse. *http://www.bmc.com*, 2000.

[BMCA06a] O. Boussaïd, R. Ben Messaoud, R. Choquet et S. Anthoard : Conception et construction d'entrepôts XML. *2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 06), Versailles*, volume B-2 de *RNTI*, pages 3–22, Juin 2006.

[BMCA06b] O. Boussaïd, R. Ben Messaoud, R. Choquet et S. Anthoard : X-Warehousing: an XML-Based Approach for Warehousing Complex Data. *10th East-European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, volume 4152 de *LNCS*, pages 39–54, September 2006.

[BMKL02] D. Barbosa, A.O. Mendelzon, J. Keenleyside et K.A Lyons : ToXgene: An Extensible Template based Data Generator for XML. *International Workshop on the Web and Databases (WebDB 02), Madison, USA*, pages 49–54, 2002.

[BPSM+04] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler et F. Yergeau : *Extensible Markup Language (XML) 1.0 (Third Edition)*. World Wide Web consortium, February 2004.

[BPT97] E. Baralis, S. Paraboschi et E. Teniente : Materialized Views Selection in a Multidimensional Database. *23rd International Conference on Very Large Data Bases (VLDB 97), Athens, Greece*, pages 156–165, 1997.

[BR01] T. Böhme et E. Rahm : XMach-1: A Benchmark for XML Data Management. *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 01), Oldenburg, Germany*, pages 264–273, 2001.

[BS96] F. Bullat et M. Schneider : Dynamic clustering in object databases exploiting effective use of relationships between objects. *10th European Conference on Object-Oriented Programming (ECOOP 96), Linz, Austria*, volume 1098 de *LNCS*, pages 344–365, July 1996.

[BT06] O. Boussaïd et B. Trousse, éditeurs. *Troisième atelier sur la fouille de données complexes dans un processus d'extraction des connaissances (EGC 06), Lille*, Janvier 2006.

[BTBD06] O. Boussaïd, A. Tanasescu, F. Bentayeb et J. Darmont : Integration and Dimensional Modelling Approaches for Complex Data Warehousing. *Journal of Global Optimization*, 2006.

[Car75] A.F. Cardenas : Analysis and performance of inverted data base structures. *Communication of the ACM*, 18(5):253–263, 1975.

[Cat91] R.G.G. Cattell : An Engineering Database Benchmark, pages 247–281. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, première édition, 1991.

[CBC93] S. Choenni, H.M. Blanken et T. Chang : Index Selection in Relational Databases. *5th International Conference on Computing and Information (ICCI 93), Ontario, Canada*, pages 491–496, 1993.

[CD97] S. Chaudhuri et U. Dayal : Data Warehousing and OLAP for Decision Support. *ACM SIGMOD International Conference on Management of Data (SIGMOD 97), Tucson, USA*, pages 507–508, 1997.

[CDF+94] M.J. Carey, D.J. DeWitt, M.J. Franklin, N.E. Hall, M.McAuliffe, J.F. Naughton, D.T. Schuh, M.H. Solomon, C.K. Tan, O. Tsatalos, S. White et M.J. Zwilling : Shoring up persistent applications. *1994 ACM SIGMOD International Conference on Management of Data (SIGMOD 94), Minneapolis, USA*, pages 383–394, May 1994.

[CDN93] M. Carey, D. DeWitt et J. Naughton : The OO7 benchmark. *ACM SIGMOD International Conference on Management of Data (SIGMOD 93), Washington, USA*, pages 12–21, 1993.

[CDN+97] M. Carey, D. DeWitt, J. Naughton, M. Asgarian, P. Brown, J. Gehrke et D. Shah : The BUCKY Object-Relational Benchmark. *1997 ACM SIGMOD International Conference on Management of Data, Tucson, USA*, pages 135–146, May 1997.

[CFLS91] M.J. Carey, M.J. Franklin, M. Livny et E.J. Shekita : Data Caching Tradeoffs in Client-Server DBMS Architectures. *1991 ACM SIGMOD International Conference on Management of Data, Denver, USA*, pages 357–366, 1991.

[CGN02] S. Chaudhuri, A.K. Gupta et V.R. Narasayya : Compressing SQL workloads. *2002 ACM SIGMOD International Conference on Management of Data (SIGMOD 02), Madison, Wisconsin*, pages 488–499, June 2002.

[CM99] S. Chaudhuri et R. Motwani : On sampling and relational operators. *IEEE Data Engineering Bulletin*, 22(4):41–46, 1999.

[CN97] S. Chaudhuri et V.R. Narasayya : An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. *23rd international Conference on Very Large Data Bases (VLDB 97), Santiago de Chile, Chile*, pages 146–155, 1997.

[CN98] S. Chaudhuri et V.R. Narasayya : AutoAdmin 'What-if' Index Analysis Utility. *ACM SIGMOD International Conference on Management of Data (SIGMOD 98), Seattle, USA*, pages 367–378, 1998.

[Com78] D. Comer : The difficulty of optimum index selection. *ACM Transactions on Database Systems*, 3(4):440–445, 1978.

[Dar99] J. Darmont : *Étude des performances de méthodes de regroupement dynamique dans les bases de données orientées-objet*. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, janvier 1999.

[Dar05] J. Darmont : Object Database Benchmarks, volume I-III de *Encyclopedia of Information Science and Technology*, pages 2146–2149. Idea Group Publishing, January 2005.

[Dar08] J. Darmont : Database Benchmarks. *Encyclopedia of Information Science and Technology, Second Edition*. Idea Group Publishing, 2008.

[DB06] J. Darmont et O. Boussaïd, éditeurs. *Processing and Managing Complex Data for Decision Support*. Idea Group Publishing, April 2006.

[DBB02] J. Darmont, O. Boussaïd et F. Bentayeb : Warehousing Web Data. *4th International Conference on Information Integration and Web-based Applications and Services (iiWAS 02), Bandung, Indonesia*, pages 148–152, September 2002.

[DBB+03] J. Darmont, O. Boussaïd, F. Bentayeb, S. Rabaseda et Y. Zellouf : Web multiform data structuring for warehousing, volume 22 de *Multimedia Systems and Applications*, pages 179–194. Kluwer Academic Publishers, 2003.

[DBB04] J. Darmont, F. Bentayeb et O. Boussaïd : Conception d'un banc d'essais décisionnel. *20èmes Journées Bases de Données Avancées (BDA 04), Montpellier*, pages 493–511, Octobre 2004.

[DBB05] J. Darmont, F. Bentayeb et O. Boussaïd : DWEB: A Data Warehouse Engineering Benchmark. *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, volume 3589 de *LNCS*, pages 85–94, August 2005.

[DBB07] J. Darmont, F. Bentayeb et O. Boussaïd : Benchmarking Data Warehouses. *International Journal of Business Intelligence and Data Mining*, 1(1), 2007.

[DBRA05] J. Darmont, O. Boussaïd, J.-C. Ralaivao et K. Aouiche : An architecture framework for complex data warehouses. *7th International Conference on Enterprise Information Systems (ICEIS 05), Miami, USA*, pages 370–373, May 2005.

[DDD+04] B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zaït et M. Ziauddin : Automatic SQL Tuning in Oracle 10g. *30th International Conference on Very Large Data Bases (VLDB 04), Toronto, Canada*, pages 1098–1109, August-September 2004.

[Dem95] M. Demarest : A Data Warehouse Evaluation Model. *Oracle Technical Journal*, 1(1):29, October 1995.

[DFR+00] J. Darmont, C. Fromantin, S. Regnier, L. Gruenwald et M. Schneider : Dynamic clustering in object-oriented databases: An advocacy for simplicity. *ECOOP 2000 Symposium on Objects and Databases, Sophia Antipolis, France*, volume 1944 de *LNCS*, pages 71–85, June 2000.

[Dio05] A. Diouf : *Bancs d'essais pour entrepôts de données XML.* Mémoire de master recherche, Université Lumière Lyon 2, 2005.

[DL99] A.L. Diaz et D. Lovell : XML Generator. *http://www.alphaworks.ibm.com/tech/xmlgenerator*, 1999.

[DO06] J. Darmont et E. Olivier : A complex data warehouse for personalized, anticipative medicine. *17th Information Resources Management Association International Conference (IRMA 06), Washington, USA*, pages 685–687, May 2006.

[DO07] J. Darmont et E. Olivier : Biomedical Data Warehouses. *Encyclopaedia of Healthcare Information Systems.* Idea Group Publishing, 2007.

[DPX04] A. Deutsch, Y. Papakonstantinou et Y. Xu : The NEXT Logical Framework for XQuery. *30th International Conference on Very Large Data Bases (VLDB 04), Toronto, Canada*, pages 168–179, September 2004.

[DS00] J. Darmont et M. Schneider : Benchmarking OODBs with a Generic Tool. *Journal of Database Management*, 11(3):16–27, Jul-Sept 2000.

[dSS99] M.F. de Souza et M.C. Sampaio : Efficient Materialization and Use of Views in Data Warehouses. *SIGMOD Record*, 28(1):78–83, 1999.

[Eck03] W.W. Eckerson : EII – The return of the virtual data warehouse? *Application Development Trends; http://www.adtmag.com/article.aspx?id=8152*, August 2003.

[FCL93] M.J. Franklin, M.J. Carey et M. Livny : Local disk caching for client-server database systems. *19th International Conference on Very Large Data Bases (VLDB 93), Dublin, Ireland*, pages 641–655, 1993.

[FF03] C. Ferris et J.A. Farrell : What are Web services? *Communications of the ACM*, 46(6):31, 2003.

[FON92] M.R. Frank, E. Omiecinski et S.B. Navathe : Adaptive and Automated Index Selection in RDBMS. *3rd International Conference on Extending Database Technology (EDBT 92), Vienna, Austria*, volume 580 de *LNCS*, pages 277–292, 1992.

[FR03] Y.A. Feldman et J. Reouven : A knowledge-based approach for index selection in relational databases. *Expert System with Applications*, 25(1):15–37, 2003.

[GM99] H. Gupta et I.S. Mumick : Selection of Views to Materialize Under a Maintenance Cost Constraint. *7th International Conference on Database Theory (ICDT 99), Jerusalem, Israel*, pages 453–470, 1999.

[GM05a] P. Gançarski et F. Masséglia, éditeurs. *Deuxième atelier sur la fouille de données complexes dans un processus d'extraction des connaissances (EGC 05), Paris*, Janvier 2005.

[GM05b] H. Gupta et I.S. Mumick : Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):24–43, 2005.

[GR98] M. Golfarelli et S. Rizzi : A methodological framework for data warehouse design. *1st ACM International Workshop on Data Warehousing and OLAP (DOLAP 98), New York, USA*, pages 3–9, 1998.

[GR00] M. Golfarelli et S. Rizzi : View materialization for nested GPSJ queries. *2nd International Workshop on Design and Management of Data Warehouses (DMDW 00), Stockholm, Sweden*, volume 28 de *CEUR Workshop Proceedings*, pages 10.1–10.9, June 2000.

[Gra93] Jim Gray : *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, deuxième édition, 1993.

[Gre04] Larry Greenfield : What to Learn About in Order to Speed Up Data Warehouse Querying. *http://www.dwinfocenter.org/fstquery.html*, 2004.

[GRS02] M. Golfarelli, S. Rizzi et E. Saltarelli : Index selection for data warehousing. *4th International Workshop on Design and Management of Data Warehouses (DMDW 02), Toronto, Canada*, pages 33–42, 2002.

[GT04] P. Gançarski et B. Trousse, éditeurs. *Premier atelier sur la fouille de données complexes dans un processus d'extraction des connaissances (EGC 04), Clermont-Ferrand*, Janvier 2004.

[Gun99] T.I. Gundem : Near optimal multiple choice index selection for relational databases. *Computers & Mathematics with Applications*, 37(2):111–120, 1999.

[Gup99] H. Gupta : *Selection and Maintenance of Views in a Data Warehouse*. Thèse de doctorat, Stanford University, 1999.

[HBH03] W. Hümmer, A. Bauer et G. Harde : XCube: XML for data warehouses. *6th International Workshop on Data Warehousing and OLAP (DOLAP 03), New Orleans, USA*, pages 33–40, 2003.

[HBKZ00] Z. He, S. M. Blackburn, L. Kirby et J. Zigman : Platypus: The design and implementation of a flexible high performance object store. *9th International Workshop on Persistent Object Systems (POS9)*, pages 100–124, September 2000.

[HD03a] Z. He et J. Darmont : DOEF: A Dynamic Object Evaluation Framework. *14th International Conference on Database and Expert Systems Applications (DEXA 03), Prague, Czech Republic*, volume 2736 de *LNCS*, pages 662–671, September 2003.

[HD03b] Z. He et J. Darmont : Une plate-forme dynamique pour l'évaluation des performances des bases de données à objets. *19èmes Journées de Bases de Données Avancées (BDA 03), Lyon*, pages 423–442, Octobre 2003.

[HD04] Z. He et J. Darmont : Une plate-forme dynamique pour l'évaluation des performances des bases de données à objets. *Ingénierie des Systèmes d'Information (RSTI série ISI)*, 9(1):109–127, 2004.

[HD05] Z. He et J. Darmont : Evaluating the dynamic behavior of database applications. *Journal of Database Management*, 16(2):21–45, April-June 2005.

[HD06] Z. He et J. Darmont : Evaluating the performance of dynamic database applications, volume 5 de *Advanced Topics in Database Research*, pages 294–319. Idea Group Publishing, Hershey, PA, USA, 2006.

[HMB00] Z. He, A. Marquez et S. Blackburn : Opportunistic prioritised clustering framework (OPCF). *ECOOP 2000 Symposium on Objects and Databases, Sophia Antipolis, France*, volume 1944 de *LNCS*, pages 86–100, June 2000.

[HRU96] V. Harinarayan, A. Rajaraman et J.D. Ullman : Implementing data cubes efficiently. *ACM SIGMOD International Conference on Management of Data (SIGMOD 96), Montreal, Canada*, pages 205–216, 1996.

[Inm02] W.H. Inmon : *Building the Data Warehouse*. John Wiley & Sons, troisième édition, 2002.

[ISR83] M.Y.L. Ip, L.V. Saxton et V.V. Raghavan : On the Selection of an Optimal Set of Indexes. *IEEE Transactions on Software Engineering*, 9(2):135–143, 1983.

[JLS99] H.V. Jagadish, L.V.S. Lakshmanan et D. Srivastava : Snakes and Sandwiches: Optimal Clustering Strategies for a Data Warehouse. *ACM SIGMOD International Conference on Management of Data, Philadelphia, USA, June 1999*, pages 37–48, 1999.

[JMF99] A.K. Jain, M.N. Murty et P.J. Flynn : Data clustering: a Review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JN03] P.E. Jouve et N. Nicoloyannis : KEROUAC: An Algorithm for Clustering Categorical Data Sets with Practical Advantages. *International Workshop on Data Mining for Actionable Knowledge (DMAK 03), Seoul, Korea*, 2003.

[KC04] R. Kimball et J. Caserta : *The Data Warehouse ETL Toolkit*. John Wiley & Sons, 2004.

[KKRSR00] G. Kappel, E. Kapsammer, S. Rausch-Schott et W. Retschitzegger : X-Ray – Towards Integrating XML and Relational Database Systems. *19th International Conference on Conceptual Modeling (ER 00), Salt Lake City, USA*, volume 1920 de *LNCS*, pages 339–353, 2000.

[KLT03] J. Kratica, I. Ljubić et D. Tošić : A Genetic Algorithm for the Index Selection Problem. *Applications of Evolutionary Computing, EvoWorkshops 2003, Essex, UK*, volume 2611 de *LNCS*, pages 281–291, 2003.

[KP04] L. Khan et V.A. Petrushin, éditeurs. *Fifth International Workshop on Multimedia Data Mining (MDM/KDD 04), Seattle, USA*, August 2004.

[KR99] Y. Kotidis et N. Roussopoulos : DynaMat: A Dynamic View Management System for Data Warehouses. *ACM SIGMOD International Conference on Management of Data (SIGMOD 99), Philadelphia, USA*, pages 371–382, 1999.

[KR02] R. Kimball et M. Ross : *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, deuxième édition, 2002.

[LKH05] C.K.S. Leung, Q.I. Khan et T. Hoque : CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns. *5th IEEE International Conference on Data Mining (ICDM 05), Houston, USA*, pages 274–281, November 2005.

[LKK00] S. Lee, S. Kim et W. Kim : The BORD Benchmark for Object-Relational Databases. *11th International Conference on Database and Expert Systems Applications (DEXA 00), London, UK*, volume 1873 de *LNCS*, pages 6–20, September 2000.

[LR98] A. Labrinidis et N. Roussopoulos : *A Performance Evaluation of Online Warehouse Update Algorithms*. Rapport technique CS-TR-3954, Department of Computer Science, University of Maryland, November 1998.

[LRA00] A. Limi, A. Runyan et V. Andersen : OLAP Datasets. *http://cgmlab.cs.dal.ca/downloadarea/datasets/*, 2000.

[LYW+05] H. Lu, J.X. Yu, G. Wang, S. Zheng, H. Jiang, G. Yu et A. Zhou : What makes the differences: benchmarking XML database implementations. *ACM Transactions on Internet Technology*, 5(1):154–194, 2005.

[MAD06a] H. Mahboubi, K. Aouiche et J. Darmont : Materialized View Selection by Query Clustering in XML Data Warehouses. *4th International Multiconference on Computer Science and Information Technology (CSIT 06), Amman, Jordan*, volume 2, pages 68–77, April 2006.

[MAD06b] H. Mahboubi, K. Aouiche et J. Darmont : Un index de jointure pour les entrepôts de données XML. *6èmes Journées Francophones Extraction et Gestion des Connaissances (EGC 06), Lille*, volume E-6 de *RNTI*, pages 89–94, Janvier 2006.

[MAD06c] N. Maiz, K. Aouiche et J. Darmont : Sélection automatique d'index et de vues matérialisées dans les entrepôts de données. *2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 06), Versailles*, volume B-2 de *RNTI*, pages 89–104, Juin 2006.

[MAG+97] J. McHugh, S. Abiteboul, R. Goldman, D. Quass et J. Widom : Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3): 54–66, 1997.

[Mah05] H. Mahboubi : *Optimisation des performances des entrepôts de données XML*. Mémoire de master recherche, INSA Lyon, juin 2005.

[Mai05] N. Maiz : *Sélection automatique d'index et de vues matérialisées dans les entrepôts de données*. Mémoire de master recherche, INSA Lyon, 2005.

[MBR04] R. Ben Messaoud, O. Boussaïd et S. Rabaseda : A New OLAP Aggregation Based on the AHC Technique. *ACM 7th International Workshop on Data Warehousing and OLAP (DOLAP 04), Washington DC, USA*, pages 65–72, November 2004.

[MDB01] S. Miniaoui, J. Darmont et O. Boussaïd : Web data modeling for integration in data warehouses. *First International Workshop on Multimedia Data and Document Engineering (MDDE 01), Lyon, France*, pages 88–97, July 2001.

[Mei02] W. Meier : eXist: An Open Source Native XML Database. *Web, Web-Services, and Database Systems, NODe 2002 Web and Database-Related Workshops, Erfurt, Germany*, volume 2593 de *LNCS*, pages 169–183, 2002.

[MH04] D.L. McGuinness et F. Van Harmelen : *OWL Web Ontology Language Overview*. World Wide Web consortium, February 2004.

[NRDR05] V. Nassis, R. Rajugan, T.S. Dillon et J.W. Rahayu : Conceptual and Systematic Design Approach for XML Document Warehouses. *International Journal of Data Warehousing & Mining*, 1(3):63–86, 2005.

[NT01] T.P. Nadeau et T.J. Teorey : A pareto model for OLAP view size estimation. *4th Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 01), Toronto, Canada*, page 13, 2001.

[NT02] T.P. Nadeau et T.J. Teorey : Achieving Scalability in OLAP Materialized View Selection. *5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 02), McLean, USA*, pages 28–34, 2002.

[Obj03] Object Management Group. *Common Warehouse Metamodel (CWM) Specification version 1.1*, March 2003.

[OG95] P.E. O'Neil et G. Graefe : Multi-table joins through bitmapped join indices. *SIGMOD Record*, 24(3):8–11, 1995.

[OPW+04] M.E. Otey, S. Parthasarathy, C. Wang, A. Veloso et W. Meira Jr. : Parallel and distributed methods for incremental frequent itemset mining. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(6):2439–2450, 2004.

[OQ97] P.E. O'Neil et D. Quass : Improved query performance with variant indexes. *ACM SIGMOD International Conference on Management of Data (SIGMOD 97), Tucson, USA*, pages 38–49, 1997.

[PAKJ+02] S. Paparizos, S. Al-Khalifa, H.V. Jagadish, L.V.S. Lakshmanan, A. Nierman, D. Srivastava et Y. Wu : Grouping in XML. *XML-Based Data Management and Multimedia Engineering (EDBT/XMLDM 02), Prague, Czech Republic*, volume 2490 de *LNCS*, pages 128–147, March 2002.

[Pas00] N. Pasquier : *Data Mining : Algorithmes d'Extraction et de Réduction des Règles d'Association dans les Bases de Données*. Thèse de doctorat, Université Blaise Pascal Clermont-Ferrand II, 2000.

[PBTL99] N. Pasquier, Y. Bastide, R. Taouil et L. Lakhal : Discovering frequent closed itemsets for association rules. *7th International Conference on Database Theory (ICDT 99), Jerusalem, Israel*, volume 1540 de *LNCS*, pages 398–416, 1999.

[PCTM03] John Poole, Dan Chang, Douglas Tolbert et David Mellor : *Common Warehouse Metamodel Developer's Guide*. John Wiley & Sons, 2003.

[PF00] M. Poess et C. Floyd : New TPC Benchmarks for Decision Support and Web Commerce. *SIGMOD Record*, 29(4):64–71, December 2000.

[PHS05] B.K. Park, H. Han et I.Y. Song : XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. *7th International Conference on Data Warehousing and Knowledge Discovery, (DaWaK 05), Copenhagen, Denmark*, pages 32–42, 2005.

[Pok01] J. Pokorný : Modelling Stars Using XML. *4th ACM International Workshop on Data Warehousing and OLAP (DOLAP 01), Atlanta, USA*, pages 24–31, 2001.

[Pok02] J. Pokorný : XML Data Warehouse: Modelling and Querying. *5th International Baltic Conference (BalticDB&IS 02), Tallin, Estonia*, pages 267–280, 2002.

[PSKL02] M. Poess, B. Smith, L. Kollar et P.A. Larson : TPC-DS: Taking Decision Support Benchmarking to the Next Level. *2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA*, pages 582–587, June 2002.

[RCD05] R. Rajugan, E. Chang et T.S. Dillon : Conceptual Design of an XML FACT Repository for Dispersed XML Document Warehouses and XML Marts. *20th International Conference on Computer and Information Technology (CIT 05), Shanghai, China*, pages 141–149, 2005.

[RPJAK02] K. Runapongsa, J.M. Patel, H.V. Jagadish et S. Al-Khalifa : The Michigan Benchmark: A Microbenchmark for XML Query Processing Systems. *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT, Hong Kong, China*, volume 2590 de *LNCS*, pages 160–161, 2002.

[RRT04] L.I. Rusu, J.W. Rahayu et D. Taniar : On Building XML Data Warehouses. *5th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 04), Exeter, UK*, pages 293–299, 2004.

[RRT05] L.I. Rusu, J.W. Rahayu et D. Taniar : A Methodology for Building XML Data Warehouse. *International Journal of Data Warehousing and Mining*, 1(2):67–92, 2005.

[RS03] S. Rizzi et E. Saltarelli : View Materialization vs. Indexing: Balancing Space Constraints in Data Warehouse Design. *15th International Conference on Advanced Information Systems Engineering (CAiSE 03), Klagenfurt, Austria*, pages 502–519, 2003.

[SCP03a] M. Serrano, C. Calero et M. Piattini : Experimental Validation of Multidimensional Data Models Metrics. *36th Hawaii International Conference on System Sciences (HICSS 03), Big Island, USA*, page 327, January 2003.

[SCP03b] M. Serrano, C. Calero et M. Piattini : Metrics for Data Warehouse Quality. *Effective Databases for Text & Document Management*, pages 156–173. IRM Press, 2003.

[SCT+04a] M. Serrano, C. Calero, J. Trujillo, S. Luján-Mora et M. Piattini : Empirical Validation of Metrics for Conceptual Models of Data Warehouses. *16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia*, volume 3084 de *LNCS*, pages 506–520, 2004.

[SCT+04b] M. Serrano, C. Calero, J. Trujillo, S. Luján-Mora et M. Piattini : Towards a Metrics Suite for Conceptual Models of Datawarehouses. *1st International Workshop on Software Audit and Metrics (SAM 04), Porto, Portugal*, pages 105–117, 2004.

[SDN00] A. Shukla, P. Deshpande et J.F. Naughton : Materialized View Selection for Multi-Cube Data Models. *7th International Conference on Extending DataBase Technology (EDBT 00), Konstanz, Germany*, pages 269–284, 2000.

[SDNR96] A. Shukla, P.M. Deshpande, J.F. Naughton et K. Ramasamy : Storage estimation for multidimensional aggregates in the presence of hierarchies. *22nd International Conference on Very Large Data Bases (VLDB 96), Bombay, India*, pages 522–531, 1996.

[SDRK02] Y. Sismanis, A. Deligiannakis, N. Roussopoulos et Y. Kotidis : Dwarf: shrinking the PetaCube. *2002 ACM SIGMOD International Conference on Management of Data, Madison, USA*, pages 464–475, June 2002.

[SLJ04] J.R. Smith, C.S. Li et A. Jhingran : A Wavelet Framework for Adapting Data Cube Views for OLAP. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):552–565, 2004.

[Sou05] Sourceforge : The Open Source Database Benchmark version 0.19. *http://osdb.sourceforge.net/*, 2005.

[SRR06] B. Shah, K. Ramachandran et V. Raghavan : A Hybrid Approach for Data Warehouse View Selection. *International Journal of Data Warehousing and Mining*, 2(2):1–37, April-June 2006.

[SWK+02] A. Schmidt, F. Waas, M. Kersten, M.J. Carey, I. Manolescu et R. Busse : XMark: A Benchmark for XML Data Management. *28th International Conference on Very Large Databases (VLDB 02), Hong Kong, China*, pages 974–985, 2002.

[Tho98] E. Thomsen : Comparing different approaches to OLAP calculations as revealed in benchmarks. *Intelligence Enterprise's Database Programming & Design; http://www.dbpd.com/vault/9805desc.htm*, May 1998.

[Tho04] E. Thomsen : *OLAP Solutions: Building Multidimensional Information Systems*. John Wiley & Sons, deuxième édition, 2004.

[TNL95] A. Tiwary, V. Narasayya et H. Levy : Evaluation of OO7 as a system and an application benchmark. *OOPSLA '95 Workshop on Object Database Behavior, Benchmarks and Performance, Austin, USA*, 1995.

[TOB93] C. Turbyfill, C. Orji et D. Bitton : AS^{3}AP – An ANSI SQL Standard Scalable and Portable Benchmark for Relational Database Systems. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, 1993.

[TP02] A. Tsalgatidou et T. Pilioura : An Overview of Standards and Related Technology in Web Services. *Distributed and Parallel Databases*, 12(2-3):135–162, 2002.

[TPC98] TPC : *TPC Benchmark D Standard Specification version 2.1*. Transaction Processing Performance Council, February 1998.

[TPC02] TPC : *TPC Benchmark W (Web Commerce) Specification version 1.8*. Transaction Processing Performance Council, February 2002.

[TPC03] TPC : *TPC Benchmark R Standard Specification version 2.2.0*. Transaction Processing Performance Council, August 2003.

[TPC05a] TPC : *TPC Benchmark App (Application Server) Specification version 1.1.1*. Transaction Processing Performance Council, August 2005.

[TPC05b] TPC : *TPC Benchmark C Standard Specification revision 5.6*. Transaction Processing Performance Council, December 2005.

[TPC05c] TPC : *TPC Benchmark DS (Decision Support) Draft Specification revision 32*. Transaction Processing Performance Council, December 2005.

[TPC05d] TPC : *TPC Benchmark H Standard Specification revision 2.3.0*. Transaction Processing Performance Council, August 2005.

[TX04] D. Theodoratos et W. Xu : Constructing search spaces for materialized view selection. *7th ACM International Workshop on Data Warehousing and OLAP (DOLAP 04), Washington DC, USA*, pages 112–121, November 2004.

[UBDB04] C. Udréa, F. Bentayeb, J. Darmont et O. Boussaïd : Intégration efficace de méthodes de fouille de données dans les SGBD. *4èmes Journées Francophones d'Extraction et de Gestion des Connaissances (EGC 04), Clermont-Ferrand*, volume 2 de *RNTI*, pages 83–94, Janvier 2004.

[URT99] H. Uchiyama, K. Runapongsa et T.J. Teorey : A Progressive View Materialization Algorithm. *2nd ACM International Workshop on Data warehousing and OLAP (DOLAP 99), Kansas City, USA*, pages 36–41, 1999.

[VBR03] B. Vrdoljak, M. Banek et S. Rizzi : Designing Web Warehouses from XML Schemas. *5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 03), Prague, Czech Republic*, pages 89–98, 2003.

[VMGM02] P. Valtchev, R. Missaoui, R. Godin et M. Meridji : Generating Frequent Itemsets Incrementally: Two Novel Approaches Based on Galois Lattice Theory. *Journal of Experimental & Theoretical Artificial Intelligence*, 2-3(4):115–142, 2002.

[VVK02] S.R. Valluri, S. Vadapalli et K. Karlapalem : View Relevance Driven Materialized View Selection in Data Warehousing Environment. *13th Australasian Database Technologies Conference (ADC 2002), Melbourne, Australia*, pages 187–196, 2002.

[VZZ+00] G. Valentin, M. Zuliani, D. Zilio, G. Lohman et A. Skelley : DB2 advisor: An optimizer smart enough to recommend its own indexes. *16th International Conference on Data Engineering (ICDE 00), San Diego, USA*, pages 101–110, 2000.

[Wal05] D. Waldt : Using XML and Databases: W3C Standards in Practice. White Paper, *The Gilbane Report; http://gilbane.com/whitepapers.pl?view=15*, January 2005.

[WB98] M.C. Wu et A.P. Buchmann : Encoded bitmap indexing for data warehouses. *14th International Conference on Data Engineering (ICDE 98), Orlando, USA*, pages 220–230, 1998.

[Wha85] K.Y. Whang : Index selection in relational databases. *International Conference on Foundations of Data Organization (FODO 85), Kyoto, Japan*, pages 487–500, May 1985.

[WMHZ02] G. Weikum, A. Monkeberg, C. Hasse et P. Zabback : 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*, pages 20–31, 2002.

[WMT05a] P. Wehrle, M. Miquel et A. Tchounikine : A Model for Distributing and Querying a Data Warehouse on a Computing Grid. *11th International Conference on Parallel and Distributed Systems (ICPADS 05), Fuduoka, Japan*, pages 203–209, July 2005.

[WMT05b] P. Wehrle, M. Miquel et A. Tchounikine : Entrepôts de données sur grilles de calcul. *1er atelier Extraction et Gestion Parallèles Distribuées de Connaissances, EGC 05, Paris*, volume E-3 de *RNTI*, Janvier 2005.

[Yao77] S.B. Yao : Approximating block accesses in database organizations. *Communication of the ACM*, 20(4):260–261, 1977.

[YHA05] Q. Yao, J. Huang et A. An : Machine Learning Approach to Identify Database Sessions Using Unlabeled Data. *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, volume 3589 de *LNCS*, pages 254–255, August 2005.

[YOK04] B.B. Yao, T. Ozsu et N. Khandelwal : XBench Benchmark and Performance Testing of XML DBMSs. *20th International Conference on Data Engineering (ICDE 04), Boston, USA*, pages 621–633, 2004.

[ZRL+04] D.C. Zilio, J. Rao, S. Lightstone, G.M. Lohman, A. Storm, C. Garcia-Arellano et S. Fadden : DB2 Design Advisor: Integrated Automatic Physical Database Design. *30th International Conference on Very Large Data Bases (VLDB 04), Toronto, Canada*, pages 1087–1097, August-September 2004.

[ZSD03] O.R. Zaiane, S. Simoff et C. Djeraba, éditeurs. *Mining Multimedia and Complex Data*, volume 2797 de *LNAI*. Springer, 2003.

[ZSG04] M. Zaman, J. Surabattula et L. Gruenwald : An Auto-Indexing Technique for Databases Based on Clustering. *15th International Workshop on Database and Expert Systems Applications (DEXA Workshops 04), Zaragoza, Spain*, pages 776–780, 2004.

[ZTR05] D.A. Zighed, S. Tsumoto et Z.W. Ras, éditeurs. *First International Workshop on Mining Complex Data (MCD 05), New Orleans, USA*, November 2005.

[ZWLZ05] J. Zhang, W. Wang, H. Liu et S. Zhang : X-warehouse: building query pattern-driven data. *14th international conference on World Wide Web (WWW 05), Chiba, Japan*, pages 896–897, May 2005.

[ZYY01] C. Zhang, X. Yao et J. Yang : An Evolutionary Approach to Materialized View Selection in a Data Warehouse Environment. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(3):282–294, 2001.